```
*******************************************************
   43526 Sun Oct 28 20:56:06 2012
new/gcc/common.opt
Implement -fstrict-calling-conventions
Stock GCC is overly willing to violate the ABI when calling local functions,
such that it passes arguments in registers on i386.  This hampers debugging
with anything other than a fully-aware DWARF debugger, and is generally not
something we desire.
Implement a flag which disables this behaviour, enabled by default.  The flag is
global, though only effective on i386, to more easily allow its globalization
later which, given the odds, is likely to be necessary.
*******************************************************
```

```
  1 ; Options for the language- and target-independent parts of the compiler.

  3 ; Copyright (C) 2003, 2004, 2005, 2006, 2007, 2008, 2009
  4 ; Free Software Foundation, Inc.
  5 ;
  6 ; This file is part of GCC.
  7 ;
  8 ; GCC is free software; you can redistribute it and/or modify it under
  9 ; the terms of the GNU General Public License as published by the Free
 10 ; Software Foundation; either version 3, or (at your option) any later
 11 ; version.
 12 ;
 13 ; GCC is distributed in the hope that it will be useful, but WITHOUT ANY
 14 ; WARRANTY; without even the implied warranty of MERCHANTABILITY or
 15 ; FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 16 ; for more details.
 17 ;
 18 ; You should have received a copy of the GNU General Public License
 19 ; along with GCC; see the file COPYING3.  If not see
 20 ; <http://www.gnu.org/licenses/>.

 22 ; See the GCC internals manual (options.texi) for a description of this file's f

 24 ; Please try to keep this file in ASCII collating order.

 26 -help
 27 Common
 28 Display this information

 30 -help=
 31 Common Report Joined
 32 --help=<class>  Display descriptions of a specific class of options.  <class> is

 34 -target-help
 35 Common
 36 Alias for --help=target

 38 ;; The following three entries are to work around the gcc driver
 39 ;; program's insatiable desire to turn options starting with a
 40 ;; double dash (--) into options starting with a dash f (-f).
 41 fhelp
 42 Common

 44 fhelp=
 45 Common Joined

 47 ftarget-help
 48 Common

 50 -param
 51 Common Separate
 52 --param <param>=<value> Set parameter <param> to value.  See below for a complet

 54 -version
```



```
 55 Common

 57 G
 58 Common Joined Separate UInteger
 59 -G<number>      Put global and static data smaller than <number> bytes into a sp

 61 O
 62 Common JoinedOrMissing Optimization
 63 -O<number>      Set optimization level to <number>

 65 Os
 66 Common Optimization
 67 Optimize for space rather than speed

 69 W
 70 Common RejectNegative
 71 This switch is deprecated; use -Wextra instead

 73 Waggregate-return
 74 Common Var(warn_aggregate_return) Warning
 75 Warn about returning structures, unions or arrays

 77 Warray-bounds
 78 Common Var(warn_array_bounds) Warning
 79 Warn if an array is accessed out of bounds

 81 Wattributes
 82 Common Var(warn_attributes) Init(1) Warning
 83 Warn about inappropriate attribute usage

 85 Wcast-align
 86 Common Var(warn_cast_align) Warning
 87 Warn about pointer casts which increase alignment

 89 Wdeprecated-declarations
 90 Common Var(warn_deprecated_decl) Init(1) Warning
 91 Warn about uses of __attribute__((deprecated)) declarations

 93 Wdisabled-optimization
 94 Common Var(warn_disabled_optimization) Warning
 95 Warn when an optimization pass is disabled

 97 Werror
 98 Common Var(warnings_are_errors)
 99 Treat all warnings as errors

101 Werror=
102 Common Joined
103 Treat specified warning as error

105 Wextra
106 Common Warning
107 Print extra (possibly unwanted) warnings

109 Wfatal-errors
110 Common Var(flag_fatal_errors)
111 Exit on the first error occurred

113 Wframe-larger-than=
114 Common RejectNegative Joined UInteger
115 -Wframe-larger-than=<number> Warn if a function's stack frame requires more than

117 Winline
118 Common Var(warn_inline) Warning
119 Warn when an inlined function cannot be inlined
```

```
121 Wlarger-than-
122 Common RejectNegative Joined UInteger Warning

124 Wlarger-than=
125 Common RejectNegative Joined UInteger Warning
126 -Wlarger-than=<number>  Warn if an object is larger than <number> bytes

128 Wlogical-op
129 Common Warning Var(warn_logical_op)
130 Warn when a logical operator is suspicously always evaluating to true or false

132 Wunsafe-loop-optimizations
133 Common Var(warn_unsafe_loop_optimizations) Warning
134 Warn if the loop cannot be optimized due to nontrivial assumptions.

136 Wmissing-noreturn
137 Common Var(warn_missing_noreturn) Warning
138 Warn about functions which might be candidates for __attribute__((noreturn))

140 Wmudflap
141 Common Var(warn_mudflap) Init(1) Warning
142 Warn about constructs not instrumented by -fmudflap

144 Woverflow
145 Common Var(warn_overflow) Init(1) Warning
146 Warn about overflow in arithmetic expressions

148 Wpacked
149 Common Var(warn_packed) Warning
150 Warn when the packed attribute has no effect on struct layout

152 Wpadded
153 Common Var(warn_padded) Warning
154 Warn when padding is required to align structure members

156 Wshadow
157 Common Var(warn_shadow) Warning
158 Warn when one local variable shadows another

160 Wstack-protector
161 Common Var(warn_stack_protect) Warning
162 Warn when not issuing stack smashing protection for some reason

164 Wstrict-aliasing
165 Common Warning
166 Warn about code which might break strict aliasing rules

168 Wstrict-aliasing=
169 Common Joined UInteger Var(warn_strict_aliasing) Init(-1) Warning
170 Warn about code which might break strict aliasing rules

172 Wstrict-overflow
173 Common Warning
174 Warn about optimizations that assume that signed overflow is undefined

176 Wstrict-overflow=
177 Common Joined UInteger Var(warn_strict_overflow) Init(-1) Warning
178 Warn about optimizations that assume that signed overflow is undefined

180 Wswitch
181 Common Var(warn_switch) Warning
182 Warn about enumerated switches, with no default, missing a case

184 Wswitch-default
185 Common Var(warn_switch_default) Warning
186 Warn about enumerated switches missing a \"default:\" statement
```



```
188 Wswitch-enum
189 Common Var(warn_switch_enum) Warning
190 Warn about all enumerated switches missing a specific case

192 Wsystem-headers
193 Common Var(warn_system_headers) Warning
194 Do not suppress warnings from system headers

196 Wtype-limits
197 Common Var(warn_type_limits) Init(-1) Warning
198 Warn if a comparison is always true or always false due to the limited range of

200 Wuninitialized
201 Common Var(warn_uninitialized) Warning
202 Warn about uninitialized automatic variables

204 Wunreachable-code
205 Common Var(warn_notreached) Warning
206 Warn about code that will never be executed

208 Wunused
209 Common Var(warn_unused) Init(0) Warning
210 Enable all -Wunused- warnings

212 Wunused-function
213 Common Var(warn_unused_function) Init(-1) Warning
214 Warn when a function is unused

216 Wunused-label
217 Common Var(warn_unused_label) Init(-1) Warning
218 Warn when a label is unused

220 Wunused-parameter
221 Common Var(warn_unused_parameter) Init(-1) Warning
222 Warn when a function parameter is unused

224 Wunused-value
225 Common Var(warn_unused_value) Init(-1) Warning
226 Warn when an expression value is unused

228 Wunused-variable
229 Common Var(warn_unused_variable) Init(-1) Warning
230 Warn when a variable is unused

232 Wcoverage-mismatch
233 Common RejectNegative Var(warn_coverage_mismatch) Warning
234 Warn instead of error in case profiles in -fprofile-use do not match

236 aux-info
237 Common Separate
238 -aux-info <file>        Emit declaration information into <file>

240 aux-info=
241 Common Joined

243 auxbase
244 Common Separate

246 auxbase-strip
247 Common Separate

249 d
250 Common Joined
251 -d<letters>     Enable dumps from specific passes of the compiler
```

```
253 dumpbase
254 Common Separate
255 -dumpbase <file>        Set the file basename to be used for dumps

257 ; The version of the C++ ABI in use.  The following values are allowed:
258 ;
259 ; 0: The version of the ABI believed most conformant with the C++ ABI
260 ;    specification.  This ABI may change as bugs are discovered and fixed.
261 ;    Therefore, 0 will not necessarily indicate the same ABI in different
262 ;    versions of G++.
263 ;
264 ; 1: The version of the ABI first used in G++ 3.2.
265 ;
266 ; 2: The version of the ABI first used in G++ 3.4.
267 ;
268 ; Additional positive integers will be assigned as new versions of
269 ; the ABI become the default version of the ABI.
270 fabi-version=
271 Common Joined UInteger Var(flag_abi_version) Init(2)

273 falign-functions
274 Common Report Var(align_functions,0) Optimization UInteger
275 Align the start of functions

277 falign-functions=
278 Common RejectNegative Joined UInteger

280 falign-jumps
281 Common Report Var(align_jumps,0) Optimization UInteger
282 Align labels which are only reached by jumping

284 falign-jumps=
285 Common RejectNegative Joined UInteger

287 falign-labels
288 Common Report Var(align_labels,0) Optimization UInteger
289 Align all labels

291 falign-labels=
292 Common RejectNegative Joined UInteger

294 falign-loops
295 Common Report Var(align_loops) Optimization UInteger
296 Align the start of loops

298 falign-loops=
299 Common RejectNegative Joined UInteger

301 ; This flag is only tested if alias checking is enabled.
302 ; 0 if pointer arguments may alias each other.  True in C.
303 ; 1 if pointer arguments may not alias each other but may alias
304 ;   global variables.
305 ; 2 if pointer arguments may not alias each other and may not
306 ;   alias global variables.
307 ; 3 if pointer arguments may not alias anything.  True in Fortran.
308 ;   Set by the front end.
309 fargument-alias
310 Common Report Var(flag_argument_noalias,0) Optimization
311 Specify that arguments may alias each other and globals

313 fargument-noalias
314 Common Report Var(flag_argument_noalias,1) VarExists Optimization
315 Assume arguments may alias globals but not each other

317 fargument-noalias-global
318 Common Report Var(flag_argument_noalias,2) VarExists Optimization
```



```
319 Assume arguments alias neither each other nor globals

321 fargument-noalias-anything
322 Common Report Var(flag_argument_noalias,3) VarExists Optimization
323 Assume arguments alias no other storage

325 fasynchronous-unwind-tables
326 Common Report Var(flag_asynchronous_unwind_tables) Optimization
327 Generate unwind tables that are exact at each instruction boundary

329 fauto-inc-dec
330 Common Report Var(flag_auto_inc_dec) Init(1)
331 Generate auto-inc/dec instructions

333 ; -fcheck-bounds causes gcc to generate array bounds checks.
334 ; For C, C++ and ObjC: defaults off.
335 ; For Java: defaults to on.
336 ; For Fortran: defaults to off.
337 fbounds-check
338 Common Report Var(flag_bounds_check)
339 Generate code to check bounds before indexing arrays

341 fbranch-count-reg
342 Common Report Var(flag_branch_on_count_reg) Init(1) Optimization
343 Replace add, compare, branch with branch on count register

345 fbranch-probabilities
346 Common Report Var(flag_branch_probabilities) Optimization
347 Use profiling information for branch probabilities

349 fbranch-target-load-optimize
350 Common Report Var(flag_branch_target_load_optimize) Optimization
351 Perform branch target load optimization before prologue / epilogue threading

353 fbranch-target-load-optimize2
354 Common Report Var(flag_branch_target_load_optimize2) Optimization
355 Perform branch target load optimization after prologue / epilogue threading

357 fbtr-bb-exclusive
358 Common Report Var(flag_btr_bb_exclusive) Optimization
359 Restrict target load migration not to re-use registers in any basic block

361 fcall-saved-
362 Common Joined RejectNegative
363 -fcall-saved-<register> Mark <register> as being preserved across functions

365 fcall-used-
366 Common Joined RejectNegative
367 -fcall-used-<register>  Mark <register> as being corrupted by function calls

369 ; Nonzero for -fcaller-saves: allocate values in regs that need to
370 ; be saved across function calls, if that produces overall better code.
371 ; Optional now, so people can test it.
372 fcaller-saves
373 Common Report Var(flag_caller_saves) Optimization
374 Save registers around function calls

376 fcheck-data-deps
377 Common Report Var(flag_check_data_deps)
378 Compare the results of several data dependence analyzers.

380 fcommon
381 Common Report Var(flag_no_common,0) Optimization
382 Do not put uninitialized globals in the common section

384 fconserve-stack
```

```
385 Common Var(flag_conserve_stack) Optimization
386 Do not perform optimizations increasing noticeably stack usage

388 fcprop-registers
389 Common Report Var(flag_cprop_registers) Optimization
390 Perform a register copy-propagation optimization pass

392 fcrossjumping
393 Common Report Var(flag_crossjumping) Optimization
394 Perform cross-jumping optimization

396 fcse-follow-jumps
397 Common Report Var(flag_cse_follow_jumps) Optimization
398 When running CSE, follow jumps to their targets

400 fcse-skip-blocks
401 Common Report Var(flag_cse_skip_blocks) Optimization
402 When running CSE, follow conditional jumps

404 fcx-limited-range
405 Common Report Var(flag_cx_limited_range) Optimization
406 Omit range reduction step when performing complex division

408 fcx-fortran-rules
409 Common Report Var(flag_cx_fortran_rules) Optimization
410 Complex multiplication and division follow Fortran rules

412 fdata-sections
413 Common Report Var(flag_data_sections) Optimization
414 Place data items into their own section

416 fdbg-cnt-list
417 Common Report
418 List all available debugging counters with their limits and counts.

420 fdbg-cnt=
421 Common RejectNegative Joined
422 -fdbg-cnt=<counter>:<limit>[,<counter>:<limit>,...]    Set the debug counter lim

424 fdebug-prefix-map=
425 Common Joined RejectNegative
426 Map one directory name to another in debug information

428 ; Nonzero for -fdefer-pop: don't pop args after each function call
429 ; instead save them up to pop many calls' args with one insns.
430 fdefer-pop
431 Common Report Var(flag_defer_pop) Optimization
432 Defer popping functions args from stack until later

434 fdelayed-branch
435 Common Report Var(flag_delayed_branch) Optimization
436 Attempt to fill delay slots of branch instructions

438 fdelete-null-pointer-checks
439 Common Report Var(flag_delete_null_pointer_checks) Optimization
440 Delete useless null pointer checks

442 fdiagnostics-show-location=
443 Common Joined RejectNegative
444 -fdiagnostics-show-location=[once|every-line]   How often to emit source locatio

446 fdiagnostics-show-option
447 Common
448 Amend appropriate diagnostic messages with the command line option that controls

450 fdump-
```



```
451 Common Joined RejectNegative
452 -fdump-<type>   Dump various compiler internals to a file

454 fdump-noaddr
455 Common Report Var(flag_dump_noaddr)
456 Suppress output of addresses in debugging dumps

458 fdump-unnumbered
459 Common Report Var(flag_dump_unnumbered) VarExists
460 Suppress output of instruction numbers, line number notes and addresses in debug

462 fdwarf2-cfi-asm
463 Common Report Var(flag_dwarf2_cfi_asm) Init(HAVE_GAS_CFI_DIRECTIVE)
464 Enable CFI tables via GAS assembler directives.

466 fearly-inlining
467 Common Report Var(flag_early_inlining) Init(1) Optimization
468 Perform early inlining

470 feliminate-dwarf2-dups
471 Common Report Var(flag_eliminate_dwarf2_dups)
472 Perform DWARF2 duplicate elimination

474 feliminate-unused-debug-symbols
475 Common Report Var(flag_debug_only_used_symbols)
476 Perform unused type elimination in debug info

478 feliminate-unused-debug-types
479 Common Report Var(flag_eliminate_unused_debug_types) Init(1)
480 Perform unused type elimination in debug info

482 femit-class-debug-always
483 Common Report Var(flag_emit_class_debug_always) Init(0)
484 Do not suppress C++ class debug information.

486 fexceptions
487 Common Report Var(flag_exceptions) Optimization
488 Enable exception handling

490 fexpensive-optimizations
491 Common Report Var(flag_expensive_optimizations) Optimization
492 Perform a number of minor, expensive optimizations

494 ffast-math
495 Common

497 ffinite-math-only
498 Common Report Var(flag_finite_math_only) Optimization
499 Assume no NaNs or infinities are generated

501 ffixed-
502 Common Joined RejectNegative
503 -ffixed-<register>      Mark <register> as being unavailable to the compiler

505 ffloat-store
506 Common Report Var(flag_float_store) Optimization
507 Don't allocate floats and doubles in extended-precision registers

509 fforce-addr
510 Common
511 Does nothing.  Preserved for backward compatibility.

513 fforward-propagate
514 Common Report Var(flag_forward_propagate) Optimization
515 Perform a forward propagation pass on RTL
```

```
517 ; Nonzero means don't put addresses of constant functions in registers.
518 ; Used for compiling the Unix kernel, where strange substitutions are
519 ; done on the assembly output.
520 ffunction-cse
521 Common Report Var(flag_no_function_cse,0)
522 Allow function addresses to be held in registers

524 ffunction-sections
525 Common Report Var(flag_function_sections)
526 Place each function into its own section

528 fgcse
529 Common Report Var(flag_gcse) Optimization
530 Perform global common subexpression elimination

532 fgcse-lm
533 Common Report Var(flag_gcse_lm) Init(1) Optimization
534 Perform enhanced load motion during global common subexpression elimination

536 fgcse-sm
537 Common Report Var(flag_gcse_sm) Init(0) Optimization
538 Perform store motion after global common subexpression elimination

540 fgcse-las
541 Common Report Var(flag_gcse_las) Init(0) Optimization
542 Perform redundant load after store elimination in global common subexpression
543 elimination

545 fgcse-after-reload
546 Common Report Var(flag_gcse_after_reload) Optimization
547 Perform global common subexpression elimination after register allocation
548 has finished

550 ; This option is not documented yet as its semantics will change.
551 fgraphite
552 Common Report Var(flag_graphite)
553 Enable in and out of Graphite representation

555 floop-strip-mine
556 Common Report Var(flag_loop_strip_mine) Optimization
557 Enable Loop Strip Mining transformation

559 floop-interchange
560 Common Report Var(flag_loop_interchange) Optimization
561 Enable Loop Interchange transformation

563 floop-block
564 Common Report Var(flag_loop_block) Optimization
565 Enable Loop Blocking transformation

567 ; This option is not documented as it does not perform any useful optimization.
568 fgraphite-identity
569 Common Report Var(flag_graphite_identity) Optimization
570 Enable Graphite Identity transformation

572 fguess-branch-probability
573 Common Report Var(flag_guess_branch_prob) Optimization
574 Enable guessing of branch probabilities

576 ; Nonzero means ignore '#ident' directives.  0 means handle them.
577 ; Generate position-independent code for executables if possible
578 ; On SVR4 targets, it also controls whether or not to emit a
579 ; string identifying the compiler.
580 fident
581 Common Report Var(flag_no_ident,0)
582 Process #ident directives
```



```
584 fif-conversion
585 Common Report Var(flag_if_conversion) Optimization
586 Perform conversion of conditional jumps to branchless equivalents

588 fif-conversion2
589 Common Report Var(flag_if_conversion2) Optimization
590 Perform conversion of conditional jumps to conditional execution

592 ; -finhibit-size-directive inhibits output of .size for ELF.
593 ; This is used only for compiling crtstuff.c,
594 ; and it may be extended to other effects
595 ; needed for crtstuff.c on other systems.
596 finhibit-size-directive
597 Common Report Var(flag_inhibit_size_directive)
598 Do not generate .size directives

600 findirect-inlining
601 Common Report Var(flag_indirect_inlining)
602 Perform indirect inlining

604 ; Nonzero means that functions declared 'inline' will be treated
605 ; as 'static'.  Prevents generation of zillions of copies of unused
606 ; static inline functions; instead, 'inlines' are written out
607 ; only when actually used.  Used in conjunction with -g.  Also
608 ; does the right thing with #pragma interface.
609 finline
610 Common Report Var(flag_no_inline,0) Init(0)
611 Pay attention to the \"inline\" keyword

613 finline-small-functions
614 Common Report Var(flag_inline_small_functions) Optimization
615 Integrate simple functions into their callers when code size is known to not gro

617 finline-functions
618 Common Report Var(flag_inline_functions) Optimization
619 Integrate simple functions into their callers

621 finline-functions-called-once
622 Common Report Var(flag_inline_functions_called_once) Init(1) Optimization
623 Integrate functions called once into their callers

625 finline-limit-
626 Common RejectNegative Joined UInteger

628 finline-limit=
629 Common RejectNegative Joined UInteger
630 -finline-limit=<number> Limit the size of inlined functions to <number>

632 finstrument-functions
633 Common Report Var(flag_instrument_function_entry_exit)
634 Instrument function entry and exit with profiling calls

636 finstrument-functions-exclude-function-list=
637 Common RejectNegative Joined
638 -finstrument-functions-exclude-function-list=name,...  Do not instrument listed

640 finstrument-functions-exclude-file-list=
641 Common RejectNegative Joined
642 -finstrument-functions-exclude-file-list=filename,...  Do not instrument functio

644 fipa-cp
645 Common Report Var(flag_ipa_cp) Optimization
646 Perform Interprocedural constant propagation

648 fipa-cp-clone
```

```
649 Common Report Var(flag_ipa_cp_clone) Optimization
650 Perform cloning to make Interprocedural constant propagation stronger

652 fipa-pure-const
653 Common Report Var(flag_ipa_pure_const) Init(0) Optimization
654 Discover pure and const functions

656 fipa-pta
657 Common Report Var(flag_ipa_pta) Init(0) Optimization
658 Perform interprocedural points-to analysis

660 fipa-reference
661 Common Report Var(flag_ipa_reference) Init(0) Optimization
662 Discover readonly and non addressable static variables

664 fipa-type-escape
665 Common Report Var(flag_ipa_type_escape) Init(0) Optimization
666 Type based escape and alias analysis

668 fipa-matrix-reorg
669 Common Report Var(flag_ipa_matrix_reorg) Optimization
670 Perform matrix layout flattening and transposing based
671 on profiling information.

673 fipa-struct-reorg
674 Common Report Var(flag_ipa_struct_reorg)
675 Perform structure layout optimizations based
676 on profiling information.

678 fira-algorithm=
679 Common Joined RejectNegative
680 -fira-algorithm=[CB|priority] Set the used IRA algorithm

682 fira-region=
683 Common Joined RejectNegative
684 -fira-region=[one|all|mixed] Set regions for IRA

686 fira-coalesce
687 Common Report Var(flag_ira_coalesce) Init(0)
688 Do optimistic coalescing.

690 fira-share-save-slots
691 Common Report Var(flag_ira_share_save_slots) Init(1)
692 Share slots for saving different hard registers.

694 fira-share-spill-slots
695 Common Report Var(flag_ira_share_spill_slots) Init(1)
696 Share stack slots for spilled pseudo-registers.

698 fira-verbose=
699 Common RejectNegative Joined UInteger
700 -fira-verbose=<number> Control IRA's level of diagnostic messages.

702 fivopts
703 Common Report Var(flag_ivopts) Init(1) Optimization
704 Optimize induction variables on trees

706 fjump-tables
707 Common Var(flag_jump_tables) Init(1) Optimization
708 Use jump tables for sufficiently large switch statements

710 fkeep-inline-functions
711 Common Report Var(flag_keep_inline_functions)
712 Generate code for functions even if they are fully inlined

714 fkeep-static-consts
```



```
715 Common Report Var(flag_keep_static_consts) Init(1)
716 Emit static const variables even if they are not used

718 fleading-underscore
719 Common Report Var(flag_leading_underscore) Init(-1)
720 Give external symbols a leading underscore

722 floop-optimize
723 Common
724 Does nothing.  Preserved for backward compatibility.

726 fmath-errno
727 Common Report Var(flag_errno_math) Init(1) Optimization
728 Set errno after built-in math functions

730 fmem-report
731 Common Report Var(mem_report)
732 Report on permanent memory allocation

734 ; This will attempt to merge constant section constants, if 1 only
735 ; string constants and constants from constant pool, if 2 also constant
736 ; variables.
737 fmerge-all-constants
738 Common Report Var(flag_merge_constants,2) Init(1) Optimization
739 Attempt to merge identical constants and constant variables

741 fmerge-constants
742 Common Report Var(flag_merge_constants,1) VarExists Optimization
743 Attempt to merge identical constants across compilation units

745 fmerge-debug-strings
746 Common Report Var(flag_merge_debug_strings) Init(1)
747 Attempt to merge identical debug strings across compilation units

749 fmessage-length=
750 Common RejectNegative Joined UInteger
751 -fmessage-length=<number>       Limit diagnostics to <number> characters per lin

753 fmodulo-sched
754 Common Report Var(flag_modulo_sched) Optimization
755 Perform SMS based modulo scheduling before the first scheduling pass

757 fmodulo-sched-allow-regmoves
758 Common Report Var(flag_modulo_sched_allow_regmoves)
759 Perform SMS based modulo scheduling with register moves allowed

761 fmove-loop-invariants
762 Common Report Var(flag_move_loop_invariants) Init(1) Optimization
763 Move loop invariant computations out of loops

765 fmudflap
766 Common RejectNegative Report Var(flag_mudflap)
767 Add mudflap bounds-checking instrumentation for single-threaded program

769 fmudflapth
770 Common RejectNegative Report VarExists Var(flag_mudflap,2)
771 Add mudflap bounds-checking instrumentation for multi-threaded program

773 fmudflapir
774 Common RejectNegative Report Var(flag_mudflap_ignore_reads)
775 Ignore read operations when inserting mudflap instrumentation

777 fdce
778 Common Var(flag_dce) Init(1) Optimization
779 Use the RTL dead code elimination pass
```

```
781 fdse
782 Common Var(flag_dse) Init(1) Optimization
783 Use the RTL dead store elimination pass

785 freschedule-modulo-scheduled-loops
786 Common Report Var(flag_resched_modulo_sched) Optimization
787 Enable/Disable the traditional scheduling in loops that already passed modulo sc

789 fnon-call-exceptions
790 Common Report Var(flag_non_call_exceptions) Optimization
791 Support synchronous non-call exceptions

793 fomit-frame-pointer
794 Common Report Var(flag_omit_frame_pointer) Optimization
795 When possible do not generate stack frames

797 foptimize-register-move
798 Common Report Var(flag_regmove) Optimization
799 Do the full register move optimization pass

801 foptimize-sibling-calls
802 Common Report Var(flag_optimize_sibling_calls) Optimization
803 Optimize sibling and tail recursive calls

805 fpre-ipa-mem-report
806 Common Report Var(pre_ipa_mem_report)
807 Report on memory allocation before interprocedural optimization

809 fpost-ipa-mem-report
810 Common Report Var(post_ipa_mem_report)
811 Report on memory allocation before interprocedural optimization

813 fpack-struct
814 Common Report Var(flag_pack_struct) Optimization
815 Pack structure members together without holes

817 fpack-struct=
818 Common RejectNegative Joined UInteger Optimization
819 -fpack-struct=<number>  Set initial maximum structure member alignment

821 fpcc-struct-return
822 Common Report Var(flag_pcc_struct_return,1) VarExists
823 Return small aggregates in memory, not registers

825 fpeel-loops
826 Common Report Var(flag_peel_loops) Optimization
827 Perform loop peeling

829 fpeephole
830 Common Report Var(flag_no_peephole,0) Optimization
831 Enable machine specific peephole optimizations

833 fpeephole2
834 Common Report Var(flag_peephole2) Optimization
835 Enable an RTL peephole pass before sched2

837 fPIC
838 Common Report Var(flag_pic,2)
839 Generate position-independent code if possible (large mode)

841 fPIE
842 Common Report Var(flag_pie,2)
843 Generate position-independent code for executables if possible (large mode)

845 fpic
846 Common Report Var(flag_pic,1) VarExists
```



```
847 Generate position-independent code if possible (small mode)

849 fpie
850 Common Report Var(flag_pie,1) VarExists
851 Generate position-independent code for executables if possible (small mode)

853 fpredictive-commoning
854 Common Report Var(flag_predictive_commoning) Optimization
855 Run predictive commoning optimization.

857 fprefetch-loop-arrays
858 Common Report Var(flag_prefetch_loop_arrays) Optimization
859 Generate prefetch instructions, if available, for arrays in loops

861 fprofile
862 Common Report Var(profile_flag)
863 Enable basic program profiling code

865 fprofile-arcs
866 Common Report Var(profile_arc_flag)
867 Insert arc-based program profiling code

869 fprofile-dir=
870 Common Joined RejectNegative
871 Set the top-level directory for storing the profile data.
872 The default is 'pwd'.

874 fprofile-correction
875 Common Report Var(flag_profile_correction)
876 Enable correction of flow inconsistent profile data input

878 fprofile-generate
879 Common
880 Enable common options for generating profile info for profile feedback directed

882 fprofile-generate=
883 Common Joined RejectNegative
884 Enable common options for generating profile info for profile feedback directed

886 fprofile-use
887 Common Var(flag_profile_use)
888 Enable common options for performing profile feedback directed optimizations

890 fprofile-use=
891 Common Joined RejectNegative
892 Enable common options for performing profile feedback directed optimizations, an

894 fprofile-values
895 Common Report Var(flag_profile_values)
896 Insert code to profile values of expressions

898 frandom-seed
899 Common

901 frandom-seed=
902 Common Joined RejectNegative
903 -frandom-seed=<string>  Make compile reproducible using <string>

905 ; This switch causes the command line that was used to create an
906 ; object file to be recorded into the object file.  The exact format
907 ; of this recording is target and binary file format dependent.
908 ; It is related to the -fverbose-asm switch, but that switch only
909 ; records information in the assembler output file as comments, so
910 ; they never reach the object file.
911 frecord-gcc-switches
912 Common Report Var(flag_record_gcc_switches)
```

```
913 Record gcc command line switches in the object file.

915 freg-struct-return
916 Common Report Var(flag_pcc_struct_return,0) VarExists Optimization
917 Return small aggregates in registers

919 fregmove
920 Common Report Var(flag_regmove) Optimization
921 Enables a register move optimization

923 frename-registers
924 Common Report Var(flag_rename_registers) Init(2) Optimization
925 Perform a register renaming optimization pass

927 freorder-blocks
928 Common Report Var(flag_reorder_blocks) Optimization
929 Reorder basic blocks to improve code placement

931 freorder-blocks-and-partition
932 Common Report Var(flag_reorder_blocks_and_partition) Optimization
933 Reorder basic blocks and partition into hot and cold sections

935 freorder-functions
936 Common Report Var(flag_reorder_functions) Optimization
937 Reorder functions to improve code placement

939 frerun-cse-after-loop
940 Common Report Var(flag_rerun_cse_after_loop) Init(2) Optimization
941 Add a common subexpression elimination pass after loop optimizations

943 frerun-loop-opt
944 Common
945 Does nothing.  Preserved for backward compatibility.

947 frounding-math
948 Common Report Var(flag_rounding_math) Optimization
949 Disable optimizations that assume default FP rounding behavior

951 fsched-interblock
952 Common Report Var(flag_schedule_interblock) Init(1) Optimization
953 Enable scheduling across basic blocks

955 fsched-spec
956 Common Report Var(flag_schedule_speculative) Init(1) Optimization
957 Allow speculative motion of non-loads

959 fsched-spec-load
960 Common Report Var(flag_schedule_speculative_load) Optimization
961 Allow speculative motion of some loads

963 fsched-spec-load-dangerous
964 Common Report Var(flag_schedule_speculative_load_dangerous) Optimization
965 Allow speculative motion of more loads

967 fsched-verbose=
968 Common RejectNegative Joined
969 -fsched-verbose=<number>        Set the verbosity level of the scheduler

971 fsched2-use-superblocks
972 Common Report Var(flag_sched2_use_superblocks) Optimization
973 If scheduling post reload, do superblock scheduling

975 fsched2-use-traces
976 Common Report Var(flag_sched2_use_traces) Optimization
977 If scheduling post reload, do trace scheduling
```



```
 979 fschedule-insns
 980 Common Report Var(flag_schedule_insns) Optimization
 981 Reschedule instructions before register allocation

 983 fschedule-insns2
 984 Common Report Var(flag_schedule_insns_after_reload) Optimization
 985 Reschedule instructions after register allocation

 987 ; This flag should be on when a target implements non-trivial
 988 ; scheduling hooks, maybe saving some information for its own sake.
 989 ; On IA64, for example, this is used for correct bundling.
 990 fselective-scheduling
 991 Common Report Var(flag_selective_scheduling) Optimization
 992 Schedule instructions using selective scheduling algorithm

 994 fselective-scheduling2
 995 Common Report Var(flag_selective_scheduling2) Optimization
 996 Run selective scheduling after reload

 998 fsel-sched-pipelining
 999 Common Report Var(flag_sel_sched_pipelining) Init(0) Optimization
1000 Perform software pipelining of inner loops during selective scheduling

1002 fsel-sched-pipelining-outer-loops
1003 Common Report Var(flag_sel_sched_pipelining_outer_loops) Init(0) Optimization
1004 Perform software pipelining of outer loops during selective scheduling

1006 fsel-sched-reschedule-pipelined
1007 Common Report Var(flag_sel_sched_reschedule_pipelined) Init(0) Optimization
1008 Reschedule pipelined regions without pipelining

1010 ; sched_stalled_insns means that insns can be moved prematurely from the queue
1011 ; of stalled insns into the ready list.
1012 fsched-stalled-insns
1013 Common Report Var(flag_sched_stalled_insns) Optimization UInteger
1014 Allow premature scheduling of queued insns

1016 fsched-stalled-insns=
1017 Common RejectNegative Joined UInteger
1018 -fsched-stalled-insns=<number>  Set number of queued insns that can be premature

1020 ; sched_stalled_insns_dep controls how many recently scheduled cycles will
1021 ; be examined for a dependency on a stalled insn that is candidate for
1022 ; premature removal from the queue of stalled insns into the ready list (has
1023 ; an effect only if the flag 'sched_stalled_insns' is set).
1024 fsched-stalled-insns-dep
1025 Common Report Var(flag_sched_stalled_insns_dep,1) Init(1) Optimization UInteger
1026 Set dependence distance checking in premature scheduling of queued insns

1028 fsched-stalled-insns-dep=
1029 Common RejectNegative Joined UInteger
1030 -fsched-stalled-insns-dep=<number>      Set dependence distance checking in prem

1032 fsection-anchors
1033 Common Report Var(flag_section_anchors) Optimization
1034 Access data in the same section from shared anchor points

1036 frtl-abstract-sequences
1037 Common Report Var(flag_rtl_seqabstr) Optimization
1038 Perform sequence abstraction optimization on RTL

1040 fsee
1041 Common Report Var(flag_see) Init(0)
1042 Eliminate redundant sign extensions using LCM.

1044 fshow-column
```

```
Common C ObjC C++ ObjC++ Report Var(flag_show_column) Init(0)
Show column numbers in diagnostics, when available.  Default off

fsignaling-nans
Common Report Var(flag_signaling_nans) Optimization
Disable optimizations observable by IEEE signaling NaNs

fsigned-zeros
Common Report Var(flag_signed_zeros) Init(1) Optimization
Disable floating point optimizations that ignore the IEEE signedness of zero

fsingle-precision-constant
Common Report Var(flag_single_precision_constant) Optimization
Convert floating point constants to single precision constants

fsplit-ivs-in-unroller
Common Report Var(flag_split_ivs_in_unroller) Init(1) Optimization
Split lifetimes of induction variables when loops are unrolled

fsplit-wide-types
Common Report Var(flag_split_wide_types) Optimization
Split wide types into independent registers

fvariable-expansion-in-unroller
Common Report Var(flag_variable_expansion_in_unroller) Optimization
Apply variable expansion when loops are unrolled

fstack-check=
Common Report RejectNegative Joined
-fstack-check=[no|generic|specific]     Insert stack checking code into the prog

fstack-check
Common Report
Insert stack checking code into the program.  Same as -fstack-check=specific

fstack-limit
Common

fstack-limit-register=
Common RejectNegative Joined
-fstack-limit-register=<register>       Trap if the stack goes past <register>

fstack-limit-symbol=
Common RejectNegative Joined
-fstack-limit-symbol=<name>     Trap if the stack goes past symbol <name>

fstack-protector
Common Report Var(flag_stack_protect, 1)
Use propolice as a stack protection method

fstack-protector-all
Common Report RejectNegative Var(flag_stack_protect, 2) VarExists
Use a stack protection method for every function

fstrength-reduce
Common
Does nothing.  Preserved for backward compatibility.

; Nonzero if we should do (language-dependent) alias analysis.
; Typically, this analysis will assume that expressions of certain
; types do not alias expressions of certain other types.  Only used
; if alias analysis (in general) is enabled.
fstrict-aliasing
Common Report Var(flag_strict_aliasing) Optimization
Assume strict aliasing rules apply
```



```
fstrict-calling-conventions
Common Report Var(flag_strict_calling_conventions) Init(1)
Use strict ABI calling conventions even for static functions

#endif /* ! codereview */
fstrict-overflow
Common Report Var(flag_strict_overflow)
Treat signed overflow as undefined

fsyntax-only
Common Report Var(flag_syntax_only)
Check for syntax errors, then stop

ftest-coverage
Common Report Var(flag_test_coverage)
Create data files needed by \"gcov\"

fthread-jumps
Common Report Var(flag_thread_jumps) Optimization
Perform jump threading optimizations

ftime-report
Common Report Var(time_report)
Report the time taken by each compiler pass

ftls-model=
Common Joined RejectNegative
-ftls-model=[global-dynamic|local-dynamic|initial-exec|local-exec]      Set the

ftoplevel-reorder
Common Report Var(flag_toplevel_reorder) Init(2) Optimization
Reorder top level functions, variables, and asms

ftracer
Common Report Var(flag_tracer)
Perform superblock formation via tail duplication

; Zero means that floating-point math operations cannot generate a
; (user-visible) trap.  This is the case, for example, in nonstop
; IEEE 754 arithmetic.
ftrapping-math
Common Report Var(flag_trapping_math) Init(1) Optimization
Assume floating-point operations can trap

ftrapv
Common Report Var(flag_trapv) Optimization
Trap for signed overflow in addition, subtraction and multiplication

ftree-ccp
Common Report Var(flag_tree_ccp) Optimization
Enable SSA-CCP optimization on trees

ftree-store-ccp
Common
Does nothing.  Preserved for backward compatibility.

ftree-ch
Common Report Var(flag_tree_ch) Optimization
Enable loop header copying on trees

ftree-copyrename
Common Report Var(flag_tree_copyrename) Optimization
Replace SSA temporaries with better names in copies

ftree-copy-prop
Common Report Var(flag_tree_copy_prop) Optimization
```

```
1177 Enable copy propagation on trees

1179 ftree-store-copy-prop
1180 Common
1181 Does nothing.  Preserved for backward compatibility.

1183 ftree-cselim
1184 Common Report Var(flag_tree_cselim) Init(2) Optimization
1185 Transform condition stores into unconditional ones

1187 ftree-switch-conversion
1188 Common Report Var(flag_tree_switch_conversion) Optimization
1189 Perform conversions of switch initializations.

1191 ftree-dce
1192 Common Report Var(flag_tree_dce) Optimization
1193 Enable SSA dead code elimination optimization on trees

1195 ftree-dominator-opts
1196 Common Report Var(flag_tree_dom) Optimization
1197 Enable dominator optimizations

1199 ftree-dse
1200 Common Report Var(flag_tree_dse) Optimization
1201 Enable dead store elimination

1203 ftree-fre
1204 Common Report Var(flag_tree_fre) Optimization
1205 Enable Full Redundancy Elimination (FRE) on trees

1207 ftree-loop-distribution
1208 Common Report Var(flag_tree_loop_distribution) Optimization
1209 Enable loop distribution on trees

1211 ftree-loop-im
1212 Common Report Var(flag_tree_loop_im) Init(1) Optimization
1213 Enable loop invariant motion on trees

1215 ftree-loop-linear
1216 Common Report Var(flag_tree_loop_linear) Optimization
1217 Enable linear loop transforms on trees

1219 ftree-loop-ivcanon
1220 Common Report Var(flag_tree_loop_ivcanon) Init(1) Optimization
1221 Create canonical induction variables in loops

1223 ftree-loop-optimize
1224 Common Report Var(flag_tree_loop_optimize) Init(1) Optimization
1225 Enable loop optimizations on tree level

1227 ftree-parallelize-loops=
1228 Common Report Joined UInteger Var(flag_tree_parallelize_loops) Init(1)
1229 Enable automatic parallelization of loops

1231 ftree-pre
1232 Common Report Var(flag_tree_pre) Optimization
1233 Enable SSA-PRE optimization on trees

1235 ftree-reassoc
1236 Common Report Var(flag_tree_reassoc) Init(1) Optimization
1237 Enable reassociation on tree level

1239 ftree-salias
1240 Common
1241 Does nothing.  Preserved for backward compatibility.
```



```
1243 ftree-sink
1244 Common Report Var(flag_tree_sink) Optimization
1245 Enable SSA code sinking on trees

1247 ftree-sra
1248 Common Report Var(flag_tree_sra) Optimization
1249 Perform scalar replacement of aggregates

1251 ftree-ter
1252 Common Report Var(flag_tree_ter) Init(1) Optimization
1253 Replace temporary expressions in the SSA->normal pass

1255 ftree-lrs
1256 Common Report Var(flag_tree_live_range_split) Optimization
1257 Perform live range splitting during the SSA->normal pass

1259 ftree-vrp
1260 Common Report Var(flag_tree_vrp) Init(0) Optimization
1261 Perform Value Range Propagation on trees

1263 funit-at-a-time
1264 Common Report Var(flag_unit_at_a_time) Init(1) Optimization
1265 Compile whole compilation unit at a time

1267 funroll-loops
1268 Common Report Var(flag_unroll_loops) Optimization
1269 Perform loop unrolling when iteration count is known

1271 funroll-all-loops
1272 Common Report Var(flag_unroll_all_loops) Optimization
1273 Perform loop unrolling for all loops

1275 ; Nonzero means that loop optimizer may assume that the induction variables
1276 ; that control loops do not overflow and that the loops with nontrivial
1277 ; exit condition are not infinite
1278 funsafe-loop-optimizations
1279 Common Report Var(flag_unsafe_loop_optimizations) Optimization
1280 Allow loop optimizations to assume that the loops behave in normal way

1282 fassociative-math
1283 Common Report Var(flag_associative_math)
1284 Allow optimization for floating-point arithmetic which may change the
1285 result of the operation due to rounding.

1287 freciprocal-math
1288 Common Report Var(flag_reciprocal_math)
1289 Same as -fassociative-math for expressions which include division.

1291 ; Nonzero means that unsafe floating-point math optimizations are allowed
1292 ; for the sake of speed.  IEEE compliance is not guaranteed, and operations
1293 ; are allowed to assume that their arguments and results are "normal"
1294 ; (e.g., nonnegative for SQRT).
1295 funsafe-math-optimizations
1296 Common Report Var(flag_unsafe_math_optimizations) Optimization
1297 Allow math optimizations that may violate IEEE or ISO standards

1299 funswitch-loops
1300 Common Report Var(flag_unswitch_loops) Optimization
1301 Perform loop unswitching

1303 funwind-tables
1304 Common Report Var(flag_unwind_tables) Optimization
1305 Just generate unwind tables for exception handling

1307 fvar-tracking
1308 Common Report Var(flag_var_tracking) VarExists Optimization
```

```
Perform variable tracking

fvar-tracking-uninit
Common Report Var(flag_var_tracking_uninit) Optimization
Perform variable tracking and also tag variables that are uninitialized

ftree-vectorize
Common Report Var(flag_tree_vectorize) Optimization
Enable loop vectorization on trees

fvect-cost-model
Common Report Var(flag_vect_cost_model) Optimization
Enable use of cost model in vectorization

ftree-vect-loop-version
Common Report Var(flag_tree_vect_loop_version) Init(1) Optimization
Enable loop versioning when doing loop vectorization on trees

ftree-vectorizer-verbose=
Common RejectNegative Joined
-ftree-vectorizer-verbose=<number>      Set the verbosity level of the vectorize

ftree-scev-cprop
Common Report Var(flag_tree_scev_cprop) Init(1) Optimization
Enable copy propagation of scalar-evolution information.

; -fverbose-asm causes extra commentary information to be produced in
; the generated assembly code (to make it more readable).  This option
; is generally only of use to those who actually need to read the
; generated assembly code (perhaps while debugging the compiler itself).
; -fno-verbose-asm, the default, causes the extra information
; to not be added and is useful when comparing two assembler files.
fverbose-asm
Common Report Var(flag_verbose_asm)
Add extra commentary to assembler output

fvisibility=
Common Joined RejectNegative
-fvisibility=[default|internal|hidden|protected]        Set the default symbol v


fvpt
Common Report Var(flag_value_profile_transformations) Optimization
Use expression value profiles in optimizations

fweb
Common Report Var(flag_web) Init(2) Optimization
Construct webs and split unrelated uses of single variable

ftree-builtin-call-dce
Common Report Var(flag_tree_builtin_call_dce) Init(0) Optimization
Enable conditional dead code elimination for builtin calls

fwhole-program
Common Report Var(flag_whole_program) Init(0) Optimization
Perform whole program optimizations

fwrapv
Common Report Var(flag_wrapv) Optimization
Assume signed arithmetic overflow wraps around

fzero-initialized-in-bss
Common Report Var(flag_zero_initialized_in_bss) Init(1)
Put zero initialized data in the bss section

g
```



```
Common JoinedOrMissing
Generate debug information in default format

gcoff
Common JoinedOrMissing Negative(gdwarf-2)
Generate debug information in COFF format

gdwarf-2
Common JoinedOrMissing Negative(gstabs)
Generate debug information in DWARF v2 format

ggdb
Common JoinedOrMissing
Generate debug information in default extended format

gstabs
Common JoinedOrMissing Negative(gstabs+)
Generate debug information in STABS format

gstabs+
Common JoinedOrMissing Negative(gvms)
Generate debug information in extended STABS format

gvms
Common JoinedOrMissing Negative(gxcoff)
Generate debug information in VMS format

gxcoff
Common JoinedOrMissing Negative(gxcoff+)
Generate debug information in XCOFF format

gxcoff+
Common JoinedOrMissing Negative(gcoff)
Generate debug information in extended XCOFF format

o
Common Joined Separate
-o <file>       Place output into <file>

p
Common Var(profile_flag)
Enable function profiling

pedantic
Common Var(pedantic)
Issue warnings needed for strict compliance to the standard

pedantic-errors
Common
Like -pedantic but issue them as errors

quiet
Common Var(quiet_flag)
Do not display functions compiled or elapsed time

version
Common Var(version_flag)
Display the compiler's version

w
Common Var(inhibit_warnings)
Suppress warnings

shared
Common RejectNegative Negative(pie)
Create a shared library
```

```
1442 pie
1443 Common RejectNegative Negative(shared)
1444 Create a position independent executable

1446 ; This comment is to ensure we retain the blank line above.
```

```
**********************************************************
  964570 Sun Oct 28 20:56:07 2012
new/gcc/config/i386/i386.c
Implement -fstrict-calling-conventions
Stock GCC is overly willing to violate the ABI when calling local functions,
such that it passes arguments in registers on i386.  This hampers debugging
with anything other than a fully-aware DWARF debugger, and is generally not
something we desire.
Implement a flag which disables this behaviour, enabled by default.  The flag is
global, though only effective on i386, to more easily allow its globalization
later which, given the odds, is likely to be necessary.
**********************************************************
______unchanged_portion_omitted_
```



```
4346 /* Return the regparm value for a function with the indicated TYPE and DECL.
4347    DECL may be NULL when calling function indirectly
4348    or considering a libcall.  */

4350 static int
4351 ix86_function_regparm (const_tree type, const_tree decl)
4352 {
4353   tree attr;
4354   int regparm;

4356   static bool error_issued;

4358   if (TARGET_64BIT)
4359     return (ix86_function_type_abi (type) == SYSV_ABI
4360             ? X86_64_REGPARM_MAX : X64_REGPARM_MAX);

4362   regparm = ix86_regparm;
4363   attr = lookup_attribute ("regparm", TYPE_ATTRIBUTES (type));
4364   if (attr)
4365     {
4366       regparm
4367         = TREE_INT_CST_LOW (TREE_VALUE (TREE_VALUE (attr)));

4369       if (decl && TREE_CODE (decl) == FUNCTION_DECL)
4370         {
4371           /* We can't use regparm(3) for nested functions because
4372              these pass static chain pointer in %ecx register.  */
4373           if (!error_issued && regparm == 3
4374               && decl_function_context (decl)
4375               && !DECL_NO_STATIC_CHAIN (decl))
4376             {
4377               error ("nested functions are limited to 2 register parameters");
4378               error_issued = true;
4379               return 0;
4380             }
4381         }

4383       return regparm;
4384     }

4386   if (lookup_attribute ("fastcall", TYPE_ATTRIBUTES (type)))
4387     return 2;

4389   /* Use register calling convention for local functions when possible.  */
4390   if (decl
4391       && TREE_CODE (decl) == FUNCTION_DECL
4392       && optimize
4393       && (TARGET_64BIT || !flag_strict_calling_conventions)
4394 #endif /* ! codereview */
4395       && !profile_flag)
4396     {
4397       /* FIXME: remove this CONST_CAST when cgraph.[ch] is constified.  */
4398       struct cgraph_local_info *i = cgraph_local_info (CONST_CAST_TREE(decl));
4399       if (i && i->local)
4400         {
4401           int local_regparm, globals = 0, regno;
4402           struct function *f;

4404           /* Make sure no regparm register is taken by a
4405              fixed register variable.  */
4406           for (local_regparm = 0; local_regparm < REGPARM_MAX; local_regparm++)
4407             if (fixed_regs[local_regparm])
4408               break;

4410           /* We can't use regparm(3) for nested functions as these use
4411              static chain pointer in third argument.  */
```

```
4412           if (local_regparm == 3
4413               && decl_function_context (decl)
4414               && !DECL_NO_STATIC_CHAIN (decl))
4415             local_regparm = 2;

4417           /* If the function realigns its stackpointer, the prologue will
4418              clobber %ecx.  If we've already generated code for the callee,
4419              the callee DECL_STRUCT_FUNCTION is gone, so we fall back to
4420              scanning the attributes for the self-realigning property.  */
4421           f = DECL_STRUCT_FUNCTION (decl);
4422           /* Since current internal arg pointer won't conflict with
4423              parameter passing regs, so no need to change stack
4424              realignment and adjust regparm number.

4426              Each fixed register usage increases register pressure,
4427              so less registers should be used for argument passing.
4428              This functionality can be overriden by an explicit
4429              regparm value.  */
4430           for (regno = 0; regno <= DI_REG; regno++)
4431             if (fixed_regs[regno])
4432               globals++;

4434           local_regparm
4435             = globals < local_regparm ? local_regparm - globals : 0;

4437           if (local_regparm > regparm)
4438             regparm = local_regparm;
4439         }
4440     }

4442   return regparm;
4443 }

4445 /* Return 1 or 2, if we can pass up to SSE_REGPARM_MAX SFmode (1) and
4446    DFmode (2) arguments in SSE registers for a function with the
4447    indicated TYPE and DECL.  DECL may be NULL when calling function
4448    indirectly or considering a libcall.  Otherwise return 0.  */

4450 static int
4451 ix86_function_sseregparm (const_tree type, const_tree decl, bool warn)
4452 {
4453   gcc_assert (!TARGET_64BIT);

4455   /* Use SSE registers to pass SFmode and DFmode arguments if requested
4456      by the sseregparm attribute.  */
4457   if (TARGET_SSEREGPARM
4458       || (type && lookup_attribute ("sseregparm", TYPE_ATTRIBUTES (type))))
4459     {
4460       if (!TARGET_SSE)
4461         {
4462           if (warn)
4463             {
4464               if (decl)
4465                 error ("Calling %qD with attribute sseregparm without "
4466                        "SSE/SSE2 enabled", decl);
4467               else
4468                 error ("Calling %qT with attribute sseregparm without "
4469                        "SSE/SSE2 enabled", type);
4470             }
4471           return 0;
4472         }

4474       return 2;
4475     }

4477   /* For local functions, pass up to SSE_REGPARM_MAX SFmode
```



```
4478      (and DFmode for SSE2) arguments in SSE registers.  */
4479   if (decl && TARGET_SSE_MATH && optimize && !profile_flag &&
4480     (TARGET_64BIT || !flag_strict_calling_conventions))
4393   if (decl && TARGET_SSE_MATH && optimize && !profile_flag)
4481     {
4482       /* FIXME: remove this CONST_CAST when cgraph.[ch] is constified.  */
4483       struct cgraph_local_info *i = cgraph_local_info (CONST_CAST_TREE(decl));
4484       if (i && i->local)
4485         return TARGET_SSE2 ? 2 : 1;
4486     }

4488   return 0;
4489 }
```

_____*unchanged_portion_omitted_*

```
**********************************************************
  651723 Sun Oct 28 20:56:09 2012
new/gcc/doc/invoke.texi
Implement -fstrict-calling-conventions
Stock GCC is overly willing to violate the ABI when calling local functions,
such that it passes arguments in registers on i386.  This hampers debugging
with anything other than a fully-aware DWARF debugger, and is generally not
something we desire.
Implement a flag which disables this behaviour, enabled by default.  The flag is
global, though only effective on i386, to more easily allow its globalization
later which, given the odds, is likely to be necessary.
**********************************************************
______unchanged_portion_omitted_

2811 The option @option{-Wextra} also prints warning messages for the
2812 following cases:

2814 @itemize @bullet

2816 @item
2817 A pointer is compared against integer zero with @samp{<}, @samp{<=},
2818 @samp{>}, or @samp{>=}.

2820 @item
2821 (C++ only) An enumerator and a non-enumerator both appear in a
2822 conditional expression.

2824 @item
2825 (C++ only) Ambiguous virtual bases.

2827 @item
2828 (C++ only) Subscripting an array which has been declared @samp{register}.

2830 @item
2831 (C++ only) Taking the address of a variable which has been declared
2832 @samp{register}.

2834 @item
2835 (C++ only) A base class is not initialized in a derived class' copy
2836 constructor.

2838 @end itemize

2840 @item -Wchar-subscripts
2841 @opindex Wchar-subscripts
2842 @opindex Wno-char-subscripts
2843 Warn if an array subscript has type @code{char}.  This is a common cause
2844 of error, as programmers often forget that this type is signed on some
2845 machines.
2846 This warning is enabled by @option{-Wall}.

2848 @item -Wcomment
2849 @opindex Wcomment
2850 @opindex Wno-comment
2851 Warn whenever a comment-start sequence @samp{/*} appears in a @samp{/*}
2852 comment, or whenever a Backslash-Newline appears in a @samp{//} comment.
2853 This warning is enabled by @option{-Wall}.

2855 @item -Wformat
2856 @opindex Wformat
2857 @opindex Wno-format
2858 @opindex ffreestanding
2859 @opindex fno-builtin
2860 Check calls to @code{printf} and @code{scanf}, etc., to make sure that
2861 the arguments supplied have types appropriate to the format string
2862 specified, and that the conversions specified in the format string make
```



```
2863 sense.  This includes standard functions, and others specified by format
2864 attributes (@pxref{Function Attributes}), in the @code{printf},
2865 @code{scanf}, @code{strftime} and @code{strfmon} (an X/Open extension,
2866 not in the C standard) families (or other target-specific families).
2867 Which functions are checked without format attributes having been
2868 specified depends on the standard version selected, and such checks of
2869 functions without the attribute specified are disabled by
2870 @option{-ffreestanding} or @option{-fno-builtin}.

2872 The formats are checked against the format features supported by GNU
2873 libc version 2.2.  These include all ISO C90 and C99 features, as well
2874 as features from the Single Unix Specification and some BSD and GNU
2875 extensions.  Other library implementations may not support all these
2876 features; GCC does not support warning about features that go beyond a
2877 particular library's limitations.  However, if @option{-pedantic} is used
2878 with @option{-Wformat}, warnings will be given about format features not
2879 in the selected standard version (but not for @code{strfmon} formats,
2880 since those are not in any version of the C standard).  @xref{C Dialect
2881 Options,,Options Controlling C Dialect}.

2883 Since @option{-Wformat} also checks for null format arguments for
2884 several functions, @option{-Wformat} also implies @option{-Wnonnull}.

2886 @option{-Wformat} is included in @option{-Wall}.  For more control over some
2887 aspects of format checking, the options @option{-Wformat-y2k},
2888 @option{-Wno-format-extra-args}, @option{-Wno-format-zero-length},
2889 @option{-Wformat-nonliteral}, @option{-Wformat-security}, and
2890 @option{-Wformat=2} are available, but are not included in @option{-Wall}.

2892 @item -Wformat-y2k
2893 @opindex Wformat-y2k
2894 @opindex Wno-format-y2k
2895 If @option{-Wformat} is specified, also warn about @code{strftime}
2896 formats which may yield only a two-digit year.

2898 @item -Wno-format-contains-nul
2899 @opindex Wno-format-contains-nul
2900 @opindex Wformat-contains-nul
2901 If @option{-Wformat} is specified, do not warn about format strings that
2902 contain NUL bytes.

2904 @item -Wno-format-extra-args
2905 @opindex Wno-format-extra-args
2906 @opindex Wformat-extra-args
2907 If @option{-Wformat} is specified, do not warn about excess arguments to a
2908 @code{printf} or @code{scanf} format function.  The C standard specifies
2909 that such arguments are ignored.

2911 Where the unused arguments lie between used arguments that are
2912 specified with @samp{$} operand number specifications, normally
2913 warnings are still given, since the implementation could not know what
2914 type to pass to @code{va_arg} to skip the unused arguments.  However,
2915 in the case of @code{scanf} formats, this option will suppress the
2916 warning if the unused arguments are all pointers, since the Single
2917 Unix Specification says that such unused arguments are allowed.

2919 @item -Wno-format-zero-length @r{(C and Objective-C only)}
2920 @opindex Wno-format-zero-length
2921 @opindex Wformat-zero-length
2922 If @option{-Wformat} is specified, do not warn about zero-length formats.
2923 The C standard specifies that zero-length formats are allowed.

2925 @item -Wformat-nonliteral
2926 @opindex Wformat-nonliteral
2927 @opindex Wno-format-nonliteral
2928 If @option{-Wformat} is specified, also warn if the format string is not a
```

```
2929 string literal and so cannot be checked, unless the format function
2930 takes its format arguments as a @code{va_list}.

2932 @item -Wformat-security
2933 @opindex Wformat-security
2934 @opindex Wno-format-security
2935 If @option{-Wformat} is specified, also warn about uses of format
2936 functions that represent possible security problems.  At present, this
2937 warns about calls to @code{printf} and @code{scanf} functions where the
2938 format string is not a string literal and there are no format arguments,
2939 as in @code{printf (foo);}.  This may be a security hole if the format
2940 string came from untrusted input and contains @samp{%n}.  (This is
2941 currently a subset of what @option{-Wformat-nonliteral} warns about, but
2942 in future warnings may be added to @option{-Wformat-security} that are not
2943 included in @option{-Wformat-nonliteral}.)

2945 @item -Wformat=2
2946 @opindex Wformat=2
2947 @opindex Wno-format=2
2948 Enable @option{-Wformat} plus format checks not included in
2949 @option{-Wformat}.  Currently equivalent to @samp{-Wformat
2950 -Wformat-nonliteral -Wformat-security -Wformat-y2k}.

2952 @item -Wnonnull @r{(C and Objective-C only)}
2953 @opindex Wnonnull
2954 @opindex Wno-nonnull
2955 Warn about passing a null pointer for arguments marked as
2956 requiring a non-null value by the @code{nonnull} function attribute.

2958 @option{-Wnonnull} is included in @option{-Wall} and @option{-Wformat}.  It
2959 can be disabled with the @option{-Wno-nonnull} option.

2961 @item -Winit-self @r{(C, C++, Objective-C and Objective-C++ only)}
2962 @opindex Winit-self
2963 @opindex Wno-init-self
2964 Warn about uninitialized variables which are initialized with themselves.
2965 Note this option can only be used with the @option{-Wuninitialized} option.

2967 For example, GCC will warn about @code{i} being uninitialized in the
2968 following snippet only when @option{-Winit-self} has been specified:
2969 @smallexample
2970 @group
2971 int f()
2972 @{
2973   int i = i;
2974   return i;
2975 @}
2976 @end group
2977 @end smallexample

2979 @item -Wimplicit-int @r{(C and Objective-C only)}
2980 @opindex Wimplicit-int
2981 @opindex Wno-implicit-int
2982 Warn when a declaration does not specify a type.
2983 This warning is enabled by @option{-Wall}.

2985 @item -Wimplicit-function-declaration @r{(C and Objective-C only)}
2986 @opindex Wimplicit-function-declaration
2987 @opindex Wno-implicit-function-declaration
2988 Give a warning whenever a function is used before being declared. In
2989 C99 mode (@option{-std=c99} or @option{-std=gnu99}), this warning is
2990 enabled by default and it is made into an error by
2991 @option{-pedantic-errors}. This warning is also enabled by
2992 @option{-Wall}.

2994 @item -Wimplicit
```



```
2995 @opindex Wimplicit
2996 @opindex Wno-implicit
2997 Same as @option{-Wimplicit-int} and @option{-Wimplicit-function-declaration}.
2998 This warning is enabled by @option{-Wall}.

3000 @item -Wignored-qualifiers @r{(C and C++ only)}
3001 @opindex Wignored-qualifiers
3002 @opindex Wno-ignored-qualifiers
3003 Warn if the return type of a function has a type qualifier
3004 such as @code{const}.  For ISO C such a type qualifier has no effect,
3005 since the value returned by a function is not an lvalue.
3006 For C++, the warning is only emitted for scalar types or @code{void}.
3007 ISO C prohibits qualified @code{void} return types on function
3008 definitions, so such return types always receive a warning
3009 even without this option.

3011 This warning is also enabled by @option{-Wextra}.

3013 @item -Wmain
3014 @opindex Wmain
3015 @opindex Wno-main
3016 Warn if the type of @samp{main} is suspicious.  @samp{main} should be
3017 a function with external linkage, returning int, taking either zero
3018 arguments, two, or three arguments of appropriate types.  This warning
3019 is enabled by default in C++ and is enabled by either @option{-Wall}
3020 or @option{-pedantic}.

3022 @item -Wmissing-braces
3023 @opindex Wmissing-braces
3024 @opindex Wno-missing-braces
3025 Warn if an aggregate or union initializer is not fully bracketed.  In
3026 the following example, the initializer for @samp{a} is not fully
3027 bracketed, but that for @samp{b} is fully bracketed.

3029 @smallexample
3030 int a[2][2] = @{ 0, 1, 2, 3 @};
3031 int b[2][2] = @{ @{ 0, 1 @}, @{ 2, 3 @} @};
3032 @end smallexample

3034 This warning is enabled by @option{-Wall}.

3036 @item -Wmissing-include-dirs @r{(C, C++, Objective-C and Objective-C++ only)}
3037 @opindex Wmissing-include-dirs
3038 @opindex Wno-missing-include-dirs
3039 Warn if a user-supplied include directory does not exist.

3041 @item -Wparentheses
3042 @opindex Wparentheses
3043 @opindex Wno-parentheses
3044 Warn if parentheses are omitted in certain contexts, such
3045 as when there is an assignment in a context where a truth value
3046 is expected, or when operators are nested whose precedence people
3047 often get confused about.

3049 Also warn if a comparison like @samp{x<=y<=z} appears; this is
3050 equivalent to @samp{(x<=y ? 1 : 0) <= z}, which is a different
3051 interpretation from that of ordinary mathematical notation.

3053 Also warn about constructions where there may be confusion to which
3054 @code{if} statement an @code{else} branch belongs.  Here is an example of
3055 such a case:

3057 @smallexample
3058 @group
3059 @{
3060   if (a)
```

```
    if (b)
      foo ();
  else
    bar ();
@}
@end group
@end smallexample

In C/C++, every @code{else} branch belongs to the innermost possible
@code{if} statement, which in this example is @code{if (b)}.  This is
often not what the programmer expected, as illustrated in the above
example by indentation the programmer chose.  When there is the
potential for this confusion, GCC will issue a warning when this flag
is specified.  To eliminate the warning, add explicit braces around
the innermost @code{if} statement so there is no way the @code{else}
could belong to the enclosing @code{if}.  The resulting code would
look like this:

@smallexample
@group
@{
  if (a)
    @{
      if (b)
        foo ();
      else
        bar ();
    @}
@}
@end group
@end smallexample

This warning is enabled by @option{-Wall}.

@item -Wsequence-point
@opindex Wsequence-point
@opindex Wno-sequence-point
Warn about code that may have undefined semantics because of violations
of sequence point rules in the C and C++ standards.

The C and C++ standards defines the order in which expressions in a C/C++
program are evaluated in terms of @dfn{sequence points}, which represent
a partial ordering between the execution of parts of the program: those
executed before the sequence point, and those executed after it.  These
occur after the evaluation of a full expression (one which is not part
of a larger expression), after the evaluation of the first operand of a
@code{&&}, @code{||}, @code{? :} or @code{,} (comma) operator, before a
function is called (but after the evaluation of its arguments and the
expression denoting the called function), and in certain other places.
Other than as expressed by the sequence point rules, the order of
evaluation of subexpressions of an expression is not specified.  All
these rules describe only a partial order rather than a total order,
since, for example, if two functions are called within one expression
with no sequence point between them, the order in which the functions
are called is not specified.  However, the standards committee have
ruled that function calls do not overlap.

It is not specified when between sequence points modifications to the
values of objects take effect.  Programs whose behavior depends on this
have undefined behavior; the C and C++ standards specify that ``Between
the previous and next sequence point an object shall have its stored
value modified at most once by the evaluation of an expression.
Furthermore, the prior value shall be read only to determine the value
to be stored.''.  If a program breaks these rules, the results on any
particular implementation are entirely unpredictable.
```



```
Examples of code with undefined behavior are @code{a = a++;}, @code{a[n]
= b[n++]} and @code{a[i++] = i;}.  Some more complicated cases are not
diagnosed by this option, and it may give an occasional false positive
result, but in general it has been found fairly effective at detecting
this sort of problem in programs.

The standard is worded confusingly, therefore there is some debate
over the precise meaning of the sequence point rules in subtle cases.
Links to discussions of the problem, including proposed formal
definitions, may be found on the GCC readings page, at
@w{@uref{http://gcc.gnu.org/readings.html}}.

This warning is enabled by @option{-Wall} for C and C++.

@item -Wreturn-type
@opindex Wreturn-type
@opindex Wno-return-type
Warn whenever a function is defined with a return-type that defaults
to @code{int}.  Also warn about any @code{return} statement with no
return-value in a function whose return-type is not @code{void}
(falling off the end of the function body is considered returning
without a value), and about a @code{return} statement with a
expression in a function whose return-type is @code{void}.

For C++, a function without return type always produces a diagnostic
message, even when @option{-Wno-return-type} is specified.  The only
exceptions are @samp{main} and functions defined in system headers.

This warning is enabled by @option{-Wall}.

@item -Wswitch
@opindex Wswitch
@opindex Wno-switch
Warn whenever a @code{switch} statement has an index of enumerated type
and lacks a @code{case} for one or more of the named codes of that
enumeration.  (The presence of a @code{default} label prevents this
warning.)  @code{case} labels outside the enumeration range also
provoke warnings when this option is used.
This warning is enabled by @option{-Wall}.

@item -Wswitch-default
@opindex Wswitch-default
@opindex Wno-switch-default
Warn whenever a @code{switch} statement does not have a @code{default}
case.

@item -Wswitch-enum
@opindex Wswitch-enum
@opindex Wno-switch-enum
Warn whenever a @code{switch} statement has an index of enumerated type
and lacks a @code{case} for one or more of the named codes of that
enumeration.  @code{case} labels outside the enumeration range also
provoke warnings when this option is used.

@item -Wsync-nand @r{(C and C++ only)}
@opindex Wsync-nand
@opindex Wno-sync-nand
Warn when @code{__sync_fetch_and_nand} and @code{__sync_nand_and_fetch}
built-in functions are used.  These functions changed semantics in GCC 4.4.

@item -Wtrigraphs
@opindex Wtrigraphs
@opindex Wno-trigraphs
Warn if any trigraphs are encountered that might change the meaning of
the program (trigraphs within comments are not warned about).
This warning is enabled by @option{-Wall}.
```

```
@item -Wunused-function
@opindex Wunused-function
@opindex Wno-unused-function
Warn whenever a static function is declared but not defined or a
non-inline static function is unused.
This warning is enabled by @option{-Wall}.

@item -Wunused-label
@opindex Wunused-label
@opindex Wno-unused-label
Warn whenever a label is declared but not used.
This warning is enabled by @option{-Wall}.

To suppress this warning use the @samp{unused} attribute
(@pxref{Variable Attributes}).

@item -Wunused-parameter
@opindex Wunused-parameter
@opindex Wno-unused-parameter
Warn whenever a function parameter is unused aside from its declaration.

To suppress this warning use the @samp{unused} attribute
(@pxref{Variable Attributes}).

@item -Wunused-variable
@opindex Wunused-variable
@opindex Wno-unused-variable
Warn whenever a local variable or non-constant static variable is unused
aside from its declaration.
This warning is enabled by @option{-Wall}.

To suppress this warning use the @samp{unused} attribute
(@pxref{Variable Attributes}).

@item -Wunused-value
@opindex Wunused-value
@opindex Wno-unused-value
Warn whenever a statement computes a result that is explicitly not
used. To suppress this warning cast the unused expression to
@samp{void}. This includes an expression-statement or the left-hand
side of a comma expression that contains no side effects. For example,
an expression such as @samp{x[i,j]} will cause a warning, while
@samp{x[(void)i,j]} will not.

This warning is enabled by @option{-Wall}.

@item -Wunused
@opindex Wunused
@opindex Wno-unused
All the above @option{-Wunused} options combined.

In order to get a warning about an unused function parameter, you must
either specify @samp{-Wextra -Wunused} (note that @samp{-Wall} implies
@samp{-Wunused}), or separately specify @option{-Wunused-parameter}.

@item -Wuninitialized
@opindex Wuninitialized
@opindex Wno-uninitialized
Warn if an automatic variable is used without first being initialized
or if a variable may be clobbered by a @code{setjmp} call. In C++,
warn if a non-static reference or non-static @samp{const} member
appears in a class without constructors.

If you want to warn about code which uses the uninitialized value of the
variable in its own initializer, use the @option{-Winit-self} option.
```



```
These warnings occur for individual uninitialized or clobbered
elements of structure, union or array variables as well as for
variables which are uninitialized or clobbered as a whole.  They do
not occur for variables or elements declared @code{volatile}.  Because
these warnings depend on optimization, the exact variables or elements
for which there are warnings will depend on the precise optimization
options and version of GCC used.

Note that there may be no warning about a variable that is used only
to compute a value that itself is never used, because such
computations may be deleted by data flow analysis before the warnings
are printed.

These warnings are made optional because GCC is not smart
enough to see all the reasons why the code might be correct
despite appearing to have an error.  Here is one example of how
this can happen:

@smallexample
@group
@{
  int x;
  switch (y)
    @{
    case 1: x = 1;
      break;
    case 2: x = 4;
      break;
    case 3: x = 5;
    @}
  foo (x);
@}
@end group
@end smallexample

@noindent
If the value of @code{y} is always 1, 2 or 3, then @code{x} is
always initialized, but GCC doesn't know this.  Here is
another common case:

@smallexample
@{
  int save_y;
  if (change_y) save_y = y, y = new_y;
  @dots{}
  if (change_y) y = save_y;
@}
@end smallexample

@noindent
This has no bug because @code{save_y} is used only if it is set.

@cindex @code{longjmp} warnings
This option also warns when a non-volatile automatic variable might be
changed by a call to @code{longjmp}.  These warnings as well are possible
only in optimizing compilation.

The compiler sees only the calls to @code{setjmp}.  It cannot know
where @code{longjmp} will be called; in fact, a signal handler could
call it at any point in the code.  As a result, you may get a warning
even when there is in fact no problem because @code{longjmp} cannot
in fact be called at the place which would cause a problem.

Some spurious warnings can be avoided if you declare all the functions
you use that never return as @code{noreturn}.  @xref{Function
```

```
Attributes}.

This warning is enabled by @option{-Wall} or @option{-Wextra}.

@item -Wunknown-pragmas
@opindex Wunknown-pragmas
@opindex Wno-unknown-pragmas
@cindex warning for unknown pragmas
@cindex unknown pragmas, warning
@cindex pragmas, warning of unknown
Warn when a #pragma directive is encountered which is not understood by
GCC@.  If this command line option is used, warnings will even be issued
for unknown pragmas in system header files.  This is not the case if
the warnings were only enabled by the @option{-Wall} command line option.

@item -Wno-pragmas
@opindex Wno-pragmas
@opindex Wpragmas
Do not warn about misuses of pragmas, such as incorrect parameters,
invalid syntax, or conflicts between pragmas.  See also
@samp{-Wunknown-pragmas}.

@item -Wstrict-aliasing
@opindex Wstrict-aliasing
@opindex Wno-strict-aliasing
This option is only active when @option{-fstrict-aliasing} is active.
It warns about code which might break the strict aliasing rules that the
compiler is using for optimization.  The warning does not catch all
cases, but does attempt to catch the more common pitfalls.  It is
included in @option{-Wall}.
It is equivalent to @option{-Wstrict-aliasing=3}

@item -Wstrict-aliasing=n
@opindex Wstrict-aliasing=n
@opindex Wno-strict-aliasing=n
This option is only active when @option{-fstrict-aliasing} is active.
It warns about code which might break the strict aliasing rules that the
compiler is using for optimization.
Higher levels correspond to higher accuracy (fewer false positives).
Higher levels also correspond to more effort, similar to the way -O works.
@option{-Wstrict-aliasing} is equivalent to @option{-Wstrict-aliasing=n},
with n=3.

Level 1: Most aggressive, quick, least accurate.
Possibly useful when higher levels
do not warn but -fstrict-aliasing still breaks the code, as it has very few
false negatives.  However, it has many false positives.
Warns for all pointer conversions between possibly incompatible types,
even if never dereferenced.  Runs in the frontend only.

Level 2: Aggressive, quick, not too precise.
May still have many false positives (not as many as level 1 though),
and few false negatives (but possibly more than level 1).
Unlike level 1, it only warns when an address is taken.  Warns about
incomplete types.  Runs in the frontend only.

Level 3 (default for @option{-Wstrict-aliasing}):
Should have very few false positives and few false
negatives.  Slightly slower than levels 1 or 2 when optimization is enabled.
Takes care of the common punn+dereference pattern in the frontend:
@code{*(int*)&some_float}.
If optimization is enabled, it also runs in the backend, where it deals
with multiple statement cases using flow-sensitive points-to information.
Only warns when the converted pointer is dereferenced.
Does not warn about incomplete types.
```



```
@item -Wstrict-overflow
@itemx -Wstrict-overflow=@var{n}
@opindex Wstrict-overflow
@opindex Wno-strict-overflow
This option is only active when @option{-fstrict-overflow} is active.
It warns about cases where the compiler optimizes based on the
assumption that signed overflow does not occur.  Note that it does not
warn about all cases where the code might overflow: it only warns
about cases where the compiler implements some optimization.  Thus
this warning depends on the optimization level.

An optimization which assumes that signed overflow does not occur is
perfectly safe if the values of the variables involved are such that
overflow never does, in fact, occur.  Therefore this warning can
easily give a false positive: a warning about code which is not
actually a problem.  To help focus on important issues, several
warning levels are defined.  No warnings are issued for the use of
undefined signed overflow when estimating how many iterations a loop
will require, in particular when determining whether a loop will be
executed at all.

@table @gcctabopt
@item -Wstrict-overflow=1
Warn about cases which are both questionable and easy to avoid.  For
example: @code{x + 1 > x}; with @option{-fstrict-overflow}, the
compiler will simplify this to @code{1}.  This level of
@option{-Wstrict-overflow} is enabled by @option{-Wall}; higher levels
are not, and must be explicitly requested.

@item -Wstrict-overflow=2
Also warn about other cases where a comparison is simplified to a
constant.  For example: @code{abs (x) >= 0}.  This can only be
simplified when @option{-fstrict-overflow} is in effect, because
@code{abs (INT_MIN)} overflows to @code{INT_MIN}, which is less than
zero.  @option{-Wstrict-overflow} (with no level) is the same as
@option{-Wstrict-overflow=2}.

@item -Wstrict-overflow=3
Also warn about other cases where a comparison is simplified.  For
example: @code{x + 1 > 1} will be simplified to @code{x > 0}.

@item -Wstrict-overflow=4
Also warn about other simplifications not covered by the above cases.
For example: @code{(x * 10) / 5} will be simplified to @code{x * 2}.

@item -Wstrict-overflow=5
Also warn about cases where the compiler reduces the magnitude of a
constant involved in a comparison.  For example: @code{x + 2 > y} will
be simplified to @code{x + 1 >= y}.  This is reported only at the
highest warning level because this simplification applies to many
comparisons, so this warning level will give a very large number of
false positives.
@end table

@item -Warray-bounds
@opindex Wno-array-bounds
@opindex Warray-bounds
This option is only active when @option{-ftree-vrp} is active
(default for -O2 and above). It warns about subscripts to arrays
that are always out of bounds. This warning is enabled by @option{-Wall}.

@item -Wno-div-by-zero
@opindex Wno-div-by-zero
@opindex Wdiv-by-zero
Do not warn about compile-time integer division by zero.  Floating point
division by zero is not warned about, as it can be a legitimate way of
```

```
3457 obtaining infinities and NaNs.

3459 @item -Wsystem-headers
3460 @opindex Wsystem-headers
3461 @opindex Wno-system-headers
3462 @cindex warnings from system headers
3463 @cindex system headers, warnings from
3464 Print warning messages for constructs found in system header files.
3465 Warnings from system headers are normally suppressed, on the assumption
3466 that they usually do not indicate real problems and would only make the
3467 compiler output harder to read.  Using this command line option tells
3468 GCC to emit warnings from system headers as if they occurred in user
3469 code.  However, note that using @option{-Wall} in conjunction with this
3470 option will @emph{not} warn about unknown pragmas in system
3471 headers---for that, @option{-Wunknown-pragmas} must also be used.

3473 @item -Wfloat-equal
3474 @opindex Wfloat-equal
3475 @opindex Wno-float-equal
3476 Warn if floating point values are used in equality comparisons.

3478 The idea behind this is that sometimes it is convenient (for the
3479 programmer) to consider floating-point values as approximations to
3480 infinitely precise real numbers.  If you are doing this, then you need
3481 to compute (by analyzing the code, or in some other way) the maximum or
3482 likely maximum error that the computation introduces, and allow for it
3483 when performing comparisons (and when producing output, but that's a
3484 different problem).  In particular, instead of testing for equality, you
3485 would check to see whether the two values have ranges that overlap; and
3486 this is done with the relational operators, so equality comparisons are
3487 probably mistaken.

3489 @item -Wtraditional @r{(C and Objective-C only)}
3490 @opindex Wtraditional
3491 @opindex Wno-traditional
3492 Warn about certain constructs that behave differently in traditional and
3493 ISO C@.  Also warn about ISO C constructs that have no traditional C
3494 equivalent, and/or problematic constructs which should be avoided.

3496 @itemize @bullet
3497 @item
3498 Macro parameters that appear within string literals in the macro body.
3499 In traditional C macro replacement takes place within string literals,
3500 but does not in ISO C@.

3502 @item
3503 In traditional C, some preprocessor directives did not exist.
3504 Traditional preprocessors would only consider a line to be a directive
3505 if the @samp{#} appeared in column 1 on the line.  Therefore
3506 @option{-Wtraditional} warns about directives that traditional C
3507 understands but would ignore because the @samp{#} does not appear as the
3508 first character on the line.  It also suggests you hide directives like
3509 @samp{#pragma} not understood by traditional C by indenting them.  Some
3510 traditional implementations would not recognize @samp{#elif}, so it
3511 suggests avoiding it altogether.

3513 @item
3514 A function-like macro that appears without arguments.

3516 @item
3517 The unary plus operator.

3519 @item
3520 The @samp{U} integer constant suffix, or the @samp{F} or @samp{L} floating point
3521 constant suffixes.  (Traditional C does support the @samp{L} suffix on integer
3522 constants.)  Note, these suffixes appear in macros defined in the system
```



```
3523 headers of most modern systems, e.g.@: the @samp{_MIN}/@samp{_MAX} macros in @co
3524 Use of these macros in user code might normally lead to spurious
3525 warnings, however GCC's integrated preprocessor has enough context to
3526 avoid warning in these cases.

3528 @item
3529 A function declared external in one block and then used after the end of
3530 the block.

3532 @item
3533 A @code{switch} statement has an operand of type @code{long}.

3535 @item
3536 A non-@code{static} function declaration follows a @code{static} one.
3537 This construct is not accepted by some traditional C compilers.

3539 @item
3540 The ISO type of an integer constant has a different width or
3541 signedness from its traditional type.  This warning is only issued if
3542 the base of the constant is ten.  I.e.@: hexadecimal or octal values, which
3543 typically represent bit patterns, are not warned about.

3545 @item
3546 Usage of ISO string concatenation is detected.

3548 @item
3549 Initialization of automatic aggregates.

3551 @item
3552 Identifier conflicts with labels.  Traditional C lacks a separate
3553 namespace for labels.

3555 @item
3556 Initialization of unions.  If the initializer is zero, the warning is
3557 omitted.  This is done under the assumption that the zero initializer in
3558 user code appears conditioned on e.g.@: @code{__STDC__} to avoid missing
3559 initializer warnings and relies on default initialization to zero in the
3560 traditional C case.

3562 @item
3563 Conversions by prototypes between fixed/floating point values and vice
3564 versa.  The absence of these prototypes when compiling with traditional
3565 C would cause serious problems.  This is a subset of the possible
3566 conversion warnings, for the full set use @option{-Wtraditional-conversion}.

3568 @item
3569 Use of ISO C style function definitions.  This warning intentionally is
3570 @emph{not} issued for prototype declarations or variadic functions
3571 because these ISO C features will appear in your code when using
3572 libiberty's traditional C compatibility macros, @code{PARAMS} and
3573 @code{VPARAMS}.  This warning is also bypassed for nested functions
3574 because that feature is already a GCC extension and thus not relevant to
3575 traditional C compatibility.
3576 @end itemize

3578 @item -Wtraditional-conversion @r{(C and Objective-C only)}
3579 @opindex Wtraditional-conversion
3580 @opindex Wno-traditional-conversion
3581 Warn if a prototype causes a type conversion that is different from what
3582 would happen to the same argument in the absence of a prototype.  This
3583 includes conversions of fixed point to floating and vice versa, and
3584 conversions changing the width or signedness of a fixed point argument
3585 except when the same as the default promotion.

3587 @item -Wdeclaration-after-statement @r{(C and Objective-C only)}
3588 @opindex Wdeclaration-after-statement
```

```
3589 @opindex Wno-declaration-after-statement
3590 Warn when a declaration is found after a statement in a block.  This
3591 construct, known from C++, was introduced with ISO C99 and is by default
3592 allowed in GCC@.  It is not supported by ISO C90 and was not supported by
3593 GCC versions before GCC 3.0.  @xref{Mixed Declarations}.

3595 @item -Wundef
3596 @opindex Wundef
3597 @opindex Wno-undef
3598 Warn if an undefined identifier is evaluated in an @samp{#if} directive.

3600 @item -Wno-endif-labels
3601 @opindex Wno-endif-labels
3602 @opindex Wendif-labels
3603 Do not warn whenever an @samp{#else} or an @samp{#endif} are followed by text.

3605 @item -Wshadow
3606 @opindex Wshadow
3607 @opindex Wno-shadow
3608 Warn whenever a local variable shadows another local variable, parameter or
3609 global variable or whenever a built-in function is shadowed.

3611 @item -Wlarger-than=@var{len}
3612 @opindex Wlarger-than=@var{len}
3613 @opindex Wlarger-than-@var{len}
3614 Warn whenever an object of larger than @var{len} bytes is defined.

3616 @item -Wframe-larger-than=@var{len}
3617 @opindex Wframe-larger-than
3618 Warn if the size of a function frame is larger than @var{len} bytes.
3619 The computation done to determine the stack frame size is approximate
3620 and not conservative.
3621 The actual requirements may be somewhat greater than @var{len}
3622 even if you do not get a warning.  In addition, any space allocated
3623 via @code{alloca}, variable-length arrays, or related constructs
3624 is not included by the compiler when determining
3625 whether or not to issue a warning.

3627 @item -Wunsafe-loop-optimizations
3628 @opindex Wunsafe-loop-optimizations
3629 @opindex Wno-unsafe-loop-optimizations
3630 Warn if the loop cannot be optimized because the compiler could not
3631 assume anything on the bounds of the loop indices.  With
3632 @option{-funsafe-loop-optimizations} warn if the compiler made
3633 such assumptions.

3635 @item -Wno-pedantic-ms-format @r{(MinGW targets only)}
3636 @opindex Wno-pedantic-ms-format
3637 @opindex Wpedantic-ms-format
3638 Disables the warnings about non-ISO @code{printf} / @code{scanf} format
3639 width specifiers @code{I32}, @code{I64}, and @code{I} used on Windows targets
3640 depending on the MS runtime, when you are using the options @option{-Wformat}
3641 and @option{-pedantic} without gnu-extensions.

3643 @item -Wpointer-arith
3644 @opindex Wpointer-arith
3645 @opindex Wno-pointer-arith
3646 Warn about anything that depends on the ``size of'' a function type or
3647 of @code{void}.  GNU C assigns these types a size of 1, for
3648 convenience in calculations with @code{void *} pointers and pointers
3649 to functions.  In C++, warn also when an arithmetic operation involves
3650 @code{NULL}.  This warning is also enabled by @option{-pedantic}.

3652 @item -Wtype-limits
3653 @opindex Wtype-limits
3654 @opindex Wno-type-limits
```



```
3655 Warn if a comparison is always true or always false due to the limited
3656 range of the data type, but do not warn for constant expressions.  For
3657 example, warn if an unsigned variable is compared against zero with
3658 @samp{<} or @samp{>=}.  This warning is also enabled by
3659 @option{-Wextra}.

3661 @item -Wbad-function-cast @r{(C and Objective-C only)}
3662 @opindex Wbad-function-cast
3663 @opindex Wno-bad-function-cast
3664 Warn whenever a function call is cast to a non-matching type.
3665 For example, warn if @code{int malloc()} is cast to @code{anything *}.

3667 @item -Wc++-compat @r{(C and Objective-C only)}
3668 Warn about ISO C constructs that are outside of the common subset of
3669 ISO C and ISO C++, e.g.@: request for implicit conversion from
3670 @code{void *} to a pointer to non-@code{void} type.

3672 @item -Wc++0x-compat @r{(C++ and Objective-C++ only)}
3673 Warn about C++ constructs whose meaning differs between ISO C++ 1998 and
3674 ISO C++ 200x, e.g., identifiers in ISO C++ 1998 that will become keywords
3675 in ISO C++ 200x.  This warning is enabled by @option{-Wall}.

3677 @item -Wcast-qual
3678 @opindex Wcast-qual
3679 @opindex Wno-cast-qual
3680 Warn whenever a pointer is cast so as to remove a type qualifier from
3681 the target type.  For example, warn if a @code{const char *} is cast
3682 to an ordinary @code{char *}.

3684 @item -Wcast-align
3685 @opindex Wcast-align
3686 @opindex Wno-cast-align
3687 Warn whenever a pointer is cast such that the required alignment of the
3688 target is increased.  For example, warn if a @code{char *} is cast to
3689 an @code{int *} on machines where integers can only be accessed at
3690 two- or four-byte boundaries.

3692 @item -Wwrite-strings
3693 @opindex Wwrite-strings
3694 @opindex Wno-write-strings
3695 When compiling C, give string constants the type @code{const
3696 char[@var{length}]} so that copying the address of one into a
3697 non-@code{const} @code{char *} pointer will get a warning.  These
3698 warnings will help you find at compile time code that can try to write
3699 into a string constant, but only if you have been very careful about
3700 using @code{const} in declarations and prototypes.  Otherwise, it will
3701 just be a nuisance. This is why we did not make @option{-Wall} request
3702 these warnings.

3704 When compiling C++, warn about the deprecated conversion from string
3705 literals to @code{char *}.  This warning is enabled by default for C++
3706 programs.

3708 @item -Wclobbered
3709 @opindex Wclobbered
3710 @opindex Wno-clobbered
3711 Warn for variables that might be changed by @samp{longjmp} or
3712 @samp{vfork}.  This warning is also enabled by @option{-Wextra}.

3714 @item -Wconversion
3715 @opindex Wconversion
3716 @opindex Wno-conversion
3717 Warn for implicit conversions that may alter a value. This includes
3718 conversions between real and integer, like @code{abs (x)} when
3719 @code{x} is @code{double}; conversions between signed and unsigned,
3720 like @code{unsigned ui = -1}; and conversions to smaller types, like
```

```
@code{sqrtf (M_PI)}. Do not warn for explicit casts like @code{abs
((int) x)} and @code{ui = (unsigned) -1}, or if the value is not
changed by the conversion like in @code{abs (2.0)}.  Warnings about
conversions between signed and unsigned integers can be disabled by
using @option{-Wno-sign-conversion}.

For C++, also warn for conversions between @code{NULL} and non-pointer
types; confusing overload resolution for user-defined conversions; and
conversions that will never use a type conversion operator:
conversions to @code{void}, the same type, a base class or a reference
to them. Warnings about conversions between signed and unsigned
integers are disabled by default in C++ unless
@option{-Wsign-conversion} is explicitly enabled.

@item -Wempty-body
@opindex Wempty-body
@opindex Wno-empty-body
Warn if an empty body occurs in an @samp{if}, @samp{else} or @samp{do
while} statement.  This warning is also enabled by @option{-Wextra}.

@item -Wenum-compare @r{(C++ and Objective-C++ only)}
@opindex Wenum-compare
@opindex Wno-enum-compare
Warn about a comparison between values of different enum types. This
warning is enabled by default.

@item -Wsign-compare
@opindex Wsign-compare
@opindex Wno-sign-compare
@cindex warning for comparison of signed and unsigned values
@cindex comparison of signed and unsigned values, warning
@cindex signed and unsigned values, comparison warning
Warn when a comparison between signed and unsigned values could produce
an incorrect result when the signed value is converted to unsigned.
This warning is also enabled by @option{-Wextra}; to get the other warnings
of @option{-Wextra} without this warning, use @samp{-Wextra -Wno-sign-compare}.

@item -Wsign-conversion
@opindex Wsign-conversion
@opindex Wno-sign-conversion
Warn for implicit conversions that may change the sign of an integer
value, like assigning a signed integer expression to an unsigned
integer variable. An explicit cast silences the warning. In C, this
option is enabled also by @option{-Wconversion}.

@item -Waddress
@opindex Waddress
@opindex Wno-address
Warn about suspicious uses of memory addresses. These include using
the address of a function in a conditional expression, such as
@code{void func(void); if (func)}, and comparisons against the memory
address of a string literal, such as @code{if (x == "abc")}.  Such
uses typically indicate a programmer error: the address of a function
always evaluates to true, so their use in a conditional usually
indicate that the programmer forgot the parentheses in a function
call; and comparisons against string literals result in unspecified
behavior and are not portable in C, so they usually indicate that the
programmer intended to use @code{strcmp}.  This warning is enabled by
@option{-Wall}.

@item -Wlogical-op
@opindex Wlogical-op
@opindex Wno-logical-op
Warn about suspicious uses of logical operators in expressions.
This includes using logical operators in contexts where a
bit-wise operator is likely to be expected.
```



```
@item -Waggregate-return
@opindex Waggregate-return
@opindex Wno-aggregate-return
Warn if any functions that return structures or unions are defined or
called.  (In languages where you can return an array, this also elicits
a warning.)

@item -Wno-attributes
@opindex Wno-attributes
@opindex Wattributes
Do not warn if an unexpected @code{__attribute__} is used, such as
unrecognized attributes, function attributes applied to variables,
etc.  This will not stop errors for incorrect use of supported
attributes.

@item -Wno-builtin-macro-redefined
@opindex Wno-builtin-macro-redefined
@opindex Wbuiltin-macro-redefined
Do not warn if certain built-in macros are redefined.  This suppresses
warnings for redefinition of @code{__TIMESTAMP__}, @code{__TIME__},
@code{__DATE__}, @code{__FILE__}, and @code{__BASE_FILE__}.

@item -Wstrict-prototypes @r{(C and Objective-C only)}
@opindex Wstrict-prototypes
@opindex Wno-strict-prototypes
Warn if a function is declared or defined without specifying the
argument types.  (An old-style function definition is permitted without
a warning if preceded by a declaration which specifies the argument
types.)

@item -Wold-style-declaration @r{(C and Objective-C only)}
@opindex Wold-style-declaration
@opindex Wno-old-style-declaration
Warn for obsolescent usages, according to the C Standard, in a
declaration. For example, warn if storage-class specifiers like
@code{static} are not the first things in a declaration.  This warning
is also enabled by @option{-Wextra}.

@item -Wold-style-definition @r{(C and Objective-C only)}
@opindex Wold-style-definition
@opindex Wno-old-style-definition
Warn if an old-style function definition is used.  A warning is given
even if there is a previous prototype.

@item -Wmissing-parameter-type @r{(C and Objective-C only)}
@opindex Wmissing-parameter-type
@opindex Wno-missing-parameter-type
A function parameter is declared without a type specifier in K&R-style
functions:

@smallexample
void foo(bar) @{ @}
@end smallexample

This warning is also enabled by @option{-Wextra}.

@item -Wmissing-prototypes @r{(C and Objective-C only)}
@opindex Wmissing-prototypes
@opindex Wno-missing-prototypes
Warn if a global function is defined without a previous prototype
declaration.  This warning is issued even if the definition itself
provides a prototype.  The aim is to detect global functions that fail
to be declared in header files.

@item -Wmissing-declarations
```

```
@opindex Wmissing-declarations
@opindex Wno-missing-declarations
Warn if a global function is defined without a previous declaration.
Do so even if the definition itself provides a prototype.
Use this option to detect global functions that are not declared in
header files.  In C++, no warnings are issued for function templates,
or for inline functions, or for functions in anonymous namespaces.

@item -Wmissing-field-initializers
@opindex Wmissing-field-initializers
@opindex Wno-missing-field-initializers
@opindex W
@opindex Wextra
@opindex Wno-extra
Warn if a structure's initializer has some fields missing.  For
example, the following code would cause such a warning, because
@code{x.h} is implicitly zero:

@smallexample
struct s @{ int f, g, h; @};
struct s x = @{ 3, 4 @};
@end smallexample

This option does not warn about designated initializers, so the following
modification would not trigger a warning:

@smallexample
struct s @{ int f, g, h; @};
struct s x = @{ .f = 3, .g = 4 @};
@end smallexample

This warning is included in @option{-Wextra}.  To get other @option{-Wextra}
warnings without this one, use @samp{-Wextra -Wno-missing-field-initializers}.

@item -Wmissing-noreturn
@opindex Wmissing-noreturn
@opindex Wno-missing-noreturn
Warn about functions which might be candidates for attribute @code{noreturn}.
Note these are only possible candidates, not absolute ones.  Care should
be taken to manually verify functions actually do not ever return before
adding the @code{noreturn} attribute, otherwise subtle code generation
bugs could be introduced.  You will not get a warning for @code{main} in
hosted C environments.

@item -Wmissing-format-attribute
@opindex Wmissing-format-attribute
@opindex Wno-missing-format-attribute
@opindex Wformat
@opindex Wno-format
Warn about function pointers which might be candidates for @code{format}
attributes.  Note these are only possible candidates, not absolute ones.
GCC will guess that function pointers with @code{format} attributes that
are used in assignment, initialization, parameter passing or return
statements should have a corresponding @code{format} attribute in the
resulting type.  I.e.@: the left-hand side of the assignment or
initialization, the type of the parameter variable, or the return type
of the containing function respectively should also have a @code{format}
attribute to avoid the warning.

GCC will also warn about function definitions which might be
candidates for @code{format} attributes.  Again, these are only
possible candidates.  GCC will guess that @code{format} attributes
might be appropriate for any function that calls a function like
@code{vprintf} or @code{vscanf}, but this might not always be the
case, and some functions for which @code{format} attributes are
appropriate may not be detected.
```



```
@item -Wno-multichar
@opindex Wno-multichar
@opindex Wmultichar
Do not warn if a multicharacter constant (@samp{'FOOF'}) is used.
Usually they indicate a typo in the user's code, as they have
implementation-defined values, and should not be used in portable code.

@item -Wnormalized=<none|id|nfc|nfkc>
@opindex Wnormalized=
@cindex NFC
@cindex NFKC
@cindex character set, input normalization
In ISO C and ISO C++, two identifiers are different if they are
different sequences of characters.  However, sometimes when characters
outside the basic ASCII character set are used, you can have two
different character sequences that look the same.  To avoid confusion,
the ISO 10646 standard sets out some @dfn{normalization rules} which
when applied ensure that two sequences that look the same are turned into
the same sequence.  GCC can warn you if you are using identifiers which
have not been normalized; this option controls that warning.

There are four levels of warning that GCC supports.  The default is
@option{-Wnormalized=nfc}, which warns about any identifier which is
not in the ISO 10646 ''C'' normalized form, @dfn{NFC}.  NFC is the
recommended form for most uses.

Unfortunately, there are some characters which ISO C and ISO C++ allow
in identifiers that when turned into NFC aren't allowable as
identifiers.  That is, there's no way to use these symbols in portable
ISO C or C++ and have all your identifiers in NFC@.
@option{-Wnormalized=id} suppresses the warning for these characters.
It is hoped that future versions of the standards involved will correct
this, which is why this option is not the default.

You can switch the warning off for all characters by writing
@option{-Wnormalized=none}.  You would only want to do this if you
were using some other normalization scheme (like ''D''), because
otherwise you can easily create bugs that are literally impossible to see.

Some characters in ISO 10646 have distinct meanings but look identical
in some fonts or display methodologies, especially once formatting has
been applied.  For instance @code{\u207F}, ''SUPERSCRIPT LATIN SMALL
LETTER N'', will display just like a regular @code{n} which has been
placed in a superscript.  ISO 10646 defines the @dfn{NFKC}
normalization scheme to convert all these into a standard form as
well, and GCC will warn if your code is not in NFKC if you use
@option{-Wnormalized=nfkc}.  This warning is comparable to warning
about every identifier that contains the letter O because it might be
confused with the digit 0, and so is not the default, but may be
useful as a local coding convention if the programming environment is
unable to be fixed to display these characters distinctly.

@item -Wno-deprecated
@opindex Wno-deprecated
@opindex Wdeprecated
Do not warn about usage of deprecated features.  @xref{Deprecated Features}.

@item -Wno-deprecated-declarations
@opindex Wno-deprecated-declarations
@opindex Wdeprecated-declarations
Do not warn about uses of functions (@pxref{Function Attributes}),
variables (@pxref{Variable Attributes}), and types (@pxref{Type
Attributes}) marked as deprecated by using the @code{deprecated}
attribute.
```

```
@item -Wno-overflow
@opindex Wno-overflow
@opindex Woverflow
Do not warn about compile-time overflow in constant expressions.

@item -Woverride-init @r{(C and Objective-C only)}
@opindex Woverride-init
@opindex Wno-override-init
@opindex W
@opindex Wextra
@opindex Wno-extra
Warn if an initialized field without side effects is overridden when
using designated initializers (@pxref{Designated Inits, , Designated
Initializers}).

This warning is included in @option{-Wextra}.  To get other
@option{-Wextra} warnings without this one, use @samp{-Wextra
-Wno-override-init}.

@item -Wpacked
@opindex Wpacked
@opindex Wno-packed
Warn if a structure is given the packed attribute, but the packed
attribute has no effect on the layout or size of the structure.
Such structures may be mis-aligned for little benefit.  For
instance, in this code, the variable @code{f.x} in @code{struct bar}
will be misaligned even though @code{struct bar} does not itself
have the packed attribute:

@smallexample
@group
struct foo @{
  int x;
  char a, b, c, d;
@} __attribute__((packed));
struct bar @{
  char z;
  struct foo f;
@};
@end group
@end smallexample

@item -Wpacked-bitfield-compat
@opindex Wpacked-bitfield-compat
@opindex Wno-packed-bitfield-compat
The 4.1, 4.2 and 4.3 series of GCC ignore the @code{packed} attribute
on bit-fields of type @code{char}.  This has been fixed in GCC 4.4 but
the change can lead to differences in the structure layout.  GCC
informs you when the offset of such a field has changed in GCC 4.4.
For example there is no longer a 4-bit padding between field @code{a}
and @code{b} in this structure:

@smallexample
struct foo
@{
  char a:4;
  char b:8;
@} __attribute__ ((packed));
@end smallexample

This warning is enabled by default.  Use
@option{-Wno-packed-bitfield-compat} to disable this warning.

@item -Wpadded
@opindex Wpadded
@opindex Wno-padded
```



```
Warn if padding is included in a structure, either to align an element
of the structure or to align the whole structure.  Sometimes when this
happens it is possible to rearrange the fields of the structure to
reduce the padding and so make the structure smaller.

@item -Wredundant-decls
@opindex Wredundant-decls
@opindex Wno-redundant-decls
Warn if anything is declared more than once in the same scope, even in
cases where multiple declaration is valid and changes nothing.

@item -Wnested-externs @r{(C and Objective-C only)}
@opindex Wnested-externs
@opindex Wno-nested-externs
Warn if an @code{extern} declaration is encountered within a function.

@item -Wunreachable-code
@opindex Wunreachable-code
@opindex Wno-unreachable-code
Warn if the compiler detects that code will never be executed.

This option is intended to warn when the compiler detects that at
least a whole line of source code will never be executed, because
some condition is never satisfied or because it is after a
procedure that never returns.

It is possible for this option to produce a warning even though there
are circumstances under which part of the affected line can be executed,
so care should be taken when removing apparently-unreachable code.

For instance, when a function is inlined, a warning may mean that the
line is unreachable in only one inlined copy of the function.

This option is not made part of @option{-Wall} because in a debugging
version of a program there is often substantial code which checks
correct functioning of the program and is, hopefully, unreachable
because the program does work.  Another common use of unreachable
code is to provide behavior which is selectable at compile-time.

@item -Winline
@opindex Winline
@opindex Wno-inline
Warn if a function can not be inlined and it was declared as inline.
Even with this option, the compiler will not warn about failures to
inline functions declared in system headers.

The compiler uses a variety of heuristics to determine whether or not
to inline a function.  For example, the compiler takes into account
the size of the function being inlined and the amount of inlining
that has already been done in the current function.  Therefore,
seemingly insignificant changes in the source program can cause the
warnings produced by @option{-Winline} to appear or disappear.

@item -Wno-invalid-offsetof @r{(C++ and Objective-C++ only)}
@opindex Wno-invalid-offsetof
@opindex Winvalid-offsetof
Suppress warnings from applying the @samp{offsetof} macro to a non-POD
type.  According to the 1998 ISO C++ standard, applying @samp{offsetof}
to a non-POD type is undefined.  In existing C++ implementations,
however, @samp{offsetof} typically gives meaningful results even when
applied to certain kinds of non-POD types. (Such as a simple
@samp{struct} that fails to be a POD type only by virtue of having a
constructor.)  This flag is for users who are aware that they are
writing nonportable code and who have deliberately chosen to ignore the
warning about it.
```

```
4117 The restrictions on @samp{offsetof} may be relaxed in a future version
4118 of the C++ standard.

4120 @item -Wno-int-to-pointer-cast @r{(C and Objective-C only)}
4121 @opindex Wno-int-to-pointer-cast
4122 @opindex Wint-to-pointer-cast
4123 Suppress warnings from casts to pointer type of an integer of a
4124 different size.

4126 @item -Wno-pointer-to-int-cast @r{(C and Objective-C only)}
4127 @opindex Wno-pointer-to-int-cast
4128 @opindex Wpointer-to-int-cast
4129 Suppress warnings from casts from a pointer to an integer type of a
4130 different size.

4132 @item -Winvalid-pch
4133 @opindex Winvalid-pch
4134 @opindex Wno-invalid-pch
4135 Warn if a precompiled header (@pxref{Precompiled Headers}) is found in
4136 the search path but can't be used.

4138 @item -Wlong-long
4139 @opindex Wlong-long
4140 @opindex Wno-long-long
4141 Warn if @samp{long long} type is used.  This is default.  To inhibit
4142 the warning messages, use @option{-Wno-long-long}.  Flags
4143 @option{-Wlong-long} and @option{-Wno-long-long} are taken into account
4144 only when @option{-pedantic} flag is used.

4146 @item -Wvariadic-macros
4147 @opindex Wvariadic-macros
4148 @opindex Wno-variadic-macros
4149 Warn if variadic macros are used in pedantic ISO C90 mode, or the GNU
4150 alternate syntax when in pedantic ISO C99 mode.  This is default.
4151 To inhibit the warning messages, use @option{-Wno-variadic-macros}.

4153 @item -Wvla
4154 @opindex Wvla
4155 @opindex Wno-vla
4156 Warn if variable length array is used in the code.
4157 @option{-Wno-vla} will prevent the @option{-pedantic} warning of
4158 the variable length array.

4160 @item -Wvolatile-register-var
4161 @opindex Wvolatile-register-var
4162 @opindex Wno-volatile-register-var
4163 Warn if a register variable is declared volatile.  The volatile
4164 modifier does not inhibit all optimizations that may eliminate reads
4165 and/or writes to register variables.  This warning is enabled by
4166 @option{-Wall}.

4168 @item -Wdisabled-optimization
4169 @opindex Wdisabled-optimization
4170 @opindex Wno-disabled-optimization
4171 Warn if a requested optimization pass is disabled.  This warning does
4172 not generally indicate that there is anything wrong with your code; it
4173 merely indicates that GCC's optimizers were unable to handle the code
4174 effectively.  Often, the problem is that your code is too big or too
4175 complex; GCC will refuse to optimize programs when the optimization
4176 itself is likely to take inordinate amounts of time.

4178 @item -Wpointer-sign @r{(C and Objective-C only)}
4179 @opindex Wpointer-sign
4180 @opindex Wno-pointer-sign
4181 Warn for pointer argument passing or assignment with different signedness.
4182 This option is only supported for C and Objective-C@.  It is implied by
```



```
4183 @option{-Wall} and by @option{-pedantic}, which can be disabled with
4184 @option{-Wno-pointer-sign}.

4186 @item -Wstack-protector
4187 @opindex Wstack-protector
4188 @opindex Wno-stack-protector
4189 This option is only active when @option{-fstack-protector} is active.  It
4190 warns about functions that will not be protected against stack smashing.

4192 @item -Wno-mudflap
4193 @opindex Wno-mudflap
4194 Suppress warnings about constructs that cannot be instrumented by
4195 @option{-fmudflap}.

4197 @item -Woverlength-strings
4198 @opindex Woverlength-strings
4199 @opindex Wno-overlength-strings
4200 Warn about string constants which are longer than the ''minimum
4201 maximum'' length specified in the C standard.  Modern compilers
4202 generally allow string constants which are much longer than the
4203 standard's minimum limit, but very portable programs should avoid
4204 using longer strings.

4206 The limit applies @emph{after} string constant concatenation, and does
4207 not count the trailing NUL@.  In C89, the limit was 509 characters; in
4208 C99, it was raised to 4095.  C++98 does not specify a normative
4209 minimum maximum, so we do not diagnose overlength strings in C++@.

4211 This option is implied by @option{-pedantic}, and can be disabled with
4212 @option{-Wno-overlength-strings}.
4213 @end table

4215 @node Debugging Options
4216 @section Options for Debugging Your Program or GCC
4217 @cindex options, debugging
4218 @cindex debugging information options

4220 GCC has various special options that are used for debugging
4221 either your program or GCC:

4223 @table @gcctabopt
4224 @item -g
4225 @opindex g
4226 Produce debugging information in the operating system's native format
4227 (stabs, COFF, XCOFF, or DWARF 2)@.  GDB can work with this debugging
4228 information.

4230 On most systems that use stabs format, @option{-g} enables use of extra
4231 debugging information that only GDB can use; this extra information
4232 makes debugging work better in GDB but will probably make other debuggers
4233 crash or
4234 refuse to read the program.  If you want to control for certain whether
4235 to generate the extra information, use @option{-gstabs+}, @option{-gstabs},
4236 @option{-gxcoff+}, @option{-gxcoff}, or @option{-gvms} (see below).

4238 GCC allows you to use @option{-g} with
4239 @option{-O}.  The shortcuts taken by optimized code may occasionally
4240 produce surprising results: some variables you declared may not exist
4241 at all; flow of control may briefly move where you did not expect it;
4242 some statements may not be executed because they compute constant
4243 results or their values were already at hand; some statements may
4244 execute in different places because they were moved out of loops.

4246 Nevertheless it proves possible to debug optimized output.  This makes
4247 it reasonable to use the optimizer for programs that might have bugs.
```

```
4249 The following options are useful when GCC is generated with the
4250 capability for more than one debugging format.

4252 @item -ggdb
4253 @opindex ggdb
4254 Produce debugging information for use by GDB@.  This means to use the
4255 most expressive format available (DWARF 2, stabs, or the native format
4256 if neither of those are supported), including GDB extensions if at all
4257 possible.

4259 @item -gstabs
4260 @opindex gstabs
4261 Produce debugging information in stabs format (if that is supported),
4262 without GDB extensions.  This is the format used by DBX on most BSD
4263 systems.  On MIPS, Alpha and System V Release 4 systems this option
4264 produces stabs debugging output which is not understood by DBX or SDB@.
4265 On System V Release 4 systems this option requires the GNU assembler.

4267 @item -feliminate-unused-debug-symbols
4268 @opindex feliminate-unused-debug-symbols
4269 Produce debugging information in stabs format (if that is supported),
4270 for only symbols that are actually used.

4272 @item -femit-class-debug-always
4273 Instead of emitting debugging information for a C++ class in only one
4274 object file, emit it in all object files using the class.  This option
4275 should be used only with debuggers that are unable to handle the way GCC
4276 normally emits debugging information for classes because using this
4277 option will increase the size of debugging information by as much as a
4278 factor of two.

4280 @item -gstabs+
4281 @opindex gstabs+
4282 Produce debugging information in stabs format (if that is supported),
4283 using GNU extensions understood only by the GNU debugger (GDB)@.  The
4284 use of these extensions is likely to make other debuggers crash or
4285 refuse to read the program.

4287 @item -gcoff
4288 @opindex gcoff
4289 Produce debugging information in COFF format (if that is supported).
4290 This is the format used by SDB on most System V systems prior to
4291 System V Release 4.

4293 @item -gxcoff
4294 @opindex gxcoff
4295 Produce debugging information in XCOFF format (if that is supported).
4296 This is the format used by the DBX debugger on IBM RS/6000 systems.

4298 @item -gxcoff+
4299 @opindex gxcoff+
4300 Produce debugging information in XCOFF format (if that is supported),
4301 using GNU extensions understood only by the GNU debugger (GDB)@.  The
4302 use of these extensions is likely to make other debuggers crash or
4303 refuse to read the program, and may cause assemblers other than the GNU
4304 assembler (GAS) to fail with an error.

4306 @item -gdwarf-2
4307 @opindex gdwarf-2
4308 Produce debugging information in DWARF version 2 format (if that is
4309 supported).  This is the format used by DBX on IRIX 6.  With this
4310 option, GCC uses features of DWARF version 3 when they are useful;
4311 version 3 is upward compatible with version 2, but may still cause
4312 problems for older debuggers.

4314 @item -gvms
```



```
4315 @opindex gvms
4316 Produce debugging information in VMS debug format (if that is
4317 supported).  This is the format used by DEBUG on VMS systems.

4319 @item -g@var{level}
4320 @itemx -ggdb@var{level}
4321 @itemx -gstabs@var{level}
4322 @itemx -gcoff@var{level}
4323 @itemx -gxcoff@var{level}
4324 @itemx -gvms@var{level}
4325 Request debugging information and also use @var{level} to specify how
4326 much information.  The default level is 2.

4328 Level 0 produces no debug information at all.  Thus, @option{-g0} negates
4329 @option{-g}.

4331 Level 1 produces minimal information, enough for making backtraces in
4332 parts of the program that you don't plan to debug.  This includes
4333 descriptions of functions and external variables, but no information
4334 about local variables and no line numbers.

4336 Level 3 includes extra information, such as all the macro definitions
4337 present in the program.  Some debuggers support macro expansion when
4338 you use @option{-g3}.

4340 @option{-gdwarf-2} does not accept a concatenated debug level, because
4341 GCC used to support an option @option{-gdwarf} that meant to generate
4342 debug information in version 1 of the DWARF format (which is very
4343 different from version 2), and it would have been too confusing.  That
4344 debug format is long obsolete, but the option cannot be changed now.
4345 Instead use an additional @option{-g@var{level}} option to change the
4346 debug level for DWARF2.

4348 @item -feliminate-dwarf2-dups
4349 @opindex feliminate-dwarf2-dups
4350 Compress DWARF2 debugging information by eliminating duplicated
4351 information about each symbol.  This option only makes sense when
4352 generating DWARF2 debugging information with @option{-gdwarf-2}.

4354 @item -femit-struct-debug-baseonly
4355 Emit debug information for struct-like types
4356 only when the base name of the compilation source file
4357 matches the base name of file in which the struct was defined.

4359 This option substantially reduces the size of debugging information,
4360 but at significant potential loss in type information to the debugger.
4361 See @option{-femit-struct-debug-reduced} for a less aggressive option.
4362 See @option{-femit-struct-debug-detailed} for more detailed control.

4364 This option works only with DWARF 2.

4366 @item -femit-struct-debug-reduced
4367 Emit debug information for struct-like types
4368 only when the base name of the compilation source file
4369 matches the base name of file in which the type was defined,
4370 unless the struct is a template or defined in a system header.

4372 This option significantly reduces the size of debugging information,
4373 with some potential loss in type information to the debugger.
4374 See @option{-femit-struct-debug-baseonly} for a more aggressive option.
4375 See @option{-femit-struct-debug-detailed} for more detailed control.

4377 This option works only with DWARF 2.

4379 @item -femit-struct-debug-detailed@r{[}=@var{spec-list}@r{]}
4380 Specify the struct-like types
```

```
4381 for which the compiler will generate debug information.
4382 The intent is to reduce duplicate struct debug information
4383 between different object files within the same program.

4385 This option is a detailed version of
4386 @option{-femit-struct-debug-reduced} and @option{-femit-struct-debug-baseonly},
4387 which will serve for most needs.

4389 A specification has the syntax
4390 [@samp{dir:}|@samp{ind:}][@samp{ord:}|@samp{gen:}](@samp{any}|@samp{sys}|@samp{b

4392 The optional first word limits the specification to
4393 structs that are used directly (@samp{dir:}) or used indirectly (@samp{ind:}).
4394 A struct type is used directly when it is the type of a variable, member.
4395 Indirect uses arise through pointers to structs.
4396 That is, when use of an incomplete struct would be legal, the use is indirect.
4397 An example is
4398 @samp{struct one direct; struct two * indirect;}.

4400 The optional second word limits the specification to
4401 ordinary structs (@samp{ord:}) or generic structs (@samp{gen:}).
4402 Generic structs are a bit complicated to explain.
4403 For C++, these are non-explicit specializations of template classes,
4404 or non-template classes within the above.
4405 Other programming languages have generics,
4406 but @samp{-femit-struct-debug-detailed} does not yet implement them.

4408 The third word specifies the source files for those
4409 structs for which the compiler will emit debug information.
4410 The values @samp{none} and @samp{any} have the normal meaning.
4411 The value @samp{base} means that
4412 the base of name of the file in which the type declaration appears
4413 must match the base of the name of the main compilation file.
4414 In practice, this means that
4415 types declared in @file{foo.c} and @file{foo.h} will have debug information,
4416 but types declared in other header will not.
4417 The value @samp{sys} means those types satisfying @samp{base}
4418 or declared in system or compiler headers.

4420 You may need to experiment to determine the best settings for your application.

4422 The default is @samp{-femit-struct-debug-detailed=all}.

4424 This option works only with DWARF 2.

4426 @item -fno-merge-debug-strings
4427 @opindex fmerge-debug-strings
4428 @opindex fno-merge-debug-strings
4429 Direct the linker to not merge together strings in the debugging
4430 information which are identical in different object files.  Merging is
4431 not supported by all assemblers or linkers.  Merging decreases the size
4432 of the debug information in the output file at the cost of increasing
4433 link processing time.  Merging is enabled by default.

4435 @item -fdebug-prefix-map=@var{old}=@var{new}
4436 @opindex fdebug-prefix-map
4437 When compiling files in directory @file{@var{old}}, record debugging
4438 information describing them as in @file{@var{new}} instead.

4440 @item -fno-dwarf2-cfi-asm
4441 @opindex fdwarf2-cfi-asm
4442 @opindex fno-dwarf2-cfi-asm
4443 Emit DWARF 2 unwind info as compiler generated @code{.eh_frame} section
4444 instead of using GAS @code{.cfi_*} directives.

4446 @cindex @command{prof}
```



```
4447 @item -p
4448 @opindex p
4449 Generate extra code to write profile information suitable for the
4450 analysis program @command{prof}.  You must use this option when compiling
4451 the source files you want data about, and you must also use it when
4452 linking.

4454 @cindex @command{gprof}
4455 @item -pg
4456 @opindex pg
4457 Generate extra code to write profile information suitable for the
4458 analysis program @command{gprof}.  You must use this option when compiling
4459 the source files you want data about, and you must also use it when
4460 linking.

4462 @item -Q
4463 @opindex Q
4464 Makes the compiler print out each function name as it is compiled, and
4465 print some statistics about each pass when it finishes.

4467 @item -ftime-report
4468 @opindex ftime-report
4469 Makes the compiler print some statistics about the time consumed by each
4470 pass when it finishes.

4472 @item -fmem-report
4473 @opindex fmem-report
4474 Makes the compiler print some statistics about permanent memory
4475 allocation when it finishes.

4477 @item -fpre-ipa-mem-report
4478 @opindex fpre-ipa-mem-report
4479 @item -fpost-ipa-mem-report
4480 @opindex fpost-ipa-mem-report
4481 Makes the compiler print some statistics about permanent memory
4482 allocation before or after interprocedural optimization.

4484 @item -fprofile-arcs
4485 @opindex fprofile-arcs
4486 Add code so that program flow @dfn{arcs} are instrumented.  During
4487 execution the program records how many times each branch and call is
4488 executed and how many times it is taken or returns.  When the compiled
4489 program exits it saves this data to a file called
4490 @file{@var{auxname}.gcda} for each source file.  The data may be used for
4491 profile-directed optimizations (@option{-fbranch-probabilities}), or for
4492 test coverage analysis (@option{-ftest-coverage}).  Each object file's
4493 @var{auxname} is generated from the name of the output file, if
4494 explicitly specified and it is not the final executable, otherwise it is
4495 the basename of the source file.  In both cases any suffix is removed
4496 (e.g.@: @file{foo.gcda} for input file @file{dir/foo.c}, or
4497 @file{dir/foo.gcda} for output file specified as @option{-o dir/foo.o}).
4498 @xref{Cross-profiling}.

4500 @cindex @command{gcov}
4501 @item --coverage
4502 @opindex coverage

4504 This option is used to compile and link code instrumented for coverage
4505 analysis.  The option is a synonym for @option{-fprofile-arcs}
4506 @option{-ftest-coverage} (when compiling) and @option{-lgcov} (when
4507 linking).  See the documentation for those options for more details.

4509 @itemize

4511 @item
4512 Compile the source files with @option{-fprofile-arcs} plus optimization
```

```
4513 and code generation options.  For test coverage analysis, use the
4514 additional @option{-ftest-coverage} option.  You do not need to profile
4515 every source file in a program.

4517 @item
4518 Link your object files with @option{-lgcov} or @option{-fprofile-arcs}
4519 (the latter implies the former).

4521 @item
4522 Run the program on a representative workload to generate the arc profile
4523 information.  This may be repeated any number of times.  You can run
4524 concurrent instances of your program, and provided that the file system
4525 supports locking, the data files will be correctly updated.  Also
4526 @code{fork} calls are detected and correctly handled (double counting
4527 will not happen).

4529 @item
4530 For profile-directed optimizations, compile the source files again with
4531 the same optimization and code generation options plus
4532 @option{-fbranch-probabilities} (@pxref{Optimize Options,,Options that
4533 Control Optimization}).

4535 @item
4536 For test coverage analysis, use @command{gcov} to produce human readable
4537 information from the @file{.gcno} and @file{.gcda} files.  Refer to the
4538 @command{gcov} documentation for further information.

4540 @end itemize

4542 With @option{-fprofile-arcs}, for each function of your program GCC
4543 creates a program flow graph, then finds a spanning tree for the graph.
4544 Only arcs that are not on the spanning tree have to be instrumented: the
4545 compiler adds code to count the number of times that these arcs are
4546 executed.  When an arc is the only exit or only entrance to a block, the
4547 instrumentation code can be added to the block; otherwise, a new basic
4548 block must be created to hold the instrumentation code.

4550 @need 2000
4551 @item -ftest-coverage
4552 @opindex ftest-coverage
4553 Produce a notes file that the @command{gcov} code-coverage utility
4554 (@pxref{Gcov,, @command{gcov}---a Test Coverage Program}) can use to
4555 show program coverage.  Each source file's note file is called
4556 @file{@var{auxname}.gcno}.  Refer to the @option{-fprofile-arcs} option
4557 above for a description of @var{auxname} and instructions on how to
4558 generate test coverage data.  Coverage data will match the source files
4559 more closely, if you do not optimize.

4561 @item -fdbg-cnt-list
4562 @opindex fdbg-cnt-list
4563 Print the name and the counter upperbound for all debug counters.

4565 @item -fdbg-cnt=@var{counter-value-list}
4566 @opindex fdbg-cnt
4567 Set the internal debug counter upperbound. @var{counter-value-list}
4568 is a comma-separated list of @var{name}:@var{value} pairs
4569 which sets the upperbound of each debug counter @var{name} to @var{value}.
4570 All debug counters have the initial upperbound of @var{UINT_MAX},
4571 thus dbg_cnt() returns true always unless the upperbound is set by this option.
4572 e.g. With -fdbg-cnt=dce:10,tail_call:0
4573 dbg_cnt(dce) will return true only for first 10 invocations
4574 and dbg_cnt(tail_call) will return false always.

4576 @item -d@var{letters}
4577 @itemx -fdump-rtl-@var{pass}
4578 @opindex d
```



```
4579 Says to make debugging dumps during compilation at times specified by
4580 @var{letters}.    This is used for debugging the RTL-based passes of the
4581 compiler.  The file names for most of the dumps are made by appending a
4582 pass number and a word to the @var{dumpname}.  @var{dumpname} is generated
4583 from the name of the output file, if explicitly specified and it is not
4584 an executable, otherwise it is the basename of the source file. These
4585 switches may have different effects when @option{-E} is used for
4586 preprocessing.

4588 Debug dumps can be enabled with a @option{-fdump-rtl} switch or some
4589 @option{-d} option @var{letters}.  Here are the possible
4590 letters for use in @var{pass} and @var{letters}, and their meanings:

4592 @table @gcctabopt

4594 @item -fdump-rtl-alignments
4595 @opindex fdump-rtl-alignments
4596 Dump after branch alignments have been computed.

4598 @item -fdump-rtl-asmcons
4599 @opindex fdump-rtl-asmcons
4600 Dump after fixing rtl statements that have unsatisfied in/out constraints.

4602 @item -fdump-rtl-auto_inc_dec
4603 @opindex fdump-rtl-auto_inc_dec
4604 Dump after auto-inc-dec discovery.  This pass is only run on
4605 architectures that have auto inc or auto dec instructions.

4607 @item -fdump-rtl-barriers
4608 @opindex fdump-rtl-barriers
4609 Dump after cleaning up the barrier instructions.

4611 @item -fdump-rtl-bbpart
4612 @opindex fdump-rtl-bbpart
4613 Dump after partitioning hot and cold basic blocks.

4615 @item -fdump-rtl-bbro
4616 @opindex fdump-rtl-bbro
4617 Dump after block reordering.

4619 @item -fdump-rtl-btl1
4620 @itemx -fdump-rtl-btl2
4621 @opindex fdump-rtl-btl2
4622 @opindex fdump-rtl-btl2
4623 @option{-fdump-rtl-btl1} and @option{-fdump-rtl-btl2} enable dumping
4624 after the two branch
4625 target load optimization passes.

4627 @item -fdump-rtl-bypass
4628 @opindex fdump-rtl-bypass
4629 Dump after jump bypassing and control flow optimizations.

4631 @item -fdump-rtl-combine
4632 @opindex fdump-rtl-combine
4633 Dump after the RTL instruction combination pass.

4635 @item -fdump-rtl-compgotos
4636 @opindex fdump-rtl-compgotos
4637 Dump after duplicating the computed gotos.

4639 @item -fdump-rtl-ce1
4640 @itemx -fdump-rtl-ce2
4641 @itemx -fdump-rtl-ce3
4642 @opindex fdump-rtl-ce1
4643 @opindex fdump-rtl-ce2
4644 @opindex fdump-rtl-ce3
```

```
@option{-fdump-rtl-ce1}, @option{-fdump-rtl-ce2}, and
@option{-fdump-rtl-ce3} enable dumping after the three
if conversion passes.

@itemx -fdump-rtl-cprop_hardreg
@opindex fdump-rtl-cprop_hardreg
Dump after hard register copy propagation.

@itemx -fdump-rtl-csa
@opindex fdump-rtl-csa
Dump after combining stack adjustments.

@item -fdump-rtl-cse1
@itemx -fdump-rtl-cse2
@opindex fdump-rtl-cse1
@opindex fdump-rtl-cse2
@option{-fdump-rtl-cse1} and @option{-fdump-rtl-cse2} enable dumping after
the two common sub-expression elimination passes.

@itemx -fdump-rtl-dce
@opindex fdump-rtl-dce
Dump after the standalone dead code elimination passes.

@itemx -fdump-rtl-dbr
@opindex fdump-rtl-dbr
Dump after delayed branch scheduling.

@item -fdump-rtl-dce1
@itemx -fdump-rtl-dce2
@opindex fdump-rtl-dce1
@opindex fdump-rtl-dce2
@option{-fdump-rtl-dce1} and @option{-fdump-rtl-dce2} enable dumping after
the two dead store elimination passes.

@item -fdump-rtl-eh
@opindex fdump-rtl-eh
Dump after finalization of EH handling code.

@item -fdump-rtl-eh_ranges
@opindex fdump-rtl-eh_ranges
Dump after conversion of EH handling range regions.

@item -fdump-rtl-expand
@opindex fdump-rtl-expand
Dump after RTL generation.

@item -fdump-rtl-fwprop1
@itemx -fdump-rtl-fwprop2
@opindex fdump-rtl-fwprop1
@opindex fdump-rtl-fwprop2
@option{-fdump-rtl-fwprop1} and @option{-fdump-rtl-fwprop2} enable
dumping after the two forward propagation passes.

@item -fdump-rtl-gcse1
@itemx -fdump-rtl-gcse2
@opindex fdump-rtl-gcse1
@opindex fdump-rtl-gcse2
@option{-fdump-rtl-gcse1} and @option{-fdump-rtl-gcse2} enable dumping
after global common subexpression elimination.

@item -fdump-rtl-init-regs
@opindex fdump-rtl-init-regs
Dump after the initialization of the registers.

@item -fdump-rtl-initvals
@opindex fdump-rtl-initvals
```



```
Dump after the computation of the initial value sets.

@itemx -fdump-rtl-into_cfglayout
@opindex fdump-rtl-into_cfglayout
Dump after converting to cfglayout mode.

@item -fdump-rtl-ira
@opindex fdump-rtl-ira
Dump after iterated register allocation.

@item -fdump-rtl-jump
@opindex fdump-rtl-jump
Dump after the second jump optimization.

@item -fdump-rtl-loop2
@opindex fdump-rtl-loop2
@option{-fdump-rtl-loop2} enables dumping after the rtl
loop optimization passes.

@item -fdump-rtl-mach
@opindex fdump-rtl-mach
Dump after performing the machine dependent reorganization pass, if that
pass exists.

@item -fdump-rtl-mode_sw
@opindex fdump-rtl-mode_sw
Dump after removing redundant mode switches.

@item -fdump-rtl-rnreg
@opindex fdump-rtl-rnreg
Dump after register renumbering.

@itemx -fdump-rtl-outof_cfglayout
@opindex fdump-rtl-outof_cfglayout
Dump after converting from cfglayout mode.

@item -fdump-rtl-peephole2
@opindex fdump-rtl-peephole2
Dump after the peephole pass.

@item -fdump-rtl-postreload
@opindex fdump-rtl-postreload
Dump after post-reload optimizations.

@itemx -fdump-rtl-pro_and_epilogue
@opindex fdump-rtl-pro_and_epilogue
Dump after generating the function pro and epilogues.

@item -fdump-rtl-regmove
@opindex fdump-rtl-regmove
Dump after the register move pass.

@item -fdump-rtl-sched1
@itemx -fdump-rtl-sched2
@opindex fdump-rtl-sched1
@opindex fdump-rtl-sched2
@option{-fdump-rtl-sched1} and @option{-fdump-rtl-sched2} enable dumping
after the basic block scheduling passes.

@item -fdump-rtl-see
@opindex fdump-rtl-see
Dump after sign extension elimination.

@item -fdump-rtl-seqabstr
@opindex fdump-rtl-seqabstr
Dump after common sequence discovery.
```

```
4778 @item -fdump-rtl-shorten
4779 @opindex fdump-rtl-shorten
4780 Dump after shortening branches.

4782 @item -fdump-rtl-sibling
4783 @opindex fdump-rtl-sibling
4784 Dump after sibling call optimizations.

4786 @item -fdump-rtl-split1
4787 @itemx -fdump-rtl-split2
4788 @itemx -fdump-rtl-split3
4789 @itemx -fdump-rtl-split4
4790 @itemx -fdump-rtl-split5
4791 @opindex fdump-rtl-split1
4792 @opindex fdump-rtl-split2
4793 @opindex fdump-rtl-split3
4794 @opindex fdump-rtl-split4
4795 @opindex fdump-rtl-split5
4796 @option{-fdump-rtl-split1}, @option{-fdump-rtl-split2},
4797 @option{-fdump-rtl-split3}, @option{-fdump-rtl-split4} and
4798 @option{-fdump-rtl-split5} enable dumping after five rounds of
4799 instruction splitting.

4801 @item -fdump-rtl-sms
4802 @opindex fdump-rtl-sms
4803 Dump after modulo scheduling.  This pass is only run on some
4804 architectures.

4806 @item -fdump-rtl-stack
4807 @opindex fdump-rtl-stack
4808 Dump after conversion from GCC's "flat register file" registers to the
4809 x87's stack-like registers.  This pass is only run on x86 variants.

4811 @item -fdump-rtl-subreg1
4812 @itemx -fdump-rtl-subreg2
4813 @opindex fdump-rtl-subreg1
4814 @opindex fdump-rtl-subreg2
4815 @option{-fdump-rtl-subreg1} and @option{-fdump-rtl-subreg2} enable dumping after
4816 the two subreg expansion passes.

4818 @item -fdump-rtl-unshare
4819 @opindex fdump-rtl-unshare
4820 Dump after all rtl has been unshared.

4822 @item -fdump-rtl-vartrack
4823 @opindex fdump-rtl-vartrack
4824 Dump after variable tracking.

4826 @item -fdump-rtl-vregs
4827 @opindex fdump-rtl-vregs
4828 Dump after converting virtual registers to hard registers.

4830 @item -fdump-rtl-web
4831 @opindex fdump-rtl-web
4832 Dump after live range splitting.

4834 @item -fdump-rtl-regclass
4835 @itemx -fdump-rtl-subregs_of_mode_init
4836 @itemx -fdump-rtl-subregs_of_mode_finish
4837 @itemx -fdump-rtl-dfinit
4838 @itemx -fdump-rtl-dfinish
4839 @opindex fdump-rtl-regclass
4840 @opindex fdump-rtl-subregs_of_mode_init
4841 @opindex fdump-rtl-subregs_of_mode_finish
4842 @opindex fdump-rtl-dfinit
```



```
4843 @opindex fdump-rtl-dfinish
4844 These dumps are defined but always produce empty files.

4846 @item -fdump-rtl-all
4847 @opindex fdump-rtl-all
4848 Produce all the dumps listed above.

4850 @item -dA
4851 @opindex dA
4852 Annotate the assembler output with miscellaneous debugging information.

4854 @item -dD
4855 @opindex dD
4856 Dump all macro definitions, at the end of preprocessing, in addition to
4857 normal output.

4859 @item -dH
4860 @opindex dH
4861 Produce a core dump whenever an error occurs.

4863 @item -dm
4864 @opindex dm
4865 Print statistics on memory usage, at the end of the run, to
4866 standard error.

4868 @item -dp
4869 @opindex dp
4870 Annotate the assembler output with a comment indicating which
4871 pattern and alternative was used.  The length of each instruction is
4872 also printed.

4874 @item -dP
4875 @opindex dP
4876 Dump the RTL in the assembler output as a comment before each instruction.
4877 Also turns on @option{-dp} annotation.

4879 @item -dv
4880 @opindex dv
4881 For each of the other indicated dump files (@option{-fdump-rtl-@var{pass}}),
4882 dump a representation of the control flow graph suitable for viewing with VCG
4883 to @file{@var{file}.@var{pass}.vcg}.

4885 @item -dx
4886 @opindex dx
4887 Just generate RTL for a function instead of compiling it.  Usually used
4888 with @option{-fdump-rtl-expand}.

4890 @item -dy
4891 @opindex dy
4892 Dump debugging information during parsing, to standard error.
4893 @end table

4895 @item -fdump-noaddr
4896 @opindex fdump-noaddr
4897 When doing debugging dumps, suppress address output.  This makes it more
4898 feasible to use diff on debugging dumps for compiler invocations with
4899 different compiler binaries and/or different
4900 text / bss / data / heap / stack / dso start locations.

4902 @item -fdump-unnumbered
4903 @opindex fdump-unnumbered
4904 When doing debugging dumps, suppress instruction numbers and address output.
4905 This makes it more feasible to use diff on debugging dumps for compiler
4906 invocations with different options, in particular with and without
4907 @option{-g}.
```

```
@item -fdump-translation-unit @r{(C++ only)}
@itemx -fdump-translation-unit-@var{options} @r{(C++ only)}
@opindex fdump-translation-unit
Dump a representation of the tree structure for the entire translation
unit to a file.  The file name is made by appending @file{.tu} to the
source file name.  If the @samp{-@var{options}} form is used, @var{options}
controls the details of the dump as described for the
@option{-fdump-tree} options.

@item -fdump-class-hierarchy @r{(C++ only)}
@itemx -fdump-class-hierarchy-@var{options} @r{(C++ only)}
@opindex fdump-class-hierarchy
Dump a representation of each class's hierarchy and virtual function
table layout to a file.  The file name is made by appending @file{.class}
to the source file name.  If the @samp{-@var{options}} form is used,
@var{options} controls the details of the dump as described for the
@option{-fdump-tree} options.

@item -fdump-ipa-@var{switch}
@opindex fdump-ipa
Control the dumping at various stages of inter-procedural analysis
language tree to a file.  The file name is generated by appending a switch
specific suffix to the source file name.  The following dumps are possible:

@table @samp
@item all
Enables all inter-procedural analysis dumps.

@item cgraph
Dumps information about call-graph optimization, unused function removal,
and inlining decisions.

@item inline
Dump after function inlining.

@end table

@item -fdump-statistics-@var{option}
@opindex -fdump-statistics
Enable and control dumping of pass statistics in a separate file.  The
file name is generated by appending a suffix ending in @samp{.statistics}
to the source file name.  If the @samp{-@var{option}} form is used,
@samp{-stats} will cause counters to be summed over the whole compilation unit
while @samp{-details} will dump every event as the passes generate them.
The default with no option is to sum counters for each function compiled.

@item -fdump-tree-@var{switch}
@itemx -fdump-tree-@var{switch}-@var{options}
@opindex fdump-tree
Control the dumping at various stages of processing the intermediate
language tree to a file.  The file name is generated by appending a switch
specific suffix to the source file name.  If the @samp{-@var{options}}
form is used, @var{options} is a list of @samp{-} separated options that
control the details of the dump.  Not all options are applicable to all
dumps, those which are not meaningful will be ignored.  The following
options are available

@table @samp
@item address
Print the address of each node.  Usually this is not meaningful as it
changes according to the environment and source file.  Its primary use
is for tying up a dump file with a debug environment.
@item slim
Inhibit dumping of members of a scope or body of a function merely
because that scope has been reached.  Only dump such items when they
are directly reachable by some other path.  When dumping pretty-printed
```



```
trees, this option inhibits dumping the bodies of control structures.
@item raw
Print a raw representation of the tree.  By default, trees are
pretty-printed into a C-like representation.
@item details
Enable more detailed dumps (not honored by every dump option).
@item stats
Enable dumping various statistics about the pass (not honored by every dump
option).
@item blocks
Enable showing basic block boundaries (disabled in raw dumps).
@item vops
Enable showing virtual operands for every statement.
@item lineno
Enable showing line numbers for statements.
@item uid
Enable showing the unique ID (@code{DECL_UID}) for each variable.
@item verbose
Enable showing the tree dump for each statement.
@item all
Turn on all options, except @option{raw}, @option{slim}, @option{verbose}
and @option{lineno}.
@end table

The following tree dumps are possible:
@table @samp

@item original
Dump before any tree based optimization, to @file{@var{file}.original}.

@item optimized
Dump after all tree based optimization, to @file{@var{file}.optimized}.

@item gimple
@opindex fdump-tree-gimple
Dump each function before and after the gimplification pass to a file.  The
file name is made by appending @file{.gimple} to the source file name.

@item cfg
@opindex fdump-tree-cfg
Dump the control flow graph of each function to a file.  The file name is
made by appending @file{.cfg} to the source file name.

@item vcg
@opindex fdump-tree-vcg
Dump the control flow graph of each function to a file in VCG format.  The
file name is made by appending @file{.vcg} to the source file name.  Note
that if the file contains more than one function, the generated file cannot
be used directly by VCG@.  You will need to cut and paste each function's
graph into its own separate file first.

@item ch
@opindex fdump-tree-ch
Dump each function after copying loop headers.  The file name is made by
appending @file{.ch} to the source file name.

@item ssa
@opindex fdump-tree-ssa
Dump SSA related information to a file.  The file name is made by appending
@file{.ssa} to the source file name.

@item alias
@opindex fdump-tree-alias
Dump aliasing information for each function.  The file name is made by
appending @file{.alias} to the source file name.
```

```
@item ccp
@opindex fdump-tree-ccp
Dump each function after CCP@.  The file name is made by appending
@file{.ccp} to the source file name.

@item storeccp
@opindex fdump-tree-storeccp
Dump each function after STORE-CCP@.  The file name is made by appending
@file{.storeccp} to the source file name.

@item pre
@opindex fdump-tree-pre
Dump trees after partial redundancy elimination.  The file name is made
by appending @file{.pre} to the source file name.

@item fre
@opindex fdump-tree-fre
Dump trees after full redundancy elimination.  The file name is made
by appending @file{.fre} to the source file name.

@item copyprop
@opindex fdump-tree-copyprop
Dump trees after copy propagation.  The file name is made
by appending @file{.copyprop} to the source file name.

@item store_copyprop
@opindex fdump-tree-store_copyprop
Dump trees after store copy-propagation.  The file name is made
by appending @file{.store_copyprop} to the source file name.

@item dce
@opindex fdump-tree-dce
Dump each function after dead code elimination.  The file name is made by
appending @file{.dce} to the source file name.

@item mudflap
@opindex fdump-tree-mudflap
Dump each function after adding mudflap instrumentation.  The file name is
made by appending @file{.mudflap} to the source file name.

@item sra
@opindex fdump-tree-sra
Dump each function after performing scalar replacement of aggregates.  The
file name is made by appending @file{.sra} to the source file name.

@item sink
@opindex fdump-tree-sink
Dump each function after performing code sinking.  The file name is made
by appending @file{.sink} to the source file name.

@item dom
@opindex fdump-tree-dom
Dump each function after applying dominator tree optimizations.  The file
name is made by appending @file{.dom} to the source file name.

@item dse
@opindex fdump-tree-dse
Dump each function after applying dead store elimination.  The file
name is made by appending @file{.dse} to the source file name.

@item phiopt
@opindex fdump-tree-phiopt
Dump each function after optimizing PHI nodes into straightline code.  The file
name is made by appending @file{.phiopt} to the source file name.

@item forwprop
```



```
@opindex fdump-tree-forwprop
Dump each function after forward propagating single use variables.  The file
name is made by appending @file{.forwprop} to the source file name.

@item copyrename
@opindex fdump-tree-copyrename
Dump each function after applying the copy rename optimization.  The file
name is made by appending @file{.copyrename} to the source file name.

@item nrv
@opindex fdump-tree-nrv
Dump each function after applying the named return value optimization on
generic trees.  The file name is made by appending @file{.nrv} to the source
file name.

@item vect
@opindex fdump-tree-vect
Dump each function after applying vectorization of loops.  The file name is
made by appending @file{.vect} to the source file name.

@item vrp
@opindex fdump-tree-vrp
Dump each function after Value Range Propagation (VRP).  The file name
is made by appending @file{.vrp} to the source file name.

@item all
@opindex fdump-tree-all
Enable all the available tree dumps with the flags provided in this option.
@end table

@item -ftree-vectorizer-verbose=@var{n}
@opindex ftree-vectorizer-verbose
This option controls the amount of debugging output the vectorizer prints.
This information is written to standard error, unless
@option{-fdump-tree-all} or @option{-fdump-tree-vect} is specified,
in which case it is output to the usual dump listing file, @file{.vect}.
For @var{n}=0 no diagnostic information is reported.
If @var{n}=1 the vectorizer reports each loop that got vectorized,
and the total number of loops that got vectorized.
If @var{n}=2 the vectorizer also reports non-vectorized loops that passed
the first analysis phase (vect_analyze_loop_form) - i.e.@: countable,
inner-most, single-bb, single-entry/exit loops.  This is the same verbosity
level that @option{-fdump-tree-vect-stats} uses.
Higher verbosity levels mean either more information dumped for each
reported loop, or same amount of information reported for more loops:
If @var{n}=3, alignment related information is added to the reports.
If @var{n}=4, data-references related information (e.g.@: memory dependences,
memory access-patterns) is added to the reports.
If @var{n}=5, the vectorizer reports also non-vectorized inner-most loops
that did not pass the first analysis phase (i.e., may not be countable, or
may have complicated control-flow).
If @var{n}=6, the vectorizer reports also non-vectorized nested loops.
For @var{n}=7, all the information the vectorizer generates during its
analysis and transformation is reported.  This is the same verbosity level
that @option{-fdump-tree-vect-details} uses.

@item -frandom-seed=@var{string}
@opindex frandom-string
This option provides a seed that GCC uses when it would otherwise use
random numbers.  It is used to generate certain symbol names
that have to be different in every compiled file.  It is also used to
place unique stamps in coverage data files and the object files that
produce them.  You can use the @option{-frandom-seed} option to produce
reproducibly identical object files.

The @var{string} should be different for every file you compile.
```

```
@item -fsched-verbose=@var{n}
@opindex fsched-verbose
On targets that use instruction scheduling, this option controls the
amount of debugging output the scheduler prints.  This information is
written to standard error, unless @option{-fdump-rtl-sched1} or
@option{-fdump-rtl-sched2} is specified, in which case it is output
to the usual dump listing file, @file{.sched} or @file{.sched2}
respectively.  However for @var{n} greater than nine, the output is
always printed to standard error.

For @var{n} greater than zero, @option{-fsched-verbose} outputs the
same information as @option{-fdump-rtl-sched1} and @option{-fdump-rtl-sched2}.
For @var{n} greater than one, it also output basic block probabilities,
detailed ready list information and unit/insn info.  For @var{n} greater
than two, it includes RTL at abort point, control-flow and regions info.
And for @var{n} over four, @option{-fsched-verbose} also includes
dependence info.

@item -save-temps
@opindex save-temps
Store the usual ''temporary'' intermediate files permanently; place them
in the current directory and name them based on the source file.  Thus,
compiling @file{foo.c} with @samp{-c -save-temps} would produce files
@file{foo.i} and @file{foo.s}, as well as @file{foo.o}.  This creates a
preprocessed @file{foo.i} output file even though the compiler now
normally uses an integrated preprocessor.

When used in combination with the @option{-x} command line option,
@option{-save-temps} is sensible enough to avoid over writing an
input source file with the same extension as an intermediate file.
The corresponding intermediate file may be obtained by renaming the
source file before using @option{-save-temps}.

@item -time
@opindex time
Report the CPU time taken by each subprocess in the compilation
sequence.  For C source files, this is the compiler proper and assembler
(plus the linker if linking is done).  The output looks like this:

@smallexample
# cc1 0.12 0.01
# as 0.00 0.01
@end smallexample

The first number on each line is the ''user time'', that is time spent
executing the program itself.  The second number is ''system time'',
time spent executing operating system routines on behalf of the program.
Both numbers are in seconds.

@item -fvar-tracking
@opindex fvar-tracking
Run variable tracking pass.  It computes where variables are stored at each
position in code.  Better debugging information is then generated
(if the debugging information format supports this information).

It is enabled by default when compiling with optimization (@option{-Os},
@option{-O}, @option{-O2}, @dots{}), debugging information (@option{-g}) and
the debug info format supports it.

@item -print-file-name=@var{library}
@opindex print-file-name
Print the full absolute name of the library file @var{library} that
would be used when linking---and don't do anything else.  With this
option, GCC does not compile or link anything; it just prints the
file name.
```



```
@item -print-multi-directory
@opindex print-multi-directory
Print the directory name corresponding to the multilib selected by any
other switches present in the command line.  This directory is supposed
to exist in @env{GCC_EXEC_PREFIX}.

@item -print-multi-lib
@opindex print-multi-lib
Print the mapping from multilib directory names to compiler switches
that enable them.  The directory name is separated from the switches by
@samp{;}, and each switch starts with an @samp{@@} instead of the
@samp{-}, without spaces between multiple switches.  This is supposed to
ease shell-processing.

@item -print-prog-name=@var{program}
@opindex print-prog-name
Like @option{-print-file-name}, but searches for a program such as @samp{cpp}.

@item -print-libgcc-file-name
@opindex print-libgcc-file-name
Same as @option{-print-file-name=libgcc.a}.

This is useful when you use @option{-nostdlib} or @option{-nodefaultlibs}
but you do want to link with @file{libgcc.a}.  You can do

@smallexample
gcc -nostdlib @var{files}@dots{} `gcc -print-libgcc-file-name`
@end smallexample

@item -print-search-dirs
@opindex print-search-dirs
Print the name of the configured installation directory and a list of
program and library directories @command{gcc} will search---and don't do anythin

This is useful when @command{gcc} prints the error message
@samp{installation problem, cannot exec cpp0: No such file or directory}.
To resolve this you either need to put @file{cpp0} and the other compiler
components where @command{gcc} expects to find them, or you can set the environm
variable @env{GCC_EXEC_PREFIX} to the directory where you installed them.
Don't forget the trailing @samp{/}.
@xref{Environment Variables}.

@item -print-sysroot
@opindex print-sysroot
Print the target sysroot directory that will be used during
compilation.  This is the target sysroot specified either at configure
time or using the @option{--sysroot} option, possibly with an extra
suffix that depends on compilation options.  If no target sysroot is
specified, the option prints nothing.

@item -print-sysroot-headers-suffix
@opindex print-sysroot-headers-suffix
Print the suffix added to the target sysroot when searching for
headers, or give an error if the compiler is not configured with such
a suffix---and don't do anything else.

@item -dumpmachine
@opindex dumpmachine
Print the compiler's target machine (for example,
@samp{i686-pc-linux-gnu})---and don't do anything else.

@item -dumpversion
@opindex dumpversion
Print the compiler version (for example, @samp{3.0})---and don't do
anything else.
```

```
@item -dumpspecs
@opindex dumpspecs
Print the compiler's built-in specs---and don't do anything else.  (This
is used when GCC itself is being built.)  @xref{Spec Files}.

@item -feliminate-unused-debug-types
@opindex feliminate-unused-debug-types
Normally, when producing DWARF2 output, GCC will emit debugging
information for all types declared in a compilation
unit, regardless of whether or not they are actually used
in that compilation unit.  Sometimes this is useful, such as
if, in the debugger, you want to cast a value to a type that is
not actually used in your program (but is declared).  More often,
however, this results in a significant amount of wasted space.
With this option, GCC will avoid producing debug symbol output
for types that are nowhere used in the source file being compiled.
@end table

@node Optimize Options
@section Options That Control Optimization
@cindex optimize options
@cindex options, optimization

These options control various sorts of optimizations.

Without any optimization option, the compiler's goal is to reduce the
cost of compilation and to make debugging produce the expected
results.  Statements are independent: if you stop the program with a
breakpoint between statements, you can then assign a new value to any
variable or change the program counter to any other statement in the
function and get exactly the results you would expect from the source
code.

Turning on optimization flags makes the compiler attempt to improve
the performance and/or code size at the expense of compilation time
and possibly the ability to debug the program.

The compiler performs optimization based on the knowledge it has of the
program.  Compiling multiple files at once to a single output file mode allows
the compiler to use information gained from all of the files when compiling
each of them.

Not all optimizations are controlled directly by a flag.  Only
optimizations that have a flag are listed.

@table @gcctabopt
@item -O
@itemx -O1
@opindex O
@opindex O1
Optimize.  Optimizing compilation takes somewhat more time, and a lot
more memory for a large function.

With @option{-O}, the compiler tries to reduce code size and execution
time, without performing any optimizations that take a great deal of
compilation time.

@option{-O} turns on the following optimization flags:
@gccoptlist{
-fauto-inc-dec @gol
-fcprop-registers @gol
-fdce @gol
-fdefer-pop @gol
-fdelayed-branch @gol
-fdse @gol
```



```
-fguess-branch-probability @gol
-fif-conversion2 @gol
-fif-conversion @gol
-finline-small-functions @gol
-fipa-pure-const @gol
-fipa-reference @gol
-fmerge-constants
-fsplit-wide-types @gol
-ftree-builtin-call-dce @gol
-ftree-ccp @gol
-ftree-ch @gol
-ftree-copyrename @gol
-ftree-dce @gol
-ftree-dominator-opts @gol
-ftree-dse @gol
-ftree-fre @gol
-ftree-sra @gol
-ftree-ter @gol
-funit-at-a-time}

@option{-O} also turns on @option{-fomit-frame-pointer} on machines
where doing so does not interfere with debugging.

@item -O2
@opindex O2
Optimize even more.  GCC performs nearly all supported optimizations
that do not involve a space-speed tradeoff.
As compared to @option{-O}, this option increases both compilation time
and the performance of the generated code.

@option{-O2} turns on all optimization flags specified by @option{-O}.  It
also turns on the following optimization flags:
@gccoptlist{-fthread-jumps @gol
-falign-functions  -falign-jumps @gol
-falign-loops  -falign-labels @gol
-fcaller-saves @gol
-fcrossjumping @gol
-fcse-follow-jumps  -fcse-skip-blocks @gol
-fdelete-null-pointer-checks @gol
-fexpensive-optimizations @gol
-fgcse  -fgcse-lm  @gol
-findirect-inlining @gol
-foptimize-sibling-calls @gol
-fpeephole2 @gol
-fregmove @gol
-freorder-blocks  -freorder-functions @gol
-frerun-cse-after-loop  @gol
-fsched-interblock  -fsched-spec @gol
-fschedule-insns  -fschedule-insns2 @gol
-fstrict-aliasing -fstrict-overflow @gol
-ftree-switch-conversion @gol
-ftree-pre @gol
-ftree-vrp}

Please note the warning under @option{-fgcse} about
invoking @option{-O2} on programs that use computed gotos.

@item -O3
@opindex O3
Optimize yet more.  @option{-O3} turns on all optimizations specified
by @option{-O2} and also turns on the @option{-finline-functions},
@option{-funswitch-loops}, @option{-fpredictive-commoning},
@option{-fgcse-after-reload} and @option{-ftree-vectorize} options.

@item -O0
@opindex O0
```

Reduce compilation time and make debugging produce the expected
results. This is the default.

@item -Os
@opindex Os
Optimize for size. @option{-Os} enables all @option{-O2} optimizations that
do not typically increase code size. It also performs further
optimizations designed to reduce code size.

@option{-Os} disables the following optimization flags:
@gccoptlist{-falign-functions -falign-jumps -falign-loops @gol
-falign-labels -freorder-blocks -freorder-blocks-and-partition @gol
-fprefetch-loop-arrays -ftree-vect-loop-version}

If you use multiple @option{-O} options, with or without level numbers,
the last such option is the one that is effective.
@end table

Options of the form @option{-f@var{flag}} specify machine-independent
flags. Most flags have both positive and negative forms; the negative
form of @option{-ffoo} would be @option{-fno-foo}. In the table
below, only one of the forms is listed---the one you typically will
use. You can figure out the other form by either removing @samp{no-}
or adding it.

The following options control specific optimizations. They are either
activated by @option{-O} options or are related to ones that are. You
can use the following flags in the rare cases when ''fine-tuning'' of
optimizations to be performed is desired.

@table @gcctabopt
@item -fno-default-inline
@opindex fno-default-inline
Do not make member functions inline by default merely because they are
defined inside the class scope (C++ only). Otherwise, when you specify
@w{@option{-O}}, member functions defined inside class scope are compiled
inline by default; i.e., you don't need to add @samp{inline} in front of
the member function name.

@item -fno-defer-pop
@opindex fno-defer-pop
Always pop the arguments to each function call as soon as that function
returns. For machines which must pop arguments after a function call,
the compiler normally lets arguments accumulate on the stack for several
function calls and pops them all at once.

Disabled at levels @option{-O}, @option{-O2}, @option{-O3}, @option{-Os}.

@item -fforward-propagate
@opindex fforward-propagate
Perform a forward propagation pass on RTL@. The pass tries to combine two
instructions and checks if the result can be simplified. If loop unrolling
is active, two passes are performed and the second is scheduled after
loop unrolling.

This option is enabled by default at optimization levels @option{-O2},
@option{-O3}, @option{-Os}.

@item -fomit-frame-pointer
@opindex fomit-frame-pointer
Don't keep the frame pointer in a register for functions that
don't need one. This avoids the instructions to save, set up and
restore frame pointers; it also makes an extra register available
in many functions. @strong{It also makes debugging impossible on
some machines.}

On some machines, such as the VAX, this flag has no effect, because
the standard calling sequence automatically handles the frame pointer
and nothing is saved by pretending it doesn't exist. The
machine-description macro @code{FRAME_POINTER_REQUIRED} controls
whether a target machine supports this flag. @xref{Registers,,Register
Usage, gccint, GNU Compiler Collection (GCC) Internals}.

Enabled at levels @option{-O}, @option{-O2}, @option{-O3}, @option{-Os}.

@item -foptimize-sibling-calls
@opindex foptimize-sibling-calls
Optimize sibling and tail recursive calls.

Enabled at levels @option{-O2}, @option{-O3}, @option{-Os}.

@item -fno-inline
@opindex fno-inline
Don't pay attention to the @code{inline} keyword. Normally this option
is used to keep the compiler from expanding any functions inline.
Note that if you are not optimizing, no functions can be expanded inline.

@item -finline-small-functions
@opindex finline-small-functions
Integrate functions into their callers when their body is smaller than expected
function call code (so overall size of program gets smaller). The compiler
heuristically decides which functions are simple enough to be worth integrating
in this way.

Enabled at level @option{-O2}.

@item -findirect-inlining
@opindex findirect-inlining
Inline also indirect calls that are discovered to be known at compile
time thanks to previous inlining. This option has any effect only
when inlining itself is turned on by the @option{-finline-functions}
or @option{-finline-small-functions} options.

Enabled at level @option{-O2}.

@item -finline-functions
@opindex finline-functions
Integrate all simple functions into their callers. The compiler
heuristically decides which functions are simple enough to be worth
integrating in this way.

If all calls to a given function are integrated, and the function is
declared @code{static}, then the function is normally not output as
assembler code in its own right.

Enabled at level @option{-O3}.

@item -finline-functions-called-once
@opindex finline-functions-called-once
Consider all @code{static} functions called once for inlining into their
caller even if they are not marked @code{inline}. If a call to a given
function is integrated, then the function is not output as assembler code
in its own right.

Enabled at levels @option{-O1}, @option{-O2}, @option{-O3} and @option{-Os}.

@item -fearly-inlining
@opindex fearly-inlining
Inline functions marked by @code{always_inline} and functions whose body seems
smaller than the function call overhead early before doing
@option{-fprofile-generate} instrumentation and real inlining pass. Doing so
makes profiling significantly cheaper and usually inlining faster on programs

having large chains of nested wrapper functions.

Enabled by default.

@item -finline-limit=@var{n}
@opindex finline-limit
By default, GCC limits the size of functions that can be inlined. This flag
allows coarse control of this limit. @var{n} is the size of functions that
can be inlined in number of pseudo instructions.

Inlining is actually controlled by a number of parameters, which may be
specified individually by using @option{--param @var{name}=@var{value}}.
The @option{-finline-limit=@var{n}} option sets some of these parameters
as follows:

@table @gcctabopt
@item max-inline-insns-single
is set to @var{n}/2.
@item max-inline-insns-auto
is set to @var{n}/2.
@end table

See below for a documentation of the individual
parameters controlling inlining and for the defaults of these parameters.

@emph{Note:} there may be no value to @option{-finline-limit} that results
in default behavior.

@emph{Note:} pseudo instruction represents, in this particular context, an
abstract measurement of function's size. In no way does it represent a count
of assembly instructions and as such its exact meaning might change from one
release to an another.

@item -fkeep-inline-functions
@opindex fkeep-inline-functions
In C, emit @code{static} functions that are declared @code{inline}
into the object file, even if the function has been inlined into all
of its callers. This switch does not affect functions using the
@code{extern inline} extension in GNU C89@. In C++, emit any and all
inline functions into the object file.

@item -fkeep-static-consts
@opindex fkeep-static-consts
Emit variables declared @code{static const} when optimization isn't turned
on, even if the variables aren't referenced.

GCC enables this option by default. If you want to force the compiler to
check if the variable was referenced, regardless of whether or not
optimization is turned on, use the @option{-fno-keep-static-consts} option.

@item -fmerge-constants
@opindex fmerge-constants
Attempt to merge identical constants (string constants and floating point
constants) across compilation units.

This option is the default for optimized compilation if the assembler and
linker support it. Use @option{-fno-merge-constants} to inhibit this
behavior.

Enabled at levels @option{-O}, @option{-O2}, @option{-O3}, @option{-Os}.

@item -fmerge-all-constants
@opindex fmerge-all-constants
Attempt to merge identical constants and identical variables.

This option implies @option{-fmerge-constants}. In addition to

@option{-fmerge-constants} this considers e.g.@: even constant initialized
arrays or initialized constant variables with integral or floating point
types. Languages like C or C++ require each variable, including multiple
instances of the same variable in recursive calls, to have distinct locations,
so using this option will result in non-conforming
behavior.

@item -fmodulo-sched
@opindex fmodulo-sched
Perform swing modulo scheduling immediately before the first scheduling
pass. This pass looks at innermost loops and reorders their
instructions by overlapping different iterations.

@item -fmodulo-sched-allow-regmoves
@opindex fmodulo-sched-allow-regmoves
Perform more aggressive SMS based modulo scheduling with register moves
allowed. By setting this flag certain anti-dependences edges will be
deleted which will trigger the generation of reg-moves based on the
life-range analysis. This option is effective only with
@option{-fmodulo-sched} enabled.

@item -fno-branch-count-reg
@opindex fno-branch-count-reg
Do not use ''decrement and branch'' instructions on a count register,
but instead generate a sequence of instructions that decrement a
register, compare it against zero, then branch based upon the result.
This option is only meaningful on architectures that support such
instructions, which include x86, PowerPC, IA-64 and S/390.

The default is @option{-fbranch-count-reg}.

@item -fno-function-cse
@opindex fno-function-cse
Do not put function addresses in registers; make each instruction that
calls a constant function contain the function's address explicitly.

This option results in less efficient code, but some strange hacks
that alter the assembler output may be confused by the optimizations
performed when this option is not used.

The default is @option{-ffunction-cse}

@item -fno-zero-initialized-in-bss
@opindex fno-zero-initialized-in-bss
If the target supports a BSS section, GCC by default puts variables that
are initialized to zero into BSS@. This can save space in the resulting
code.

This option turns off this behavior because some programs explicitly
rely on variables going to the data section. E.g., so that the
resulting executable can find the beginning of that section and/or make
assumptions based on that.

The default is @option{-fzero-initialized-in-bss}.

@item -fmudflap -fmudflapth -fmudflapir
@opindex fmudflap
@opindex fmudflapth
@opindex fmudflapir
@cindex bounds checking
@cindex mudflap
For front-ends that support it (C and C++), instrument all risky
pointer/array dereferencing operations, some standard library
string/heap functions, and some other associated constructs with
range/validity tests. Modules so instrumented should be immune to
buffer overflows, invalid heap use, and some other classes of C/C++

programming errors. The instrumentation relies on a separate runtime
library (@file{libmudflap}), which will be linked into a program if
@option{-fmudflap} is given at link time. Run-time behavior of the
instrumented program is controlled by the @env{MUDFLAP_OPTIONS}
environment variable. See @code{env MUDFLAP_OPTIONS=-help a.out}
for its options.

Use @option{-fmudflapth} instead of @option{-fmudflap} to compile and to
link if your program is multi-threaded. Use @option{-fmudflapir}, in
addition to @option{-fmudflap} or @option{-fmudflapth}, if
instrumentation should ignore pointer reads. This produces less
instrumentation (and therefore faster execution) and still provides
some protection against outright memory corrupting writes, but allows
erroneously read data to propagate within a program.

@item -fthread-jumps
@opindex fthread-jumps
Perform optimizations where we check to see if a jump branches to a
location where another comparison subsumed by the first is found. If
so, the first branch is redirected to either the destination of the
second branch or a point immediately following it, depending on whether
the condition is known to be true or false.

Enabled at levels @option{-O2}, @option{-O3}, @option{-Os}.

@item -fsplit-wide-types
@opindex fsplit-wide-types
When using a type that occupies multiple registers, such as @code{long
long} on a 32-bit system, split the registers apart and allocate them
independently. This normally generates better code for those types,
but may make debugging more difficult.

Enabled at levels @option{-O}, @option{-O2}, @option{-O3},
@option{-Os}.

@item -fcse-follow-jumps
@opindex fcse-follow-jumps
In common subexpression elimination (CSE), scan through jump instructions
when the target of the jump is not reached by any other path. For
example, when CSE encounters an @code{if} statement with an
@code{else} clause, CSE will follow the jump when the condition
tested is false.

Enabled at levels @option{-O2}, @option{-O3}, @option{-Os}.

@item -fcse-skip-blocks
@opindex fcse-skip-blocks
This is similar to @option{-fcse-follow-jumps}, but causes CSE to
follow jumps which conditionally skip over blocks. When CSE
encounters a simple @code{if} statement with no else clause,
@option{-fcse-skip-blocks} causes CSE to follow the jump around the
body of the @code{if}.

Enabled at levels @option{-O2}, @option{-O3}, @option{-Os}.

@item -frerun-cse-after-loop
@opindex frerun-cse-after-loop
Re-run common subexpression elimination after loop optimizations has been
performed.

Enabled at levels @option{-O2}, @option{-O3}, @option{-Os}.

@item -fgcse
@opindex fgcse
Perform a global common subexpression elimination pass.
This pass also performs global constant and copy propagation.

@emph{Note:} When compiling a program using computed gotos, a GCC
extension, you may get better runtime performance if you disable
the global common subexpression elimination pass by adding
@option{-fno-gcse} to the command line.

Enabled at levels @option{-O2}, @option{-O3}, @option{-Os}.

@item -fgcse-lm
@opindex fgcse-lm
When @option{-fgcse-lm} is enabled, global common subexpression elimination will
attempt to move loads which are only killed by stores into themselves. This
allows a loop containing a load/store sequence to be changed to a load outside
the loop, and a copy/store within the loop.

Enabled by default when gcse is enabled.

@item -fgcse-sm
@opindex fgcse-sm
When @option{-fgcse-sm} is enabled, a store motion pass is run after
global common subexpression elimination. This pass will attempt to move
stores out of loops. When used in conjunction with @option{-fgcse-lm},
loops containing a load/store sequence can be changed to a load before
the loop and a store after the loop.

Not enabled at any optimization level.

@item -fgcse-las
@opindex fgcse-las
When @option{-fgcse-las} is enabled, the global common subexpression
elimination pass eliminates redundant loads that come after stores to the
same memory location (both partial and full redundancies).

Not enabled at any optimization level.

@item -fgcse-after-reload
@opindex fgcse-after-reload
When @option{-fgcse-after-reload} is enabled, a redundant load elimination
pass is performed after reload. The purpose of this pass is to cleanup
redundant spilling.

@item -funsafe-loop-optimizations
@opindex funsafe-loop-optimizations
If given, the loop optimizer will assume that loop indices do not
overflow, and that the loops with nontrivial exit condition are not
infinite. This enables a wider range of loop optimizations even if
the loop optimizer itself cannot prove that these assumptions are valid.
Using @option{-Wunsafe-loop-optimizations}, the compiler will warn you
if it finds this kind of loop.

@item -fcrossjumping
@opindex fcrossjumping
Perform cross-jumping transformation. This transformation unifies equivalent co
resulting code may or may not perform better than without cross-jumping.

Enabled at levels @option{-O2}, @option{-O3}, @option{-Os}.

@item -fauto-inc-dec
@opindex fauto-inc-dec
Combine increments or decrements of addresses with memory accesses.
This pass is always skipped on architectures that do not have
instructions to support this. Enabled by default at @option{-O} and
higher on architectures that support this.

@item -fdce
@opindex fdce

```
5833 Perform dead code elimination (DCE) on RTL@.
5834 Enabled by default at @option{-O} and higher.

5836 @item -fdse
5837 @opindex fdse
5838 Perform dead store elimination (DSE) on RTL@.
5839 Enabled by default at @option{-O} and higher.

5841 @item -fif-conversion
5842 @opindex fif-conversion
5843 Attempt to transform conditional jumps into branch-less equivalents.  This
5844 include use of conditional moves, min, max, set flags and abs instructions, and
5845 some tricks doable by standard arithmetics.  The use of conditional execution
5846 on chips where it is available is controlled by @code{if-conversion2}.

5848 Enabled at levels @option{-O}, @option{-O2}, @option{-O3}, @option{-Os}.

5850 @item -fif-conversion2
5851 @opindex fif-conversion2
5852 Use conditional execution (where available) to transform conditional jumps into
5853 branch-less equivalents.

5855 Enabled at levels @option{-O}, @option{-O2}, @option{-O3}, @option{-Os}.

5857 @item -fdelete-null-pointer-checks
5858 @opindex fdelete-null-pointer-checks
5859 Use global dataflow analysis to identify and eliminate useless checks
5860 for null pointers.  The compiler assumes that dereferencing a null
5861 pointer would have halted the program.  If a pointer is checked after
5862 it has already been dereferenced, it cannot be null.

5864 In some environments, this assumption is not true, and programs can
5865 safely dereference null pointers.  Use
5866 @option{-fno-delete-null-pointer-checks} to disable this optimization
5867 for programs which depend on that behavior.

5869 Enabled at levels @option{-O2}, @option{-O3}, @option{-Os}.

5871 @item -fexpensive-optimizations
5872 @opindex fexpensive-optimizations
5873 Perform a number of minor optimizations that are relatively expensive.

5875 Enabled at levels @option{-O2}, @option{-O3}, @option{-Os}.

5877 @item -foptimize-register-move
5878 @itemx -fregmove
5879 @opindex foptimize-register-move
5880 @opindex fregmove
5881 Attempt to reassign register numbers in move instructions and as
5882 operands of other simple instructions in order to maximize the amount of
5883 register tying.  This is especially helpful on machines with two-operand
5884 instructions.

5886 Note @option{-fregmove} and @option{-foptimize-register-move} are the same
5887 optimization.

5889 Enabled at levels @option{-O2}, @option{-O3}, @option{-Os}.

5891 @item -fira-algorithm=@var{algorithm}
5892 Use specified coloring algorithm for the integrated register
5893 allocator.  The @var{algorithm} argument should be @code{priority} or
5894 @code{CB}.  The first algorithm specifies Chow's priority coloring,
5895 the second one specifies Chaitin-Briggs coloring.  The second
5896 algorithm can be unimplemented for some architectures.  If it is
5897 implemented, it is the default because Chaitin-Briggs coloring as a
5898 rule generates a better code.
```



```
5900 @item -fira-region=@var{region}
5901 Use specified regions for the integrated register allocator.  The
5902 @var{region} argument should be one of @code{all}, @code{mixed}, or
5903 @code{one}.  The first value means using all loops as register
5904 allocation regions, the second value which is the default means using
5905 all loops except for loops with small register pressure as the
5906 regions, and third one means using all function as a single region.
5907 The first value can give best result for machines with small size and
5908 irregular register set, the third one results in faster and generates
5909 decent code and the smallest size code, and the default value usually
5910 give the best results in most cases and for most architectures.

5912 @item -fira-coalesce
5913 @opindex fira-coalesce
5914 Do optimistic register coalescing.  This option might be profitable for
5915 architectures with big regular register files.

5917 @item -fno-ira-share-save-slots
5918 @opindex fno-ira-share-save-slots
5919 Switch off sharing stack slots used for saving call used hard
5920 registers living through a call.  Each hard register will get a
5921 separate stack slot and as a result function stack frame will be
5922 bigger.

5924 @item -fno-ira-share-spill-slots
5925 @opindex fno-ira-share-spill-slots
5926 Switch off sharing stack slots allocated for pseudo-registers.  Each
5927 pseudo-register which did not get a hard register will get a separate
5928 stack slot and as a result function stack frame will be bigger.

5930 @item -fira-verbose=@var{n}
5931 @opindex fira-verbose
5932 Set up how verbose dump file for the integrated register allocator
5933 will be.  Default value is 5.  If the value is greater or equal to 10,
5934 the dump file will be stderr as if the value were @var{n} minus 10.

5936 @item -fdelayed-branch
5937 @opindex fdelayed-branch
5938 If supported for the target machine, attempt to reorder instructions
5939 to exploit instruction slots available after delayed branch
5940 instructions.

5942 Enabled at levels @option{-O}, @option{-O2}, @option{-O3}, @option{-Os}.

5944 @item -fschedule-insns
5945 @opindex fschedule-insns
5946 If supported for the target machine, attempt to reorder instructions to
5947 eliminate execution stalls due to required data being unavailable.  This
5948 helps machines that have slow floating point or memory load instructions
5949 by allowing other instructions to be issued until the result of the load
5950 or floating point instruction is required.

5952 Enabled at levels @option{-O2}, @option{-O3}, @option{-Os}.

5954 @item -fschedule-insns2
5955 @opindex fschedule-insns2
5956 Similar to @option{-fschedule-insns}, but requests an additional pass of
5957 instruction scheduling after register allocation has been done.  This is
5958 especially useful on machines with a relatively small number of
5959 registers and where memory load instructions take more than one cycle.

5961 Enabled at levels @option{-O2}, @option{-O3}, @option{-Os}.

5963 @item -fno-sched-interblock
5964 @opindex fno-sched-interblock
```

```
5965 Don't schedule instructions across basic blocks.  This is normally
5966 enabled by default when scheduling before register allocation, i.e.@:
5967 with @option{-fschedule-insns} or at @option{-O2} or higher.

5969 @item -fno-sched-spec
5970 @opindex fno-sched-spec
5971 Don't allow speculative motion of non-load instructions.  This is normally
5972 enabled by default when scheduling before register allocation, i.e.@:
5973 with @option{-fschedule-insns} or at @option{-O2} or higher.

5975 @item -fsched-spec-load
5976 @opindex fsched-spec-load
5977 Allow speculative motion of some load instructions.  This only makes
5978 sense when scheduling before register allocation, i.e.@: with
5979 @option{-fschedule-insns} or at @option{-O2} or higher.

5981 @item -fsched-spec-load-dangerous
5982 @opindex fsched-spec-load-dangerous
5983 Allow speculative motion of more load instructions.  This only makes
5984 sense when scheduling before register allocation, i.e.@: with
5985 @option{-fschedule-insns} or at @option{-O2} or higher.

5987 @item -fsched-stalled-insns
5988 @itemx -fsched-stalled-insns=@var{n}
5989 @opindex fsched-stalled-insns
5990 Define how many insns (if any) can be moved prematurely from the queue
5991 of stalled insns into the ready list, during the second scheduling pass.
5992 @option{-fno-sched-stalled-insns} means that no insns will be moved
5993 prematurely, @option{-fsched-stalled-insns=0} means there is no limit
5994 on how many queued insns can be moved prematurely.
5995 @option{-fsched-stalled-insns} without a value is equivalent to
5996 @option{-fsched-stalled-insns=1}.

5998 @item -fsched-stalled-insns-dep
5999 @itemx -fsched-stalled-insns-dep=@var{n}
6000 @opindex fsched-stalled-insns-dep
6001 Define how many insn groups (cycles) will be examined for a dependency
6002 on a stalled insn that is candidate for premature removal from the queue
6003 of stalled insns.  This has an effect only during the second scheduling pass,
6004 and only if @option{-fsched-stalled-insns} is used.
6005 @option{-fno-sched-stalled-insns-dep} is equivalent to
6006 @option{-fsched-stalled-insns-dep=0}.
6007 @option{-fsched-stalled-insns-dep} without a value is equivalent to
6008 @option{-fsched-stalled-insns-dep=1}.

6010 @item -fsched2-use-superblocks
6011 @opindex fsched2-use-superblocks
6012 When scheduling after register allocation, do use superblock scheduling
6013 algorithm.  Superblock scheduling allows motion across basic block boundaries
6014 resulting on faster schedules.  This option is experimental, as not all machine
6015 descriptions used by GCC model the CPU closely enough to avoid unreliable
6016 results from the algorithm.

6018 This only makes sense when scheduling after register allocation, i.e.@: with
6019 @option{-fschedule-insns2} or at @option{-O2} or higher.

6021 @item -fsched2-use-traces
6022 @opindex fsched2-use-traces
6023 Use @option{-fsched2-use-superblocks} algorithm when scheduling after register
6024 allocation and additionally perform code duplication in order to increase the
6025 size of superblocks using tracer pass.  See @option{-ftracer} for details on
6026 trace formation.

6028 This mode should produce faster but significantly longer programs.  Also
6029 without @option{-fbranch-probabilities} the traces constructed may not
6030 match the reality and hurt the performance.  This only makes
```



```
6031 sense when scheduling after register allocation, i.e.@: with
6032 @option{-fschedule-insns2} or at @option{-O2} or higher.

6034 @item -fsee
6035 @opindex fsee
6036 Eliminate redundant sign extension instructions and move the non-redundant
6037 ones to optimal placement using lazy code motion (LCM).

6039 @item -freschedule-modulo-scheduled-loops
6040 @opindex freschedule-modulo-scheduled-loops
6041 The modulo scheduling comes before the traditional scheduling, if a loop
6042 was modulo scheduled we may want to prevent the later scheduling passes
6043 from changing its schedule, we use this option to control that.

6045 @item -fselective-scheduling
6046 @opindex fselective-scheduling
6047 Schedule instructions using selective scheduling algorithm.  Selective
6048 scheduling runs instead of the first scheduler pass.

6050 @item -fselective-scheduling2
6051 @opindex fselective-scheduling2
6052 Schedule instructions using selective scheduling algorithm.  Selective
6053 scheduling runs instead of the second scheduler pass.

6055 @item -fsel-sched-pipelining
6056 @opindex fsel-sched-pipelining
6057 Enable software pipelining of innermost loops during selective scheduling.
6058 This option has no effect until one of @option{-fselective-scheduling} or
6059 @option{-fselective-scheduling2} is turned on.

6061 @item -fsel-sched-pipelining-outer-loops
6062 @opindex fsel-sched-pipelining-outer-loops
6063 When pipelining loops during selective scheduling, also pipeline outer loops.
6064 This option has no effect until @option{-fsel-sched-pipelining} is turned on.

6066 @item -fcaller-saves
6067 @opindex fcaller-saves
6068 Enable values to be allocated in registers that will be clobbered by
6069 function calls, by emitting extra instructions to save and restore the
6070 registers around such calls.  Such allocation is done only when it
6071 seems to result in better code than would otherwise be produced.

6073 This option is always enabled by default on certain machines, usually
6074 those which have no call-preserved registers to use instead.

6076 Enabled at levels @option{-O2}, @option{-O3}, @option{-Os}.

6078 @item -fconserve-stack
6079 @opindex fconserve-stack
6080 Attempt to minimize stack usage.  The compiler will attempt to use less
6081 stack space, even if that makes the program slower.  This option
6082 implies setting the @option{large-stack-frame} parameter to 100
6083 and the @option{large-stack-frame-growth} parameter to 400.

6085 @item -ftree-reassoc
6086 @opindex ftree-reassoc
6087 Perform reassociation on trees.  This flag is enabled by default
6088 at @option{-O} and higher.

6090 @item -ftree-pre
6091 @opindex ftree-pre
6092 Perform partial redundancy elimination (PRE) on trees.  This flag is
6093 enabled by default at @option{-O2} and @option{-O3}.

6095 @item -ftree-fre
6096 @opindex ftree-fre
```

Perform full redundancy elimination (FRE) on trees. The difference
between FRE and PRE is that FRE only considers expressions
that are computed on all paths leading to the redundant computation.
This analysis is faster than PRE, though it exposes fewer redundancies.
This flag is enabled by default at @option{-O} and higher.

@item -ftree-copy-prop
@opindex ftree-copy-prop
Perform copy propagation on trees. This pass eliminates unnecessary
copy operations. This flag is enabled by default at @option{-O} and
higher.

@item -fipa-pure-const
@opindex fipa-pure-const
Discover which functions are pure or constant.
Enabled by default at @option{-O} and higher.

@item -fipa-reference
@opindex fipa-reference
Discover which static variables do not escape cannot escape the
compilation unit.
Enabled by default at @option{-O} and higher.

@item -fipa-struct-reorg
@opindex fipa-struct-reorg
Perform structure reorganization optimization, that change C-like structures
layout in order to better utilize spatial locality. This transformation is
affective for programs containing arrays of structures. Available in two
compilation modes: profile-based (enabled with @option{-fprofile-generate})
or static (which uses built-in heuristics). Require @option{-fipa-type-escape}
to provide the safety of this transformation. It works only in whole program
mode, so it requires @option{-fwhole-program} and @option{-combine} to be
enabled. Structures considered @samp{cold} by this transformation are not
affected (see @option{--param struct-reorg-cold-struct-ratio=@var{value}}).

With this flag, the program debug info reflects a new structure layout.

@item -fipa-pta
@opindex fipa-pta
Perform interprocedural pointer analysis. This option is experimental
and does not affect generated code.

@item -fipa-cp
@opindex fipa-cp
Perform interprocedural constant propagation.
This optimization analyzes the program to determine when values passed
to functions are constants and then optimizes accordingly.
This optimization can substantially increase performance
if the application has constants passed to functions.
This flag is enabled by default at @option{-O2}, @option{-Os} and @option{-O3}.

@item -fipa-cp-clone
@opindex fipa-cp-clone
Perform function cloning to make interprocedural constant propagation stronger.
When enabled, interprocedural constant propagation will perform function cloning
when externally visible function can be called with constant arguments.
Because this optimization can create multiple copies of functions,
it may significantly increase code size
(see @option{--param ipcp-unit-growth=@var{value}}).
This flag is enabled by default at @option{-O3}.

@item -fipa-matrix-reorg
@opindex fipa-matrix-reorg
Perform matrix flattening and transposing.
Matrix flattening tries to replace a m-dimensional matrix
with its equivalent n-dimensional matrix, where n < m.

This reduces the level of indirection needed for accessing the elements
of the matrix. The second optimization is matrix transposing that
attempts to change the order of the matrix's dimensions in order to
improve cache locality.
Both optimizations need the @option{-fwhole-program} flag.
Transposing is enabled only if profiling information is available.

@item -ftree-sink
@opindex ftree-sink
Perform forward store motion on trees. This flag is
enabled by default at @option{-O} and higher.

@item -ftree-ccp
@opindex ftree-ccp
Perform sparse conditional constant propagation (CCP) on trees. This
pass only operates on local scalar variables and is enabled by default
at @option{-O} and higher.

@item -ftree-switch-conversion
Perform conversion of simple initializations in a switch to
initializations from a scalar array. This flag is enabled by default
at @option{-O2} and higher.

@item -ftree-dce
@opindex ftree-dce
Perform dead code elimination (DCE) on trees. This flag is enabled by
default at @option{-O} and higher.

@item -ftree-builtin-call-dce
@opindex ftree-builtin-call-dce
Perform conditional dead code elimination (DCE) for calls to builtin functions
that may set @code{errno} but are otherwise side-effect free. This flag is
enabled by default at @option{-O2} and higher if @option{-Os} is not also
specified.

@item -ftree-dominator-opts
@opindex ftree-dominator-opts
Perform a variety of simple scalar cleanups (constant/copy
propagation, redundancy elimination, range propagation and expression
simplification) based on a dominator tree traversal. This also
performs jump threading (to reduce jumps to jumps). This flag is
enabled by default at @option{-O} and higher.

@item -ftree-dse
@opindex ftree-dse
Perform dead store elimination (DSE) on trees. A dead store is a store into
a memory location which will later be overwritten by another store without
any intervening loads. In this case the earlier store can be deleted. This
flag is enabled by default at @option{-O} and higher.

@item -ftree-ch
@opindex ftree-ch
Perform loop header copying on trees. This is beneficial since it increases
effectiveness of code motion optimizations. It also saves one jump. This flag
is enabled by default at @option{-O} and higher. It is not enabled
for @option{-Os}, since it usually increases code size.

@item -ftree-loop-optimize
@opindex ftree-loop-optimize
Perform loop optimizations on trees. This flag is enabled by default
at @option{-O} and higher.

@item -ftree-loop-linear
@opindex ftree-loop-linear
Perform linear loop transformations on tree. This flag can improve cache

```
6229 performance and allow further loop optimizations to take place.

6231 @item -floop-interchange
6232 Perform loop interchange transformations on loops.  Interchanging two
6233 nested loops switches the inner and outer loops.  For example, given a
6234 loop like:
6235 @smallexample
6236 DO J = 1, M
6237   DO I = 1, N
6238     A(J, I) = A(J, I) * C
6239   ENDDO
6240 ENDDO
6241 @end smallexample
6242 loop interchange will transform the loop as if the user had written:
6243 @smallexample
6244 DO I = 1, N
6245   DO J = 1, M
6246     A(J, I) = A(J, I) * C
6247   ENDDO
6248 ENDDO
6249 @end smallexample
6250 which can be beneficial when @code{N} is larger than the caches,
6251 because in Fortran, the elements of an array are stored in memory
6252 contiguously by column, and the original loop iterates over rows,
6253 potentially creating at each access a cache miss.  This optimization
6254 applies to all the languages supported by GCC and is not limited to
6255 Fortran.  To use this code transformation, GCC has to be configured
6256 with @option{--with-ppl} and @option{--with-cloog} to enable the
6257 Graphite loop transformation infrastructure.

6259 @item -floop-strip-mine
6260 Perform loop strip mining transformations on loops.  Strip mining
6261 splits a loop into two nested loops.  The outer loop has strides
6262 equal to the strip size and the inner loop has strides of the
6263 original loop within a strip.  For example, given a loop like:
6264 @smallexample
6265 DO I = 1, N
6266   A(I) = A(I) + C
6267 ENDDO
6268 @end smallexample
6269 loop strip mining will transform the loop as if the user had written:
6270 @smallexample
6271 DO II = 1, N, 4
6272   DO I = II, min (II + 3, N)
6273     A(I) = A(I) + C
6274   ENDDO
6275 ENDDO
6276 @end smallexample
6277 This optimization applies to all the languages supported by GCC and is
6278 not limited to Fortran.  To use this code transformation, GCC has to
6279 be configured with @option{--with-ppl} and @option{--with-cloog} to
6280 enable the Graphite loop transformation infrastructure.

6282 @item -floop-block
6283 Perform loop blocking transformations on loops.  Blocking strip mines
6284 each loop in the loop nest such that the memory accesses of the
6285 element loops fit inside caches.  For example, given a loop like:
6286 @smallexample
6287 DO I = 1, N
6288   DO J = 1, M
6289     A(J, I) = B(I) + C(J)
6290   ENDDO
6291 ENDDO
6292 @end smallexample
6293 loop blocking will transform the loop as if the user had written:
6294 @smallexample
```



```
6295 DO II = 1, N, 64
6296   DO JJ = 1, M, 64
6297     DO I = II, min (II + 63, N)
6298       DO J = JJ, min (JJ + 63, M)
6299         A(J, I) = B(I) + C(J)
6300       ENDDO
6301     ENDDO
6302   ENDDO
6303 ENDDO
6304 @end smallexample
6305 which can be beneficial when @code{M} is larger than the caches,
6306 because the innermost loop will iterate over a smaller amount of data
6307 that can be kept in the caches.  This optimization applies to all the
6308 languages supported by GCC and is not limited to Fortran.  To use this
6309 code transformation, GCC has to be configured with @option{--with-ppl}
6310 and @option{--with-cloog} to enable the Graphite loop transformation
6311 infrastructure.

6313 @item -fcheck-data-deps
6314 @opindex fcheck-data-deps
6315 Compare the results of several data dependence analyzers.  This option
6316 is used for debugging the data dependence analyzers.

6318 @item -ftree-loop-distribution
6319 Perform loop distribution.  This flag can improve cache performance on
6320 big loop bodies and allow further loop optimizations, like
6321 parallelization or vectorization, to take place.  For example, the loop
6322 @smallexample
6323 DO I = 1, N
6324   A(I) = B(I) + C
6325   D(I) = E(I) * F
6326 ENDDO
6327 @end smallexample
6328 is transformed to
6329 @smallexample
6330 DO I = 1, N
6331    A(I) = B(I) + C
6332 ENDDO
6333 DO I = 1, N
6334    D(I) = E(I) * F
6335 ENDDO
6336 @end smallexample

6338 @item -ftree-loop-im
6339 @opindex ftree-loop-im
6340 Perform loop invariant motion on trees.  This pass moves only invariants that
6341 would be hard to handle at RTL level (function calls, operations that expand to
6342 nontrivial sequences of insns).  With @option{-funswitch-loops} it also moves
6343 operands of conditions that are invariant out of the loop, so that we can use
6344 just trivial invariantness analysis in loop unswitching.  The pass also includes
6345 store motion.

6347 @item -ftree-loop-ivcanon
6348 @opindex ftree-loop-ivcanon
6349 Create a canonical counter for number of iterations in the loop for that
6350 determining number of iterations requires complicated analysis.  Later
6351 optimizations then may determine the number easily.  Useful especially
6352 in connection with unrolling.

6354 @item -fivopts
6355 @opindex fivopts
6356 Perform induction variable optimizations (strength reduction, induction
6357 variable merging and induction variable elimination) on trees.

6359 @item -ftree-parallelize-loops=n
6360 @opindex ftree-parallelize-loops
```

Parallelize loops, i.e., split their iteration space to run in n threads.
This is only possible for loops whose iterations are independent
and can be arbitrarily reordered. The optimization is only
profitable on multiprocessor machines, for loops that are CPU-intensive,
rather than constrained e.g.@: by memory bandwidth. This option
implies @option{-pthread}, and thus is only supported on targets
that have support for @option{-pthread}.

@item -ftree-sra
@opindex ftree-sra
Perform scalar replacement of aggregates. This pass replaces structure
references with scalars to prevent committing structures to memory too
early. This flag is enabled by default at @option{-O} and higher.

@item -ftree-copyrename
@opindex ftree-copyrename
Perform copy renaming on trees. This pass attempts to rename compiler
temporaries to other variables at copy locations, usually resulting in
variable names which more closely resemble the original variables. This flag
is enabled by default at @option{-O} and higher.

@item -ftree-ter
@opindex ftree-ter
Perform temporary expression replacement during the SSA->normal phase. Single
use/single def temporaries are replaced at their use location with their
defining expression. This results in non-GIMPLE code, but gives the expanders
much more complex trees to work on resulting in better RTL generation. This is
enabled by default at @option{-O} and higher.

@item -ftree-vectorize
@opindex ftree-vectorize
Perform loop vectorization on trees. This flag is enabled by default at
@option{-O3}.

@item -ftree-vect-loop-version
@opindex ftree-vect-loop-version
Perform loop versioning when doing loop vectorization on trees. When a loop
appears to be vectorizable except that data alignment or data dependence cannot
be determined at compile time then vectorized and non-vectorized versions of
the loop are generated along with runtime checks for alignment or dependence
to control which version is executed. This option is enabled by default
except at level @option{-Os} where it is disabled.

@item -fvect-cost-model
@opindex fvect-cost-model
Enable cost model for vectorization.

@item -ftree-vrp
@opindex ftree-vrp
Perform Value Range Propagation on trees. This is similar to the
constant propagation pass, but instead of values, ranges of values are
propagated. This allows the optimizers to remove unnecessary range
checks like array bound checks and null pointer checks. This is
enabled by default at @option{-O2} and higher. Null pointer check
elimination is only done if @option{-fdelete-null-pointer-checks} is
enabled.

@item -ftracer
@opindex ftracer
Perform tail duplication to enlarge superblock size. This transformation
simplifies the control flow of the function allowing other optimizations to do
better job.

@item -funroll-loops
@opindex funroll-loops
Unroll loops whose number of iterations can be determined at compile

time or upon entry to the loop. @option{-funroll-loops} implies
@option{-frerun-cse-after-loop}. This option makes code larger,
and may or may not make it run faster.

@item -funroll-all-loops
@opindex funroll-all-loops
Unroll all loops, even if their number of iterations is uncertain when
the loop is entered. This usually makes programs run more slowly.
@option{-funroll-all-loops} implies the same options as
@option{-funroll-loops},

@item -fsplit-ivs-in-unroller
@opindex fsplit-ivs-in-unroller
Enables expressing of values of induction variables in later iterations
of the unrolled loop using the value in the first iteration. This breaks
long dependency chains, thus improving efficiency of the scheduling passes.

Combination of @option{-fweb} and CSE is often sufficient to obtain the
same effect. However in cases the loop body is more complicated than
a single basic block, this is not reliable. It also does not work at all
on some of the architectures due to restrictions in the CSE pass.

This optimization is enabled by default.

@item -fvariable-expansion-in-unroller
@opindex fvariable-expansion-in-unroller
With this option, the compiler will create multiple copies of some
local variables when unrolling a loop which can result in superior code.

@item -fpredictive-commoning
@opindex fpredictive-commoning
Perform predictive commoning optimization, i.e., reusing computations
(especially memory loads and stores) performed in previous
iterations of loops.

This option is enabled at level @option{-O3}.

@item -fprefetch-loop-arrays
@opindex fprefetch-loop-arrays
If supported by the target machine, generate instructions to prefetch
memory to improve the performance of loops that access large arrays.

This option may generate better or worse code; results are highly
dependent on the structure of loops within the source code.

Disabled at level @option{-Os}.

@item -fno-peephole
@itemx -fno-peephole2
@opindex fno-peephole
@opindex fno-peephole2
Disable any machine-specific peephole optimizations. The difference
between @option{-fno-peephole} and @option{-fno-peephole2} is in how they
are implemented in the compiler; some targets use one, some use the
other, a few use both.

@option{-fpeephole} is enabled by default.
@option{-fpeephole2} enabled at levels @option{-O2}, @option{-O3}, @option{-Os}.

@item -fno-guess-branch-probability
@opindex fno-guess-branch-probability
Do not guess branch probabilities using heuristics.

GCC will use heuristics to guess branch probabilities if they are
not provided by profiling feedback (@option{-fprofile-arcs}). These
heuristics are based on the control flow graph. If some branch probabilities

```
6493 are specified by @samp{__builtin_expect}, then the heuristics will be
6494 used to guess branch probabilities for the rest of the control flow graph,
6495 taking the @samp{__builtin_expect} info into account.  The interactions
6496 between the heuristics and @samp{__builtin_expect} can be complex, and in
6497 some cases, it may be useful to disable the heuristics so that the effects
6498 of @samp{__builtin_expect} are easier to understand.

6500 The default is @option{-fguess-branch-probability} at levels
6501 @option{-O}, @option{-O2}, @option{-O3}, @option{-Os}.

6503 @item -freorder-blocks
6504 @opindex freorder-blocks
6505 Reorder basic blocks in the compiled function in order to reduce number of
6506 taken branches and improve code locality.

6508 Enabled at levels @option{-O2}, @option{-O3}.

6510 @item -freorder-blocks-and-partition
6511 @opindex freorder-blocks-and-partition
6512 In addition to reordering basic blocks in the compiled function, in order
6513 to reduce number of taken branches, partitions hot and cold basic blocks
6514 into separate sections of the assembly and .o files, to improve
6515 paging and cache locality performance.

6517 This optimization is automatically turned off in the presence of
6518 exception handling, for linkonce sections, for functions with a user-defined
6519 section attribute and on any architecture that does not support named
6520 sections.

6522 @item -freorder-functions
6523 @opindex freorder-functions
6524 Reorder functions in the object file in order to
6525 improve code locality.  This is implemented by using special
6526 subsections @code{.text.hot} for most frequently executed functions and
6527 @code{.text.unlikely} for unlikely executed functions.  Reordering is done by
6528 the linker so object file format must support named sections and linker must
6529 place them in a reasonable way.

6531 Also profile feedback must be available in to make this option effective.  See
6532 @option{-fprofile-arcs} for details.

6534 Enabled at levels @option{-O2}, @option{-O3}, @option{-Os}.

6536 @item -fstrict-aliasing
6537 @opindex fstrict-aliasing
6538 Allow the compiler to assume the strictest aliasing rules applicable to
6539 the language being compiled.  For C (and C++), this activates
6540 optimizations based on the type of expressions.  In particular, an
6541 object of one type is assumed never to reside at the same address as an
6542 object of a different type, unless the types are almost the same.  For
6543 example, an @code{unsigned int} can alias an @code{int}, but not a
6544 @code{void*} or a @code{double}.  A character type may alias any other
6545 type.

6547 @anchor{Type-punning}Pay special attention to code like this:
6548 @smallexample
6549 union a_union @{
6550   int i;
6551   double d;
6552 @};

6554 int f() @{
6555   union a_union t;
6556   t.d = 3.0;
6557   return t.i;
6558 @}
```



```
6559 @end smallexample
6560 The practice of reading from a different union member than the one most
6561 recently written to (called ``type-punning'') is common.  Even with
6562 @option{-fstrict-aliasing}, type-punning is allowed, provided the memory
6563 is accessed through the union type.  So, the code above will work as
6564 expected.  @xref{Structures unions enumerations and bit-fields
6565 implementation}.  However, this code might not:
6566 @smallexample
6567 int f() @{
6568   union a_union t;
6569   int* ip;
6570   t.d = 3.0;
6571   ip = &t.i;
6572   return *ip;
6573 @}
6574 @end smallexample

6576 Similarly, access by taking the address, casting the resulting pointer
6577 and dereferencing the result has undefined behavior, even if the cast
6578 uses a union type, e.g.:
6579 @smallexample
6580 int f() @{
6581   double d = 3.0;
6582   return ((union a_union *) &d)->i;
6583 @}
6584 @end smallexample

6586 The @option{-fstrict-aliasing} option is enabled at levels
6587 @option{-O2}, @option{-O3}, @option{-Os}.

6589 @item -fstrict-calling-conventions
6590 @opindex mstrict-calling-conventions
6591 Use strict ABI calling conventions even with local functions.
6592 This disable certain optimizations that may cause GCC to call local
6593 functions in a manner other than that described by the ABI.

6595 #endif /* ! codereview */
6596 @item -fstrict-overflow
6597 @opindex fstrict-overflow
6598 Allow the compiler to assume strict signed overflow rules, depending
6599 on the language being compiled.  For C (and C++) this means that
6600 overflow when doing arithmetic with signed numbers is undefined, which
6601 means that the compiler may assume that it will not happen.  This
6602 permits various optimizations.  For example, the compiler will assume
6603 that an expression like @code{i + 10 > i} will always be true for
6604 signed @code{i}.  This assumption is only valid if signed overflow is
6605 undefined, as the expression is false if @code{i + 10} overflows when
6606 using twos complement arithmetic.  When this option is in effect any
6607 attempt to determine whether an operation on signed numbers will
6608 overflow must be written carefully to not actually involve overflow.

6610 This option also allows the compiler to assume strict pointer
6611 semantics: given a pointer to an object, if adding an offset to that
6612 pointer does not produce a pointer to the same object, the addition is
6613 undefined.  This permits the compiler to conclude that @code{p + u >
6614 p} is always true for a pointer @code{p} and unsigned integer
6615 @code{u}.  This assumption is only valid because pointer wraparound is
6616 undefined, as the expression is false if @code{p + u} overflows using
6617 twos complement arithmetic.

6619 See also the @option{-fwrapv} option.  Using @option{-fwrapv} means
6620 that integer signed overflow is fully defined: it wraps.  When
6621 @option{-fwrapv} is used, there is no difference between
6622 @option{-fstrict-overflow} and @option{-fno-strict-overflow} for
6623 integers.  With @option{-fwrapv} certain types of overflow are
6624 permitted.  For example, if the compiler gets an overflow when doing
```

```
arithmetic on constants, the overflowed value can still be used with
@option{-fwrapv}, but not otherwise.

The @option{-fstrict-overflow} option is enabled at levels
@option{-O2}, @option{-O3}, @option{-Os}.

@item -falign-functions
@itemx -falign-functions=@var{n}
@opindex falign-functions
Align the start of functions to the next power-of-two greater than
@var{n}, skipping up to @var{n} bytes.  For instance,
@option{-falign-functions=32} aligns functions to the next 32-byte
boundary, but @option{-falign-functions=24} would align to the next
32-byte boundary only if this can be done by skipping 23 bytes or less.

@option{-fno-align-functions} and @option{-falign-functions=1} are
equivalent and mean that functions will not be aligned.

Some assemblers only support this flag when @var{n} is a power of two;
in that case, it is rounded up.

If @var{n} is not specified or is zero, use a machine-dependent default.

Enabled at levels @option{-O2}, @option{-O3}.

@item -falign-labels
@itemx -falign-labels=@var{n}
@opindex falign-labels
Align all branch targets to a power-of-two boundary, skipping up to
@var{n} bytes like @option{-falign-functions}.  This option can easily
make code slower, because it must insert dummy operations for when the
branch target is reached in the usual flow of the code.

@option{-fno-align-labels} and @option{-falign-labels=1} are
equivalent and mean that labels will not be aligned.

If @option{-falign-loops} or @option{-falign-jumps} are applicable and
are greater than this value, then their values are used instead.

If @var{n} is not specified or is zero, use a machine-dependent default
which is very likely to be @samp{1}, meaning no alignment.

Enabled at levels @option{-O2}, @option{-O3}.

@item -falign-loops
@itemx -falign-loops=@var{n}
@opindex falign-loops
Align loops to a power-of-two boundary, skipping up to @var{n} bytes
like @option{-falign-functions}.  The hope is that the loop will be
executed many times, which will make up for any execution of the dummy
operations.

@option{-fno-align-loops} and @option{-falign-loops=1} are
equivalent and mean that loops will not be aligned.

If @var{n} is not specified or is zero, use a machine-dependent default.

Enabled at levels @option{-O2}, @option{-O3}.

@item -falign-jumps
@itemx -falign-jumps=@var{n}
@opindex falign-jumps
Align branch targets to a power-of-two boundary, for branch targets
where the targets can only be reached by jumping, skipping up to @var{n}
bytes like @option{-falign-functions}.  In this case, no dummy operations
need be executed.
```



```
@option{-fno-align-jumps} and @option{-falign-jumps=1} are
equivalent and mean that loops will not be aligned.

If @var{n} is not specified or is zero, use a machine-dependent default.

Enabled at levels @option{-O2}, @option{-O3}.

@item -funit-at-a-time
@opindex funit-at-a-time
This option is left for compatibility reasons. @option{-funit-at-a-time}
has no effect, while @option{-fno-unit-at-a-time} implies
@option{-fno-toplevel-reorder} and @option{-fno-section-anchors}.

Enabled by default.

@item -fno-toplevel-reorder
@opindex fno-toplevel-reorder
Do not reorder top-level functions, variables, and @code{asm}
statements.  Output them in the same order that they appear in the
input file.  When this option is used, unreferenced static variables
will not be removed.  This option is intended to support existing code
which relies on a particular ordering.  For new code, it is better to
use attributes.

Enabled at level @option{-O0}.  When disabled explicitly, it also imply
@option{-fno-section-anchors} that is otherwise enabled at @option{-O0} on some
targets.

@item -fweb
@opindex fweb
Constructs webs as commonly used for register allocation purposes and assign
each web individual pseudo register.  This allows the register allocation pass
to operate on pseudos directly, but also strengthens several other optimization
passes, such as CSE, loop optimizer and trivial dead code remover.  It can,
however, make debugging impossible, since variables will no longer stay in a
``home register''.

Enabled by default with @option{-funroll-loops}.

@item -fwhole-program
@opindex fwhole-program
Assume that the current compilation unit represents whole program being
compiled.  All public functions and variables with the exception of @code{main}
and those merged by attribute @code{externally_visible} become static functions
and in a affect gets more aggressively optimized by interprocedural optimizers.
While this option is equivalent to proper use of @code{static} keyword for
programs consisting of single file, in combination with option
@option{--combine} this flag can be used to compile most of smaller scale C
programs since the functions and variables become local for the whole combined
compilation unit, not for the single source file itself.

This option is not supported for Fortran programs.

@item -fcprop-registers
@opindex fcprop-registers
After register allocation and post-register allocation instruction splitting,
we perform a copy-propagation pass to try to reduce scheduling dependencies
and occasionally eliminate the copy.

Enabled at levels @option{-O}, @option{-O2}, @option{-O3}, @option{-Os}.

@item -fprofile-correction
@opindex fprofile-correction
Profiles collected using an instrumented binary for multi-threaded programs may
be inconsistent due to missed counter updates. When this option is specified,
```

GCC will use heuristics to correct or smooth out such inconsistencies. By
default, GCC will emit an error message when an inconsistent profile is detected

@item -fprofile-dir=@var{path}
@opindex fprofile-dir

Set the directory to search the profile data files in to @var{path}.
This option affects only the profile data generated by
@option{-fprofile-generate}, @option{-ftest-coverage}, @option{-fprofile-arcs}
and used by @option{-fprofile-use} and @option{-fbranch-probabilities}
and its related options.
By default, GCC will use the current directory as @var{path}
thus the profile data file will appear in the same directory as the object file.

@item -fprofile-generate
@itemx -fprofile-generate=@var{path}
@opindex fprofile-generate

Enable options usually used for instrumenting application to produce
profile useful for later recompilation with profile feedback based
optimization. You must use @option{-fprofile-generate} both when
compiling and when linking your program.

The following options are enabled: @code{-fprofile-arcs}, @code{-fprofile-values

If @var{path} is specified, GCC will look at the @var{path} to find
the profile feedback data files. See @option{-fprofile-dir}.

@item -fprofile-use
@itemx -fprofile-use=@var{path}
@opindex fprofile-use
Enable profile feedback directed optimizations, and optimizations
generally profitable only with profile feedback available.

The following options are enabled: @code{-fbranch-probabilities}, @code{-fvpt},
@code{-funroll-loops}, @code{-fpeel-loops}, @code{-ftracer}

By default, GCC emits an error message if the feedback profiles do not
match the source code. This error can be turned into a warning by using
@option{-Wcoverage-mismatch}. Note this may result in poorly optimized
code.

If @var{path} is specified, GCC will look at the @var{path} to find
the profile feedback data files. See @option{-fprofile-dir}.
@end table

The following options control compiler behavior regarding floating
point arithmetic. These options trade off between speed and
correctness. All must be specifically enabled.

@table @gcctabopt
@item -ffloat-store
@opindex ffloat-store
Do not store floating point variables in registers, and inhibit other
options that might change whether a floating point value is taken from a
register or memory.

@cindex floating point precision
This option prevents undesirable excess precision on machines such as
the 68000 where the floating registers (of the 68881) keep more
precision than a @code{double} is supposed to have. Similarly for the
x86 architecture. For most programs, the excess precision does only
good, but a few programs rely on the precise definition of IEEE floating
point. Use @option{-ffloat-store} for such programs, after modifying
them to store all pertinent intermediate computations into variables.

@item -ffast-math
@opindex ffast-math
Sets @option{-fno-math-errno}, @option{-funsafe-math-optimizations},
@option{-ffinite-math-only}, @option{-fno-rounding-math},
@option{-fno-signaling-nans} and @option{-fcx-limited-range}.

This option causes the preprocessor macro @code{__FAST_MATH__} to be defined.

This option is not turned on by any @option{-O} option since
it can result in incorrect output for programs which depend on
an exact implementation of IEEE or ISO rules/specifications for
math functions. It may, however, yield faster code for programs
that do not require the guarantees of these specifications.

@item -fno-math-errno
@opindex fno-math-errno
Do not set ERRNO after calling math functions that are executed
with a single instruction, e.g., sqrt. A program that relies on
IEEE exceptions for math error handling may want to use this flag
for speed while maintaining IEEE arithmetic compatibility.

This option is not turned on by any @option{-O} option since
it can result in incorrect output for programs which depend on
an exact implementation of IEEE or ISO rules/specifications for
math functions. It may, however, yield faster code for programs
that do not require the guarantees of these specifications.

The default is @option{-fmath-errno}.

On Darwin systems, the math library never sets @code{errno}. There is
therefore no reason for the compiler to consider the possibility that
it might, and @option{-fno-math-errno} is the default.

@item -funsafe-math-optimizations
@opindex funsafe-math-optimizations

Allow optimizations for floating-point arithmetic that (a) assume
that arguments and results are valid and (b) may violate IEEE or
ANSI standards. When used at link-time, it may include libraries
or startup files that change the default FPU control word or other
similar optimizations.

This option is not turned on by any @option{-O} option since
it can result in incorrect output for programs which depend on
an exact implementation of IEEE or ISO rules/specifications for
math functions. It may, however, yield faster code for programs
that do not require the guarantees of these specifications.
Enables @option{-fno-signed-zeros}, @option{-fno-trapping-math},
@option{-fassociative-math} and @option{-freciprocal-math}.

The default is @option{-fno-unsafe-math-optimizations}.

@item -fassociative-math
@opindex fassociative-math

Allow re-association of operands in series of floating-point operations.
This violates the ISO C and C++ language standard by possibly changing
computation result. NOTE: re-ordering may change the sign of zero as
well as ignore NaNs and inhibit or create underflow or overflow (and
thus cannot be used on a code which relies on rounding behavior like
@code{(x + 2**52) - 2**52)}. May also reorder floating-point comparisons
and thus may not be used when ordered comparisons are required.
This option requires that both @option{-fno-signed-zeros} and
@option{-fno-trapping-math} be in effect. Moreover, it doesn't make
much sense with @option{-frounding-math}.

```
The default is @option{-fno-associative-math}.

@item -freciprocal-math
@opindex freciprocal-math

Allow the reciprocal of a value to be used instead of dividing by
the value if this enables optimizations.  For example @code{x / y}
can be replaced with @code{x * (1/y)} which is useful if @code{(1/y)}
is subject to common subexpression elimination.  Note that this loses
precision and increases the number of flops operating on the value.

The default is @option{-fno-reciprocal-math}.

@item -ffinite-math-only
@opindex ffinite-math-only
Allow optimizations for floating-point arithmetic that assume
that arguments and results are not NaNs or +-Infs.

This option is not turned on by any @option{-O} option since
it can result in incorrect output for programs which depend on
an exact implementation of IEEE or ISO rules/specifications for
math functions. It may, however, yield faster code for programs
that do not require the guarantees of these specifications.

The default is @option{-fno-finite-math-only}.

@item -fno-signed-zeros
@opindex fno-signed-zeros
Allow optimizations for floating point arithmetic that ignore the
signedness of zero.  IEEE arithmetic specifies the behavior of
distinct +0.0 and @minus{}0.0 values, which then prohibits simplification
of expressions such as x+0.0 or 0.0*x (even with @option{-ffinite-math-only}).
This option implies that the sign of a zero result isn't significant.

The default is @option{-fsigned-zeros}.

@item -fno-trapping-math
@opindex fno-trapping-math
Compile code assuming that floating-point operations cannot generate
user-visible traps.  These traps include division by zero, overflow,
underflow, inexact result and invalid operation.  This option requires
that @option{-fno-signaling-nans} be in effect.  Setting this option may
allow faster code if one relies on ''non-stop'' IEEE arithmetic, for example.

This option should never be turned on by any @option{-O} option since
it can result in incorrect output for programs which depend on
an exact implementation of IEEE or ISO rules/specifications for
math functions.

The default is @option{-ftrapping-math}.

@item -frounding-math
@opindex frounding-math
Disable transformations and optimizations that assume default floating
point rounding behavior.  This is round-to-zero for all floating point
to integer conversions, and round-to-nearest for all other arithmetic
truncations.  This option should be specified for programs that change
the FP rounding mode dynamically, or that may be executed with a
non-default rounding mode.  This option disables constant folding of
floating point expressions at compile-time (which may be affected by
rounding mode) and arithmetic transformations that are unsafe in the
presence of sign-dependent rounding modes.

The default is @option{-fno-rounding-math}.

This option is experimental and does not currently guarantee to
```



```
disable all GCC optimizations that are affected by rounding mode.
Future versions of GCC may provide finer control of this setting
using C99's @code{FENV_ACCESS} pragma.  This command line option
will be used to specify the default state for @code{FENV_ACCESS}.

@item -frtl-abstract-sequences
@opindex frtl-abstract-sequences
It is a size optimization method. This option is to find identical
sequences of code, which can be turned into pseudo-procedures  and
then  replace  all  occurrences with  calls to  the  newly created
subroutine. It is kind of an opposite of @option{-finline-functions}.
This optimization runs at RTL level.

@item -fsignaling-nans
@opindex fsignaling-nans
Compile code assuming that IEEE signaling NaNs may generate user-visible
traps during floating-point operations.  Setting this option disables
optimizations that may change the number of exceptions visible with
signaling NaNs.  This option implies @option{-ftrapping-math}.

This option causes the preprocessor macro @code{__SUPPORT_SNAN__} to
be defined.

The default is @option{-fno-signaling-nans}.

This option is experimental and does not currently guarantee to
disable all GCC optimizations that affect signaling NaN behavior.

@item -fsingle-precision-constant
@opindex fsingle-precision-constant
Treat floating point constant as single precision constant instead of
implicitly converting it to double precision constant.

@item -fcx-limited-range
@opindex fcx-limited-range
When enabled, this option states that a range reduction step is not
needed when performing complex division.  Also, there is no checking
whether the result of a complex multiplication or division is @code{NaN
+ I*NaN}, with an attempt to rescue the situation in that case.  The
default is @option{-fno-cx-limited-range}, but is enabled by
@option{-ffast-math}.

This option controls the default setting of the ISO C99
@code{CX_LIMITED_RANGE} pragma.  Nevertheless, the option applies to
all languages.

@item -fcx-fortran-rules
@opindex fcx-fortran-rules
Complex multiplication and division follow Fortran rules.  Range
reduction is done as part of complex division, but there is no checking
whether the result of a complex multiplication or division is @code{NaN
+ I*NaN}, with an attempt to rescue the situation in that case.

The default is @option{-fno-cx-fortran-rules}.

@end table

The following options control optimizations that may improve
performance, but are not enabled by any @option{-O} options.  This
section includes experimental options that may produce broken code.

@table @gcctabopt
@item -fbranch-probabilities
@opindex fbranch-probabilities
After running a program compiled with @option{-fprofile-arcs}
(@pxref{Debugging Options,, Options for Debugging Your Program or
```

```
@command{gcc}}), you can compile it a second time using
@option{-fbranch-probabilities}, to improve optimizations based on
the number of times each branch was taken.  When the program
compiled with @option{-fprofile-arcs} exits it saves arc execution
counts to a file called @file{@var{sourcename}.gcda} for each source
file.  The information in this data file is very dependent on the
structure of the generated code, so you must use the same source code
and the same optimization options for both compilations.

With @option{-fbranch-probabilities}, GCC puts a
@samp{REG_BR_PROB} note on each @samp{JUMP_INSN} and @samp{CALL_INSN}.
These can be used to improve optimization.  Currently, they are only
used in one place: in @file{reorg.c}, instead of guessing which path a
branch is mostly to take, the @samp{REG_BR_PROB} values are used to
exactly determine which path is taken more often.

@item -fprofile-values
@opindex fprofile-values
If combined with @option{-fprofile-arcs}, it adds code so that some
data about values of expressions in the program is gathered.

With @option{-fbranch-probabilities}, it reads back the data gathered
from profiling values of expressions and adds @samp{REG_VALUE_PROFILE}
notes to instructions for their later usage in optimizations.

Enabled with @option{-fprofile-generate} and @option{-fprofile-use}.

@item -fvpt
@opindex fvpt
If combined with @option{-fprofile-arcs}, it instructs the compiler to add
a code to gather information about values of expressions.

With @option{-fbranch-probabilities}, it reads back the data gathered
and actually performs the optimizations based on them.
Currently the optimizations include specialization of division operation
using the knowledge about the value of the denominator.

@item -frename-registers
@opindex frename-registers
Attempt to avoid false dependencies in scheduled code by making use
of registers left over after register allocation.  This optimization
will most benefit processors with lots of registers.  Depending on the
debug information format adopted by the target, however, it can
make debugging impossible, since variables will no longer stay in
a ``home register''.

Enabled by default with @option{-funroll-loops}.

@item -ftracer
@opindex ftracer
Perform tail duplication to enlarge superblock size.  This transformation
simplifies the control flow of the function allowing other optimizations to do
better job.

Enabled with @option{-fprofile-use}.

@item -funroll-loops
@opindex funroll-loops
Unroll loops whose number of iterations can be determined at compile time or
upon entry to the loop.  @option{-funroll-loops} implies
@option{-frerun-cse-after-loop}, @option{-fweb} and @option{-frename-registers}.
It also turns on complete loop peeling (i.e.@: complete removal of loops with
small constant number of iterations).  This option makes code larger, and may
or may not make it run faster.

Enabled with @option{-fprofile-use}.
```



```
@item -funroll-all-loops
@opindex funroll-all-loops
Unroll all loops, even if their number of iterations is uncertain when
the loop is entered.  This usually makes programs run more slowly.
@option{-funroll-all-loops} implies the same options as
@option{-funroll-loops}.

@item -fpeel-loops
@opindex fpeel-loops
Peels the loops for that there is enough information that they do not
roll much (from profile feedback).  It also turns on complete loop peeling
(i.e.@: complete removal of loops with small constant number of iterations).

Enabled with @option{-fprofile-use}.

@item -fmove-loop-invariants
@opindex fmove-loop-invariants
Enables the loop invariant motion pass in the RTL loop optimizer.  Enabled
at level @option{-O1}

@item -funswitch-loops
@opindex funswitch-loops
Move branches with loop invariant conditions out of the loop, with duplicates
of the loop on both branches (modified according to result of the condition).

@item -ffunction-sections
@itemx -fdata-sections
@opindex ffunction-sections
@opindex fdata-sections
Place each function or data item into its own section in the output
file if the target supports arbitrary sections.  The name of the
function or the name of the data item determines the section's name
in the output file.

Use these options on systems where the linker can perform optimizations
to improve locality of reference in the instruction space.  Most systems
using the ELF object format and SPARC processors running Solaris 2 have
linkers with such optimizations.  AIX may have these optimizations in
the future.

Only use these options when there are significant benefits from doing
so.  When you specify these options, the assembler and linker will
create larger object and executable files and will also be slower.
You will not be able to use @code{gprof} on all systems if you
specify this option and you may have problems with debugging if
you specify both this option and @option{-g}.

@item -fbranch-target-load-optimize
@opindex fbranch-target-load-optimize
Perform branch target register load optimization before prologue / epilogue
threading.
The use of target registers can typically be exposed only during reload,
thus hoisting loads out of loops and doing inter-block scheduling needs
a separate optimization pass.

@item -fbranch-target-load-optimize2
@opindex fbranch-target-load-optimize2
Perform branch target register load optimization after prologue / epilogue
threading.

@item -fbtr-bb-exclusive
@opindex fbtr-bb-exclusive
When performing branch target register load optimization, don't reuse
branch target registers in within any basic block.
```

```
@item -fstack-protector
@opindex fstack-protector
Emit extra code to check for buffer overflows, such as stack smashing
attacks.  This is done by adding a guard variable to functions with
vulnerable objects.  This includes functions that call alloca, and
functions with buffers larger than 8 bytes.  The guards are initialized
when a function is entered and then checked when the function exits.
If a guard check fails, an error message is printed and the program exits.

@item -fstack-protector-all
@opindex fstack-protector-all
Like @option{-fstack-protector} except that all functions are protected.

@item -fsection-anchors
@opindex fsection-anchors
Try to reduce the number of symbolic address calculations by using
shared ``anchor'' symbols to address nearby objects.  This transformation
can help to reduce the number of GOT entries and GOT accesses on some
targets.

For example, the implementation of the following function @code{foo}:

@smallexample
static int a, b, c;
int foo (void) @{ return a + b + c; @}
@end smallexample

would usually calculate the addresses of all three variables, but if you
compile it with @option{-fsection-anchors}, it will access the variables
from a common anchor point instead.  The effect is similar to the
following pseudocode (which isn't valid C):

@smallexample
int foo (void)
@{
  register int *xr = &x;
  return xr[&a - &x] + xr[&b - &x] + xr[&c - &x];
@}
@end smallexample

Not all targets support this option.

@item --param @var{name}=@var{value}
@opindex param
In some places, GCC uses various constants to control the amount of
optimization that is done.  For example, GCC will not inline functions
that contain more that a certain number of instructions.  You can
control some of these constants on the command-line using the
@option{--param} option.

The names of specific parameters, and the meaning of the values, are
tied to the internals of the compiler, and are subject to change
without notice in future releases.

In each case, the @var{value} is an integer.  The allowable choices for
@var{name} are given in the following table:

@table @gcctabopt
@item sra-max-structure-size
The maximum structure size, in bytes, at which the scalar replacement
of aggregates (SRA) optimization will perform block copies.  The
default value, 0, implies that GCC will select the most appropriate
size itself.

@item sra-field-structure-ratio
The threshold ratio (as a percentage) between instantiated fields and
```



```
the complete structure size.  We say that if the ratio of the number
of bytes in instantiated fields to the number of bytes in the complete
structure exceeds this parameter, then block copies are not used.  The
default is 75.

@item struct-reorg-cold-struct-ratio
The threshold ratio (as a percentage) between a structure frequency
and the frequency of the hottest structure in the program.  This parameter
is used by struct-reorg optimization enabled by @option{-fipa-struct-reorg}.
We say that if the ratio of a structure frequency, calculated by profiling,
to the hottest structure frequency in the program is less than this
parameter, then structure reorganization is not applied to this structure.
The default is 10.

@item predictable-branch-cost-outcome
When branch is predicted to be taken with probability lower than this threshold
(in percent), then it is considered well predictable. The default is 10.

@item max-crossjump-edges
The maximum number of incoming edges to consider for crossjumping.
The algorithm used by @option{-fcrossjumping} is @math{O(N^2)} in
the number of edges incoming to each block.  Increasing values mean
more aggressive optimization, making the compile time increase with
probably small improvement in executable size.

@item min-crossjump-insns
The minimum number of instructions which must be matched at the end
of two blocks before crossjumping will be performed on them.  This
value is ignored in the case where all instructions in the block being
crossjumped from are matched.  The default value is 5.

@item max-grow-copy-bb-insns
The maximum code size expansion factor when copying basic blocks
instead of jumping.  The expansion is relative to a jump instruction.
The default value is 8.

@item max-goto-duplication-insns
The maximum number of instructions to duplicate to a block that jumps
to a computed goto.  To avoid @math{O(N^2)} behavior in a number of
passes, GCC factors computed gotos early in the compilation process,
and unfactors them as late as possible.  Only computed jumps at the
end of a basic blocks with no more than max-goto-duplication-insns are
unfactored.  The default value is 8.

@item max-delay-slot-insn-search
The maximum number of instructions to consider when looking for an
instruction to fill a delay slot.  If more than this arbitrary number of
instructions is searched, the time savings from filling the delay slot
will be minimal so stop searching.  Increasing values mean more
aggressive optimization, making the compile time increase with probably
small improvement in executable run time.

@item max-delay-slot-live-search
When trying to fill delay slots, the maximum number of instructions to
consider when searching for a block with valid live register
information.  Increasing this arbitrarily chosen value means more
aggressive optimization, increasing the compile time.  This parameter
should be removed when the delay slot code is rewritten to maintain the
control-flow graph.

@item max-gcse-memory
The approximate maximum amount of memory that will be allocated in
order to perform the global common subexpression elimination
optimization.  If more memory than specified is required, the
optimization will not be done.
```

```
@item max-gcse-passes
The maximum number of passes of GCSE to run.  The default is 1.

@item max-pending-list-length
The maximum number of pending dependencies scheduling will allow
before flushing the current state and starting over.  Large functions
with few branches or calls can create excessively large lists which
needlessly consume memory and resources.

@item max-inline-insns-single
Several parameters control the tree inliner used in gcc.
This number sets the maximum number of instructions (counted in GCC's
internal representation) in a single function that the tree inliner
will consider for inlining.  This only affects functions declared
inline and methods implemented in a class declaration (C++).
The default value is 450.

@item max-inline-insns-auto
When you use @option{-finline-functions} (included in @option{-O3}),
a lot of functions that would otherwise not be considered for inlining
by the compiler will be investigated.  To those functions, a different
(more restrictive) limit compared to functions declared inline can
be applied.
The default value is 90.

@item large-function-insns
The limit specifying really large functions.  For functions larger than this
limit after inlining, inlining is constrained by
@option{--param large-function-growth}.  This parameter is useful primarily
to avoid extreme compilation time caused by non-linear algorithms used by the
backend.
The default value is 2700.

@item large-function-growth
Specifies maximal growth of large function caused by inlining in percents.
The default value is 100 which limits large function growth to 2.0 times
the original size.

@item large-unit-insns
The limit specifying large translation unit.  Growth caused by inlining of
units larger than this limit is limited by @option{--param inline-unit-growth}.
For small units this might be too tight (consider unit consisting of function A
that is inline and B that just calls A three time.  If B is small relative to
A, the growth of unit is 300\% and yet such inlining is very sane.  For very
large units consisting of small inlineable functions however the overall unit
growth limit is needed to avoid exponential explosion of code size.  Thus for
smaller units, the size is increased to @option{--param large-unit-insns}
before applying @option{--param inline-unit-growth}.  The default is 10000

@item inline-unit-growth
Specifies maximal overall growth of the compilation unit caused by inlining.
The default value is 30 which limits unit growth to 1.3 times the original
size.

@item ipcp-unit-growth
Specifies maximal overall growth of the compilation unit caused by
interprocedural constant propagation.  The default value is 10 which limits
unit growth to 1.1 times the original size.

@item large-stack-frame
The limit specifying large stack frames.  While inlining the algorithm is trying
to not grow past this limit too much.  Default value is 256 bytes.

@item large-stack-frame-growth
Specifies maximal growth of large stack frames caused by inlining in percents.
The default value is 1000 which limits large stack frame growth to 11 times
```



```
the original size.

@item max-inline-insns-recursive
@itemx max-inline-insns-recursive-auto
Specifies maximum number of instructions out-of-line copy of self recursive inli
function can grow into by performing recursive inlining.

For functions declared inline @option{--param max-inline-insns-recursive} is
taken into account.  For function not declared inline, recursive inlining
happens only when @option{-finline-functions} (included in @option{-O3}) is
enabled and @option{--param max-inline-insns-recursive-auto} is used.  The
default value is 450.

@item max-inline-recursive-depth
@itemx max-inline-recursive-depth-auto
Specifies maximum recursion depth used by the recursive inlining.

For functions declared inline @option{--param max-inline-recursive-depth} is
taken into account.  For function not declared inline, recursive inlining
happens only when @option{-finline-functions} (included in @option{-O3}) is
enabled and @option{--param max-inline-recursive-depth-auto} is used.  The
default value is 8.

@item min-inline-recursive-probability
Recursive inlining is profitable only for function having deep recursion
in average and can hurt for function having little recursion depth by
increasing the prologue size or complexity of function body to other
optimizers.

When profile feedback is available (see @option{-fprofile-generate}) the actual
recursion depth can be guessed from probability that function will recurse via
given call expression.  This parameter limits inlining only to call expression
whose probability exceeds given threshold (in percents).  The default value is
10.

@item inline-call-cost
Specify cost of call instruction relative to simple arithmetics operations
(having cost of 1).  Increasing this cost disqualifies inlining of non-leaf
functions and at the same time increases size of leaf function that is believed
reduce function size by being inlined.  In effect it increases amount of
inlining for code having large abstraction penalty (many functions that just
pass the arguments to other functions) and decrease inlining for code with low
abstraction penalty.  The default value is 12.

@item min-vect-loop-bound
The minimum number of iterations under which a loop will not get vectorized
when @option{-ftree-vectorize} is used.  The number of iterations after
vectorization needs to be greater than the value specified by this option
to allow vectorization.  The default value is 0.

@item max-unrolled-insns
The maximum number of instructions that a loop should have if that loop
is unrolled, and if the loop is unrolled, it determines how many times
the loop code is unrolled.

@item max-average-unrolled-insns
The maximum number of instructions biased by probabilities of their execution
that a loop should have if that loop is unrolled, and if the loop is unrolled,
it determines how many times the loop code is unrolled.

@item max-unroll-times
The maximum number of unrollings of a single loop.

@item max-peeled-insns
The maximum number of instructions that a loop should have if that loop
is peeled, and if the loop is peeled, it determines how many times
```

```
7417 the loop code is peeled.

7419 @item max-peel-times
7420 The maximum number of peelings of a single loop.

7422 @item max-completely-peeled-insns
7423 The maximum number of insns of a completely peeled loop.

7425 @item max-completely-peel-times
7426 The maximum number of iterations of a loop to be suitable for complete peeling.

7428 @item max-completely-peel-loop-nest-depth
7429 The maximum depth of a loop nest suitable for complete peeling.

7431 @item max-unswitch-insns
7432 The maximum number of insns of an unswitched loop.

7434 @item max-unswitch-level
7435 The maximum number of branches unswitched in a single loop.

7437 @item lim-expensive
7438 The minimum cost of an expensive expression in the loop invariant motion.

7440 @item iv-consider-all-candidates-bound
7441 Bound on number of candidates for induction variables below that
7442 all candidates are considered for each use in induction variable
7443 optimizations.  Only the most relevant candidates are considered
7444 if there are more candidates, to avoid quadratic time complexity.

7446 @item iv-max-considered-uses
7447 The induction variable optimizations give up on loops that contain more
7448 induction variable uses.

7450 @item iv-always-prune-cand-set-bound
7451 If number of candidates in the set is smaller than this value,
7452 we always try to remove unnecessary ivs from the set during its
7453 optimization when a new iv is added to the set.

7455 @item scev-max-expr-size
7456 Bound on size of expressions used in the scalar evolutions analyzer.
7457 Large expressions slow the analyzer.

7459 @item omega-max-vars
7460 The maximum number of variables in an Omega constraint system.
7461 The default value is 128.

7463 @item omega-max-geqs
7464 The maximum number of inequalities in an Omega constraint system.
7465 The default value is 256.

7467 @item omega-max-eqs
7468 The maximum number of equalities in an Omega constraint system.
7469 The default value is 128.

7471 @item omega-max-wild-cards
7472 The maximum number of wildcard variables that the Omega solver will
7473 be able to insert.  The default value is 18.

7475 @item omega-hash-table-size
7476 The size of the hash table in the Omega solver.  The default value is
7477 550.

7479 @item omega-max-keys
7480 The maximal number of keys used by the Omega solver.  The default
7481 value is 500.
```



```
7483 @item omega-eliminate-redundant-constraints
7484 When set to 1, use expensive methods to eliminate all redundant
7485 constraints.  The default value is 0.

7487 @item vect-max-version-for-alignment-checks
7488 The maximum number of runtime checks that can be performed when
7489 doing loop versioning for alignment in the vectorizer.  See option
7490 ftree-vect-loop-version for more information.

7492 @item vect-max-version-for-alias-checks
7493 The maximum number of runtime checks that can be performed when
7494 doing loop versioning for alias in the vectorizer.  See option
7495 ftree-vect-loop-version for more information.

7497 @item max-iterations-to-track

7499 The maximum number of iterations of a loop the brute force algorithm
7500 for analysis of # of iterations of the loop tries to evaluate.

7502 @item hot-bb-count-fraction
7503 Select fraction of the maximal count of repetitions of basic block in program
7504 given basic block needs to have to be considered hot.

7506 @item hot-bb-frequency-fraction
7507 Select fraction of the maximal frequency of executions of basic block in
7508 function given basic block needs to have to be considered hot

7510 @item max-predicted-iterations
7511 The maximum number of loop iterations we predict statically.  This is useful
7512 in cases where function contain single loop with known bound and other loop
7513 with unknown.  We predict the known number of iterations correctly, while
7514 the unknown number of iterations average to roughly 10.  This means that the
7515 loop without bounds would appear artificially cold relative to the other one.

7517 @item align-threshold

7519 Select fraction of the maximal frequency of executions of basic block in
7520 function given basic block will get aligned.

7522 @item align-loop-iterations

7524 A loop expected to iterate at lest the selected number of iterations will get
7525 aligned.

7527 @item tracer-dynamic-coverage
7528 @itemx tracer-dynamic-coverage-feedback

7530 This value is used to limit superblock formation once the given percentage of
7531 executed instructions is covered.  This limits unnecessary code size
7532 expansion.

7534 The @option{tracer-dynamic-coverage-feedback} is used only when profile
7535 feedback is available.  The real profiles (as opposed to statically estimated
7536 ones) are much less balanced allowing the threshold to be larger value.

7538 @item tracer-max-code-growth
7539 Stop tail duplication once code growth has reached given percentage.  This is
7540 rather hokey argument, as most of the duplicates will be eliminated later in
7541 cross jumping, so it may be set to much higher values than is the desired code
7542 growth.

7544 @item tracer-min-branch-ratio

7546 Stop reverse growth when the reverse probability of best edge is less than this
7547 threshold (in percent).
```

```
@item tracer-min-branch-ratio
@itemx tracer-min-branch-ratio-feedback

Stop forward growth if the best edge do have probability lower than this
threshold.

Similarly to @option{tracer-dynamic-coverage} two values are present, one for
compilation for profile feedback and one for compilation without.  The value
for compilation with profile feedback needs to be more conservative (higher) in
order to make tracer effective.

@item max-cse-path-length

Maximum number of basic blocks on path that cse considers.  The default is 10.

@item max-cse-insns
The maximum instructions CSE process before flushing. The default is 1000.

@item max-aliased-vops

Maximum number of virtual operands per function allowed to represent
aliases before triggering the alias partitioning heuristic.  Alias
partitioning reduces compile times and memory consumption needed for
aliasing at the expense of precision loss in alias information.  The
default value for this parameter is 100 for -O1, 500 for -O2 and 1000
for -O3.

Notice that if a function contains more memory statements than the
value of this parameter, it is not really possible to achieve this
reduction.  In this case, the compiler will use the number of memory
statements as the value for @option{max-aliased-vops}.

@item avg-aliased-vops

Average number of virtual operands per statement allowed to represent
aliases before triggering the alias partitioning heuristic.  This
works in conjunction with @option{max-aliased-vops}.  If a function
contains more than @option{max-aliased-vops} virtual operators, then
memory symbols will be grouped into memory partitions until either the
total number of virtual operators is below @option{max-aliased-vops}
or the average number of virtual operators per memory statement is
below @option{avg-aliased-vops}.  The default value for this parameter
is 1 for -O1 and -O2, and 3 for -O3.

@item ggc-min-expand

GCC uses a garbage collector to manage its own memory allocation.  This
parameter specifies the minimum percentage by which the garbage
collector's heap should be allowed to expand between collections.
Tuning this may improve compilation speed; it has no effect on code
generation.

The default is 30% + 70% * (RAM/1GB) with an upper bound of 100% when
RAM >= 1GB@.  If @code{getrlimit} is available, the notion of "RAM" is
the smallest of actual RAM and @code{RLIMIT_DATA} or @code{RLIMIT_AS}.  If
GCC is not able to calculate RAM on a particular platform, the lower
bound of 30% is used.  Setting this parameter and
@option{ggc-min-heapsize} to zero causes a full collection to occur at
every opportunity.  This is extremely slow, but can be useful for
debugging.

@item ggc-min-heapsize

Minimum size of the garbage collector's heap before it begins bothering
to collect garbage.  The first collection occurs after the heap expands
by @option{ggc-min-expand}% beyond @option{ggc-min-heapsize}.  Again,
```



```
tuning this may improve compilation speed, and has no effect on code
generation.

The default is the smaller of RAM/8, RLIMIT_RSS, or a limit which
tries to ensure that RLIMIT_DATA or RLIMIT_AS are not exceeded, but
with a lower bound of 4096 (four megabytes) and an upper bound of
131072 (128 megabytes).  If GCC is not able to calculate RAM on a
particular platform, the lower bound is used.  Setting this parameter
very large effectively disables garbage collection.  Setting this
parameter and @option{ggc-min-expand} to zero causes a full collection
to occur at every opportunity.

@item max-reload-search-insns
The maximum number of instruction reload should look backward for equivalent
register.  Increasing values mean more aggressive optimization, making the
compile time increase with probably slightly better performance.  The default
value is 100.

@item max-cselib-memory-locations
The maximum number of memory locations cselib should take into account.
Increasing values mean more aggressive optimization, making the compile time
increase with probably slightly better performance.  The default value is 500.

@item reorder-blocks-duplicate
@itemx reorder-blocks-duplicate-feedback

Used by basic block reordering pass to decide whether to use unconditional
branch or duplicate the code on its destination.  Code is duplicated when its
estimated size is smaller than this value multiplied by the estimated size of
unconditional jump in the hot spots of the program.

The @option{reorder-block-duplicate-feedback} is used only when profile
feedback is available and may be set to higher values than
@option{reorder-block-duplicate} since information about the hot spots is more
accurate.

@item max-sched-ready-insns
The maximum number of instructions ready to be issued the scheduler should
consider at any given time during the first scheduling pass.  Increasing
values mean more thorough searches, making the compilation time increase
with probably little benefit.  The default value is 100.

@item max-sched-region-blocks
The maximum number of blocks in a region to be considered for
interblock scheduling.  The default value is 10.

@item max-pipeline-region-blocks
The maximum number of blocks in a region to be considered for
pipelining in the selective scheduler.  The default value is 15.

@item max-sched-region-insns
The maximum number of insns in a region to be considered for
interblock scheduling.  The default value is 100.

@item max-pipeline-region-insns
The maximum number of insns in a region to be considered for
pipelining in the selective scheduler.  The default value is 200.

@item min-spec-prob
The minimum probability (in percents) of reaching a source block
for interblock speculative scheduling.  The default value is 40.

@item max-sched-extend-regions-iters
The maximum number of iterations through CFG to extend regions.
0 - disable region extension,
N - do at most N iterations.
```

The default value is 0.

@item max-sched-insn-conflict-delay
The maximum conflict delay for an insn to be considered for speculative motion.
The default value is 3.

@item sched-spec-prob-cutoff
The minimal probability of speculation success (in percents), so that
speculative insn will be scheduled.
The default value is 40.

@item sched-mem-true-dep-cost
Minimal distance (in CPU cycles) between store and load targeting same
memory locations. The default value is 1.

@item selsched-max-lookahead
The maximum size of the lookahead window of selective scheduling. It is a
depth of search for available instructions.
The default value is 50.

@item selsched-max-sched-times
The maximum number of times that an instruction will be scheduled during
selective scheduling. This is the limit on the number of iterations
through which the instruction may be pipelined. The default value is 2.

@item selsched-max-insns-to-rename
The maximum number of best instructions in the ready list that are considered
for renaming in the selective scheduler. The default value is 2.

@item max-last-value-rtl
The maximum size measured as number of RTLs that can be recorded in an expressio
in combiner for a pseudo register as last known value of that register. The def
is 10000.

@item integer-share-limit
Small integer constants can use a shared data structure, reducing the
compiler’s memory usage and increasing its speed. This sets the maximum
value of a shared integer constant. The default value is 256.

@item min-virtual-mappings
Specifies the minimum number of virtual mappings in the incremental
SSA updater that should be registered to trigger the virtual mappings
heuristic defined by virtual-mappings-ratio. The default value is
100.

@item virtual-mappings-ratio
If the number of virtual mappings is virtual-mappings-ratio bigger
than the number of virtual symbols to be updated, then the incremental
SSA updater switches to a full update for those symbols. The default
ratio is 3.

@item ssp-buffer-size
The minimum size of buffers (i.e.@: arrays) that will receive stack smashing
protection when @option{-fstack-protection} is used.

@item max-jump-thread-duplication-stmts
Maximum number of statements allowed in a block that needs to be
duplicated when threading jumps.

@item max-fields-for-field-sensitive
Maximum number of fields in a structure we will treat in
a field sensitive manner during pointer analysis. The default is zero
for -O0, and -O1 and 100 for -Os, -O2, and -O3.

@item prefetch-latency
Estimate on average number of instructions that are executed before

prefetch finishes. The distance we prefetch ahead is proportional
to this constant. Increasing this number may also lead to less
streams being prefetched (see @option{simultaneous-prefetches}).

@item simultaneous-prefetches
Maximum number of prefetches that can run at the same time.

@item l1-cache-line-size
The size of cache line in L1 cache, in bytes.

@item l1-cache-size
The size of L1 cache, in kilobytes.

@item l2-cache-size
The size of L2 cache, in kilobytes.

@item use-canonical-types
Whether the compiler should use the ‘‘canonical’’ type system. By
default, this should always be 1, which uses a more efficient internal
mechanism for comparing types in C++ and Objective-C++. However, if
bugs in the canonical type system are causing compilation failures,
set this value to 0 to disable canonical types.

@item switch-conversion-max-branch-ratio
Switch initialization conversion will refuse to create arrays that are
bigger than @option{switch-conversion-max-branch-ratio} times the number of
branches in the switch.

@item max-partial-antic-length
Maximum length of the partial antic set computed during the tree
partial redundancy elimination optimization (@option{-ftree-pre}) when
optimizing at @option{-O3} and above. For some sorts of source code
the enhanced partial redundancy elimination optimization can run away,
consuming all of the memory available on the host machine. This
parameter sets a limit on the length of the sets that are computed,
which prevents the runaway behavior. Setting a value of 0 for
this parameter will allow an unlimited set length.

@item sccvn-max-scc-size
Maximum size of a strongly connected component (SCC) during SCCVN
processing. If this limit is hit, SCCVN processing for the whole
function will not be done and optimizations depending on it will
be disabled. The default maximum SCC size is 10000.

@item ira-max-loops-num
IRA uses a regional register allocation by default. If a function
contains loops more than number given by the parameter, only at most
given number of the most frequently executed loops will form regions
for the regional register allocation. The default value of the
parameter is 100.

@item ira-max-conflict-table-size
Although IRA uses a sophisticated algorithm of compression conflict
table, the table can be still big for huge functions. If the conflict
table for a function could be more than size in MB given by the
parameter, the conflict table is not built and faster, simpler, and
lower quality register allocation algorithm will be used. The
algorithm do not use pseudo-register conflicts. The default value of
the parameter is 2000.

@item loop-invariant-max-bbs-in-loop
Loop invariant motion can be very expensive, both in compile time and
in amount of needed compile time memory, with very large loops. Loops
with more basic blocks than this parameter won’t have loop invariant
motion optimization performed on them. The default value of the
parameter is 1000 for -O1 and 10000 for -O2 and above.

```
@end table
@end table

@node Preprocessor Options
@section Options Controlling the Preprocessor
@cindex preprocessor options
@cindex options, preprocessor

These options control the C preprocessor, which is run on each C source
file before actual compilation.

If you use the @option{-E} option, nothing is done except preprocessing.
Some of these options make sense only together with @option{-E} because
they cause the preprocessor output to be unsuitable for actual
compilation.

@table @gcctabopt
@item -Wp,@var{option}
@opindex Wp
You can use @option{-Wp,@var{option}} to bypass the compiler driver
and pass @var{option} directly through to the preprocessor.  If
@var{option} contains commas, it is split into multiple options at the
commas.  However, many options are modified, translated or interpreted
by the compiler driver before being passed to the preprocessor, and
@option{-Wp} forcibly bypasses this phase.  The preprocessor's direct
interface is undocumented and subject to change, so whenever possible
you should avoid using @option{-Wp} and let the driver handle the
options instead.

@item -Xpreprocessor @var{option}
@opindex Xpreprocessor
Pass @var{option} as an option to the preprocessor.  You can use this to
supply system-specific preprocessor options which GCC does not know how to
recognize.

If you want to pass an option that takes an argument, you must use
@option{-Xpreprocessor} twice, once for the option and once for the argument.
@end table

@include cppopts.texi

@node Assembler Options
@section Passing Options to the Assembler

@c prevent bad page break with this line
You can pass options to the assembler.

@table @gcctabopt
@item -Wa,@var{option}
@opindex Wa
Pass @var{option} as an option to the assembler.  If @var{option}
contains commas, it is split into multiple options at the commas.

@item -Xassembler @var{option}
@opindex Xassembler
Pass @var{option} as an option to the assembler.  You can use this to
supply system-specific assembler options which GCC does not know how to
recognize.

If you want to pass an option that takes an argument, you must use
@option{-Xassembler} twice, once for the option and once for the argument.

@end table

@node Link Options
```



```
@section Options for Linking
@cindex link options
@cindex options, linking

These options come into play when the compiler links object files into
an executable output file.  They are meaningless if the compiler is
not doing a link step.

@table @gcctabopt
@cindex file names
@item @var{object-file-name}
A file name that does not end in a special recognized suffix is
considered to name an object file or library.  (Object files are
distinguished from libraries by the linker according to the file
contents.)  If linking is done, these object files are used as input
to the linker.

@item -c
@itemx -S
@itemx -E
@opindex c
@opindex S
@opindex E
If any of these options is used, then the linker is not run, and
object file names should not be used as arguments.  @xref{Overall
Options}.

@cindex Libraries
@item -l@var{library}
@itemx -l @var{library}
@opindex l
Search the library named @var{library} when linking.  (The second
alternative with the library as a separate argument is only for
POSIX compliance and is not recommended.)

It makes a difference where in the command you write this option; the
linker searches and processes libraries and object files in the order they
are specified.  Thus, @samp{foo.o -lz bar.o} searches library @samp{z}
after file @file{foo.o} but before @file{bar.o}.  If @file{bar.o} refers
to functions in @samp{z}, those functions may not be loaded.

The linker searches a standard list of directories for the library,
which is actually a file named @file{lib@var{library}.a}.  The linker
then uses this file as if it had been specified precisely by name.

The directories searched include several standard system directories
plus any that you specify with @option{-L}.

Normally the files found this way are library files---archive files
whose members are object files.  The linker handles an archive file by
scanning through it for members which define symbols that have so far
been referenced but not defined.  But if the file that is found is an
ordinary object file, it is linked in the usual fashion.  The only
difference between using an @option{-l} option and specifying a file name
is that @option{-l} surrounds @var{library} with @samp{lib} and @samp{.a}
and searches several directories.

@item -lobjc
@opindex lobjc
You need this special case of the @option{-l} option in order to
link an Objective-C or Objective-C++ program.

@item -nostartfiles
@opindex nostartfiles
Do not use the standard system startup files when linking.
The standard system libraries are used normally, unless @option{-nostdlib}
```

```
or @option{-nodefaultlibs} is used.

@item -nodefaultlibs
@opindex nodefaultlibs
Do not use the standard system libraries when linking.
Only the libraries you specify will be passed to the linker.
The standard startup files are used normally, unless @option{-nostartfiles}
is used.  The compiler may generate calls to @code{memcmp},
@code{memset}, @code{memcpy} and @code{memmove}.
These entries are usually resolved by entries in
libc.  These entry points should be supplied through some other
mechanism when this option is specified.

@item -nostdlib
@opindex nostdlib
Do not use the standard system startup files or libraries when linking.
No startup files and only the libraries you specify will be passed to
the linker.  The compiler may generate calls to @code{memcmp}, @code{memset},
@code{memcpy} and @code{memmove}.
These entries are usually resolved by entries in
libc.  These entry points should be supplied through some other
mechanism when this option is specified.

@cindex @option{-lgcc}, use with @option{-nostdlib}
@cindex @option{-nostdlib} and unresolved references
@cindex unresolved references and @option{-nostdlib}
@cindex @option{-lgcc}, use with @option{-nodefaultlibs}
@cindex @option{-nodefaultlibs} and unresolved references
@cindex unresolved references and @option{-nodefaultlibs}
One of the standard libraries bypassed by @option{-nostdlib} and
@option{-nodefaultlibs} is @file{libgcc.a}, a library of internal subroutines
that GCC uses to overcome shortcomings of particular machines, or special
needs for some languages.
(@xref{Interface,,Interfacing to GCC Output,gccint,GNU Compiler
Collection (GCC) Internals},
for more discussion of @file{libgcc.a}.)
In most cases, you need @file{libgcc.a} even when you want to avoid
other standard libraries.  In other words, when you specify @option{-nostdlib}
or @option{-nodefaultlibs} you should usually specify @option{-lgcc} as well.
This ensures that you have no unresolved references to internal GCC
library subroutines.  (For example, @samp{__main}, used to ensure C++
constructors will be called; @pxref{Collect2,,@code{collect2}, gccint,
GNU Compiler Collection (GCC) Internals}.)

@item -pie
@opindex pie
Produce a position independent executable on targets which support it.
For predictable results, you must also specify the same set of options
that were used to generate code (@option{-fpie}, @option{-fPIE},
or model suboptions) when you specify this option.

@item -rdynamic
@opindex rdynamic
Pass the flag @option{-export-dynamic} to the ELF linker, on targets
that support it. This instructs the linker to add all symbols, not
only used ones, to the dynamic symbol table. This option is needed
for some uses of @code{dlopen} or to allow obtaining backtraces
from within a program.

@item -s
@opindex s
Remove all symbol table and relocation information from the executable.

@item -static
@opindex static
On systems that support dynamic linking, this prevents linking with the shared
```



```
libraries.  On other systems, this option has no effect.

@item -shared
@opindex shared
Produce a shared object which can then be linked with other objects to
form an executable.  Not all systems support this option.  For predictable
results, you must also specify the same set of options that were used to
generate code (@option{-fpic}, @option{-fPIC}, or model suboptions)
when you specify this option.@footnote{On some systems, @samp{gcc -shared}
needs to build supplementary stub code for constructors to work.  On
multi-libbed systems, @samp{gcc -shared} must select the correct support
libraries to link against.  Failing to supply the correct flags may lead
to subtle defects.  Supplying them in cases where they are not necessary
is innocuous.}

@item -shared-libgcc
@itemx -static-libgcc
@opindex shared-libgcc
@opindex static-libgcc
On systems that provide @file{libgcc} as a shared library, these options
force the use of either the shared or static version respectively.
If no shared version of @file{libgcc} was built when the compiler was
configured, these options have no effect.

There are several situations in which an application should use the
shared @file{libgcc} instead of the static version.  The most common
of these is when the application wishes to throw and catch exceptions
across different shared libraries.  In that case, each of the libraries
as well as the application itself should use the shared @file{libgcc}.

Therefore, the G++ and GCJ drivers automatically add
@option{-shared-libgcc} whenever you build a shared library or a main
executable, because C++ and Java programs typically use exceptions, so
this is the right thing to do.

If, instead, you use the GCC driver to create shared libraries, you may
find that they will not always be linked with the shared @file{libgcc}.
If GCC finds, at its configuration time, that you have a non-GNU linker
or a GNU linker that does not support option @option{--eh-frame-hdr},
it will link the shared version of @file{libgcc} into shared libraries
by default.  Otherwise, it will take advantage of the linker and optimize
away the linking with the shared version of @file{libgcc}, linking with
the static version of libgcc by default.  This allows exceptions to
propagate through such shared libraries, without incurring relocation
costs at library load time.

However, if a library or main executable is supposed to throw or catch
exceptions, you must link it using the G++ or GCJ driver, as appropriate
for the languages used in the program, or using the option
@option{-shared-libgcc}, such that it is linked with the shared
@file{libgcc}.

@item -symbolic
@opindex symbolic
Bind references to global symbols when building a shared object.  Warn
about any unresolved references (unless overridden by the link editor
option @samp{-Xlinker -z -Xlinker defs}).  Only a few systems support
this option.

@item -T @var{script}
@opindex T
@cindex linker script
Use @var{script} as the linker script.  This option is supported by most
systems using the GNU linker.  On some targets, such as bare-board
targets without an operating system, the @option{-T} option may be required
when linking to avoid references to undefined symbols.
```

```
@item -Xlinker @var{option}
@opindex Xlinker
Pass @var{option} as an option to the linker.  You can use this to
supply system-specific linker options which GCC does not know how to
recognize.

If you want to pass an option that takes a separate argument, you must use
@option{-Xlinker} twice, once for the option and once for the argument.
For example, to pass @option{-assert definitions}, you must write
@samp{-Xlinker -assert -Xlinker definitions}.  It does not work to write
@option{-Xlinker "-assert definitions"}, because this passes the entire
string as a single argument, which is not what the linker expects.

When using the GNU linker, it is usually more convenient to pass
arguments to linker options using the @option{@var{option}=@var{value}}
syntax than as separate arguments.  For example, you can specify
@samp{-Xlinker -Map=output.map} rather than
@samp{-Xlinker -Map -Xlinker output.map}.  Other linkers may not support
this syntax for command-line options.

@item -Wl,@var{option}
@opindex Wl
Pass @var{option} as an option to the linker.  If @var{option} contains
commas, it is split into multiple options at the commas.  You can use this
syntax to pass an argument to the option.
For example, @samp{-Wl,-Map,output.map} passes @samp{-Map output.map} to the
linker.  When using the GNU linker, you can also get the same effect with
@samp{-Wl,-Map=output.map}.

@item -u @var{symbol}
@opindex u
Pretend the symbol @var{symbol} is undefined, to force linking of
library modules to define it.  You can use @option{-u} multiple times with
different symbols to force loading of additional library modules.
@end table

@node Directory Options
@section Options for Directory Search
@cindex directory options
@cindex options, directory search
@cindex search path

These options specify directories to search for header files, for
libraries and for parts of the compiler:

@table @gcctabopt
@item -I@var{dir}
@opindex I
Add the directory @var{dir} to the head of the list of directories to be
searched for header files.  This can be used to override a system header
file, substituting your own version, since these directories are
searched before the system header file directories.  However, you should
not use this option to add directories that contain vendor-supplied
system header files (use @option{-isystem} for that).  If you use more than
one @option{-I} option, the directories are scanned in left-to-right
order; the standard system directories come after.

If a standard system include directory, or a directory specified with
@option{-isystem}, is also specified with @option{-I}, the @option{-I}
option will be ignored.  The directory will still be searched but as a
system directory at its normal position in the system include chain.
This is to ensure that GCC's procedure to fix buggy system headers and
the ordering for the include_next directive are not inadvertently changed.
If you really need to change the search order for system directories,
use the @option{-nostdinc} and/or @option{-isystem} options.
```



```
@item -iquote@var{dir}
@opindex iquote
Add the directory @var{dir} to the head of the list of directories to
be searched for header files only for the case of @samp{#include
"@var{file}"}; they are not searched for @samp{#include <@var{file}>},
otherwise just like @option{-I}.

@item -L@var{dir}
@opindex L
Add directory @var{dir} to the list of directories to be searched
for @option{-l}.

@item -B@var{prefix}
@opindex B
This option specifies where to find the executables, libraries,
include files, and data files of the compiler itself.

The compiler driver program runs one or more of the subprograms
@file{cpp}, @file{cc1}, @file{as} and @file{ld}.  It tries
@var{prefix} as a prefix for each program it tries to run, both with and
without @samp{@var{machine}/@var{version}/} (@pxref{Target Options}).

For each subprogram to be run, the compiler driver first tries the
@option{-B} prefix, if any.  If that name is not found, or if @option{-B}
was not specified, the driver tries two standard prefixes, which are
@file{/usr/lib/gcc/} and @file{/usr/local/lib/gcc/}.  If neither of
those results in a file name that is found, the unmodified program
name is searched for using the directories specified in your
@env{PATH} environment variable.

The compiler will check to see if the path provided by the @option{-B}
refers to a directory, and if necessary it will add a directory
separator character at the end of the path.

@option{-B} prefixes that effectively specify directory names also apply
to libraries in the linker, because the compiler translates these
options into @option{-L} options for the linker.  They also apply to
includes files in the preprocessor, because the compiler translates these
options into @option{-isystem} options for the preprocessor.  In this case,
the compiler appends @samp{include} to the prefix.

The run-time support file @file{libgcc.a} can also be searched for using
the @option{-B} prefix, if needed.  If it is not found there, the two
standard prefixes above are tried, and that is all.  The file is left
out of the link if it is not found by those means.

Another way to specify a prefix much like the @option{-B} prefix is to use
the environment variable @env{GCC_EXEC_PREFIX}.  @xref{Environment
Variables}.

As a special kludge, if the path provided by @option{-B} is
@file{[dir/]stage@var{N}/}, where @var{N} is a number in the range 0 to
9, then it will be replaced by @file{[dir/]include}.  This is to help
with boot-strapping the compiler.

@item -specs=@var{file}
@opindex specs
Process @var{file} after the compiler reads in the standard @file{specs}
file, in order to override the defaults that the @file{gcc} driver
program uses when determining what switches to pass to @file{cc1},
@file{cc1plus}, @file{as}, @file{ld}, etc.  More than one
@option{-specs=@var{file}} can be specified on the command line, and they
are processed in order, from left to right.

@item --sysroot=@var{dir}
```

```
@opindex sysroot
Use @var{dir} as the logical root directory for headers and libraries.
For example, if the compiler would normally search for headers in
@file{/usr/include} and libraries in @file{/usr/lib}, it will instead
search @file{@var{dir}/usr/include} and @file{@var{dir}/usr/lib}.

If you use both this option and the @option{-isysroot} option, then
the @option{--sysroot} option will apply to libraries, but the
@option{-isysroot} option will apply to header files.

The GNU linker (beginning with version 2.16) has the necessary support
for this option.  If your linker does not support this option, the
header file aspect of @option{--sysroot} will still work, but the
library aspect will not.

@item -I-
@opindex I-
This option has been deprecated.  Please use @option{-iquote} instead for
@option{-I} directories before the @option{-I-} and remove the @option{-I-}.
Any directories you specify with @option{-I} options before the @option{-I-}
option are searched only for the case of @samp{#include "@var{file}"};
they are not searched for @samp{#include <@var{file}>}.

If additional directories are specified with @option{-I} options after
the @option{-I-}, these directories are searched for all @samp{#include}
directives.  (Ordinarily @emph{all} @option{-I} directories are used
this way.)

In addition, the @option{-I-} option inhibits the use of the current
directory (where the current input file came from) as the first search
directory for @samp{#include "@var{file}"}.  There is no way to
override this effect of @option{-I-}.  With @option{-I.} you can specify
searching the directory which was current when the compiler was
invoked.  That is not exactly the same as what the preprocessor does
by default, but it is often satisfactory.

@option{-I-} does not inhibit the use of the standard system directories
for header files.  Thus, @option{-I-} and @option{-nostdinc} are
independent.
@end table

@c man end

@node Spec Files
@section Specifying subprocesses and the switches to pass to them
@cindex Spec Files

@command{gcc} is a driver program.  It performs its job by invoking a
sequence of other programs to do the work of compiling, assembling and
linking.  GCC interprets its command-line parameters and uses these to
deduce which programs it should invoke, and which command-line options
it ought to place on their command lines.  This behavior is controlled
by @dfn{spec strings}.  In most cases there is one spec string for each
program that GCC can invoke, but a few programs have multiple spec
strings to control their behavior.  The spec strings built into GCC can
be overridden by using the @option{-specs=} command-line switch to specify
a spec file.

@dfn{Spec files} are plaintext files that are used to construct spec
strings.  They consist of a sequence of directives separated by blank
lines.  The type of directive is determined by the first non-whitespace
character on the line and it can be one of the following:

@table @code
@item %@var{command}
Issues a @var{command} to the spec file processor.  The commands that can
```



```
appear here are:

@table @code
@item %include <@var{file}>
@cindex %include
Search for @var{file} and insert its text at the current point in the
specs file.

@item %include_noerr <@var{file}>
@cindex %include_noerr
Just like @samp{%include}, but do not generate an error message if the include
file cannot be found.

@item %rename @var{old_name} @var{new_name}
@cindex %rename
Rename the spec string @var{old_name} to @var{new_name}.

@end table

@item *[@var{spec_name}]:
This tells the compiler to create, override or delete the named spec
string.  All lines after this directive up to the next directive or
blank line are considered to be the text for the spec string.  If this
results in an empty string then the spec will be deleted.  (Or, if the
spec did not exist, then nothing will happened.)  Otherwise, if the spec
does not currently exist a new spec will be created.  If the spec does
exist then its contents will be overridden by the text of this
directive, unless the first character of that text is the @samp{+}
character, in which case the text will be appended to the spec.

@item [@var{suffix}]:
Creates a new @samp{[@var{suffix}] spec} pair.  All lines after this directive
and up to the next directive or blank line are considered to make up the
spec string for the indicated suffix.  When the compiler encounters an
input file with the named suffix, it will processes the spec string in
order to work out how to compile that file.  For example:

@smallexample
.ZZ:
z-compile -input %i
@end smallexample

This says that any input file whose name ends in @samp{.ZZ} should be
passed to the program @samp{z-compile}, which should be invoked with the
command-line switch @option{-input} and with the result of performing the
@samp{%i} substitution.  (See below.)

As an alternative to providing a spec string, the text that follows a
suffix directive can be one of the following:

@table @code
@item @@@var{language}
This says that the suffix is an alias for a known @var{language}.  This is
similar to using the @option{-x} command-line switch to GCC to specify a
language explicitly.  For example:

@smallexample
.ZZ:
@@c++
@end smallexample

Says that .ZZ files are, in fact, C++ source files.

@item #@var{name}
This causes an error messages saying:
```

```
@smallexample
@var{name} compiler not installed on this system.
@end smallexample
@end table

GCC already has an extensive list of suffixes built into it.
This directive will add an entry to the end of the list of suffixes, but
since the list is searched from the end backwards, it is effectively
possible to override earlier entries using this technique.

@end table

GCC has the following spec strings built into it.  Spec files can
override these strings or create their own.  Note that individual
targets can also add their own spec strings to this list.

@smallexample
asm          Options to pass to the assembler
asm_final    Options to pass to the assembler post-processor
cpp          Options to pass to the C preprocessor
cc1          Options to pass to the C compiler
cc1plus      Options to pass to the C++ compiler
endfile      Object files to include at the end of the link
link         Options to pass to the linker
lib          Libraries to include on the command line to the linker
libgcc       Decides which GCC support library to pass to the linker
linker       Sets the name of the linker
predefines   Defines to be passed to the C preprocessor
signed_char  Defines to pass to CPP to say whether @code{char} is signed
             by default
startfile    Object files to include at the start of the link
@end smallexample

Here is a small example of a spec file:

@smallexample
%rename lib                 old_lib

*lib:
--start-group -lgcc -lc -leval1 --end-group %(old_lib)
@end smallexample

This example renames the spec called @samp{lib} to @samp{old_lib} and
then overrides the previous definition of @samp{lib} with a new one.
The new definition adds in some extra command-line options before
including the text of the old definition.

@dfn{Spec strings} are a list of command-line options to be passed to their
corresponding program.  In addition, the spec strings can contain
@samp{%}-prefixed sequences to substitute variable text or to
conditionally insert text into the command line.  Using these constructs
it is possible to generate quite complex command lines.

Here is a table of all defined @samp{%}-sequences for spec
strings.  Note that spaces are not generated automatically around the
results of expanding these sequences.  Therefore you can concatenate them
together or combine them with constant text in a single argument.

@table @code
@item %%
Substitute one @samp{%} into the program name or argument.

@item %i
Substitute the name of the input file being processed.

@item %b
```



```
Substitute the basename of the input file being processed.
This is the substring up to (and not including) the last period
and not including the directory.

@item %B
This is the same as @samp{%b}, but include the file suffix (text after
the last period).

@item %d
Marks the argument containing or following the @samp{%d} as a
temporary file name, so that that file will be deleted if GCC exits
successfully.  Unlike @samp{%g}, this contributes no text to the
argument.

@item %g@var{suffix}
Substitute a file name that has suffix @var{suffix} and is chosen
once per compilation, and mark the argument in the same way as
@samp{%d}.  To reduce exposure to denial-of-service attacks, the file
name is now chosen in a way that is hard to predict even when previously
chosen file names are known.  For example, @samp{%g.s @dots{} %g.o @dots{} %g.s}
might turn into @samp{ccUVUUAU.s ccXYAXZ12.o ccUVUUAU.s}.  @var{suffix} matches
the regexp @samp{[.A-Za-z]*} or the special string @samp{%O}, which is
treated exactly as if @samp{%O} had been preprocessed.  Previously, @samp{%g}
was simply substituted with a file name chosen once per compilation,
without regard to any appended suffix (which was therefore treated
just like ordinary text), making such attacks more likely to succeed.

@item %u@var{suffix}
Like @samp{%g}, but generates a new temporary file name even if
@samp{%u@var{suffix}} was already seen.

@item %U@var{suffix}
Substitutes the last file name generated with @samp{%u@var{suffix}}, generating
new one if there is no such last file name.  In the absence of any
@samp{%u@var{suffix}}, this is just like @samp{%g@var{suffix}}, except they don’
the same suffix @emph{space}, so @samp{%g.s @dots{} %U.s @dots{} %g.s @dots{} %U
would involve the generation of two distinct file names, one
for each @samp{%g.s} and another for each @samp{%U.s}.  Previously, @samp{%U} wa
simply substituted with a file name chosen for the previous @samp{%u},
without regard to any appended suffix.

@item %j@var{suffix}
Substitutes the name of the @code{HOST_BIT_BUCKET}, if any, and if it is
writable, and if save-temps is off; otherwise, substitute the name
of a temporary file, just like @samp{%u}.  This temporary file is not
meant for communication between processes, but rather as a junk
disposal mechanism.

@item %|@var{suffix}
@itemx %m@var{suffix}
Like @samp{%g}, except if @option{-pipe} is in effect.  In that case
@samp{%|} substitutes a single dash and @samp{%m} substitutes nothing at
all.  These are the two most common ways to instruct a program that it
should read from standard input or write to standard output.  If you
need something more elaborate you can use an @samp{%@{pipe:@code{X}@}}
construct: see for example @file{f/lang-specs.h}.

@item %.@var{SUFFIX}
Substitutes @var{.SUFFIX} for the suffixes of a matched switch’s args
when it is subsequently output with @samp{%*}.  @var{SUFFIX} is
terminated by the next space or %.

@item %w
Marks the argument containing or following the @samp{%w} as the
designated output file of this compilation.  This puts the argument
into the sequence of arguments that @samp{%o} will substitute later.
```

```
@item %o
Substitutes the names of all the output files, with spaces
automatically placed around them.  You should write spaces
around the @samp{%o} as well or the results are undefined.
@samp{%o} is for use in the specs for running the linker.
Input files whose names have no recognized suffix are not compiled
at all, but they are included among the output files, so they will
be linked.

@item %O
Substitutes the suffix for object files.  Note that this is
handled specially when it immediately follows @samp{%g, %u, or %U},
because of the need for those to form complete file names.  The
handling is such that @samp{%O} is treated exactly as if it had already
been substituted, except that @samp{%g, %u, and %U} do not currently
support additional @var{suffix} characters following @samp{%O} as they would
following, for example, @samp{.o}.

@item %p
Substitutes the standard macro predefinitions for the
current target machine.  Use this when running @code{cpp}.

@item %P
Like @samp{%p}, but puts @samp{__} before and after the name of each
predefined macro, except for macros that start with @samp{__} or with
@samp{_@var{L}}, where @var{L} is an uppercase letter.  This is for ISO
C@.

@item %I
Substitute any of @option{-iprefix} (made from @env{GCC_EXEC_PREFIX}),
@option{-isysroot} (made from @env{TARGET_SYSTEM_ROOT}),
@option{-isystem} (made from @env{COMPILER_PATH} and @option{-B} options)
and @option{-imultilib} as necessary.

@item %s
Current argument is the name of a library or startup file of some sort.
Search for that file in a standard list of directories and substitute
the full name found.

@item %e@var{str}
Print @var{str} as an error message.  @var{str} is terminated by a newline.
Use this when inconsistent options are detected.

@item %(@var{name})
Substitute the contents of spec string @var{name} at this point.

@item %[@var{name}]
Like @samp{%(@dots{})} but put @samp{__} around @option{-D} arguments.

@item %x@{@var{option}@}
Accumulate an option for @samp{%X}.

@item %X
Output the accumulated linker options specified by @option{-Wl} or a @samp{%x}
spec string.

@item %Y
Output the accumulated assembler options specified by @option{-Wa}.

@item %Z
Output the accumulated preprocessor options specified by @option{-Wp}.

@item %a
Process the @code{asm} spec.  This is used to compute the
switches to be passed to the assembler.
```



```
@item %A
Process the @code{asm_final} spec.  This is a spec string for
passing switches to an assembler post-processor, if such a program is
needed.

@item %l
Process the @code{link} spec.  This is the spec for computing the
command line passed to the linker.  Typically it will make use of the
@samp{%L %G %S %D and %E} sequences.

@item %D
Dump out a @option{-L} option for each directory that GCC believes might
contain startup files.  If the target supports multilibs then the
current multilib directory will be prepended to each of these paths.

@item %L
Process the @code{lib} spec.  This is a spec string for deciding which
libraries should be included on the command line to the linker.

@item %G
Process the @code{libgcc} spec.  This is a spec string for deciding
which GCC support library should be included on the command line to the linker.

@item %S
Process the @code{startfile} spec.  This is a spec for deciding which
object files should be the first ones passed to the linker.  Typically
this might be a file named @file{crt0.o}.

@item %E
Process the @code{endfile} spec.  This is a spec string that specifies
the last object files that will be passed to the linker.

@item %C
Process the @code{cpp} spec.  This is used to construct the arguments
to be passed to the C preprocessor.

@item %1
Process the @code{cc1} spec.  This is used to construct the options to be
passed to the actual C compiler (@samp{cc1}).

@item %2
Process the @code{cc1plus} spec.  This is used to construct the options to be
passed to the actual C++ compiler (@samp{cc1plus}).

@item %*
Substitute the variable part of a matched option.  See below.
Note that each comma in the substituted string is replaced by
a single space.

@item %<@code{S}
Remove all occurrences of @code{-S} from the command line.  Note---this
command is position dependent.  @samp{%} commands in the spec string
before this one will see @code{-S}, @samp{%} commands in the spec string
after this one will not.

@item %:@var{function}(@var{args})
Call the named function @var{function}, passing it @var{args}.
@var{args} is first processed as a nested spec string, then split
into an argument vector in the usual fashion.  The function returns
a string which is processed as if it had appeared literally as part
of the current spec.

The following built-in spec functions are provided:

@table @code
```

```
@item @code{getenv}
The @code{getenv} spec function takes two arguments: an environment
variable name and a string.  If the environment variable is not
defined, a fatal error is issued.  Otherwise, the return value is the
value of the environment variable concatenated with the string.  For
example, if @env{TOPDIR} is defined as @file{/path/to/top}, then:

@smallexample
%:getenv(TOPDIR /include)
@end smallexample

expands to @file{/path/to/top/include}.

@item @code{if-exists}
The @code{if-exists} spec function takes one argument, an absolute
pathname to a file.  If the file exists, @code{if-exists} returns the
pathname.  Here is a small example of its usage:

@smallexample
*startfile:
crt0%O%s %:if-exists(crti%O%s) crtbegin%O%s
@end smallexample

@item @code{if-exists-else}
The @code{if-exists-else} spec function is similar to the @code{if-exists}
spec function, except that it takes two arguments.  The first argument is
an absolute pathname to a file.  If the file exists, @code{if-exists-else}
returns the pathname.  If it does not exist, it returns the second argument.
This way, @code{if-exists-else} can be used to select one file or another,
based on the existence of the first.  Here is a small example of its usage:

@smallexample
*startfile:
crt0%O%s %:if-exists(crti%O%s) \
%:if-exists-else(crtbeginT%O%s crtbegin%O%s)
@end smallexample

@item @code{replace-outfile}
The @code{replace-outfile} spec function takes two arguments.  It looks for the
first argument in the outfiles array and replaces it with the second argument.
is a small example of its usage:

@smallexample
%@{fgnu-runtime:%:replace-outfile(-lobjc -lobjc-gnu)@}
@end smallexample

@item @code{print-asm-header}
The @code{print-asm-header} function takes no arguments and simply
prints a banner like:

@smallexample
Assembler options
=================

Use "-Wa,OPTION" to pass "OPTION" to the assembler.
@end smallexample

It is used to separate compiler options from assembler options
in the @option{--target-help} output.
@end table

@item %@{@code{S}@}
Substitutes the @code{-S} switch, if that switch was given to GCC@.
If that switch was not specified, this substitutes nothing.  Note that
the leading dash is omitted when specifying this option, and it is
automatically inserted if the substitution is performed.  Thus the spec
```



```
string @samp{%@{foo@}} would match the command-line option @option{-foo}
and would output the command line option @option{-foo}.

@item %W@{@code{S}@}
Like %@{@code{S}@} but mark last argument supplied within as a file to be
deleted on failure.

@item %@{@code{S}*@}
Substitutes all the switches specified to GCC whose names start
with @code{-S}, but which also take an argument.  This is used for
switches like @option{-o}, @option{-D}, @option{-I}, etc.
GCC considers @option{-o foo} as being
one switch whose names starts with @samp{o}.  %@{o*@} would substitute this
text, including the space.  Thus two arguments would be generated.

@item %@{@code{S}*&@code{T}*@}
Like %@{@code{S}*@}, but preserve order of @code{S} and @code{T} options
(the order of @code{S} and @code{T} in the spec is not significant).
There can be any number of ampersand-separated variables; for each the
wild card is optional.  Useful for CPP as @samp{%@{D*&U*&A*@}}.

@item %@{@code{S}:@code{X}@}
Substitutes @code{X}, if the @samp{-S} switch was given to GCC@.

@item %@{!@code{S}:@code{X}@}
Substitutes @code{X}, if the @samp{-S} switch was @emph{not} given to GCC@.

@item %@{@code{S}*:@code{X}@}
Substitutes @code{X} if one or more switches whose names start with
@code{-S} are specified to GCC@.  Normally @code{X} is substituted only
once, no matter how many such switches appeared.  However, if @code{%*}
appears somewhere in @code{X}, then @code{X} will be substituted once
for each matching switch, with the @code{%*} replaced by the part of
that switch that matched the @code{*}.

@item %@{.@code{S}:@code{X}@}
Substitutes @code{X}, if processing a file with suffix @code{S}.

@item %@{!.@code{S}:@code{X}@}
Substitutes @code{X}, if @emph{not} processing a file with suffix @code{S}.

@item %@{,@code{S}:@code{X}@}
Substitutes @code{X}, if processing a file for language @code{S}.

@item %@{!,@code{S}:@code{X}@}
Substitutes @code{X}, if not processing a file for language @code{S}.

@item %@{@code{S}|@code{P}:@code{X}@}
Substitutes @code{X} if either @code{-S} or @code{-P} was given to
GCC@.  This may be combined with @samp{!}, @samp{.}, @samp{,}, and
@code{*} sequences as well, although they have a stronger binding than
the @samp{|}.  If @code{%*} appears in @code{X}, all of the
alternatives must be starred, and only the first matching alternative
is substituted.

For example, a spec string like this:

@smallexample
%@{.c:-foo@} %@{!.c:-bar@} %@{.c|d:-baz@} %@{!.c|d:-boggle@}
@end smallexample

will output the following command-line options from the following input
command-line options:

@smallexample
fred.c        -foo -baz
```

```
jim.d         -bar -boggle
-d fred.c     -foo -baz -boggle
-d jim.d      -bar -baz -boggle
@end smallexample

@item %@{S:X; T:Y; :D@}

If @code{S} was given to GCC, substitutes @code{X}; else if @code{T} was
given to GCC, substitutes @code{Y}; else substitutes @code{D}.  There can
be as many clauses as you need.  This may be combined with @code{.},
@code{,}, @code{!}, @code{|}, and @code{*} as needed.


@end table

The conditional text @code{X} in a %@{@code{S}:@code{X}@} or similar
construct may contain other nested @samp{%} constructs or spaces, or
even newlines.  They are processed as usual, as described above.
Trailing white space in @code{X} is ignored.  White space may also
appear anywhere on the left side of the colon in these constructs,
except between @code{.} or @code{*} and the corresponding word.

The @option{-O}, @option{-f}, @option{-m}, and @option{-W} switches are
handled specifically in these constructs.  If another value of
@option{-O} or the negated form of a @option{-f}, @option{-m}, or
@option{-W} switch is found later in the command line, the earlier
switch value is ignored, except with @{@code{S}*@} where @code{S} is
just one letter, which passes all matching options.

The character @samp{|} at the beginning of the predicate text is used to
indicate that a command should be piped to the following command, but
only if @option{-pipe} is specified.

It is built into GCC which switches take arguments and which do not.
(You might think it would be useful to generalize this to allow each
compiler's spec to say which switches take arguments.  But this cannot
be done in a consistent fashion.  GCC cannot even decide which input
files have been specified without knowing which switches take arguments,
and it must know which input files to compile in order to tell which
compilers to run).

GCC also knows implicitly that arguments starting in @option{-l} are to be
treated as compiler output files, and passed to the linker in their
proper position among the other output files.

@c man begin OPTIONS

@node Target Options
@section Specifying Target Machine and Compiler Version
@cindex target options
@cindex cross compiling
@cindex specifying machine version
@cindex specifying compiler version and target machine
@cindex compiler version, specifying
@cindex target machine, specifying

The usual way to run GCC is to run the executable called @file{gcc}, or
@file{<machine>-gcc} when cross-compiling, or
@file{<machine>-gcc-<version>} to run a version other than the one that
was installed last.  Sometimes this is inconvenient, so GCC provides
options that will switch to another cross-compiler or version.

@table @gcctabopt
@item -b @var{machine}
@opindex b
The argument @var{machine} specifies the target machine for compilation.
```



```
The value to use for @var{machine} is the same as was specified as the
machine type when configuring GCC as a cross-compiler.  For
example, if a cross-compiler was configured with @samp{configure
arm-elf}, meaning to compile for an arm processor with elf binaries,
then you would specify @option{-b arm-elf} to run that cross compiler.
Because there are other options beginning with @option{-b}, the
configuration must contain a hyphen, or @option{-b} alone should be one
argument followed by the configuration in the next argument.

@item -V @var{version}
@opindex V
The argument @var{version} specifies which version of GCC to run.
This is useful when multiple versions are installed.  For example,
@var{version} might be @samp{4.0}, meaning to run GCC version 4.0.
@end table

The @option{-V} and @option{-b} options work by running the
@file{<machine>-gcc-<version>} executable, so there's no real reason to
use them if you can just run that directly.

@node Submodel Options
@section Hardware Models and Configurations
@cindex submodel options
@cindex specifying hardware config
@cindex hardware models and configurations, specifying
@cindex machine dependent options

Earlier we discussed the standard option @option{-b} which chooses among
different installed compilers for completely different target
machines, such as VAX vs.@: 68000 vs.@: 80386.

In addition, each of these target machine types can have its own
special options, starting with @samp{-m}, to choose among various
hardware models or configurations---for example, 68010 vs 68020,
floating coprocessor or none.  A single installed version of the
compiler can compile for any model or configuration, according to the
options specified.

Some configurations of the compiler also support additional special
options, usually for compatibility with other compilers on the same
platform.

@c This list is ordered alphanumerically by subsection name.
@c It should be the same order and spelling as these options are listed
@c in Machine Dependent Options

@menu
* ARC Options::
* ARM Options::
* AVR Options::
* Blackfin Options::
* CRIS Options::
* CRX Options::
* Darwin Options::
* DEC Alpha Options::
* DEC Alpha/VMS Options::
* FR30 Options::
* FRV Options::
* GNU/Linux Options::
* H8/300 Options::
* HPPA Options::
* i386 and x86-64 Options::
* i386 and x86-64 Windows Options::
* IA-64 Options::
* M32C Options::
```

```
* M32R/D Options::
* M680x0 Options::
* M68hc1x Options::
* MCore Options::
* MIPS Options::
* MMIX Options::
* MN10300 Options::
* PDP-11 Options::
* picoChip Options::
* PowerPC Options::
* RS/6000 and PowerPC Options::
* S/390 and zSeries Options::
* Score Options::
* SH Options::
* SPARC Options::
* SPU Options::
* System V Options::
* V850 Options::
* VAX Options::
* VxWorks Options::
* x86-64 Options::
* Xstormy16 Options::
* Xtensa Options::
* zSeries Options::
@end menu

@node ARC Options
@subsection ARC Options
@cindex ARC Options

These options are defined for ARC implementations:

@table @gcctabopt
@item -EL
@opindex EL
Compile code for little endian mode.  This is the default.

@item -EB
@opindex EB
Compile code for big endian mode.

@item -mmangle-cpu
@opindex mmangle-cpu
Prepend the name of the cpu to all public symbol names.
In multiple-processor systems, there are many ARC variants with different
instruction and register set characteristics.  This flag prevents code
compiled for one cpu to be linked with code compiled for another.
No facility exists for handling variants that are ``almost identical''.
This is an all or nothing option.

@item -mcpu=@var{cpu}
@opindex mcpu
Compile code for ARC variant @var{cpu}.
Which variants are supported depend on the configuration.
All variants support @option{-mcpu=base}, this is the default.

@item -mtext=@var{text-section}
@itemx -mdata=@var{data-section}
@itemx -mrodata=@var{readonly-data-section}
@opindex mtext
@opindex mdata
@opindex mrodata
Put functions, data, and readonly data in @var{text-section},
@var{data-section}, and @var{readonly-data-section} respectively
by default.  This can be overridden with the @code{section} attribute.
@xref{Variable Attributes}.
```



```
@item -mfix-cortex-m3-ldrd
@opindex mfix-cortex-m3-ldrd
Some Cortex-M3 cores can cause data corruption when @code{ldrd} instructions
with overlapping destination and base registers are used.  This option avoids
generating these instructions.  This option is enabled by default when
@option{-mcpu=cortex-m3} is specified.

@end table

@node ARM Options
@subsection ARM Options
@cindex ARM options

These @samp{-m} options are defined for Advanced RISC Machines (ARM)
architectures:

@table @gcctabopt
@item -mabi=@var{name}
@opindex mabi
Generate code for the specified ABI@.  Permissible values are: @samp{apcs-gnu},
@samp{atpcs}, @samp{aapcs}, @samp{aapcs-linux} and @samp{iwmmxt}.

@item -mapcs-frame
@opindex mapcs-frame
Generate a stack frame that is compliant with the ARM Procedure Call
Standard for all functions, even if this is not strictly necessary for
correct execution of the code.  Specifying @option{-fomit-frame-pointer}
with this option will cause the stack frames not to be generated for
leaf functions.  The default is @option{-mno-apcs-frame}.

@item -mapcs
@opindex mapcs
This is a synonym for @option{-mapcs-frame}.

@ignore
@c not currently implemented
@item -mapcs-stack-check
@opindex mapcs-stack-check
Generate code to check the amount of stack space available upon entry to
every function (that actually uses some stack space).  If there is
insufficient space available then either the function
@samp{__rt_stkovf_split_small} or @samp{__rt_stkovf_split_big} will be
called, depending upon the amount of stack space required.  The run time
system is required to provide these functions.  The default is
@option{-mno-apcs-stack-check}, since this produces smaller code.

@c not currently implemented
@item -mapcs-float
@opindex mapcs-float
Pass floating point arguments using the float point registers.  This is
one of the variants of the APCS@.  This option is recommended if the
target hardware has a floating point unit or if a lot of floating point
arithmetic is going to be performed by the code.  The default is
@option{-mno-apcs-float}, since integer only code is slightly increased in
size if @option{-mapcs-float} is used.

@c not currently implemented
@item -mapcs-reentrant
@opindex mapcs-reentrant
Generate reentrant, position independent code.  The default is
@option{-mno-apcs-reentrant}.
@end ignore

@item -mthumb-interwork
@opindex mthumb-interwork
```

```
Generate code which supports calling between the ARM and Thumb
instruction sets.  Without this option the two instruction sets cannot
be reliably used inside one program.  The default is
@option{-mno-thumb-interwork}, since slightly larger code is generated
when @option{-mthumb-interwork} is specified.

@item -mno-sched-prolog
@opindex mno-sched-prolog
Prevent the reordering of instructions in the function prolog, or the
merging of those instruction with the instructions in the function's
body.  This means that all functions will start with a recognizable set
of instructions (or in fact one of a choice from a small set of
different function prologues), and this information can be used to
locate the start if functions inside an executable piece of code.  The
default is @option{-msched-prolog}.

@item -mfloat-abi=@var{name}
@opindex mfloat-abi
Specifies which floating-point ABI to use.  Permissible values
are: @samp{soft}, @samp{softfp} and @samp{hard}.

Specifying @samp{soft} causes GCC to generate output containing
library calls for floating-point operations.
@samp{softfp} allows the generation of code using hardware floating-point
instructions, but still uses the soft-float calling conventions.
@samp{hard} allows generation of floating-point instructions
and uses FPU-specific calling conventions.

Using @option{-mfloat-abi=hard} with VFP coprocessors is not supported.
Use @option{-mfloat-abi=softfp} with the appropriate @option{-mfpu} option
to allow the compiler to generate code that makes use of the hardware
floating-point capabilities for these CPUs.

The default depends on the specific target configuration.  Note that
the hard-float and soft-float ABIs are not link-compatible; you must
compile your entire program with the same ABI, and link with a
compatible set of libraries.

@item -mhard-float
@opindex mhard-float
Equivalent to @option{-mfloat-abi=hard}.

@item -msoft-float
@opindex msoft-float
Equivalent to @option{-mfloat-abi=soft}.

@item -mlittle-endian
@opindex mlittle-endian
Generate code for a processor running in little-endian mode.  This is
the default for all standard configurations.

@item -mbig-endian
@opindex mbig-endian
Generate code for a processor running in big-endian mode; the default is
to compile code for a little-endian processor.

@item -mwords-little-endian
@opindex mwords-little-endian
This option only applies when generating code for big-endian processors.
Generate code for a little-endian word order but a big-endian byte
order.  That is, a byte order of the form @samp{32107654}.  Note: this
option should only be used if you require compatibility with code for
big-endian ARM processors generated by versions of the compiler prior to
2.8.

@item -mcpu=@var{name}
```



```
@opindex mcpu
This specifies the name of the target ARM processor.  GCC uses this name
to determine what kind of instructions it can emit when generating
assembly code.  Permissible names are: @samp{arm2}, @samp{arm250},
@samp{arm3}, @samp{arm6}, @samp{arm60}, @samp{arm600}, @samp{arm610},
@samp{arm620}, @samp{arm7}, @samp{arm7m}, @samp{arm7d}, @samp{arm7dm},
@samp{arm7di}, @samp{arm7dmi}, @samp{arm70}, @samp{arm700},
@samp{arm700i}, @samp{arm710}, @samp{arm710c}, @samp{arm7100},
@samp{arm720},
@samp{arm7500}, @samp{arm7500fe}, @samp{arm7tdmi}, @samp{arm7tdmi-s},
@samp{arm710t}, @samp{arm720t}, @samp{arm740t},
@samp{strongarm}, @samp{strongarm110}, @samp{strongarm1100},
@samp{strongarm1110},
@samp{arm8}, @samp{arm810}, @samp{arm9}, @samp{arm9e}, @samp{arm920},
@samp{arm920t}, @samp{arm922t}, @samp{arm946e-s}, @samp{arm966e-s},
@samp{arm968e-s}, @samp{arm926ej-s}, @samp{arm940t}, @samp{arm9tdmi},
@samp{arm10tdmi}, @samp{arm1020t}, @samp{arm1026ej-s},
@samp{arm10e}, @samp{arm1020e}, @samp{arm1022e},
@samp{arm1136j-s}, @samp{arm1136jf-s}, @samp{mpcore}, @samp{mpcorenovfp},
@samp{arm1156t2-s}, @samp{arm1176jz-s}, @samp{arm1176jzf-s},
@samp{cortex-a8}, @samp{cortex-a9},
@samp{cortex-r4}, @samp{cortex-r4f}, @samp{cortex-m3},
@samp{cortex-m1},
@samp{xscale}, @samp{iwmmxt}, @samp{iwmmxt2}, @samp{ep9312}.

@item -mtune=@var{name}
@opindex mtune
This option is very similar to the @option{-mcpu=} option, except that
instead of specifying the actual target processor type, and hence
restricting which instructions can be used, it specifies that GCC should
tune the performance of the code as if the target were of the type
specified in this option, but still choosing the instructions that it
will generate based on the cpu specified by a @option{-mcpu=} option.
For some ARM implementations better performance can be obtained by using
this option.

@item -march=@var{name}
@opindex march
This specifies the name of the target ARM architecture.  GCC uses this
name to determine what kind of instructions it can emit when generating
assembly code.  This option can be used in conjunction with or instead
of the @option{-mcpu=} option.  Permissible names are: @samp{armv2},
@samp{armv2a}, @samp{armv3}, @samp{armv3m}, @samp{armv4}, @samp{armv4t},
@samp{armv5}, @samp{armv5t}, @samp{armv5e}, @samp{armv5te},
@samp{armv6}, @samp{armv6j},
@samp{armv6t2}, @samp{armv6z}, @samp{armv6zk}, @samp{armv6-m},
@samp{armv7}, @samp{armv7-a}, @samp{armv7-r}, @samp{armv7-m},
@samp{iwmmxt}, @samp{iwmmxt2}, @samp{ep9312}.

@item -mfpu=@var{name}
@itemx -mfpe=@var{number}
@itemx -mfp=@var{number}
@opindex mfpu
@opindex mfpe
@opindex mfp
This specifies what floating point hardware (or hardware emulation) is
available on the target.  Permissible names are: @samp{fpa}, @samp{fpe2},
@samp{fpe3}, @samp{maverick}, @samp{vfp}, @samp{vfpv3}, @samp{vfpv3-d16} and
@samp{neon}.  @option{-mfp} and @option{-mfpe}
are synonyms for @option{-mfpu}=@samp{fpe}@var{number}, for compatibility
with older versions of GCC@.

If @option{-msoft-float} is specified this specifies the format of
floating point values.

@item -mstructure-size-boundary=@var{n}
```

```
@opindex mstructure-size-boundary
The size of all structures and unions will be rounded up to a multiple
of the number of bits set by this option.  Permissible values are 8, 32
and 64.  The default value varies for different toolchains.  For the COFF
targeted toolchain the default value is 8.  A value of 64 is only allowed
if the underlying ABI supports it.

Specifying the larger number can produce faster, more efficient code, but
can also increase the size of the program.  Different values are potentially
incompatible.  Code compiled with one value cannot necessarily expect to
work with code or libraries compiled with another value, if they exchange
information using structures or unions.

@item -mabort-on-noreturn
@opindex mabort-on-noreturn
Generate a call to the function @code{abort} at the end of a
@code{noreturn} function.  It will be executed if the function tries to
return.

@item -mlong-calls
@itemx -mno-long-calls
@opindex mlong-calls
@opindex mno-long-calls
Tells the compiler to perform function calls by first loading the
address of the function into a register and then performing a subroutine
call on this register.  This switch is needed if the target function
will lie outside of the 64 megabyte addressing range of the offset based
version of subroutine call instruction.

Even if this switch is enabled, not all function calls will be turned
into long calls.  The heuristic is that static functions, functions
which have the @samp{short-call} attribute, functions that are inside
the scope of a @samp{#pragma no_long_calls} directive and functions whose
definitions have already been compiled within the current compilation
unit, will not be turned into long calls.  The exception to this rule is
that weak function definitions, functions with the @samp{long-call}
attribute or the @samp{section} attribute, and functions that are within
the scope of a @samp{#pragma long_calls} directive, will always be
turned into long calls.

This feature is not enabled by default.  Specifying
@option{-mno-long-calls} will restore the default behavior, as will
placing the function calls within the scope of a @samp{#pragma
long_calls_off} directive.  Note these switches have no effect on how
the compiler generates code to handle function calls via function
pointers.

@item -msingle-pic-base
@opindex msingle-pic-base
Treat the register used for PIC addressing as read-only, rather than
loading it in the prologue for each function.  The run-time system is
responsible for initializing this register with an appropriate value
before execution begins.

@item -mpic-register=@var{reg}
@opindex mpic-register
Specify the register to be used for PIC addressing.  The default is R10
unless stack-checking is enabled, when R9 is used.

@item -mcirrus-fix-invalid-insns
@opindex mcirrus-fix-invalid-insns
@opindex mno-cirrus-fix-invalid-insns
Insert NOPs into the instruction stream to in order to work around
problems with invalid Maverick instruction combinations.  This option
is only valid if the @option{-mcpu=ep9312} option has been used to
enable generation of instructions for the Cirrus Maverick floating
```



```
point co-processor.  This option is not enabled by default, since the
problem is only present in older Maverick implementations.  The default
can be re-enabled by use of the @option{-mno-cirrus-fix-invalid-insns}
switch.

@item -mpoke-function-name
@opindex mpoke-function-name
Write the name of each function into the text section, directly
preceding the function prologue.  The generated code is similar to this:

@smallexample
     t0
         .ascii "arm_poke_function_name", 0
         .align
     t1
         .word 0xff000000 + (t1 - t0)
     arm_poke_function_name
         mov     ip, sp
         stmfd   sp!, @{fp, ip, lr, pc@}
         sub     fp, ip, #4
@end smallexample

When performing a stack backtrace, code can inspect the value of
@code{pc} stored at @code{fp + 0}.  If the trace function then looks at
location @code{pc - 12} and the top 8 bits are set, then we know that
there is a function name embedded immediately preceding this location
and has length @code{((pc[-3]) & 0xff000000)}.

@item -mthumb
@opindex mthumb
Generate code for the Thumb instruction set.  The default is to
use the 32-bit ARM instruction set.
This option automatically enables either 16-bit Thumb-1 or
mixed 16/32-bit Thumb-2 instructions based on the @option{-mcpu=@var{name}}
and @option{-march=@var{name}} options.

@item -mtpcs-frame
@opindex mtpcs-frame
Generate a stack frame that is compliant with the Thumb Procedure Call
Standard for all non-leaf functions.  (A leaf function is one that does
not call any other functions.)  The default is @option{-mno-tpcs-frame}.

@item -mtpcs-leaf-frame
@opindex mtpcs-leaf-frame
Generate a stack frame that is compliant with the Thumb Procedure Call
Standard for all leaf functions.  (A leaf function is one that does
not call any other functions.)  The default is @option{-mno-apcs-leaf-frame}.

@item -mcallee-super-interworking
@opindex mcallee-super-interworking
Gives all externally visible functions in the file being compiled an ARM
instruction set header which switches to Thumb mode before executing the
rest of the function.  This allows these functions to be called from
non-interworking code.

@item -mcaller-super-interworking
@opindex mcaller-super-interworking
Allows calls via function pointers (including virtual functions) to
execute correctly regardless of whether the target code has been
compiled for interworking or not.  There is a small overhead in the cost
of executing a function pointer if this option is enabled.

@item -mtp=@var{name}
@opindex mtp
Specify the access model for the thread local storage pointer.  The valid
models are @option{soft}, which generates calls to @code{__aeabi_read_tp},
```

@option{cp15}, which fetches the thread pointer from @code{cp15} directly
(supported in the arm6k architecture), and @option{auto}, which uses the
best available method for the selected processor. The default setting is
@option{auto}.

@item -mword-relocations
@opindex mword-relocations
Only generate absolute relocations on word sized values (i.e. R_ARM_ABS32).
This is enabled by default on targets (uClinux, SymbianOS) where the runtime
loader imposes this restriction, and when @option{-fpic} or @option{-fPIC}
is specified.

@end table

@node AVR Options
@subsection AVR Options
@cindex AVR Options

These options are defined for AVR implementations:

@table @gcctabopt
@item -mmcu=@var{mcu}
@opindex mmcu
Specify ATMEL AVR instruction set or MCU type.

Instruction set avr1 is for the minimal AVR core, not supported by the C
compiler, only for assembler programs (MCU types: at90s1200, attiny10,
attiny11, attiny12, attiny15, attiny28).

Instruction set avr2 (default) is for the classic AVR core with up to
8K program memory space (MCU types: at90s2313, at90s2323, attiny22,
at90s2333, at90s2343, at90s4414, at90s4433, at90s4434, at90s8515,
at90c8534, at90s8535).

Instruction set avr3 is for the classic AVR core with up to 128K program
memory space (MCU types: atmega103, atmega603, at43usb320, at76c711).

Instruction set avr4 is for the enhanced AVR core with up to 8K program
memory space (MCU types: atmega8, atmega83, atmega85).

Instruction set avr5 is for the enhanced AVR core with up to 128K program
memory space (MCU types: atmega16, atmega161, atmega163, atmega32, atmega323,
atmega64, atmega128, at43usb355, at94k).

@item -msize
@opindex msize
Output instruction sizes to the asm file.

@item -mno-interrupts
@opindex mno-interrupts
Generated code is not compatible with hardware interrupts.
Code size will be smaller.

@item -mcall-prologues
@opindex mcall-prologues
Functions prologues/epilogues expanded as call to appropriate
subroutines. Code size will be smaller.

@item -mno-tablejump
@opindex mno-tablejump
Do not generate tablejump insns which sometimes increase code size.
The option is now deprecated in favor of the equivalent
@option{-fno-jump-tables}

@item -mtiny-stack
@opindex mtiny-stack

Change only the low 8 bits of the stack pointer.

@item -mint8
@opindex mint8
Assume int to be 8 bit integer. This affects the sizes of all types: A
char will be 1 byte, an int will be 1 byte, an long will be 2 bytes
and long long will be 4 bytes. Please note that this option does not
comply to the C standards, but it will provide you with smaller code
size.
@end table

@node Blackfin Options
@subsection Blackfin Options
@cindex Blackfin Options

@table @gcctabopt
@item -mcpu=@var{cpu}@r{[}-@var{sirevision}@r{]}
@opindex mcpu=
Specifies the name of the target Blackfin processor. Currently, @var{cpu}
can be one of @samp{bf512}, @samp{bf514}, @samp{bf516}, @samp{bf518},
@samp{bf522}, @samp{bf523}, @samp{bf524}, @samp{bf525}, @samp{bf526},
@samp{bf527}, @samp{bf531}, @samp{bf532}, @samp{bf533},
@samp{bf534}, @samp{bf536}, @samp{bf537}, @samp{bf538}, @samp{bf539},
@samp{bf542}, @samp{bf544}, @samp{bf547}, @samp{bf548}, @samp{bf549},
@samp{bf561}.
The optional @var{sirevision} specifies the silicon revision of the target
Blackfin processor. Any workarounds available for the targeted silicon revision
will be enabled. If @var{sirevision} is @samp{none}, no workarounds are enabled
If @var{sirevision} is @samp{any}, all workarounds for the targeted processor
will be enabled. The @code{__SILICON_REVISION__} macro is defined to two
hexadecimal digits representing the major and minor numbers in the silicon
revision. If @var{sirevision} is @samp{none}, the @code{__SILICON_REVISION__}
is not defined. If @var{sirevision} is @samp{any}, the
@code{__SILICON_REVISION__} is defined to be @code{0xffff}.
If this optional @var{sirevision} is not used, GCC assumes the latest known
silicon revision of the targeted Blackfin processor.

Support for @samp{bf561} is incomplete. For @samp{bf561},
Only the processor macro is defined.
Without this option, @samp{bf532} is used as the processor by default.
The corresponding predefined processor macros for @var{cpu} is to
be defined. And for @samp{bfin-elf} toolchain, this causes the hardware BSP
provided by libgloss to be linked in if @option{-msim} is not given.

@item -msim
@opindex msim
Specifies that the program will be run on the simulator. This causes
the simulator BSP provided by libgloss to be linked in. This option
has effect only for @samp{bfin-elf} toolchain.
Certain other options, such as @option{-mid-shared-library} and
@option{-mfdpic}, imply @option{-msim}.

@item -momit-leaf-frame-pointer
@opindex momit-leaf-frame-pointer
Don't keep the frame pointer in a register for leaf functions. This
avoids the instructions to save, set up and restore frame pointers and
makes an extra register available in leaf functions. The option
@option{-fomit-frame-pointer} removes the frame pointer for all functions
which might make debugging harder.

@item -mspecld-anomaly
@opindex mspecld-anomaly
When enabled, the compiler will ensure that the generated code does not
contain speculative loads after jump instructions. If this option is used,
@code{__WORKAROUND_SPECULATIVE_LOADS} is defined.

```
@item -mno-specld-anomaly
@opindex mno-specld-anomaly
Don't generate extra code to prevent speculative loads from occurring.

@item -mcsync-anomaly
@opindex mcsync-anomaly
When enabled, the compiler will ensure that the generated code does not
contain CSYNC or SSYNC instructions too soon after conditional branches.
If this option is used, @code{__WORKAROUND_SPECULATIVE_SYNCS} is defined.

@item -mno-csync-anomaly
@opindex mno-csync-anomaly
Don't generate extra code to prevent CSYNC or SSYNC instructions from
occurring too soon after a conditional branch.

@item -mlow-64k
@opindex mlow-64k
When enabled, the compiler is free to take advantage of the knowledge that
the entire program fits into the low 64k of memory.

@item -mno-low-64k
@opindex mno-low-64k
Assume that the program is arbitrarily large.  This is the default.

@item -mstack-check-l1
@opindex mstack-check-l1
Do stack checking using information placed into L1 scratchpad memory by the
uClinux kernel.

@item -mid-shared-library
@opindex mid-shared-library
Generate code that supports shared libraries via the library ID method.
This allows for execute in place and shared libraries in an environment
without virtual memory management.  This option implies @option{-fPIC}.
With a @samp{bfin-elf} target, this option implies @option{-msim}.

@item -mno-id-shared-library
@opindex mno-id-shared-library
Generate code that doesn't assume ID based shared libraries are being used.
This is the default.

@item -mleaf-id-shared-library
@opindex mleaf-id-shared-library
Generate code that supports shared libraries via the library ID method,
but assumes that this library or executable won't link against any other
ID shared libraries.  That allows the compiler to use faster code for jumps
and calls.

@item -mno-leaf-id-shared-library
@opindex mno-leaf-id-shared-library
Do not assume that the code being compiled won't link against any ID shared
libraries.  Slower code will be generated for jump and call insns.

@item -mshared-library-id=n
@opindex mshared-library-id
Specified the identification number of the ID based shared library being
compiled.  Specifying a value of 0 will generate more compact code, specifying
other values will force the allocation of that number to the current
library but is no more space or time efficient than omitting this option.

@item -msep-data
@opindex msep-data
Generate code that allows the data segment to be located in a different
area of memory from the text segment.  This allows for execute in place in
an environment without virtual memory management by eliminating relocations
against the text section.
```



```
@item -mno-sep-data
@opindex mno-sep-data
Generate code that assumes that the data segment follows the text segment.
This is the default.

@item -mlong-calls
@itemx -mno-long-calls
@opindex mlong-calls
@opindex mno-long-calls
Tells the compiler to perform function calls by first loading the
address of the function into a register and then performing a subroutine
call on this register.  This switch is needed if the target function
will lie outside of the 24 bit addressing range of the offset based
version of subroutine call instruction.

This feature is not enabled by default.  Specifying
@option{-mno-long-calls} will restore the default behavior.  Note these
switches have no effect on how the compiler generates code to handle
function calls via function pointers.

@item -mfast-fp
@opindex mfast-fp
Link with the fast floating-point library. This library relaxes some of
the IEEE floating-point standard's rules for checking inputs against
Not-a-Number (NAN), in the interest of performance.

@item -minline-plt
@opindex minline-plt
Enable inlining of PLT entries in function calls to functions that are
not known to bind locally.  It has no effect without @option{-mfdpic}.

@item -mmulticore
@opindex mmulticore
Build standalone application for multicore Blackfin processor. Proper
start files and link scripts will be used to support multicore.
This option defines @code{__BFIN_MULTICORE}. It can only be used with
@option{-mcpu=bf561@r{[}-@var{sirevision}@r{]}}. It can be used with
@option{-mcorea} or @option{-mcoreb}. If it's used without
@option{-mcorea} or @option{-mcoreb}, single application/dual core
programming model is used. In this model, the main function of Core B
should be named as coreb_main. If it's used with @option{-mcorea} or
@option{-mcoreb}, one application per core programming model is used.
If this option is not used, single core application programming
model is used.

@item -mcorea
@opindex mcorea
Build standalone application for Core A of BF561 when using
one application per core programming model. Proper start files
and link scripts will be used to support Core A. This option
defines @code{__BFIN_COREA}. It must be used with @option{-mmulticore}.

@item -mcoreb
@opindex mcoreb
Build standalone application for Core B of BF561 when using
one application per core programming model. Proper start files
and link scripts will be used to support Core B. This option
defines @code{__BFIN_COREB}. When this option is used, coreb_main
should be used instead of main. It must be used with
@option{-mmulticore}.

@item -msdram
@opindex msdram
Build standalone application for SDRAM. Proper start files and
link scripts will be used to put the application into SDRAM.
```

```
Loader should initialize SDRAM before loading the application
into SDRAM. This option defines @code{__BFIN_SDRAM}.

@item -micplb
@opindex micplb
Assume that ICPLBs are enabled at runtime.  This has an effect on certain
anomaly workarounds.  For Linux targets, the default is to assume ICPLBs
are enabled; for standalone applications the default is off.
@end table

@node CRIS Options
@subsection CRIS Options
@cindex CRIS Options

These options are defined specifically for the CRIS ports.

@table @gcctabopt
@item -march=@var{architecture-type}
@itemx -mcpu=@var{architecture-type}
@opindex march
@opindex mcpu
Generate code for the specified architecture.  The choices for
@var{architecture-type} are @samp{v3}, @samp{v8} and @samp{v10} for
respectively ETRAX@w{ }4, ETRAX@w{ }100, and ETRAX@w{ }100@w{ }LX@.
Default is @samp{v0} except for cris-axis-linux-gnu, where the default is
@samp{v10}.

@item -mtune=@var{architecture-type}
@opindex mtune
Tune to @var{architecture-type} everything applicable about the generated
code, except for the ABI and the set of available instructions.  The
choices for @var{architecture-type} are the same as for
@option{-march=@var{architecture-type}}.

@item -mmax-stack-frame=@var{n}
@opindex mmax-stack-frame
Warn when the stack frame of a function exceeds @var{n} bytes.

@item -metrax4
@itemx -metrax100
@opindex metrax4
@opindex metrax100
The options @option{-metrax4} and @option{-metrax100} are synonyms for
@option{-march=v3} and @option{-march=v8} respectively.

@item -mmul-bug-workaround
@itemx -mno-mul-bug-workaround
@opindex mmul-bug-workaround
@opindex mno-mul-bug-workaround
Work around a bug in the @code{muls} and @code{mulu} instructions for CPU
models where it applies.  This option is active by default.

@item -mpdebug
@opindex mpdebug
Enable CRIS-specific verbose debug-related information in the assembly
code.  This option also has the effect to turn off the @samp{#NO_APP}
formatted-code indicator to the assembler at the beginning of the
assembly file.

@item -mcc-init
@opindex mcc-init
Do not use condition-code results from previous instruction; always emit
compare and test instructions before use of condition codes.

@item -mno-side-effects
@opindex mno-side-effects
```



```
Do not emit instructions with side-effects in addressing modes other than
post-increment.

@item -mstack-align
@itemx -mno-stack-align
@itemx -mdata-align
@itemx -mno-data-align
@itemx -mconst-align
@itemx -mno-const-align
@opindex mstack-align
@opindex mno-stack-align
@opindex mdata-align
@opindex mno-data-align
@opindex mconst-align
@opindex mno-const-align
These options (no-options) arranges (eliminate arrangements) for the
stack-frame, individual data and constants to be aligned for the maximum
single data access size for the chosen CPU model.  The default is to
arrange for 32-bit alignment.  ABI details such as structure layout are
not affected by these options.

@item -m32-bit
@itemx -m16-bit
@itemx -m8-bit
@opindex m32-bit
@opindex m16-bit
@opindex m8-bit
Similar to the stack- data- and const-align options above, these options
arrange for stack-frame, writable data and constants to all be 32-bit,
16-bit or 8-bit aligned.  The default is 32-bit alignment.

@item -mno-prologue-epilogue
@itemx -mprologue-epilogue
@opindex mno-prologue-epilogue
@opindex mprologue-epilogue
With @option{-mno-prologue-epilogue}, the normal function prologue and
epilogue that sets up the stack-frame are omitted and no return
instructions or return sequences are generated in the code.  Use this
option only together with visual inspection of the compiled code: no
warnings or errors are generated when call-saved registers must be saved,
or storage for local variable needs to be allocated.

@item -mno-gotplt
@itemx -mgotplt
@opindex mno-gotplt
@opindex mgotplt
With @option{-fpic} and @option{-fPIC}, don't generate (do generate)
instruction sequences that load addresses for functions from the PLT part
of the GOT rather than (traditional on other architectures) calls to the
PLT@.  The default is @option{-mgotplt}.

@item -melf
@opindex melf
Legacy no-op option only recognized with the cris-axis-elf and
cris-axis-linux-gnu targets.

@item -mlinux
@opindex mlinux
Legacy no-op option only recognized with the cris-axis-linux-gnu target.

@item -sim
@opindex sim
This option, recognized for the cris-axis-elf arranges
to link with input-output functions from a simulator library.  Code,
initialized data and zero-initialized data are allocated consecutively.
```

```
@item -sim2
@opindex sim2
Like @option{-sim}, but pass linker options to locate initialized data at
0x40000000 and zero-initialized data at 0x80000000.
@end table

@node CRX Options
@subsection CRX Options
@cindex CRX Options

These options are defined specifically for the CRX ports.

@table @gcctabopt

@item -mmac
@opindex mmac
Enable the use of multiply-accumulate instructions. Disabled by default.

@item -mpush-args
@opindex mpush-args
Push instructions will be used to pass outgoing arguments when functions
are called. Enabled by default.
@end table

@node Darwin Options
@subsection Darwin Options
@cindex Darwin options

These options are defined for all architectures running the Darwin operating
system.

FSF GCC on Darwin does not create ''fat'' object files; it will create
an object file for the single architecture that it was built to
target.  Apple's GCC on Darwin does create ''fat'' files if multiple
@option{-arch} options are used; it does so by running the compiler or
linker multiple times and joining the results together with
@file{lipo}.

The subtype of the file created (like @samp{ppc7400} or @samp{ppc970} or
@samp{i686}) is determined by the flags that specify the ISA
that GCC is targetting, like @option{-mcpu} or @option{-march}.  The
@option{-force_cpusubtype_ALL} option can be used to override this.

The Darwin tools vary in their behavior when presented with an ISA
mismatch.  The assembler, @file{as}, will only permit instructions to
be used that are valid for the subtype of the file it is generating,
so you cannot put 64-bit instructions in an @samp{ppc750} object file.
The linker for shared libraries, @file{/usr/bin/libtool}, will fail
and print an error if asked to create a shared library with a less
restrictive subtype than its input files (for instance, trying to put
a @samp{ppc970} object file in a @samp{ppc7400} library).  The linker
for executables, @file{ld}, will quietly give the executable the most
restrictive subtype of any of its input files.

@table @gcctabopt
@item -F@var{dir}
@opindex F
Add the framework directory @var{dir} to the head of the list of
directories to be searched for header files.  These directories are
interleaved with those specified by @option{-I} options and are
scanned in a left-to-right order.

A framework directory is a directory with frameworks in it.  A
framework is a directory with a @samp{"Headers"} and/or
@samp{"PrivateHeaders"} directory contained directly in it that ends
in @samp{".framework"}.  The name of a framework is the name of this
```



```
directory excluding the @samp{".framework"}.  Headers associated with
the framework are found in one of those two directories, with
@samp{"Headers"} being searched first.  A subframework is a framework
directory that is in a framework's @samp{"Frameworks"} directory.
Includes of subframework headers can only appear in a header of a
framework that contains the subframework, or in a sibling subframework
header.  Two subframeworks are siblings if they occur in the same
framework.  A subframework should not have the same name as a
framework, a warning will be issued if this is violated.  Currently a
subframework cannot have subframeworks, in the future, the mechanism
may be extended to support this.  The standard frameworks can be found
in @samp{"/System/Library/Frameworks"} and
@samp{"/Library/Frameworks"}.  An example include looks like
@code{#include <Framework/header.h>}, where @samp{Framework} denotes
the name of the framework and header.h is found in the
@samp{"PrivateHeaders"} or @samp{"Headers"} directory.

@item -iframework@var{dir}
@opindex iframework
Like @option{-F} except the directory is a treated as a system
directory.  The main difference between this @option{-iframework} and
@option{-F} is that with @option{-iframework} the compiler does not
warn about constructs contained within header files found via
@var{dir}.  This option is valid only for the C family of languages.

@item -gused
@opindex gused
Emit debugging information for symbols that are used.  For STABS
debugging format, this enables @option{-feliminate-unused-debug-symbols}.
This is by default ON@.

@item -gfull
@opindex gfull
Emit debugging information for all symbols and types.

@item -mmacosx-version-min=@var{version}
The earliest version of MacOS X that this executable will run on
is @var{version}.  Typical values of @var{version} include @code{10.1},
@code{10.2}, and @code{10.3.9}.

If the compiler was built to use the system's headers by default,
then the default for this option is the system version on which the
compiler is running, otherwise the default is to make choices which
are compatible with as many systems and code bases as possible.

@item -mkernel
@opindex mkernel
Enable kernel development mode.  The @option{-mkernel} option sets
@option{-static}, @option{-fno-common}, @option{-fno-cxa-atexit},
@option{-fno-exceptions}, @option{-fno-non-call-exceptions},
@option{-fapple-kext}, @option{-fno-weak} and @option{-fno-rtti} where
applicable.  This mode also sets @option{-mno-altivec},
@option{-msoft-float}, @option{-fno-builtin} and
@option{-mlong-branch} for PowerPC targets.

@item -mone-byte-bool
@opindex mone-byte-bool
Override the defaults for @samp{bool} so that @samp{sizeof(bool)==1}.
By default @samp{sizeof(bool)} is @samp{4} when compiling for
Darwin/PowerPC and @samp{1} when compiling for Darwin/x86, so this
option has no effect on x86.

@strong{Warning:} The @option{-mone-byte-bool} switch causes GCC
to generate code that is not binary compatible with code generated
without that switch.  Using this switch may require recompiling all
other modules in a program, including system libraries.  Use this
```

```
switch to conform to a non-default data model.

@item -mfix-and-continue
@itemx -ffix-and-continue
@itemx -findirect-data
@opindex mfix-and-continue
@opindex ffix-and-continue
@opindex findirect-data
Generate code suitable for fast turn around development.  Needed to
enable gdb to dynamically load @code{.o} files into already running
programs.  @option{-findirect-data} and @option{-ffix-and-continue}
are provided for backwards compatibility.

@item -all_load
@opindex all_load
Loads all members of static archive libraries.
See man ld(1) for more information.

@item -arch_errors_fatal
@opindex arch_errors_fatal
Cause the errors having to do with files that have the wrong architecture
to be fatal.

@item -bind_at_load
@opindex bind_at_load
Causes the output file to be marked such that the dynamic linker will
bind all undefined references when the file is loaded or launched.

@item -bundle
@opindex bundle
Produce a Mach-o bundle format file.
See man ld(1) for more information.

@item -bundle_loader @var{executable}
@opindex bundle_loader
This option specifies the @var{executable} that will be loading the build
output file being linked.  See man ld(1) for more information.

@item -dynamiclib
@opindex dynamiclib
When passed this option, GCC will produce a dynamic library instead of
an executable when linking, using the Darwin @file{libtool} command.

@item -force_cpusubtype_ALL
@opindex force_cpusubtype_ALL
This causes GCC's output file to have the @var{ALL} subtype, instead of
one controlled by the @option{-mcpu} or @option{-march} option.

@item -allowable_client  @var{client_name}
@itemx -client_name
@itemx -compatibility_version
@itemx -current_version
@itemx -dead_strip
@itemx -dependency-file
@itemx -dylib_file
@itemx -dylinker_install_name
@itemx -dynamic
@itemx -exported_symbols_list
@itemx -filelist
@itemx -flat_namespace
@itemx -force_flat_namespace
@itemx -headerpad_max_install_names
@itemx -image_base
@itemx -init
@itemx -install_name
@itemx -keep_private_externs
```



```
@itemx -multi_module
@itemx -multiply_defined
@itemx -multiply_defined_unused
@itemx -noall_load
@itemx -no_dead_strip_inits_and_terms
@itemx -nofixprebinding
@itemx -nomultidefs
@itemx -noprebind
@itemx -noseglinkedit
@itemx -pagezero_size
@itemx -prebind
@itemx -prebind_all_twolevel_modules
@itemx -private_bundle
@itemx -read_only_relocs
@itemx -sectalign
@itemx -sectobjectsymbols
@itemx -whyload
@itemx -seg1addr
@itemx -sectcreate
@itemx -sectobjectsymbols
@itemx -sectorder
@itemx -segaddr
@itemx -segs_read_only_addr
@itemx -segs_read_write_addr
@itemx -seg_addr_table
@itemx -seg_addr_table_filename
@itemx -seglinkedit
@itemx -segprot
@itemx -segs_read_only_addr
@itemx -segs_read_write_addr
@itemx -single_module
@itemx -static
@itemx -sub_library
@itemx -sub_umbrella
@itemx -twolevel_namespace
@itemx -umbrella
@itemx -undefined
@itemx -unexported_symbols_list
@itemx -weak_reference_mismatches
@itemx -whatsloaded
@opindex allowable_client
@opindex client_name
@opindex compatibility_version
@opindex current_version
@opindex dead_strip
@opindex dependency-file
@opindex dylib_file
@opindex dylinker_install_name
@opindex dynamic
@opindex exported_symbols_list
@opindex filelist
@opindex flat_namespace
@opindex force_flat_namespace
@opindex headerpad_max_install_names
@opindex image_base
@opindex init
@opindex install_name
@opindex keep_private_externs
@opindex multi_module
@opindex multiply_defined
@opindex multiply_defined_unused
@opindex noall_load
@opindex no_dead_strip_inits_and_terms
@opindex nofixprebinding
@opindex nomultidefs
@opindex noprebind
```

```
@opindex noseglinkedit
@opindex pagezero_size
@opindex prebind
@opindex prebind_all_twolevel_modules
@opindex private_bundle
@opindex read_only_relocs
@opindex sectalign
@opindex sectobjectsymbols
@opindex whyload
@opindex seg1addr
@opindex sectcreate
@opindex sectobjectsymbols
@opindex sectorder
@opindex segaddr
@opindex segs_read_only_addr
@opindex segs_read_write_addr
@opindex seg_addr_table
@opindex seg_addr_table_filename
@opindex seglinkedit
@opindex segprot
@opindex segs_read_only_addr
@opindex segs_read_write_addr
@opindex single_module
@opindex static
@opindex sub_library
@opindex sub_umbrella
@opindex twolevel_namespace
@opindex umbrella
@opindex undefined
@opindex unexported_symbols_list
@opindex weak_reference_mismatches
@opindex whatsloaded
These options are passed to the Darwin linker.  The Darwin linker man page
describes them in detail.
@end table

@node DEC Alpha Options
@subsection DEC Alpha Options

These @samp{-m} options are defined for the DEC Alpha implementations:

@table @gcctabopt
@item -mno-soft-float
@itemx -msoft-float
@opindex mno-soft-float
@opindex msoft-float
Use (do not use) the hardware floating-point instructions for
floating-point operations.  When @option{-msoft-float} is specified,
functions in @file{libgcc.a} will be used to perform floating-point
operations.  Unless they are replaced by routines that emulate the
floating-point operations, or compiled in such a way as to call such
emulations routines, these routines will issue floating-point
operations.   If you are compiling for an Alpha without floating-point
operations, you must ensure that the library is built so as not to call
them.

Note that Alpha implementations without floating-point operations are
required to have floating-point registers.

@item -mfp-reg
@itemx -mno-fp-regs
@opindex mfp-reg
@opindex mno-fp-regs
Generate code that uses (does not use) the floating-point register set.
@option{-mno-fp-regs} implies @option{-msoft-float}.  If the floating-point
register set is not used, floating point operands are passed in integer
```



```
registers as if they were integers and floating-point results are passed
in @code{$0} instead of @code{$f0}.  This is a non-standard calling sequence,
so any function with a floating-point argument or return value called by code
compiled with @option{-mno-fp-regs} must also be compiled with that
option.

A typical use of this option is building a kernel that does not use,
and hence need not save and restore, any floating-point registers.

@item -mieee
@opindex mieee
The Alpha architecture implements floating-point hardware optimized for
maximum performance.  It is mostly compliant with the IEEE floating
point standard.  However, for full compliance, software assistance is
required.  This option generates code fully IEEE compliant code
@emph{except} that the @var{inexact-flag} is not maintained (see below).
If this option is turned on, the preprocessor macro @code{_IEEE_FP} is
defined during compilation.  The resulting code is less efficient but is
able to correctly support denormalized numbers and exceptional IEEE
values such as not-a-number and plus/minus infinity.  Other Alpha
compilers call this option @option{-ieee_with_no_inexact}.

@item -mieee-with-inexact
@opindex mieee-with-inexact
This is like @option{-mieee} except the generated code also maintains
the IEEE @var{inexact-flag}.  Turning on this option causes the
generated code to implement fully-compliant IEEE math.  In addition to
@code{_IEEE_FP}, @code{_IEEE_FP_EXACT} is defined as a preprocessor
macro.  On some Alpha implementations the resulting code may execute
significantly slower than the code generated by default.  Since there is
very little code that depends on the @var{inexact-flag}, you should
normally not specify this option.  Other Alpha compilers call this
option @option{-ieee_with_inexact}.

@item -mfp-trap-mode=@var{trap-mode}
@opindex mfp-trap-mode
This option controls what floating-point related traps are enabled.
Other Alpha compilers call this option @option{-fptm @var{trap-mode}}.
The trap mode can be set to one of four values:

@table @samp
@item n
This is the default (normal) setting.  The only traps that are enabled
are the ones that cannot be disabled in software (e.g., division by zero
trap).

@item u
In addition to the traps enabled by @samp{n}, underflow traps are enabled
as well.

@item su
Like @samp{u}, but the instructions are marked to be safe for software
completion (see Alpha architecture manual for details).

@item sui
Like @samp{su}, but inexact traps are enabled as well.
@end table

@item -mfp-rounding-mode=@var{rounding-mode}
@opindex mfp-rounding-mode
Selects the IEEE rounding mode.  Other Alpha compilers call this option
@option{-fprm @var{rounding-mode}}.  The @var{rounding-mode} can be one
of:

@table @samp
@item n
```

```
Normal IEEE rounding mode.  Floating point numbers are rounded towards
the nearest machine number or towards the even machine number in case
of a tie.

@item m
Round towards minus infinity.

@item c
Chopped rounding mode.  Floating point numbers are rounded towards zero.

@item d
Dynamic rounding mode.  A field in the floating point control register
(@var{fpcr}, see Alpha architecture reference manual) controls the
rounding mode in effect.  The C library initializes this register for
rounding towards plus infinity.  Thus, unless your program modifies the
@var{fpcr}, @samp{d} corresponds to round towards plus infinity.
@end table

@item -mtrap-precision=@var{trap-precision}
@opindex mtrap-precision
In the Alpha architecture, floating point traps are imprecise.  This
means without software assistance it is impossible to recover from a
floating trap and program execution normally needs to be terminated.
GCC can generate code that can assist operating system trap handlers
in determining the exact location that caused a floating point trap.
Depending on the requirements of an application, different levels of
precisions can be selected:

@table @samp
@item p
Program precision.  This option is the default and means a trap handler
can only identify which program caused a floating point exception.

@item f
Function precision.  The trap handler can determine the function that
caused a floating point exception.

@item i
Instruction precision.  The trap handler can determine the exact
instruction that caused a floating point exception.
@end table

Other Alpha compilers provide the equivalent options called
@option{-scope_safe} and @option{-resumption_safe}.

@item -mieee-conformant
@opindex mieee-conformant
This option marks the generated code as IEEE conformant.  You must not
use this option unless you also specify @option{-mtrap-precision=i} and either
@option{-mfp-trap-mode=su} or @option{-mfp-trap-mode=sui}.  Its only effect
is to emit the line @samp{.eflag 48} in the function prologue of the
generated assembly file.  Under DEC Unix, this has the effect that
IEEE-conformant math library routines will be linked in.

@item -mbuild-constants
@opindex mbuild-constants
Normally GCC examines a 32- or 64-bit integer constant to
see if it can construct it from smaller constants in two or three
instructions.  If it cannot, it will output the constant as a literal and
generate code to load it from the data segment at runtime.

Use this option to require GCC to construct @emph{all} integer constants
using code, even if it takes more instructions (the maximum is six).

You would typically use this option to build a shared library dynamic
loader.  Itself a shared library, it must relocate itself in memory
```



```
before it can find the variables and constants in its own data segment.

@item -malpha-as
@itemx -mgas
@opindex malpha-as
@opindex mgas
Select whether to generate code to be assembled by the vendor-supplied
assembler (@option{-malpha-as}) or by the GNU assembler @option{-mgas}.

@item -mbwx
@itemx -mno-bwx
@itemx -mcix
@itemx -mno-cix
@itemx -mfix
@itemx -mno-fix
@itemx -mmax
@itemx -mno-max
@opindex mbwx
@opindex mno-bwx
@opindex mcix
@opindex mno-cix
@opindex mfix
@opindex mno-fix
@opindex mmax
@opindex mno-max
Indicate whether GCC should generate code to use the optional BWX,
CIX, FIX and MAX instruction sets.  The default is to use the instruction
sets supported by the CPU type specified via @option{-mcpu=} option or that
of the CPU on which GCC was built if none was specified.

@item -mfloat-vax
@itemx -mfloat-ieee
@opindex mfloat-vax
@opindex mfloat-ieee
Generate code that uses (does not use) VAX F and G floating point
arithmetic instead of IEEE single and double precision.

@item -mexplicit-relocs
@itemx -mno-explicit-relocs
@opindex mexplicit-relocs
@opindex mno-explicit-relocs
Older Alpha assemblers provided no way to generate symbol relocations
except via assembler macros.  Use of these macros does not allow
optimal instruction scheduling.  GNU binutils as of version 2.12
supports a new syntax that allows the compiler to explicitly mark
which relocations should apply to which instructions.  This option
is mostly useful for debugging, as GCC detects the capabilities of
the assembler when it is built and sets the default accordingly.

@item -msmall-data
@itemx -mlarge-data
@opindex msmall-data
@opindex mlarge-data
When @option{-mexplicit-relocs} is in effect, static data is
accessed via @dfn{gp-relative} relocations.  When @option{-msmall-data}
is used, objects 8 bytes long or smaller are placed in a @dfn{small data area}
(the @code{.sdata} and @code{.sbss} sections) and are accessed via
16-bit relocations off of the @code{$gp} register.  This limits the
size of the small data area to 64KB, but allows the variables to be
directly accessed via a single instruction.

The default is @option{-mlarge-data}.  With this option the data area
is limited to just below 2GB@.  Programs that require more than 2GB of
data must use @code{malloc} or @code{mmap} to allocate the data in the
heap instead of in the program's data segment.
```

When generating code for shared libraries, @option{-fpic} implies
@option{-msmall-data} and @option{-fPIC} implies @option{-mlarge-data}.

@item -msmall-text
@itemx -mlarge-text
@opindex msmall-text
@opindex mlarge-text
When @option{-msmall-text} is used, the compiler assumes that the
code of the entire program (or shared library) fits in 4MB, and is
thus reachable with a branch instruction. When @option{-msmall-data}
is used, the compiler can assume that all local symbols share the
same @code{$gp} value, and thus reduce the number of instructions
required for a function call from 4 to 1.

The default is @option{-mlarge-text}.

@item -mcpu=@var{cpu_type}
@opindex mcpu
Set the instruction set and instruction scheduling parameters for
machine type @var{cpu_type}. You can specify either the @samp{EV}
style name or the corresponding chip number. GCC supports scheduling
parameters for the EV4, EV5 and EV6 family of processors and will
choose the default values for the instruction set from the processor
you specify. If you do not specify a processor type, GCC will default
to the processor on which the compiler was built.

Supported values for @var{cpu_type} are

@table @samp
@item ev4
@itemx ev45
@itemx 21064
Schedules as an EV4 and has no instruction set extensions.

@item ev5
@itemx 21164
Schedules as an EV5 and has no instruction set extensions.

@item ev56
@itemx 21164a
Schedules as an EV5 and supports the BWX extension.

@item pca56
@itemx 21164pc
@itemx 21164PC
Schedules as an EV5 and supports the BWX and MAX extensions.

@item ev6
@itemx 21264
Schedules as an EV6 and supports the BWX, FIX, and MAX extensions.

@item ev67
@itemx 21264a
Schedules as an EV6 and supports the BWX, CIX, FIX, and MAX extensions.
@end table

Native Linux/GNU toolchains also support the value @samp{native},
which selects the best architecture option for the host processor.
@option{-mcpu=native} has no effect if GCC does not recognize
the processor.

@item -mtune=@var{cpu_type}
@opindex mtune
Set only the instruction scheduling parameters for machine type
@var{cpu_type}. The instruction set is not changed.

Native Linux/GNU toolchains also support the value @samp{native},
which selects the best architecture option for the host processor.
@option{-mtune=native} has no effect if GCC does not recognize
the processor.

@item -mmemory-latency=@var{time}
@opindex mmemory-latency
Sets the latency the scheduler should assume for typical memory
references as seen by the application. This number is highly
dependent on the memory access patterns used by the application
and the size of the external cache on the machine.

Valid options for @var{time} are

@table @samp
@item @var{number}
A decimal number representing clock cycles.

@item L1
@itemx L2
@itemx L3
@itemx main
The compiler contains estimates of the number of clock cycles for
``typical'' EV4 & EV5 hardware for the Level 1, 2 & 3 caches
(also called Dcache, Scache, and Bcache), as well as to main memory.
Note that L3 is only valid for EV5.

@end table
@end table

@node DEC Alpha/VMS Options
@subsection DEC Alpha/VMS Options

These @samp{-m} options are defined for the DEC Alpha/VMS implementations:

@table @gcctabopt
@item -mvms-return-codes
@opindex mvms-return-codes
Return VMS condition codes from main. The default is to return POSIX
style condition (e.g.@: error) codes.
@end table

@node FR30 Options
@subsection FR30 Options
@cindex FR30 Options

These options are defined specifically for the FR30 port.

@table @gcctabopt

@item -msmall-model
@opindex msmall-model
Use the small address space model. This can produce smaller code, but
it does assume that all symbolic values and addresses will fit into a
20-bit range.

@item -mno-lsim
@opindex mno-lsim
Assume that run-time support has been provided and so there is no need
to include the simulator library (@file{libsim.a}) on the linker
command line.

@end table

@node FRV Options
@subsection FRV Options

```
@cindex FRV Options

@table @gcctabopt
@item -mgpr-32
@opindex mgpr-32

Only use the first 32 general purpose registers.

@item -mgpr-64
@opindex mgpr-64

Use all 64 general purpose registers.

@item -mfpr-32
@opindex mfpr-32

Use only the first 32 floating point registers.

@item -mfpr-64
@opindex mfpr-64

Use all 64 floating point registers

@item -mhard-float
@opindex mhard-float

Use hardware instructions for floating point operations.

@item -msoft-float
@opindex msoft-float

Use library routines for floating point operations.

@item -malloc-cc
@opindex malloc-cc

Dynamically allocate condition code registers.

@item -mfixed-cc
@opindex mfixed-cc

Do not try to dynamically allocate condition code registers, only
use @code{icc0} and @code{fcc0}.

@item -mdword
@opindex mdword

Change ABI to use double word insns.

@item -mno-dword
@opindex mno-dword

Do not use double word instructions.

@item -mdouble
@opindex mdouble

Use floating point double instructions.

@item -mno-double
@opindex mno-double

Do not use floating point double instructions.

@item -mmedia
@opindex mmedia
```



```
Use media instructions.

@item -mno-media
@opindex mno-media

Do not use media instructions.

@item -mmuladd
@opindex mmuladd

Use multiply and add/subtract instructions.

@item -mno-muladd
@opindex mno-muladd

Do not use multiply and add/subtract instructions.

@item -mfdpic
@opindex mfdpic

Select the FDPIC ABI, that uses function descriptors to represent
pointers to functions.  Without any PIC/PIE-related options, it
implies @option{-fPIE}.  With @option{-fpic} or @option{-fpie}, it
assumes GOT entries and small data are within a 12-bit range from the
GOT base address; with @option{-fPIC} or @option{-fPIE}, GOT offsets
are computed with 32 bits.
With a @samp{bfin-elf} target, this option implies @option{-msim}.

@item -minline-plt
@opindex minline-plt

Enable inlining of PLT entries in function calls to functions that are
not known to bind locally.  It has no effect without @option{-mfdpic}.
It's enabled by default if optimizing for speed and compiling for
shared libraries (i.e., @option{-fPIC} or @option{-fpic}), or when an
optimization option such as @option{-O3} or above is present in the
command line.

@item -mTLS
@opindex TLS

Assume a large TLS segment when generating thread-local code.

@item -mtls
@opindex tls

Do not assume a large TLS segment when generating thread-local code.

@item -mgprel-ro
@opindex mgprel-ro

Enable the use of @code{GPREL} relocations in the FDPIC ABI for data
that is known to be in read-only sections.  It's enabled by default,
except for @option{-fpic} or @option{-fpie}: even though it may help
make the global offset table smaller, it trades 1 instruction for 4.
With @option{-fPIC} or @option{-fPIE}, it trades 3 instructions for 4,
one of which may be shared by multiple symbols, and it avoids the need
for a GOT entry for the referenced symbol, so it's more likely to be a
win.  If it is not, @option{-mno-gprel-ro} can be used to disable it.

@item -multilib-library-pic
@opindex multilib-library-pic

Link with the (library, not FD) pic libraries.  It's implied by
@option{-mlibrary-pic}, as well as by @option{-fPIC} and
```

@option{-fpic} without @option{-mfdpic}. You should never have to use
it explicitly.

@item -mlinked-fp
@opindex mlinked-fp

Follow the EABI requirement of always creating a frame pointer whenever
a stack frame is allocated. This option is enabled by default and can
be disabled with @option{-mno-linked-fp}.

@item -mlong-calls
@opindex mlong-calls

Use indirect addressing to call functions outside the current
compilation unit. This allows the functions to be placed anywhere
within the 32-bit address space.

@item -malign-labels
@opindex malign-labels

Try to align labels to an 8-byte boundary by inserting nops into the
previous packet. This option only has an effect when VLIW packing
is enabled. It doesn't create new packets; it merely adds nops to
existing ones.

@item -mlibrary-pic
@opindex mlibrary-pic

Generate position-independent EABI code.

@item -macc-4
@opindex macc-4

Use only the first four media accumulator registers.

@item -macc-8
@opindex macc-8

Use all eight media accumulator registers.

@item -mpack
@opindex mpack

Pack VLIW instructions.

@item -mno-pack
@opindex mno-pack

Do not pack VLIW instructions.

@item -mno-eflags
@opindex mno-eflags

Do not mark ABI switches in e_flags.

@item -mcond-move
@opindex mcond-move

Enable the use of conditional-move instructions (default).

This switch is mainly for debugging the compiler and will likely be removed
in a future version.

@item -mno-cond-move
@opindex mno-cond-move

Disable the use of conditional-move instructions.

This switch is mainly for debugging the compiler and will likely be removed
in a future version.

@item -mscc
@opindex mscc

Enable the use of conditional set instructions (default).

This switch is mainly for debugging the compiler and will likely be removed
in a future version.

@item -mno-scc
@opindex mno-scc

Disable the use of conditional set instructions.

This switch is mainly for debugging the compiler and will likely be removed
in a future version.

@item -mcond-exec
@opindex mcond-exec

Enable the use of conditional execution (default).

This switch is mainly for debugging the compiler and will likely be removed
in a future version.

@item -mno-cond-exec
@opindex mno-cond-exec

Disable the use of conditional execution.

This switch is mainly for debugging the compiler and will likely be removed
in a future version.

@item -mvliw-branch
@opindex mvliw-branch

Run a pass to pack branches into VLIW instructions (default).

This switch is mainly for debugging the compiler and will likely be removed
in a future version.

@item -mno-vliw-branch
@opindex mno-vliw-branch

Do not run a pass to pack branches into VLIW instructions.

This switch is mainly for debugging the compiler and will likely be removed
in a future version.

@item -mmulti-cond-exec
@opindex mmulti-cond-exec

Enable optimization of @code{&&} and @code{||} in conditional execution
(default).

This switch is mainly for debugging the compiler and will likely be removed
in a future version.

@item -mno-multi-cond-exec
@opindex mno-multi-cond-exec

Disable optimization of @code{&&} and @code{||} in conditional execution.

```
This switch is mainly for debugging the compiler and will likely be removed
in a future version.

@item -mnested-cond-exec
@opindex mnested-cond-exec

Enable nested conditional execution optimizations (default).

This switch is mainly for debugging the compiler and will likely be removed
in a future version.

@item -mno-nested-cond-exec
@opindex mno-nested-cond-exec

Disable nested conditional execution optimizations.

This switch is mainly for debugging the compiler and will likely be removed
in a future version.

@item -moptimize-membar
@opindex moptimize-membar

This switch removes redundant @code{membar} instructions from the
compiler generated code.  It is enabled by default.

@item -mno-optimize-membar
@opindex mno-optimize-membar

This switch disables the automatic removal of redundant @code{membar}
instructions from the generated code.

@item -mtomcat-stats
@opindex mtomcat-stats

Cause gas to print out tomcat statistics.

@item -mcpu=@var{cpu}
@opindex mcpu

Select the processor type for which to generate code.  Possible values are
@samp{frv}, @samp{fr550}, @samp{tomcat}, @samp{fr500}, @samp{fr450},
@samp{fr405}, @samp{fr400}, @samp{fr300} and @samp{simple}.

@end table

@node GNU/Linux Options
@subsection GNU/Linux Options

These @samp{-m} options are defined for GNU/Linux targets:

@table @gcctabopt
@item -mglibc
@opindex mglibc
Use the GNU C library instead of uClibc.  This is the default except
on @samp{*-*-linux-*uclibc*} targets.

@item -muclibc
@opindex muclibc
Use uClibc instead of the GNU C library.  This is the default on
@samp{*-*-linux-*uclibc*} targets.
@end table

@node H8/300 Options
@subsection H8/300 Options
```



```
These @samp{-m} options are defined for the H8/300 implementations:

@table @gcctabopt
@item -mrelax
@opindex mrelax
Shorten some address references at link time, when possible; uses the
linker option @option{-relax}.  @xref{H8/300,, @code{ld} and the H8/300,
ld, Using ld}, for a fuller description.

@item -mh
@opindex mh
Generate code for the H8/300H@.

@item -ms
@opindex ms
Generate code for the H8S@.

@item -mn
@opindex mn
Generate code for the H8S and H8/300H in the normal mode.  This switch
must be used either with @option{-mh} or @option{-ms}.

@item -ms2600
@opindex ms2600
Generate code for the H8S/2600.  This switch must be used with @option{-ms}.

@item -mint32
@opindex mint32
Make @code{int} data 32 bits by default.

@item -malign-300
@opindex malign-300
On the H8/300H and H8S, use the same alignment rules as for the H8/300.
The default for the H8/300H and H8S is to align longs and floats on 4
byte boundaries.
@option{-malign-300} causes them to be aligned on 2 byte boundaries.
This option has no effect on the H8/300.
@end table

@node HPPA Options
@subsection HPPA Options
@cindex HPPA Options

These @samp{-m} options are defined for the HPPA family of computers:

@table @gcctabopt
@item -march=@var{architecture-type}
@opindex march
Generate code for the specified architecture.  The choices for
@var{architecture-type} are @samp{1.0} for PA 1.0, @samp{1.1} for PA
1.1, and @samp{2.0} for PA 2.0 processors.  Refer to
@file{/usr/lib/sched.models} on an HP-UX system to determine the proper
architecture option for your machine.  Code compiled for lower numbered
architectures will run on higher numbered architectures, but not the
other way around.

@item -mpa-risc-1-0
@itemx -mpa-risc-1-1
@itemx -mpa-risc-2-0
@opindex mpa-risc-1-0
@opindex mpa-risc-1-1
@opindex mpa-risc-2-0
Synonyms for @option{-march=1.0}, @option{-march=1.1}, and @option{-march=2.0} r

@item -mbig-switch
@opindex mbig-switch
```

Generate code suitable for big switch tables. Use this option only if the assembler/linker complain about out of range branches within a switch table.

@item -mjump-in-delay
@opindex mjump-in-delay
Fill delay slots of function calls with unconditional jump instructions by modifying the return pointer for the function call to be the target of the conditional jump.

@item -mdisable-fpregs
@opindex mdisable-fpregs
Prevent floating point registers from being used in any manner. This is necessary for compiling kernels which perform lazy context switching of floating point registers. If you use this option and attempt to perform floating point operations, the compiler will abort.

@item -mdisable-indexing
@opindex mdisable-indexing
Prevent the compiler from using indexing address modes. This avoids some rather obscure problems when compiling MIG generated code under MACH@.

@item -mno-space-regs
@opindex mno-space-regs
Generate code that assumes the target has no space registers. This allows GCC to generate faster indirect calls and use unscaled index address modes.

Such code is suitable for level 0 PA systems and kernels.

@item -mfast-indirect-calls
@opindex mfast-indirect-calls
Generate code that assumes calls never cross space boundaries. This allows GCC to emit code which performs faster indirect calls.

This option will not work in the presence of shared libraries or nested functions.

@item -mfixed-range=@var{register-range}
@opindex mfixed-range
Generate code treating the given register range as fixed registers.
A fixed register is one that the register allocator can not use. This is useful when compiling kernel code. A register range is specified as two registers separated by a dash. Multiple register ranges can be specified separated by a comma.

@item -mlong-load-store
@opindex mlong-load-store
Generate 3-instruction load and store sequences as sometimes required by the HP-UX 10 linker. This is equivalent to the @samp{+k} option to the HP compilers.

@item -mportable-runtime
@opindex mportable-runtime
Use the portable calling conventions proposed by HP for ELF systems.

@item -mgas
@opindex mgas
Enable the use of assembler directives only GAS understands.

@item -mschedule=@var{cpu-type}
@opindex mschedule
Schedule code according to the constraints for the machine type @var{cpu-type}. The choices for @var{cpu-type} are @samp{700} @samp{7100}, @samp{7100LC}, @samp{7200}, @samp{7300} and @samp{8000}. Refer to @file{/usr/lib/sched.models} on an HP-UX system to determine the proper scheduling option for your machine. The default scheduling is

@samp{8000}.

@item -mlinker-opt
@opindex mlinker-opt
Enable the optimization pass in the HP-UX linker. Note this makes symbolic debugging impossible. It also triggers a bug in the HP-UX 8 and HP-UX 9 linkers in which they give bogus error messages when linking some programs.

@item -msoft-float
@opindex msoft-float
Generate output containing library calls for floating point.
@strong{Warning:} the requisite libraries are not available for all HPPA targets. Normally the facilities of the machine's usual C compiler are used, but this cannot be done directly in cross-compilation. You must make your own arrangements to provide suitable library functions for cross-compilation.

@option{-msoft-float} changes the calling convention in the output file; therefore, it is only useful if you compile @emph{all} of a program with this option. In particular, you need to compile @file{libgcc.a}, the library that comes with GCC, with @option{-msoft-float} in order for this to work.

@item -msio
@opindex msio
Generate the predefine, @code{_SIO}, for server IO@. The default is @option{-mwsio}. This generates the predefines, @code{__hp9000s700}, @code{__hp9000s700__} and @code{_WSIO}, for workstation IO@. These options are available under HP-UX and HI-UX@.

@item -mgnu-ld
@opindex gnu-ld
Use GNU ld specific options. This passes @option{-shared} to ld when building a shared library. It is the default when GCC is configured, explicitly or implicitly, with the GNU linker. This option does not have any affect on which ld is called, it only changes what parameters are passed to that ld. The ld that is called is determined by the @option{--with-ld} configure option, GCC's program search path, and finally by the user's @env{PATH}. The linker used by GCC can be printed using @samp{which 'gcc -print-prog-name=ld'}. This option is only available on the 64 bit HP-UX GCC, i.e.@: configured with @samp{hppa*64*-*-hpux*}.

@item -mhp-ld
@opindex hp-ld
Use HP ld specific options. This passes @option{-b} to ld when building a shared library and passes @option{+Accept TypeMismatch} to ld on all links. It is the default when GCC is configured, explicitly or implicitly, with the HP linker. This option does not have any affect on which ld is called, it only changes what parameters are passed to that ld. The ld that is called is determined by the @option{--with-ld} configure option, GCC's program search path, and finally by the user's @env{PATH}. The linker used by GCC can be printed using @samp{which 'gcc -print-prog-name=ld'}. This option is only available on the 64 bit HP-UX GCC, i.e.@: configured with @samp{hppa*64*-*-hpux*}.

@item -mlong-calls
@opindex mno-long-calls
Generate code that uses long call sequences. This ensures that a call is always able to reach linker generated stubs. The default is to generate long calls only when the distance from the call site to the beginning of the function or translation unit, as the case may be, exceeds a predefined limit set by the branch type being used. The limits for normal calls are 7,600,000 and 240,000 bytes, respectively for the PA 2.0 and PA 1.X architectures. Sibcalls are always limited at 240,000 bytes.

Distances are measured from the beginning of functions when using the
@option{-ffunction-sections} option, or when using the @option{-mgas}
and @option{-mno-portable-runtime} options together under HP-UX with
the SOM linker.

It is normally not desirable to use this option as it will degrade
performance. However, it may be useful in large applications,
particularly when partial linking is used to build the application.

The types of long calls used depends on the capabilities of the
assembler and linker, and the type of code being generated. The
impact on systems that support long absolute calls, and long pic
symbol-difference or pc-relative calls should be relatively small.
However, an indirect call is used on 32-bit ELF systems in pic code
and it is quite long.

@item -munix=@var{unix-std}
@opindex march
Generate compiler predefines and select a startfile for the specified
UNIX standard. The choices for @var{unix-std} are @samp{93}, @samp{95}
and @samp{98}. @samp{93} is supported on all HP-UX versions. @samp{95}
is available on HP-UX 10.10 and later. @samp{98} is available on HP-UX
11.11 and later. The default values are @samp{93} for HP-UX 10.00,
@samp{95} for HP-UX 10.10 though to 11.00, and @samp{98} for HP-UX 11.11
and later.

@option{-munix=93} provides the same predefines as GCC 3.3 and 3.4.
@option{-munix=95} provides additional predefines for @code{XOPEN_UNIX}
and @code{_XOPEN_SOURCE_EXTENDED}, and the startfile @file{unix95.o}.
@option{-munix=98} provides additional predefines for @code{_XOPEN_UNIX},
@code{_XOPEN_SOURCE_EXTENDED}, @code{_INCLUDE__STDC_A1_SOURCE} and
@code{_INCLUDE_XOPEN_SOURCE_500}, and the startfile @file{unix98.o}.

It is @emph{important} to note that this option changes the interfaces
for various library routines. It also affects the operational behavior
of the C library. Thus, @emph{extreme} care is needed in using this
option.

Library code that is intended to operate with more than one UNIX
standard must test, set and restore the variable @var{__xpg4_extended_mask}
as appropriate. Most GNU software doesn't provide this capability.

@item -nolibdld
@opindex nolibdld
Suppress the generation of link options to search libdld.sl when the
@option{-static} option is specified on HP-UX 10 and later.

@item -static
@opindex static
The HP-UX implementation of setlocale in libc has a dependency on
libdld.sl. There isn't an archive version of libdld.sl. Thus,
when the @option{-static} option is specified, special link options
are needed to resolve this dependency.

On HP-UX 10 and later, the GCC driver adds the necessary options to
link with libdld.sl when the @option{-static} option is specified.
This causes the resulting binary to be dynamic. On the 64-bit port,
the linkers generate dynamic binaries by default in any case. The
@option{-nolibdld} option can be used to prevent the GCC driver from
adding these link options.

@item -threads
@opindex threads
Add support for multithreading with the @dfn{dce thread} library
under HP-UX@. This option sets flags for both the preprocessor and
linker.

@end table

@node i386 and x86-64 Options
@subsection Intel 386 and AMD x86-64 Options
@cindex i386 Options
@cindex x86-64 Options
@cindex Intel 386 Options
@cindex AMD x86-64 Options

These @samp{-m} options are defined for the i386 and x86-64 family of
computers:

@table @gcctabopt
@item -mtune=@var{cpu-type}
@opindex mtune
Tune to @var{cpu-type} everything applicable about the generated code, except
for the ABI and the set of available instructions. The choices for
@var{cpu-type} are:
@table @emph
@item generic
Produce code optimized for the most common IA32/AMD64/EM64T processors.
If you know the CPU on which your code will run, then you should use
the corresponding @option{-mtune} option instead of
@option{-mtune=generic}. But, if you do not know exactly what CPU users
of your application will have, then you should use this option.

As new processors are deployed in the marketplace, the behavior of this
option will change. Therefore, if you upgrade to a newer version of
GCC, the code generated option will change to reflect the processors
that were most common when that version of GCC was released.

There is no @option{-march=generic} option because @option{-march}
indicates the instruction set the compiler can use, and there is no
generic instruction set applicable to all processors. In contrast,
@option{-mtune} indicates the processor (or, in this case, collection of
processors) for which the code is optimized.
@item native
This selects the CPU to tune for at compilation time by determining
the processor type of the compiling machine. Using @option{-mtune=native}
will produce code optimized for the local machine under the constraints
of the selected instruction set. Using @option{-march=native} will
enable all instruction subsets supported by the local machine (hence
the result might not run on different machines).
@item i386
Original Intel's i386 CPU@.
@item i486
Intel's i486 CPU@. (No scheduling is implemented for this chip.)
@item i586, pentium
Intel Pentium CPU with no MMX support.
@item pentium-mmx
Intel PentiumMMX CPU based on Pentium core with MMX instruction set support.
@item pentiumpro
Intel PentiumPro CPU@.
@item i686
Same as @code{generic}, but when used as @code{march} option, PentiumPro
instruction set will be used, so the code will run on all i686 family chips.
@item pentium2
Intel Pentium2 CPU based on PentiumPro core with MMX instruction set support.
@item pentium3, pentium3m
Intel Pentium3 CPU based on PentiumPro core with MMX and SSE instruction set
support.
@item pentium-m
Low power version of Intel Pentium3 CPU with MMX, SSE and SSE2 instruction set
support. Used by Centrino notebooks.
@item pentium4, pentium4m
Intel Pentium4 CPU with MMX, SSE and SSE2 instruction set support.

```
@item prescott
Improved version of Intel Pentium4 CPU with MMX, SSE, SSE2 and SSE3 instruction
set support.
@item nocona
Improved version of Intel Pentium4 CPU with 64-bit extensions, MMX, SSE,
SSE2 and SSE3 instruction set support.
@item core2
Intel Core2 CPU with 64-bit extensions, MMX, SSE, SSE2, SSE3 and SSSE3
instruction set support.
@item k6
AMD K6 CPU with MMX instruction set support.
@item k6-2, k6-3
Improved versions of AMD K6 CPU with MMX and 3dNOW!@: instruction set support.
@item athlon, athlon-tbird
AMD Athlon CPU with MMX, 3dNOW!, enhanced 3dNOW!@: and SSE prefetch instructions
support.
@item athlon-4, athlon-xp, athlon-mp
Improved AMD Athlon CPU with MMX, 3dNOW!, enhanced 3dNOW!@: and full SSE
instruction set support.
@item k8, opteron, athlon64, athlon-fx
AMD K8 core based CPUs with x86-64 instruction set support.  (This supersets
MMX, SSE, SSE2, 3dNOW!, enhanced 3dNOW!@: and 64-bit instruction set extensions.
@item k8-sse3, opteron-sse3, athlon64-sse3
Improved versions of k8, opteron and athlon64 with SSE3 instruction set support.
@item amdfam10, barcelona
AMD Family 10h core based CPUs with x86-64 instruction set support.  (This
supersets MMX, SSE, SSE2, SSE3, SSE4A, 3dNOW!, enhanced 3dNOW!, ABM and 64-bit
instruction set extensions.)
@item winchip-c6
IDT Winchip C6 CPU, dealt in same way as i486 with additional MMX instruction
set support.
@item winchip2
IDT Winchip2 CPU, dealt in same way as i486 with additional MMX and 3dNOW!@:
instruction set support.
@item c3
Via C3 CPU with MMX and 3dNOW!@: instruction set support.  (No scheduling is
implemented for this chip.)
@item c3-2
Via C3-2 CPU with MMX and SSE instruction set support.  (No scheduling is
implemented for this chip.)
@item geode
Embedded AMD CPU with MMX and 3dNOW! instruction set support.
@end table

While picking a specific @var{cpu-type} will schedule things appropriately
for that particular chip, the compiler will not generate any code that
does not run on the i386 without the @option{-march=@var{cpu-type}} option
being used.

@item -march=@var{cpu-type}
@opindex march
Generate instructions for the machine type @var{cpu-type}.  The choices
for @var{cpu-type} are the same as for @option{-mtune}.  Moreover,
specifying @option{-march=@var{cpu-type}} implies @option{-mtune=@var{cpu-type}}

@item -mcpu=@var{cpu-type}
@opindex mcpu
A deprecated synonym for @option{-mtune}.

@item -mfpmath=@var{unit}
@opindex march
Generate floating point arithmetics for selected unit @var{unit}.  The choices
for @var{unit} are:

@table @samp
@item 387
```



```
Use the standard 387 floating point coprocessor present majority of chips and
emulated otherwise.  Code compiled with this option will run almost everywhere.
The temporary results are computed in 80bit precision instead of precision
specified by the type resulting in slightly different results compared to most
of other chips.  See @option{-ffloat-store} for more detailed description.

This is the default choice for i386 compiler.

@item sse
Use scalar floating point instructions present in the SSE instruction set.
This instruction set is supported by Pentium3 and newer chips, in the AMD line
by Athlon-4, Athlon-xp and Athlon-mp chips.  The earlier version of SSE
instruction set supports only single precision arithmetics, thus the double and
extended precision arithmetics is still done using 387.  Later version, present
only in Pentium4 and the future AMD x86-64 chips supports double precision
arithmetics too.

For the i386 compiler, you need to use @option{-march=@var{cpu-type}}, @option{-
or @option{-msse2} switches to enable SSE extensions and make this option
effective.  For the x86-64 compiler, these extensions are enabled by default.

The resulting code should be considerably faster in the majority of cases and av
the numerical instability problems of 387 code, but may break some existing
code that expects temporaries to be 80bit.

This is the default choice for the x86-64 compiler.

@item sse,387
@itemx sse+387
@itemx both
Attempt to utilize both instruction sets at once.  This effectively double the
amount of available registers and on chips with separate execution units for
387 and SSE the execution resources too.  Use this option with care, as it is
still experimental, because the GCC register allocator does not model separate
functional units well resulting in instable performance.
@end table

@item -masm=@var{dialect}
@opindex masm=@var{dialect}
Output asm instructions using selected @var{dialect}.  Supported
choices are @samp{intel} or @samp{att} (the default one).  Darwin does
not support @samp{intel}.

@item -mieee-fp
@itemx -mno-ieee-fp
@opindex mieee-fp
@opindex mno-ieee-fp
Control whether or not the compiler uses IEEE floating point
comparisons.  These handle correctly the case where the result of a
comparison is unordered.

@item -msoft-float
@opindex msoft-float
Generate output containing library calls for floating point.
@strong{Warning:} the requisite libraries are not part of GCC@.
Normally the facilities of the machine's usual C compiler are used, but
this can't be done directly in cross-compilation.  You must make your
own arrangements to provide suitable library functions for
cross-compilation.

On machines where a function returns floating point results in the 80387
register stack, some floating point opcodes may be emitted even if
@option{-msoft-float} is used.

@item -mno-fp-ret-in-387
@opindex mno-fp-ret-in-387
```

```
Do not use the FPU registers for return values of functions.

The usual calling convention has functions return values of types
@code{float} and @code{double} in an FPU register, even if there
is no FPU@.  The idea is that the operating system should emulate
an FPU@.

The option @option{-mno-fp-ret-in-387} causes such values to be returned
in ordinary CPU registers instead.

@item -mno-fancy-math-387
@opindex mno-fancy-math-387
Some 387 emulators do not support the @code{sin}, @code{cos} and
@code{sqrt} instructions for the 387.  Specify this option to avoid
generating those instructions.  This option is the default on FreeBSD,
OpenBSD and NetBSD@.  This option is overridden when @option{-march}
indicates that the target cpu will always have an FPU and so the
instruction will not need emulation.  As of revision 2.6.1, these
instructions are not generated unless you also use the
@option{-funsafe-math-optimizations} switch.

@item -malign-double
@itemx -mno-align-double
@opindex malign-double
@opindex mno-align-double
Control whether GCC aligns @code{double}, @code{long double}, and
@code{long long} variables on a two word boundary or a one word
boundary.  Aligning @code{double} variables on a two word boundary will
produce code that runs somewhat faster on a @samp{Pentium} at the
expense of more memory.

On x86-64, @option{-malign-double} is enabled by default.

@strong{Warning:} if you use the @option{-malign-double} switch,
structures containing the above types will be aligned differently than
the published application binary interface specifications for the 386
and will not be binary compatible with structures in code compiled
without that switch.

@item -m96bit-long-double
@itemx -m128bit-long-double
@opindex m96bit-long-double
@opindex m128bit-long-double
These switches control the size of @code{long double} type.  The i386
application binary interface specifies the size to be 96 bits,
so @option{-m96bit-long-double} is the default in 32 bit mode.

Modern architectures (Pentium and newer) would prefer @code{long double}
to be aligned to an 8 or 16 byte boundary.  In arrays or structures
conforming to the ABI, this would not be possible.  So specifying a
@option{-m128bit-long-double} will align @code{long double}
to a 16 byte boundary by padding the @code{long double} with an additional
32 bit zero.

In the x86-64 compiler, @option{-m128bit-long-double} is the default choice as
its ABI specifies that @code{long double} is to be aligned on 16 byte boundary.

Notice that neither of these options enable any extra precision over the x87
standard of 80 bits for a @code{long double}.

@strong{Warning:} if you override the default value for your target ABI, the
structures and arrays containing @code{long double} variables will change
their size as well as function calling convention for function taking
@code{long double} will be modified.  Hence they will not be binary
compatible with arrays or structures in code compiled without that switch.
```



```
@item -mlarge-data-threshold=@var{number}
@opindex mlarge-data-threshold=@var{number}
When @option{-mcmodel=medium} is specified, the data greater than
@var{threshold} are placed in large data section.  This value must be the
same across all object linked into the binary and defaults to 65535.

@item -mrtd
@opindex mrtd
Use a different function-calling convention, in which functions that
take a fixed number of arguments return with the @code{ret} @var{num}
instruction, which pops their arguments while returning.  This saves one
instruction in the caller since there is no need to pop the arguments
there.

You can specify that an individual function is called with this calling
sequence with the function attribute @samp{stdcall}.  You can also
override the @option{-mrtd} option by using the function attribute
@samp{cdecl}.  @xref{Function Attributes}.

@strong{Warning:} this calling convention is incompatible with the one
normally used on Unix, so you cannot use it if you need to call
libraries compiled with the Unix compiler.

Also, you must provide function prototypes for all functions that
take variable numbers of arguments (including @code{printf});
otherwise incorrect code will be generated for calls to those
functions.

In addition, seriously incorrect code will result if you call a
function with too many arguments.  (Normally, extra arguments are
harmlessly ignored.)

@item -mregparm=@var{num}
@opindex mregparm
Control how many registers are used to pass integer arguments.  By
default, no registers are used to pass arguments, and at most 3
registers can be used.  You can control this behavior for a specific
function by using the function attribute @samp{regparm}.
@xref{Function Attributes}.

@strong{Warning:} if you use this switch, and
@var{num} is nonzero, then you must build all modules with the same
value, including any libraries.  This includes the system libraries and
startup modules.

@item -msseregparm
@opindex msseregparm
Use SSE register passing conventions for float and double arguments
and return values.  You can control this behavior for a specific
function by using the function attribute @samp{sseregparm}.
@xref{Function Attributes}.

@strong{Warning:} if you use this switch then you must build all
modules with the same value, including any libraries.  This includes
the system libraries and startup modules.

@item -mpc32
@itemx -mpc64
@itemx -mpc80
@opindex mpc32
@opindex mpc64
@opindex mpc80

Set 80387 floating-point precision to 32, 64 or 80 bits.  When @option{-mpc32}
is specified, the significands of results of floating-point operations are
rounded to 24 bits (single precision); @option{-mpc64} rounds the
```

```
significands of results of floating-point operations to 53 bits (double
precision) and @option{-mpc80} rounds the significands of results of
floating-point operations to 64 bits (extended double precision), which is
the default.  When this option is used, floating-point operations in higher
precisions are not available to the programmer without setting the FPU
control word explicitly.

Setting the rounding of floating-point operations to less than the default
80 bits can speed some programs by 2% or more.  Note that some mathematical
libraries assume that extended precision (80 bit) floating-point operations
are enabled by default; routines in such libraries could suffer significant
loss of accuracy, typically through so-called "catastrophic cancellation",
when this option is used to set the precision to less than extended precision.

@item -mstackrealign
@opindex mstackrealign
Realign the stack at entry.  On the Intel x86, the @option{-mstackrealign}
option will generate an alternate prologue and epilogue that realigns the
runtime stack if necessary.  This supports mixing legacy codes that keep
a 4-byte aligned stack with modern codes that keep a 16-byte stack for
SSE compatibility.  See also the attribute @code{force_align_arg_pointer},
applicable to individual functions.

@item -mpreferred-stack-boundary=@var{num}
@opindex mpreferred-stack-boundary
Attempt to keep the stack boundary aligned to a 2 raised to @var{num}
byte boundary.  If @option{-mpreferred-stack-boundary} is not specified,
the default is 4 (16 bytes or 128 bits).

@item -mincoming-stack-boundary=@var{num}
@opindex mincoming-stack-boundary
Assume the incoming stack is aligned to a 2 raised to @var{num} byte
boundary.  If @option{-mincoming-stack-boundary} is not specified,
the one specified by @option{-mpreferred-stack-boundary} will be used.

On Pentium and PentiumPro, @code{double} and @code{long double} values
should be aligned to an 8 byte boundary (see @option{-malign-double}) or
suffer significant run time performance penalties.  On Pentium III, the
Streaming SIMD Extension (SSE) data type @code{__m128} may not work
properly if it is not 16 byte aligned.

To ensure proper alignment of this values on the stack, the stack boundary
must be as aligned as that required by any value stored on the stack.
Further, every function must be generated such that it keeps the stack
aligned.  Thus calling a function compiled with a higher preferred
stack boundary from a function compiled with a lower preferred stack
boundary will most likely misalign the stack.  It is recommended that
libraries that use callbacks always use the default setting.

This extra alignment does consume extra stack space, and generally
increases code size.  Code that is sensitive to stack space usage, such
as embedded systems and operating system kernels, may want to reduce the
preferred alignment to @option{-mpreferred-stack-boundary=2}.

@item -mmmx
@itemx -mno-mmx
@itemx -msse
@itemx -mno-sse
@itemx -msse2
@itemx -mno-sse2
@itemx -msse3
@itemx -mno-sse3
@itemx -mssse3
@itemx -mno-ssse3
@itemx -msse4.1
@itemx -mno-sse4.1
```



```
@itemx -msse4.2
@itemx -mno-sse4.2
@itemx -msse4
@itemx -mno-sse4
@itemx -mavx
@itemx -mno-avx
@itemx -maes
@itemx -mno-aes
@itemx -mpclmul
@itemx -mno-pclmul
@itemx -msse4a
@itemx -mno-sse4a
@itemx -msse5
@itemx -mno-sse5
@itemx -m3dnow
@itemx -mno-3dnow
@itemx -mpopcnt
@itemx -mno-popcnt
@itemx -mabm
@itemx -mno-abm
@opindex mmmx
@opindex mno-mmx
@opindex msse
@opindex mno-sse
@opindex m3dnow
@opindex mno-3dnow
These switches enable or disable the use of instructions in the MMX,
SSE, SSE2, SSE3, SSSE3, SSE4.1, AVX, AES, PCLMUL, SSE4A, SSE5, ABM or
3DNow!@: extended instruction sets.
These extensions are also available as built-in functions: see
@ref{X86 Built-in Functions}, for details of the functions enabled and
disabled by these switches.

To have SSE/SSE2 instructions generated automatically from floating-point
code (as opposed to 387 instructions), see @option{-mfpmath=sse}.

GCC depresses SSEx instructions when @option{-mavx} is used. Instead, it
generates new AVX instructions or AVX equivalence for all SSEx instructions
when needed.

These options will enable GCC to use these extended instructions in
generated code, even without @option{-mfpmath=sse}.  Applications which
perform runtime CPU detection must compile separate files for each
supported architecture, using the appropriate flags.  In particular,
the file containing the CPU detection code should be compiled without
these options.

@item -mcld
@opindex mcld
This option instructs GCC to emit a @code{cld} instruction in the prologue
of functions that use string instructions.  String instructions depend on
the DF flag to select between autoincrement or autodecrement mode.  While the
ABI specifies the DF flag to be cleared on function entry, some operating
systems violate this specification by not clearing the DF flag in their
exception dispatchers.  The exception handler can be invoked with the DF flag
set which leads to wrong direction mode, when string instructions are used.
This option can be enabled by default on 32-bit x86 targets by configuring
GCC with the @option{--enable-cld} configure option.  Generation of @code{cld}
instructions can be suppressed with the @option{-mno-cld} compiler option
in this case.

@item -mcx16
@opindex mcx16
This option will enable GCC to use CMPXCHG16B instruction in generated code.
CMPXCHG16B allows for atomic operations on 128-bit double quadword (or oword)
data types.  This is useful for high resolution counters that could be updated
```

```
by multiple processors (or cores).  This instruction is generated as part of
atomic built-in functions: see @ref{Atomic Builtins} for details.

@item -msahf
@opindex msahf
This option will enable GCC to use SAHF instruction in generated 64-bit code.
Early Intel CPUs with Intel 64 lacked LAHF and SAHF instructions supported
by AMD64 until introduction of Pentium 4 G1 step in December 2005.  LAHF and
SAHF are load and store instructions, respectively, for certain status flags.
In 64-bit mode, SAHF instruction is used to optimize @code{fmod}, @code{drem}
or @code{remainder} built-in functions: see @ref{Other Builtins} for details.

@item -mrecip
@opindex mrecip
This option will enable GCC to use RCPSS and RSQRTSS instructions (and their
vectorized variants RCPPS and RSQRTPS) with an additional Newton-Raphson step
to increase precision instead of DIVSS and SQRTSS (and their vectorized
variants) for single precision floating point arguments.  These instructions
are generated only when @option{-funsafe-math-optimizations} is enabled
together with @option{-finite-math-only} and @option{-fno-trapping-math}.
Note that while the throughput of the sequence is higher than the throughput
of the non-reciprocal instruction, the precision of the sequence can be
decreased by up to 2 ulp (i.e. the inverse of 1.0 equals 0.99999994).

@item -mveclibabi=@var{type}
@opindex mveclibabi
Specifies the ABI type to use for vectorizing intrinsics using an
external library.  Supported types are @code{svml} for the Intel short
vector math library and @code{acml} for the AMD math core library style
of interfacing.  GCC will currently emit calls to @code{vmldExp2},
@code{vmldLn2}, @code{vmldLog102}, @code{vmldLog102}, @code{vmldPow2},
@code{vmldTanh2}, @code{vmldTan2}, @code{vmldAtan2}, @code{vmldAtanh2},
@code{vmldCbrt2}, @code{vmldSinh2}, @code{vmldSin2}, @code{vmldAsinh2},
@code{vmldAsin2}, @code{vmldCosh2}, @code{vmldCos2}, @code{vmldAcosh2},
@code{vmldAcos2}, @code{vmlsExp4}, @code{vmlsLn4}, @code{vmlsLog104},
@code{vmlsLog104}, @code{vmlsPow4}, @code{vmlsTanh4}, @code{vmlsTan4},
@code{vmlsAtan4}, @code{vmlsAtanh4}, @code{vmlsCbrt4}, @code{vmlsSinh4},
@code{vmlsSin4}, @code{vmlsAsinh4}, @code{vmlsAsin4}, @code{vmlsCosh4},
@code{vmlsCos4}, @code{vmlsAcosh4} and @code{vmlsAcos4} for corresponding
function type when @option{-mveclibabi=svml} is used and @code{__vrd2_sin},
@code{__vrd2_cos}, @code{__vrd2_exp}, @code{__vrd2_log}, @code{__vrd2_log2},
@code{__vrd2_log10}, @code{__vrs4_sinf}, @code{__vrs4_cosf},
@code{__vrs4_expf}, @code{__vrs4_logf}, @code{__vrs4_log2f},
@code{__vrs4_log10f} and @code{__vrs4_powf} for corresponding function type
when @option{-mveclibabi=acml} is used. Both @option{-ftree-vectorize} and
@option{-funsafe-math-optimizations} have to be enabled. A SVML or ACML ABI
compatible library will have to be specified at link time.

@item -mpush-args
@itemx -mno-push-args
@opindex mpush-args
@opindex mno-push-args
Use PUSH operations to store outgoing parameters.  This method is shorter
and usually equally fast as method using SUB/MOV operations and is enabled
by default.  In some cases disabling it may improve performance because of
improved scheduling and reduced dependencies.

@item -maccumulate-outgoing-args
@opindex maccumulate-outgoing-args
If enabled, the maximum amount of space required for outgoing arguments will be
computed in the function prologue.  This is faster on most modern CPUs
because of reduced dependencies, improved scheduling and reduced stack usage
when preferred stack boundary is not equal to 2.  The drawback is a notable
increase in code size.  This switch implies @option{-mno-push-args}.

@item -mthreads
```



```
@opindex mthreads
Support thread-safe exception handling on @samp{Mingw32}.  Code that relies
on thread-safe exception handling must compile and link all code with the
@option{-mthreads} option.  When compiling, @option{-mthreads} defines
@option{-D_MT}; when linking, it links in a special thread helper library
@option{-lmingwthrd} which cleans up per thread exception handling data.

@item -mno-align-stringops
@opindex mno-align-stringops
Do not align destination of inlined string operations.  This switch reduces
code size and improves performance in case the destination is already aligned,
but GCC doesn't know about it.

@item -minline-all-stringops
@opindex minline-all-stringops
By default GCC inlines string operations only when destination is known to be
aligned at least to 4 byte boundary.  This enables more inlining, increase code
size, but may improve performance of code that depends on fast memcpy, strlen
and memset for short lengths.

@item -minline-stringops-dynamically
@opindex minline-stringops-dynamically
For string operation of unknown size, inline runtime checks so for small
blocks inline code is used, while for large blocks library call is used.

@item -mstringop-strategy=@var{alg}
@opindex mstringop-strategy=@var{alg}
Overwrite internal decision heuristic about particular algorithm to inline
string operation with.  The allowed values are @code{rep_byte},
@code{rep_4byte}, @code{rep_8byte} for expanding using i386 @code{rep} prefix
of specified size, @code{byte_loop}, @code{loop}, @code{unrolled_loop} for
expanding inline loop, @code{libcall} for always expanding library call.

@item -momit-leaf-frame-pointer
@opindex momit-leaf-frame-pointer
Don't keep the frame pointer in a register for leaf functions.  This
avoids the instructions to save, set up and restore frame pointers and
makes an extra register available in leaf functions.  The option
@option{-fomit-frame-pointer} removes the frame pointer for all functions
which might make debugging harder.

@item -mtls-direct-seg-refs
@itemx -mno-tls-direct-seg-refs
@opindex mtls-direct-seg-refs
Controls whether TLS variables may be accessed with offsets from the
TLS segment register (@code{%gs} for 32-bit, @code{%fs} for 64-bit),
or whether the thread base pointer must be added.  Whether or not this
is legal depends on the operating system, and whether it maps the
segment to cover the entire TLS area.

For systems that use GNU libc, the default is on.

@item -mfused-madd
@itemx -mno-fused-madd
@opindex mfused-madd
Enable automatic generation of fused floating point multiply-add instructions
if the ISA supports such instructions.  The -mfused-madd option is on by
default.  The fused multiply-add instructions have a different
rounding behavior compared to executing a multiply followed by an add.

@item -msse2avx
@itemx -mno-sse2avx
@opindex msse2avx
Specify that the assembler should encode SSE instructions with VEX
prefix.  The option @option{-mavx} turns this on by default.
@end table
```

```
These @samp{-m} switches are supported in addition to the above
on AMD x86-64 processors in 64-bit environments.

@table @gcctabopt
@item -m32
@itemx -m64
@opindex m32
@opindex m64
Generate code for a 32-bit or 64-bit environment.
The 32-bit environment sets int, long and pointer to 32 bits and
generates code that runs on any i386 system.
The 64-bit environment sets int to 32 bits and long and pointer
to 64 bits and generates code for AMD's x86-64 architecture. For
darwin only the -m64 option turns off the @option{-fno-pic} and
@option{-mdynamic-no-pic} options.

@item -mno-red-zone
@opindex no-red-zone
Do not use a so called red zone for x86-64 code.  The red zone is mandated
by the x86-64 ABI, it is a 128-byte area beyond the location of the
stack pointer that will not be modified by signal or interrupt handlers
and therefore can be used for temporary data without adjusting the stack
pointer.  The flag @option{-mno-red-zone} disables this red zone.

@item -mcmodel=small
@opindex mcmodel=small
Generate code for the small code model: the program and its symbols must
be linked in the lower 2 GB of the address space.  Pointers are 64 bits.
Programs can be statically or dynamically linked.  This is the default
code model.

@item -mcmodel=kernel
@opindex mcmodel=kernel
Generate code for the kernel code model.  The kernel runs in the
negative 2 GB of the address space.
This model has to be used for Linux kernel code.

@item -mcmodel=medium
@opindex mcmodel=medium
Generate code for the medium model: The program is linked in the lower 2
GB of the address space.  Small symbols are also placed there.  Symbols
with sizes larger than @option{-mlarge-data-threshold} are put into
large data or bss sections and can be located above 2GB.  Programs can
be statically or dynamically linked.

@item -mcmodel=large
@opindex mcmodel=large
Generate code for the large model: This model makes no assumptions
about addresses and sizes of sections.

@item -msave-args
@opindex msave-args
Save integer arguments on the stack at function entry.
@end table

@node IA-64 Options
@subsection IA-64 Options
@cindex IA-64 Options

These are the @samp{-m} options defined for the Intel IA-64 architecture.

@table @gcctabopt
@item -mbig-endian
@opindex mbig-endian
Generate code for a big endian target.  This is the default for HP-UX@.
```



```
@item -mlittle-endian
@opindex mlittle-endian
Generate code for a little endian target.  This is the default for AIX5
and GNU/Linux.

@item -mgnu-as
@itemx -mno-gnu-as
@opindex mgnu-as
@opindex mno-gnu-as
Generate (or don't) code for the GNU assembler.  This is the default.
@c Also, this is the default if the configure option @option{--with-gnu-as}
@c is used.

@item -mgnu-ld
@itemx -mno-gnu-ld
@opindex mgnu-ld
@opindex mno-gnu-ld
Generate (or don't) code for the GNU linker.  This is the default.
@c Also, this is the default if the configure option @option{--with-gnu-ld}
@c is used.

@item -mno-pic
@opindex mno-pic
Generate code that does not use a global pointer register.  The result
is not position independent code, and violates the IA-64 ABI@.

@item -mvolatile-asm-stop
@itemx -mno-volatile-asm-stop
@opindex mvolatile-asm-stop
@opindex mno-volatile-asm-stop
Generate (or don't) a stop bit immediately before and after volatile asm
statements.

@item -mregister-names
@itemx -mno-register-names
@opindex mregister-names
@opindex mno-register-names
Generate (or don't) @samp{in}, @samp{loc}, and @samp{out} register names for
the stacked registers.  This may make assembler output more readable.

@item -mno-sdata
@itemx -msdata
@opindex mno-sdata
@opindex msdata
Disable (or enable) optimizations that use the small data section.  This may
be useful for working around optimizer bugs.

@item -mconstant-gp
@opindex mconstant-gp
Generate code that uses a single constant global pointer value.  This is
useful when compiling kernel code.

@item -mauto-pic
@opindex mauto-pic
Generate code that is self-relocatable.  This implies @option{-mconstant-gp}.
This is useful when compiling firmware code.

@item -minline-float-divide-min-latency
@opindex minline-float-divide-min-latency
Generate code for inline divides of floating point values
using the minimum latency algorithm.

@item -minline-float-divide-max-throughput
@opindex minline-float-divide-max-throughput
Generate code for inline divides of floating point values
```

```
using the maximum throughput algorithm.

@item -minline-int-divide-min-latency
@opindex minline-int-divide-min-latency
Generate code for inline divides of integer values
using the minimum latency algorithm.

@item -minline-int-divide-max-throughput
@opindex minline-int-divide-max-throughput
Generate code for inline divides of integer values
using the maximum throughput algorithm.

@item -minline-sqrt-min-latency
@opindex minline-sqrt-min-latency
Generate code for inline square roots
using the minimum latency algorithm.

@item -minline-sqrt-max-throughput
@opindex minline-sqrt-max-throughput
Generate code for inline square roots
using the maximum throughput algorithm.

@item -mno-dwarf2-asm
@itemx -mdwarf2-asm
@opindex mno-dwarf2-asm
@opindex mdwarf2-asm
Don't (or do) generate assembler code for the DWARF2 line number debugging
info.  This may be useful when not using the GNU assembler.

@item -mearly-stop-bits
@itemx -mno-early-stop-bits
@opindex mearly-stop-bits
@opindex mno-early-stop-bits
Allow stop bits to be placed earlier than immediately preceding the
instruction that triggered the stop bit.  This can improve instruction
scheduling, but does not always do so.

@item -mfixed-range=@var{register-range}
@opindex mfixed-range
Generate code treating the given register range as fixed registers.
A fixed register is one that the register allocator can not use.  This is
useful when compiling kernel code.  A register range is specified as
two registers separated by a dash.  Multiple register ranges can be
specified separated by a comma.

@item -mtls-size=@var{tls-size}
@opindex mtls-size
Specify bit size of immediate TLS offsets.  Valid values are 14, 22, and
64.

@item -mtune=@var{cpu-type}
@opindex mtune
Tune the instruction scheduling for a particular CPU, Valid values are
itanium, itanium1, merced, itanium2, and mckinley.

@item -mt
@itemx -pthread
@opindex mt
@opindex pthread
Add support for multithreading using the POSIX threads library.  This
option sets flags for both the preprocessor and linker.  It does
not affect the thread safety of object code produced by the compiler or
that of libraries supplied with it.  These are HP-UX specific flags.

@item -milp32
@itemx -mlp64
```



```
@opindex milp32
@opindex mlp64
Generate code for a 32-bit or 64-bit environment.
The 32-bit environment sets int, long and pointer to 32 bits.
The 64-bit environment sets int to 32 bits and long and pointer
to 64 bits.  These are HP-UX specific flags.

@item -mno-sched-br-data-spec
@itemx -msched-br-data-spec
@opindex mno-sched-br-data-spec
@opindex msched-br-data-spec
(Dis/En)able data speculative scheduling before reload.
This will result in generation of the ld.a instructions and
the corresponding check instructions (ld.c / chk.a).
The default is 'disable'.

@item -msched-ar-data-spec
@itemx -mno-sched-ar-data-spec
@opindex msched-ar-data-spec
@opindex mno-sched-ar-data-spec
(En/Dis)able data speculative scheduling after reload.
This will result in generation of the ld.a instructions and
the corresponding check instructions (ld.c / chk.a).
The default is 'enable'.

@item -mno-sched-control-spec
@itemx -msched-control-spec
@opindex mno-sched-control-spec
@opindex msched-control-spec
(Dis/En)able control speculative scheduling.  This feature is
available only during region scheduling (i.e.@: before reload).
This will result in generation of the ld.s instructions and
the corresponding check instructions chk.s .
The default is 'disable'.

@item -msched-br-in-data-spec
@itemx -mno-sched-br-in-data-spec
@opindex msched-br-in-data-spec
@opindex mno-sched-br-in-data-spec
(En/Dis)able speculative scheduling of the instructions that
are dependent on the data speculative loads before reload.
This is effective only with @option{-msched-br-data-spec} enabled.
The default is 'enable'.

@item -msched-ar-in-data-spec
@itemx -mno-sched-ar-in-data-spec
@opindex msched-ar-in-data-spec
@opindex mno-sched-ar-in-data-spec
(En/Dis)able speculative scheduling of the instructions that
are dependent on the data speculative loads after reload.
This is effective only with @option{-msched-ar-data-spec} enabled.
The default is 'enable'.

@item -msched-in-control-spec
@itemx -mno-sched-in-control-spec
@opindex msched-in-control-spec
@opindex mno-sched-in-control-spec
(En/Dis)able speculative scheduling of the instructions that
are dependent on the control speculative loads.
This is effective only with @option{-msched-control-spec} enabled.
The default is 'enable'.

@item -msched-ldc
@itemx -mno-sched-ldc
@opindex msched-ldc
@opindex mno-sched-ldc
```

```
(En/Dis)able use of simple data speculation checks ld.c .
If disabled, only chk.a instructions will be emitted to check
data speculative loads.
The default is 'enable'.

@item -mno-sched-control-ldc
@itemx -msched-control-ldc
@opindex mno-sched-control-ldc
@opindex msched-control-ldc
(Dis/En)able use of ld.c instructions to check control speculative loads.
If enabled, in case of control speculative load with no speculatively
scheduled dependent instructions this load will be emitted as ld.sa and
ld.c will be used to check it.
The default is 'disable'.

@item -mno-sched-spec-verbose
@itemx -msched-spec-verbose
@opindex mno-sched-spec-verbose
@opindex msched-spec-verbose
(Dis/En)able printing of the information about speculative motions.

@item -mno-sched-prefer-non-data-spec-insns
@itemx -msched-prefer-non-data-spec-insns
@opindex mno-sched-prefer-non-data-spec-insns
@opindex msched-prefer-non-data-spec-insns
If enabled, data speculative instructions will be chosen for schedule
only if there are no other choices at the moment.  This will make
the use of the data speculation much more conservative.
The default is 'disable'.

@item -mno-sched-prefer-non-control-spec-insns
@itemx -msched-prefer-non-control-spec-insns
@opindex mno-sched-prefer-non-control-spec-insns
@opindex msched-prefer-non-control-spec-insns
If enabled, control speculative instructions will be chosen for schedule
only if there are no other choices at the moment.  This will make
the use of the control speculation much more conservative.
The default is 'disable'.

@item -mno-sched-count-spec-in-critical-path
@itemx -msched-count-spec-in-critical-path
@opindex mno-sched-count-spec-in-critical-path
@opindex msched-count-spec-in-critical-path
If enabled, speculative dependencies will be considered during
computation of the instructions priorities.  This will make the use of the
speculation a bit more conservative.
The default is 'disable'.

@end table

@node M32C Options
@subsection M32C Options
@cindex M32C options

@table @gcctabopt
@item -mcpu=@var{name}
@opindex mcpu=
Select the CPU for which code is generated.  @var{name} may be one of
@samp{r8c} for the R8C/Tiny series, @samp{m16c} for the M16C (up to
/60) series, @samp{m32cm} for the M16C/80 series, or @samp{m32c} for
the M32C/80 series.

@item -msim
@opindex msim
Specifies that the program will be run on the simulator.  This causes
an alternate runtime library to be linked in which supports, for
```



```
example, file I/O@.  You must not use this option when generating
programs that will run on real hardware; you must provide your own
runtime library for whatever I/O functions are needed.

@item -memregs=@var{number}
@opindex memregs=
Specifies the number of memory-based pseudo-registers GCC will use
during code generation.  These pseudo-registers will be used like real
registers, so there is a tradeoff between GCC's ability to fit the
code into available registers, and the performance penalty of using
memory instead of registers.  Note that all modules in a program must
be compiled with the same value for this option.  Because of that, you
must not use this option with the default runtime libraries gcc
builds.

@end table

@node M32R/D Options
@subsection M32R/D Options
@cindex M32R/D options

These @option{-m} options are defined for Renesas M32R/D architectures:

@table @gcctabopt
@item -m32r2
@opindex m32r2
Generate code for the M32R/2@.

@item -m32rx
@opindex m32rx
Generate code for the M32R/X@.

@item -m32r
@opindex m32r
Generate code for the M32R@.  This is the default.

@item -mmodel=small
@opindex mmodel=small
Assume all objects live in the lower 16MB of memory (so that their addresses
can be loaded with the @code{ld24} instruction), and assume all subroutines
are reachable with the @code{bl} instruction.
This is the default.

The addressability of a particular object can be set with the
@code{model} attribute.

@item -mmodel=medium
@opindex mmodel=medium
Assume objects may be anywhere in the 32-bit address space (the compiler
will generate @code{seth/add3} instructions to load their addresses), and
assume all subroutines are reachable with the @code{bl} instruction.

@item -mmodel=large
@opindex mmodel=large
Assume objects may be anywhere in the 32-bit address space (the compiler
will generate @code{seth/add3} instructions to load their addresses), and
assume subroutines may not be reachable with the @code{bl} instruction
(the compiler will generate the much slower @code{seth/add3/jl}
instruction sequence).

@item -msdata=none
@opindex msdata=none
Disable use of the small data area.  Variables will be put into
one of @samp{.data}, @samp{bss}, or @samp{.rodata} (unless the
@code{section} attribute has been specified).
This is the default.
```

```
The small data area consists of sections @samp{.sdata} and @samp{.sbss}.
Objects may be explicitly put in the small data area with the
@code{section} attribute using one of these sections.

@item -msdata=sdata
@opindex msdata=sdata
Put small global and static data in the small data area, but do not
generate special code to reference them.

@item -msdata=use
@opindex msdata=use
Put small global and static data in the small data area, and generate
special instructions to reference them.

@item -G @var{num}
@opindex G
@cindex smaller data references
Put global and static objects less than or equal to @var{num} bytes
into the small data or bss sections instead of the normal data or bss
sections.  The default value of @var{num} is 8.
The @option{-msdata} option must be set to one of @samp{sdata} or @samp{use}
for this option to have any effect.

All modules should be compiled with the same @option{-G @var{num}} value.
Compiling with different values of @var{num} may or may not work; if it
doesn't the linker will give an error message---incorrect code will not be
generated.

@item -mdebug
@opindex mdebug
Makes the M32R specific code in the compiler display some statistics
that might help in debugging programs.

@item -malign-loops
@opindex malign-loops
Align all loops to a 32-byte boundary.

@item -mno-align-loops
@opindex mno-align-loops
Do not enforce a 32-byte alignment for loops.  This is the default.

@item -missue-rate=@var{number}
@opindex missue-rate=@var{number}
Issue @var{number} instructions per cycle.  @var{number} can only be 1
or 2.

@item -mbranch-cost=@var{number}
@opindex mbranch-cost=@var{number}
@var{number} can only be 1 or 2.  If it is 1 then branches will be
preferred over conditional code, if it is 2, then the opposite will
apply.

@item -mflush-trap=@var{number}
@opindex mflush-trap=@var{number}
Specifies the trap number to use to flush the cache.  The default is
12.  Valid numbers are between 0 and 15 inclusive.

@item -mno-flush-trap
@opindex mno-flush-trap
Specifies that the cache cannot be flushed by using a trap.

@item -mflush-func=@var{name}
@opindex mflush-func=@var{name}
Specifies the name of the operating system function to call to flush
the cache.  The default is @emph{_flush_cache}, but a function call
```



```
will only be used if a trap is not available.

@item -mno-flush-func
@opindex mno-flush-func
Indicates that there is no OS function for flushing the cache.

@end table

@node M680x0 Options
@subsection M680x0 Options
@cindex M680x0 options

These are the @samp{-m} options defined for M680x0 and ColdFire processors.
The default settings depend on which architecture was selected when
the compiler was configured; the defaults for the most common choices
are given below.

@table @gcctabopt
@item -march=@var{arch}
@opindex march
Generate code for a specific M680x0 or ColdFire instruction set
architecture.  Permissible values of @var{arch} for M680x0
architectures are: @samp{68000}, @samp{68010}, @samp{68020},
@samp{68030}, @samp{68040}, @samp{68060} and @samp{cpu32}.  ColdFire
architectures are selected according to Freescale's ISA classification
and the permissible values are: @samp{isaa}, @samp{isaaplus},
@samp{isab} and @samp{isac}.

gcc defines a macro @samp{__mcf@var{arch}__} whenever it is generating
code for a ColdFire target.  The @var{arch} in this macro is one of the
@option{-march} arguments given above.

When used together, @option{-march} and @option{-mtune} select code
that runs on a family of similar processors but that is optimized
for a particular microarchitecture.

@item -mcpu=@var{cpu}
@opindex mcpu
Generate code for a specific M680x0 or ColdFire processor.
The M680x0 @var{cpu}s are: @samp{68000}, @samp{68010}, @samp{68020},
@samp{68030}, @samp{68040}, @samp{68060}, @samp{68302}, @samp{68332}
and @samp{cpu32}.  The ColdFire @var{cpu}s are given by the table
below, which also classifies the CPUs into families:

@multitable @columnfractions 0.20 0.80
@item @strong{Family} @tab @strong{@samp{-mcpu} arguments}
@item @samp{51qe} @tab @samp{51qe}
@item @samp{5206} @tab @samp{5202} @samp{5204} @samp{5206}
@item @samp{5206e} @tab @samp{5206e}
@item @samp{5208} @tab @samp{5207} @samp{5208}
@item @samp{5211a} @tab @samp{5210a} @samp{5211a}
@item @samp{5213} @tab @samp{5211} @samp{5212} @samp{5213}
@item @samp{5216} @tab @samp{5214} @samp{5216}
@item @samp{52235} @tab @samp{52230} @samp{52231} @samp{52232} @samp{52233} @sam
@item @samp{5225} @tab @samp{5224} @samp{5225}
@item @samp{5235} @tab @samp{5232} @samp{5233} @samp{5234} @samp{5235} @samp{523
@item @samp{5249} @tab @samp{5249}
@item @samp{5250} @tab @samp{5250}
@item @samp{5271} @tab @samp{5270} @samp{5271}
@item @samp{5272} @tab @samp{5272}
@item @samp{5275} @tab @samp{5274} @samp{5275}
@item @samp{5282} @tab @samp{5280} @samp{5281} @samp{5282} @samp{528x}
@item @samp{5307} @tab @samp{5307}
@item @samp{5329} @tab @samp{5327} @samp{5328} @samp{5329} @samp{532x}
@item @samp{5373} @tab @samp{5372} @samp{5373} @samp{537x}
@item @samp{5407} @tab @samp{5407}
```

```
@item @samp{5475} @tab @samp{5470} @samp{5471} @samp{5472} @samp{5473} @samp{547
@end multitable

@option{-mcpu=@var{cpu}} overrides @option{-march=@var{arch}} if
@var{arch} is compatible with @var{cpu}.  Other combinations of
@option{-mcpu} and @option{-march} are rejected.

gcc defines the macro @samp{__mcf_cpu_@var{cpu}} when ColdFire target
@var{cpu} is selected.  It also defines @samp{__mcf_family_@var{family}},
where the value of @var{family} is given by the table above.

@item -mtune=@var{tune}
@opindex mtune
Tune the code for a particular microarchitecture, within the
constraints set by @option{-march} and @option{-mcpu}.
The M680x0 microarchitectures are: @samp{68000}, @samp{68010},
@samp{68020}, @samp{68030}, @samp{68040}, @samp{68060}
and @samp{cpu32}.  The ColdFire microarchitectures
are: @samp{cfv1}, @samp{cfv2}, @samp{cfv3}, @samp{cfv4} and @samp{cfv4e}.

You can also use @option{-mtune=68020-40} for code that needs
to run relatively well on 68020, 68030 and 68040 targets.
@option{-mtune=68020-60} is similar but includes 68060 targets
as well.  These two options select the same tuning decisions as
@option{-m68020-40} and @option{-m68020-60} respectively.

gcc defines the macros @samp{__mc@var{arch}} and @samp{__mc@var{arch}__}
when tuning for 680x0 architecture @var{arch}.  It also defines
@samp{mc@var{arch}} unless either @option{-ansi} or a non-GNU @option{-std}
option is used.  If gcc is tuning for a range of architectures,
as selected by @option{-mtune=68020-40} or @option{-mtune=68020-60},
it defines the macros for every architecture in the range.

gcc also defines the macro @samp{__m@var{uarch}__} when tuning for
ColdFire microarchitecture @var{uarch}, where @var{uarch} is one
of the arguments given above.

@item -m68000
@itemx -mc68000
@opindex m68000
@opindex mc68000
Generate output for a 68000.  This is the default
when the compiler is configured for 68000-based systems.
It is equivalent to @option{-march=68000}.

Use this option for microcontrollers with a 68000 or EC000 core,
including the 68008, 68302, 68306, 68307, 68322, 68328 and 68356.

@item -m68010
@opindex m68010
Generate output for a 68010.  This is the default
when the compiler is configured for 68010-based systems.
It is equivalent to @option{-march=68010}.

@item -m68020
@itemx -mc68020
@opindex m68020
@opindex mc68020
Generate output for a 68020.  This is the default
when the compiler is configured for 68020-based systems.
It is equivalent to @option{-march=68020}.

@item -m68030
@opindex m68030
Generate output for a 68030.  This is the default when the compiler is
configured for 68030-based systems.  It is equivalent to
```



```
@option{-march=68030}.

@item -m68040
@opindex m68040
Generate output for a 68040.  This is the default when the compiler is
configured for 68040-based systems.  It is equivalent to
@option{-march=68040}.

This option inhibits the use of 68881/68882 instructions that have to be
emulated by software on the 68040.  Use this option if your 68040 does not
have code to emulate those instructions.

@item -m68060
@opindex m68060
Generate output for a 68060.  This is the default when the compiler is
configured for 68060-based systems.  It is equivalent to
@option{-march=68060}.

This option inhibits the use of 68020 and 68881/68882 instructions that
have to be emulated by software on the 68060.  Use this option if your 68060
does not have code to emulate those instructions.

@item -mcpu32
@opindex mcpu32
Generate output for a CPU32.  This is the default
when the compiler is configured for CPU32-based systems.
It is equivalent to @option{-march=cpu32}.

Use this option for microcontrollers with a
CPU32 or CPU32+ core, including the 68330, 68331, 68332, 68333, 68334,
68336, 68340, 68341, 68349 and 68360.

@item -m5200
@opindex m5200
Generate output for a 520X ColdFire CPU@.  This is the default
when the compiler is configured for 520X-based systems.
It is equivalent to @option{-mcpu=5206}, and is now deprecated
in favor of that option.

Use this option for microcontroller with a 5200 core, including
the MCF5202, MCF5203, MCF5204 and MCF5206.

@item -m5206e
@opindex m5206e
Generate output for a 5206e ColdFire CPU@.  The option is now
deprecated in favor of the equivalent @option{-mcpu=5206e}.

@item -m528x
@opindex m528x
Generate output for a member of the ColdFire 528X family.
The option is now deprecated in favor of the equivalent
@option{-mcpu=528x}.

@item -m5307
@opindex m5307
Generate output for a ColdFire 5307 CPU@.  The option is now deprecated
in favor of the equivalent @option{-mcpu=5307}.

@item -m5407
@opindex m5407
Generate output for a ColdFire 5407 CPU@.  The option is now deprecated
in favor of the equivalent @option{-mcpu=5407}.

@item -mcfv4e
@opindex mcfv4e
Generate output for a ColdFire V4e family CPU (e.g.@: 547x/548x).
```

```
This includes use of hardware floating point instructions.
The option is equivalent to @option{-mcpu=547x}, and is now
deprecated in favor of that option.

@item -m68020-40
@opindex m68020-40
Generate output for a 68040, without using any of the new instructions.
This results in code which can run relatively efficiently on either a
68020/68881 or a 68030 or a 68040.  The generated code does use the
68881 instructions that are emulated on the 68040.

The option is equivalent to @option{-march=68020} @option{-mtune=68020-40}.

@item -m68020-60
@opindex m68020-60
Generate output for a 68060, without using any of the new instructions.
This results in code which can run relatively efficiently on either a
68020/68881 or a 68030 or a 68040.  The generated code does use the
68881 instructions that are emulated on the 68060.

The option is equivalent to @option{-march=68020} @option{-mtune=68020-60}.

@item -mhard-float
@itemx -m68881
@opindex mhard-float
@opindex m68881
Generate floating-point instructions.  This is the default for 68020
and above, and for ColdFire devices that have an FPU@.  It defines the
macro @samp{__HAVE_68881__} on M680x0 targets and @samp{__mcffpu__}
on ColdFire targets.

@item -msoft-float
@opindex msoft-float
Do not generate floating-point instructions; use library calls instead.
This is the default for 68000, 68010, and 68832 targets.  It is also
the default for ColdFire devices that have no FPU.

@item -mdiv
@itemx -mno-div
@opindex mdiv
@opindex mno-div
Generate (do not generate) ColdFire hardware divide and remainder
instructions.  If @option{-march} is used without @option{-mcpu},
the default is ``on'' for ColdFire architectures and ``off'' for M680x0
architectures.  Otherwise, the default is taken from the target CPU
(either the default CPU, or the one specified by @option{-mcpu}).  For
example, the default is ``off'' for @option{-mcpu=5206} and ``on'' for
@option{-mcpu=5206e}.

gcc defines the macro @samp{__mcfhwdiv__} when this option is enabled.

@item -mshort
@opindex mshort
Consider type @code{int} to be 16 bits wide, like @code{short int}.
Additionally, parameters passed on the stack are also aligned to a
16-bit boundary even on targets whose API mandates promotion to 32-bit.

@item -mno-short
@opindex mno-short
Do not consider type @code{int} to be 16 bits wide.  This is the default.

@item -mnobitfield
@itemx -mno-bitfield
@opindex mnobitfield
@opindex mno-bitfield
Do not use the bit-field instructions.  The @option{-m68000}, @option{-mcpu32}
```



```
and @option{-m5200} options imply @w{@option{-mnobitfield}}.

@item -mbitfield
@opindex mbitfield
Do use the bit-field instructions.  The @option{-m68020} option implies
@option{-mbitfield}.  This is the default if you use a configuration
designed for a 68020.

@item -mrtd
@opindex mrtd
Use a different function-calling convention, in which functions
that take a fixed number of arguments return with the @code{rtd}
instruction, which pops their arguments while returning.  This
saves one instruction in the caller since there is no need to pop
the arguments there.

This calling convention is incompatible with the one normally
used on Unix, so you cannot use it if you need to call libraries
compiled with the Unix compiler.

Also, you must provide function prototypes for all functions that
take variable numbers of arguments (including @code{printf});
otherwise incorrect code will be generated for calls to those
functions.

In addition, seriously incorrect code will result if you call a
function with too many arguments.  (Normally, extra arguments are
harmlessly ignored.)

The @code{rtd} instruction is supported by the 68010, 68020, 68030,
68040, 68060 and CPU32 processors, but not by the 68000 or 5200.

@item -mno-rtd
@opindex mno-rtd
Do not use the calling conventions selected by @option{-mrtd}.
This is the default.

@item -malign-int
@itemx -mno-align-int
@opindex malign-int
@opindex mno-align-int
Control whether GCC aligns @code{int}, @code{long}, @code{long long},
@code{float}, @code{double}, and @code{long double} variables on a 32-bit
boundary (@option{-malign-int}) or a 16-bit boundary (@option{-mno-align-int}).
Aligning variables on 32-bit boundaries produces code that runs somewhat
faster on processors with 32-bit busses at the expense of more memory.

@strong{Warning:} if you use the @option{-malign-int} switch, GCC will
align structures containing the above types  differently than
most published application binary interface specifications for the m68k.

@item -mpcrel
@opindex mpcrel
Use the pc-relative addressing mode of the 68000 directly, instead of
using a global offset table.  At present, this option implies @option{-fpic},
allowing at most a 16-bit offset for pc-relative addressing.  @option{-fPIC} is
not presently supported with @option{-mpcrel}, though this could be supported fo
68020 and higher processors.

@item -mno-strict-align
@itemx -mstrict-align
@opindex mno-strict-align
@opindex mstrict-align
Do not (do) assume that unaligned memory references will be handled by
the system.
```

```
@item -msep-data
Generate code that allows the data segment to be located in a different
area of memory from the text segment.  This allows for execute in place in
an environment without virtual memory management.  This option implies
@option{-fPIC}.

@item -mno-sep-data
Generate code that assumes that the data segment follows the text segment.
This is the default.

@item -mid-shared-library
Generate code that supports shared libraries via the library ID method.
This allows for execute in place and shared libraries in an environment
without virtual memory management.  This option implies @option{-fPIC}.

@item -mno-id-shared-library
Generate code that doesn't assume ID based shared libraries are being used.
This is the default.

@item -mshared-library-id=n
Specified the identification number of the ID based shared library being
compiled.  Specifying a value of 0 will generate more compact code, specifying
other values will force the allocation of that number to the current
library but is no more space or time efficient than omitting this option.

@item -mxgot
@itemx -mno-xgot
@opindex mxgot
@opindex mno-xgot
When generating position-independent code for ColdFire, generate code
that works if the GOT has more than 8192 entries.  This code is
larger and slower than code generated without this option.  On M680x0
processors, this option is not needed; @option{-fPIC} suffices.

GCC normally uses a single instruction to load values from the GOT@.
While this is relatively efficient, it only works if the GOT
is smaller than about 64k.  Anything larger causes the linker
to report an error such as:

@cindex relocation truncated to fit (ColdFire)
@smallexample
relocation truncated to fit: R_68K_GOT16O foobar
@end smallexample

If this happens, you should recompile your code with @option{-mxgot}.
It should then work with very large GOTs.  However, code generated with
@option{-mxgot} is less efficient, since it takes 4 instructions to fetch
the value of a global symbol.

Note that some linkers, including newer versions of the GNU linker,
can create multiple GOTs and sort GOT entries.  If you have such a linker,
you should only need to use @option{-mxgot} when compiling a single
object file that accesses more than 8192 GOT entries.  Very few do.

These options have no effect unless GCC is generating
position-independent code.

@end table

@node M68hc1x Options
@subsection M68hc1x Options
@cindex M68hc1x options

These are the @samp{-m} options defined for the 68hc11 and 68hc12
microcontrollers.  The default values for these options depends on
which style of microcontroller was selected when the compiler was configured;
```



```
the defaults for the most common choices are given below.

@table @gcctabopt
@item -m6811
@itemx -m68hc11
@opindex m6811
@opindex m68hc11
Generate output for a 68HC11.  This is the default
when the compiler is configured for 68HC11-based systems.

@item -m6812
@itemx -m68hc12
@opindex m6812
@opindex m68hc12
Generate output for a 68HC12.  This is the default
when the compiler is configured for 68HC12-based systems.

@item -m68S12
@itemx -m68hcs12
@opindex m68S12
@opindex m68hcs12
Generate output for a 68HCS12.

@item -mauto-incdec
@opindex mauto-incdec
Enable the use of 68HC12 pre and post auto-increment and auto-decrement
addressing modes.

@item -minmax
@itemx -nominmax
@opindex minmax
@opindex mnominmax
Enable the use of 68HC12 min and max instructions.

@item -mlong-calls
@itemx -mno-long-calls
@opindex mlong-calls
@opindex mno-long-calls
Treat all calls as being far away (near).  If calls are assumed to be
far away, the compiler will use the @code{call} instruction to
call a function and the @code{rtc} instruction for returning.

@item -mshort
@opindex mshort
Consider type @code{int} to be 16 bits wide, like @code{short int}.

@item -msoft-reg-count=@var{count}
@opindex msoft-reg-count
Specify the number of pseudo-soft registers which are used for the
code generation.  The maximum number is 32.  Using more pseudo-soft
register may or may not result in better code depending on the program.
The default is 4 for 68HC11 and 2 for 68HC12.

@end table

@node MCore Options
@subsection MCore Options
@cindex MCore options

These are the @samp{-m} options defined for the Motorola M*Core
processors.

@table @gcctabopt

@item -mhardlit
@itemx -mno-hardlit
```

```
@opindex mhardlit
@opindex mno-hardlit
Inline constants into the code stream if it can be done in two
instructions or less.

@item -mdiv
@itemx -mno-div
@opindex mdiv
@opindex mno-div
Use the divide instruction.  (Enabled by default).

@item -mrelax-immediate
@itemx -mno-relax-immediate
@opindex mrelax-immediate
@opindex mno-relax-immediate
Allow arbitrary sized immediates in bit operations.

@item -mwide-bitfields
@itemx -mno-wide-bitfields
@opindex mwide-bitfields
@opindex mno-wide-bitfields
Always treat bit-fields as int-sized.

@item -m4byte-functions
@itemx -mno-4byte-functions
@opindex m4byte-functions
@opindex mno-4byte-functions
Force all functions to be aligned to a four byte boundary.

@item -mcallgraph-data
@itemx -mno-callgraph-data
@opindex mcallgraph-data
@opindex mno-callgraph-data
Emit callgraph information.

@item -mslow-bytes
@itemx -mno-slow-bytes
@opindex mslow-bytes
@opindex mno-slow-bytes
Prefer word access when reading byte quantities.

@item -mlittle-endian
@itemx -mbig-endian
@opindex mlittle-endian
@opindex mbig-endian
Generate code for a little endian target.

@item -m210
@itemx -m340
@opindex m210
@opindex m340
Generate code for the 210 processor.

@item -mno-lsim
@opindex no-lsim
Assume that run-time support has been provided and so omit the
simulator library (@file{libsim.a)} from the linker command line.

@item -mstack-increment=@var{size}
@opindex mstack-increment
Set the maximum amount for a single stack increment operation.  Large
values can increase the speed of programs which contain functions
that need a large amount of stack space, but they can also trigger a
segmentation fault if the stack is extended too much.  The default
value is 0x1000.
```



```
@end table

@node MIPS Options
@subsection MIPS Options
@cindex MIPS options

@table @gcctabopt

@item -EB
@opindex EB
Generate big-endian code.

@item -EL
@opindex EL
Generate little-endian code.  This is the default for @samp{mips*el-*-*}
configurations.

@item -march=@var{arch}
@opindex march
Generate code that will run on @var{arch}, which can be the name of a
generic MIPS ISA, or the name of a particular processor.
The ISA names are:
@samp{mips1}, @samp{mips2}, @samp{mips3}, @samp{mips4},
@samp{mips32}, @samp{mips32r2}, @samp{mips64} and @samp{mips64r2}.
The processor names are:
@samp{4kc}, @samp{4km}, @samp{4kp}, @samp{4ksc},
@samp{4kec}, @samp{4kem}, @samp{4kep}, @samp{4ksd},
@samp{5kc}, @samp{5kf},
@samp{20kc},
@samp{24kc}, @samp{24kf2_1}, @samp{24kf1_1},
@samp{24kec}, @samp{24kef2_1}, @samp{24kef1_1},
@samp{34kc}, @samp{34kf2_1}, @samp{34kf1_1},
@samp{74kc}, @samp{74kf2_1}, @samp{74kf1_1}, @samp{74kf3_2},
@samp{loongson2e}, @samp{loongson2f},
@samp{m4k},
@samp{octeon},
@samp{orion},
@samp{r2000}, @samp{r3000}, @samp{r3900}, @samp{r4000}, @samp{r4400},
@samp{r4600}, @samp{r4650}, @samp{r6000}, @samp{r8000},
@samp{rm7000}, @samp{rm9000},
@samp{r10000}, @samp{r12000}, @samp{r14000}, @samp{r16000},
@samp{sb1},
@samp{sr71000},
@samp{vr4100}, @samp{vr4111}, @samp{vr4120}, @samp{vr4130}, @samp{vr4300},
@samp{vr5000}, @samp{vr5400}, @samp{vr5500}
and @samp{xlr}.
The special value @samp{from-abi} selects the
most compatible architecture for the selected ABI (that is,
@samp{mips1} for 32-bit ABIs and @samp{mips3} for 64-bit ABIs)@.

Native Linux/GNU toolchains also support the value @samp{native},
which selects the best architecture option for the host processor.
@option{-march=native} has no effect if GCC does not recognize
the processor.

In processor names, a final @samp{000} can be abbreviated as @samp{k}
(for example, @samp{-march=r2k}).  Prefixes are optional, and
@samp{vr} may be written @samp{r}.

Names of the form @samp{@var{n}f2_1} refer to processors with
FPUs clocked at half the rate of the core, names of the form
@samp{@var{n}f1_1} refer to processors with FPUs clocked at the same
rate as the core, and names of the form @samp{@var{n}f3_2} refer to
processors with FPUs clocked a ratio of 3:2 with respect to the core.
For compatibility reasons, @samp{@var{n}f} is accepted as a synonym
for @samp{@var{n}f2_1} while @samp{@var{n}x} and @samp{@var{b}fx} are
```

```
accepted as synonyms for @samp{@var{n}f1_1}.

GCC defines two macros based on the value of this option.  The first
is @samp{_MIPS_ARCH}, which gives the name of target architecture, as
a string.  The second has the form @samp{_MIPS_ARCH_@var{foo}},
where @var{foo} is the capitalized value of @samp{_MIPS_ARCH}@.
For example, @samp{-march=r2000} will set @samp{_MIPS_ARCH}
to @samp{"r2000"} and define the macro @samp{_MIPS_ARCH_R2000}.

Note that the @samp{_MIPS_ARCH} macro uses the processor names given
above.  In other words, it will have the full prefix and will not
abbreviate @samp{000} as @samp{k}.  In the case of @samp{from-abi},
the macro names the resolved architecture (either @samp{"mips1"} or
@samp{"mips3"}).  It names the default architecture when no
@option{-march} option is given.

@item -mtune=@var{arch}
@opindex mtune
Optimize for @var{arch}.  Among other things, this option controls
the way instructions are scheduled, and the perceived cost of arithmetic
operations.  The list of @var{arch} values is the same as for
@option{-march}.

When this option is not used, GCC will optimize for the processor
specified by @option{-march}.  By using @option{-march} and
@option{-mtune} together, it is possible to generate code that will
run on a family of processors, but optimize the code for one
particular member of that family.

@samp{-mtune} defines the macros @samp{_MIPS_TUNE} and
@samp{_MIPS_TUNE_@var{foo}}, which work in the same way as the
@samp{-march} ones described above.

@item -mips1
@opindex mips1
Equivalent to @samp{-march=mips1}.

@item -mips2
@opindex mips2
Equivalent to @samp{-march=mips2}.

@item -mips3
@opindex mips3
Equivalent to @samp{-march=mips3}.

@item -mips4
@opindex mips4
Equivalent to @samp{-march=mips4}.

@item -mips32
@opindex mips32
Equivalent to @samp{-march=mips32}.

@item -mips32r2
@opindex mips32r2
Equivalent to @samp{-march=mips32r2}.

@item -mips64
@opindex mips64
Equivalent to @samp{-march=mips64}.

@item -mips64r2
@opindex mips64r2
Equivalent to @samp{-march=mips64r2}.

@item -mips16
```



```
@itemx -mno-mips16
@opindex mips16
@opindex mno-mips16
Generate (do not generate) MIPS16 code.  If GCC is targetting a
MIPS32 or MIPS64 architecture, it will make use of the MIPS16e ASE@.

MIPS16 code generation can also be controlled on a per-function basis
by means of @code{mips16} and @code{nomips16} attributes.
@xref{Function Attributes}, for more information.

@item -mflip-mips16
@opindex mflip-mips16
Generate MIPS16 code on alternating functions.  This option is provided
for regression testing of mixed MIPS16/non-MIPS16 code generation, and is
not intended for ordinary use in compiling user code.

@item -minterlink-mips16
@itemx -mno-interlink-mips16
@opindex minterlink-mips16
@opindex mno-interlink-mips16
Require (do not require) that non-MIPS16 code be link-compatible with
MIPS16 code.

For example, non-MIPS16 code cannot jump directly to MIPS16 code;
it must either use a call or an indirect jump.  @option{-minterlink-mips16}
therefore disables direct jumps unless GCC knows that the target of the
jump is not MIPS16.

@item -mabi=32
@itemx -mabi=o64
@itemx -mabi=n32
@itemx -mabi=64
@itemx -mabi=eabi
@opindex mabi=32
@opindex mabi=o64
@opindex mabi=n32
@opindex mabi=64
@opindex mabi=eabi
Generate code for the given ABI@.

Note that the EABI has a 32-bit and a 64-bit variant.  GCC normally
generates 64-bit code when you select a 64-bit architecture, but you
can use @option{-mgp32} to get 32-bit code instead.

For information about the O64 ABI, see
@w{@uref{http://gcc.gnu.org/projects/mipso64-abi.html}}.

GCC supports a variant of the o32 ABI in which floating-point registers
are 64 rather than 32 bits wide.  You can select this combination with
@option{-mabi=32} @option{-mfp64}.  This ABI relies on the @samp{mthc1}
and @samp{mfhc1} instructions and is therefore only supported for
MIPS32R2 processors.

The register assignments for arguments and return values remain the
same, but each scalar value is passed in a single 64-bit register
rather than a pair of 32-bit registers.  For example, scalar
floating-point values are returned in @samp{$f0} only, not a
@samp{$f0}/@samp{$f1} pair.  The set of call-saved registers also
remains the same, but all 64 bits are saved.

@item -mabicalls
@itemx -mno-abicalls
@opindex mabicalls
@opindex mno-abicalls
Generate (do not generate) code that is suitable for SVR4-style
dynamic objects.  @option{-mabicalls} is the default for SVR4-based
```

systems.

@item -mshared
@itemx -mno-shared
Generate (do not generate) code that is fully position-independent,
and that can therefore be linked into shared libraries. This option
only affects @option{-mabicalls}.

All @option{-mabicalls} code has traditionally been position-independent,
regardless of options like @option{-fPIC} and @option{-fpic}. However,
as an extension, the GNU toolchain allows executables to use absolute
accesses for locally-binding symbols. It can also use shorter GP
initialization sequences and generate direct calls to locally-defined
functions. This mode is selected by @option{-mno-shared}.

@option{-mno-shared} depends on binutils 2.16 or higher and generates
objects that can only be linked by the GNU linker. However, the option
does not affect the ABI of the final executable; it only affects the ABI
of relocatable objects. Using @option{-mno-shared} will generally make
executables both smaller and quicker.

@option{-mshared} is the default.

@item -mplt
@itemx -mno-plt
@opindex mplt
@opindex mno-plt
Assume (do not assume) that the static and dynamic linkers
support PLTs and copy relocations. This option only affects
@samp{-mno-shared -mabicalls}. For the n64 ABI, this option
has no effect without @samp{-msym32}.

You can make @option{-mplt} the default by configuring
GCC with @option{--with-mips-plt}. The default is
@option{-mno-plt} otherwise.

@item -mxgot
@itemx -mno-xgot
@opindex mxgot
@opindex mno-xgot
Lift (do not lift) the usual restrictions on the size of the global
offset table.

GCC normally uses a single instruction to load values from the GOT@.
While this is relatively efficient, it will only work if the GOT
is smaller than about 64k. Anything larger will cause the linker
to report an error such as:

@cindex relocation truncated to fit (MIPS)
@smallexample
relocation truncated to fit: R_MIPS_GOT16 foobar
@end smallexample

If this happens, you should recompile your code with @option{-mxgot}.
It should then work with very large GOTs, although it will also be
less efficient, since it will take three instructions to fetch the
value of a global symbol.

Note that some linkers can create multiple GOTs. If you have such a
linker, you should only need to use @option{-mxgot} when a single object
file accesses more than 64k's worth of GOT entries. Very few do.

These options have no effect unless GCC is generating position
independent code.

@item -mgp32

@opindex mgp32
Assume that general-purpose registers are 32 bits wide.

@item -mgp64
@opindex mgp64
Assume that general-purpose registers are 64 bits wide.

@item -mfp32
@opindex mfp32
Assume that floating-point registers are 32 bits wide.

@item -mfp64
@opindex mfp64
Assume that floating-point registers are 64 bits wide.

@item -mhard-float
@opindex mhard-float
Use floating-point coprocessor instructions.

@item -msoft-float
@opindex msoft-float
Do not use floating-point coprocessor instructions. Implement
floating-point calculations using library calls instead.

@item -msingle-float
@opindex msingle-float
Assume that the floating-point coprocessor only supports single-precision
operations.

@item -mdouble-float
@opindex mdouble-float
Assume that the floating-point coprocessor supports double-precision
operations. This is the default.

@item -mllsc
@itemx -mno-llsc
@opindex mllsc
@opindex mno-llsc
Use (do not use) @samp{ll}, @samp{sc}, and @samp{sync} instructions to
implement atomic memory built-in functions. When neither option is
specified, GCC will use the instructions if the target architecture
supports them.

@option{-mllsc} is useful if the runtime environment can emulate the
instructions and @option{-mno-llsc} can be useful when compiling for
nonstandard ISAs. You can make either option the default by
configuring GCC with @option{--with-llsc} and @option{--without-llsc}
respectively. @option{--with-llsc} is the default for some
configurations; see the installation documentation for details.

@item -mdsp
@itemx -mno-dsp
@opindex mdsp
@opindex mno-dsp
Use (do not use) revision 1 of the MIPS DSP ASE@.
@xref{MIPS DSP Built-in Functions}. This option defines the
preprocessor macro @samp{__mips_dsp}. It also defines
@samp{__mips_dsp_rev} to 1.

@item -mdspr2
@itemx -mno-dspr2
@opindex mdspr2
@opindex mno-dspr2
Use (do not use) revision 2 of the MIPS DSP ASE@.
@xref{MIPS DSP Built-in Functions}. This option defines the
preprocessor macros @samp{__mips_dsp} and @samp{__mips_dspr2}.

It also defines @samp{__mips_dsp_rev} to 2.

@item -msmartmips
@itemx -mno-smartmips
@opindex msmartmips
@opindex mno-smartmips
Use (do not use) the MIPS SmartMIPS ASE.

@item -mpaired-single
@itemx -mno-paired-single
@opindex mpaired-single
@opindex mno-paired-single
Use (do not use) paired-single floating-point instructions.
@xref{MIPS Paired-Single Support}. This option requires
hardware floating-point support to be enabled.

@item -mdmx
@itemx -mno-mdmx
@opindex mdmx
@opindex mno-mdmx
Use (do not use) MIPS Digital Media Extension instructions.
This option can only be used when generating 64-bit code and requires
hardware floating-point support to be enabled.

@item -mips3d
@itemx -mno-mips3d
@opindex mips3d
@opindex mno-mips3d
Use (do not use) the MIPS-3D ASE@. @xref{MIPS-3D Built-in Functions}.
The option @option{-mips3d} implies @option{-mpaired-single}.

@item -mmt
@itemx -mno-mt
@opindex mmt
@opindex mno-mt
Use (do not use) MT Multithreading instructions.

@item -mlong64
@opindex mlong64
Force @code{long} types to be 64 bits wide. See @option{-mlong32} for
an explanation of the default and the way that the pointer size is
determined.

@item -mlong32
@opindex mlong32
Force @code{long}, @code{int}, and pointer types to be 32 bits wide.

The default size of @code{int}s, @code{long}s and pointers depends on
the ABI@. All the supported ABIs use 32-bit @code{int}s. The n64 ABI
uses 64-bit @code{long}s, as does the 64-bit EABI; the others use
32-bit @code{long}s. Pointers are the same size as @code{long}s,
or the same size as integer registers, whichever is smaller.

@item -msym32
@itemx -mno-sym32
@opindex msym32
@opindex mno-sym32
Assume (do not assume) that all symbols have 32-bit values, regardless
of the selected ABI@. This option is useful in combination with
@option{-mabi=64} and @option{-mno-abicalls} because it allows GCC
to generate shorter and faster references to symbolic addresses.

@item -G @var{num}
@opindex G
Put definitions of externally-visible data in a small data section
if that data is no bigger than @var{num} bytes. GCC can then access

the data more efficiently; see @option{-mgpopt} for details.

The default @option{-G} option depends on the configuration.

@item -mlocal-sdata
@itemx -mno-local-sdata
@opindex mlocal-sdata
@opindex mno-local-sdata
Extend (do not extend) the @option{-G} behavior to local data too,
such as to static variables in C@. @option{-mlocal-sdata} is the
default for all configurations.

If the linker complains that an application is using too much small data,
you might want to try rebuilding the less performance-critical parts with
@option{-mno-local-sdata}. You might also want to build large
libraries with @option{-mno-local-sdata}, so that the libraries leave
more room for the main program.

@item -mextern-sdata
@itemx -mno-extern-sdata
@opindex mextern-sdata
@opindex mno-extern-sdata
Assume (do not assume) that externally-defined data will be in
a small data section if that data is within the @option{-G} limit.
@option{-mextern-sdata} is the default for all configurations.

If you compile a module @var{Mod} with @option{-mextern-sdata} @option{-G
@var{num}} @option{-mgpopt}, and @var{Mod} references a variable @var{Var}
that is no bigger than @var{num} bytes, you must make sure that @var{Var}
is placed in a small data section. If @var{Var} is defined by another
module, you must either compile that module with a high-enough
@option{-G} setting or attach a @code{section} attribute to @var{Var}'s
definition. If @var{Var} is common, you must link the application
with a high-enough @option{-G} setting.

The easiest way of satisfying these restrictions is to compile
and link every module with the same @option{-G} option. However,
you may wish to build a library that supports several different
small data limits. You can do this by compiling the library with
the highest supported @option{-G} setting and additionally using
@option{-mno-extern-sdata} to stop the library from making assumptions
about externally-defined data.

@item -mgpopt
@itemx -mno-gpopt
@opindex mgpopt
@opindex mno-gpopt
Use (do not use) GP-relative accesses for symbols that are known to be
in a small data section; see @option{-G}, @option{-mlocal-sdata} and
@option{-mextern-sdata}. @option{-mgpopt} is the default for all
configurations.

@option{-mno-gpopt} is useful for cases where the @code{$gp} register
might not hold the value of @code{_gp}. For example, if the code is
part of a library that might be used in a boot monitor, programs that
call boot monitor routines will pass an unknown value in @code{$gp}.
(In such situations, the boot monitor itself would usually be compiled
with @option{-G0}.)

@option{-mno-gpopt} implies @option{-mno-local-sdata} and
@option{-mno-extern-sdata}.

@item -membedded-data
@itemx -mno-embedded-data
@opindex membedded-data
@opindex mno-embedded-data

```
Allocate variables to the read-only data section first if possible, then
next in the small data section if possible, otherwise in data.  This gives
slightly slower code than the default, but reduces the amount of RAM required
when executing, and thus may be preferred for some embedded systems.

@item -muninit-const-in-rodata
@itemx -mno-uninit-const-in-rodata
@opindex muninit-const-in-rodata
@opindex mno-uninit-const-in-rodata
Put uninitialized @code{const} variables in the read-only data section.
This option is only meaningful in conjunction with @option{-membedded-data}.

@item -mcode-readable=@var{setting}
@opindex mcode-readable
Specify whether GCC may generate code that reads from executable sections.
There are three possible settings:

@table @gcctabopt
@item -mcode-readable=yes
Instructions may freely access executable sections.  This is the
default setting.

@item -mcode-readable=pcrel
MIPS16 PC-relative load instructions can access executable sections,
but other instructions must not do so.  This option is useful on 4KSc
and 4KSd processors when the code TLBs have the Read Inhibit bit set.
It is also useful on processors that can be configured to have a dual
instruction/data SRAM interface and that, like the M4K, automatically
redirect PC-relative loads to the instruction RAM.

@item -mcode-readable=no
Instructions must not access executable sections.  This option can be
useful on targets that are configured to have a dual instruction/data
SRAM interface but that (unlike the M4K) do not automatically redirect
PC-relative loads to the instruction RAM.
@end table

@item -msplit-addresses
@itemx -mno-split-addresses
@opindex msplit-addresses
@opindex mno-split-addresses
Enable (disable) use of the @code{%hi()} and @code{%lo()} assembler
relocation operators.  This option has been superseded by
@option{-mexplicit-relocs} but is retained for backwards compatibility.

@item -mexplicit-relocs
@itemx -mno-explicit-relocs
@opindex mexplicit-relocs
@opindex mno-explicit-relocs
Use (do not use) assembler relocation operators when dealing with symbolic
addresses.  The alternative, selected by @option{-mno-explicit-relocs},
is to use assembler macros instead.

@option{-mexplicit-relocs} is the default if GCC was configured
to use an assembler that supports relocation operators.

@item -mcheck-zero-division
@itemx -mno-check-zero-division
@opindex mcheck-zero-division
@opindex mno-check-zero-division
Trap (do not trap) on integer division by zero.

The default is @option{-mcheck-zero-division}.

@item -mdivide-traps
@itemx -mdivide-breaks
```



```
@opindex mdivide-traps
@opindex mdivide-breaks
MIPS systems check for division by zero by generating either a
conditional trap or a break instruction.  Using traps results in
smaller code, but is only supported on MIPS II and later.  Also, some
versions of the Linux kernel have a bug that prevents trap from
generating the proper signal (@code{SIGFPE}).  Use @option{-mdivide-traps} to
allow conditional traps on architectures that support them and
@option{-mdivide-breaks} to force the use of breaks.

The default is usually @option{-mdivide-traps}, but this can be
overridden at configure time using @option{--with-divide=breaks}.
Divide-by-zero checks can be completely disabled using
@option{-mno-check-zero-division}.

@item -mmemcpy
@itemx -mno-memcpy
@opindex mmemcpy
@opindex mno-memcpy
Force (do not force) the use of @code{memcpy()} for non-trivial block
moves.  The default is @option{-mno-memcpy}, which allows GCC to inline
most constant-sized copies.

@item -mlong-calls
@itemx -mno-long-calls
@opindex mlong-calls
@opindex mno-long-calls
Disable (do not disable) use of the @code{jal} instruction.  Calling
functions using @code{jal} is more efficient but requires the caller
and callee to be in the same 256 megabyte segment.

This option has no effect on abicalls code.  The default is
@option{-mno-long-calls}.

@item -mmad
@itemx -mno-mad
@opindex mmad
@opindex mno-mad
Enable (disable) use of the @code{mad}, @code{madu} and @code{mul}
instructions, as provided by the R4650 ISA@.

@item -mfused-madd
@itemx -mno-fused-madd
@opindex mfused-madd
@opindex mno-fused-madd
Enable (disable) use of the floating point multiply-accumulate
instructions, when they are available.  The default is
@option{-mfused-madd}.

When multiply-accumulate instructions are used, the intermediate
product is calculated to infinite precision and is not subject to
the FCSR Flush to Zero bit.  This may be undesirable in some
circumstances.

@item -nocpp
@opindex nocpp
Tell the MIPS assembler to not run its preprocessor over user
assembler files (with a @samp{.s} suffix) when assembling them.

@item -mfix-r4000
@itemx -mno-fix-r4000
@opindex mfix-r4000
@opindex mno-fix-r4000
Work around certain R4000 CPU errata:
@itemize @minus
@item
```

```
A double-word or a variable shift may give an incorrect result if executed
immediately after starting an integer division.
@item
A double-word or a variable shift may give an incorrect result if executed
while an integer multiplication is in progress.
@item
An integer division may give an incorrect result if started in a delay slot
of a taken branch or a jump.
@end itemize

@item -mfix-r4400
@itemx -mno-fix-r4400
@opindex mfix-r4400
@opindex mno-fix-r4400
Work around certain R4400 CPU errata:
@itemize @minus
@item
A double-word or a variable shift may give an incorrect result if executed
immediately after starting an integer division.
@end itemize

@item -mfix-r10000
@itemx -mno-fix-r10000
@opindex mfix-r10000
@opindex mno-fix-r10000
Work around certain R10000 errata:
@itemize @minus
@item
@code{ll}/@code{sc} sequences may not behave atomically on revisions
prior to 3.0.  They may deadlock on revisions 2.6 and earlier.
@end itemize

This option can only be used if the target architecture supports
branch-likely instructions.  @option{-mfix-r10000} is the default when
@option{-march=r10000} is used; @option{-mno-fix-r10000} is the default
otherwise.

@item -mfix-vr4120
@itemx -mno-fix-vr4120
@opindex mfix-vr4120
Work around certain VR4120 errata:
@itemize @minus
@item
@code{dmultu} does not always produce the correct result.
@item
@code{div} and @code{ddiv} do not always produce the correct result if one
of the operands is negative.
@end itemize
The workarounds for the division errata rely on special functions in
@file{libgcc.a}.  At present, these functions are only provided by
the @code{mips64vr*-elf} configurations.

Other VR4120 errata require a nop to be inserted between certain pairs of
instructions.  These errata are handled by the assembler, not by GCC itself.

@item -mfix-vr4130
@opindex mfix-vr4130
Work around the VR4130 @code{mflo}/@code{mfhi} errata.  The
workarounds are implemented by the assembler rather than by GCC,
although GCC will avoid using @code{mflo} and @code{mfhi} if the
VR4130 @code{macc}, @code{macchi}, @code{dmacc} and @code{dmacchi}
instructions are available instead.

@item -mfix-sb1
@itemx -mno-fix-sb1
@opindex mfix-sb1
```



```
Work around certain SB-1 CPU core errata.
(This flag currently works around the SB-1 revision 2
''F1'' and ''F2'' floating point errata.)

@item -mr10k-cache-barrier=@var{setting}
@opindex mr10k-cache-barrier
Specify whether GCC should insert cache barriers to avoid the
side-effects of speculation on R10K processors.

In common with many processors, the R10K tries to predict the outcome
of a conditional branch and speculatively executes instructions from
the ''taken'' branch.  It later aborts these instructions if the
predicted outcome was wrong.  However, on the R10K, even aborted
instructions can have side effects.

This problem only affects kernel stores and, depending on the system,
kernel loads.  As an example, a speculatively-executed store may load
the target memory into cache and mark the cache line as dirty, even if
the store itself is later aborted.  If a DMA operation writes to the
same area of memory before the ''dirty'' line is flushed, the cached
data will overwrite the DMA-ed data.  See the R10K processor manual
for a full description, including other potential problems.

One workaround is to insert cache barrier instructions before every memory
access that might be speculatively executed and that might have side
effects even if aborted.  @option{-mr10k-cache-barrier=@var{setting}}
controls GCC's implementation of this workaround.  It assumes that
aborted accesses to any byte in the following regions will not have
side effects:

@enumerate
@item
the memory occupied by the current function's stack frame;

@item
the memory occupied by an incoming stack argument;

@item
the memory occupied by an object with a link-time-constant address.
@end enumerate

It is the kernel's responsibility to ensure that speculative
accesses to these regions are indeed safe.

If the input program contains a function declaration such as:

@smallexample
void foo (void);
@end smallexample

then the implementation of @code{foo} must allow @code{j foo} and
@code{jal foo} to be executed speculatively.  GCC honors this
restriction for functions it compiles itself.  It expects non-GCC
functions (such as hand-written assembly code) to do the same.

The option has three forms:

@table @gcctabopt
@item -mr10k-cache-barrier=load-store
Insert a cache barrier before a load or store that might be
speculatively executed and that might have side effects even
if aborted.

@item -mr10k-cache-barrier=store
Insert a cache barrier before a store that might be speculatively
executed and that might have side effects even if aborted.
```

```
@item -mr10k-cache-barrier=none
Disable the insertion of cache barriers.  This is the default setting.
@end table

@item -mflush-func=@var{func}
@itemx -mno-flush-func
@opindex mflush-func
Specifies the function to call to flush the I and D caches, or to not
call any such function.  If called, the function must take the same
arguments as the common @code{_flush_func()}, that is, the address of the
memory range for which the cache is being flushed, the size of the
memory range, and the number 3 (to flush both caches).  The default
depends on the target GCC was configured for, but commonly is either
@samp{_flush_func} or @samp{__cpu_flush}.

@item mbranch-cost=@var{num}
@opindex mbranch-cost
Set the cost of branches to roughly @var{num} ``simple'' instructions.
This cost is only a heuristic and is not guaranteed to produce
consistent results across releases.  A zero cost redundantly selects
the default, which is based on the @option{-mtune} setting.

@item -mbranch-likely
@itemx -mno-branch-likely
@opindex mbranch-likely
@opindex mno-branch-likely
Enable or disable use of Branch Likely instructions, regardless of the
default for the selected architecture.  By default, Branch Likely
instructions may be generated if they are supported by the selected
architecture.  An exception is for the MIPS32 and MIPS64 architectures
and processors which implement those architectures; for those, Branch
Likely instructions will not be generated by default because the MIPS32
and MIPS64 architectures specifically deprecate their use.

@item -mfp-exceptions
@itemx -mno-fp-exceptions
@opindex mfp-exceptions
Specifies whether FP exceptions are enabled.  This affects how we schedule
FP instructions for some processors.  The default is that FP exceptions are
enabled.

For instance, on the SB-1, if FP exceptions are disabled, and we are emitting
64-bit code, then we can use both FP pipes.  Otherwise, we can only use one
FP pipe.

@item -mvr4130-align
@itemx -mno-vr4130-align
@opindex mvr4130-align
The VR4130 pipeline is two-way superscalar, but can only issue two
instructions together if the first one is 8-byte aligned.  When this
option is enabled, GCC will align pairs of instructions that it
thinks should execute in parallel.

This option only has an effect when optimizing for the VR4130.
It normally makes code faster, but at the expense of making it bigger.
It is enabled by default at optimization level @option{-O3}.
@end table

@node MMIX Options
@subsection MMIX Options
@cindex MMIX Options

These options are defined for the MMIX:

@table @gcctabopt
```



```
@item -mlibfuncs
@itemx -mno-libfuncs
@opindex mlibfuncs
@opindex mno-libfuncs
Specify that intrinsic library functions are being compiled, passing all
values in registers, no matter the size.

@item -mepsilon
@itemx -mno-epsilon
@opindex mepsilon
@opindex mno-epsilon
Generate floating-point comparison instructions that compare with respect
to the @code{rE} epsilon register.

@item -mabi=mmixware
@itemx -mabi=gnu
@opindex mabi-mmixware
@opindex mabi=gnu
Generate code that passes function parameters and return values that (in
the called function) are seen as registers @code{$0} and up, as opposed to
the GNU ABI which uses global registers @code{$231} and up.

@item -mzero-extend
@itemx -mno-zero-extend
@opindex mzero-extend
@opindex mno-zero-extend
When reading data from memory in sizes shorter than 64 bits, use (do not
use) zero-extending load instructions by default, rather than
sign-extending ones.

@item -mknuthdiv
@itemx -mno-knuthdiv
@opindex mknuthdiv
@opindex mno-knuthdiv
Make the result of a division yielding a remainder have the same sign as
the divisor.  With the default, @option{-mno-knuthdiv}, the sign of the
remainder follows the sign of the dividend.  Both methods are
arithmetically valid, the latter being almost exclusively used.

@item -mtoplevel-symbols
@itemx -mno-toplevel-symbols
@opindex mtoplevel-symbols
@opindex mno-toplevel-symbols
Prepend (do not prepend) a @samp{:} to all global symbols, so the assembly
code can be used with the @code{PREFIX} assembly directive.

@item -melf
@opindex melf
Generate an executable in the ELF format, rather than the default
@samp{mmo} format used by the @command{mmix} simulator.

@item -mbranch-predict
@itemx -mno-branch-predict
@opindex mbranch-predict
@opindex mno-branch-predict
Use (do not use) the probable-branch instructions, when static branch
prediction indicates a probable branch.

@item -mbase-addresses
@itemx -mno-base-addresses
@opindex mbase-addresses
@opindex mno-base-addresses
Generate (do not generate) code that uses @emph{base addresses}.  Using a
base address automatically generates a request (handled by the assembler
and the linker) for a constant to be set up in a global register.  The
register is used for one or more base address requests within the range 0
```

```
to 255 from the value held in the register.  The generally leads to short
and fast code, but the number of different data items that can be
addressed is limited.  This means that a program that uses lots of static
data may require @option{-mno-base-addresses}.

@item -msingle-exit
@itemx -mno-single-exit
@opindex msingle-exit
@opindex mno-single-exit
Force (do not force) generated code to have a single exit point in each
function.
@end table

@node MN10300 Options
@subsection MN10300 Options
@cindex MN10300 options

These @option{-m} options are defined for Matsushita MN10300 architectures:

@table @gcctabopt
@item -mmult-bug
@opindex mmult-bug
Generate code to avoid bugs in the multiply instructions for the MN10300
processors.  This is the default.

@item -mno-mult-bug
@opindex mno-mult-bug
Do not generate code to avoid bugs in the multiply instructions for the
MN10300 processors.

@item -mam33
@opindex mam33
Generate code which uses features specific to the AM33 processor.

@item -mno-am33
@opindex mno-am33
Do not generate code which uses features specific to the AM33 processor.  This
is the default.

@item -mreturn-pointer-on-d0
@opindex mreturn-pointer-on-d0
When generating a function which returns a pointer, return the pointer
in both @code{a0} and @code{d0}.  Otherwise, the pointer is returned
only in a0, and attempts to call such functions without a prototype
would result in errors.  Note that this option is on by default; use
@option{-mno-return-pointer-on-d0} to disable it.

@item -mno-crt0
@opindex mno-crt0
Do not link in the C run-time initialization object file.

@item -mrelax
@opindex mrelax
Indicate to the linker that it should perform a relaxation optimization pass
to shorten branches, calls and absolute memory addresses.  This option only
has an effect when used on the command line for the final link step.

This option makes symbolic debugging impossible.
@end table

@node PDP-11 Options
@subsection PDP-11 Options
@cindex PDP-11 Options

These options are defined for the PDP-11:
```



```
@table @gcctabopt
@item -mfpu
@opindex mfpu
Use hardware FPP floating point.  This is the default.  (FIS floating
point on the PDP-11/40 is not supported.)

@item -msoft-float
@opindex msoft-float
Do not use hardware floating point.

@item -mac0
@opindex mac0
Return floating-point results in ac0 (fr0 in Unix assembler syntax).

@item -mno-ac0
@opindex mno-ac0
Return floating-point results in memory.  This is the default.

@item -m40
@opindex m40
Generate code for a PDP-11/40.

@item -m45
@opindex m45
Generate code for a PDP-11/45.  This is the default.

@item -m10
@opindex m10
Generate code for a PDP-11/10.

@item -mbcopy-builtin
@opindex bcopy-builtin
Use inline @code{movmemhi} patterns for copying memory.  This is the
default.

@item -mbcopy
@opindex mbcopy
Do not use inline @code{movmemhi} patterns for copying memory.

@item -mint16
@itemx -mno-int32
@opindex mint16
@opindex mno-int32
Use 16-bit @code{int}.  This is the default.

@item -mint32
@itemx -mno-int16
@opindex mint32
@opindex mno-int16
Use 32-bit @code{int}.

@item -mfloat64
@itemx -mno-float32
@opindex mfloat64
@opindex mno-float32
Use 64-bit @code{float}.  This is the default.

@item -mfloat32
@itemx -mno-float64
@opindex mfloat32
@opindex mno-float64
Use 32-bit @code{float}.

@item -mabshi
@opindex mabshi
Use @code{abshi2} pattern.  This is the default.
```

```
@item -mno-abshi
@opindex mno-abshi
Do not use @code{abshi2} pattern.

@item -mbranch-expensive
@opindex mbranch-expensive
Pretend that branches are expensive.  This is for experimenting with
code generation only.

@item -mbranch-cheap
@opindex mbranch-cheap
Do not pretend that branches are expensive.  This is the default.

@item -msplit
@opindex msplit
Generate code for a system with split I&D@.

@item -mno-split
@opindex mno-split
Generate code for a system without split I&D@.  This is the default.

@item -munix-asm
@opindex munix-asm
Use Unix assembler syntax.  This is the default when configured for
@samp{pdp11-*-bsd}.

@item -mdec-asm
@opindex mdec-asm
Use DEC assembler syntax.  This is the default when configured for any
PDP-11 target other than @samp{pdp11-*-bsd}.
@end table

@node picoChip Options
@subsection picoChip Options
@cindex picoChip options

These @samp{-m} options are defined for picoChip implementations:

@table @gcctabopt

@item -mae=@var{ae_type}
@opindex mcpu
Set the instruction set, register set, and instruction scheduling
parameters for array element type @var{ae_type}.  Supported values
for @var{ae_type} are @samp{ANY}, @samp{MUL}, and @samp{MAC}.

@option{-mae=ANY} selects a completely generic AE type.  Code
generated with this option will run on any of the other AE types.  The
code will not be as efficient as it would be if compiled for a specific
AE type, and some types of operation (e.g., multiplication) will not
work properly on all types of AE.

@option{-mae=MUL} selects a MUL AE type.  This is the most useful AE type
for compiled code, and is the default.

@option{-mae=MAC} selects a DSP-style MAC AE.  Code compiled with this
option may suffer from poor performance of byte (char) manipulation,
since the DSP AE does not provide hardware support for byte load/stores.

@item -msymbol-as-address
Enable the compiler to directly use a symbol name as an address in a
load/store instruction, without first loading it into a
register.  Typically, the use of this option will generate larger
programs, which run faster than when the option isn't used.  However, the
results vary from program to program, so it is left as a user option,
```



```
rather than being permanently enabled.

@item -mno-inefficient-warnings
Disables warnings about the generation of inefficient code.  These
warnings can be generated, for example, when compiling code which
performs byte-level memory operations on the MAC AE type.  The MAC AE has
no hardware support for byte-level memory operations, so all byte
load/stores must be synthesized from word load/store operations.  This is
inefficient and a warning will be generated indicating to the programmer
that they should rewrite the code to avoid byte operations, or to target
an AE type which has the necessary hardware support.  This option enables
the warning to be turned off.

@end table

@node PowerPC Options
@subsection PowerPC Options
@cindex PowerPC options

These are listed under @xref{RS/6000 and PowerPC Options}.

@node RS/6000 and PowerPC Options
@subsection IBM RS/6000 and PowerPC Options
@cindex RS/6000 and PowerPC Options
@cindex IBM RS/6000 and PowerPC Options

These @samp{-m} options are defined for the IBM RS/6000 and PowerPC:
@table @gcctabopt
@item -mpower
@itemx -mno-power
@itemx -mpower2
@itemx -mno-power2
@itemx -mpowerpc
@itemx -mno-powerpc
@itemx -mpowerpc-gpopt
@itemx -mno-powerpc-gpopt
@itemx -mpowerpc-gfxopt
@itemx -mno-powerpc-gfxopt
@itemx -mpowerpc64
@itemx -mno-powerpc64
@itemx -mmfcrf
@itemx -mno-mfcrf
@itemx -mpopcntb
@itemx -mno-popcntb
@itemx -mfprnd
@itemx -mno-fprnd
@itemx -mcmpb
@itemx -mno-cmpb
@itemx -mmfpgpr
@itemx -mno-mfpgpr
@itemx -mhard-dfp
@itemx -mno-hard-dfp
@opindex mpower
@opindex mno-power
@opindex mpower2
@opindex mno-power2
@opindex mpowerpc
@opindex mno-powerpc
@opindex mpowerpc-gpopt
@opindex mno-powerpc-gpopt
@opindex mpowerpc-gfxopt
@opindex mno-powerpc-gfxopt
@opindex mpowerpc64
@opindex mno-powerpc64
@opindex mmfcrf
@opindex mno-mfcrf
```

```
@opindex mpopcntb
@opindex mno-popcntb
@opindex mfprnd
@opindex mno-fprnd
@opindex mcmpb
@opindex mno-cmpb
@opindex mmfpgpr
@opindex mno-mfpgpr
@opindex mhard-dfp
@opindex mno-hard-dfp
GCC supports two related instruction set architectures for the
RS/6000 and PowerPC@.  The @dfn{POWER} instruction set are those
instructions supported by the @samp{rios} chip set used in the original
RS/6000 systems and the @dfn{PowerPC} instruction set is the
architecture of the Freescale MPC5xx, MPC6xx, MPC8xx microprocessors, and
the IBM 4xx, 6xx, and follow-on microprocessors.

Neither architecture is a subset of the other.  However there is a
large common subset of instructions supported by both.  An MQ
register is included in processors supporting the POWER architecture.

You use these options to specify which instructions are available on the
processor you are using.  The default value of these options is
determined when configuring GCC@.  Specifying the
@option{-mcpu=@var{cpu_type}} overrides the specification of these
options.  We recommend you use the @option{-mcpu=@var{cpu_type}} option
rather than the options listed above.

The @option{-mpower} option allows GCC to generate instructions that
are found only in the POWER architecture and to use the MQ register.
Specifying @option{-mpower2} implies @option{-power} and also allows GCC
to generate instructions that are present in the POWER2 architecture but
not the original POWER architecture.

The @option{-mpowerpc} option allows GCC to generate instructions that
are found only in the 32-bit subset of the PowerPC architecture.
Specifying @option{-mpowerpc-gpopt} implies @option{-mpowerpc} and also allows
GCC to use the optional PowerPC architecture instructions in the
General Purpose group, including floating-point square root.  Specifying
@option{-mpowerpc-gfxopt} implies @option{-mpowerpc} and also allows GCC to
use the optional PowerPC architecture instructions in the Graphics
group, including floating-point select.

The @option{-mmfcrf} option allows GCC to generate the move from
condition register field instruction implemented on the POWER4
processor and other processors that support the PowerPC V2.01
architecture.
The @option{-mpopcntb} option allows GCC to generate the popcount and
double precision FP reciprocal estimate instruction implemented on the
POWER5 processor and other processors that support the PowerPC V2.02
architecture.
The @option{-mfprnd} option allows GCC to generate the FP round to
integer instructions implemented on the POWER5+ processor and other
processors that support the PowerPC V2.03 architecture.
The @option{-mcmpb} option allows GCC to generate the compare bytes
instruction implemented on the POWER6 processor and other processors
that support the PowerPC V2.05 architecture.
The @option{-mmfpgpr} option allows GCC to generate the FP move to/from
general purpose register instructions implemented on the POWER6X
processor and other processors that support the extended PowerPC V2.05
architecture.
The @option{-mhard-dfp} option allows GCC to generate the decimal floating
point instructions implemented on some POWER processors.

The @option{-mpowerpc64} option allows GCC to generate the additional
64-bit instructions that are found in the full PowerPC64 architecture
```



```
and to treat GPRs as 64-bit, doubleword quantities.  GCC defaults to
@option{-mno-powerpc64}.

If you specify both @option{-mno-power} and @option{-mno-powerpc}, GCC
will use only the instructions in the common subset of both
architectures plus some special AIX common-mode calls, and will not use
the MQ register.  Specifying both @option{-mpower} and @option{-mpowerpc}
permits GCC to use any instruction from either architecture and to
allow use of the MQ register; specify this for the Motorola MPC601.

@item -mnew-mnemonics
@itemx -mold-mnemonics
@opindex mnew-mnemonics
@opindex mold-mnemonics
Select which mnemonics to use in the generated assembler code.  With
@option{-mnew-mnemonics}, GCC uses the assembler mnemonics defined for
the PowerPC architecture.  With @option{-mold-mnemonics} it uses the
assembler mnemonics defined for the POWER architecture.  Instructions
defined in only one architecture have only one mnemonic; GCC uses that
mnemonic irrespective of which of these options is specified.

GCC defaults to the mnemonics appropriate for the architecture in
use.  Specifying @option{-mcpu=@var{cpu_type}} sometimes overrides the
value of these option.  Unless you are building a cross-compiler, you
should normally not specify either @option{-mnew-mnemonics} or
@option{-mold-mnemonics}, but should instead accept the default.

@item -mcpu=@var{cpu_type}
@opindex mcpu
Set architecture type, register usage, choice of mnemonics, and
instruction scheduling parameters for machine type @var{cpu_type}.
Supported values for @var{cpu_type} are @samp{401}, @samp{403},
@samp{405}, @samp{405fp}, @samp{440}, @samp{440fp}, @samp{464}, @samp{464fp},
@samp{505}, @samp{601}, @samp{602}, @samp{603}, @samp{603e}, @samp{604},
@samp{604e}, @samp{620}, @samp{630}, @samp{740}, @samp{7400},
@samp{7450}, @samp{750}, @samp{801}, @samp{821}, @samp{823},
@samp{860}, @samp{970}, @samp{8540}, @samp{e300c2}, @samp{e300c3},
@samp{e500mc}, @samp{ec603e}, @samp{G3}, @samp{G4}, @samp{G5},
@samp{power}, @samp{power2}, @samp{power3}, @samp{power4},
@samp{power5}, @samp{power5+}, @samp{power6}, @samp{power6x}, @samp{power7}
@samp{common}, @samp{powerpc}, @samp{powerpc64}, @samp{rios},
@samp{rios1}, @samp{rios2}, @samp{rsc}, and @samp{rs64}.

@option{-mcpu=common} selects a completely generic processor.  Code
generated under this option will run on any POWER or PowerPC processor.
GCC will use only the instructions in the common subset of both
architectures, and will not use the MQ register.  GCC assumes a generic
processor model for scheduling purposes.

@option{-mcpu=power}, @option{-mcpu=power2}, @option{-mcpu=powerpc}, and
@option{-mcpu=powerpc64} specify generic POWER, POWER2, pure 32-bit
PowerPC (i.e., not MPC601), and 64-bit PowerPC architecture machine
types, with an appropriate, generic processor model assumed for
scheduling purposes.

The other options specify a specific processor.  Code generated under
those options will run best on that processor, and may not run at all on
others.

The @option{-mcpu} options automatically enable or disable the
following options:

@gccoptlist{-maltivec  -mfprnd  -mhard-float  -mmfcrf  -mmultiple @gol
-mnew-mnemonics  -mpopcntb  -mpower  -mpower2  -mpowerpc64 @gol
-mpowerpc-gpopt  -mpowerpc-gfxopt  -msingle-float -mdouble-float @gol
-msimple-fpu -mstring  -mmulhw  -mdlmzb  -mmfpgpr}
```

The particular options set for any particular CPU will vary between
compiler versions, depending on what setting seems to produce optimal
code for that CPU; it doesn't necessarily reflect the actual hardware's
capabilities. If you wish to set an individual option to a particular
value, you may specify it after the @option{-mcpu} option, like
@samp{-mcpu=970 -mno-altivec}.

On AIX, the @option{-maltivec} and @option{-mpowerpc64} options are
not enabled or disabled by the @option{-mcpu} option at present because
AIX does not have full support for these options. You may still
enable or disable them individually if you're sure it'll work in your
environment.

@item -mtune=@var{cpu_type}
@opindex mtune
Set the instruction scheduling parameters for machine type
@var{cpu_type}, but do not set the architecture type, register usage, or
choice of mnemonics, as @option{-mcpu=@var{cpu_type}} would. The same
values for @var{cpu_type} are used for @option{-mtune} as for
@option{-mcpu}. If both are specified, the code generated will use the
architecture, registers, and mnemonics set by @option{-mcpu}, but the
scheduling parameters set by @option{-mtune}.

@item -mswdiv
@itemx -mno-swdiv
@opindex mswdiv
@opindex mno-swdiv
Generate code to compute division as reciprocal estimate and iterative
refinement, creating opportunities for increased throughput. This
feature requires: optional PowerPC Graphics instruction set for single
precision and FRE instruction for double precision, assuming divides
cannot generate user-visible traps, and the domain values not include
Infinities, denormals or zero denominator.

@item -maltivec
@itemx -mno-altivec
@opindex maltivec
@opindex mno-altivec
Generate code that uses (does not use) AltiVec instructions, and also
enable the use of built-in functions that allow more direct access to
the AltiVec instruction set. You may also need to set
@option{-mabi=altivec} to adjust the current ABI with AltiVec ABI
enhancements.

@item -mvrsave
@itemx -mno-vrsave
@opindex mvrsave
@opindex mno-vrsave
Generate VRSAVE instructions when generating AltiVec code.

@item -mgen-cell-microcode
@opindex mgen-cell-microcode
Generate Cell microcode instructions

@item -mwarn-cell-microcode
@opindex mwarn-cell-microcode
Warning when a Cell microcode instruction is going to emitted. An example
of a Cell microcode instruction is a variable shift.

@item -msecure-plt
@opindex msecure-plt
Generate code that allows ld and ld.so to build executables and shared
libraries with non-exec .plt and .got sections. This is a PowerPC
32-bit SYSV ABI option.

@item -mbss-plt
@opindex mbss-plt
Generate code that uses a BSS .plt section that ld.so fills in, and
requires .plt and .got sections that are both writable and executable.
This is a PowerPC 32-bit SYSV ABI option.

@item -misel
@itemx -mno-isel
@opindex misel
@opindex mno-isel
This switch enables or disables the generation of ISEL instructions.

@item -misel=@var{yes/no}
This switch has been deprecated. Use @option{-misel} and
@option{-mno-isel} instead.

@item -mspe
@itemx -mno-spe
@opindex mspe
@opindex mno-spe
This switch enables or disables the generation of SPE simd
instructions.

@item -mpaired
@itemx -mno-paired
@opindex mpaired
@opindex mno-paired
This switch enables or disables the generation of PAIRED simd
instructions.

@item -mspe=@var{yes/no}
This option has been deprecated. Use @option{-mspe} and
@option{-mno-spe} instead.

@item -mfloat-gprs=@var{yes/single/double/no}
@itemx -mfloat-gprs
@opindex mfloat-gprs
This switch enables or disables the generation of floating point
operations on the general purpose registers for architectures that
support it.

The argument @var{yes} or @var{single} enables the use of
single-precision floating point operations.

The argument @var{double} enables the use of single and
double-precision floating point operations.

The argument @var{no} disables floating point operations on the
general purpose registers.

This option is currently only available on the MPC854x.

@item -m32
@itemx -m64
@opindex m32
@opindex m64
Generate code for 32-bit or 64-bit environments of Darwin and SVR4
targets (including GNU/Linux). The 32-bit environment sets int, long
and pointer to 32 bits and generates code that runs on any PowerPC
variant. The 64-bit environment sets int to 32 bits and long and
pointer to 64 bits, and generates code for PowerPC64, as for
@option{-mpowerpc64}.

@item -mfull-toc
@itemx -mno-fp-in-toc
@itemx -mno-sum-in-toc

```
@itemx -mminimal-toc
@opindex mfull-toc
@opindex mno-fp-in-toc
@opindex mno-sum-in-toc
@opindex mminimal-toc
Modify generation of the TOC (Table Of Contents), which is created for
every executable file.  The @option{-mfull-toc} option is selected by
default.  In that case, GCC will allocate at least one TOC entry for
each unique non-automatic variable reference in your program.  GCC
will also place floating-point constants in the TOC@.  However, only
16,384 entries are available in the TOC@.

If you receive a linker error message that saying you have overflowed
the available TOC space, you can reduce the amount of TOC space used
with the @option{-mno-fp-in-toc} and @option{-mno-sum-in-toc} options.
@option{-mno-fp-in-toc} prevents GCC from putting floating-point
constants in the TOC and @option{-mno-sum-in-toc} forces GCC to
generate code to calculate the sum of an address and a constant at
run-time instead of putting that sum into the TOC@.  You may specify one
or both of these options.  Each causes GCC to produce very slightly
slower and larger code at the expense of conserving TOC space.

If you still run out of space in the TOC even when you specify both of
these options, specify @option{-mminimal-toc} instead.  This option causes
GCC to make only one TOC entry for every file.  When you specify this
option, GCC will produce code that is slower and larger but which
uses extremely little TOC space.  You may wish to use this option
only on files that contain less frequently executed code.

@item -maix64
@itemx -maix32
@opindex maix64
@opindex maix32
Enable 64-bit AIX ABI and calling convention: 64-bit pointers, 64-bit
@code{long} type, and the infrastructure needed to support them.
Specifying @option{-maix64} implies @option{-mpowerpc64} and
@option{-mpowerpc}, while @option{-maix32} disables the 64-bit ABI and
implies @option{-mno-powerpc64}.  GCC defaults to @option{-maix32}.

@item -mxl-compat
@itemx -mno-xl-compat
@opindex mxl-compat
@opindex mno-xl-compat
Produce code that conforms more closely to IBM XL compiler semantics
when using AIX-compatible ABI@.  Pass floating-point arguments to
prototyped functions beyond the register save area (RSA) on the stack
in addition to argument FPRs.  Do not assume that most significant
double in 128-bit long double value is properly rounded when comparing
values and converting to double.  Use XL symbol names for long double
support routines.

The AIX calling convention was extended but not initially documented to
handle an obscure K&R C case of calling a function that takes the
address of its arguments with fewer arguments than declared.  IBM XL
compilers access floating point arguments which do not fit in the
RSA from the stack when a subroutine is compiled without
optimization.  Because always storing floating-point arguments on the
stack is inefficient and rarely needed, this option is not enabled by
default and only is necessary when calling subroutines compiled by IBM
XL compilers without optimization.

@item -mpe
@opindex mpe
Support @dfn{IBM RS/6000 SP} @dfn{Parallel Environment} (PE)@.  Link an
application written to use message passing with special startup code to
enable the application to run.  The system must have PE installed in the
```



```
standard location (@file{/usr/lpp/ppe.poe/}), or the @file{specs} file
must be overridden with the @option{-specs=} option to specify the
appropriate directory location.  The Parallel Environment does not
support threads, so the @option{-mpe} option and the @option{-pthread}
option are incompatible.

@item -malign-natural
@itemx -malign-power
@opindex malign-natural
@opindex malign-power
On AIX, 32-bit Darwin, and 64-bit PowerPC GNU/Linux, the option
@option{-malign-natural} overrides the ABI-defined alignment of larger
types, such as floating-point doubles, on their natural size-based boundary.
The option @option{-malign-power} instructs GCC to follow the ABI-specified
alignment rules.  GCC defaults to the standard alignment defined in the ABI@.

On 64-bit Darwin, natural alignment is the default, and @option{-malign-power}
is not supported.

@item -msoft-float
@itemx -mhard-float
@opindex msoft-float
@opindex mhard-float
Generate code that does not use (uses) the floating-point register set.
Software floating point emulation is provided if you use the
@option{-msoft-float} option, and pass the option to GCC when linking.

@item -msingle-float
@itemx -mdouble-float
@opindex msingle-float
@opindex mdouble-float
Generate code for single or double-precision floating point operations.
@option{-mdouble-float} implies @option{-msingle-float}.

@item -msimple-fpu
@opindex msimple-fpu
Do not generate sqrt and div instructions for hardware floating point unit.

@item -mfpu
@opindex mfpu
Specify type of floating point unit.  Valid values are @var{sp_lite}
(equivalent to -msingle-float -msimple-fpu), @var{dp_lite} (equivalent
to -mdouble-float -msimple-fpu), @var{sp_full} (equivalent to -msingle-float),
and @var{dp_full} (equivalent to -mdouble-float).

@item -mxilinx-fpu
@opindex mxilinx-fpu
Perform optimizations for floating point unit on Xilinx PPC 405/440.

@item -mmultiple
@itemx -mno-multiple
@opindex mmultiple
@opindex mno-multiple
Generate code that uses (does not use) the load multiple word
instructions and the store multiple word instructions.  These
instructions are generated by default on POWER systems, and not
generated on PowerPC systems.  Do not use @option{-mmultiple} on little
endian PowerPC systems, since those instructions do not work when the
processor is in little endian mode.  The exceptions are PPC740 and
PPC750 which permit the instructions usage in little endian mode.

@item -mstring
@itemx -mno-string
@opindex mstring
@opindex mno-string
Generate code that uses (does not use) the load string instructions
```

and the store string word instructions to save multiple registers and
do small block moves. These instructions are generated by default on
POWER systems, and not generated on PowerPC systems. Do not use
@option{-mstring} on little endian PowerPC systems, since those
instructions do not work when the processor is in little endian mode.
The exceptions are PPC740 and PPC750 which permit the instructions
usage in little endian mode.

@item -mupdate
@itemx -mno-update
@opindex mupdate
@opindex mno-update
Generate code that uses (does not use) the load or store instructions
that update the base register to the address of the calculated memory
location. These instructions are generated by default. If you use
@option{-mno-update}, there is a small window between the time that the
stack pointer is updated and the address of the previous frame is
stored, which means code that walks the stack frame across interrupts or
signals may get corrupted data.

@item -mavoid-indexed-addresses
@item -mno-avoid-indexed-addresses
@opindex mavoid-indexed-addresses
@opindex mno-avoid-indexed-addresses
Generate code that tries to avoid (not avoid) the use of indexed load
or store instructions. These instructions can incur a performance
penalty on Power6 processors in certain situations, such as when
stepping through large arrays that cross a 16M boundary. This option
is enabled by default when targetting Power6 and disabled otherwise.

@item -mfused-madd
@itemx -mno-fused-madd
@opindex mfused-madd
@opindex mno-fused-madd
Generate code that uses (does not use) the floating point multiply and
accumulate instructions. These instructions are generated by default if
hardware floating is used.

@item -mmulhw
@itemx -mno-mulhw
@opindex mmulhw
@opindex mno-mulhw
Generate code that uses (does not use) the half-word multiply and
multiply-accumulate instructions on the IBM 405, 440 and 464 processors.
These instructions are generated by default when targetting those
processors.

@item -mdlmzb
@itemx -mno-dlmzb
@opindex mdlmzb
@opindex mno-dlmzb
Generate code that uses (does not use) the string-search @samp{dlmzb}
instruction on the IBM 405, 440 and 464 processors. This instruction is
generated by default when targetting those processors.

@item -mno-bit-align
@itemx -mbit-align
@opindex mno-bit-align
@opindex mbit-align
On System V.4 and embedded PowerPC systems do not (do) force structures
and unions that contain bit-fields to be aligned to the base type of the
bit-field.

For example, by default a structure containing nothing but 8
@code{unsigned} bit-fields of length 1 would be aligned to a 4 byte
boundary and have a size of 4 bytes. By using @option{-mno-bit-align},

the structure would be aligned to a 1 byte boundary and be one byte in
size.

@item -mno-strict-align
@itemx -mstrict-align
@opindex mno-strict-align
@opindex mstrict-align
On System V.4 and embedded PowerPC systems do not (do) assume that
unaligned memory references will be handled by the system.

@item -mrelocatable
@itemx -mno-relocatable
@opindex mrelocatable
@opindex mno-relocatable
On embedded PowerPC systems generate code that allows (does not allow)
the program to be relocated to a different address at runtime. If you
use @option{-mrelocatable} on any module, all objects linked together must
be compiled with @option{-mrelocatable} or @option{-mrelocatable-lib}.

@item -mrelocatable-lib
@itemx -mno-relocatable-lib
@opindex mrelocatable-lib
@opindex mno-relocatable-lib
On embedded PowerPC systems generate code that allows (does not allow)
the program to be relocated to a different address at runtime. Modules
compiled with @option{-mrelocatable-lib} can be linked with either modules
compiled without @option{-mrelocatable} and @option{-mrelocatable-lib} or
with modules compiled with the @option{-mrelocatable} options.

@item -mno-toc
@itemx -mtoc
@opindex mno-toc
@opindex mtoc
On System V.4 and embedded PowerPC systems do not (do) assume that
register 2 contains a pointer to a global area pointing to the addresses
used in the program.

@item -mlittle
@itemx -mlittle-endian
@opindex mlittle
@opindex mlittle-endian
On System V.4 and embedded PowerPC systems compile code for the
processor in little endian mode. The @option{-mlittle-endian} option is
the same as @option{-mlittle}.

@item -mbig
@itemx -mbig-endian
@opindex mbig
@opindex mbig-endian
On System V.4 and embedded PowerPC systems compile code for the
processor in big endian mode. The @option{-mbig-endian} option is
the same as @option{-mbig}.

@item -mdynamic-no-pic
@opindex mdynamic-no-pic
On Darwin and Mac OS X systems, compile code so that it is not
relocatable, but that its external references are relocatable. The
resulting code is suitable for applications, but not shared
libraries.

@item -mprioritize-restricted-insns=@var{priority}
@opindex mprioritize-restricted-insns
This option controls the priority that is assigned to
dispatch-slot restricted instructions during the second scheduling
pass. The argument @var{priority} takes the value @var{0/1/2} to assign
@var{no/highest/second-highest} priority to dispatch slot restricted

```
instructions.

@item -msched-costly-dep=@var{dependence_type}
@opindex msched-costly-dep
This option controls which dependences are considered costly
by the target during instruction scheduling.  The argument
@var{dependence_type} takes one of the following values:
@var{no}: no dependence is costly,
@var{all}: all dependences are costly,
@var{true_store_to_load}: a true dependence from store to load is costly,
@var{store_to_load}: any dependence from store to load is costly,
@var{number}: any dependence which latency >= @var{number} is costly.

@item -minsert-sched-nops=@var{scheme}
@opindex minsert-sched-nops
This option controls which nop insertion scheme will be used during
the second scheduling pass.  The argument @var{scheme} takes one of the
following values:
@var{no}: Don't insert nops.
@var{pad}: Pad with nops any dispatch group which has vacant issue slots,
according to the scheduler's grouping.
@var{regroup_exact}: Insert nops to force costly dependent insns into
separate groups.  Insert exactly as many nops as needed to force an insn
to a new group, according to the estimated processor grouping.
@var{number}: Insert nops to force costly dependent insns into
separate groups.  Insert @var{number} nops to force an insn to a new group.

@item -mcall-sysv
@opindex mcall-sysv
On System V.4 and embedded PowerPC systems compile code using calling
conventions that adheres to the March 1995 draft of the System V
Application Binary Interface, PowerPC processor supplement.  This is the
default unless you configured GCC using @samp{powerpc-*-eabiaix}.

@item -mcall-sysv-eabi
@opindex mcall-sysv-eabi
Specify both @option{-mcall-sysv} and @option{-meabi} options.

@item -mcall-sysv-noeabi
@opindex mcall-sysv-noeabi
Specify both @option{-mcall-sysv} and @option{-mno-eabi} options.

@item -mcall-solaris
@opindex mcall-solaris
On System V.4 and embedded PowerPC systems compile code for the Solaris
operating system.

@item -mcall-linux
@opindex mcall-linux
On System V.4 and embedded PowerPC systems compile code for the
Linux-based GNU system.

@item -mcall-gnu
@opindex mcall-gnu
On System V.4 and embedded PowerPC systems compile code for the
Hurd-based GNU system.

@item -mcall-netbsd
@opindex mcall-netbsd
On System V.4 and embedded PowerPC systems compile code for the
NetBSD operating system.

@item -maix-struct-return
@opindex maix-struct-return
Return all structures in memory (as specified by the AIX ABI)@.
```



```
@item -msvr4-struct-return
@opindex msvr4-struct-return
Return structures smaller than 8 bytes in registers (as specified by the
SVR4 ABI)@.

@item -mabi=@var{abi-type}
@opindex mabi
Extend the current ABI with a particular extension, or remove such extension.
Valid values are @var{altivec}, @var{no-altivec}, @var{spe},
@var{no-spe}, @var{ibmlongdouble}, @var{ieeelongdouble}@.

@item -mabi=spe
@opindex mabi=spe
Extend the current ABI with SPE ABI extensions.  This does not change
the default ABI, instead it adds the SPE ABI extensions to the current
ABI@.

@item -mabi=no-spe
@opindex mabi=no-spe
Disable Booke SPE ABI extensions for the current ABI@.

@item -mabi=ibmlongdouble
@opindex mabi=ibmlongdouble
Change the current ABI to use IBM extended precision long double.
This is a PowerPC 32-bit SYSV ABI option.

@item -mabi=ieeelongdouble
@opindex mabi=ieeelongdouble
Change the current ABI to use IEEE extended precision long double.
This is a PowerPC 32-bit Linux ABI option.

@item -mprototype
@itemx -mno-prototype
@opindex mprototype
@opindex mno-prototype
On System V.4 and embedded PowerPC systems assume that all calls to
variable argument functions are properly prototyped.  Otherwise, the
compiler must insert an instruction before every non prototyped call to
set or clear bit 6 of the condition code register (@var{CR}) to
indicate whether floating point values were passed in the floating point
registers in case the function takes a variable arguments.  With
@option{-mprototype}, only calls to prototyped variable argument functions
will set or clear the bit.

@item -msim
@opindex msim
On embedded PowerPC systems, assume that the startup module is called
@file{sim-crt0.o} and that the standard C libraries are @file{libsim.a} and
@file{libc.a}.  This is the default for @samp{powerpc-*-eabisim}
configurations.

@item -mmvme
@opindex mmvme
On embedded PowerPC systems, assume that the startup module is called
@file{crt0.o} and the standard C libraries are @file{libmvme.a} and
@file{libc.a}.

@item -mads
@opindex mads
On embedded PowerPC systems, assume that the startup module is called
@file{crt0.o} and the standard C libraries are @file{libads.a} and
@file{libc.a}.

@item -myellowknife
@opindex myellowknife
On embedded PowerPC systems, assume that the startup module is called
```

```
@file{crt0.o} and the standard C libraries are @file{libyk.a} and
@file{libc.a}.

@item -mvxworks
@opindex mvxworks
On System V.4 and embedded PowerPC systems, specify that you are
compiling for a VxWorks system.

@item -memb
@opindex memb
On embedded PowerPC systems, set the @var{PPC_EMB} bit in the ELF flags
header to indicate that @samp{eabi} extended relocations are used.

@item -meabi
@itemx -mno-eabi
@opindex meabi
@opindex mno-eabi
On System V.4 and embedded PowerPC systems do (do not) adhere to the
Embedded Applications Binary Interface (eabi) which is a set of
modifications to the System V.4 specifications.  Selecting @option{-meabi}
means that the stack is aligned to an 8 byte boundary, a function
@code{__eabi} is called to from @code{main} to set up the eabi
environment, and the @option{-msdata} option can use both @code{r2} and
@code{r13} to point to two separate small data areas.  Selecting
@option{-mno-eabi} means that the stack is aligned to a 16 byte boundary,
do not call an initialization function from @code{main}, and the
@option{-msdata} option will only use @code{r13} to point to a single
small data area.  The @option{-meabi} option is on by default if you
configured GCC using one of the @samp{powerpc*-*-eabi*} options.

@item -msdata=eabi
@opindex msdata=eabi
On System V.4 and embedded PowerPC systems, put small initialized
@code{const} global and static data in the @samp{.sdata2} section, which
is pointed to by register @code{r2}.  Put small initialized
non-@code{const} global and static data in the @samp{.sdata} section,
which is pointed to by register @code{r13}.  Put small uninitialized
global and static data in the @samp{.sbss} section, which is adjacent to
the @samp{.sdata} section.  The @option{-msdata=eabi} option is
incompatible with the @option{-mrelocatable} option.  The
@option{-msdata=eabi} option also sets the @option{-memb} option.

@item -msdata=sysv
@opindex msdata=sysv
On System V.4 and embedded PowerPC systems, put small global and static
data in the @samp{.sdata} section, which is pointed to by register
@code{r13}.  Put small uninitialized global and static data in the
@samp{.sbss} section, which is adjacent to the @samp{.sdata} section.
The @option{-msdata=sysv} option is incompatible with the
@option{-mrelocatable} option.

@item -msdata=default
@itemx -msdata
@opindex msdata=default
@opindex msdata
On System V.4 and embedded PowerPC systems, if @option{-meabi} is used,
compile code the same as @option{-msdata=eabi}, otherwise compile code the
same as @option{-msdata=sysv}.

@item -msdata=data
@opindex msdata=data
On System V.4 and embedded PowerPC systems, put small global
data in the @samp{.sdata} section.  Put small uninitialized global
data in the @samp{.sbss} section.  Do not use register @code{r13}
to address small data however.  This is the default behavior unless
other @option{-msdata} options are used.
```



```
@item -msdata=none
@itemx -mno-sdata
@opindex msdata=none
@opindex mno-sdata
On embedded PowerPC systems, put all initialized global and static data
in the @samp{.data} section, and all uninitialized data in the
@samp{.bss} section.

@item -G @var{num}
@opindex G
@cindex smaller data references (PowerPC)
@cindex .sdata/.sdata2 references (PowerPC)
On embedded PowerPC systems, put global and static items less than or
equal to @var{num} bytes into the small data or bss sections instead of
the normal data or bss section.  By default, @var{num} is 8.  The
@option{-G @var{num}} switch is also passed to the linker.
All modules should be compiled with the same @option{-G @var{num}} value.

@item -mregnames
@itemx -mno-regnames
@opindex mregnames
@opindex mno-regnames
On System V.4 and embedded PowerPC systems do (do not) emit register
names in the assembly language output using symbolic forms.

@item -mlongcall
@itemx -mno-longcall
@opindex mlongcall
@opindex mno-longcall
By default assume that all calls are far away so that a longer more
expensive calling sequence is required.  This is required for calls
further than 32 megabytes (33,554,432 bytes) from the current location.
A short call will be generated if the compiler knows
the call cannot be that far away.  This setting can be overridden by
the @code{shortcall} function attribute, or by @code{#pragma
longcall(0)}.

Some linkers are capable of detecting out-of-range calls and generating
glue code on the fly.  On these systems, long calls are unnecessary and
generate slower code.  As of this writing, the AIX linker can do this,
as can the GNU linker for PowerPC/64.  It is planned to add this feature
to the GNU linker for 32-bit PowerPC systems as well.

On Darwin/PPC systems, @code{#pragma longcall} will generate ``jbsr
callee, L42'', plus a ``branch island'' (glue code).  The two target
addresses represent the callee and the ``branch island''.  The
Darwin/PPC linker will prefer the first address and generate a ``bl
callee'' if the PPC ``bl'' instruction will reach the callee directly;
otherwise, the linker will generate ``bl L42'' to call the ``branch
island''.  The ``branch island'' is appended to the body of the
calling function; it computes the full 32-bit address of the callee
and jumps to it.

On Mach-O (Darwin) systems, this option directs the compiler emit to
the glue for every direct call, and the Darwin linker decides whether
to use or discard it.

In the future, we may cause GCC to ignore all longcall specifications
when the linker is known to generate glue.

@item -pthread
@opindex pthread
Adds support for multithreading with the @dfn{pthreads} library.
This option sets flags for both the preprocessor and linker.
```

```
@end table

@node S/390 and zSeries Options
@subsection S/390 and zSeries Options
@cindex S/390 and zSeries Options

These are the @samp{-m} options defined for the S/390 and zSeries architecture.

@table @gcctabopt
@item -mhard-float
@itemx -msoft-float
@opindex mhard-float
@opindex msoft-float
Use (do not use) the hardware floating-point instructions and registers
for floating-point operations.  When @option{-msoft-float} is specified,
functions in @file{libgcc.a} will be used to perform floating-point
operations.  When @option{-mhard-float} is specified, the compiler
generates IEEE floating-point instructions.  This is the default.

@item -mhard-dfp
@itemx -mno-hard-dfp
@opindex mhard-dfp
@opindex mno-hard-dfp
Use (do not use) the hardware decimal-floating-point instructions for
decimal-floating-point operations.  When @option{-mno-hard-dfp} is
specified, functions in @file{libgcc.a} will be used to perform
decimal-floating-point operations.  When @option{-mhard-dfp} is
specified, the compiler generates decimal-floating-point hardware
instructions.  This is the default for @option{-march=z9-ec} or higher.

@item -mlong-double-64
@itemx -mlong-double-128
@opindex mlong-double-64
@opindex mlong-double-128
These switches control the size of @code{long double} type. A size
of 64bit makes the @code{long double} type equivalent to the @code{double}
type. This is the default.

@item -mbackchain
@itemx -mno-backchain
@opindex mbackchain
@opindex mno-backchain
Store (do not store) the address of the caller's frame as backchain pointer
into the callee's stack frame.
A backchain may be needed to allow debugging using tools that do not understand
DWARF-2 call frame information.
When @option{-mno-packed-stack} is in effect, the backchain pointer is stored
at the bottom of the stack frame; when @option{-mpacked-stack} is in effect,
the backchain is placed into the topmost word of the 96/160 byte register
save area.

In general, code compiled with @option{-mbackchain} is call-compatible with
code compiled with @option{-mmo-backchain}; however, use of the backchain
for debugging purposes usually requires that the whole binary is built with
@option{-mbackchain}.  Note that the combination of @option{-mbackchain},
@option{-mpacked-stack} and @option{-mhard-float} is not supported.  In order
to build a linux kernel use @option{-msoft-float}.

The default is to not maintain the backchain.

@item -mpacked-stack
@itemx -mno-packed-stack
@opindex mpacked-stack
@opindex mno-packed-stack
Use (do not use) the packed stack layout.  When @option{-mno-packed-stack} is
specified, the compiler uses the all fields of the 96/160 byte register save
```



```
area only for their default purpose; unused fields still take up stack space.
When @option{-mpacked-stack} is specified, register save slots are densely
packed at the top of the register save area; unused space is reused for other
purposes, allowing for more efficient use of the available stack space.
However, when @option{-mbackchain} is also in effect, the topmost word of
the save area is always used to store the backchain, and the return address
register is always saved two words below the backchain.

As long as the stack frame backchain is not used, code generated with
@option{-mpacked-stack} is call-compatible with code generated with
@option{-mno-packed-stack}.  Note that some non-FSF releases of GCC 2.95 for
S/390 or zSeries generated code that uses the stack frame backchain at run
time, not just for debugging purposes.  Such code is not call-compatible
with code compiled with @option{-mpacked-stack}.  Also, note that the
combination of @option{-mbackchain},
@option{-mpacked-stack} and @option{-mhard-float} is not supported.  In order
to build a linux kernel use @option{-msoft-float}.

The default is to not use the packed stack layout.

@item -msmall-exec
@itemx -mno-small-exec
@opindex msmall-exec
@opindex mno-small-exec
Generate (or do not generate) code using the @code{bras} instruction
to do subroutine calls.
This only works reliably if the total executable size does not
exceed 64k.  The default is to use the @code{basr} instruction instead,
which does not have this limitation.

@item -m64
@itemx -m31
@opindex m64
@opindex m31
When @option{-m31} is specified, generate code compliant to the
GNU/Linux for S/390 ABI@.  When @option{-m64} is specified, generate
code compliant to the GNU/Linux for zSeries ABI@.  This allows GCC in
particular to generate 64-bit instructions.  For the @samp{s390}
targets, the default is @option{-m31}, while the @samp{s390x}
targets default to @option{-m64}.

@item -mzarch
@itemx -mesa
@opindex mzarch
@opindex mesa
When @option{-mzarch} is specified, generate code using the
instructions available on z/Architecture.
When @option{-mesa} is specified, generate code using the
instructions available on ESA/390.  Note that @option{-mesa} is
not possible with @option{-m64}.
When generating code compliant to the GNU/Linux for S/390 ABI,
the default is @option{-mesa}.  When generating code compliant
to the GNU/Linux for zSeries ABI, the default is @option{-mzarch}.

@item -mmvcle
@itemx -mno-mvcle
@opindex mmvcle
@opindex mno-mvcle
Generate (or do not generate) code using the @code{mvcle} instruction
to perform block moves.  When @option{-mno-mvcle} is specified,
use a @code{mvc} loop instead.  This is the default unless optimizing for
size.

@item -mdebug
@itemx -mno-debug
@opindex mdebug
```

```
@opindex mno-debug
Print (or do not print) additional debug information when compiling.
The default is to not print debug information.

@item -march=@var{cpu-type}
@opindex march
Generate code that will run on @var{cpu-type}, which is the name of a system
representing a certain processor type.  Possible values for
@var{cpu-type} are @samp{g5}, @samp{g6}, @samp{z900}, @samp{z990},
@samp{z9-109}, @samp{z9-ec} and @samp{z10}.
When generating code using the instructions available on z/Architecture,
the default is @option{-march=z900}.  Otherwise, the default is
@option{-march=g5}.

@item -mtune=@var{cpu-type}
@opindex mtune
Tune to @var{cpu-type} everything applicable about the generated code,
except for the ABI and the set of available instructions.
The list of @var{cpu-type} values is the same as for @option{-march}.
The default is the value used for @option{-march}.

@item -mtpf-trace
@itemx -mno-tpf-trace
@opindex mtpf-trace
@opindex mno-tpf-trace
Generate code that adds (does not add) in TPF OS specific branches to trace
routines in the operating system.  This option is off by default, even
when compiling for the TPF OS@.

@item -mfused-madd
@itemx -mno-fused-madd
@opindex mfused-madd
@opindex mno-fused-madd
Generate code that uses (does not use) the floating point multiply and
accumulate instructions.  These instructions are generated by default if
hardware floating point is used.

@item -mwarn-framesize=@var{framesize}
@opindex mwarn-framesize
Emit a warning if the current function exceeds the given frame size.  Because
this is a compile time check it doesn't need to be a real problem when the progr
runs.  It is intended to identify functions which most probably cause
a stack overflow.  It is useful to be used in an environment with limited stack
size e.g.@: the linux kernel.

@item -mwarn-dynamicstack
@opindex mwarn-dynamicstack
Emit a warning if the function calls alloca or uses dynamically
sized arrays.  This is generally a bad idea with a limited stack size.

@item -mstack-guard=@var{stack-guard}
@itemx -mstack-size=@var{stack-size}
@opindex mstack-guard
@opindex mstack-size
If these options are provided the s390 back end emits additional instructions in
the function prologue which trigger a trap if the stack size is @var{stack-guard
bytes above the @var{stack-size} (remember that the stack on s390 grows downward
If the @var{stack-guard} option is omitted the smallest power of 2 larger than
the frame size of the compiled function is chosen.
These options are intended to be used to help debugging stack overflow problems.
The additionally emitted code causes only little overhead and hence can also be
used in production like systems without greater performance degradation.  The gi
values have to be exact powers of 2 and @var{stack-size} has to be greater than
@var{stack-guard} without exceeding 64k.
In order to be efficient the extra code makes the assumption that the stack star
at an address aligned to the value given by @var{stack-size}.
```



```
The @var{stack-guard} option can only be used in conjunction with @var{stack-siz
@end table

@node Score Options
@subsection Score Options
@cindex Score Options

These options are defined for Score implementations:

@table @gcctabopt
@item -meb
@opindex meb
Compile code for big endian mode.  This is the default.

@item -mel
@opindex mel
Compile code for little endian mode.

@item -mnhwloop
@opindex mnhwloop
Disable generate bcnz instruction.

@item -muls
@opindex muls
Enable generate unaligned load and store instruction.

@item -mmac
@opindex mmac
Enable the use of multiply-accumulate instructions. Disabled by default.

@item -mscore5
@opindex mscore5
Specify the SCORE5 as the target architecture.

@item -mscore5u
@opindex mscore5u
Specify the SCORE5U of the target architecture.

@item -mscore7
@opindex mscore7
Specify the SCORE7 as the target architecture. This is the default.

@item -mscore7d
@opindex mscore7d
Specify the SCORE7D as the target architecture.
@end table

@node SH Options
@subsection SH Options

These @samp{-m} options are defined for the SH implementations:

@table @gcctabopt
@item -m1
@opindex m1
Generate code for the SH1.

@item -m2
@opindex m2
Generate code for the SH2.

@item -m2e
Generate code for the SH2e.

@item -m3
@opindex m3
```

```
Generate code for the SH3.

@item -m3e
@opindex m3e
Generate code for the SH3e.

@item -m4-nofpu
@opindex m4-nofpu
Generate code for the SH4 without a floating-point unit.

@item -m4-single-only
@opindex m4-single-only
Generate code for the SH4 with a floating-point unit that only
supports single-precision arithmetic.

@item -m4-single
@opindex m4-single
Generate code for the SH4 assuming the floating-point unit is in
single-precision mode by default.

@item -m4
@opindex m4
Generate code for the SH4.

@item -m4a-nofpu
@opindex m4a-nofpu
Generate code for the SH4al-dsp, or for a SH4a in such a way that the
floating-point unit is not used.

@item -m4a-single-only
@opindex m4a-single-only
Generate code for the SH4a, in such a way that no double-precision
floating point operations are used.

@item -m4a-single
@opindex m4a-single
Generate code for the SH4a assuming the floating-point unit is in
single-precision mode by default.

@item -m4a
@opindex m4a
Generate code for the SH4a.

@item -m4al
@opindex m4al
Same as @option{-m4a-nofpu}, except that it implicitly passes
@option{-dsp} to the assembler.  GCC doesn't generate any DSP
instructions at the moment.

@item -mb
@opindex mb
Compile code for the processor in big endian mode.

@item -ml
@opindex ml
Compile code for the processor in little endian mode.

@item -mdalign
@opindex mdalign
Align doubles at 64-bit boundaries.  Note that this changes the calling
conventions, and thus some functions from the standard C library will
not work unless you recompile it first with @option{-mdalign}.

@item -mrelax
@opindex mrelax
Shorten some address references at link time, when possible; uses the
```



```
linker option @option{-relax}.

@item -mbigtable
@opindex mbigtable
Use 32-bit offsets in @code{switch} tables.  The default is to use
16-bit offsets.

@item -mbitops
@opindex mbitops
Enable the use of bit manipulation instructions on SH2A.

@item -mfmovd
@opindex mfmovd
Enable the use of the instruction @code{fmovd}.

@item -mhitachi
@opindex mhitachi
Comply with the calling conventions defined by Renesas.

@item -mrenesas
@opindex mhitachi
Comply with the calling conventions defined by Renesas.

@item -mno-renesas
@opindex mhitachi
Comply with the calling conventions defined for GCC before the Renesas
conventions were available.  This option is the default for all
targets of the SH toolchain except for @samp{sh-symbianelf}.

@item -mnomacsave
@opindex mnomacsave
Mark the @code{MAC} register as call-clobbered, even if
@option{-mhitachi} is given.

@item -mieee
@opindex mieee
Increase IEEE-compliance of floating-point code.
At the moment, this is equivalent to @option{-fno-finite-math-only}.
When generating 16 bit SH opcodes, getting IEEE-conforming results for
comparisons of NANs / infinities incurs extra overhead in every
floating point comparison, therefore the default is set to
@option{-ffinite-math-only}.

@item -minline-ic_invalidate
@opindex minline-ic_invalidate
Inline code to invalidate instruction cache entries after setting up
nested function trampolines.
This option has no effect if -musermode is in effect and the selected
code generation option (e.g. -m4) does not allow the use of the icbi
instruction.
If the selected code generation option does not allow the use of the icbi
instruction, and -musermode is not in effect, the inlined code will
manipulate the instruction cache address array directly with an associative
write.  This not only requires privileged mode, but it will also
fail if the cache line had been mapped via the TLB and has become unmapped.

@item -misize
@opindex misize
Dump instruction size and location in the assembly code.

@item -mpadstruct
@opindex mpadstruct
This option is deprecated.  It pads structures to multiple of 4 bytes,
which is incompatible with the SH ABI@.

@item -mspace
```

```
@opindex mspace
Optimize for space instead of speed.  Implied by @option{-Os}.

@item -mprefergot
@opindex mprefergot
When generating position-independent code, emit function calls using
the Global Offset Table instead of the Procedure Linkage Table.

@item -musermode
@opindex musermode
Don't generate privileged mode only code; implies -mno-inline-ic_invalidate
if the inlined code would not work in user mode.
This is the default when the target is @code{sh-*-linux*}.

@item -multcost=@var{number}
@opindex multcost=@var{number}
Set the cost to assume for a multiply insn.

@item -mdiv=@var{strategy}
@opindex mdiv=@var{strategy}
Set the division strategy to use for SHmedia code.  @var{strategy} must be
one of: call, call2, fp, inv, inv:minlat, inv20u, inv20l, inv:call,
inv:call2, inv:fp .
"fp" performs the operation in floating point.  This has a very high latency,
but needs only a few instructions, so it might be a good choice if
your code has enough easily exploitable ILP to allow the compiler to
schedule the floating point instructions together with other instructions.
Division by zero causes a floating point exception.
"inv" uses integer operations to calculate the inverse of the divisor,
and then multiplies the dividend with the inverse.  This strategy allows
cse and hoisting of the inverse calculation.  Division by zero calculates
an unspecified result, but does not trap.
"inv:minlat" is a variant of "inv" where if no cse / hoisting opportunities
have been found, or if the entire operation has been hoisted to the same
place, the last stages of the inverse calculation are intertwined with the
final multiply to reduce the overall latency, at the expense of using a few
more instructions, and thus offering fewer scheduling opportunities with
other code.
"call" calls a library function that usually implements the inv:minlat
strategy.
This gives high code density for m5-*media-nofpu compilations.
"call2" uses a different entry point of the same library function, where it
assumes that a pointer to a lookup table has already been set up, which
exposes the pointer load to cse / code hoisting optimizations.
"inv:call", "inv:call2" and "inv:fp" all use the "inv" algorithm for initial
code generation, but if the code stays unoptimized, revert to the "call",
"call2", or "fp" strategies, respectively.  Note that the
potentially-trapping side effect of division by zero is carried by a
separate instruction, so it is possible that all the integer instructions
are hoisted out, but the marker for the side effect stays where it is.
A recombination to fp operations or a call is not possible in that case.
"inv20u" and "inv20l" are variants of the "inv:minlat" strategy.  In the case
that the inverse calculation was nor separated from the multiply, they speed
up division where the dividend fits into 20 bits (plus sign where applicable),
by inserting a test to skip a number of operations in this case; this test
slows down the case of larger dividends.  inv20u assumes the case of a such
a small dividend to be unlikely, and inv20l assumes it to be likely.

@item -mdivsi3_libfunc=@var{name}
@opindex mdivsi3_libfunc=@var{name}
Set the name of the library function used for 32 bit signed division to
@var{name}.  This only affect the name used in the call and inv:call
division strategies, and the compiler will still expect the same
sets of input/output/clobbered registers as if this option was not present.

@item -mfixed-range=@var{register-range}
```



```
@opindex mfixed-range
Generate code treating the given register range as fixed registers.
A fixed register is one that the register allocator can not use.  This is
useful when compiling kernel code.  A register range is specified as
two registers separated by a dash.  Multiple register ranges can be
specified separated by a comma.

@item -madjust-unroll
@opindex madjust-unroll
Throttle unrolling to avoid thrashing target registers.
This option only has an effect if the gcc code base supports the
TARGET_ADJUST_UNROLL_MAX target hook.

@item -mindexed-addressing
@opindex mindexed-addressing
Enable the use of the indexed addressing mode for SHmedia32/SHcompact.
This is only safe if the hardware and/or OS implement 32 bit wrap-around
semantics for the indexed addressing mode.  The architecture allows the
implementation of processors with 64 bit MMU, which the OS could use to
get 32 bit addressing, but since no current hardware implementation supports
this or any other way to make the indexed addressing mode safe to use in
the 32 bit ABI, the default is -mno-indexed-addressing.

@item -mgettrcost=@var{number}
@opindex mgettrcost=@var{number}
Set the cost assumed for the gettr instruction to @var{number}.
The default is 2 if @option{-mpt-fixed} is in effect, 100 otherwise.

@item -mpt-fixed
@opindex mpt-fixed
Assume pt* instructions won't trap.  This will generally generate better
scheduled code, but is unsafe on current hardware.  The current architecture
definition says that ptabs and ptrel trap when the target anded with 3 is 3.
This has the unintentional effect of making it unsafe to schedule ptabs /
ptrel before a branch, or hoist it out of a loop.  For example,
__do_global_ctors, a part of libgcc that runs constructors at program
startup, calls functions in a list which is delimited by @minus{}1.  With the
-mpt-fixed option, the ptabs will be done before testing against @minus{}1.
That means that all the constructors will be run a bit quicker, but when
the loop comes to the end of the list, the program crashes because ptabs
loads @minus{}1 into a target register.  Since this option is unsafe for any
hardware implementing the current architecture specification, the default
is -mno-pt-fixed.  Unless the user specifies a specific cost with
@option{-mgettrcost}, -mno-pt-fixed also implies @option{-mgettrcost=100};
this deters register allocation using target registers for storing
ordinary integers.

@item -minvalid-symbols
@opindex minvalid-symbols
Assume symbols might be invalid.  Ordinary function symbols generated by
the compiler will always be valid to load with movi/shori/ptabs or
movi/shori/ptrel, but with assembler and/or linker tricks it is possible
to generate symbols that will cause ptabs / ptrel to trap.
This option is only meaningful when @option{-mno-pt-fixed} is in effect.
It will then prevent cross-basic-block cse, hoisting and most scheduling
of symbol loads.  The default is @option{-mno-invalid-symbols}.
@end table

@node SPARC Options
@subsection SPARC Options
@cindex SPARC options

These @samp{-m} options are supported on the SPARC:

@table @gcctabopt
@item -mno-app-regs
```

```
@itemx -mapp-regs
@opindex mno-app-regs
@opindex mapp-regs
Specify @option{-mapp-regs} to generate output using the global registers
2 through 4, which the SPARC SVR4 ABI reserves for applications.  This
is the default.

To be fully SVR4 ABI compliant at the cost of some performance loss,
specify @option{-mno-app-regs}.  You should compile libraries and system
software with this option.

@item -mfpu
@itemx -mhard-float
@opindex mfpu
@opindex mhard-float
Generate output containing floating point instructions.  This is the
default.

@item -mno-fpu
@itemx -msoft-float
@opindex mno-fpu
@opindex msoft-float
Generate output containing library calls for floating point.
@strong{Warning:} the requisite libraries are not available for all SPARC
targets.  Normally the facilities of the machine's usual C compiler are
used, but this cannot be done directly in cross-compilation.  You must make
your own arrangements to provide suitable library functions for
cross-compilation.  The embedded targets @samp{sparc-*-aout} and
@samp{sparclite-*-*} do provide software floating point support.

@option{-msoft-float} changes the calling convention in the output file;
therefore, it is only useful if you compile @emph{all} of a program with
this option.  In particular, you need to compile @file{libgcc.a}, the
library that comes with GCC, with @option{-msoft-float} in order for
this to work.

@item -mhard-quad-float
@opindex mhard-quad-float
Generate output containing quad-word (long double) floating point
instructions.

@item -msoft-quad-float
@opindex msoft-quad-float
Generate output containing library calls for quad-word (long double)
floating point instructions.  The functions called are those specified
in the SPARC ABI@.  This is the default.

As of this writing, there are no SPARC implementations that have hardware
support for the quad-word floating point instructions.  They all invoke
a trap handler for one of these instructions, and then the trap handler
emulates the effect of the instruction.  Because of the trap handler overhead,
this is much slower than calling the ABI library routines.  Thus the
@option{-msoft-quad-float} option is the default.

@item -mno-unaligned-doubles
@itemx -munaligned-doubles
@opindex mno-unaligned-doubles
@opindex munaligned-doubles
Assume that doubles have 8 byte alignment.  This is the default.

With @option{-munaligned-doubles}, GCC assumes that doubles have 8 byte
alignment only if they are contained in another type, or if they have an
absolute address.  Otherwise, it assumes they have 4 byte alignment.
Specifying this option avoids some rare compatibility problems with code
generated by other compilers.  It is not the default because it results
in a performance loss, especially for floating point code.
```



```
@item -mno-faster-structs
@itemx -mfaster-structs
@opindex mno-faster-structs
@opindex mfaster-structs
With @option{-mfaster-structs}, the compiler assumes that structures
should have 8 byte alignment.  This enables the use of pairs of
@code{ldd} and @code{std} instructions for copies in structure
assignment, in place of twice as many @code{ld} and @code{st} pairs.
However, the use of this changed alignment directly violates the SPARC
ABI@.  Thus, it's intended only for use on targets where the developer
acknowledges that their resulting code will not be directly in line with
the rules of the ABI@.

@item -mimpure-text
@opindex mimpure-text
@option{-mimpure-text}, used in addition to @option{-shared}, tells
the compiler to not pass @option{-z text} to the linker when linking a
shared object.  Using this option, you can link position-dependent
code into a shared object.

@option{-mimpure-text} suppresses the ''relocations remain against
allocatable but non-writable sections'' linker error message.
However, the necessary relocations will trigger copy-on-write, and the
shared object is not actually shared across processes.  Instead of
using @option{-mimpure-text}, you should compile all source code with
@option{-fpic} or @option{-fPIC}.

This option is only available on SunOS and Solaris.

@item -mcpu=@var{cpu_type}
@opindex mcpu
Set the instruction set, register set, and instruction scheduling parameters
for machine type @var{cpu_type}.  Supported values for @var{cpu_type} are
@samp{v7}, @samp{cypress}, @samp{v8}, @samp{supersparc}, @samp{sparclite},
@samp{f930}, @samp{f934}, @samp{hypersparc}, @samp{sparclite86x},
@samp{sparclet}, @samp{tsc701}, @samp{v9}, @samp{ultrasparc},
@samp{ultrasparc3}, @samp{niagara} and @samp{niagara2}.

Default instruction scheduling parameters are used for values that select
an architecture and not an implementation.  These are @samp{v7}, @samp{v8},
@samp{sparclite}, @samp{sparclet}, @samp{v9}.

Here is a list of each supported architecture and their supported
implementations.

@smallexample
    v7:             cypress
    v8:             supersparc, hypersparc
    sparclite:      f930, f934, sparclite86x
    sparclet:       tsc701
    v9:             ultrasparc, ultrasparc3, niagara, niagara2
@end smallexample

By default (unless configured otherwise), GCC generates code for the V7
variant of the SPARC architecture.  With @option{-mcpu=cypress}, the compiler
additionally optimizes it for the Cypress CY7C602 chip, as used in the
SPARCStation/SPARCServer 3xx series.  This is also appropriate for the older
SPARCStation 1, 2, IPX etc.

With @option{-mcpu=v8}, GCC generates code for the V8 variant of the SPARC
architecture.  The only difference from V7 code is that the compiler emits
the integer multiply and integer divide instructions which exist in SPARC-V8
but not in SPARC-V7.  With @option{-mcpu=supersparc}, the compiler additionally
optimizes it for the SuperSPARC chip, as used in the SPARCStation 10, 1000 and
2000 series.
```

With @option{-mcpu=sparclite}, GCC generates code for the SPARClite variant of
the SPARC architecture. This adds the integer multiply, integer divide step
and scan (@code{ffs}) instructions which exist in SPARClite but not in SPARC-V7.
With @option{-mcpu=f930}, the compiler additionally optimizes it for the
Fujitsu MB86930 chip, which is the original SPARClite, with no FPU@. With
@option{-mcpu=f934}, the compiler additionally optimizes it for the Fujitsu
MB86934 chip, which is the more recent SPARClite with FPU@.

With @option{-mcpu=sparclet}, GCC generates code for the SPARClet variant of
the SPARC architecture. This adds the integer multiply, multiply/accumulate,
integer divide step and scan (@code{ffs}) instructions which exist in SPARClet
but not in SPARC-V7. With @option{-mcpu=tsc701}, the compiler additionally
optimizes it for the TEMIC SPARClet chip.

With @option{-mcpu=v9}, GCC generates code for the V9 variant of the SPARC
architecture. This adds 64-bit integer and floating-point move instructions,
3 additional floating-point condition code registers and conditional move
instructions. With @option{-mcpu=ultrasparc}, the compiler additionally
optimizes it for the Sun UltraSPARC I/II/IIi chips. With
@option{-mcpu=ultrasparc3}, the compiler additionally optimizes it for the
Sun UltraSPARC III/III+/IIIi/IIIi+/IV/IV+ chips. With
@option{-mcpu=niagara}, the compiler additionally optimizes it for
Sun UltraSPARC T1 chips. With @option{-mcpu=niagara2}, the compiler
additionally optimizes it for Sun UltraSPARC T2 chips.

@item -mtune=@var{cpu_type}
@opindex mtune
Set the instruction scheduling parameters for machine type
@var{cpu_type}, but do not set the instruction set or register set that the
option @option{-mcpu=@var{cpu_type}} would.

The same values for @option{-mcpu=@var{cpu_type}} can be used for
@option{-mtune=@var{cpu_type}}, but the only useful values are those
that select a particular cpu implementation. Those are @samp{cypress},
@samp{supersparc}, @samp{hypersparc}, @samp{f930}, @samp{f934},
@samp{sparclite86x}, @samp{tsc701}, @samp{ultrasparc},
@samp{ultrasparc3}, @samp{niagara}, and @samp{niagara2}.

@item -mv8plus
@itemx -mno-v8plus
@opindex mv8plus
@opindex mno-v8plus
With @option{-mv8plus}, GCC generates code for the SPARC-V8+ ABI@. The
difference from the V8 ABI is that the global and out registers are
considered 64-bit wide. This is enabled by default on Solaris in 32-bit
mode for all SPARC-V9 processors.

@item -mvis
@itemx -mno-vis
@opindex mvis
@opindex mno-vis
With @option{-mvis}, GCC generates code that takes advantage of the UltraSPARC
Visual Instruction Set extensions. The default is @option{-mno-vis}.

@item -mno-integer-ldd-std
@opindex mno-integer-ldd-std
With @option{-mno-integer-ldd-std}, GCC does not use the @code{ldd}
and @code{std} instructions for integer operands in 32-bit mode. This
is for use with legacy code using 64-bit quantities which are not
64-bit aligned.

@item -massume-32bit-callers
@opindex massume-32bit-callers
With @option{-massume-32bit-callers}, The type promotion of function
arguments is altered such that integer arguments smaller than the word

size are extended in the callee rather than the caller. This is
necessary for system calls from 32bit processes to 64bit kernels in
certain systems. This option should not be used in any situation
other than compiling the kernels of such systems, and has not been
tested outside of that scenario.
@end table

These @samp{-m} options are supported in addition to the above
on SPARC-V9 processors in 64-bit environments:

@table @gcctabopt
@item -mlittle-endian
@opindex mlittle-endian
Generate code for a processor running in little-endian mode. It is only
available for a few configurations and most notably not on Solaris and Linux.

@item -m32
@itemx -m64
@opindex m32
@opindex m64
Generate code for a 32-bit or 64-bit environment.
The 32-bit environment sets int, long and pointer to 32 bits.
The 64-bit environment sets int to 32 bits and long and pointer
to 64 bits.

@item -mcmodel=medlow
@opindex mcmodel=medlow
Generate code for the Medium/Low code model: 64-bit addresses, programs
must be linked in the low 32 bits of memory. Programs can be statically
or dynamically linked.

@item -mcmodel=medmid
@opindex mcmodel=medmid
Generate code for the Medium/Middle code model: 64-bit addresses, programs
must be linked in the low 44 bits of memory, the text and data segments must
be less than 2GB in size and the data segment must be located within 2GB of
the text segment.

@item -mcmodel=medany
@opindex mcmodel=medany
Generate code for the Medium/Anywhere code model: 64-bit addresses, programs
may be linked anywhere in memory, the text and data segments must be less
than 2GB in size and the data segment must be located within 2GB of the
text segment.

@item -mcmodel=embmedany
@opindex mcmodel=embmedany
Generate code for the Medium/Anywhere code model for embedded systems:
64-bit addresses, the text and data segments must be less than 2GB in
size, both starting anywhere in memory (determined at link time). The
global register %g4 points to the base of the data segment. Programs
are statically linked and PIC is not supported.

@item -mstack-bias
@itemx -mno-stack-bias
@opindex mstack-bias
@opindex mno-stack-bias
With @option{-mstack-bias}, GCC assumes that the stack pointer, and
frame pointer if present, are offset by @minus{}2047 which must be added back
when making stack frame references. This is the default in 64-bit mode.
Otherwise, assume no such offset is present.
@end table

These switches are supported in addition to the above on Solaris:

@table @gcctabopt

```
@item -threads
@opindex threads
Add support for multithreading using the Solaris threads library.  This
option sets flags for both the preprocessor and linker.  This option does
not affect the thread safety of object code produced by the compiler or
that of libraries supplied with it.

@item -pthreads
@opindex pthreads
Add support for multithreading using the POSIX threads library.  This
option sets flags for both the preprocessor and linker.  This option does
not affect the thread safety of object code produced  by the compiler or
that of libraries supplied with it.

@item -pthread
@opindex pthread
This is a synonym for @option{-pthreads}.
@end table

@node SPU Options
@subsection SPU Options
@cindex SPU options

These @samp{-m} options are supported on the SPU:

@table @gcctabopt
@item -mwarn-reloc
@itemx -merror-reloc
@opindex mwarn-reloc
@opindex merror-reloc

The loader for SPU does not handle dynamic relocations.  By default, GCC
will give an error when it generates code that requires a dynamic
relocation.  @option{-mno-error-reloc} disables the error,
@option{-mwarn-reloc} will generate a warning instead.

@item -msafe-dma
@itemx -munsafe-dma
@opindex msafe-dma
@opindex munsafe-dma

Instructions which initiate or test completion of DMA must not be
reordered with respect to loads and stores of the memory which is being
accessed.  Users typically address this problem using the volatile
keyword, but that can lead to inefficient code in places where the
memory is known to not change.  Rather than mark the memory as volatile
we treat the DMA instructions as potentially effecting all memory.  With
@option{-munsafe-dma} users must use the volatile keyword to protect
memory accesses.

@item -mbranch-hints
@opindex mbranch-hints

By default, GCC will generate a branch hint instruction to avoid
pipeline stalls for always taken or probably taken branches.  A hint
will not be generated closer than 8 instructions away from its branch.
There is little reason to disable them, except for debugging purposes,
or to make an object a little bit smaller.

@item -msmall-mem
@itemx -mlarge-mem
@opindex msmall-mem
@opindex mlarge-mem

By default, GCC generates code assuming that addresses are never larger
than 18 bits.  With @option{-mlarge-mem} code is generated that assumes
```



```
a full 32 bit address.

@item -mstdmain
@opindex mstdmain

By default, GCC links against startup code that assumes the SPU-style
main function interface (which has an unconventional parameter list).
With @option{-mstdmain}, GCC will link your program against startup
code that assumes a C99-style interface to @code{main}, including a
local copy of @code{argv} strings.

@item -mfixed-range=@var{register-range}
@opindex mfixed-range
Generate code treating the given register range as fixed registers.
A fixed register is one that the register allocator can not use.  This is
useful when compiling kernel code.  A register range is specified as
two registers separated by a dash.  Multiple register ranges can be
specified separated by a comma.

@item -mdual-nops
@itemx -mdual-nops=@var{n}
@opindex mdual-nops
By default, GCC will insert nops to increase dual issue when it expects
it to increase performance.  @var{n} can be a value from 0 to 10.  A
smaller @var{n} will insert fewer nops.  10 is the default, 0 is the
same as @option{-mno-dual-nops}.  Disabled with @option{-Os}.

@item -mhint-max-nops=@var{n}
@opindex mhint-max-nops
Maximum number of nops to insert for a branch hint.  A branch hint must
be at least 8 instructions away from the branch it is effecting.  GCC
will insert up to @var{n} nops to enforce this, otherwise it will not
generate the branch hint.

@item -mhint-max-distance=@var{n}
@opindex mhint-max-distance
The encoding of the branch hint instruction limits the hint to be within
256 instructions of the branch it is effecting.  By default, GCC makes
sure it is within 125.

@item -msafe-hints
@opindex msafe-hints
Work around a hardware bug which causes the SPU to stall indefinitely.
By default, GCC will insert the @code{hbrp} instruction to make sure
this stall won't happen.

@end table

@node System V Options
@subsection Options for System V

These additional options are available on System V Release 4 for
compatibility with other compilers on those systems:

@table @gcctabopt
@item -G
@opindex G
Create a shared object.
It is recommended that @option{-symbolic} or @option{-shared} be used instead.

@item -Qy
@opindex Qy
Identify the versions of each tool used by the compiler, in a
@code{.ident} assembler directive in the output.

@item -Qn
```

```
@opindex Qn
Refrain from adding @code{.ident} directives to the output file (this is
the default).

@item -YP,@var{dirs}
@opindex YP
Search the directories @var{dirs}, and no others, for libraries
specified with @option{-l}.

@item -Ym,@var{dir}
@opindex Ym
Look in the directory @var{dir} to find the M4 preprocessor.
The assembler uses this option.
@c This is supposed to go with a -Yd for predefined M4 macro files, but
@c the generic assembler that comes with Solaris takes just -Ym.
@end table

@node V850 Options
@subsection V850 Options
@cindex V850 Options

These @samp{-m} options are defined for V850 implementations:

@table @gcctabopt
@item -mlong-calls
@itemx -mno-long-calls
@opindex mlong-calls
@opindex mno-long-calls
Treat all calls as being far away (near).  If calls are assumed to be
far away, the compiler will always load the functions address up into a
register, and call indirect through the pointer.

@item -mno-ep
@itemx -mep
@opindex mno-ep
@opindex mep
Do not optimize (do optimize) basic blocks that use the same index
pointer 4 or more times to copy pointer into the @code{ep} register, and
use the shorter @code{sld} and @code{sst} instructions.  The @option{-mep}
option is on by default if you optimize.

@item -mno-prolog-function
@itemx -mprolog-function
@opindex mno-prolog-function
@opindex mprolog-function
Do not use (do use) external functions to save and restore registers
at the prologue and epilogue of a function.  The external functions
are slower, but use less code space if more than one function saves
the same number of registers.  The @option{-mprolog-function} option
is on by default if you optimize.

@item -mspace
@opindex mspace
Try to make the code as small as possible.  At present, this just turns
on the @option{-mep} and @option{-mprolog-function} options.

@item -mtda=@var{n}
@opindex mtda
Put static or global variables whose size is @var{n} bytes or less into
the tiny data area that register @code{ep} points to.  The tiny data
area can hold up to 256 bytes in total (128 bytes for byte references).

@item -msda=@var{n}
@opindex msda
Put static or global variables whose size is @var{n} bytes or less into
the small data area that register @code{gp} points to.  The small data
```



```
area can hold up to 64 kilobytes.

@item -mzda=@var{n}
@opindex mzda
Put static or global variables whose size is @var{n} bytes or less into
the first 32 kilobytes of memory.

@item -mv850
@opindex mv850
Specify that the target processor is the V850.

@item -mbig-switch
@opindex mbig-switch
Generate code suitable for big switch tables.  Use this option only if
the assembler/linker complain about out of range branches within a switch
table.

@item -mapp-regs
@opindex mapp-regs
This option will cause r2 and r5 to be used in the code generated by
the compiler.  This setting is the default.

@item -mno-app-regs
@opindex mno-app-regs
This option will cause r2 and r5 to be treated as fixed registers.

@item -mv850e1
@opindex mv850e1
Specify that the target processor is the V850E1.  The preprocessor
constants @samp{__v850e1__} and @samp{__v850e__} will be defined if
this option is used.

@item -mv850e
@opindex mv850e
Specify that the target processor is the V850E@.  The preprocessor
constant @samp{__v850e__} will be defined if this option is used.

If neither @option{-mv850} nor @option{-mv850e} nor @option{-mv850e1}
are defined then a default target processor will be chosen and the
relevant @samp{__v850*__} preprocessor constant will be defined.

The preprocessor constants @samp{__v850} and @samp{__v851__} are always
defined, regardless of which processor variant is the target.

@item -mdisable-callt
@opindex mdisable-callt
This option will suppress generation of the CALLT instruction for the
v850e and v850e1 flavors of the v850 architecture.  The default is
@option{-mno-disable-callt} which allows the CALLT instruction to be used.

@end table

@node VAX Options
@subsection VAX Options
@cindex VAX options

These @samp{-m} options are defined for the VAX:

@table @gcctabopt
@item -munix
@opindex munix
Do not output certain jump instructions (@code{aobleq} and so on)
that the Unix assembler for the VAX cannot handle across long
ranges.

@item -mgnu
```

```
@opindex mgnu
Do output those jump instructions, on the assumption that you
will assemble with the GNU assembler.

@item -mg
@opindex mg
Output code for g-format floating point numbers instead of d-format.
@end table

@node VxWorks Options
@subsection VxWorks Options
@cindex VxWorks Options

The options in this section are defined for all VxWorks targets.
Options specific to the target hardware are listed with the other
options for that target.

@table @gcctabopt
@item -mrtp
@opindex mrtp
GCC can generate code for both VxWorks kernels and real time processes
(RTPs).  This option switches from the former to the latter.  It also
defines the preprocessor macro @code{__RTP__}.

@item -non-static
@opindex non-static
Link an RTP executable against shared libraries rather than static
libraries.  The options @option{-static} and @option{-shared} can
also be used for RTPs (@pxref{Link Options}); @option{-static}
is the default.

@item -Bstatic
@itemx -Bdynamic
@opindex Bstatic
@opindex Bdynamic
These options are passed down to the linker.  They are defined for
compatibility with Diab.

@item -Xbind-lazy
@opindex Xbind-lazy
Enable lazy binding of function calls.  This option is equivalent to
@option{-Wl,-z,now} and is defined for compatibility with Diab.

@item -Xbind-now
@opindex Xbind-now
Disable lazy binding of function calls.  This option is the default and
is defined for compatibility with Diab.
@end table

@node x86-64 Options
@subsection x86-64 Options
@cindex x86-64 options

These are listed under @xref{i386 and x86-64 Options}.

@node i386 and x86-64 Windows Options
@subsection i386 and x86-64 Windows Options
@cindex i386 and x86-64 Windows Options

These additional options are available for Windows targets:

@table @gcctabopt
@item -mconsole
@opindex mconsole
This option is available for Cygwin and MinGW targets.  It
specifies that a console application is to be generated, by
```



```
instructing the linker to set the PE header subsystem type
required for console applications.
This is the default behaviour for Cygwin and MinGW targets.

@item -mcygwin
@opindex mcygwin
This option is available for Cygwin targets.  It specifies that
the Cygwin internal interface is to be used for predefined
preprocessor macros, C runtime libraries and related linker
paths and options.  For Cygwin targets this is the default behaviour.
This option is deprecated and will be removed in a future release.

@item -mno-cygwin
@opindex mno-cygwin
This option is available for Cygwin targets.  It specifies that
the MinGW internal interface is to be used instead of Cygwin's, by
setting MinGW-related predefined macros and linker paths and default
library options.
This option is deprecated and will be removed in a future release.

@item -mdll
@opindex mdll
This option is available for Cygwin and MinGW targets.  It
specifies that a DLL - a dynamic link library - is to be
generated, enabling the selection of the required runtime
startup object and entry point.

@item -mnop-fun-dllimport
@opindex mnop-fun-dllimport
This option is available for Cygwin and MinGW targets.  It
specifies that the dllimport attribute should be ignored.

@item -mthread
@opindex mthread
This option is available for MinGW targets. It specifies
that MinGW-specific thread support is to be used.

@item -mwin32
@opindex mwin32
This option is available for Cygwin and MinGW targets.  It
specifies that the typical Windows pre-defined macros are to
be set in the pre-processor, but does not influence the choice
of runtime library/startup code.

@item -mwindows
@opindex mwindows
This option is available for Cygwin and MinGW targets.  It
specifies that a GUI application is to be generated by
instructing the linker to set the PE header subsystem type
appropriately.
@end table

See also under @ref{i386 and x86-64 Options} for standard options.

@node Xstormy16 Options
@subsection Xstormy16 Options
@cindex Xstormy16 Options

These options are defined for Xstormy16:

@table @gcctabopt
@item -msim
@opindex msim
Choose startup files and linker script suitable for the simulator.
@end table
```

```
@node Xtensa Options
@subsection Xtensa Options
@cindex Xtensa Options

These options are supported for Xtensa targets:

@table @gcctabopt
@item -mconst16
@itemx -mno-const16
@opindex mconst16
@opindex mno-const16
Enable or disable use of @code{CONST16} instructions for loading
constant values.  The @code{CONST16} instruction is currently not a
standard option from Tensilica.  When enabled, @code{CONST16}
instructions are always used in place of the standard @code{L32R}
instructions.  The use of @code{CONST16} is enabled by default only if
the @code{L32R} instruction is not available.

@item -mfused-madd
@itemx -mno-fused-madd
@opindex mfused-madd
@opindex mno-fused-madd
Enable or disable use of fused multiply/add and multiply/subtract
instructions in the floating-point option.  This has no effect if the
floating-point option is not also enabled.  Disabling fused multiply/add
and multiply/subtract instructions forces the compiler to use separate
instructions for the multiply and add/subtract operations.  This may be
desirable in some cases where strict IEEE 754-compliant results are
required: the fused multiply add/subtract instructions do not round the
intermediate result, thereby producing results with @emph{more} bits of
precision than specified by the IEEE standard.  Disabling fused multiply
add/subtract instructions also ensures that the program output is not
sensitive to the compiler's ability to combine multiply and add/subtract
operations.

@item -mserialize-volatile
@itemx -mno-serialize-volatile
@opindex mserialize-volatile
@opindex mno-serialize-volatile
When this option is enabled, GCC inserts @code{MEMW} instructions before
@code{volatile} memory references to guarantee sequential consistency.
The default is @option{-mserialize-volatile}.  Use
@option{-mno-serialize-volatile} to omit the @code{MEMW} instructions.

@item -mtext-section-literals
@itemx -mno-text-section-literals
@opindex mtext-section-literals
@opindex mno-text-section-literals
Control the treatment of literal pools.  The default is
@option{-mno-text-section-literals}, which places literals in a separate
section in the output file.  This allows the literal pool to be placed
in a data RAM/ROM, and it also allows the linker to combine literal
pools from separate object files to remove redundant literals and
improve code size.  With @option{-mtext-section-literals}, the literals
are interspersed in the text section in order to keep them as close as
possible to their references.  This may be necessary for large assembly
files.

@item -mtarget-align
@itemx -mno-target-align
@opindex mtarget-align
@opindex mno-target-align
When this option is enabled, GCC instructs the assembler to
automatically align instructions to reduce branch penalties at the
expense of some code density.  The assembler attempts to widen density
instructions to align branch targets and the instructions following call
```



```
instructions.  If there are not enough preceding safe density
instructions to align a target, no widening will be performed.  The
default is @option{-mtarget-align}.  These options do not affect the
treatment of auto-aligned instructions like @code{LOOP}, which the
assembler will always align, either by widening density instructions or
by inserting no-op instructions.

@item -mlongcalls
@itemx -mno-longcalls
@opindex mlongcalls
@opindex mno-longcalls
When this option is enabled, GCC instructs the assembler to translate
direct calls to indirect calls unless it can determine that the target
of a direct call is in the range allowed by the call instruction.  This
translation typically occurs for calls to functions in other source
files.  Specifically, the assembler translates a direct @code{CALL}
instruction into an @code{L32R} followed by a @code{CALLX} instruction.
The default is @option{-mno-longcalls}.  This option should be used in
programs where the call target can potentially be out of range.  This
option is implemented in the assembler, not the compiler, so the
assembly code generated by GCC will still show direct call
instructions---look at the disassembled object code to see the actual
instructions.  Note that the assembler will use an indirect call for
every cross-file call, not just those that really will be out of range.
@end table

@node zSeries Options
@subsection zSeries Options
@cindex zSeries options

These are listed under @xref{S/390 and zSeries Options}.

@node Code Gen Options
@section Options for Code Generation Conventions
@cindex code generation conventions
@cindex options, code generation
@cindex run-time options

These machine-independent options control the interface conventions
used in code generation.

Most of them have both positive and negative forms; the negative form
of @option{-ffoo} would be @option{-fno-foo}.  In the table below, only
one of the forms is listed---the one which is not the default.  You
can figure out the other form by either removing @samp{no-} or adding
it.

@table @gcctabopt
@item -fbounds-check
@opindex fbounds-check
For front-ends that support it, generate additional code to check that
indices used to access arrays are within the declared range.  This is
currently only supported by the Java and Fortran front-ends, where
this option defaults to true and false respectively.

@item -ftrapv
@opindex ftrapv
This option generates traps for signed overflow on addition, subtraction,
multiplication operations.

@item -fwrapv
@opindex fwrapv
This option instructs the compiler to assume that signed arithmetic
overflow of addition, subtraction and multiplication wraps around
using twos-complement representation.  This flag enables some optimizations
and disables others.  This option is enabled by default for the Java
```

front-end, as required by the Java language specification.

@item -fexceptions
@opindex fexceptions
Enable exception handling. Generates extra code needed to propagate
exceptions. For some targets, this implies GCC will generate frame
unwind information for all functions, which can produce significant data
size overhead, although it does not affect execution. If you do not
specify this option, GCC will enable it by default for languages like
C++ which normally require exception handling, and disable it for
languages like C that do not normally require it. However, you may need
to enable this option when compiling C code that needs to interoperate
properly with exception handlers written in C++. You may also wish to
disable this option if you are compiling older C++ programs that don't
use exception handling.

@item -fnon-call-exceptions
@opindex fnon-call-exceptions
Generate code that allows trapping instructions to throw exceptions.
Note that this requires platform-specific runtime support that does
not exist everywhere. Moreover, it only allows @emph{trapping}
instructions to throw exceptions, i.e.@: memory references or floating
point instructions. It does not allow exceptions to be thrown from
arbitrary signal handlers such as @code{SIGALRM}.

@item -funwind-tables
@opindex funwind-tables
Similar to @option{-fexceptions}, except that it will just generate any needed
static data, but will not affect the generated code in any other way.
You will normally not enable this option; instead, a language processor
that needs this handling would enable it on your behalf.

@item -fasynchronous-unwind-tables
@opindex fasynchronous-unwind-tables
Generate unwind table in dwarf2 format, if supported by target machine. The
table is exact at each instruction boundary, so it can be used for stack
unwinding from asynchronous events (such as debugger or garbage collector).

@item -fpcc-struct-return
@opindex fpcc-struct-return
Return ''short'' @code{struct} and @code{union} values in memory like
longer ones, rather than in registers. This convention is less
efficient, but it has the advantage of allowing intercallability between
GCC-compiled files and files compiled with other compilers, particularly
the Portable C Compiler (pcc).

The precise convention for returning structures in memory depends
on the target configuration macros.

Short structures and unions are those whose size and alignment match
that of some integer type.

@strong{Warning:} code compiled with the @option{-fpcc-struct-return}
switch is not binary compatible with code compiled with the
@option{-freg-struct-return} switch.
Use it to conform to a non-default application binary interface.

@item -freg-struct-return
@opindex freg-struct-return
Return @code{struct} and @code{union} values in registers when possible.
This is more efficient for small structures than
@option{-fpcc-struct-return}.

If you specify neither @option{-fpcc-struct-return} nor
@option{-freg-struct-return}, GCC defaults to whichever convention is
standard for the target. If there is no standard convention, GCC

defaults to @option{-fpcc-struct-return}, except on targets where GCC is
the principal compiler. In those cases, we can choose the standard, and
we chose the more efficient register return alternative.

@strong{Warning:} code compiled with the @option{-freg-struct-return}
switch is not binary compatible with code compiled with the
@option{-fpcc-struct-return} switch.
Use it to conform to a non-default application binary interface.

@item -fshort-enums
@opindex fshort-enums
Allocate to an @code{enum} type only as many bytes as it needs for the
declared range of possible values. Specifically, the @code{enum} type
will be equivalent to the smallest integer type which has enough room.

@strong{Warning:} the @option{-fshort-enums} switch causes GCC to generate
code that is not binary compatible with code generated without that switch.
Use it to conform to a non-default application binary interface.

@item -fshort-double
@opindex fshort-double
Use the same size for @code{double} as for @code{float}.

@strong{Warning:} the @option{-fshort-double} switch causes GCC to generate
code that is not binary compatible with code generated without that switch.
Use it to conform to a non-default application binary interface.

@item -fshort-wchar
@opindex fshort-wchar
Override the underlying type for @samp{wchar_t} to be @samp{short
unsigned int} instead of the default for the target. This option is
useful for building programs to run under WINE@.

@strong{Warning:} the @option{-fshort-wchar} switch causes GCC to generate
code that is not binary compatible with code generated without that switch.
Use it to conform to a non-default application binary interface.

@item -fno-common
@opindex fno-common
In C code, controls the placement of uninitialized global variables.
Unix C compilers have traditionally permitted multiple definitions of
such variables in different compilation units by placing the variables
in a common block.
This is the behavior specified by @option{-fcommon}, and is the default
for GCC on most targets.
On the other hand, this behavior is not required by ISO C, and on some
targets may carry a speed or code size penalty on variable references.
The @option{-fno-common} option specifies that the compiler should place
uninitialized global variables in the data section of the object file,
rather than generating them as common blocks.
This has the effect that if the same variable is declared
(without @code{extern}) in two different compilations,
you will get a multiple-definition error when you link them.
In this case, you must compile with @option{-fcommon} instead.
Compiling with @option{-fno-common} is useful on targets for which
it provides better performance, or if you wish to verify that the
program will work on other systems which always treat uninitialized
variable declarations this way.

@item -fno-ident
@opindex fno-ident
Ignore the @samp{#ident} directive.

@item -finhibit-size-directive
@opindex finhibit-size-directive
Don't output a @code{.size} assembler directive, or anything else that

would cause trouble if the function is split in the middle, and the
two halves are placed at locations far apart in memory. This option is
used when compiling @file{crtstuff.c}; you should not need to use it
for anything else.

@item -fverbose-asm
@opindex fverbose-asm
Put extra commentary information in the generated assembly code to
make it more readable. This option is generally only of use to those
who actually need to read the generated assembly code (perhaps while
debugging the compiler itself).

@option{-fno-verbose-asm}, the default, causes the
extra information to be omitted and is useful when comparing two assembler
files.

@item -frecord-gcc-switches
@opindex frecord-gcc-switches
This switch causes the command line that was used to invoke the
compiler to be recorded into the object file that is being created.
This switch is only implemented on some targets and the exact format
of the recording is target and binary file format dependent, but it
usually takes the form of a section containing ASCII text. This
switch is related to the @option{-fverbose-asm} switch, but that
switch only records information in the assembler output file as
comments, so it never reaches the object file.

@item -fpic
@opindex fpic
@cindex global offset table
@cindex PIC
Generate position-independent code (PIC) suitable for use in a shared
library, if supported for the target machine. Such code accesses all
constant addresses through a global offset table (GOT)@. The dynamic
loader resolves the GOT entries when the program starts (the dynamic
loader is not part of GCC; it is part of the operating system). If
the GOT size for the linked executable exceeds a machine-specific
maximum size, you get an error message from the linker indicating that
@option{-fpic} does not work; in that case, recompile with @option{-fPIC}
instead. (These maximums are 8k on the SPARC and 32k
on the m68k and RS/6000. The 386 has no such limit.)

Position-independent code requires special support, and therefore works
only on certain machines. For the 386, GCC supports PIC for System V
but not for the Sun 386i. Code generated for the IBM RS/6000 is always
position-independent.

When this flag is set, the macros @code{__pic__} and @code{__PIC__}
are defined to 1.

@item -fPIC
@opindex fPIC
If supported for the target machine, emit position-independent code,
suitable for dynamic linking and avoiding any limit on the size of the
global offset table. This option makes a difference on the m68k,
PowerPC and SPARC@.

Position-independent code requires special support, and therefore works
only on certain machines.

When this flag is set, the macros @code{__pic__} and @code{__PIC__}
are defined to 2.

@item -fpie
@itemx -fPIE
@opindex fpie

@opindex fPIE
These options are similar to @option{-fpic} and @option{-fPIC}, but
generated position independent code can be only linked into executables.
Usually these options are used when @option{-pie} GCC option will be
used during linking.

@option{-fpie} and @option{-fPIE} both define the macros
@code{__pie__} and @code{__PIE__}. The macros have the value 1
for @option{-fpie} and 2 for @option{-fPIE}.

@item -fno-jump-tables
@opindex fno-jump-tables
Do not use jump tables for switch statements even where it would be
more efficient than other code generation strategies. This option is
of use in conjunction with @option{-fpic} or @option{-fPIC} for
building code which forms part of a dynamic linker and cannot
reference the address of a jump table. On some targets, jump tables
do not require a GOT and this option is not needed.

@item -ffixed-@var{reg}
@opindex ffixed
Treat the register named @var{reg} as a fixed register; generated code
should never refer to it (except perhaps as a stack pointer, frame
pointer or in some other fixed role).

@var{reg} must be the name of a register. The register names accepted
are machine-specific and are defined in the @code{REGISTER_NAMES}
macro in the machine description macro file.

This flag does not have a negative form, because it specifies a
three-way choice.

@item -fcall-used-@var{reg}
@opindex fcall-used
Treat the register named @var{reg} as an allocable register that is
clobbered by function calls. It may be allocated for temporaries or
variables that do not live across a call. Functions compiled this way
will not save and restore the register @var{reg}.

It is an error to used this flag with the frame pointer or stack pointer.
Use of this flag for other registers that have fixed pervasive roles in
the machine’s execution model will produce disastrous results.

This flag does not have a negative form, because it specifies a
three-way choice.

@item -fcall-saved-@var{reg}
@opindex fcall-saved
Treat the register named @var{reg} as an allocable register saved by
functions. It may be allocated even for temporaries or variables that
live across a call. Functions compiled this way will save and restore
the register @var{reg} if they use it.

It is an error to used this flag with the frame pointer or stack pointer.
Use of this flag for other registers that have fixed pervasive roles in
the machine’s execution model will produce disastrous results.

A different sort of disaster will result from the use of this flag for
a register in which function values may be returned.

This flag does not have a negative form, because it specifies a
three-way choice.

@item -fpack-struct[=@var{n}]
@opindex fpack-struct
Without a value specified, pack all structure members together without

holes. When a value is specified (which must be a small power of two), pack
structure members according to this value, representing the maximum
alignment (that is, objects with default alignment requirements larger than
this will be output potentially unaligned at the next fitting location.

@strong{Warning:} the @option{-fpack-struct} switch causes GCC to generate
code that is not binary compatible with code generated without that switch.
Additionally, it makes the code suboptimal.
Use it to conform to a non-default application binary interface.

@item -finstrument-functions
@opindex finstrument-functions
Generate instrumentation calls for entry and exit to functions. Just
after function entry and just before function exit, the following
profiling functions will be called with the address of the current
function and its call site. (On some platforms,
@code{__builtin_return_address} does not work beyond the current
function, so the call site information may not be available to the
profiling functions otherwise.)

```
@smallexample
void __cyg_profile_func_enter (void *this_fn,
                               void *call_site);
void __cyg_profile_func_exit  (void *this_fn,
                               void *call_site);
@end smallexample
```

The first argument is the address of the start of the current function,
which may be looked up exactly in the symbol table.

This instrumentation is also done for functions expanded inline in other
functions. The profiling calls will indicate where, conceptually, the
inline function is entered and exited. This means that addressable
versions of such functions must be available. If all your uses of a
function are expanded inline, this may mean an additional expansion of
code size. If you use @samp{extern inline} in your C code, an
addressable version of such functions must be provided. (This is
normally the case anyways, but if you get lucky and the optimizer always
expands the functions inline, you might have gotten away without
providing static copies.)

A function may be given the attribute @code{no_instrument_function}, in
which case this instrumentation will not be done. This can be used, for
example, for the profiling functions listed above, high-priority
interrupt routines, and any functions from which the profiling functions
cannot safely be called (perhaps signal handlers, if the profiling
routines generate output or allocate memory).

@item -finstrument-functions-exclude-file-list=@var{file},@var{file},@dots{}
@opindex finstrument-functions-exclude-file-list

Set the list of functions that are excluded from instrumentation (see
the description of @code{-finstrument-functions}). If the file that
contains a function definition matches with one of @var{file}, then
that function is not instrumented. The match is done on substrings:
if the @var{file} parameter is a substring of the file name, it is
considered to be a match.

For example,
@code{-finstrument-functions-exclude-file-list=/bits/stl,include/sys}
will exclude any inline function defined in files whose pathnames
contain @code{/bits/stl} or @code{include/sys}.

If, for some reason, you want to include letter @code{','} in one of
@var{sym}, write @code{'\,'}. For example,
@code{-finstrument-functions-exclude-file-list='\,\,tmp'}

(note the single quote surrounding the option).

@item -finstrument-functions-exclude-function-list=@var{sym},@var{sym},@dots{}
@opindex finstrument-functions-exclude-function-list

This is similar to @code{-finstrument-functions-exclude-file-list},
but this option sets the list of function names to be excluded from
instrumentation. The function name to be matched is its user-visible
name, such as @code{vector<int> blah(const vector<int> &)}, not the
internal mangled name (e.g., @code{_Z4blahRSt6vectorIiSaIiEE}). The
match is done on substrings: if the @var{sym} parameter is a substring
of the function name, it is considered to be a match.

@item -fstack-check
@opindex fstack-check
Generate code to verify that you do not go beyond the boundary of the
stack. You should specify this flag if you are running in an
environment with multiple threads, but only rarely need to specify it in
a single-threaded environment since stack overflow is automatically
detected on nearly all systems if there is only one stack.

Note that this switch does not actually cause checking to be done; the
operating system or the language runtime must do that. The switch causes
generation of code to ensure that they see the stack being extended.

You can additionally specify a string parameter: @code{no} means no
checking, @code{generic} means force the use of old-style checking,
@code{specific} means use the best checking method and is equivalent
to bare @option{-fstack-check}.

Old-style checking is a generic mechanism that requires no specific
target support in the compiler but comes with the following drawbacks:

@enumerate
@item
Modified allocation strategy for large objects: they will always be
allocated dynamically if their size exceeds a fixed threshold.

@item
Fixed limit on the size of the static frame of functions: when it is
topped by a particular function, stack checking is not reliable and
a warning is issued by the compiler.

@item
Inefficiency: because of both the modified allocation strategy and the
generic implementation, the performances of the code are hampered.
@end enumerate

Note that old-style stack checking is also the fallback method for
@code{specific} if no target support has been added in the compiler.

@item -fstack-limit-register=@var{reg}
@itemx -fstack-limit-symbol=@var{sym}
@itemx -fno-stack-limit
@opindex fstack-limit-register
@opindex fstack-limit-symbol
@opindex fno-stack-limit
Generate code to ensure that the stack does not grow beyond a certain value,
either the value of a register or the address of a symbol. If the stack
would grow beyond the value, a signal is raised. For most targets,
the signal is raised before the stack overruns the boundary, so
it is possible to catch the signal without taking special precautions.

For instance, if the stack starts at absolute address @samp{0x80000000}
and grows downwards, you can use the flags
@option{-fstack-limit-symbol=__stack_limit} and

```
@option{-Wl,--defsym,__stack_limit=0x7ffe0000} to enforce a stack limit
of 128KB@.  Note that this may only work with the GNU linker.

@cindex aliasing of parameters
@cindex parameters, aliased
@item -fargument-alias
@itemx -fargument-noalias
@itemx -fargument-noalias-global
@itemx -fargument-noalias-anything
@opindex fargument-alias
@opindex fargument-noalias
@opindex fargument-noalias-global
@opindex fargument-noalias-anything
Specify the possible relationships among parameters and between
parameters and global data.

@option{-fargument-alias} specifies that arguments (parameters) may
alias each other and may alias global storage.@*
@option{-fargument-noalias} specifies that arguments do not alias
each other, but may alias global storage.@*
@option{-fargument-noalias-global} specifies that arguments do not
alias each other and do not alias global storage.
@option{-fargument-noalias-anything} specifies that arguments do not
alias any other storage.

Each language will automatically use whatever option is required by
the language standard.  You should not need to use these options yourself.

@item -fleading-underscore
@opindex fleading-underscore
This option and its counterpart, @option{-fno-leading-underscore}, forcibly
change the way C symbols are represented in the object file.  One use
is to help link with legacy assembly code.

@strong{Warning:} the @option{-fleading-underscore} switch causes GCC to
generate code that is not binary compatible with code generated without that
switch.  Use it to conform to a non-default application binary interface.
Not all targets provide complete support for this switch.

@item -ftls-model=@var{model}
@opindex ftls-model
Alter the thread-local storage model to be used (@pxref{Thread-Local}).
The @var{model} argument should be one of @code{global-dynamic},
@code{local-dynamic}, @code{initial-exec} or @code{local-exec}.

The default without @option{-fpic} is @code{initial-exec}; with
@option{-fpic} the default is @code{global-dynamic}.

@item -fvisibility=@var{default|internal|hidden|protected}
@opindex fvisibility
Set the default ELF image symbol visibility to the specified option---all
symbols will be marked with this unless overridden within the code.
Using this feature can very substantially improve linking and
load times of shared object libraries, produce more optimized
code, provide near-perfect API export and prevent symbol clashes.
It is @strong{strongly} recommended that you use this in any shared objects
you distribute.

Despite the nomenclature, @code{default} always means public ie;
available to be linked against from outside the shared object.
@code{protected} and @code{internal} are pretty useless in real-world
usage so the only other commonly used option will be @code{hidden}.
The default if @option{-fvisibility} isn't specified is
@code{default}, i.e., make every
symbol public---this causes the same behavior as previous versions of
GCC@.
```



```
A good explanation of the benefits offered by ensuring ELF
symbols have the correct visibility is given by ``How To Write
Shared Libraries'' by Ulrich Drepper (which can be found at
@w{@uref{http://people.redhat.com/~drepper/}})---however a superior
solution made possible by this option to marking things hidden when
the default is public is to make the default hidden and mark things
public.  This is the norm with DLL's on Windows and with @option{-fvisibility=hi
and @code{__attribute__ ((visibility("default")))} instead of
@code{__declspec(dllexport)} you get almost identical semantics with
identical syntax.  This is a great boon to those working with
cross-platform projects.

For those adding visibility support to existing code, you may find
@samp{#pragma GCC visibility} of use.  This works by you enclosing
the declarations you wish to set visibility for with (for example)
@samp{#pragma GCC visibility push(hidden)} and
@samp{#pragma GCC visibility pop}.
Bear in mind that symbol visibility should be viewed @strong{as
part of the API interface contract} and thus all new code should
always specify visibility when it is not the default ie; declarations
only for use within the local DSO should @strong{always} be marked explicitly
as hidden as so to avoid PLT indirection overheads---making this
abundantly clear also aids readability and self-documentation of the code.
Note that due to ISO C++ specification requirements, operator new and
operator delete must always be of default visibility.

Be aware that headers from outside your project, in particular system
headers and headers from any other library you use, may not be
expecting to be compiled with visibility other than the default.  You
may need to explicitly say @samp{#pragma GCC visibility push(default)}
before including any such headers.

@samp{extern} declarations are not affected by @samp{-fvisibility}, so
a lot of code can be recompiled with @samp{-fvisibility=hidden} with
no modifications.  However, this means that calls to @samp{extern}
functions with no explicit visibility will use the PLT, so it is more
effective to use @samp{__attribute ((visibility))} and/or
@samp{#pragma GCC visibility} to tell the compiler which @samp{extern}
declarations should be treated as hidden.

Note that @samp{-fvisibility} does affect C++ vague linkage
entities. This means that, for instance, an exception class that will
be thrown between DSOs must be explicitly marked with default
visibility so that the @samp{type_info} nodes will be unified between
the DSOs.

An overview of these techniques, their benefits and how to use them
is at @w{@uref{http://gcc.gnu.org/wiki/Visibility}}.

@end table

@c man end

@node Environment Variables
@section Environment Variables Affecting GCC
@cindex environment variables

@c man begin ENVIRONMENT
This section describes several environment variables that affect how GCC
operates.  Some of them work by specifying directories or prefixes to use
when searching for various kinds of files.  Some are used to specify other
aspects of the compilation environment.

Note that you can also specify places to search using options such as
@option{-B}, @option{-I} and @option{-L} (@pxref{Directory Options}).  These
```

take precedence over places specified using environment variables, which
in turn take precedence over those specified by the configuration of GCC@.
@xref{Driver,, Controlling the Compilation Driver @file{gcc}, gccint,
GNU Compiler Collection (GCC) Internals}.

@table @env
@item LANG
@itemx LC_CTYPE
@c @itemx LC_COLLATE
@itemx LC_MESSAGES
@c @itemx LC_MONETARY
@c @itemx LC_NUMERIC
@c @itemx LC_TIME
@itemx LC_ALL
@findex LANG
@findex LC_CTYPE
@c @findex LC_COLLATE
@findex LC_MESSAGES
@c @findex LC_MONETARY
@c @findex LC_NUMERIC
@c @findex LC_TIME
@findex LC_ALL
@cindex locale
These environment variables control the way that GCC uses
localization information that allow GCC to work with different
national conventions. GCC inspects the locale categories
@env{LC_CTYPE} and @env{LC_MESSAGES} if it has been configured to do
so. These locale categories can be set to any value supported by your
installation. A typical value is @samp{en_GB.UTF-8} for English in the United
Kingdom encoded in UTF-8.

The @env{LC_CTYPE} environment variable specifies character
classification. GCC uses it to determine the character boundaries in
a string; this is needed for some multibyte encodings that contain quote
and escape characters that would otherwise be interpreted as a string
end or escape.

The @env{LC_MESSAGES} environment variable specifies the language to
use in diagnostic messages.

If the @env{LC_ALL} environment variable is set, it overrides the value
of @env{LC_CTYPE} and @env{LC_MESSAGES}; otherwise, @env{LC_CTYPE}
and @env{LC_MESSAGES} default to the value of the @env{LANG}
environment variable. If none of these variables are set, GCC
defaults to traditional C English behavior.

@item TMPDIR
@findex TMPDIR
If @env{TMPDIR} is set, it specifies the directory to use for temporary
files. GCC uses temporary files to hold the output of one stage of
compilation which is to be used as input to the next stage: for example,
the output of the preprocessor, which is the input to the compiler
proper.

@item GCC_EXEC_PREFIX
@findex GCC_EXEC_PREFIX
If @env{GCC_EXEC_PREFIX} is set, it specifies a prefix to use in the
names of the subprograms executed by the compiler. No slash is added
when this prefix is combined with the name of a subprogram, but you can
specify a prefix that ends with a slash if you wish.

If @env{GCC_EXEC_PREFIX} is not set, GCC will attempt to figure out
an appropriate prefix to use based on the pathname it was invoked with.

If GCC cannot find the subprogram using the specified prefix, it
tries looking in the usual places for the subprogram.

The default value of @env{GCC_EXEC_PREFIX} is
@file{@var{prefix}/lib/gcc/} where @var{prefix} is the prefix to
the installed compiler. In many cases @var{prefix} is the value
of @code{prefix} when you ran the @file{configure} script.

Other prefixes specified with @option{-B} take precedence over this prefix.

This prefix is also used for finding files such as @file{crt0.o} that are
used for linking.

In addition, the prefix is used in an unusual way in finding the
directories to search for header files. For each of the standard
directories whose name normally begins with @samp{/usr/local/lib/gcc}
(more precisely, with the value of @env{GCC_INCLUDE_DIR}), GCC tries
replacing that beginning with the specified prefix to produce an
alternate directory name. Thus, with @option{-Bfoo/}, GCC will search
@file{foo/bar} where it would normally search @file{/usr/local/lib/bar}.
These alternate directories are searched first; the standard directories
come next. If a standard directory begins with the configured
@var{prefix} then the value of @var{prefix} is replaced by
@env{GCC_EXEC_PREFIX} when looking for header files.

@item COMPILER_PATH
@findex COMPILER_PATH
The value of @env{COMPILER_PATH} is a colon-separated list of
directories, much like @env{PATH}. GCC tries the directories thus
specified when searching for subprograms, if it can't find the
subprograms using @env{GCC_EXEC_PREFIX}.

@item LIBRARY_PATH
@findex LIBRARY_PATH
The value of @env{LIBRARY_PATH} is a colon-separated list of
directories, much like @env{PATH}. When configured as a native compiler,
GCC tries the directories thus specified when searching for special
linker files, if it can't find them using @env{GCC_EXEC_PREFIX}. Linking
using GCC also uses these directories when searching for ordinary
libraries for the @option{-l} option (but directories specified with
@option{-L} come first).

@item LANG
@findex LANG
@cindex locale definition
This variable is used to pass locale information to the compiler. One way in
which this information is used is to determine the character set to be used
when character literals, string literals and comments are parsed in C and C++.
When the compiler is configured to allow multibyte characters,
the following values for @env{LANG} are recognized:

@table @samp
@item C-JIS
Recognize JIS characters.
@item C-SJIS
Recognize SJIS characters.
@item C-EUCJP
Recognize EUCJP characters.
@end table

If @env{LANG} is not defined, or if it has some other value, then the
compiler will use mblen and mbtowc as defined by the default locale to
recognize and translate multibyte characters.
@end table

@noindent
Some additional environments variables affect the behavior of the
preprocessor.

```
@include cppenv.texi

@c man end

@node Precompiled Headers
@section Using Precompiled Headers
@cindex precompiled headers
@cindex speed of compilation

Often large projects have many header files that are included in every
source file.  The time the compiler takes to process these header files
over and over again can account for nearly all of the time required to
build the project.  To make builds faster, GCC allows users to
'precompile' a header file; then, if builds can use the precompiled
header file they will be much faster.

To create a precompiled header file, simply compile it as you would any
other file, if necessary using the @option{-x} option to make the driver
treat it as a C or C++ header file.  You will probably want to use a
tool like @command{make} to keep the precompiled header up-to-date when
the headers it contains change.

A precompiled header file will be searched for when @code{#include} is
seen in the compilation.  As it searches for the included file
(@pxref{Search Path,,Search Path,cpp,The C Preprocessor}) the
compiler looks for a precompiled header in each directory just before it
looks for the include file in that directory.  The name searched for is
the name specified in the @code{#include} with @samp{.gch} appended.  If
the precompiled header file can't be used, it is ignored.

For instance, if you have @code{#include "all.h"}, and you have
@file{all.h.gch} in the same directory as @file{all.h}, then the
precompiled header file will be used if possible, and the original
header will be used otherwise.

Alternatively, you might decide to put the precompiled header file in a
directory and use @option{-I} to ensure that directory is searched
before (or instead of) the directory containing the original header.
Then, if you want to check that the precompiled header file is always
used, you can put a file of the same name as the original header in this
directory containing an @code{#error} command.

This also works with @option{-include}.  So yet another way to use
precompiled headers, good for projects not designed with precompiled
header files in mind, is to simply take most of the header files used by
a project, include them from another header file, precompile that header
file, and @option{-include} the precompiled header.  If the header files
have guards against multiple inclusion, they will be skipped because
they've already been included (in the precompiled header).

If you need to precompile the same header file for different
languages, targets, or compiler options, you can instead make a
@emph{directory} named like @file{all.h.gch}, and put each precompiled
header in the directory, perhaps using @option{-o}.  It doesn't matter
what you call the files in the directory, every precompiled header in
the directory will be considered.  The first precompiled header
encountered in the directory that is valid for this compilation will
be used; they're searched in no particular order.

There are many other possibilities, limited only by your imagination,
good sense, and the constraints of your build system.

A precompiled header file can be used only when these conditions apply:

@itemize
```



```
@item
Only one precompiled header can be used in a particular compilation.

@item
A precompiled header can't be used once the first C token is seen.  You
can have preprocessor directives before a precompiled header; you can
even include a precompiled header from inside another header, so long as
there are no C tokens before the @code{#include}.

@item
The precompiled header file must be produced for the same language as
the current compilation.  You can't use a C precompiled header for a C++
compilation.

@item
The precompiled header file must have been produced by the same compiler
binary as the current compilation is using.

@item
Any macros defined before the precompiled header is included must
either be defined in the same way as when the precompiled header was
generated, or must not affect the precompiled header, which usually
means that they don't appear in the precompiled header at all.

The @option{-D} option is one way to define a macro before a
precompiled header is included; using a @code{#define} can also do it.
There are also some options that define macros implicitly, like
@option{-O} and @option{-Wdeprecated}; the same rule applies to macros
defined this way.

@item If debugging information is output when using the precompiled
header, using @option{-g} or similar, the same kind of debugging information
must have been output when building the precompiled header.  However,
a precompiled header built using @option{-g} can be used in a compilation
when no debugging information is being output.

@item The same @option{-m} options must generally be used when building
and using the precompiled header.  @xref{Submodel Options},
for any cases where this rule is relaxed.

@item Each of the following options must be the same when building and using
the precompiled header:

@gccoptlist{-fexceptions}

@item
Some other command-line options starting with @option{-f},
@option{-p}, or @option{-O} must be defined in the same way as when
the precompiled header was generated.  At present, it's not clear
which options are safe to change and which are not; the safest choice
is to use exactly the same options when generating and using the
precompiled header.  The following are known to be safe:

@gccoptlist{-fmessage-length=  -fpreprocessed  -fsched-interblock @gol
-fsched-spec  -fsched-spec-load  -fsched-spec-load-dangerous @gol
-fsched-verbose=<number>  -fschedule-insns  -fvisibility= @gol
-pedantic-errors}

@end itemize

For all of these except the last, the compiler will automatically
ignore the precompiled header if the conditions aren't met.  If you
find an option combination that doesn't work and doesn't cause the
precompiled header to be ignored, please consider filing a bug report,
see @ref{Bugs}.
```

If you do use differing options when generating and using the
precompiled header, the actual behavior will be a mixture of the
behavior for the options. For instance, if you use @option{-g} to
generate the precompiled header but not when using it, you may or may
not get debugging information for routines in the precompiled header.

@node Running Protoize
@section Running Protoize

The program @code{protoize} is an optional part of GCC@. You can use
it to add prototypes to a program, thus converting the program to ISO
C in one respect. The companion program @code{unprotoize} does the
reverse: it removes argument types from any prototypes that are found.

When you run these programs, you must specify a set of source files as
command line arguments. The conversion programs start out by compiling
these files to see what functions they define. The information gathered
about a file @var{foo} is saved in a file named @file{@var{foo}.X}.

After scanning comes actual conversion. The specified files are all
eligible to be converted; any files they include (whether sources or
just headers) are eligible as well.

But not all the eligible files are converted. By default,
@code{protoize} and @code{unprotoize} convert only source and header
files in the current directory. You can specify additional directories
whose files should be converted with the @option{-d @var{directory}}
option. You can also specify particular files to exclude with the
@option{-x @var{file}} option. A file is converted if it is eligible, its
directory name matches one of the specified directory names, and its
name within the directory has not been excluded.

Basic conversion with @code{protoize} consists of rewriting most
function definitions and function declarations to specify the types of
the arguments. The only ones not rewritten are those for varargs
functions.

@code{protoize} optionally inserts prototype declarations at the
beginning of the source file, to make them available for any calls that
precede the function's definition. Or it can insert prototype
declarations with block scope in the blocks where undeclared functions
are called.

Basic conversion with @code{unprotoize} consists of rewriting most
function declarations to remove any argument types, and rewriting
function definitions to the old-style pre-ISO form.

Both conversion programs print a warning for any function declaration or
definition that they can't convert. You can suppress these warnings
with @option{-q}.

The output from @code{protoize} or @code{unprotoize} replaces the
original source file. The original file is renamed to a name ending
with @samp{.save} (for DOS, the saved filename ends in @samp{.sav}
without the original @samp{.c} suffix). If the @samp{.save} (@samp{.sav}
for DOS) file already exists, then the source file is simply discarded.

@code{protoize} and @code{unprotoize} both depend on GCC itself to
scan the program and collect information about the functions it uses.
So neither of these programs will work until GCC is installed.

Here is a table of the options you can use with @code{protoize} and
@code{unprotoize}. Each option works with both programs unless
otherwise stated.

@table @code

@item -B @var{directory}
Look for the file @file{SYSCALLS.c.X} in @var{directory}, instead of the
usual directory (normally @file{/usr/local/lib}). This file contains
prototype information about standard system functions. This option
applies only to @code{protoize}.

@item -c @var{compilation-options}
Use @var{compilation-options} as the options when running @command{gcc} to
produce the @samp{.X} files. The special option @option{-aux-info} is
always passed in addition, to tell @command{gcc} to write a @samp{.X} file.

Note that the compilation options must be given as a single argument to
@code{protoize} or @code{unprotoize}. If you want to specify several
@command{gcc} options, you must quote the entire set of compilation options
to make them a single word in the shell.

There are certain @command{gcc} arguments that you cannot use, because they
would produce the wrong kind of output. These include @option{-g},
@option{-O}, @option{-c}, @option{-S}, and @option{-o} If you include these in
the @var{compilation-options}, they are ignored.

@item -C
Rename files to end in @samp{.C} (@samp{.cc} for DOS-based file
systems) instead of @samp{.c}. This is convenient if you are converting
a C program to C++. This option applies only to @code{protoize}.

@item -g
Add explicit global declarations. This means inserting explicit
declarations at the beginning of each source file for each function
that is called in the file and was not declared. These declarations
precede the first function definition that contains a call to an
undeclared function. This option applies only to @code{protoize}.

@item -i @var{string}
Indent old-style parameter declarations with the string @var{string}.
This option applies only to @code{protoize}.

@code{unprotoize} converts prototyped function definitions to old-style
function definitions, where the arguments are declared between the
argument list and the initial @samp{@{}. By default, @code{unprotoize}
uses five spaces as the indentation. If you want to indent with just
one space instead, use @option{-i " "}.

@item -k
Keep the @samp{.X} files. Normally, they are deleted after conversion
is finished.

@item -l
Add explicit local declarations. @code{protoize} with @option{-l} inserts
a prototype declaration for each function in each block which calls the
function without any declaration. This option applies only to
@code{protoize}.

@item -n
Make no real changes. This mode just prints information about the conversions
that would have been done without @option{-n}.

@item -N
Make no @samp{.save} files. The original files are simply deleted.
Use this option with caution.

@item -p @var{program}
Use the program @var{program} as the compiler. Normally, the name
@file{gcc} is used.

@item -q

```
16657 Work quietly.  Most warnings are suppressed.

16659 @item -v
16660 Print the version number, just like @option{-v} for @command{gcc}.
16661 @end table

16663 If you need special compiler options to compile one of your program's
16664 source files, then you should generate that file's @samp{.X} file
16665 specially, by running @command{gcc} on that source file with the
16666 appropriate options and the option @option{-aux-info}.  Then run
16667 @code{protoize} on the entire set of files.  @code{protoize} will use
16668 the existing @samp{.X} file because it is newer than the source file.
16669 For example:

16671 @smallexample
16672 gcc -Dfoo=bar file1.c -aux-info file1.X
16673 protoize *.c
16674 @end smallexample

16676 @noindent
16677 You need to include the special files along with the rest in the
16678 @code{protoize} command, even though their @samp{.X} files already
16679 exist, because otherwise they won't get converted.

16681 @xref{Protoize Caveats}, for more information on how to use
16682 @code{protoize} successfully.
```

****
483 Sun Oct 28 20:56:11 2012
new/gcc/testsuite/gcc.target/i386/local.c
Implement -fstrict-calling-conventions
Stock GCC is overly willing to violate the ABI when calling local functions,
such that it passes arguments in registers on i386. This hampers debugging
with anything other than a fully-aware DWARF debugger, and is generally not
something we desire.
Implement a flag which disables this behaviour, enabled by default. The flag is
global, though only effective on i386, to more easily allow its globalization
later which, given the odds, is likely to be necessary.
****

```
   1 /* { dg-do compile } */
   2 /* { dg-options "-O2 -funit-at-a-time -fno-strict-calling-conventions" { target
   3 /* { dg-options "-O2 -funit-at-a-time" { target lp64 } } */
   2 /* { dg-options "-O2 -funit-at-a-time" } */
   4 /* { dg-final { scan-assembler "magic\[^\\n\]*eax" { target ilp32 } } } */
   5 /* { dg-final { scan-assembler "magic\[^\\n\]*edi" { target lp64 } } } */

   7 /* Verify that local calling convention is used.  */
   8 static t(int) __attribute__ ((noinline));
   9 m()
  10 {
  11         t(1);
  12 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
     376 Sun Oct 28 20:56:11 2012
new/gcc/testsuite/gcc.target/i386/save-args-1.c
testsuite/save-args-1: Actually test the full thing, rather than just the bit I
**********************************************************
```

```
/* Test -msave-args */
/* { dg-do compile { target { { i?86-*-solaris2.* } && lp64 } } } */
/* { dg-options "-msave-args" } */
/* { dg-final { scan-assembler "movq\t%rsi, -32\\(%rbp\\)" } } */
/* { dg-final { scan-assembler "movq\t%rsi, -16\\(%rbp\\)" } } */
/* { dg-final { scan-assembler "movq\t%rdi, -8\\(%rbp\\)" } } */

int
foo(int argc, char **argv)
{
    return (1);
}
#endif /* ! codereview */
```

```
**********************************************************
     373 Sun Oct 28 20:56:11 2012
new/gcc/testsuite/gcc.target/i386/strict-cc.c
Implement -fstrict-calling-conventions
Stock GCC is overly willing to violate the ABI when calling local functions,
such that it passes arguments in registers on i386.  This hampers debugging
with anything other than a fully-aware DWARF debugger, and is generally not
something we desire.
Implement a flag which disables this behaviour, enabled by default.  The flag is
global, though only effective on i386, to more easily allow its globalization
later which, given the odds, is likely to be necessary.
**********************************************************
```

```
/* { dg-do compile { target { ilp32 } } } */
/* { dg-options "-O2 -funit-at-a-time -fstrict-calling-conventions"  } */
/* { dg-final { scan-assembler "pushl.*\\\$1" } } */

#include <stdio.h>

/* Verify that local calling convention is not used if strict conventions.  */
static t(int) __attribute__ ((noinline));
m()
{
        t(1);
}

static t(int a)
{
    printf("%d\n", a);
}
#endif /* ! codereview */
```

```
***********************************************************
    7746 Sun Oct 28 20:56:11 2012
new/libiberty/testsuite/test-demangle.c
libiberty/testsuite: Avoid conflicting getline()
***********************************************************
______unchanged_portion_omitted_

  42 static unsigned int lineno;

  44 /* Safely read a single line of arbitrary length from standard input.  */

  46 #define LINELEN 80

  48 static void
  49 _getline(buf)
  49 getline(buf)
  50      struct line *buf;
  51 {
  52   char *data = buf->data;
  53   size_t alloc = buf->alloced;
  54   size_t count = 0;
  55   int c;

  57   if (data == 0)
  58     {
  59       data = xmalloc (LINELEN);
  60       alloc = LINELEN;
  61     }

  63   /* Skip comment lines.  */
  64   while ((c = getchar()) == '#')
  65     {
  66       while ((c = getchar()) != EOF && c != '\n');
  67       lineno++;
  68     }

  70   /* c is the first character on the line, and it's not a comment
  71      line: copy this line into the buffer and return.  */
  72   while (c != EOF && c != '\n')
  73     {
  74       if (count + 1 >= alloc)
  75         {
  76           alloc *= 2;
  77           data = xrealloc (data, alloc);
  78         }
  79       data[count++] = c;
  80       c = getchar();
  81     }
  82   lineno++;
  83   data[count] = '\0';

  85   buf->data = data;
  86   buf->alloced = alloc;
  87 }
_____unchanged_portion_omitted_

 149 /* The tester operates on a data file consisting of groups of lines:
 150    options
 151    input to be demangled
 152    expected output

 154    Supported options:
 155      --format=<name>     Sets the demangling style.
 156      --no-params         There are two lines of expected output; the first
 157                          is with DMGL_PARAMS, the second is without it.
 158      --is-v3-ctor        Calls is_gnu_v3_mangled_ctor on input; expected
```



```
 159                          output is an integer representing ctor_kind.
 160      --is-v3-dtor        Likewise, but for dtors.
 161      --ret-postfix       Passes the DMGL_RET_POSTFIX option

 163    For compatibility, just in case it matters, the options line may be
 164    empty, to mean --format=auto.  If it doesn't start with --, then it
 165    may contain only a format name.
 166 */

 168 int
 169 main(argc, argv)
 170      int argc;
 171      char **argv;
 172 {
 173   enum demangling_styles style = auto_demangling;
 174   int no_params;
 175   int is_v3_ctor;
 176   int is_v3_dtor;
 177   int ret_postfix;
 178   struct line format;
 179   struct line input;
 180   struct line expect;
 181   char *result;
 182   int failures = 0;
 183   int tests = 0;

 185   if (argc > 1)
 186     {
 187       fprintf (stderr, "usage: %s < test-set\n", argv[0]);
 188       return 2;
 189     }

 191   format.data = 0;
 192   input.data = 0;
 193   expect.data = 0;

 195   for (;;)
 196     {
 197       const char *inp;
 198
 199       _getline (&format);
 199       getline (&format);
 200       if (feof (stdin))
 201         break;

 203       _getline (&input);
 204       _getline (&expect);
 203       getline (&input);
 204       getline (&expect);

 206       inp = protect_end (input.data);

 208       tests++;

 210       no_params = 0;
 211       ret_postfix = 0;
 212       is_v3_ctor = 0;
 213       is_v3_dtor = 0;
 214       if (format.data[0] == '\0')
 215         style = auto_demangling;
 216       else if (format.data[0] != '-')
 217         {
 218           style = cplus_demangle_name_to_style (format.data);
 219           if (style == unknown_demangling)
 220             {
 221               printf ("FAIL at line %d: unknown demangling style %s\n",
```

```
222                       lineno, format.data);
223               failures++;
224               continue;
225             }
226         }
227       else
228         {
229           char *p;
230           char *opt;

232           p = format.data;
233           while (*p != '\0')
234             {
235               char c;

237               opt = p;
238               p += strcspn (p, " \t=");
239               c = *p;
240               *p = '\0';
241               if (strcmp (opt, "--format") == 0 && c == '=')
242                 {
243                   char *fstyle;

245                   *p = c;
246                   ++p;
247                   fstyle = p;
248                   p += strcspn (p, " \t");
249                   c = *p;
250                   *p = '\0';
251                   style = cplus_demangle_name_to_style (fstyle);
252                   if (style == unknown_demangling)
253                     {
254                       printf ("FAIL at line %d: unknown demangling style %s\n",
255                               lineno, fstyle);
256                       failures++;
257                       continue;
258                     }
259                 }
260               else if (strcmp (opt, "--no-params") == 0)
261                 no_params = 1;
262               else if (strcmp (opt, "--is-v3-ctor") == 0)
263                 is_v3_ctor = 1;
264               else if (strcmp (opt, "--is-v3-dtor") == 0)
265                 is_v3_dtor = 1;
266               else if (strcmp (opt, "--ret-postfix") == 0)
267                 ret_postfix = 1;
268               else
269                 {
270                   printf ("FAIL at line %d: unrecognized option %s\n",
271                           lineno, opt);
272                   failures++;
273                   continue;
274                 }
275               *p = c;
276               p += strspn (p, " \t");
277             }
278         }

280       if (is_v3_ctor || is_v3_dtor)
281         {
282           char buf[20];

284           if (is_v3_ctor)
285             {
286               enum gnu_v3_ctor_kinds kc;
```



```
288               kc = is_gnu_v3_mangled_ctor (inp);
289               sprintf (buf, "%d", (int) kc);
290             }
291           else
292             {
293               enum gnu_v3_dtor_kinds kd;

295               kd = is_gnu_v3_mangled_dtor (inp);
296               sprintf (buf, "%d", (int) kd);
297             }

299           if (strcmp (buf, expect.data) != 0)
300             {
301               fail (lineno, format.data, input.data, buf, expect.data);
302               failures++;
303             }

305           continue;
306         }

308       cplus_demangle_set_style (style);

310       result = cplus_demangle (inp,
311                                DMGL_PARAMS|DMGL_ANSI|DMGL_TYPES
312                                |(ret_postfix ? DMGL_RET_POSTFIX : 0));

314       if (result
315           ? strcmp (result, expect.data)
316           : strcmp (input.data, expect.data))
317         {
318           fail (lineno, format.data, input.data, result, expect.data);
319           failures++;
320         }
321       free (result);

323       if (no_params)
324         {
325           _getline (&expect);
325           getline (&expect);
326           result = cplus_demangle (inp, DMGL_ANSI|DMGL_TYPES);

328           if (result
329               ? strcmp (result, expect.data)
330               : strcmp (input.data, expect.data))
331             {
332               fail (lineno, format.data, input.data, result, expect.data);
333               failures++;
334             }
335           free (result);
336         }
337     }

339   free (format.data);
340   free (input.data);
341   free (expect.data);

343   printf ("%s: %d tests, %d failures\n", argv[0], tests, failures);
344   return failures ? 1 : 0;
345 }
```

*unchanged_portion_omitted_*