

Chapter 5.1 Program Development and Debugging

Objectives of this Chapter

Having studied this chapter you will be able to:

- Use labels within 80286 programs
- Understand the use of the EQU assembler directive
- Explain the need for program debugging
- Use the following PAT program debugging facilities:
 - Register Examine/Modify
 - Breakpoints
 - Single Step
 - Disassemble
 - Keypad Restart

Introduction

Recall that the assembler is the program which converts a program written in mnemonics (source program) into a machine code program (object program).

Most assemblers not only translate mnemonics into machine code but also allow the programmer to use **labels** in place of immediate data or addresses.

Use of Labels

Labels can be used within a source file in place of hexadecimal values or addresses. The "value" of the label is defined using an EQU assembler directive.

For example:

```
fstnum equ $4600 defines "fstnum" as 00004600H
```

Create the source program below using Merlin:

```
ORG 0100H ;Defines the start address for
;object code as 0080:0100H
MEMORY EQU 1000H ;Defines "MEMORY" as 1000H
FIRST EQU 12H ;Defines "FIRST" as 12H
SECOND EQU 34H ;Defines "SECOND" as 34H
MOV AL, FIRST ;Moves the value defined by the
;label "FIRST" into the AL
;Register
ADD AL, SECOND ;Adds the value defined by the
;label "SECOND" to the AL
;Register
MOV BYTE PTR DS:MEMORY, AL ;Saves the result in the memory
;location defined by the label
;"MEMORY"
MOV BX, 0000H ;Returns to PAT system
MOV AH, 04H
INT 28H
```

Assemble the object program and produce a listing on the PC screen or your printer by pressing Alt-L or Alt-P respectively. Examine the listing and you will find that the assembler has produced an object code program which inserts "12H" in place of "FIRST", "34H" in place of "SECOND" and "1000H" in place of "MEMORY".

There are a number of rules for the use of labels:

- 1 Labels must begin with a letter but may include numbers. So "NUMB9" is permissible, whereas "9NUMB" is not acceptable. Lower or upper case letters may be used.
- 2 Labels are limited to 8 characters.
- 3 A label must **not** be a reserved word or a 80286 mnemonic. A list of reserved words can be found in the D2000 Cross Assembler User Manual.

You have seen how labels can be used as operands in place of data and addresses in the previous example. Labels may also be used **before** the operator to "mark" points within a program for future reference. Create the source program below using Merlin:

```

                                ;Defines the start address for
                                ;object code as 0080:0200H
                                ;Defines "VAL1" as 2222H
                                ;Defines "VAL2" as 3333H
                                ;Defines "MEM1" as 2000H
ORG 0200H

VAL1 EQU 2222H
VAL2 EQU 3333H
MEM1 EQU 2000H

BEGIN: MOV AX, VAL1                ;Moves the value defined by the
                                        ;label "VAL1" into the AX
                                        ;Register
                                        ;Jumps forward to the label "NEXT"
        JMP NEXT

LAST:  MOV DS:MEM1, AX              ;Saves result in location "MEM1"
        MOV BX, 0000H              ;Returns to PAT System
        MOV AH, 04H
        INT 28H

NEXT:  ADD AX, VAL2                ;Adds the value defined by the
                                        ;label "VAL2" to the AX
                                        ;Register
                                        ;Jumps backward to the label "LAST"
        JMP LAST

```

Here we have used a JUMP instruction. This transfers program execution to a point other than the next location in sequence. Jumps can be forward or backward. Notice that the labels "BEGIN", "LAST" and "NEXT" have not been defined using an EQU directive. Each has a colon (:) to indicate an **address** to the assembler.

The arrows are **not** part of the source program. These are just to help you understand how the program works.

Assemble the object program and produce a listing on the PC screen or your printer as before. You should now find that the object code program has "2222_H" in place of "VAL1", "3333_H" in place of "VAL2" and "2000_H" in place of "MEM1".

Notice how the instructions "JMP NEXT" and "JMP LAST" are translated to "JMP 0211" and "JMP 0206" respectively.

So, you have seen two ways of defining an address label:

- 1 Using an EQU assembler directive (eg "MEM1" above).
- 2 Inserting the label just before an instruction (eg "LAST" above).

There are other ways of defining labels which will be introduced as you progress through this text.

Program Debugging

Debugging is the process of finding and then rectifying faults within programs. Simple techniques include checking through a source program for typing errors, examining the contents of memory, etc.

However, some problems will require the use of more sophisticated debugging tools. The D2000 80286 Development System provides a number of debugging facilities in Terminal Mode:

Recall that the "M" command allows the contents of memory locations to be inspected. Also, single or multiple bytes within memory can be modified using the "C" command. This command also allows ASCII codes to be entered directly.

Register Examine/Modify

The "R" command allows 80286 registers to be examined and modified. If you now enter "R" from Terminal Mode, the display will show the contents of all 80286 registers thus:

```

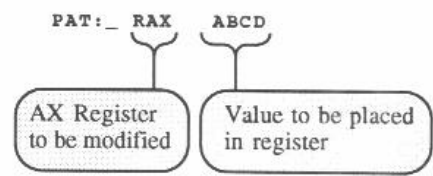
IP  AX  BX  CX  DX  SI  DI  BP  DS  CS  ES  SS  SP  FL
0100 0000 0000 0000 0000 0000 0000 0000 0080 0080 0080 0F80 0800 0046
0080:0100 0000          ADD    [BX+SI],AL
    
```

The first line lists the 80286 registers. We have already seen the several of these. The functions of the other registers will be explained in later chapters.

The second line shows the contents of each register. If you have just switched your system on you will probably find most registers contain 0000H.

The last line shows the current contents of the Program Counter Register, the machine code instruction at that location and the assembly language mnemonics for that instruction.

The contents of a register can be changed by specifying the register and then the required value. For example, at the "PAT" prompt, enter:



This will change the word in the AX Register to ABCDH. Use the "R" command to examine the contents of the 80286 registers again and you should find:

```

IP  AX  BX  CX  DX  SI  DI  BP  DS  CS  ES  SS  SP  FL
0100 ABCD 0000 0000 0000 0000 0000 0000 0080 0080 0080 0F80 0800 0046
0080:0100 0000          ADD    [BX+SI],AL
    
```

Modify the contents of the BX Register to 1234H by entering "RBX 1234" and examine the registers again. You should now find:

```

IP  AX  BX  CX  DX  SI  DI  BP  DS  CS  ES  SS  SP  FL
0100 ABCD 1234 0000 0000 0000 0000 0000 0080 0080 0080 0F80 0800 0046
0080:0100 0000          ADD    [BX+SI],AL
    
```

Breakpoints

A breakpoint can be set within a program. When the breakpoint is reached, the program is halted and the contents of registers and memory locations at that point can be examined and/or modified. This is most easily explained by reference to an example:

Exercise 5

Create the simple source program shown below and then assemble the object program:

```

ORG 0300H                ;Defines the start address for
                        ;object code as 0080:0300H

MOV AX,1234H             ;Moves the value 1234H into the
                        ;AX Register
MOV BL,56H               ;Moves the value 56H into the
                        ;BL Register
MOV CX,789AH            ;Moves the value 789AH into the
                        ;CX Register
MOV BX,0000H            ;Returns to PAT system
MOV AH,04H
INT 28H
    
```

Once the object program can be assembled without any errors, save it under a suitable filename (eg: BRKTST1.ASM) Next produce a listing of this program on your printer and also transfer the program to PAT. You should see from your listing that the second instruction is at location 0080:0303H and the third at location 0080:0305H. Now, a breakpoint is easily set by entering "B" followed by the required address. So, to set a breakpoint at location 0080:0303H for instance, enter "B 0303" at the "PAT:" prompt.

Run the program in the usual way by entering "G 0300". The screen will then display:

```

*** Breakpoint Hit ***
IP  AX  BX  CX  DX  SI  DI  BP  DS  CS  ES  SS  SP  FL
0303 1234 0000 0000 0000 0000 0000 0000 0080 0080 0080 0F80 0800 0046
0080:0303 B356                MOV  BL,56
    
```

The program has been executed from the start but has been halted **before** the instruction at 0080:0303H. This display shows that the contents of the AX Register are 1234H after the **first** instruction has been executed. Notice that IP Register has the value 0303H. This is the location of the next instruction to be executed.

The bottom line of the display shows the machine code and mnemonics for the **next** instruction to be executed. A breakpoint can be removed by entering "K"

followed by the required address. So, to clear the breakpoint you have just set, enter "K 0303".

Up to 8 separate breakpoints can be set. The command "K*" will clear all breakpoints. Now set a breakpoint at location 0080:0305H. Run the program again from 0080:0300H and the screen will show:

```

*** Breakpoint Hit ***
IP  AX  BX  CX  DX  SI  DI  BP  DS  CS  ES  SS  SP  FL
0305 1234 0056 0000 0000 0000 0000 0000 0080 0080 0080 0F80 0800 0046
0080:0305 B99A78          MOV  CX,789A
    
```

Notice that now the second instruction has been executed, the third instruction is the next to be executed. Clear all breakpoints using the command "K*".

Single Step

The Single Step or "Trace" facility allows a program to be stepped through, instruction by instruction.

At each step the contents of 80286 registers are displayed on the screen.

The program example used for the explanation of breakpoints on the previous page can also be used to explain the action of single step. If this program is not already loaded, load it from disk (recall the filename eg: BRKTST1.ASM) and transfer to PAT.

Exercise 6

Use the source program you created for Exercise 5 to produce an object program and transfer it to PAT.

To single step from 0080:0300H, enter "T 0300".

The first instruction will be executed and the screen will display the contents of each register thus:

```

IP  AX  BX  CX  DX  SI  DI  BP  DS  CS  ES  SS  SP  FL
0303 1234 0000 0000 0000 0000 0000 0000 0080 0080 0080 0F80 0800 0046
0080:0303 B356          MOV  BL,56
    
```

As with breakpoints, the screen shows how the first instruction has affected the Registers. Each time you press the Return (Enter) key, the next instruction is executed and the contents of each register displayed at that point. Further ways of using the Single Step facility are described in the PAT User Manual.

Disassemble

Recall that **assembly** is the process of producing machine code object code from a source program. **Disassembly** is the reverse process. A 80286 Assembly Language mnemonic listing is produced from a machine code object program. The disassembler cannot, of course, reproduce comments but a source listing is far easier to understand than machine code.

So, a **disassembler** takes object code and presents it in assembly language form. This can be a very useful tool if source program files have been lost or are otherwise unavailable. The disassembler command is "D". To disassemble a program in PAT memory, enter "D", followed by the start address. For example, to disassemble the first few instructions of the Applications Module Demonstration Programs held in EPROM:

Enter "D F000:6000" (this is the start address for the demonstration programs), the PC screen will then display a disassembled listing thus:

```

PAT: D F000:6000
F000:6000 E82F00      CALL    6032
F000:6003 803E1200AA     CMP     [0012],AA
F000:6008 741C          JE      6026
F000:600A C6061200AA     MOV     [0012],AA
F000:600F E86600      CALL    6078
F000:6012 C606140000     MOV     [0014],00
F000:6017 C606080015     MOV     [0008],15
F000:601C C606090015     MOV     [0009],15
PAT:
    
```

You should recognize MOVE instruction mnemonics shown in the disassembled listing.

The length of the code to be disassembled can be specified by adding a semi-colon and the number of instructions to be disassembled to the command line. For example, to disassemble 20H instructions from F000:6000H:

```
D F000:6000;20
```

Keypad Restart

This facility allows control to be returned to the PAT keypad. Simply enter "P" and press the "Return" ("Enter") key. The PAT display shows:



This indicates that control has reverted to the PAT keypad. If you now press the PAT reset key, the PAT display will blank and the Terminal software is again in control.

Here are some of the problems for which you have already written 80286 machine code programs. Rewrite these using Merlin and then assemble. Verify correct operation by transferring to PAT and executing.

Practical Assignment

- 2 Write a program, starting at location 0080:0100_H, which will add the bytes 4D_H and 67_H, saving the result in memory location 0080:1000_H.

Student Assessment 5

1. Consider the 80286 Assembly Language program shown below.

```
VAL1      EQU 1234H
VAL2      EQU 5678H
          ORG 0500H
START:    MOV AX, VAL1
          MOV BX, VAL2
          HLT
```

The value loaded into the BX Register is:

- a 0500H
 - b 1234H
 - c 5678H
 - d 68ACH
2. Debugging is often necessary because user programs:
- a may require the registers and memory locations used to be specified.
 - b may not be entirely correct when first executed.
 - c can change the contents of ROM.
 - d can only run for some of the time.
3. The Terminal Mode key sequence required to set the DX Register to FEDCH is:
- a DDX FEDC
 - b RDX FEDC
 - c SDX FEDC
 - d ZDX FEDC
4. To insert a breakpoint at 0080:0204H, the required Terminal Mode key sequence is:
- a B 0200
 - b B 0202
 - c B 0204
 - d B 0206
5. The Terminal Mode key sequence required to disassemble the program which starts at location F000:8200H is:
- a C F000:8200
 - b D F000:8200
 - c M F000:8200
 - d T F000:8200