CHAPTER 6
ASSEMBLER PROGRAMMING UNDER ETOS

6.1 INTRODUCTION

This chapter contains information about assembly language programming. Coding is performed directly at the machine level. To develop assembler programs, you must utilize one of the assemblers distributed with the system (PAL8, SABR, RALF or MACREL). Your program must be developed under OS/8, but it may be executed under COS as a core image file. All "V" extension programs under COS (e.g., SYSGET, DFDIR) were developed in this manner. If you write a program which will operate under COS, make sure that you do not utilize an operating system dependent feature of OS/8, such as the user service routines.

All instructions, which may be used in assembly language coding, are listed in this chapter. Most instructions work identically (from the program's standpoint) to the way that they do on a stand-alone PDP-8. If you utilize a real time task which clears user mode, all input/output transfers (IOTs) work exactly as they do on a stand-alone PDP8. Due to the close correlation between ETOS IOT's and stand-alone PDP8 IOTs, the following manuals may be utilized for additional information: Introduction to Programming, the OS/8 Handbook and the OS/8 Software Support Manual. All of these manuals are reference books only. To learn assembler, there is an excellent student workbook and teacher's edition set called PDP-8 Programming, written by R. W. Southern. These books may be ordered from the Algonquin College Bookstore, 1385 Woodroffe Avenue, Ontario, Canada K26 1V8.

6.2 TERMINAL IOTs

6.2.1 KCF 6030 NOP

This instruction is ignored.

6.2.2 KSF 6031 SKIP ON NON-EMPTY INPUT BUFFER

If there are characters in the input buffer, skip the next instruction. If the input buffer is empty, and the next instruction is JMP .-1, the program ceases running until a break character has been entered.

6.2.3 KCC 6032 CLEAR AC AND ADVANCE IF CHARACTER READ

Clear the AC and advance the input buffer pointer past the current character, if the current character has already been read by a previous KRS. If the current character has not yet been read, the pointer is not advanced.

6.2.4 KCL 6033 CLEAR KEYBOARD BUFFER

Clear the keyboard buffer. All characters in the buffer are discarded.

6.2.5 KRS 6034 OR INPUT CHARACTER WITH AC

Inclusively OR the current input character with the AC; do not advance the buffer pointer beyond the current character. If no characters are in the input buffer, return to the program with the AC unchanged. Successive KRS's with no intervening KCC's will OR the same character.

6.2.6 KRB 6036 READ KEYBOARD BUFFER

Jam transfer the next input character to the AC, and advance the buffer pointer beyond that character. If no characters are in the input buffer, put the program in input and wait until a break character is entered. Note that when one uses KRB to read characters, "KSF; JMP .-1" is unnecessary.

6.2.7 GETSTAT 6037 GET STATUS WORDS

Read the status words KSTAT, BREAK, ^C^C restart address, and ^P restart address into the Instruction Field starting at the location whose address is in the AC. The AC is cleared upon return. For more information on the meaning of the status words, KSTAT and BREAK, see 3.7.6 and 3.7.5. The ^C^C restart address is the location in field Ø to dispatch to, if you type ^C^C. The ^P restart address is the location in field Ø to dispatch to, if you type ^P. After saving the current status with this IOT, you may wish to change the status via the IOT SETSTAT (see 6.2.14). A sample calling sequence is contained in Figure 6-1.

Figure 6-1
Sample Use of GETSTAT

```
GETSTAT=6037
CLA     CLL
TAD     (ARGLST          /POINT TO ARGLST
GETSTAT                  /SAVE THE STATUS WORDS
.
.
.
ARGLST, 0000             /SAVE KSTAT WORD HERE
        0000             /SAVE BREAK MASK HERE
        0000             /SAVE ADDRESS TO GO TO ON DOUBLE ^C
        0000             /SAVE ADDRESS TO GO TO ON ^P
```

6.2.8 TSF 6041 SKIP

Skip the next instruction unconditionally.

6.2.9 TCF 6042 NOP

This instruction is ignored.

6.2.10 TCL 6043 CLEAR OUTPUT BUFFER

Initialize the teleprinter and immediately clear the output buffer. This is useful for OS/8, for instance when ^C is entered, so that output, which may be up to 200 characters ahead, ceases immediately.

6.2.11 TPC 6044 LOAD PRINTER BUFFER

Put the character in the AC into the output buffer for printing. If the output buffer is full, put the program in input/output, wait (stop it running) until the printer is within a system defined number of characters of emptying the buffer (see 1.2). Restarting the job before output finishes helps insure that output will be continuous. The AC is not cleared.

6.2.12 TSK 6045 SKIP ON TERMINAL FLAG

Skip the next instruction unconditionally.

6.2.13 TLS 6046 LOAD BUFFER PRINTER

Exactly the same as TPC.

6.2.14 SETSTAT 6047 SET STATUS WORDS

Set the status words KSTAT, BREAK, ^C^C restart address, and ^P restart address using the information from the Instruction Field, starting at the location whose address is in the AC. The AC is cleared upon return. For more information on the status words KSTAT and BREAK, see 3.7.6 and 3.7.5. The ^C^C restart address

is the location in field Ø to dispatch to, if you type ^C^C. The ^P restart address is the location in field Ø to dispatch to if you type ^P. Before executing this IOT, you may wish to save the current status via the IOT GETSTAT (see 6.2.7). A sample calling sequence is contained in Figure 6-2.

Figure 6-2
Sample Use of SETSTAT

```
KSTAT=WWWW
BREAK=XXXX
CTCRES=YYYY
CTPRES=ZZZZ
SETSTAT=6Ø47
CLA     CLL
TAD     (ARGLST          /POINTER TO ARGLST
SETSTAT                  /NEW STATUS WORDS
.
.
.
ARGLST, KSTAT            /ETOS STATUS
        BREAK            /BREAK MASK
        CTCRES           /ADDRESS IN FIELD Ø TO GO TO ON
                         /DOUBLE ^C
        CTPRES           /ADDRESS IN FIELD Ø TO GO TO ON ^P
```

6.3 EXTENDED MEMORY IOTs

All references to "data field" and "instruction field" refer to the virtual data and instruction fields. There is no way to determine which physical fields your program is using.

6.3.1 CDF 62n1 CHANGE DATA FIELD

Change to Data Field n. This works from the program's standpoint exactly the way it does on a stand-alone PDP-8. However, a CDF to a non-existent field results in a CDF Ø. You should be sure you have allocated enough memory or unpredictable results will occur.