****
    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>
****

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */


/*
 * The Sun Studio and GCC (patched for opensolaris/illumos) compilers
 * implement a argument saving scheme on amd64 via the -Wu,save-args or
 * options.  When the option is specified, INTEGER type function arguments
 * passed via registers will be saved on the stack immediately after %rbp, and
 * will not be modified through out the life of the routine.
 *
 *                               +--------+
 *              %rbp    -->     |  %rbp  |
 *                               +--------+
 *              -0x8(%rbp)      |  %rdi  |
 *                               +--------+
 *              -0x10(%rbp)     |  %rsi  |
 *                               +--------+
 *              -0x18(%rbp)     |  %rdx  |
 *                               +--------+
 *              -0x20(%rbp)     |  %rcx  |
 *                               +--------+
 *              -0x28(%rbp)     |  %r8   |
 *                               +--------+
 *              -0x30(%rbp)     |  %r9   |
 *                               +--------+
 *
 *
 * For example, for the following function,
 *
 * void
 * foo(int a1, int a2, int a3, int a4, int a5, int a6, int a7)
 * {
 * ...
 * }
 *
```

```
 * Disassembled code will look something like the following:
 *
 *     pushq   %rbp
 *     movq    %rsp, %rbp
 *     subq    $imm8, %rsp                     **
 *     movq    %rdi, -0x8(%rbp)
 *     movq    %rsi, -0x10(%rbp)
 *     movq    %rdx, -0x18(%rbp)
 *     movq    %rcx, -0x20(%rbp)
 *     movq    %r8, -0x28(%rbp)
 *     movq    %r9, -0x30(%rbp)
 *     ...
 * or
 *     pushq   %rbp
 *     movq    %rsp, %rbp
 *     subq    $imm8, %rsp                     **
 *     movq    %r9, -0x30(%rbp)
 *     movq    %r8, -0x28(%rbp)
 *     movq    %rcx, -0x20(%rbp)
 *     movq    %rdx, -0x18(%rbp)
 *     movq    %rsi, -0x10(%rbp)
 *     movq    %rdi, -0x8(%rbp)
 *     ...
 * or
 *     pushq   %rbp
 *     movq    %rsp, %rbp
 *     pushq   %rdi
 *     pushq   %rsi
 *     pushq   %rdx
 *     pushq   %rcx
 *     pushq   %r8
 *     pushq   %r9
 *
 * **: The space being reserved is in addition to what the current
 *     function prolog already reserves.
 *
 * We loop through the first SAVEARGS_INSN_SEQ_LEN bytes of the function
 * looking for each argument saving instruction we would expect to see.
 * looking for each argument saving instruction we would expect to see.  We
 * loop byte-by-byte, rather than doing anything smart about insn lengths,
 * only deviating from this when we know we have our insn, and can skip the
 * rest of it.
 *
 * If there are odd number of arguments to a function, additional space is
 * reserved on the stack to maintain 16-byte alignment.  For example,
 *
 *     argc == 0: no argument saving.
 *     argc == 3: save 3, but space for 4 is reserved
 *     argc == 7: save 6.
 */

#include <sys/sysmacros.h>
#include <sys/types.h>
#include <libdisasm.h>
#include <string.h>

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

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

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

```
#define INSTR2(ins, off) (ins[(off)] + (ins[(off) + 1] << 8))
#define INSTR3(ins, off)        \
        (ins[(off)] + (ins[(off) + 1] << 8) + (ins[(off + 2)] << 16))
#define INSTR4(ins, off)        \
        (ins[(off)] + (ins[(off) + 1] << 8) + (ins[(off + 2)] << 16) + \
        (ins[(off) + 3] << 24))

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

static const uint16_t save_instr_push[] = {
        0x57,   /* pushq %rdi */
        0x56,   /* pushq %rsi */
        0x52,   /* pushq %rdx */
        0x51,   /* pushq %rcx */
        0x5041, /* pushq %r8 */
        0x5141  /* pushq %r9 */
};

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

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

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

typedef struct {
        uint8_t *data;
        size_t size;
} text_t;

static int
do_read(void *data, uint64_t addr, void *buf, size_t len)
{
        text_t  *t = data;
```

```
        if (addr > t->size)
                return (-1);

        len = MIN(len, t->size - addr);

        (void) memcpy(buf, (char *)t->data + addr, len);

        return (len);
}

/* ARGSUSED */
int
do_lookup(void *data, uint64_t addr, char *buf, size_t buflen, uint64_t *start,
    size_t *symlen)
{
        /* We don’t actually need lookup info */
        return (-1);
}

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

        t.data = ins;
        t.size = size;

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

static boolean_t
has_saved_fp(dis_handle_t *dhp, uint8_t *ins, int size)
has_saved_fp(uint8_t *ins, int size)
{
        int             i, j;
        uint32_t        n;
        boolean_t       found_push = B_FALSE;
        int             sz = 0;

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

                if (found_push == B_FALSE) {
                        if (sz != 1)
                                continue;
        int found_push = 0;

        for (i = 0; i < size; i++) {
                if (found_push == 0) {
                        n = INSTR1(ins, i);
                        for (j = 0; j <= NUM_FP_PUSHES; j++)
                                if (save_fp_pushes[j] == n) {
                                        found_push = B_TRUE;
                                        found_push = 1;
                                        break;
                                }
                } else {
                        if (sz != 3)
                                continue;
#endif /* ! codereview */
                        n = INSTR3(ins, i);
                        for (j = 0; j <= NUM_FP_MOVS; j++)
                                if (save_fp_movs[j] == n)
```

```
                                        return (B_TRUE);
                                        return (1);
                }
        }

        return (B_FALSE);
        return (0);
}

int
saveargs_has_args(uint8_t *ins, size_t size, uint_t argc, int start_index)
{
        int             i, j;
        uint32_t        n;
        uint8_t         found = 0;
        size_t          sz = 0;
        dis_handle_t    *dhp = NULL;
        int             ret = SAVEARGS_NO_ARGS;
#endif /* ! codereview */

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

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

        if (!has_saved_fp(dhp, ins, size)) {
                dis_handle_destroy(dhp);
        if (!has_saved_fp(ins, size))
                return (SAVEARGS_NO_ARGS);
        }
#endif /* ! codereview */

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

                if (n == save_instr[j]) {
                        i += 3;
                        if (++j >= argc)
                                return (start_index ? SAVEARGS_STRUCT_ARGS :
                                    SAVEARGS_TRAD_ARGS);
                }
        }

        /*
         * Compare against regular implementation
         * Compare against GCC implementation
         */
        found = 0;
        for (i = 0; i < size; i += sz) {
                sz = instr_size(dhp, ins, i, size);

                if (sz == -1)
                        break;
                else if (sz != 4)
                        continue;

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

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

                                if (found == ((1 << argc) - 1)) {
                                        ret = start_index ?
                                            SAVEARGS_STRUCT_ARGS :
                                            SAVEARGS_TRAD_ARGS;
                                        goto done;
                                }

                                break;
                        }
                        i += 3;
                        if (--j < start_index)
                                return (SAVEARGS_TRAD_ARGS);
                }
        }

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

                for (j = start_index; j < argc; j++) {
                        if (sz == 2) /* Two byte */
                                n = INSTR2(ins, i);
                        else if (sz == 1)
                                n = INSTR1(ins, i);
                        else
                                continue;
        for (i = 4, j = start_index; i <= size - 2; i += 1) {
                n = (i >= (8 - start_index)) ? INSTR2(ins, i) : INSTR1(ins, i);

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

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

                                break;
                        }
                        if (i >= (8 - start_index))
                                i += 1;
                        if (++j >= argc)
                                return (SAVEARGS_TRAD_ARGS);
                }
        }

        /*
         * Look for a GCC-style returned structure.
         */
        found = 0;
        /* Look for a GCC-style returned structure */
        if (start_index != 0) {
                for (i = 0; i < size; i += sz) {
                        sz = instr_size(dhp, ins, i, size);

                        if (sz == -1)
                                break;
```

```
			else if (sz != 4)
				continue;

		for (i = 4, j = argc - 2; i <= size - 4; i++) {
			n = INSTR4(ins, i);

			/* argc is inclusive of start_index, allow for that */
			for (j = 0; j < (argc - start_index); j++) {
#endif /* ! codereview */
				if (n == save_instr_sr[j]) {
					found |= (1 << j);

					if (found ==
					    ((1 << (argc - start_index)) - 1)) {
						ret = SAVEARGS_TRAD_ARGS;
						goto done;
					}

					break;
				}
				i += 3;
				if (--j >= (argc - 1))
					return (SAVEARGS_TRAD_ARGS);
			}
		}
	}

done:
	dis_handle_destroy(dhp);
	return (ret);
	return (SAVEARGS_NO_ARGS);
}
```
______***unchanged_portion_omitted_***

```
**********************************************************
     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>
**********************************************************
testmatch:

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

functional:

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

dump:

    Display each function in a given object we believe to have saved
    arguments.
#endif /* ! codereview */
```

```
**********************************************************
     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>
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

#
# Copyright 2012, Richard Lowe.
#

include $(SRC)/Makefile.master
include $(SRC)/Makefile.master.64

.KEEP_STATE:

OBJECTS = dump.o saveargs.o dis_tables.o
PROG = dump

CFLAGS += -m64
CPPFLAGS += -I$(SRC)/common/saveargs
CERRWARN += -_gcc=-Wno-parentheses
CERRWARN += -_gcc=-Wno-uninitialized

LDFLAGS += -lctf -lelf
LDLIBS64 += -ldisasm

C99MODE = $(C99_ENABLE)

%.o: $(SRC)/common/saveargs/%.c
        $(COMPILE.c) -o $@ $<

$(PROG): $(OBJECTS)
        $(LINK.c) -o $@ $(OBJECTS) $(LDLIBS64)

clean:
        $(RM) $(OBJECTS) $(PROG)

clobber: clean

all: $(PROG)

install: all
#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>
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

#include <err.h>
#include <fcntl.h>
#include <gelf.h>
#include <libctf.h>
#include <saveargs.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <unistd.h>

extern const char *__progname;

typedef struct symtab_sym {
        GElf_Sym ss_sym;
        char *ss_name;
        ctf_funcinfo_t ss_finfo;
        uint8_t *ss_data;
        size_t ss_size;
} symtab_sym_t;

static void
walk_symtab(Elf *elf, char *fname, ctf_file_t *fp,
    void (*callback)(ctf_file_t *, symtab_sym_t *))
{
        Elf_Scn *stab = NULL;
        Elf_Scn *text = NULL;
        Elf_Data *stabdata = NULL;
        Elf_Data *textdata = NULL;
        GElf_Ehdr ehdr;
        GElf_Shdr stabshdr;
        GElf_Shdr textshdr;
        int fd, foundtext = 0, foundstab = 0;
        symtab_sym_t ss;

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

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

                if (stabshdr.sh_type == SHT_SYMTAB) {
                        foundstab = 1;
                        break;
                }
        }
```

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

                if (strcmp(".text", elf_strptr(elf,
                    ehdr.e_shstrndx, (size_t)textshdr.sh_name)) == 0) {
                        foundtext = 1;
                        break;
                }
        }

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

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

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

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

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

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

out:
        (void) elf_end(elf);
        (void) close(fd);
}

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

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

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

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

int
main(int argc, char **argv)
{
        Elf             *elf;
        ctf_file_t      *ctfp;
        int errp, fd;

        if (ctf_version(CTF_VERSION) == -1)
                errx(1, "mismatched libctf versions\n");

        if (elf_version(EV_CURRENT) == EV_NONE)
                errx(1, "mismatched libelf versions\n");

        if (argc != 2)
                errx(2, "usage: %s <file>\n", __progname);

        if ((ctfp = ctf_open(argv[1], &errp)) == NULL)
                errx(1, "failed to ctf_open file: %s: %s\n", argv[1],
                    ctf_errmsg(errp));

        if ((fd = open(argv[1], O_RDONLY)) == -1)
                errx(1, "could not open %s\n", argv[1]);

        if ((elf = elf_begin(fd, ELF_C_READ, NULL)) == NULL)
                errx(1, "could not interpret ELF from %s\n",
                    argv[1]);

        walk_symtab(elf, argv[1], ctfp, check_sym);

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

```
**********************************************************
     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>
**********************************************************
```

```
#endif /* ! codereview */
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

#
# Copyright 2012, Richard Lowe.
#

include $(SRC)/Makefile.master
include $(SRC)/Makefile.master.64

.KEEP_STATE:

OBJECTS = testmatch.o saveargs.o data.o
PROG = testmatch

LDLIBS64 += -ldisasm

#endif /* ! codereview */
CPPFLAGS += -I$(SRC)/common/saveargs
ASFLAGS += -P
AS_CPPFLAGS += -D_ASM
CERRWARN += -_gcc=-Wno-parentheses
CERRWARN += -_gcc=-Wno-uninitialized
#endif /* ! codereview */

%.o: $(SRC)/common/saveargs/%.c
        $(COMPILE.c) -o $@ $<

$(PROG): $(OBJECTS)
        $(LINK.c) -o $@ $(OBJECTS) $(LDLIBS64)
        $(LINK.c) -o $@ $(OBJECTS) -lc

clean:
        $(RM) $(OBJECTS) $(PROG)

clobber: clean

all: $(PROG)

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_
```

```
FUNC(gcc_mov_noorder)
pushq   %rbp
movq    %rsp, %rbp
movq    %rcx,-0x20(%rbp)
movq    %rbx,-0x28(%rbp)
movq    %rdi,-0x8(%rbp)
movq    %rdx,-0x18(%rbp)
movq    %rsi,-0x10(%rbp)
subq    $0x50,%rsp
SET_SIZE(gcc_mov_noorder, gcc_mov_noorder_end)

#endif /* ! codereview */
FUNC(gcc_mov_big_struct_ret)
pushq   %rbp
movq    %rsp,%rbp
movq    %rbx,-0x28(%rbp)
movq    %r8,-0x20(%rbp)
movq    %rcx,-0x18(%rbp)
movq    %rdx,-0x10(%rbp)
movq    %rsi,-0x8(%rbp)
subq    $0x50,%rsp
SET_SIZE(gcc_mov_big_struct_ret, gcc_mov_big_struct_ret_end)

FUNC(gcc_mov_struct_noorder)
pushq   %rbp
movq    %rsp,%rbp
movq    %rcx,-0x18(%rbp)
movq    %r8,-0x20(%rbp)
movq    %rsi,-0x8(%rbp)
movq    %rdx,-0x10(%rbp)
movq    %rbx,-0x28(%rbp)
subq    $0x50,%rsp
SET_SIZE(gcc_mov_struct_noorder, gcc_mov_struct_noorder_end)

#endif /* ! codereview */
FUNC(gcc_mov_big_struct_ret_and_spill)
pushq   %rbp
movq    %rsp,%rbp
movq    %rbx,-0x38(%rbp)
movq    %r9,-0x28(%rbp)
movq    %r8,-0x20(%rbp)
movq    %rcx,-0x18(%rbp)
movq    %rdx,-0x10(%rbp)
movq    %rsi,-0x8(%rbp)
subq    $0x90,%rsp
SET_SIZE(gcc_mov_big_struct_ret_and_spill, gcc_mov_big_struct_ret_and_spill_end)

FUNC(gcc_mov_small_struct_ret)
pushq   %rbp
movq    %rsp,%rbp
movq    %rbx,-0x28(%rbp)
movq    %rcx,-0x20(%rbp)
movq    %rdx,-0x18(%rbp)
movq    %rsi,-0x10(%rbp)
movq    %rdi,-0x8(%rbp)
subq    $0x50,%rsp
SET_SIZE(gcc_mov_small_struct_ret, gcc_mov_small_struct_ret_end)
```

```
FUNC(gcc_mov_small_struct_ret_and_spill)
pushq   %rbp
movq    %rsp,%rbp
movq    %rbx,-0x38(%rbp)
movq    %r9,-0x30(%rbp)
movq    %r8,-0x28(%rbp)
movq    %rcx,-0x20(%rbp)
movq    %rdx,-0x18(%rbp)
movq    %rsi,-0x10(%rbp)
movq    %rdi,-0x8(%rbp)
subq    $0x90,%rsp
SET_SIZE(gcc_mov_small_struct_ret_and_spill, gcc_mov_small_struct_ret_and_spill_

FUNC(gcc_mov_stack_spill)
pushq   %rbp
movq    %rsp,%rbp
movq    %rbx,-0x38(%rbp)
movq    %r9,-0x30(%rbp)
movq    %r8,-0x28(%rbp)
movq    %rcx,-0x20(%rbp)
movq    %rdx,-0x18(%rbp)
movq    %rsi,-0x10(%rbp)
movq    %rdi,-0x8(%rbp)
subq    $0x90,%rsp
SET_SIZE(gcc_mov_stack_spill, gcc_mov_stack_spill_end)

FUNC(gcc_push_align)
pushq   %rbp
movq    %rsp,%rbp
pushq   %rdi
pushq   %rsi
pushq   %rdx
pushq   %rcx
pushq   %r8
subq    $0x8,%rsp
subq    $0x30,%rsp
SET_SIZE(gcc_push_align, gcc_push_align_end)

FUNC(gcc_push_basic)
pushq   %rbp
movq    %rsp,%rbp
pushq   %rdi
pushq   %rsi
pushq   %rdx
pushq   %rcx
subq    $0x20,%rsp
SET_SIZE(gcc_push_basic, gcc_push_basic_end)

FUNC(gcc_push_noorder)
pushq   %rbp
movq    %rsp,%rbp
pushq   %rsi
pushq   %rdi
pushq   %rcx
pushq   %rdx
subq    $0x20,%rsp
SET_SIZE(gcc_push_noorder, gcc_push_noorder_end)

#endif /* ! codereview */
FUNC(gcc_push_big_struct_ret)
pushq   %rbp
movq    %rsp,%rbp
pushq   %rsi
pushq   %rdx
pushq   %rcx
pushq   %r8
subq    $0x30,%rsp
SET_SIZE(gcc_push_big_struct_ret, gcc_push_big_struct_ret_end)

FUNC(gcc_push_struct_noorder)
pushq   %rbp
movq    %rsp,%rbp
pushq   %rdx
pushq   %rsi
pushq   %r8
pushq   %rcx
subq    $0x30,%rsp
SET_SIZE(gcc_push_struct_noorder, gcc_push_struct_noorder_end)

#endif /* ! codereview */
FUNC(gcc_push_big_struct_ret_and_spill)
pushq   %rbp
movq    %rsp,%rbp
pushq   %rsi
pushq   %rdx
pushq   %rcx
pushq   %r8
pushq   %r9
subq    $0x8,%rsp
subq    $0x50,%rsp
SET_SIZE(gcc_push_big_struct_ret_and_spill, gcc_push_big_struct_ret_and_spill_en

FUNC(gcc_push_small_struct_ret)
pushq   %rbp
movq    %rsp,%rbp
pushq   %rdi
pushq   %rsi
pushq   %rdx
pushq   %rcx
subq    $0x20,%rsp
SET_SIZE(gcc_push_small_struct_ret, gcc_push_small_struct_ret_end)

FUNC(gcc_push_small_struct_ret_and_spill)
pushq   %rbp
movq    %rsp,%rbp
pushq   %rdi
pushq   %rsi
pushq   %rdx
pushq   %rcx
pushq   %r8
pushq   %r9
subq    $0x50,%rsp
SET_SIZE(gcc_push_small_struct_ret_and_spill, gcc_push_small_struct_ret_and_spil

FUNC(gcc_push_stack_spill)
pushq   %rbp
movq    %rsp,%rbp
pushq   %rdi
pushq   %rsi
pushq   %rdx
pushq   %rcx
pushq   %r8
pushq   %r9
subq    $0x50,%rsp
SET_SIZE(gcc_push_stack_spill, gcc_push_stack_spill_end)

FUNC(ss_mov_align)
pushq   %rbp
movq    %rsp,%rbp
subq    $0x30,%rsp
movq    %rdi,-0x8(%rbp)
```

```
movq    %rsi,-0x10(%rbp)
movq    %rdx,-0x18(%rbp)
movq    %rcx,-0x20(%rbp)
movq    %r8,-0x28(%rbp)
SET_SIZE(ss_mov_align, ss_mov_align_end)

FUNC(ss_mov_basic)
pushq   %rbp
movq    %rsp,%rbp
subq    $0x20,%rsp
movq    %rdi,-0x8(%rbp)
movq    %rsi,-0x10(%rbp)
movq    %rdx,-0x18(%rbp)
movq    %rcx,-0x20(%rbp)
SET_SIZE(ss_mov_basic, ss_mov_basic_end)

FUNC(ss_mov_big_struct_ret)
pushq   %rbp
movq    %rsp,%rbp
subq    $0x30,%rsp
movq    %rdi,-0x8(%rbp)
movq    %rsi,-0x10(%rbp)
movq    %rdx,-0x18(%rbp)
movq    %rcx,-0x20(%rbp)
movq    %r8,-0x28(%rbp)
SET_SIZE(ss_mov_big_struct_ret, ss_mov_big_struct_ret_end)

FUNC(ss_mov_big_struct_ret_and_spill)
pushq   %rbp
movq    %rsp,%rbp
subq    $0x50,%rsp
movq    %rdi,-0x8(%rbp)
movq    %rsi,-0x10(%rbp)
movq    %rdx,-0x18(%rbp)
movq    %rcx,-0x20(%rbp)
movq    %r8,-0x28(%rbp)
movq    %r9,-0x30(%rbp)
SET_SIZE(ss_mov_big_struct_ret_and_spill, ss_mov_big_struct_ret_and_spill_end)

FUNC(ss_mov_small_struct_ret)
pushq   %rbp
movq    %rsp,%rbp
subq    $0x20,%rsp
movq    %rdi,-0x8(%rbp)
movq    %rsi,-0x10(%rbp)
movq    %rdx,-0x18(%rbp)
movq    %rcx,-0x20(%rbp)
SET_SIZE(ss_mov_small_struct_ret, ss_mov_small_struct_ret_end)

FUNC(ss_mov_small_struct_ret_and_spill)
pushq   %rbp
movq    %rsp,%rbp
subq    $0x50,%rsp
movq    %rdi,-0x8(%rbp)
movq    %rsi,-0x10(%rbp)
movq    %rdx,-0x18(%rbp)
movq    %rcx,-0x20(%rbp)
movq    %r8,-0x28(%rbp)
movq    %r9,-0x30(%rbp)
SET_SIZE(ss_mov_small_struct_ret_and_spill, ss_mov_small_struct_ret_and_spill_en

FUNC(ss_mov_stack_spill)
pushq   %rbp
movq    %rsp,%rbp
subq    $0x50,%rsp
movq    %rdi,-0x8(%rbp)
movq    %rsi,-0x10(%rbp)
movq    %rdx,-0x18(%rbp)
movq    %rcx,-0x20(%rbp)
movq    %r8,-0x28(%rbp)
movq    %r9,-0x30(%rbp)
SET_SIZE(ss_mov_stack_spill, ss_mov_stack_spill_end)

/* DTrace instrumentation */
FUNC(dtrace_instrumented)
int     $0x3
movq    %rsp, %rbp
movq    %rbx,-0x28(%rbp)
movq    %rcx,-0x20(%rbp)
movq    %rdx,-0x18(%rbp)
movq    %rsi,-0x10(%rbp)
movq    %rdi,-0x8(%rbp)
subq    $0x50,%rsp
SET_SIZE(dtrace_instrumented, dtrace_instrumented_end)

/*
 * System functions with special characteristics, be they non-initial FP save,
 * gaps between FP save and argument saving, or gaps between saved arguments.
 */
FUNC(kmem_alloc)
leaq    -0x1(%rdi),%rax
pushq   %rbp
movq    %rax,%rdx
movq    %rsp,%rbp
subq    $0x30,%rsp
shrq    $0x3,%rdx
movq    %r12,-0x28(%rbp)
movq    %rbx,-0x30(%rbp)
cmpq    $0x1ff,%rdx
movq    %r13,-0x20(%rbp)
movq    %r14,-0x18(%rbp)
movq    %rsi,-0x10(%rbp)
movq    %rdi,-0x8(%rbp)
movq    %rdi,%r12
SET_SIZE(kmem_alloc, kmem_alloc_end)

FUNC(uts_kill)
pushq   %rbp
movq    %rsp,%rbp
subq    $0x50,%rsp
movq    %rbx,-0x28(%rbp)
leaq    -0x50(%rbp),%rbx
movq    %r12,-0x20(%rbp)
movq    %r13,-0x18(%rbp)
movq    %rsi,-0x10(%rbp)
movl    %edi,%r12d
movq    %rdi,-0x8(%rbp)
SET_SIZE(uts_kill, uts_kill_end)

FUNC(av1394_ic_bitreverse)
movq    %rdi,%rdx
movq    $0x5555555555555555,%rax
movq    $0x3333333333333333,%rcx
shrq    $0x1,%rdx
pushq   %rbp
andq    %rax,%rdx
andq    %rdi,%rax
addq    %rax,%rax
movq    %rsp,%rbp
subq    $0x10,%rsp
orq     %rdx,%rax
movq    %rdi,-0x8(%rbp)
```

```
SET_SIZE(av1394_ic_bitreverse, av1394_ic_bitreverse_end)

/* Problematic functions which should not match */

FUNC(no_fp) /* No frame pointer */
movq    %rdi, %rsi
movq    %rsi, %rdi
movq    %rbx,-0x28(%rbp)
movq    %rcx,-0x20(%rbp)
movq    %rdx,-0x18(%rbp)
movq    %rsi,-0x10(%rbp)
movq    %rdi,-0x8(%rbp)
subq    $0x50,%rsp
SET_SIZE(no_fp, no_fp_end)

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

/* Big structure return, but with an SSE type */
FUNC(big_struct_ret_w_float)
pushq  %rbp
movq   %rsp,%rbp
movq   %rsi,-0x8(%rbp)
subq   $0x50,%rsp
movq   %rsi,-0x48(%rbp)
movq   -0x48(%rbp),%rax
movq   %rax,%rsi
movl   $0x400f60,%edi
movl   $0x0,%eax
movl   $0x1770,%edi
movl   $0x0,%eax
leave
ret
SET_SIZE(big_struct_ret_w_float, big_struct_ret_w_float_end)

FUNC(big_struct_arg_by_value)
pushq   %rbp
movq    %rsp,%rbp
movq    %rdi,-0x8(%rbp)
subq    $0x40,%rsp
SET_SIZE(big_struct_arg_by_value, big_struct_arg_by_value_end)

FUNC(small_struct_arg_by_value)
pushq   %rbp
movq    %rsp,%rbp
movq    %rdx,-0x18(%rbp)
movq    %rsi,-0x10(%rbp)
movq    %rdi,-0x8(%rbp)
subq    $0x50,%rsp
SET_SIZE(small_struct_arg_by_value, small_struct_arg_by_value_end)

FUNC(interleaved_argument_saves)
pushq   %rbp
movq    %rdi,%rax
shlq    $0x21,%rax
movq    %rsp,%rbp
shrq    $0x29,%rax
subq    $0x30,%rsp
movq    %rdi,-0x8(%rbp)
movq    %rbx,-0x28(%rbp)
movzbl  %dil,%edi
```

```
movq    %rcx,-0x20(%rbp)
movq    %rdx,-0x18(%rbp)
movq    %rsi,-0x10(%rbp)
movq    0x0(,%rax,8),%rax
SET_SIZE(interleaved_argument_saves, interleaved_argument_saves_end)
#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>
**********************************************************
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2012, Richard Lowe.
 */

#include <stdio.h>
#include <sys/types.h>
#include <saveargs.h>

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

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

#define TEST_GOOD(name, argc)                                   \
        if (saveargs_has_args(name, SIZE_OF(name), argc, 0) ==  \
            SAVEARGS_TRAD_ARGS)                                 \
    if (saveargs_has_args(name, SIZE_OF(name), argc, 0) != 0)   \
                printf("Pass: %s\n", #name);                    \
        else                                                    \
                printf("FAIL: %s\n", #name);

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

/*
 * GCC deals with structures differently, so TRAD args is actually correct for
 * this
 */
#define TEST_GOOD_GSTRUCT(name, argc)                           \
        if (saveargs_has_args(name, SIZE_OF(name), argc, 1) ==  \
            SAVEARGS_TRAD_ARGS)                                 \
    if (saveargs_has_args(name, SIZE_OF(name), argc, 1) != 0)   \
                printf("Pass: %s\n", #name);                    \
        else                                                    \
                printf("FAIL: %s\n", #name);

#define TEST_BAD(name, argc)                                    \
        if (saveargs_has_args(name, SIZE_OF(name), argc, 0) ==  \
                SAVEARGS_NO_ARGS)                               \
    if (saveargs_has_args(name, SIZE_OF(name), argc, 0) == 0)   \
                printf("Pass: %s\n", #name);                    \
        else                                                    \
                printf("FAIL: %s\n", #name);
```

```
#define TEST_BAD_STRUCT(name, argc)                             \
        if (saveargs_has_args(name, SIZE_OF(name), argc, 1) ==  \
                SAVEARGS_NO_ARGS)                               \
                printf("Pass: %s\n", #name);                    \
        else                                                    \
                printf("FAIL: %s\n", #name);

#define TEST_BAD_GSTRUCT(name, argc)                            \
        if (saveargs_has_args(name, SIZE_OF(name), argc, 1) ==  \
                SAVEARGS_NO_ARGS)                               \
    if (saveargs_has_args(name, SIZE_OF(name), argc, 1) == 0)   \
                printf("Pass: %s\n", #name);                    \
        else                                                    \
                printf("FAIL: %s\n", #name);

DEF_TEST(gcc_mov_align);
DEF_TEST(gcc_mov_basic);
DEF_TEST(gcc_mov_noorder);
DEF_TEST(gcc_mov_struct_noorder);
#endif /* ! codereview */
DEF_TEST(gcc_mov_big_struct_ret);
DEF_TEST(gcc_mov_big_struct_ret_and_spill);
DEF_TEST(gcc_mov_small_struct_ret);
DEF_TEST(gcc_mov_small_struct_ret_and_spill);
DEF_TEST(gcc_mov_stack_spill);

DEF_TEST(gcc_push_align);
DEF_TEST(gcc_push_basic);
DEF_TEST(gcc_push_noorder);
DEF_TEST(gcc_push_struct_noorder);
#endif /* ! codereview */
DEF_TEST(gcc_push_big_struct_ret);
DEF_TEST(gcc_push_big_struct_ret_and_spill);
DEF_TEST(gcc_push_small_struct_ret);
DEF_TEST(gcc_push_small_struct_ret_and_spill);
DEF_TEST(gcc_push_stack_spill);

DEF_TEST(ss_mov_align);
DEF_TEST(ss_mov_basic);
DEF_TEST(ss_mov_big_struct_ret);
DEF_TEST(ss_mov_big_struct_ret_and_spill);
DEF_TEST(ss_mov_small_struct_ret);
DEF_TEST(ss_mov_small_struct_ret_and_spill);
DEF_TEST(ss_mov_stack_spill);

DEF_TEST(dtrace_instrumented);
DEF_TEST(kmem_alloc);
DEF_TEST(uts_kill);
DEF_TEST(av1394_ic_bitreverse);

DEF_TEST(small_struct_ret_w_float);
DEF_TEST(big_struct_ret_w_float);

DEF_TEST(interleaved_argument_saves);

#endif /* ! codereview */
/*
 * Functions which should not match
 *
 * no_fp                          -- valid save-args sequence with no saved FP
 * big_struct_arg_by_value        -- function with big struct passed by value
 * small_struct_arg_by_value      -- function with small struct passed by value
 */
DEF_TEST(no_fp);
DEF_TEST(big_struct_arg_by_value);
DEF_TEST(small_struct_arg_by_value);

int
main(int argc, char **argv)
{
        TEST_GOOD(kmem_alloc, 2);
        TEST_GOOD(uts_kill, 2);
        TEST_GOOD(av1394_ic_bitreverse, 1);
        TEST_GOOD(dtrace_instrumented, 4);
        TEST_GOOD_GSTRUCT(big_struct_ret_w_float, 1);
        TEST_GOOD_STRUCT(big_struct_ret_w_float, 1);
        TEST_BAD(no_fp, 5);

        TEST_GOOD(gcc_mov_align, 5);
        TEST_GOOD(gcc_push_align, 5);
        TEST_GOOD(ss_mov_align, 5);

        TEST_GOOD(gcc_mov_basic, 4);
        TEST_GOOD(gcc_push_basic, 4);
        TEST_GOOD(ss_mov_basic, 4);

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

        TEST_GOOD_GSTRUCT(gcc_mov_big_struct_ret, 4);
        TEST_GOOD_GSTRUCT(gcc_push_big_struct_ret, 4);
        TEST_GOOD_STRUCT(gcc_mov_big_struct_ret, 4);
        TEST_GOOD_STRUCT(gcc_push_big_struct_ret, 4);
        TEST_GOOD_STRUCT(ss_mov_big_struct_ret, 4);

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

        TEST_GOOD_GSTRUCT(gcc_mov_big_struct_ret_and_spill, 8);
        TEST_GOOD_GSTRUCT(gcc_push_big_struct_ret_and_spill, 8);
        TEST_GOOD_STRUCT(gcc_mov_big_struct_ret_and_spill, 8);
        TEST_GOOD_STRUCT(gcc_push_big_struct_ret_and_spill, 8);
        TEST_GOOD_STRUCT(ss_mov_big_struct_ret_and_spill, 8);

        TEST_GOOD(gcc_mov_small_struct_ret, 4);
        TEST_GOOD(gcc_push_small_struct_ret, 4);
        TEST_GOOD(ss_mov_small_struct_ret, 4);

        TEST_GOOD(gcc_mov_small_struct_ret_and_spill, 8);
        TEST_GOOD(gcc_push_small_struct_ret_and_spill, 8);
        TEST_GOOD(ss_mov_small_struct_ret_and_spill, 8);

        TEST_GOOD(gcc_mov_stack_spill, 8);
        TEST_GOOD(gcc_push_stack_spill, 8);
        TEST_GOOD(ss_mov_stack_spill, 8);

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

        TEST_BAD(small_struct_ret_w_float, 1);

        TEST_GOOD(interleaved_argument_saves, 4);

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

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

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

#ifndef _LIBDISASM_H
#define _LIBDISASM_H

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

#include <sys/types.h>

#ifdef  __cplusplus
extern "C" {
#endif

typedef struct dis_handle dis_handle_t;

#define DIS_DEFAULT             0x0

/* SPARC disassembler flags */
#define DIS_SPARC_V8            0x01
#define DIS_SPARC_V9            0x02
#define DIS_SPARC_V9_SGI        0x04
#define DIS_SPARC_V9_OPL        0x08

/* x86 diassembler flags (mutually exclusive) */
#define DIS_X86_SIZE16          0x08
#define DIS_X86_SIZE32          0x10
#define DIS_X86_SIZE64          0x20

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

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

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

```
extern void dis_handle_destroy(dis_handle_t *);

extern int dis_disassemble(dis_handle_t *, uint64_t, char *, size_t);
extern uint64_t dis_previnstr(dis_handle_t *, uint64_t, int n);
extern void dis_set_data(dis_handle_t *, void *);
extern void dis_flags_set(dis_handle_t *, int f);
extern void dis_flags_clear(dis_handle_t *, int f);
extern int dis_max_instrlen(dis_handle_t *);
extern int dis_instrlen(dis_handle_t *, uint64_t);
#endif /* ! codereview */

/* libdisasm errors */
#define E_DIS_NOMEM             1       /* Out of memory */
#define E_DIS_INVALFLAG         2       /* Invalid flag for this architecture */

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

#ifdef  __cplusplus
}
#endif

#endif  /* _LIBDISASM_H */
```

```
**********************************************************
    1507 Fri Feb 22 23:58:10 2013
new/usr/src/lib/libdisasm/common/mapfile-vers
saveargs: let disasm do the lifting
**********************************************************
```

```
#
# MAPFILE HEADER START
#
# WARNING:  STOP NOW.  DO NOT MODIFY THIS FILE.
# Object versioning must comply with the rules detailed in
#
#       usr/src/lib/README.mapfiles
#
# You should not be making modifications here until you've read the most current
# copy of that file. If you need help, contact a gatekeeper for guidance.
#
# MAPFILE HEADER END
#

$mapfile_version 2

SYMBOL_VERSION SUNWprivate_1.1 {
    global:
        dis_disassemble;
        dis_errno;
        dis_handle_create;
        dis_handle_destroy;
        dis_instrlen;
#endif /* ! codereview */
        dis_max_instrlen;
        dis_previnstr;
        dis_set_data;
        dis_flags_set;
        dis_flags_clear;
        dis_strerror;
    local:
        *;
};
```

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

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

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

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

#include <libdisasm.h>
#include <stdlib.h>
#include <stdio.h>

#include "dis_tables.h"
#include "libdisasm_impl.h"

struct dis_handle {
        void            *dh_data;
        int             dh_flags;
        dis_lookup_f    dh_lookup;
        dis_read_f      dh_read;
        int             dh_mode;
        dis86_t         dh_dis;
        uint64_t        dh_addr;
        uint64_t        dh_end;
};
______unchanged_portion_omitted_

int
dis_instrlen(dis_handle_t *dhp, uint64_t pc)
{
        if (dis_disassemble(dhp, pc, NULL, 0) != 0)
                return (-1);

        return (dhp->dh_addr - pc);
}
#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
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

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

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


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

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

```
 *     traverse the table
 * If not valid,
 *     return an error
 *
 *
 * To keep dis happy, for sparc, instead of actually returning an error, if
 * the instruction cannot be disassembled, we instead merely place the value
 * of the instruction into the output buffer.
 *
 * Adding new instructions:
 *
 * With the above information, it hopefully makes it clear how to add support
 * for decoding new instructions.  Presumably, with new instructions will come
 * a new dissassembly mode (I.e. DIS_SPARC_V8, DIS_SPARC_V9, etc.).
 *
 * If the dissassembled format does not correspond to one of the existing
 * formats, a new formatter will have to be written.  The 'flags' value of
 * inst_t is intended to instruct the corresponding formatter about how to
 * output the instruction.
 *
 * If the corresponding entry in the correct table is currently unoccupied,
 * simply replace the INVALID entry with the correct definition.  The INST and
 * TABLE macros are suggested to be used for this.  If there is already an
 * instruction defined, then the entry must be placed in an overlay table.  If
 * no overlay table exists for the instruction table, one will need to be
 * created.
 */

#include <libdisasm.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/byteorder.h>
#include <string.h>

#include "libdisasm_impl.h"
#include "dis_sparc.h"

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

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

dis_handle_t *
dis_handle_create(int flags, void *data, dis_lookup_f lookup_func,
    dis_read_f read_func)
{

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

        if ((flags & (DIS_SPARC_V8|DIS_SPARC_V9|DIS_SPARC_V9_SGI)) == 0) {
                (void) dis_seterrno(E_DIS_INVALFLAG);
                return (NULL);
        }

        if ((dhp = dis_zalloc(sizeof (struct dis_handle))) == NULL) {
                (void) dis_seterrno(E_DIS_NOMEM);
                return (NULL);
        }
        dhp->dh_lookup = lookup_func;
        dhp->dh_read = read_func;
        dhp->dh_flags = flags;
        dhp->dh_data = data;
        dhp->dh_debug = DIS_DEBUG_COMPAT;

#if !defined(DIS_STANDALONE)

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

        opt2 = strdup(opt);
        if (opt2 == NULL) {
                dis_handle_destroy(dhp);
                (void) dis_seterrno(E_DIS_NOMEM);
                return (NULL);
        }
        save = opt2;

        while (opt2 != NULL) {
                end = strchr(opt2, ',');

                if (end != 0)
                        *end++ = '\0';

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

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

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

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

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

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

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

                opt2 = end;
        }
        free(save);
#endif /* DIS_STANDALONE */
        return (dhp);
}
```

______***unchanged_portion_omitted_***

```
int
dis_instrlen(dis_handle_t *dhp, uint64_t pc)
{
        return (4);
}

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

```
        const table_t *tp = &initial_table;
        const inst_t *inp = NULL;

        uint32_t instr;
        uint32_t idx = 0;

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

        dhp->dh_buf    = buf;
        dhp->dh_buflen = buflen;
        dhp->dh_addr   = addr;

        buf[0] = '\0';

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

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

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

                inp = dis_get_overlay(dhp, tp, idx);

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

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

                break;
        }

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

error:

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

        return (0);
}

static uint32_t
dis_get_bits(uint32_t instr, int offset, int length)
{
        uint32_t mask, val;
        int i;

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

        mask = mask << (offset - length + 1);
        val = instr & mask;

        val = val >> (offset - length + 1);

        return (val);
}

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

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

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

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

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

        return (ip);
}

#if !defined(DIS_STANDALONE)
static void
do_binary(uint32_t instr)
{
        (void) fprintf(stderr, "DISASM: ");
        prt_binary(instr, 32);
        (void) fprintf(stderr, "\n");
}
#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>
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright (c) 1997, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2012 DEY Storage Systems, Inc.  All rights reserved.
#

LIBRARY = libproc.a
VERS = .1

CMNOBJS =       \
        P32ton.o        \
        Pcontrol.o      \
        Pcore.o         \
        Pexecname.o     \
        Pfdinfo.o       \
        Pgcore.o        \
        Pidle.o         \
        Pisprocdir.o    \
        Plwpregs.o      \
        Pservice.o      \
        Psymtab.o       \
        Psymtab_machelf32.o \
        $(CMNOBJS64)    \
        Pscantext.o     \
        Pstack.o        \
        Psyscall.o      \
        Putil.o         \
        Pzone.o         \
        pr_door.o       \
        pr_exit.o       \
        pr_fcntl.o      \
        pr_getitimer.o  \
        pr_getrctl.o    \
        pr_getrlimit.o  \
        pr_getsockname.o \
        pr_ioctl.o      \
        pr_lseek.o      \
        pr_memcntl.o    \
        pr_meminfo.o    \
```

```
        pr_mmap.o       \
        pr_open.o       \
        pr_pbind.o      \
        pr_rename.o     \
        pr_sigaction.o  \
        pr_stat.o       \
        pr_statvfs.o    \
        pr_tasksys.o    \
        pr_waitid.o     \
        proc_get_info.o \
        proc_names.o    \
        proc_arg.o      \
        proc_set.o      \
        proc_stdio.o

ISAOBJS =       \
        Pisadep.o

amd64_SAVEOBJS = \
        saveargs.o

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

SAVEOBJS = $($(MACH64)_SAVEOBJS)

OBJECTS = $(CMNOBJS) $(ISAOBJS) $(SAVEOBJS)

# include library definitions
include ../../Makefile.lib
include ../../Makefile.rootfs

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

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

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

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

CERRWARN +=     -_gcc=-Wno-uninitialized
CERRWARN +=     -_gcc=-Wno-parentheses
CERRWARN +=     -_gcc=-Wno-type-limits
CERRWARN +=     -_gcc=-Wno-unused-label

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

.KEEP_STATE:

all: $(LIBS)

lint: lintcheck

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

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

```
	$(POST_PROCESS_O)

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

```
**********************************************************
    1275 Fri Feb 22 23:58:12 2013
new/usr/src/lib/libproc/amd64/Makefile
saveargs: let disasm do the lifting
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "%Z%%M% %I%     %E% SMI"
#

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

include ../Makefile.com
include ../../Makefile.lib.64

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

install: all $(ROOTLIBS64) $(ROOTLINKS64)

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