

```

*****
8804 Fri Feb 22 23:58:01 2013
new/usr/src/common/saveargs/saveargs.c
saveargs: let disasm do the lifting
3544 save-args matcher could be considerably more robust
3545 save-args matcher should accept saves maybe out-of-order
Reviewed by: Joshua M. Clulow <josh@sysmgr.org>
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */
21 /*
22 * Copyright 2007 Sun Microsystems, Inc. All rights reserved.
23 * Use is subject to license terms.
24 */

27 /*
28 * The Sun Studio and GCC (patched for opensolaris/illumos) compilers
29 * implement a argument saving scheme on amd64 via the -Wu,save-args or
30 * options. When the option is specified, INTEGER type function arguments
31 * passed via registers will be saved on the stack immediately after %rbp, and
32 * will not be modified through out the life of the routine.
33 *
34 *
35 *          %rbp    -->    +-----+
36 *                          | %rbp  |
37 *          -0x8(%rbp)    +-----+
38 *                          | %rdi  |
39 *          -0x10(%rbp)   +-----+
40 *                          | %rsi  |
41 *          -0x18(%rbp)   +-----+
42 *                          | %rdx  |
43 *          -0x20(%rbp)   +-----+
44 *                          | %rcx  |
45 *          -0x28(%rbp)   +-----+
46 *                          | %r8   |
47 *          -0x30(%rbp)   +-----+
48 *                          | %r9   |
49 *                          +-----+
50 *
51 * For example, for the following function,
52 *
53 * void
54 * foo(int a1, int a2, int a3, int a4, int a5, int a6, int a7)
55 * {
56 *     ...
57 * }
58 *

```

```

59 * Disassembled code will look something like the following:
60 *
61 *     pushq    %rbp
62 *     movq    %rsp, %rbp
63 *     subq    $imm8, %rsp
64 *     movq    %rdi, -0x8(%rbp)
65 *     movq    %rsi, -0x10(%rbp)
66 *     movq    %rdx, -0x18(%rbp)
67 *     movq    %rcx, -0x20(%rbp)
68 *     movq    %r8, -0x28(%rbp)
69 *     movq    %r9, -0x30(%rbp)
70 *     ...
71 * or
72 *     pushq    %rbp
73 *     movq    %rsp, %rbp
74 *     subq    $imm8, %rsp
75 *     movq    %r9, -0x30(%rbp)
76 *     movq    %r8, -0x28(%rbp)
77 *     movq    %rcx, -0x20(%rbp)
78 *     movq    %rdx, -0x18(%rbp)
79 *     movq    %rsi, -0x10(%rbp)
80 *     movq    %rdi, -0x8(%rbp)
81 *     ...
82 * or
83 *     pushq    %rbp
84 *     movq    %rsp, %rbp
85 *     pushq    %rdi
86 *     pushq    %rsi
87 *     pushq    %rdx
88 *     pushq    %rcx
89 *     pushq    %r8
90 *     pushq    %r9
91 *
92 * **: The space being reserved is in addition to what the current
93 *     function prolog already reserves.
94 *
95 * We loop through the first SAVEARGS_INSN_SEQ_LEN bytes of the function
96 * looking for each argument saving instruction we would expect to see.
97 * looking for each argument saving instruction we would expect to see. We
98 * loop byte-by-byte, rather than doing anything smart about insn lengths,
99 * only deviating from this when we know we have our insn, and can skip the
100 * rest of it.
101 *
102 * If there are odd number of arguments to a function, additional space is
103 * reserved on the stack to maintain 16-byte alignment. For example,
104 *
105 *     argc == 0: no argument saving.
106 *     argc == 3: save 3, but space for 4 is reserved
107 *     argc == 7: save 6.
108 */

106 #include <sys/sysmacros.h>
107 #include <sys/types.h>
108 #include <libdisasm.h>
109 #include <string.h>

111 #endif /* ! codereview */
112 #include <saveargs.h>

114 /*
115 * Size of the instruction sequence arrays. It should correspond to
116 * the maximum number of arguments passed via registers.
117 */
118 #define INSTR_ARRAY_SIZE        6

120 #define INSTR1(ins, off) (ins[(off)])

```

```

121 #define INSTR2(ins, off) (ins[(off)] + (ins[(off) + 1] << 8))
122 #define INSTR3(ins, off) \
123     (ins[(off)] + (ins[(off) + 1] << 8) + (ins[(off) + 2] << 16))
124 #define INSTR4(ins, off) \
125     (ins[(off)] + (ins[(off) + 1] << 8) + (ins[(off) + 2] << 16) + \
126     (ins[(off) + 3] << 24))

128 /*
129  * Sun Studio 10 patch implementation saves %rdi first;
130  * GCC 3.4.3 Sun branch implementation saves them in reverse order.
131  */
132 static const uint32_t save_instr[INSTR_ARRAY_SIZE] = {
133     0xf87d8948, /* movq %rdi, -0x8(%rbp) */
134     0xf0758948, /* movq %rsi, -0x10(%rbp) */
135     0xe8558948, /* movq %rdx, -0x18(%rbp) */
136     0xe04d8948, /* movq %rcx, -0x20(%rbp) */
137     0xd845894c, /* movq %r8, -0x28(%rbp) */
138     0xd04d894c /* movq %r9, -0x30(%rbp) */
139 };

141 static const uint16_t save_instr_push[] = {
142     0x57, /* pushq %rdi */
143     0x56, /* pushq %rsi */
144     0x52, /* pushq %rdx */
145     0x51, /* pushq %rcx */
146     0x5041, /* pushq %r8 */
147     0x5141 /* pushq %r9 */
148 };

150 /*
151  * If the return type of a function is a structure greater than 16 bytes in
152  * size, %rdi will contain the address to which it should be stored, and
153  * arguments will begin at %rsi. Studio will push all of the normal argument
154  * registers anyway, GCC will start pushing at %rsi, so we need a separate
155  * pattern.
156  */
157 static const uint32_t save_instr_sr[INSTR_ARRAY_SIZE-1] = {
158     0xf8758948, /* movq %rsi, -0x8(%rbp) */
159     0xf0558948, /* movq %rdx, -0x10(%rbp) */
160     0xe84d8948, /* movq %rcx, -0x18(%rbp) */
161     0xe045894c, /* movq %r8, -0x20(%rbp) */
162     0xd84d894c /* movq %r9, -0x28(%rbp) */
163 };

165 static const uint8_t save_fp_pushes[] = {
166     0x55, /* pushq %rbp */
167     0xcc /* int $0x3 */
168 };
169 #define NUM_FP_PUSHES (sizeof (save_fp_pushes) / sizeof (save_fp_pushes[0]))

171 static const uint32_t save_fp_movs[] = {
172     0x00e58948, /* movq %rsp,%rbp, encoding 1 */
173     0x00ec8b48, /* movq %rsp,%rbp, encoding 2 */
174 };
175 #define NUM_FP_MOVS (sizeof (save_fp_movs) / sizeof (save_fp_movs[0]))

177 typedef struct {
178     uint8_t *data;
179     size_t size;
180 } text_t;

182 static int
183 do_read(void *data, uint64_t addr, void *buf, size_t len)
184 {
185     text_t *t = data;

```

```

187     if (addr > t->size)
188         return (-1);

190     len = MIN(len, t->size - addr);
191
192     (void) memcpy(buf, (char *)t->data + addr, len);
193
194     return (len);
195 }

197 /* ARGSUSED */
198 int
199 do_lookup(void *data, uint64_t addr, char *buf, size_t buflen, uint64_t *start,
200          size_t *symlen)
201 {
202     /* We don't actually need lookup info */
203     return (-1);
204 }

206 #endif /* ! codereview */
207 static int
208 instr_size(dis_handle_t *dhp, uint8_t *ins, unsigned int i, size_t size)
209 {
210     text_t t;

212     t.data = ins;
213     t.size = size;

215     dis_set_data(dhp, &t);
216     return (dis_instrlen(dhp, i));
217 }

219 static boolean_t
220 has_saved_fp(dis_handle_t *dhp, uint8_t *ins, int size)
221 has_saved_fp(uint8_t *ins, int size)
222 {
223     int i, j;
224     uint32_t n;
225     boolean_t found_push = B_FALSE;
226     int sz = 0;

227     for (i = 0; i < size; i += sz) {
228         if ((sz = instr_size(dhp, ins, i, size)) == -1)
229             return (B_FALSE);

231         if (found_push == B_FALSE) {
232             if (sz != 1)
233                 continue;

235             int found_push = 0;

237             for (i = 0; i < size; i++) {
238                 if (found_push == 0) {
239                     n = INSTR1(ins, i);
240                     for (j = 0; j <= NUM_FP_PUSHES; j++)
241                         if (save_fp_pushes[j] == n) {
242                             found_push = B_TRUE;
243                             found_push = 1;
244                             break;
245                         }
246                     } else {
247                         if (sz != 3)
248                             continue;

250                     n = INSTR3(ins, i);
251                     for (j = 0; j <= NUM_FP_MOVS; j++)
252                         if (save_fp_movs[j] == n)

```

```

248         return (B_TRUE);
126         return (1);
249     }
250 }

252 return (B_FALSE);
130 return (0);
253 }

255 int
256 saveargs_has_args(uint8_t *ins, size_t size, uint_t argc, int start_index)
257 {
258     int            i, j;
259     uint32_t       n;
260     uint8_t        found = 0;
261     size_t         sz = 0;
262     dis_handle_t   *dhp = NULL;
263     int            ret = SAVEARGS_NO_ARGS;
264 #endif /* ! codereview */

266     argc = MIN((start_index + argc), INSTR_ARRAY_SIZE);

268     if ((dhp = dis_handle_create(DIS_X86_SIZE64, NULL, do_lookup,
269 do_read)) == NULL)
270         return (SAVEARGS_NO_ARGS);

272     if (!has_saved_fp(dhp, ins, size)) {
273         dis_handle_destroy(dhp);
138     if (!has_saved_fp(ins, size))
274         return (SAVEARGS_NO_ARGS);
275 }
276 #endif /* ! codereview */

278 /*
279  * For each possible style of argument saving, walk the insn stream as
280  * we've been given it, and set bit N in 'found' if we find the
281  * instruction saving the Nth argument.
140  * Compare against Sun Studio implementation
282  */
142 for (i = 4, j = 0; i <= size - 4; i++) {
143     n = INSTR4(ins, i);

145     if (n == save_instr[j]) {
146         i += 3;
147         if (++j >= argc)
148             return (start_index ? SAVEARGS_STRUCT_ARGS :
149 SAVEARGS_TRAD_ARGS);
150     }
151 }

284 /*
285  * Compare against regular implementation
154  * Compare against GCC implementation
286  */
287 found = 0;
288 for (i = 0; i < size; i += sz) {
289     sz = instr_size(dhp, ins, i, size);

291     if (sz == -1)
292         break;
293     else if (sz != 4)
294         continue;

156 for (i = 4, j = argc - 1; i <= size - 4; i++) {
296     n = INSTR4(ins, i);

```

```

298         for (j = 0; j < argc; j++) {
299 #endif /* ! codereview */
300             if (n == save_instr[j]) {
301                 found |= (1 << j);

303                 if (found == ((1 << argc) - 1)) {
304                     ret = start_index ?
305                         SAVEARGS_STRUCT_ARGS :
306                         SAVEARGS_TRAD_ARGS;
307                     goto done;
308                 }

310                 break;
311             }
159             i += 3;
160             if (--j < start_index)
161                 return (SAVEARGS_TRAD_ARGS);
312         }
313     }

315 /*
316  * Compare against GCC push-based implementation
317  */
318 found = 0;
319 for (i = 0; i < size; i += sz) {
320     if ((sz = instr_size(dhp, ins, i, size)) == -1)
321         break;

323     for (j = start_index; j < argc; j++) {
324         if (sz == 2) /* Two byte */
325             n = INSTR2(ins, i);
326         else if (sz == 1)
327             n = INSTR1(ins, i);
328         else
329             continue;
168         for (i = 4, j = start_index; i <= size - 2; i += 1) {
169             n = (i >= (8 - start_index)) ? INSTR2(ins, i) : INSTR1(ins, i);

331             if (n == save_instr_push[j]) {
332                 found |= (1 << (j - start_index));

334                 if (found == ((1 << (argc - start_index)) - 1))
335                     ret = SAVEARGS_TRAD_ARGS;
336                 goto done;
337             }

339             break;
340         }
172         if (i >= (8 - start_index))
173             i += 1;
174         if (++j >= argc)
175             return (SAVEARGS_TRAD_ARGS);
341     }
342 }

344 /*
345  * Look for a GCC-style returned structure.
346  */
347 found = 0;
179 /* Look for a GCC-style returned structure */
348 if (start_index != 0) {
349     for (i = 0; i < size; i += sz) {
350         sz = instr_size(dhp, ins, i, size);

352         if (sz == -1)
353             break;

```

```
354         else if (sz != 4)
355             continue;
181         for (i = 4, j = argc - 2; i <= size - 4; i++) {
357             n = INSTR4(ins, i);
359             /* argc is inclusive of start_index, allow for that */
360             for (j = 0; j < (argc - start_index); j++) {
361 #endif /* ! codereview */
362                 if (n == save_instr_sr[j]) {
363                     found |= (1 << j);
365                     if (found ==
366                         ((1 << (argc - start_index)) - 1)) {
367                         ret = SAVEARGS_TRAD_ARGS;
368                         goto done;
369                     }
371                     break;
372                 }
184                 i += 3;
185                 if (--j >= (argc - 1))
186                     return (SAVEARGS_TRAD_ARGS);
373             }
374         }
375     }
377 done:
378     dis_handle_destroy(dhp);
379     return (ret);
191     return (SAVEARGS_NO_ARGS);
380 }
_____unchanged_portion_omitted_____
```

new/usr/src/common/saveargs/tests/README

1

347 Fri Feb 22 23:58:03 2013

new/usr/src/common/saveargs/tests/README

3544 save-args matcher could be considerably more robust

3545 save-args matcher should accept saves maybe out-of-order

Reviewed by: Joshua M. Clulow <josh@sysmgr.org>

1 testmatch:

3 A stub program that tests the saveargs matcher against a variety of
4 function prologues (assembled from data.s)

6 functional:

8 Actually test the full chunk of the (libproc) side of the code, running
9 pstack on the range of test apps.

11 **dump:**

13 Display each function in a given object we believe to have saved
14 arguments.

15 #endif /* !codereview */

new/usr/src/common/saveargs/tests/dump/Makefile

1

964 Fri Feb 22 23:58:05 2013

new/usr/src/common/saveargs/tests/dump/Makefile

saveargs: let disasm do the lifting

3544 save-args matcher could be considerably more robust

3545 save-args matcher should accept saves maybe out-of-order

Reviewed by: Joshua M. Clulow <josh@sysmgr.org>

```
1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet at
9 # http://www.illumos.org/license/CDDL.
10 #
11 #
12 #
13 # Copyright 2012, Richard Lowe.
14 #
15 #
16 include $(SRC)/Makefile.master
17 include $(SRC)/Makefile.master.64
18 #
19 .KEEP_STATE:
20 #
21 OBJECTS = dump.o saveargs.o dis_tables.o
22 PROG = dump
23 #
24 CFLAGS += -m64
25 CPPFLAGS += -I$(SRC)/common/saveargs
26 CERRWARN += -_gcc=-Wno-parentheses
27 CERRWARN += -_gcc=-Wno-uninitialized
28 #
29 LDFLAGS += -lctf -lelf
30 LDLIBS64 += -ldisasm
31 #
32 C99MODE = $(C99_ENABLE)
33 #
34 %.o: $(SRC)/common/saveargs/%.c
35     $(COMPILE.c) -o $@ $<
36 #
37 $(PROG): $(OBJECTS)
38     $(LINK.c) -o $@ $(OBJECTS) $(LDLIBS64)
39 #
40 clean:
41     $(RM) $(OBJECTS) $(PROG)
42 #
43 clobber: clean
44 #
45 all: $(PROG)
46 #
47 install: all
48 #endif /* ! codereview */
```

```

*****
3653 Fri Feb 22 23:58:06 2013
new/usr/src/common/saveargs/tests/dump/dump.c
3544 save-args matcher could be considerably more robust
3545 save-args matcher should accept saves maybe out-of-order
Reviewed by: Joshua M. Clulow <josh@sysmgr.org>
*****
1 /*
2  * This file and its contents are supplied under the terms of the
3  * Common Development and Distribution License ("CDDL"), version 1.0.
4  * You may only use this file in accordance with the terms of version
5  * 1.0 of the CDDL.
6  *
7  * A full copy of the text of the CDDL should have accompanied this
8  * source. A copy of the CDDL is also available via the Internet at
9  * http://www.illumos.org/license/CDDL.
10 */

12 #include <err.h>
13 #include <fcntl.h>
14 #include <gelf.h>
15 #include <libctf.h>
16 #include <saveargs.h>
17 #include <stdarg.h>
18 #include <stdio.h>
19 #include <stdlib.h>
20 #include <strings.h>
21 #include <unistd.h>

23 extern const char *__progname;

25 typedef struct symtab_sym {
26     GElf_Sym ss_sym;
27     char *ss_name;
28     ctf_funcinfo_t ss_finfo;
29     uint8_t *ss_data;
30     size_t ss_size;
31 } symtab_sym_t;

33 static void
34 walk_symbtab(Elf *elf, char *fname, ctf_file_t *fp,
35             void (*callback)(ctf_file_t *, symtab_sym_t *))
36 {
37     Elf_Scn *stab = NULL;
38     Elf_Scn *text = NULL;
39     Elf_Data *stabdata = NULL;
40     Elf_Data *textdata = NULL;
41     GElf_Ehdr ehdr;
42     GElf_Shdr stabshdr;
43     GElf_Shdr textshdr;
44     int fd, foundtext = 0, foundstab = 0;
45     symtab_sym_t ss;

47     if ((gelf_getehdr(elf, &ehdr)) == NULL)
48         errx(1, "could not read ELF header from %s\n",
49             fname);

51     while ((stab = elf_nextscn(elf, stab)) != NULL) {
52         (void) gelf_getshdr(stab, &stabshdr);

54         if (stabshdr.sh_type == SHT_SYMTAB) {
55             foundstab = 1;
56             break;
57         }
58     }

```

```

60     while ((text = elf_nextscn(elf, text)) != NULL) {
61         (void) gelf_getshdr(text, &textshdr);

63         if (strcmp(".text", elf_strptr(elf,
64             ehdr.e_shstrndx, (size_t)textshdr.sh_name)) == 0) {
65             foundtext = 1;
66             break;
67         }
68     }

70     if (!foundstab || !foundtext)
71         goto out;

73     stabdata = elf_getdata(stab, NULL);
74     textdata = elf_rawdata(text, NULL);
75     for (unsigned symdx = 0;
76          symdx < (stabshdr.sh_size / stabshdr.sh_entsize);
77          symdx++) {
78         (void) gelf_getsym(stabdata, symdx, &ss.ss_sym);

80         if ((GELF_ST_TYPE(ss.ss_sym.st_info) != STT_FUNC) ||
81             (ss.ss_sym.st_shndx == SHN_UNDEF))
82             continue;

84         ss.ss_name = elf_strptr(elf, stabshdr.sh_link,
85             ss.ss_sym.st_name);
86         ss.ss_data = ((uint8_t *) (textdata->d_buf)) +
87             (ss.ss_sym.st_value - textshdr.sh_addr);

89         if (ctf_func_info(fp, symdx, &ss.ss_finfo) == CTF_ERR) {
90             fprintf(stderr, "failed to get funcinfo for: %s\n",
91                 ss.ss_name);
92             continue;
93         }

95         (void) callback(fp, &ss);
96     }

98 out:
99     (void) elf_end(elf);
100    (void) close(fd);
101 }

103 void
104 check_sym(ctf_file_t *ctfp, symtab_sym_t *ss)
105 {
106     int rettype = ctf_type_kind(ctfp, ss->ss_finfo.ctc_return);
107     int start_index = 0;

109     if (ss->ss_finfo.ctc_argc == 0) /* No arguments, no point */
110         return;

112     if (((rettype == CTF_K_STRUCT) || (rettype == CTF_K_UNION)) &&
113         ctf_type_size(ctfp, ss->ss_finfo.ctc_return) > 16)
114         start_index = 1;

116     if (saveargs_has_args(ss->ss_data, ss->ss_sym.st_size,
117         ss->ss_finfo.ctc_argc, start_index) != SAVEARGS_NO_ARGS)
118         printf("%s has %d saved args\n", ss->ss_name,
119             ss->ss_finfo.ctc_argc);
120 }

122 int
123 main(int argc, char **argv)
124 {
125     Elf *elf;

```

```
126     ctf_file_t      *ctfp;
127     int errp, fd;
128
129     if (ctf_version(CTF_VERSION) == -1)
130         errx(1, "mismatched libctf versions\n");
131
132     if (elf_version(EV_CURRENT) == EV_NONE)
133         errx(1, "mismatched libelf versions\n");
134
135     if (argc != 2)
136         errx(2, "usage: %s <file>\n", __progname);
137
138     if ((ctfp = ctf_open(argv[1], &errp)) == NULL)
139         errx(1, "failed to ctf_open file: %s: %s\n", argv[1],
140             ctf_errmsg(errp));
141
142     if ((fd = open(argv[1], O_RDONLY)) == -1)
143         errx(1, "could not open %s\n", argv[1]);
144
145     if ((elf = elf_begin(fd, ELF_C_READ, NULL)) == NULL)
146         errx(1, "could not interpret ELF from %s\n",
147             argv[1]);
148
149     walk_syntab(elf, argv[1], ctfp, check_sym);
150
151     return (0);
152 }
153 #endif /* ! codereview */
```


new/usr/src/common/saveargs/tests/testmatch/Makefile

1

941 Fri Feb 22 23:58:06 2013

new/usr/src/common/saveargs/tests/testmatch/Makefile

saveargs: let disasm do the lifting

3544 save-args matcher could be considerably more robust

3545 save-args matcher should accept saves maybe out-of-order

Reviewed by: Joshua M. Clulow <josh@sysmgr.org>

```
2 #endif /* ! codereview */
3 #
4 # This file and its contents are supplied under the terms of the
5 # Common Development and Distribution License ("CDDL"), version 1.0.
6 # You may only use this file in accordance with the terms of version
7 # 1.0 of the CDDL.
8 #
9 # A full copy of the text of the CDDL should have accompanied this
10 # source. A copy of the CDDL is also available via the Internet at
11 # http://www.illumos.org/license/CDDL.
12 #
13 #
14 #
15 # Copyright 2012, Richard Lowe.
16 #
17 #
18 include $(SRC)/Makefile.master
19 include $(SRC)/Makefile.master.64
20 #
21 .KEEP_STATE:
22 #
23 OBJECTS = testmatch.o saveargs.o data.o
24 PROG = testmatch
25 #
26 LDLIBS64 += -ldisasm
27 #
28 #endif /* ! codereview */
29 CPPFLAGS += -I$(SRC)/common/saveargs
30 ASFLAGS += -P
31 AS_CPPFLAGS += -D_ASM
32 CERRWARN += -_gcc=-Wno-parentheses
33 CERRWARN += -_gcc=-Wno-uninitialized
34 #endif /* ! codereview */
35 #
36 %.o: $(SRC)/common/saveargs/%.c
37     $(COMPILE.c) -o $@ $<
38 #
39 $(PROG): $(OBJECTS)
40     $(LINK.c) -o $@ $(OBJECTS) $(LDLIBS64)
41     $(LINK.c) -o $@ $(OBJECTS) -lc
42 #
43 clean:
44     $(RM) $(OBJECTS) $(PROG)
45 #
46 clobber: clean
47 #
48 all: $(PROG)
49 #
50 install: all
```

10515 Fri Feb 22 23:58:07 2013

new/usr/src/common/saveargs/tests/testmatch/data.s

3544 save-args matcher could be considerably more robust

3545 save-args matcher should accept saves maybe out-of-order

Reviewed by: Joshua M. Clulow <josh@sysmgr.org>

unchanged_portion_omitted_

```

67 FUNC(gcc_mov_noorder)
68 pushq   %rbp
69 movq    %rsp, %rbp
70 movq    %rcx, -0x20(%rbp)
71 movq    %rbx, -0x28(%rbp)
72 movq    %rdi, -0x8(%rbp)
73 movq    %rdx, -0x18(%rbp)
74 movq    %rsi, -0x10(%rbp)
75 subq    $0x50, %rsp
76 SET_SIZE(gcc_mov_noorder, gcc_mov_noorder_end)
77
78 #endif /* ! codereview */
79 FUNC(gcc_mov_big_struct_ret)
80 pushq   %rbp
81 movq    %rsp, %rbp
82 movq    %rbx, -0x28(%rbp)
83 movq    %r8, -0x20(%rbp)
84 movq    %rcx, -0x18(%rbp)
85 movq    %rdx, -0x10(%rbp)
86 movq    %rsi, -0x8(%rbp)
87 subq    $0x50, %rsp
88 SET_SIZE(gcc_mov_big_struct_ret, gcc_mov_big_struct_ret_end)

90 FUNC(gcc_mov_struct_noorder)
91 pushq   %rbp
92 movq    %rsp, %rbp
93 movq    %rcx, -0x18(%rbp)
94 movq    %r8, -0x20(%rbp)
95 movq    %rsi, -0x8(%rbp)
96 movq    %rdx, -0x10(%rbp)
97 movq    %rbx, -0x28(%rbp)
98 subq    $0x50, %rsp
99 SET_SIZE(gcc_mov_struct_noorder, gcc_mov_struct_noorder_end)

101 #endif /* ! codereview */
102 FUNC(gcc_mov_big_struct_ret_and_spill)
103 pushq   %rbp
104 movq    %rsp, %rbp
105 movq    %rbx, -0x38(%rbp)
106 movq    %r9, -0x28(%rbp)
107 movq    %r8, -0x20(%rbp)
108 movq    %rcx, -0x18(%rbp)
109 movq    %rdx, -0x10(%rbp)
110 movq    %rsi, -0x8(%rbp)
111 subq    $0x90, %rsp
112 SET_SIZE(gcc_mov_big_struct_ret_and_spill, gcc_mov_big_struct_ret_and_spill_end)

114 FUNC(gcc_mov_small_struct_ret)
115 pushq   %rbp
116 movq    %rsp, %rbp
117 movq    %rbx, -0x28(%rbp)
118 movq    %rcx, -0x20(%rbp)
119 movq    %rdx, -0x18(%rbp)
120 movq    %rsi, -0x10(%rbp)
121 movq    %rdi, -0x8(%rbp)
122 subq    $0x50, %rsp
123 SET_SIZE(gcc_mov_small_struct_ret, gcc_mov_small_struct_ret_end)

```

```

125 FUNC(gcc_mov_small_struct_ret_and_spill)
126 pushq   %rbp
127 movq    %rsp, %rbp
128 movq    %rbx, -0x38(%rbp)
129 movq    %r9, -0x30(%rbp)
130 movq    %r8, -0x28(%rbp)
131 movq    %rcx, -0x20(%rbp)
132 movq    %rdx, -0x18(%rbp)
133 movq    %rsi, -0x10(%rbp)
134 movq    %rdi, -0x8(%rbp)
135 subq    $0x90, %rsp
136 SET_SIZE(gcc_mov_small_struct_ret_and_spill, gcc_mov_small_struct_ret_and_spill_end)

138 FUNC(gcc_mov_stack_spill)
139 pushq   %rbp
140 movq    %rsp, %rbp
141 movq    %rbx, -0x38(%rbp)
142 movq    %r9, -0x30(%rbp)
143 movq    %r8, -0x28(%rbp)
144 movq    %rcx, -0x20(%rbp)
145 movq    %rdx, -0x18(%rbp)
146 movq    %rsi, -0x10(%rbp)
147 movq    %rdi, -0x8(%rbp)
148 subq    $0x90, %rsp
149 SET_SIZE(gcc_mov_stack_spill, gcc_mov_stack_spill_end)

151 FUNC(gcc_push_align)
152 pushq   %rbp
153 movq    %rsp, %rbp
154 pushq   %rdi
155 pushq   %rsi
156 pushq   %rdx
157 pushq   %rcx
158 pushq   %r8
159 subq    $0x8, %rsp
160 subq    $0x30, %rsp
161 SET_SIZE(gcc_push_align, gcc_push_align_end)

163 FUNC(gcc_push_basic)
164 pushq   %rbp
165 movq    %rsp, %rbp
166 pushq   %rdi
167 pushq   %rsi
168 pushq   %rdx
169 pushq   %rcx
170 subq    $0x20, %rsp
171 SET_SIZE(gcc_push_basic, gcc_push_basic_end)

173 FUNC(gcc_push_noorder)
174 pushq   %rbp
175 movq    %rsp, %rbp
176 pushq   %rsi
177 pushq   %rdi
178 pushq   %rcx
179 pushq   %rdx
180 subq    $0x20, %rsp
181 SET_SIZE(gcc_push_noorder, gcc_push_noorder_end)

183 #endif /* ! codereview */
184 FUNC(gcc_push_big_struct_ret)
185 pushq   %rbp
186 movq    %rsp, %rbp
187 pushq   %rsi
188 pushq   %rdx
189 pushq   %rcx

```

```

190 pushq  %r8
191 subq   $0x30,%rsp
192 SET_SIZE(gcc_push_big_struct_ret, gcc_push_big_struct_ret_end)

194 FUNC(gcc_push_struct_noorder)
195 pushq  %rbp
196 movq  %rsp,%rbp
197 pushq  %rdx
198 pushq  %rsi
199 pushq  %r8
200 pushq  %rcx
201 subq   $0x30,%rsp
202 SET_SIZE(gcc_push_struct_noorder, gcc_push_struct_noorder_end)
203
204 #endif /* ! codereview */
205 FUNC(gcc_push_big_struct_ret_and_spill)
206 pushq  %rbp
207 movq  %rsp,%rbp
208 pushq  %rsi
209 pushq  %rdx
210 pushq  %rcx
211 pushq  %r8
212 pushq  %r9
213 subq   $0x8,%rsp
214 subq   $0x50,%rsp
215 SET_SIZE(gcc_push_big_struct_ret_and_spill, gcc_push_big_struct_ret_and_spill_en

217 FUNC(gcc_push_small_struct_ret)
218 pushq  %rbp
219 movq  %rsp,%rbp
220 pushq  %rdi
221 pushq  %rsi
222 pushq  %rdx
223 pushq  %rcx
224 subq   $0x20,%rsp
225 SET_SIZE(gcc_push_small_struct_ret, gcc_push_small_struct_ret_end)

227 FUNC(gcc_push_small_struct_ret_and_spill)
228 pushq  %rbp
229 movq  %rsp,%rbp
230 pushq  %rdi
231 pushq  %rsi
232 pushq  %rdx
233 pushq  %rcx
234 pushq  %r8
235 pushq  %r9
236 subq   $0x50,%rsp
237 SET_SIZE(gcc_push_small_struct_ret_and_spill, gcc_push_small_struct_ret_and_spil

239 FUNC(gcc_push_stack_spill)
240 pushq  %rbp
241 movq  %rsp,%rbp
242 pushq  %rdi
243 pushq  %rsi
244 pushq  %rdx
245 pushq  %rcx
246 pushq  %r8
247 pushq  %r9
248 subq   $0x50,%rsp
249 SET_SIZE(gcc_push_stack_spill, gcc_push_stack_spill_end)

251 FUNC(ss_mov_align)
252 pushq  %rbp
253 movq  %rsp,%rbp
254 subq   $0x30,%rsp
255 movq  %rdi,-0x8(%rbp)

```

```

256 movq  %rsi,-0x10(%rbp)
257 movq  %rdx,-0x18(%rbp)
258 movq  %rcx,-0x20(%rbp)
259 movq  %r8,-0x28(%rbp)
260 SET_SIZE(ss_mov_align, ss_mov_align_end)

262 FUNC(ss_mov_basic)
263 pushq  %rbp
264 movq  %rsp,%rbp
265 subq  $0x20,%rsp
266 movq  %rdi,-0x8(%rbp)
267 movq  %rsi,-0x10(%rbp)
268 movq  %rdx,-0x18(%rbp)
269 movq  %rcx,-0x20(%rbp)
270 SET_SIZE(ss_mov_basic, ss_mov_basic_end)

272 FUNC(ss_mov_big_struct_ret)
273 pushq  %rbp
274 movq  %rsp,%rbp
275 subq  $0x30,%rsp
276 movq  %rdi,-0x8(%rbp)
277 movq  %rsi,-0x10(%rbp)
278 movq  %rdx,-0x18(%rbp)
279 movq  %rcx,-0x20(%rbp)
280 movq  %r8,-0x28(%rbp)
281 SET_SIZE(ss_mov_big_struct_ret, ss_mov_big_struct_ret_end)

283 FUNC(ss_mov_big_struct_ret_and_spill)
284 pushq  %rbp
285 movq  %rsp,%rbp
286 subq  $0x50,%rsp
287 movq  %rdi,-0x8(%rbp)
288 movq  %rsi,-0x10(%rbp)
289 movq  %rdx,-0x18(%rbp)
290 movq  %rcx,-0x20(%rbp)
291 movq  %r8,-0x28(%rbp)
292 movq  %r9,-0x30(%rbp)
293 SET_SIZE(ss_mov_big_struct_ret_and_spill, ss_mov_big_struct_ret_and_spill_end)

295 FUNC(ss_mov_small_struct_ret)
296 pushq  %rbp
297 movq  %rsp,%rbp
298 subq  $0x20,%rsp
299 movq  %rdi,-0x8(%rbp)
300 movq  %rsi,-0x10(%rbp)
301 movq  %rdx,-0x18(%rbp)
302 movq  %rcx,-0x20(%rbp)
303 SET_SIZE(ss_mov_small_struct_ret, ss_mov_small_struct_ret_end)

305 FUNC(ss_mov_small_struct_ret_and_spill)
306 pushq  %rbp
307 movq  %rsp,%rbp
308 subq  $0x50,%rsp
309 movq  %rdi,-0x8(%rbp)
310 movq  %rsi,-0x10(%rbp)
311 movq  %rdx,-0x18(%rbp)
312 movq  %rcx,-0x20(%rbp)
313 movq  %r8,-0x28(%rbp)
314 movq  %r9,-0x30(%rbp)
315 SET_SIZE(ss_mov_small_struct_ret_and_spill, ss_mov_small_struct_ret_and_spill_en

317 FUNC(ss_mov_stack_spill)
318 pushq  %rbp
319 movq  %rsp,%rbp
320 subq  $0x50,%rsp
321 movq  %rdi,-0x8(%rbp)

```

```

322 movq    %rsi,-0x10(%rbp)
323 movq    %rdx,-0x18(%rbp)
324 movq    %rcx,-0x20(%rbp)
325 movq    %r8,-0x28(%rbp)
326 movq    %r9,-0x30(%rbp)
327 SET_SIZE(ss_mov_stack_spill, ss_mov_stack_spill_end)

329 /* DTrace instrumentation */
330 FUNC(dtrace_instrumented)
331 int     $0x3
332 movq    %rsp,%rbp
333 movq    %rbx,-0x28(%rbp)
334 movq    %rcx,-0x20(%rbp)
335 movq    %rdx,-0x18(%rbp)
336 movq    %rsi,-0x10(%rbp)
337 movq    %rdi,-0x8(%rbp)
338 subq    $0x50,%rsp
339 SET_SIZE(dtrace_instrumented, dtrace_instrumented_end)

341 /*
342  * System functions with special characteristics, be they non-initial FP save,
343  * gaps between FP save and argument saving, or gaps between saved arguments.
344  */
345 FUNC(kmem_alloc)
346 leaq    -0x1(%rdi),%rax
347 pushq   %rbp
348 movq    %rax,%rdx
349 movq    %rsp,%rbp
350 subq    $0x30,%rsp
351 shrq    $0x3,%rdx
352 movq    %r12,-0x28(%rbp)
353 movq    %rbx,-0x30(%rbp)
354 cmpq    $0x1ff,%rdx
355 movq    %r13,-0x20(%rbp)
356 movq    %r14,-0x18(%rbp)
357 movq    %rsi,-0x10(%rbp)
358 movq    %rdi,-0x8(%rbp)
359 movq    %rdi,%r12
360 SET_SIZE(kmem_alloc, kmem_alloc_end)

362 FUNC(uts_kill)
363 pushq   %rbp
364 movq    %rsp,%rbp
365 subq    $0x50,%rsp
366 movq    %rbx,-0x28(%rbp)
367 leaq    -0x50(%rbp),%rbx
368 movq    %r12,-0x20(%rbp)
369 movq    %r13,-0x18(%rbp)
370 movq    %rsi,-0x10(%rbp)
371 movl    %edi,%r12d
372 movq    %rdi,-0x8(%rbp)
373 SET_SIZE(uts_kill, uts_kill_end)

375 FUNC(avl394_ic_bitreverse)
376 movq    %rdi,%rdx
377 movq    $0x5555555555555555,%rax
378 movq    $0x3333333333333333,%rcx
379 shrq    $0x1,%rdx
380 pushq   %rbp
381 andq    %rax,%rdx
382 andq    %rdi,%rax
383 addq    %rax,%rax
384 movq    %rsp,%rbp
385 subq    $0x10,%rsp
386 orq     %rdx,%rax
387 movq    %rdi,-0x8(%rbp)

```

```

388 SET_SIZE(avl394_ic_bitreverse, avl394_ic_bitreverse_end)

390 /* Problematic functions which should not match */

392 FUNC(no_fp) /* No frame pointer */
393 movq    %rdi,%rsi
394 movq    %rsi,%rdi
395 movq    %rbx,-0x28(%rbp)
396 movq    %rcx,-0x20(%rbp)
397 movq    %rdx,-0x18(%rbp)
398 movq    %rsi,-0x10(%rbp)
399 movq    %rdi,-0x8(%rbp)
400 subq    $0x50,%rsp
401 SET_SIZE(no_fp, no_fp_end)

403 /* Small structure return, but with an SSE type (thus forcing it to the stack) *
404 FUNC(small_struct_ret_w_float)
405 pushq   %rbp
406 movq    %rsp,%rbp
407 movq    %rdi,-0x8(%rbp)
408 subq    $0x30,%rsp
409 SET_SIZE(small_struct_ret_w_float, small_struct_ret_w_float_end)

411 /* Big structure return, but with an SSE type */
412 FUNC(big_struct_ret_w_float)
413 pushq   %rbp
414 movq    %rsp,%rbp
415 movq    %rsi,-0x8(%rbp)
416 subq    $0x50,%rsp
417 movq    %rsi,-0x48(%rbp)
418 movq    -0x48(%rbp),%rax
419 movq    %rax,%rsi
420 movl    $0x400f60,%edi
421 movl    $0x0,%eax
422 movl    $0x1770,%edi
423 movl    $0x0,%eax
424 leave
425 ret
426 SET_SIZE(big_struct_ret_w_float, big_struct_ret_w_float_end)

428 FUNC(big_struct_arg_by_value)
429 pushq   %rbp
430 movq    %rsp,%rbp
431 movq    %rdi,-0x8(%rbp)
432 subq    $0x40,%rsp
433 SET_SIZE(big_struct_arg_by_value, big_struct_arg_by_value_end)

435 FUNC(small_struct_arg_by_value)
436 pushq   %rbp
437 movq    %rsp,%rbp
438 movq    %rdx,-0x18(%rbp)
439 movq    %rsi,-0x10(%rbp)
440 movq    %rdi,-0x8(%rbp)
441 subq    $0x50,%rsp
442 SET_SIZE(small_struct_arg_by_value, small_struct_arg_by_value_end)

444 FUNC(interleaved_argument_saves)
445 pushq   %rbp
446 movq    %rdi,%rax
447 shlq    $0x21,%rax
448 movq    %rsp,%rbp
449 shrq    $0x29,%rax
450 subq    $0x30,%rsp
451 movq    %rdi,-0x8(%rbp)
452 movq    %rbx,-0x28(%rbp)
453 movzbl  %dil,%edi

```

```
454 movq    %rcx,-0x20(%rbp)
455 movq    %rdx,-0x18(%rbp)
456 movq    %rsi,-0x10(%rbp)
457 movq    0x0(,%rax,8),%rax
458 SET_SIZE(interleaved_argument_saves, interleaved_argument_saves_end)
459 #endif /* ! codereview */
```

```

*****
4860 Fri Feb 22 23:58:09 2013
new/usr/src/common/saveargs/tests/testmatch/testmatch.c
3544 save-args matcher could be considerably more robust
3545 save-args matcher should accept saves maybe out-of-order
Reviewed by: Joshua M. Clulow <josh@sysmgr.org>
*****
1 /*
2  * This file and its contents are supplied under the terms of the
3  * Common Development and Distribution License ("CDDL"), version 1.0.
4  * You may only use this file in accordance with the terms of version
5  * 1.0 of the CDDL.
6  *
7  * A full copy of the text of the CDDL should have accompanied this
8  * source. A copy of the CDDL is also available via the Internet at
9  * http://www.illumos.org/license/CDDL.
10 */

12 /*
13  * Copyright 2012, Richard Lowe.
14 */

16 #include <stdio.h>
17 #include <sys/types.h>
18 #include <saveargs.h>

20 #define DEF_TEST(name) \
21     extern uint8_t name[]; \
22     extern int name##_end

24 #define SIZE_OF(name) ((caddr_t)&name##_end - (caddr_t)&name)

26 #define TEST_GOOD(name, argc) \
27     if (saveargs_has_args(name, SIZE_OF(name), argc, 0) == \
28         SAVEARGS_TRAD_ARGS) \
29     if (saveargs_has_args(name, SIZE_OF(name), argc, 0) != 0) \
30         printf("Pass: %s\n", #name); \
31     else \
32         printf("FAIL: %s\n", #name);

33 #define TEST_GOOD_STRUCT(name, argc) \
34     if (saveargs_has_args(name, SIZE_OF(name), argc, 1) == \
35         SAVEARGS_STRUCT_ARGS) \
36     printf("Pass: %s\n", #name); \
37     else \
38     printf("FAIL: %s\n", #name);

40 /*
41  * GCC deals with structures differently, so TRAD args is actually correct for
42  * this
43  */
44 #define TEST_GOOD_GSTRUCT(name, argc) \
45     if (saveargs_has_args(name, SIZE_OF(name), argc, 1) == \
46         SAVEARGS_TRAD_ARGS) \
47     if (saveargs_has_args(name, SIZE_OF(name), argc, 1) != 0) \
48         printf("Pass: %s\n", #name); \
49     else \
50     printf("FAIL: %s\n", #name);

51 #define TEST_BAD(name, argc) \
52     if (saveargs_has_args(name, SIZE_OF(name), argc, 0) == \
53         SAVEARGS_NO_ARGS) \
54     if (saveargs_has_args(name, SIZE_OF(name), argc, 0) == 0) \
55         printf("Pass: %s\n", #name); \
56     else \
57     printf("FAIL: %s\n", #name);

```

```

58 #define TEST_BAD_STRUCT(name, argc) \
59     if (saveargs_has_args(name, SIZE_OF(name), argc, 1) == \
60         SAVEARGS_NO_ARGS) \
61     printf("Pass: %s\n", #name); \
62     else \
63     printf("FAIL: %s\n", #name);

65 #define TEST_BAD_GSTRUCT(name, argc) \
66     if (saveargs_has_args(name, SIZE_OF(name), argc, 1) == \
67         SAVEARGS_NO_ARGS) \
68     if (saveargs_has_args(name, SIZE_OF(name), argc, 1) == 0) \
69         printf("Pass: %s\n", #name); \
70     else \
71     printf("FAIL: %s\n", #name);

72 DEF_TEST(gcc_mov_align);
73 DEF_TEST(gcc_mov_basic);
74 DEF_TEST(gcc_mov_noorder);
75 DEF_TEST(gcc_mov_struct_noorder);
76 #endif /* ! codereview */
77 DEF_TEST(gcc_mov_big_struct_ret);
78 DEF_TEST(gcc_mov_big_struct_ret_and_spill);
79 DEF_TEST(gcc_mov_small_struct_ret);
80 DEF_TEST(gcc_mov_small_struct_ret_and_spill);
81 DEF_TEST(gcc_mov_stack_spill);

83 DEF_TEST(gcc_push_align);
84 DEF_TEST(gcc_push_basic);
85 DEF_TEST(gcc_push_noorder);
86 DEF_TEST(gcc_push_struct_noorder);
87 #endif /* ! codereview */
88 DEF_TEST(gcc_push_big_struct_ret);
89 DEF_TEST(gcc_push_big_struct_ret_and_spill);
90 DEF_TEST(gcc_push_small_struct_ret);
91 DEF_TEST(gcc_push_small_struct_ret_and_spill);
92 DEF_TEST(gcc_push_stack_spill);

94 DEF_TEST(ss_mov_align);
95 DEF_TEST(ss_mov_basic);
96 DEF_TEST(ss_mov_big_struct_ret);
97 DEF_TEST(ss_mov_big_struct_ret_and_spill);
98 DEF_TEST(ss_mov_small_struct_ret);
99 DEF_TEST(ss_mov_small_struct_ret_and_spill);
100 DEF_TEST(ss_mov_stack_spill);

102 DEF_TEST(dtrace_instrumented);
103 DEF_TEST(kmem_alloc);
104 DEF_TEST(uts_kill);
105 DEF_TEST(avl1394_ic_bitreverse);

107 DEF_TEST(small_struct_ret_w_float);
108 DEF_TEST(big_struct_ret_w_float);

110 DEF_TEST(interleaved_argument_saves);

112 #endif /* ! codereview */
113 /*
114  * Functions which should not match
115  *
116  * no_fp -- valid save-args sequence with no saved FP
117  * big_struct_arg_by_value -- function with big struct passed by value
118  * small_struct_arg_by_value -- function with small struct passed by value
119  */
120 DEF_TEST(no_fp);
121 DEF_TEST(big_struct_arg_by_value);

```

```
122 DEF_TEST(small_struct_arg_by_value);

124 int
125 main(int argc, char **argv)
126 {
127     TEST_GOOD(kmem_alloc, 2);
128     TEST_GOOD(uts_kill, 2);
129     TEST_GOOD(av1394_ic_bitreverse, 1);
130     TEST_GOOD(dtrace_instrumented, 4);
131     TEST_GOOD_GSTRUCT(big_struct_ret_w_float, 1);
132     TEST_GOOD_STRUCT(big_struct_ret_w_float, 1);
133     TEST_BAD(no_fp, 5);

134     TEST_GOOD(gcc_mov_align, 5);
135     TEST_GOOD(gcc_push_align, 5);
136     TEST_GOOD(ss_mov_align, 5);

138     TEST_GOOD(gcc_mov_basic, 4);
139     TEST_GOOD(gcc_push_basic, 4);
140     TEST_GOOD(ss_mov_basic, 4);

142     TEST_GOOD(gcc_mov_noorder, 4);
143     TEST_GOOD(gcc_push_noorder, 4);

145     TEST_GOOD_GSTRUCT(gcc_mov_big_struct_ret, 4);
146     TEST_GOOD_GSTRUCT(gcc_push_big_struct_ret, 4);
147     TEST_GOOD_STRUCT(gcc_mov_big_struct_ret, 4);
148     TEST_GOOD_STRUCT(gcc_push_big_struct_ret, 4);
149     TEST_GOOD_GSTRUCT(ss_mov_big_struct_ret, 4);

149     TEST_GOOD_GSTRUCT(gcc_mov_struct_noorder, 4);
150     TEST_GOOD_GSTRUCT(gcc_push_struct_noorder, 4);

152     TEST_GOOD_GSTRUCT(gcc_mov_big_struct_ret_and_spill, 8);
153     TEST_GOOD_GSTRUCT(gcc_push_big_struct_ret_and_spill, 8);
154     TEST_GOOD_STRUCT(gcc_mov_big_struct_ret_and_spill, 8);
155     TEST_GOOD_STRUCT(gcc_push_big_struct_ret_and_spill, 8);
156     TEST_GOOD_STRUCT(ss_mov_big_struct_ret_and_spill, 8);

156     TEST_GOOD(gcc_mov_small_struct_ret, 4);
157     TEST_GOOD(gcc_push_small_struct_ret, 4);
158     TEST_GOOD(ss_mov_small_struct_ret, 4);

160     TEST_GOOD(gcc_mov_small_struct_ret_and_spill, 8);
161     TEST_GOOD(gcc_push_small_struct_ret_and_spill, 8);
162     TEST_GOOD(ss_mov_small_struct_ret_and_spill, 8);

164     TEST_GOOD(gcc_mov_stack_spill, 8);
165     TEST_GOOD(gcc_push_stack_spill, 8);
166     TEST_GOOD(ss_mov_stack_spill, 8);

168     TEST_BAD(big_struct_arg_by_value, 2);
169     TEST_BAD(small_struct_arg_by_value, 2);

171     TEST_BAD(small_struct_ret_w_float, 1);

173     TEST_GOOD(interleaved_argument_saves, 4);

175 #endif /* ! codereview */
176     return (0);
177 }
```

new/usr/src/lib/libdisasm/common/libdisasm.h

1

```
*****
2389 Fri Feb 22 23:58:10 2013
new/usr/src/lib/libdisasm/common/libdisasm.h
saveargs: let disasm do the lifting
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

22 /*
23  * Copyright 2007 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25  */

27 #ifndef _LIBDISASM_H
28 #define _LIBDISASM_H

30 #pragma ident      "%Z%M% %I%      %E% SMI"

30 #include <sys/types.h>

32 #ifdef __cplusplus
33 extern "C" {
34 #endif

36 typedef struct dis_handle dis_handle_t;

38 #define DIS_DEFAULT          0x0

40 /* SPARC disassembler flags */
41 #define DIS_SPARC_V8         0x01
42 #define DIS_SPARC_V9         0x02
43 #define DIS_SPARC_V9_SGI     0x04
44 #define DIS_SPARC_V9_OPL     0x08

46 /* x86 disassembler flags (mutually exclusive) */
47 #define DIS_X86_SIZE16       0x08
48 #define DIS_X86_SIZE32       0x10
49 #define DIS_X86_SIZE64       0x20

51 /* generic disassembler flags */
52 #define DIS_OCTAL             0x40
53 #define DIS_NOIMMSYM         0x80

55 typedef int (*dis_lookup_f)(void *, uint64_t, char *, size_t, uint64_t *,
56     size_t *);
57 typedef int (*dis_read_f)(void *, uint64_t, void *, size_t);

59 extern dis_handle_t *dis_handle_create(int, void *, dis_lookup_f, dis_read_f);
```

new/usr/src/lib/libdisasm/common/libdisasm.h

2

```
60 extern void dis_handle_destroy(dis_handle_t *);

62 extern int dis_disassemble(dis_handle_t *, uint64_t, char *, size_t);
63 extern uint64_t dis_previnstr(dis_handle_t *, uint64_t, int n);
64 extern void dis_set_data(dis_handle_t *, void *);
65 extern void dis_flags_set(dis_handle_t *, int f);
66 extern void dis_flags_clear(dis_handle_t *, int f);
67 extern int dis_max_instrlen(dis_handle_t *);
68 extern int dis_instrlen(dis_handle_t *, uint64_t);
69 #endif /* ! codereview */

71 /* libdisasm errors */
72 #define E_DIS_NOMEM          1          /* Out of memory */
73 #define E_DIS_INVALIDFLAG    2          /* Invalid flag for this architecture */

75 extern int dis_errno(void);
76 extern const char *dis_strerror(int);

78 #ifdef __cplusplus
79 }
80 #endif

82 #endif /* _LIBDISASM_H */
```


new/usr/src/lib/libdisasm/common/mapfile-vers

1

1507 Fri Feb 22 23:58:10 2013

new/usr/src/lib/libdisasm/common/mapfile-vers

saveargs: let disasm do the lifting

```
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 # Copyright (c) 2006, 2010, Oracle and/or its affiliates. All rights reserved.
23 #
24 #
25 #
26 # MAPFILE HEADER START
27 #
28 # WARNING: STOP NOW. DO NOT MODIFY THIS FILE.
29 # Object versioning must comply with the rules detailed in
30 #
31 #     usr/src/lib/README.mapfiles
32 #
33 # You should not be making modifications here until you've read the most current
34 # copy of that file. If you need help, contact a gatekeeper for guidance.
35 #
36 # MAPFILE HEADER END
37 #
38 #
39 $mapfile_version 2
40 #
41 SYMBOL_VERSION SUNWprivate_1.1 {
42     global:
43         dis_disassemble;
44         dis_errno;
45         dis_handle_create;
46         dis_handle_destroy;
47         dis_instrlen;
48 #endif /* ! codereview */
49         dis_max_instrlen;
50         dis_previnstr;
51         dis_set_data;
52         dis_flags_set;
53         dis_flags_clear;
54         dis_strerror;
55     local:
56         *;
57 };
```

```

*****
5644 Fri Feb 22 23:58:11 2013
new/usr/src/lib/libdisasm/i386/dis_i386.c
saveargs: let disasm do the lifting
*****

```

```

1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

22 /*
23  * Copyright 2007 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25  */

```

```
27 #pragma ident "%Z%M% %I% %E% SMI"
```

```
27 #include <libdisasm.h>
28 #include <stdlib.h>
29 #include <stdio.h>

```

```
31 #include "dis_tables.h"
32 #include "libdisasm_impl.h"

```

```
34 struct dis_handle {
35     void          *dh_data;
36     int           dh_flags;
37     dis_lookup_f  dh_lookup;
38     dis_read_f    dh_read;
39     int           dh_mode;
40     dis86_t       dh_dis;
41     uint64_t      dh_addr;
42     uint64_t      dh_end;
43 };

```

```
unchanged_portion_omitted
```

```
247 int
248 dis_instrlen(dis_handle_t *dhp, uint64_t pc)
249 {
250     if (dis_disassemble(dhp, pc, NULL, 0) != 0)
251         return (-1);

253     return (dhp->dh_addr - pc);
254 }
255 #endif /* ! codereview */

```

```

*****
      8400 Fri Feb 22 23:58:11 2013
new/usr/src/lib/libdisasm/sparc/dis_sparc.c
saveargs: let disasm do the lifting
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

22 /*
23  * Copyright 2007 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25 */

27 /*
28  * Copyright 2007 Jason King. All rights reserved.
29  * Use is subject to license terms.
30 */

33 #pragma ident      "%Z%M% %I%      %E% SMI"

32 /*
33  * The sparc disassembler is mostly straightforward, each instruction is
34  * represented by an inst_t structure. The inst_t definitions are organized
35  * into tables. The tables are correspond to the opcode maps documented in the
36  * various sparc architecture manuals. Each table defines the bit range of the
37  * instruction whose value act as an index into the array of instructions. A
38  * table can also refer to another table if needed. Each table also contains
39  * a function pointer of type format_fcn that knows how to output the
40  * instructions in the table, as well as handle any synthetic instructions
41  *
42  * Unfortunately, the changes from sparcv8 -> sparcv9 not only include new
43  * instructions, they sometimes renamed or just reused the same instruction to
44  * do different operations (i.e. the sparcv8 coprocessor instructions). To
45  * accommodate this, each table can define an overlay table. The overlay table
46  * is a list of (table index, architecture, new instruction definition) values.
47  *
48  *
49  * Traversal starts with the first table,
50  * get index value from the instruction
51  * if an relevant overlay entry exists for this index,
52  * grab the overlay definition
53  * else
54  * grab the definition from the array (corresponding to the index value)
55  *
56  * If the entry is an instruction,
57  * call print function of instruction.
58  * If the entry is a pointer to another table

```

```

59  * traverse the table
60  * If not valid,
61  * return an error
62  *
63  *
64  * To keep dis happy, for sparc, instead of actually returning an error, if
65  * the instruction cannot be disassembled, we instead merely place the value
66  * of the instruction into the output buffer.
67  *
68  * Adding new instructions:
69  *
70  * With the above information, it hopefully makes it clear how to add support
71  * for decoding new instructions. Presumably, with new instructions will come
72  * a new disassembly mode (I.e. DIS_SPARC_V8, DIS_SPARC_V9, etc.).
73  *
74  * If the disassembled format does not correspond to one of the existing
75  * formats, a new formatter will have to be written. The 'flags' value of
76  * inst_t is intended to instruct the corresponding formatter about how to
77  * output the instruction.
78  *
79  * If the corresponding entry in the correct table is currently unoccupied,
80  * simply replace the INVALID entry with the correct definition. The INST and
81  * TABLE macros are suggested to be used for this. If there is already an
82  * instruction defined, then the entry must be placed in an overlay table. If
83  * no overlay table exists for the instruction table, one will need to be
84  * created.
85  */

87 #include <libdisasm.h>
88 #include <stdlib.h>
89 #include <stdio.h>
90 #include <sys/types.h>
91 #include <sys/byteorder.h>
92 #include <string.h>

94 #include "libdisasm_impl.h"
95 #include "dis_sparc.h"

97 static const inst_t *dis_get_overlay(dis_handle_t *, const table_t *,
98      uint32_t);
99 static uint32_t dis_get_bits(uint32_t, int, int);

101 #if !defined(DIS_STANDALONE)
102 static void do_binary(uint32_t);
103 #endif /* DIS_STANDALONE */

105 dis_handle_t *
106 dis_handle_create(int flags, void *data, dis_lookup_f lookup_func,
107      dis_read_f read_func)
108 {

110 #if !defined(DIS_STANDALONE)
111      char *opt = NULL;
112      char *opt2, *save, *end;
113 #endif
114      dis_handle_t *dhp;

116      if ((flags & (DIS_SPARC_V8|DIS_SPARC_V9|DIS_SPARC_V9_SGI)) == 0) {
117          (void) dis_seterrno(E_DIS_INVALIDFLAG);
118          return (NULL);
119      }

121      if ((dhp = dis_zalloc(sizeof (struct dis_handle))) == NULL) {
122          (void) dis_seterrno(E_DIS_NOMEM);
123          return (NULL);
124      }

```

```

126     dhp->dh_lookup = lookup_func;
127     dhp->dh_read = read_func;
128     dhp->dh_flags = flags;
129     dhp->dh_data = data;
130     dhp->dh_debug = DIS_DEBUG_COMPAT;

132 #if !defined(DIS_STANDALONE)

134     opt = getenv("_LIBDISASM_DEBUG");
135     if (opt == NULL)
136         return (dhp);

138     opt2 = strdup(opt);
139     if (opt2 == NULL) {
140         dis_handle_destroy(dhp);
141         (void) dis_seterrno(E_DIS_NOMEM);
142         return (NULL);
143     }
144     save = opt2;

146     while (opt2 != NULL) {
147         end = strchr(opt2, ',');
148
149         if (end != 0)
150             *end++ = '\0';

152         if (strcasecmp("synth-all", opt2) == 0)
153             dhp->dh_debug |= DIS_DEBUG_SYN_ALL;

155         if (strcasecmp("compat", opt2) == 0)
156             dhp->dh_debug |= DIS_DEBUG_COMPAT;

158         if (strcasecmp("synth-none", opt2) == 0)
159             dhp->dh_debug &= ~(DIS_DEBUG_SYN_ALL|DIS_DEBUG_COMPAT);

161         if (strcasecmp("binary", opt2) == 0)
162             dhp->dh_debug |= DIS_DEBUG_PRTBIN;

164         if (strcasecmp("format", opt2) == 0)
165             dhp->dh_debug |= DIS_DEBUG_PRTFMT;

167         if (strcasecmp("all", opt2) == 0)
168             dhp->dh_debug = DIS_DEBUG_ALL;

170         if (strcasecmp("none", opt2) == 0)
171             dhp->dh_debug = DIS_DEBUG_NONE;

173         opt2 = end;
174     }
175     free(save);
176 #endif /* DIS_STANDALONE */
177     return (dhp);
178 }

```

unchanged_portion_omitted

```

229 int
230 dis_instrlen(dis_handle_t *dhp, uint64_t pc)
231 {
232     return (4);
233 }

235 int
236 #endif /* !codereview */
237 dis_disassemble(dis_handle_t *dhp, uint64_t addr, char *buf, size_t buflen)
238 {

```

```

239     const table_t *tp = &initial_table;
240     const inst_t *inp = NULL;

242     uint32_t instr;
243     uint32_t idx = 0;

245     if (dhp->dh_read(dhp->dh_data, addr, &instr, sizeof (instr)) !=
246         sizeof (instr))
247         return (-1);

249     dhp->dh_buf = buf;
250     dhp->dh_buflen = buflen;
251     dhp->dh_addr = addr;

253     buf[0] = '\0';

255     /* this allows sparc code to be tested on x86 */
256     instr = BE_32(instr);

258 #if !defined(DIS_STANDALONE)
259     if ((dhp->dh_debug & DIS_DEBUG_PRTBIN) != 0)
260         do_binary(instr);
261 #endif /* DIS_STANDALONE */

263     /* CONSTCOND */
264     while (1) {
265         idx = dis_get_bits(instr, tp->tbl_field, tp->tbl_len);
266         inp = &tp->tbl_inp[idx];

268         inp = dis_get_overlay(dhp, tp, idx);

270         if ((inp->in_type == INST_NONE) ||
271             ((inp->in_arch & dhp->dh_flags) == 0))
272             goto error;

274         if (inp->in_type == INST_TBL) {
275             tp = inp->in_data.in_tbl;
276             continue;
277         }

279         break;
280     }

282     if (tp->tbl_fmt(dhp, instr, inp, idx) == 0)
283         return (0);

285 error:

287     (void) snprintf(buf, buflen,
288                    ((dhp->dh_flags & DIS_OCTAL) != 0) ? "%011lo" : "0x%08lx",
289                    instr);

291     return (0);
292 }

294 static uint32_t
295 dis_get_bits(uint32_t instr, int offset, int length)
296 {
297     uint32_t mask, val;
298     int i;

300     for (i = 0, mask = 0; i < length; ++i)
301         mask |= (1UL << i);

303     mask = mask << (offset - length + 1);

```

```
305     val = instr & mask;
307     val = val >> (offset - length + 1);
309     return (val);
310 }

312 static const inst_t *
313 dis_get_overlay(dis_handle_t *dhp, const table_t *tp, uint32_t idx)
314 {
315     const inst_t *ip = &tp->tbl_inp[idx];
316     int i;

318     if (tp->tbl_ovp == NULL)
319         return (ip);

321     for (i = 0; tp->tbl_ovp[i].ov_idx != -1; ++i) {
322         if (tp->tbl_ovp[i].ov_idx != idx)
323             continue;

325         if ((tp->tbl_ovp[i].ov_inst.in_arch & dhp->dh_flags) == 0)
326             continue;

328         ip = &tp->tbl_ovp[i].ov_inst;
329         break;
330     }

332     return (ip);
333 }

335 #if !defined(DIS_STANDALONE)
336 static void
337 do_binary(uint32_t instr)
338 {
339     (void) fprintf(stderr, "DISASM: ");
340     prt_binary(instr, 32);
341     (void) fprintf(stderr, "\n");
342 }
343 #endif /* DIS_STANDALONE */
```

```

*****
2818 Fri Feb 22 23:58:12 2013
new/usr/src/lib/libproc/Makefile.com
saveargs: let disasm do the lifting
3544 save-args matcher could be considerably more robust
3545 save-args matcher should accept saves maybe out-of-order
Reviewed by: Joshua M. Clulow <josh@sysmgr.org>
*****
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 # Copyright (c) 1997, 2010, Oracle and/or its affiliates. All rights reserved.
23 # Copyright 2012 DEY Storage Systems, Inc. All rights reserved.
24 #

26 LIBRARY = libproc.a
27 VERS = .1

29 CMNOBJS = \
30     P32ton.o      \
31     Pcontrol.o   \
32     Pcore.o      \
33     Pexecname.o  \
34     Pfdinfo.o    \
35     Pgcrcore.o   \
36     Pidle.o      \
37     Pisprocdir.o \
38     Plwpregs.o   \
39     Pservice.o   \
40     Psymtab.o    \
41     Psymtab_machelf32.o \
42     $(CMNOBJS64) \
43     Pscantext.o  \
44     Pstack.o     \
45     Psyscall.o   \
46     Putil.o      \
47     Pzone.o      \
48     pr_door.o    \
49     pr_exit.o    \
50     pr_fcntl.o   \
51     pr_getitimer.o \
52     pr_getrctl.o \
53     pr_getrlimit.o \
54     pr_getsockname.o \
55     pr_ioctl.o   \
56     pr_lseek.o   \
57     pr_memcntl.o \
58     pr_meminfo.o \

```

```

59     pr_mmap.o    \
60     pr_open.o    \
61     pr_pbind.o   \
62     pr_rename.o  \
63     pr_sigaction.o \
64     pr_stat.o    \
65     pr_statvfs.o \
66     pr_tasksys.o \
67     pr_waitid.o  \
68     proc_get_info.o \
69     proc_names.o  \
70     proc_arg.o    \
71     proc_set.o    \
72     proc_stdio.o

74 ISAOBJS = \
75     Pisadep.o

77 amd64_SAVEOBS = \
78     saveargs.o

80 amd64_CPPFLAGS = -I$(SRC)/common/saveargs

82 SAVEOBS = $(MACH64)_SAVEOBS

77 OBJECTS = $(CMNOBJS) $(ISAOBJS) $(SAVEOBS)

79 # include library definitions
80 include ../../Makefile.lib
81 include ../../Makefile.rootfs

83 SRCS = $(CMNOBJS:%.o=../../common/%.c) $(ISAOBJS:%.o=%.c)

85 LIBS = $(DYNLIB) $(LINTLIB)
86 LDLIBS += -lrtld_db -lelf -lctf -lc
87 CPPFLAGS += $(MACH64)_CPPFLAGS

89 SRCDIR = ../../common
90 $(LINTLIB) := SRCS = $(SRCDIR)/$(LINTSRC)

92 CFLAGS += $(CCVERBOSE)
93 CPPFLAGS += -I$(SRCDIR)

95 CERRWARN += -_gcc=-Wno-uninitialized
96 CERRWARN += -_gcc=-Wno-parentheses
97 CERRWARN += -_gcc=-Wno-type-limits
98 CERRWARN += -_gcc=-Wno-unused-label

100 # All interfaces are interposable, therefore don't allow direct binding to
101 # libproc. Disable libproc from directly binding to itself, but allow libperl
102 # to directly bind to its dependencies (ie. map -Bdirect -> -zdirect). Ensure
103 # lazy loading is established (which is enabled automatically with -Bdirect).
104 BDIRECT =
105 DYNFLAGS += $(BNODIRECT) $(ZDIRECT) $(ZLAZYLOAD)

107 .KEEP_STATE:

109 all: $(LIBS)

111 lint: lintcheck

113 # include library targets
114 include ../../Makefile.targ

116 objs/%.o pics/%.o: %.c
117     $(COMPILE.c) -o $@ $<

```

new/usr/src/lib/libproc/Makefile.com

3

```
118      $(POST_PROCESS_O)
120 objs/%.o pics/%.o: $(SRC)/common/saveargs/%.c
121      $(COMPILE.c) -o $@ $<
122      $(POST_PROCESS_O)
```

new/usr/src/lib/libproc/amd64/Makefile

1

1275 Fri Feb 22 23:58:12 2013

new/usr/src/lib/libproc/amd64/Makefile

saveargs: let disasm do the lifting

```
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 # Copyright 2007 Sun Microsystems, Inc. All rights reserved.
23 # Use is subject to license terms.
24 #
25 # ident "%Z%M% %I% %E% SMI"
26 #

26 # This is a 64-bit build, and as such needs 64-bit ELF support
27 CMNOBJS64 = Psymtab_machelf64.o
28 SAVEOBJS = saveargs.o
29 #endif /* ! codereview */

31 include ../Makefile.com
32 include ../../Makefile.lib.64

34 CPPFLAGS += -D_SYSCALL32 -I$(SRC)/common/saveargs
35 LDLIBS += -ldisasm
30 CPPFLAGS += -D_SYSCALL32

37 install: all $(ROOTLIBS64) $(ROOTLINKS64)

39 objs/%.o pics/%.o: $(SRC)/common/saveargs/%.c
40 $(COMPILE.c) -o $@ $<
41 $(POST_PROCESS_O)
42 #endif /* ! codereview */
```