#130-005-002-00

digital . INTEROFFICE MEMORANDUM

SUBJECT: techniques, cute ideas, and programming tricks for the PDP-11.

DATE: August 13, 1970

TO: PDP-11 List C
The Buffer

FROM: D. Knight

DEPARTMENT: Programming

The following is a collection of items concerning all sorts of odds and ends about programming the PDP-11.

1. One of the features of the PDP-11 is the ability to trap on various conditions such as illegal instructions, reserved instructions, power failure, etc. However, if ~~any of~~ the trap vectors are not loaded with meaningful information, the occurrence of any of these traps will cause unpredictable results. By ~~using~~ defining the following, it is possible to avoid these problems as well as gain meaningful information about any unexpected traps ~~which~~ that ~~may (and will)~~ occur. This technique, which makes it easy to identify the source of a trap, is to load each unused trap vector with:

```
. = trapaddress
.WORD   .+2,Ø
```

This will load the first word of the vector with the address of the second word of the vector (which contains a halt). Thus, for ~~instance~~ example, a halt at location 6 ~~implies~~ means that a ~~trap~~ through the vector at location 4 has occurred. The old PC and status may be examined by looking at the stack pointed to by register 6.

The trap vectors of interest are:

Vector location	halt at	vector meaning
4	6	bus error, illegal instruction, stack overflow, non-existent memory, non-existent device, word at odd address, etc.
1Ø	12	~~executed~~ reserved instruction
14	16	trace trap instruction (ØØØØØ3) executed or T-bit set in status word (used by ODT)

techniques, cute ideas,
and programming tricks
for the PDP-11.

(Cont'd)

Vector location	halt at	vector meaning
2Ø	22	IOT executed (used by IOX)
24	26	Power failure or restoration
3Ø	32	EMT executed (used by FPP-11)
34	36	TRAP executed

2. Cute instructions and tricks

Note: REG refers to a register in general.

a. TST (REG)+

Add two to a register. Use with care since condition codes are clobbered, the register's contents must be even, and the value cannot be outside addressable memory.

b. TST -(REG)

Subtract two from a register. Same cautions as (a).

c. CMP (REG)+,(REG)+

Add four to a register. Same cautions as (a).

d. CMP -(REG),-(REG)

Subtract four from a register. Same cautions as (a).

e.
```
JSR REG,XXX
BNE ABC

XXX: SEZ
     RTS PC
```

This is a very useful tool, in other words, use the condition codes to pass two valued parameters or flags upon subroutine returns.

techniques, cute ideas,
and programming tricks
for the PDP-11.

f. MOV COUNT,(R6)+

Pop an undesired item from the stack and set condition codes relative to count. Similar in operation to:

```
TST (R6)+       ;Pop stack
TST COUNT       ;test count
```

g. MOV @R7,REG
CLR REG

Move a one word instruction to a register and then execute it. (It might be useful somewhere.)

h. JMP and MOV instructions are similar in effect in certain cases. The major difference is that JMP doesn't affect the condition codes while MOV does. The following is a table of equivalences:

```
JMP    XXX        MOV    XXX,R7
JMP    @R6        MOV    R6,R7
JMP    @(R6)+     MOV    (R6)+,R7
  none            MOV    @(R6)+,R7
```

Notice that the MOV instruction as used above will allow one level of deferral deeper than a JMP. Possible applications include tables of jump addresses which may be indexed through.

3. A trap handler.

The following trap handler simulates a two word JSR using a one word trap. The low order byte of the trap instruction determines the table position where the jump address is found. The subroutine return should be an RTS R7.

```
;TRAP HANDLER (LOC 34)
TRAP34: MOV @R6,2(R6)          ;DESTROY TRAP STATUS
                               ;WITH RETURN ADDRESS
        SUB #2,@R6             ;CALCULATE TRAP POSITION
        MOV @(R6)+,-(R6)       ;GET TRAP INSTRUCTION PROPER
        ADD #TABLE—1Ø44ØØ,@R6 ;CALCULATE JUMP TABLE ENTRY
        MOV @(R6)+,R7          ;JUMP TO ROUTINE
;
TABLE:  XYZ                    ;THIS TABLE CONTAINS THE
                               ;ADDRESS OF EVERY ROUTINE
                               ;TO BE CALLED IN THIS MANNER.
```

techniques, cute ideas,
and programming tricks
for the PDP-11.

4. An example of recursion.

This problem is to scan through a string of characters containing parenthesis nesting to find matching parentheses.

E.G. - find the right paren in the following string which matches the first occurrence of a left paren.

B=A(EXP(A+B(15,12)),)

Recursive form:

```
A:   JSR   R7,@R7          ;CALL B RECURSIVELY
B:   MOVB  (REG)+,RØ       ;PICK UP NEXT CHARACTER
     CMPB  #'(,RØ          ;IS IT A "("?
     BEQ   A               ;YES, TRY AGAIN
     CMPB  #'),RØ          ;IS IT A ")"?
     BNE   B               ;NO, RELOOP
     RTS   R7              ;DO A POP JUMP
```

To use the above, B is entered with REG pointing to the character following the first open parenthesis. B is called with a JSR R7,B. The recursive form is 9 words long.

A non-recursive equivalent is:

```
B:   CLR   R1
B1:  MOVB  (REG)+,RØ
     CMPB  #'(,RØ
     BNE   B2
     INC   R1
B2:  CMPB  #'),RØ
     BNE   B1
     DEC   R1
     TST   R1
     BGE   B1
     RTS   R7
```

The non-recursive example is 13 words long.

techniques, cute ideas,
and programming tricks
for the PDP-11.

5. Another cute trick

```
INCB  @(R1)      (and)      INCB  @(R1)+
TSTB  @(R1)                 TSTB  @-(R1)
```

are equivalent except that the first is four words long and the second is two words long.

6. Pseudo-assembly options

The following example shows how to make assembly options available or an assembler not having explicit optional assembly pseudo-ops. This neat nasty depends on the fact that when two pieces of code are loaded in the same place, the last one (usually!) takes precedence.

Assume the existence of two short subroutines A and B.

```
A:  INCB @R1                B:  INC   R5
    TSTB @R1                    MOVB  SP, RØ
    BPL  .-4                    RTS   R5
    MOVB @RØ,R2
    RTS  R5
```

Assume that subroutine B is to be made optional. The program may be arranged thus:

```
OPT  =  -1
TMP  =  .
B:   INC     R5
     MOVB    SP, RØ
     RTS     R5
LG   =       .-TMP
.    =       LG & OPT +TMP
A:   INCB    @R1
     TSTB    @R1
     BPL     .-4
     MOVB    @RØ,R2
     RTS     R5
```

techniques, cute ideas,
and programming tricks
for the PDP-11.

6. Pseudo-assembly options (Cont'd)

If OPT is set to -1, subprogram B will be assembled normally,
If OPT is set to 0, B will be overlaid by A.

Cautions:

1) This is not a true conditional assembly thus symbol redefinition may not be done (i.e. two subprograms named B) without some effort.

For example:

```
OPT   =   -1
B:                    ; label definition
TMP       =.
.         =   xx & -OPT +TMP2
              INC    R5
              MOVB   SP, RØ
              RTS
LG        =           .-TMP
          =           LG & OPT+TMP
          INCB        R1
          .
          .
          .
```

2) The actual amount of code to be loaded will not change, only location where it will be loaded.

techniques, cute ideas,
and programming tricks
for the PDP-11.

7. I/O buffering with IOX.

This is a relatively obscure method to double buffer input and output. There are two input buffers (I1, I2) and two output buffers (O1, O2). This is a process which processes an input buffer and in the process loads an output buffer. The process uses pointers to the buffers and buffer headers which are set by the I/O routines. Thus the process does not know (or care) which buffer is being worked upon.

```
PC=%7
BEGIN:   (do I/O resets, inits, etc.)
         .
         .
         .
         IOT                ;read into I1
         .WORD I1
         .BYTE READ,INSLOT
         MOV  #A,-(6)       ;initialize stack
B:       JSR  PC,@(6)+      ;do I/O
         .
         .    Perform processing.
         .
         BR   B             ;do it again, etc.
;
;END OF MAIN LOOP
;
;I/O CO-ROUTINES
;
A:       IOT                ;read into I2
         .WORD I2
         .BYTE READ,INSLOT
         .
         .    set parameters to process I2,O1.
         .
         JSR  PC,@(6)+      ;return to process
         IOT                ;write from O1
         .WORD O1
         .BYTE WRITE,OUTSLOT
         IOT                ;read into I1
         .WORD I1
         .BYTE READ,INSLOT
         .
         .    set parameters to process I2, O2.
         .
```

techniques, cute ideas,
and programming tricks
for the PDP-11.

```
JSR  PC,@(6)+      ;return to process
IOT                ;output from O2
.WORD O2
.BYTE WRITE,OUTSLOT
BR.  A             ;go read into I2
```

The above is a good example of the usage of the co-routine call form of JSR:

```
JSR  PC,@(6)+
```

which does a jump to the address specified on top of the stack and replaces that address with the new return address.

8. More cute tricks

   a. CMPB (R6)+,(REG)+

      will increment R6 by two and increment REG by one. This can be extended to:

```
                          1000    400
CMPB (R6)+,-(REG)         1002    377
CMPB -(R6),(REG)+         1000    400
CMPB -(R6),-(REG)          777    377
```

      This trick depends on the fact that the stack register is always incremented or decremented by two in auto-increment/decrement mode. The same cautions as in section 2a apply. Also, REG is in the range Ø-5 only.

   b. The above can be generalized slightly.

```
CMPB (REG1)+,(REG2)+
CMP  (REG1)+,(REG2)+    , ad nauseum
```

      This allows two separate registers to be incremented (or decremented, e.g. CMP (REG1)+,-(REG2)) at the same time by the same value in one instruction. This is sometimes useful in table scanning algorithms, etc. The same cautions apply as in section 2a.