

Section 2

Introduction to 80286 Programming

Chapter 2.1 Introduction to 80286 Machine Code Programming

Objectives of this Chapter

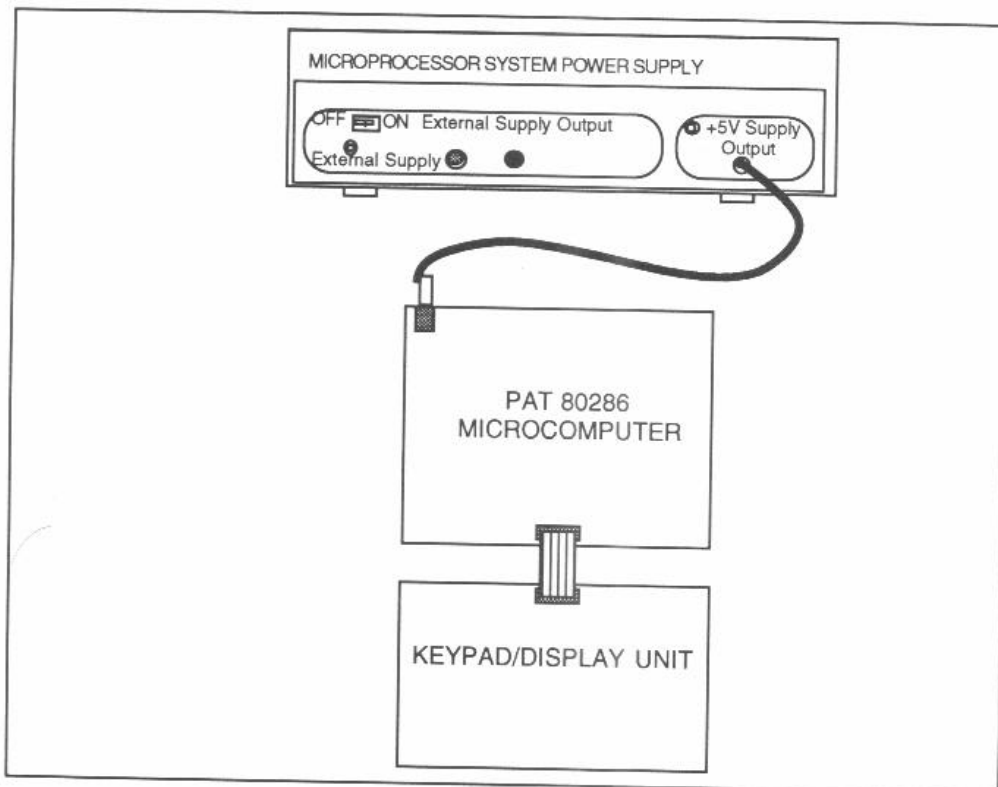
Having studied this chapter you will be able to:

- Explain the concept of Memory Segmentation
- Examine and modify the contents of PAT memory
- Explain the need for:
machine language
machine code
assembly language
- Interpret the PAT Memory Map
- Use the keypad to key in and execute a machine code program

Introduction

Connect the power supply, keypad/display and PAT board using the guidelines shown below. The Applications Module will not required initially.

- Use the special key provided to remove the cover from the Switched Faults.
- Check that all of the switches are set to the **off** position.
- Connect the Keypad/Display Unit to the PAT Microcomputer Board.
- Ensure that the power supply is switched off
- Connect the power supply to the PAT Microcomputer Board as shown below.

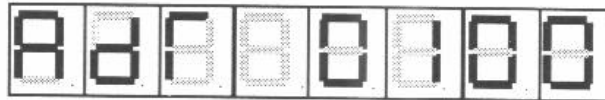


The power may now be switched on.

The PAT display will now show:

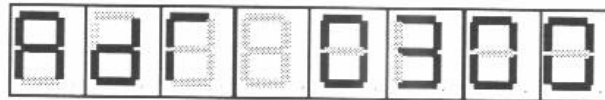


Press **M** and the display will change to:



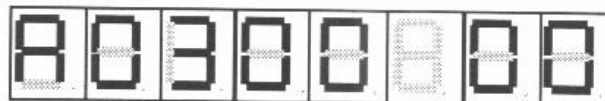
This stands for "Address 0100_H" so this means that memory address 0100_H is currently selected. Pressing any of the hexadecimal keys will change the selected memory address.

Press **0 3 0 0** and the display will show:



This shows that you have selected memory location 0300_H.

Press **M** again and the display will change to:



This indicates that the contents of location 0300_H are 00_H. You can now use the hexadecimal keys to change the contents of this location to any desired value. Change the contents of location 0300_H to AB_H.

Switch off the power for a few seconds and then switch it on again.

Examine the contents of location 0300_H again. You will find that the data AB_H has been lost. This is because location 0300_H is within RAM. Recall that RAM is **volatile** and so its contents are **lost** when the power is switched off.

When you selected location 0300_H you were actually selecting location 0300_H **in the current segment**. This concept is explained on the next page.

Memory Segmentation

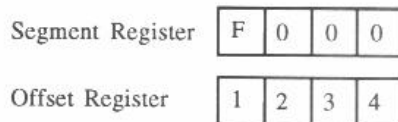
The 80286 microprocessor has an Address Bus which is 24 bits wide. The PAT microcomputer actually only uses 20 bits of the 24 available so it is possible to access 2^{20} (ie over 1 million) locations.

However, addresses are **not** defined directly by a 20 bit address register. Instead, **two** 16 bit registers are combined to produce the address:

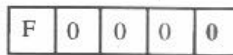
The **Segment Register** defines the start of a 16 byte memory segment. This is often called the 'Segment Base'.

The **Offset Register** defines the address **within** the Segment.

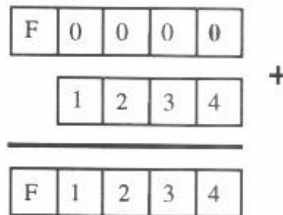
For example: Suppose the Segment Register contained $F000_H$ and the Offset Register 1234_H thus:



The 80286 will first **multiply** the contents of the Segment Register by 16_{10} (ie. 10_H):



This value and the contents of the Offset Register are then **added** to produce the **Physical Address**:



So the **Physical Address** in this case will be $F1234_H$.

The reasons for using Memory Segmentation rather than a single large address register are:

- 1 The size of microprocessor registers can be limited to 16 bits.
- 2 Different segments can be assigned to different programming tasks. This allows several programs to be run concurrently (**multitasking**).

The PAT operating system allows you to modify both the Segment and Offset Registers from the keypad.

Use the keypad to select location 0300_H again and the display will show:

A	D	E	E	0	3	0	0
---	---	---	---	---	---	---	---

Press **S** and the display will change to:

S	E	G	E	0	0	8	0
---	---	---	---	---	---	---	---

This shows that the current segment is 0080_H. Change the current segment to F000_H using the hexadecimal keypad and press the **M** key twice. The display will show the contents to be 0E_H:

A	0	3	0	0	E	0	E
---	---	---	---	---	---	---	---

Switch off the power for a few seconds and then switch it back on again. Examine location 0300_H in segment F000_H again and you will find that it is **unchanged**. This is because it is within **ROM**. Recall that ROM is **non-volatile**.

Try to change the contents of this location using the hexadecimal keys. You will find that you cannot modify this location. Recall that the contents of ROM cannot be changed by the user.

The **+** and **-** keys can be used to select the next or previous location in memory respectively. Use the **+** key to step forward through a few locations. Notice that the words will usually differ from one address to another.

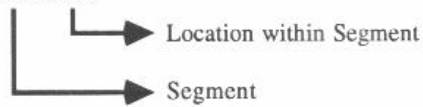
The **+** and **-** keys can also be used to select the next or previous segment, when in the 'SEG' symbol is showing.

When addresses are written down, it is conventional to show the segment and the offset separated by a colon (:) thus:

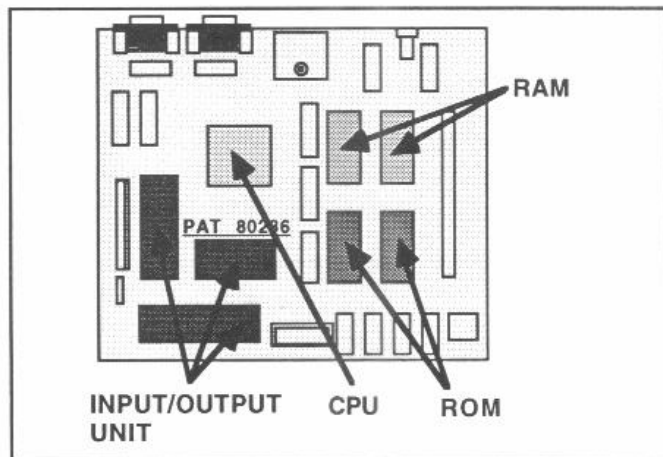
Segment :Offset

For example:

F000:0300



You can identify the ROM, RAM and other devices on the PAT board using the diagram shown below.



Notes:

.....

.....

.....

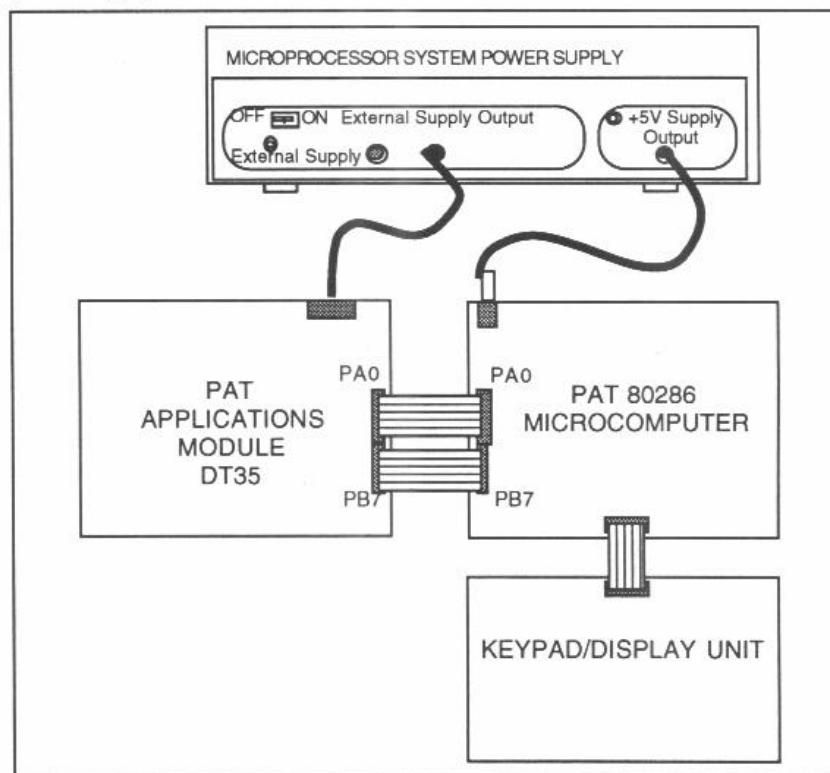
.....

.....

Programming the Microprocessor

You will already have used the PAT microcomputer for the Applications Module demonstration programs.

These programs were previously stored in the on-board EPROMs. Now you can key in a short program into RAM for the Applications Module. Firstly, switch the power off and connect the PAT and Applications Module to the power supply as shown below:

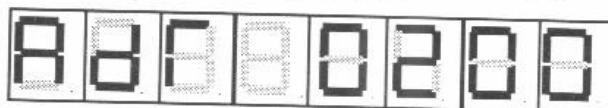


Switch the power on. The display will show



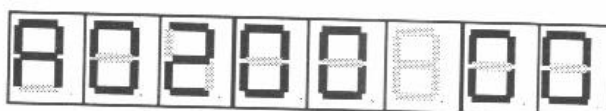
The PAT may be programmed by placing the correct machine code in successive memory locations. You are now going to key in a machine code program. Select memory address 0200_H thus:

Press **M** followed by: **0** **2** **0** **0** and the display will show:



Press **S** once and check that the current segment is 0080_H.

Now press **M** twice to enter the memory modify mode and the display will show:



The data at address 0200_H will probably be 00_H if you have just switched on. Change the word at 0200_H to B0_H by pressing: **B** **0** .

Now press the **+** key to move on to address 0201_H and change the data at this address to 03_H by pressing: **0** **3** .

Use the **[+]** and hexadecimal keys to enter the rest of this machine code program thus:

[+]	[E]	[6]
[+]	[8]	[6]
[+]	[B]	[0]
[+]	[5]	[5]
[+]	[E]	[6]
[+]	[9]	[2]
[+]	[E]	[9]
[+]	[F]	[B]
[+]	[F]	[F]

Use the **[M]**, **[+]**, **[-]** and hexadecimal keys to check that the program has been entered correctly. The list below will make this easier:

Location	Contents
0080:0200	B0
0080:0201	03
0080:0202	E6
0080:0203	86
0080:0204	B0
0080:0205	55
0080:0206	E6
0080:0207	92
0080:0208	E9
0080:0209	FB
0080:020A	FF

It is not important at this stage to understand exactly how this program works. It will display the hexadecimal value 55_H as a binary pattern on the Applications Module Port Monitor (labelled D0 to D7).

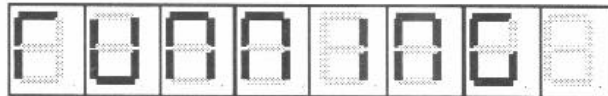
You are now ready to run this program.

Press the **G** key **once** and the display will show:



This is the address within the current segment from which program execution will begin. Change this to 0200_H by keying in: **0 2 0 0**.

Press the **G** key once again and the program will run. The display will show that the program is running thus:



Examine the Applications Module Port Monitor LEDs. You should find that the upper 8 LEDs (labelled D0 to D7) show:

D7	D6	D5	D4	D3	D2	D1	D0	● = lit
O	●	O	●	O	●	O	●	O = unlit

This is 0101 0101₂ (55_H) - the value which was to be displayed.

If you do not see this output on the port monitor, check the following:

- Has the machine code been correctly entered ?
- Is the Applications Module connected to the PAT ?
- Is the power correctly connected to the Applications Module ?

Press reset to stop the program. Change the word at address 0205_H and run the program again. You will see your new binary value on the LEDs. Experiment with several other words at address 0205_H.

Student Assessment 1

1. If the Segment Register contains F800H and the Offset is 7200H, the Physical Address will be:
 - a 016A00H
 - b 017000H
 - c 0FF200H
 - d 105200H

2. The data at PAT memory address F000:11F9H is:
 - a 00H
 - b 2EH
 - c 5DH
 - d FFH

3. The type of memory which **may** have its contents changed by the user is:
 - a Read Only Memory
 - b Write Only Memory
 - c Read/Write Memory
 - d Write Once Memory

4. Giving instructions to the microcomputer in hexadecimal form is called:
 - a Assembly
 - b Coding
 - c High Level Programming
 - d Machine Code Programming

5. The area of PAT memory available for User Programs is:
 - a 00000H to 0FFFFH
 - b 00000H to 007FFH
 - c 00800H to 0F7FFH
 - d 0F800H to 0FFFFH

Chapter 2.2 Writing Machine Code Programs

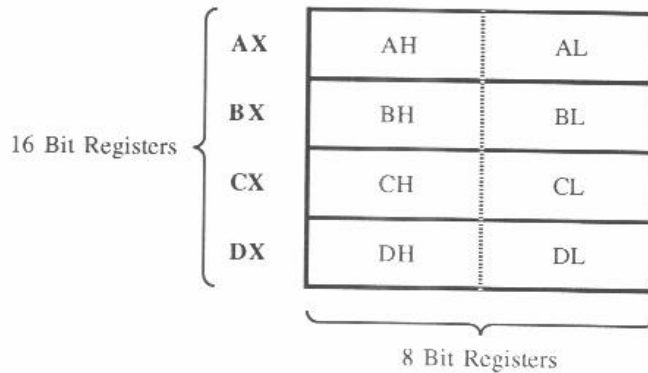
Objectives of this Chapter

Having studied this chapter you will be able to:

- Describe the basic function of the 80286 General Purpose Registers
- Explain the action of fundamental microprocessor instructions:
 - Move
 - Add
 - Increment
 - Decrement
 - Jump
- Describe the functions of operators and operands
- Code an Assembly Language program
- Enter an Assembly Language program into PAT Memory
- Produce a flowchart and write an assembly language program to solve a simple problem.

Introduction

The 80286 has four General Purpose 16 bit Registers. Each of these is made up from two 8 bit registers; one for the **high** (most significant) **byte** and one for the **low byte**. The 80286 can therefore operate on 8 bit (byte) or 16 bit (word) data.



These Registers are used to hold data for arithmetic and logical operations within programs. Registers may also be used for temporary data storage. Being internal, these registers can be accessed in a much shorter time than external memory devices.

Each of the General Purpose Registers is also assigned certain specific functions. We shall examine these as we progress through this manual.

Instruction Sets

Initially, we shall only consider a few 80286 instruction types:

Move

This will **duplicate** (or **load**) the contents of one register or memory location in another register or memory location.

Add

This will **add** the contents of one register or memory location to those of another register or memory location.

Increment

This will **add one** to the contents of a register or memory location.

Decrement

This will **subtract one** to the contents of a register or memory location.

Jump

This will **always** cause program execution to continue from a specified location **other than** the next location in sequence.

Operators and Operands

Microprocessor instructions can be thought of consisting of **two** distinct parts:

Operator

This is the part of an instruction which defines the operation which must take place. For example "Move".

Operand

This provides any additional information necessary for the microprocessor to complete the instruction. For example if the operator is "Move..." and the operand "...AX Register to BX Register". Then the overall instruction would be: "Move AX Register to BX Register".

Some instructions do **not** require an operand.

Simple Programs

As mentioned in the last chapter, the microprocessor can only interpret instructions given in **binary** form. These instruction codes are referred to as **opcodes**. The instruction set will include the opcode for every instruction.

The instruction codes for the 80286 microprocessor are given in the 80286 Programmer's Reference Manual.

Turn to Appendix B, page 70 of the 80286 Programmer's Reference Manual. This shows the opcodes for every type of MOVE Instruction. This page also shows that MOVEs can take place with 8 bit (byte) or 16 bit (word) data.

Notice that there are many forms of the MOVE instruction. However, we shall initially only consider two types of MOVE:

- 1 MOVEs between a register and another register.
- 2 MOVEs between a register and a memory location.

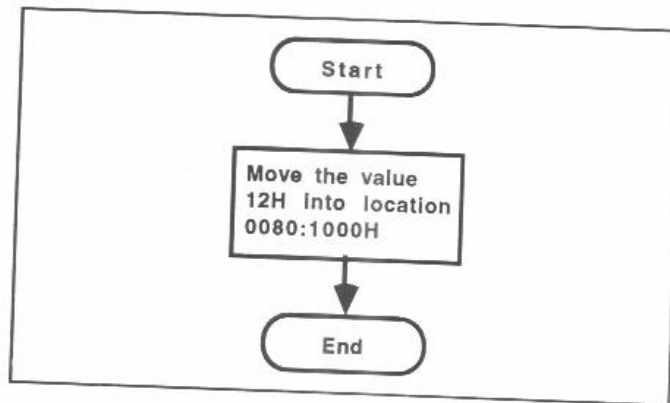
The 80286 Programmer's Reference Manual also indicates that MOVEs can take place with 8 bit (byte) or 16 bit (word) data.

Exercise 1

Write a program which will place the value 12H in memory location 0080:1000H.

Solution:

Although this is a very simple program, it is good practice to first draw a flowchart:



Essentially the assembly language program will only require one instruction:

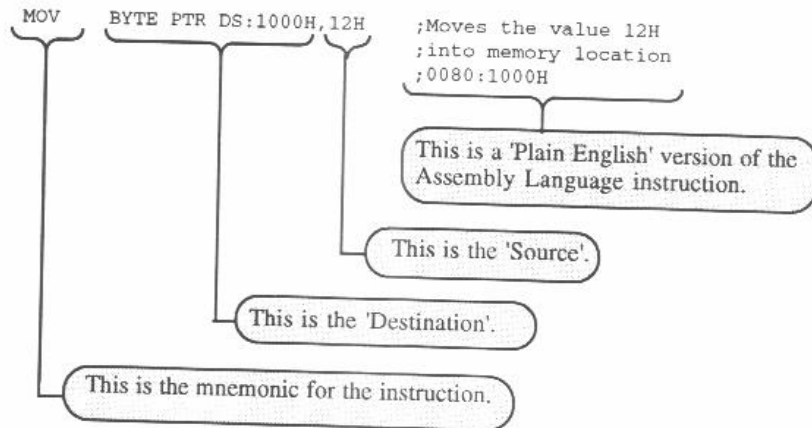
```
MOV BYTE PTR DS:1000H,12H
```

An assembly language program is much easier to understand if **comments** are added after each instruction:

```
MOV BYTE PTR DS:1000H,12H           ;Moves the value 12H
                                     ;into memory location
                                     ;0080:1000H
```

Notice that a semicolon (;) is placed in front of any comments to show that these are not actually part of the program itself.

Many 80286 instructions have a 'Source' (where the data is taken **from**) and a 'Destination' (where the data is going **to**). Our first instruction has this form:



For this instruction the source is the value 12H and the destination is location 0080:1000H.

In the destination, 'BYTE PTR' indicates 8 bit data, 'DS' stands for 'Data Segment' (Segment 0080H in the PAT) and '1000H' defines the destination address within the current segment.

It is possible to use a 'Halt' instruction to terminate the program. However, it is much more convenient to return control to the PAT System at the end of your programs. This will allow you to examine the contents of memory locations and so on.

The following program section will perform this function:

```

MOV BX,0000H
MOV AH,04H
INT 028H                                ;Returns control to the
                                        ;PAT system

```

Do not concern yourself with trying to understand this part of the program for the time being. The significance of this procedure will be explained in a subsequent chapter.

So, the final program becomes:

```

MOV BYTE PTR DS:1000H,12H      ;Moves the value 12H
                                ;into memory location
                                ;0080:1000H

MOV BX,0000H
MOV AH,04H                      ;Returns control to the
INT 028H                         ;PAT system
    
```

The program must now be **coded**, by looking up the opcodes. These can be found in the 80286 Programmer's Reference Manual.

Machine Code	Assembly Language	Comments
3E C606 0010 12	MOV BYTE PTR DS:1000H,12H	;Moves the value 12H into ;memory location 0080:1000H
BB 0000 B4 04 CD 28	MOV BX,0000H MOV AH,04H INT 028H	;Returns control to the PAT ;system

The first byte of the first instruction (3EH) defines the Data Segment as 0080H (the default value). The next two bytes (C606H) provide the code for the **operator** (opcode). The 80286 will interpret this code as "Load the memory location (within the default segment) specified by the next two bytes with the value in the byte after that". Examine the machine code and observe that the first two bytes are 0010H. This is the destination address, **low byte first**. The following byte is the source data (12H).

Coding of the instructions which return control to the PAT System can be carried out using the 80286 Programmer's Reference Manual in a similar way.

The memory locations which this program will occupy must now be defined. Anywhere in **user RAM** may be chosen. For example: starting at 0080:0100H:

Address	Codes	Assembly Language	Comments
0080:0100 0080:0103	3E C606 0010 12	MOV BYTE PTR DS:1000H,12H	;Moves the value 12H ;into memory location ;0080:1000H
0080:0106 0080:0109 0080:010B	BB 0000 B4 04 CD 28	MOV BX,0000H MOV AH,04H INT 028H	;Returns control to the ;PAT system

This type of layout is a widely accepted convention. However, other layouts may also be used. For example:

Address	Codes	Assembly Language	Comments
0080:0100	3E	MOV BYTE PTR DS:1000H,12H	;Moves the value 12H ;into memory location ;0080:1000H
0080:0101	C6		
0080:0102	06		
0080:0103	00		
0080:0104	10		
0080:0105	12	MOV BX,0000H	
0080:0106	BB		
0080:0107	00		
0080:0108	00	MOV AH,04H	;Returns control to the
0080:0109	B4		
0080:010A	04	INT 028H	;PAT system
0080:010B	CD		
0080:010C	28		

Programs should however, always be written out under the same **headings** as used above.

A Standard Programming Form is given in Appendix 1. This may be photocopied for use in writing machine code programs. Notice that there is an extra column marked "Label". It is useful in longer programs, particularly in those with loop structures, to "label" certain locations. This technique will be explained at a later stage.

Now enter this program into the PAT, using the **M** and hexadecimal keys thus:

```

M   0  1  0  0
M   3  E
+ C  6
+  0  6
+  0  0
+  1  0
+  1  2
+ B  B
+  0  0
+  0  0
+ B  4
+  0  4
+ C  D
+  2  8
    
```

The program has now been entered. To run the program, use the **G** and hexadecimal keys thus:

```

G  0  1  0  0
G
    
```

The program will now run from location 0080:0100_H. To verify correct operation, examine the contents of location 0080:0100_H thus:

```

M  1  0  0  0
M
    
```

The contents of this location should be 12_H after this program has been executed. If this does not happen, check that the machine code has been correctly entered. If this is so, repeat the procedure, paying particular attention to the required keystrokes.

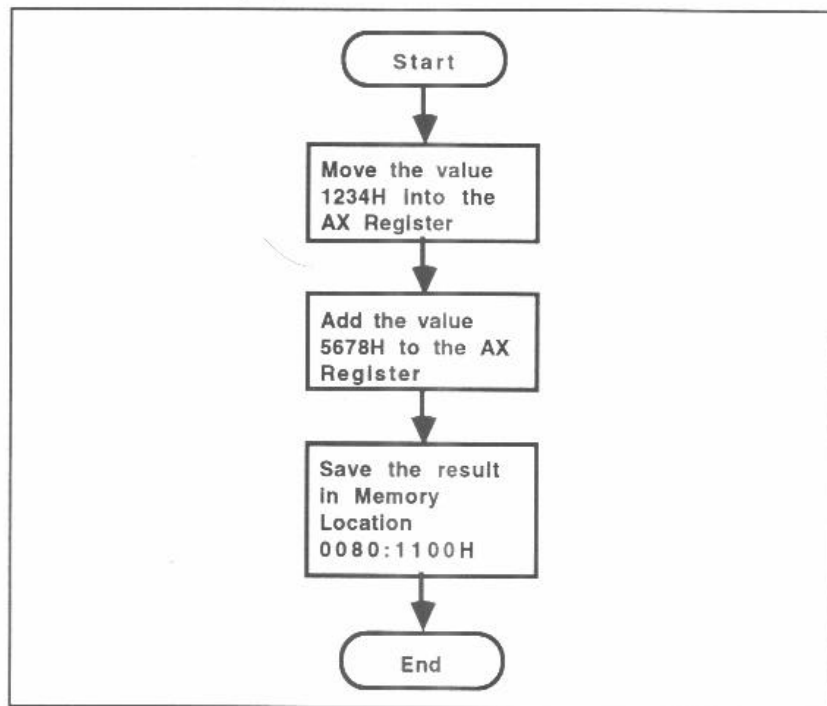
Exercise 2

Write a program, starting at location 0080:0200H, which will add the values 1234H and 5678H, saving the result at address 0080:1100H.

Solution:

In the last Exercise the Data was 8 bits wide. In this exercise we shall use 16 bit data. The most straightforward approach to this problem is to place the first value in a register and then add the second value to that register.

The flowchart for this program is:



The Assembly Language program will be:

MOV AX,1234H	;Moves the value 1234H ;into the AX Register
ADD AX,5678H	;Adds the value 5678H to ;the AX Register
MOV DS:1100H,AX	;Saves the result in ;location 0080:1100H
MOV BX,0000H	;Returns control to the
MOV AH,04H	;PAT system
INT 028H	

The codes for each instruction can be found from the 80286 Programmer's Reference Manual to be:

	Codes	Assembly Language	Comments
	B8 3412	MOV AX,1234H	;Moves the value 1234H ;into the AX Register
	05 7856	ADD AX,5678H	;Adds the value 5678H to ;the AX Register
	3E A3 0011	MOV DS:1100H,AX	;Saves the result in ;location 0080:1100H
	BB 0000	MOV BX,0000H	;Returns control to the
	B4 04	MOV AH,04H	;PAT system
	CD 28	INT 028H	

These codes must now be inserted, starting from location 0080:0200_H thus:

Address	Codes	Assembly Language	Comments
0080:0200	B8 3412	MOV AX,1234H	;Moves the value 1234H ;into the AX Register
0080:0203	05 7856	ADD AX,5678H	;Adds the value 5678H to ;the AX Register
0080:0206	3E A3 0011	MOV DS:1100H,AX	;Saves the result in ;location 0080:1100H
0080:020A	BB 0000	MOV BX,0000H	;Returns control to the
0080:020D	B4 04	MOV AH,04H	;PAT system
0080:020F	CD 28	INT 028H	

Now, use the **M** and hexadecimal keys to enter the program into PAT memory thus:

Location	Contents
0080:0200	B8
0080:0201	34
0080:0202	12
0080:0203	05
0080:0204	78
0080:0205	56
0080:0206	3E
0080:0207	A3
0080:0208	00
0080:0209	11
0080:020A	BB
0080:020B	00
0080:020C	00
0080:020D	B4
0080:020E	04
0080:020F	CD
0080:0210	28

Run the program, by using the **G** and hexadecimal keys. Remember that the start address is 0200_H.

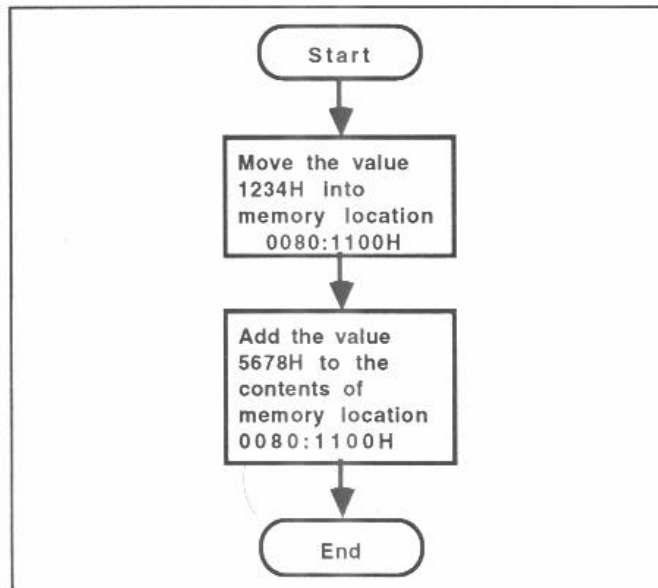
Use the **M** and hexadecimal keys to read the contents of locations 0080:1100_H and 0080:1101_H. You should find:

Location	Contents
0080:1100	AC
0080:1101	68

Now, $1234_{\text{H}} + 5678_{\text{H}} = 68\text{AC}_{\text{H}}$, so location 1100_H contains the **low** byte and location 1101_H the **high** byte.

If this does not happen, check that the machine code been correctly entered and that you have not made an error in the use of the **M** and **G** keys.

This problem could have been approached in a slightly different manner. The 80286 will allow values (also called **immediate data**) to be added **directly**, without using a Register. The flowchart for such a program would be:



The Assembly Language program will be:

```
MOV DS:1100H,1234H      ;Moves the value 1234H
                        ;into location
                        ;0080:1100H
ADD DS:1100H,5678H     ;Adds the value 5678H to
                        ;location 0080:1100H
MOV BX,0000H           ;Returns control to the
MOV AH,04H             ;PAT system
INT 028H
```

The codes for each instruction can be found from the 80286 Programmer's Reference Manual to be:

	Codes	Assembly Language	Comments
	3E C706 0011 3412	MOV DS:1100H,1234H	;Moves the value 1234H ;into location ;0080:1100H
	3E 8106 0011 7856	ADD DS:1100H,5678H	;Adds the value 5678H to ;location 0080:1100H
	BB 0000	MOV BX,0000H	;Returns control to the
	B4 04	MOV AH,04H	;PAT system
	CD 28	INT 028H	

These codes must be inserted, starting from location 0080:0200_H thus:

Address	Codes	Assembly Language	Comments
0080:0200	3E C706 0011 3412	MOV DS:1100H,1234H	;Moves the value 1234H ;into location ;0080:1100H
0080:0207	3E 8106 0011 7856	ADD DS:1100H,5678H	;Adds the value 5678H to ;location 0080:1100H
0080:020E	BB 0000	MOV BX,0000H	;Returns control to the
0080:0211	B4 04	MOV AH,04H	;PAT system
0080:0213	CD 28	INT 028H	

Having written this program, verify correct operation by keying in and then running it on the PAT. Check the contents of memory locations 1100_H and 1101_H **before** and **after** running this program to verify that the contents become AC_H and 68_H respectively.

Up to now, the machine code has been given in the text. As you have seen, finding correct machine code from the Programmer's Reference Manual is very time-consuming and prone to error. The next few chapters will explain how you can use the D2000 Development System to produce machine code programs without the need for hand-assembly of machine code.

Part of the Development System is an **assembler**. This converts assembly language mnemonics (called the **source program**) into an executable machine code program (called the **object program**). The Merlin Text Editor can be used to generate a source program which is then converted to an object program by the 80286 assembler.

Student Assessment 2

1. The number of 16 bit General Purpose Registers available within the 80286 is:
 - a 1
 - b 2
 - c 4
 - d 8

2. The 80286 instruction which copies the contents of a memory location or register into another location or register is:
 - a Move
 - b Add
 - c Increment
 - d Jump

3. The part of an instruction which provides any additional information necessary to complete that instruction is called the:
 - a Address
 - b Data
 - c Operand
 - d Operator

4. A set of codes for every possible microprocessor operation is called:
 - a a Data Set
 - b an Instruction Set
 - c an Operand Set
 - d an Operator Set

5. The Machine code for the instruction "MOV DS:2400H, BX " is:
 - a BB 0024
 - b 3E 891E 0024
 - c 3E 8B1E 0024
 - d 3E A3 0024