UNIVERSITY OF CALIFORNIA, SAN FRANCISCO

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO SANTA BARBARA • SANTA CRUZ

SCHOOL OF PHARMACY
DEPARTMENT OF PHARMACEUTICAL CHEMISTRY
COMPUTER GRAPHICS LABORATORY
PHONE: (415) 476-2299
E-MAIL: tef@cgl.ucsf.edu

SAN FRANCISCO, CALIFORNIA 94143-2240
FAX: (415) 502-1755

December 9, 1977

Professor Melvin Ferentz
Editor, *;login:*
Physics Department
Brooklyn College of CUNY
Brooklyn, New York 11210

Re: Solution to the *nargs()* problem in separate I and D space

The memory management unit in PDP-11/45 and 11/70 computers offer several advantages over those found in the other PDP-11 family computers. Among the more powerful features is the ability to separate programs into instruction segments and data segments, each segment having the ability to directly reference 64K bytes of memory.

Four PDP-11 instructions facilitate program communication between different addressing modes and instruction/data areas in memory. These are "move to/from previous instruction/data memory space" (mtpi, mfpi, mtpd, mfpd). These instructions are used extensively by UNIX to fetch and store data between user programs and the operating system.

Because of Digital Equipment Corporation's (DEC) desire to "...preserve the integrity of proprietary programs", the 'mfpi' instruction does not work correctly when executed with a processor status word equal to 17xxxx (i.e. both current and previous modes are USER). This fact prevents the C subroutine 'nargs.s' from operating as intended when instruction and data space are separated.

There are several solutions to this deficiency, among these are:

1. Implement a system call which executes (correctly) the instruction 'mfpi' with source equal to USER I space.
2. Modify the C compiler to generate an argument count on each and every procedure call.
3. Modify UNIX to use the supervisor mode memory management registers in a way different then it currently does.
4. Modify the computer hardware to work correctly.

The first of these solutions proved unacceptable in terms of overhead time; each system call on the PDP-11/70 consumes equal or greater than 320 microseconds, and 'nargs.s' has the potential for making several of these. In our real time interactive graphics environment this prohibited use of separate I&D space. The second and third solutions were unattractive from a software standards point of view.

After several telephone calls to DEC representatives and a few hours of looking at microcode and logic schematics, we have arrived at a simple modification to (at least) the PDP-11/70 cpu to allow the 'mfpi' instruction to function properly. The modification takes about 15 minutes for an experienced person to implement and involves cutting one foil etch and adding one jumper wire to the M8138-YA memory management board. The relevant section of the PDP-11/70 schematic, DEC #M8138-0-1, "SYS. STATUS REG. (SSRB)", is enclosed. Please note this is as it appears on our KB11-C central processing unit and may vary with other processors.

Also included is a listing of our modified 'nargs.s' subroutine that utilizes the "new" instruction.

We have been running under the modified hardware for several weeks now and have had no problems. In particular, absolutely no changes have been made to any other portion of the UNIX system other than the 'nargs.s' subroutine. On our system the PDP-11/70 memory management diagnostic program, MAINDEC-11-DEKBE-B, was patched to skip around the relevant portion of the diagnostic which tests the "proprietary program" feature; this is optional.

While this solution also has some drawbacks, it has proved quite successful for our application involving real time interactive computer graphics. I trust this information will be useful to others as well.

Sincerely,

Tom Ferrin
UCSF Computer Graphics Laboratory

References:

[1] KB11-C processor manual (PDP-11/70). DEC# EK-KB11C-TM-001, pp. iv-3-4 and iv-3-5.

[2] M8138-0-1 PDP-11/70 engineering drawings, system status register B (ssrb), gates e10 and e11.

[3] MAINDEC-11-DEKBE-B-D, pp. 179-182, "test 115". Patch location 73616 from 12737 to 137.

Figure 1 - KB11-C processor hardware modification:
A control circuit in the PDP-11/70 memory management unit (module M8138-YA) was modified to allow more logically consistent execution of the *mfpi* instruction.

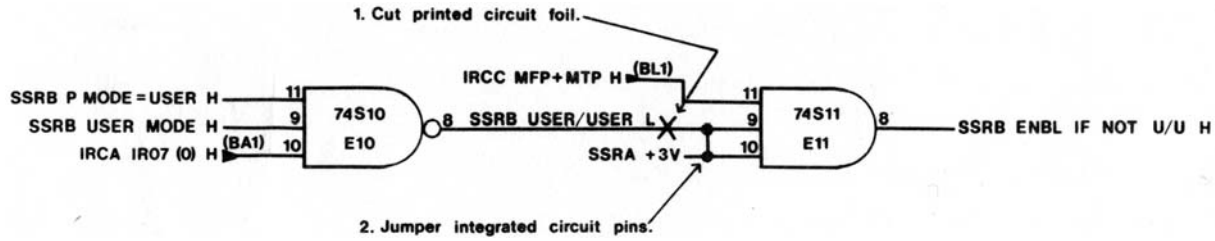

Figure 2 - Modified C-Library nargs.s listing:

```
/ C library -- nargs

/ Requires hardware mod so that mfpi instruction functions correctly in user mode
/ Thomas Ferrin   7-Dec-77

 .globl  _nargs
mfpi = 6500^tst

_nargs:
        mov     r5,-(sp)
        tstb    initf
        bne     5f
        incb    initf
        clr     r0
        mfpi    (r0)
        cmp     (sp)+,(r0)
```

```
        beq     1f
        negb    initf           / separate I&D
        br      4f
1:
        mov     $2,r0
        mfpi    (r0)
        cmp     (sp)+,(r0)
        beq     3f
        negb    initf           / separate I&D
        br      4f
5:
        blt     4f

/* -------------------------    */
3:
        mov     2(r5),r1        / pc of caller of caller
        mov     sp,r5
        clr     r0
        cmp     -4(r1),jsrsd
        bne     8f
        mov     $2,r0
8:
        cmp     (r1),tsti
        bne     1f
        add     $2,r0
        br      2f
1:
        cmp     (r1),cmpi
        bne     1f
        add     $4,r0
        br      2f
1:
        cmp     (r1),addi
        bne     1f
        add     2(r1),r0
        br      2f
1:
        cmp     (r1),jmpi
        bne     1f
        add     2(r1),r1
        add     $4,r1
        br      8b
1:
        cmp     (r1),jmpa
        bne     1f
        mov     2(r1),r1
        br      8b
1:
        cmpb    1(r1),bri+1
        bne     2f
        mov     r0,-(sp)
        mov     (r1),r0
        swab    r0
        ash     $-7,r0
        add     r0,r1
        add     $2,r1
        mov     (sp)+,r0
        br      8b
2:
        asr     r0
        mov     (sp)+,r5
        rts     pc

/* -------------------------    */
4:
```

```
        mov     2(r5),r1
        mov     sp,r5
        clr     r0
        mfpi    -4(r1)
        cmp     (sp)+,jsrsd
        bne     8f
        mov     $2,r0
8:
        mfpi    (r1)
        cmp     (sp),tsti
        bne     1f
        add     $2,r0
        br      2f
1:
        cmp     (sp),cmpi
        bne     1f
        add     $4,r0
        br      2f
1:
        cmp     (sp),addi
        bne     1f
        tst     (sp)+
        mfpi    2(r1)
        add     (sp),r0
        br      2f
1:
        cmp     (sp),jmpi
        bne     1f
        tst     (sp)+
        mfpi    2(r1)
        add     (sp)+,r1
        add     $4,r1
        br      8b
1:
        cmp     (sp),jmpa
        bne     1f
        tst     (sp)+
        mfpi    2(r1)
        mov     (sp)+,r1
        br      8b
1:
        cmpb    1(sp),bri+1
        bne     2f
        mov     r0,-(sp)
        mov     2(sp),r0
        swab    r0
        ash     $-7,r0
        add     r0,r1
        add     $2,r1
        mov     (sp)+,r0
        tst     (sp)+
        br      8b
2:
        tst     (sp)+
        asr     r0
        mov     (sp)+,r5
        rts     pc

 .data
jsrsd:  jsr     pc,*$0
tsti:   tst     (sp)+
cmpi:   cmp     (sp)+,(sp)+
addi:   add     $0,sp
jmpi:   jmp     0
bri:    br      .
```

```
jmpa:   jmp     *$0
initf:  .=.+1   / 0 == not initialized
                / 1 == initialized
                / 377 == init + I&D separated
```