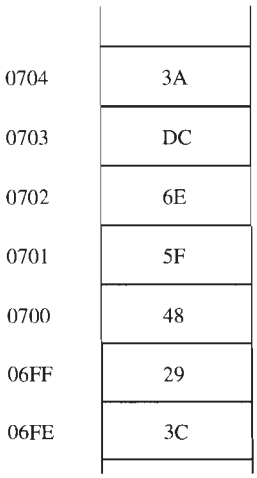
CHAPTER 4 SELF-TEST

1. Identify the op-code and operand for each instruction below.
   (a) INC AX
   (b) WAIT
   (c) MOV DX,CX
   (d) MOV AL,[BX+6]
2. Which of the following instructions are *invalid*? Explain your answer.
   (a) MOV AX, SI
   (b) MOV AL,1234H
   (c) MOV CX,BL
   (d) MOV [100H], EAX

Figure 4–18.
Figure for Self-Test 6.



3. Determine the contents of register DH after the following instructions have executed. Assume the content of memory is as shown in Figure 4–2.
   MOV BX,0021H
   MOV DH,[BX]
4. Using the direct addressing mode, give the single instruction that adds 1 to the memory word stored at address 0200H.
5. The following instructions are executed on an 80386 in Real Mode. Which of these instructions, if any, are *invalid*? Explain.
   (a) INC EAX
   (b) MOV CX,[AX+32]
   (c) AND AL,[10000000H]
   (d) MOV EBX,[SI]
   (e) MOV DX,[ESI] ;ESI = 0000A000H
6. Refer to Figure 4–18 and determine the contents of registers AX, BX, and SP after the following instructions have executed:

```
MOV   SP,0700H
POP   AX
POP   BX
```

7. Assume register AL = B7H and the instruction AND AL,3CH is given. Determine the new value of register AL.
8. The program below will (a) loop (b) exit the loop when the data input is the carriage return character.

```
A1:   IN      AL,36H    ;Input a byte to AL from port 36
      CMP     AL,0DH    ;Compare AL with 0DH
      LOOPE   A1        ;Loop if equal
```

9. If register AX = 100H and register BX = 200H, determine the contents of registers AX and DX after the instruction MUL BX is given.
10. What is the DEBUG command to begin assembling instructions at location 560H in the default code segment?

11. To enter hexadecimal codes into memory with DEBUG, use the _____ command. To display these same codes in hexadecimal and ASCII, use the _____ command.
12. Assume a machine code program resides in memory beginning at address 350H. If the program is 20 bytes long, what is the DEBUG command to display the program instructions in mnemonic form?
13. Typing *g* in DEBUG causes the program whose offset address is stored in register _____ to begin executing.
14. The instruction MOV AL,[200] moves the (a) byte (b) word (c) doubleword at offset address 200 in the (a) code (b) data (c) stack (d) extra segment to register AL.

ANALYSIS AND DESIGN QUESTIONS

Section 4.1

4.1 Calculate the *physical address* of the memory operand for the instruction MOV AX,[BP+12H]. Assume BP = 0350H, DS = E000H, and SS = F910H.

4.2 For each of the following, determine the value of the operand after the instruction has executed. Assume register BX = 0600H for each question, and the contents of memory are as shown in Figure 4–3. *Hint:* Recall that the 80x86 processors access memory operands using *little endian* format.

(a) INC BX

(b) INC BYTE PTR [BX]

(c) INC WORD PTR [BX]

(d) INC DWORD PTR [BX]

4.3 The instruction PUSHA pushes the following registers onto the stack: AX, CX, DX, BX, SP, BP, SI, and DI. Determine the value of register SP after the following instructions have executed.

```
MOV     SP,F000H
PUSHA
```

4.4 Determine the memory offset address of the operand in the instruction MOV AX,[BX+SI*8+5]. Assume BX = 7E00H and SI = 0004H.

4.5 Thinking of the data in Figure 4–19 as an array of bytes with a base address of 8800H, write a program that initializes a pointer to the array base and then adds array elements two and six, storing the result as element 12. Use the base-plus-displacement addressing mode.

4.6 The data in Figure 4–19 could be thought of as an array with four elements per record and a base address of 8800H. Element zero of record one would then be 40H. Keeping this organization in mind, write a program that adds elements three of records one and two. Store the result as element two, record three, of a new array with base address A000H. Use the base-plus-index × scaling factor plus displacement addressing mode. *Hint:* Use register SI as a record pointer.

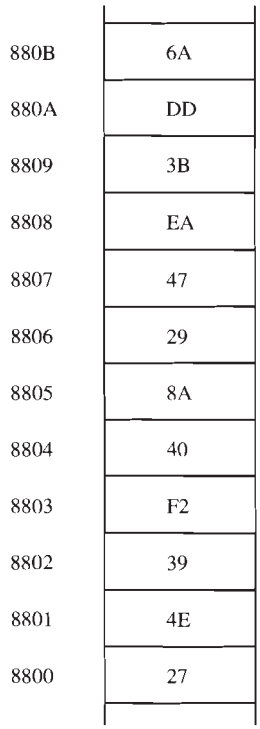


Figure 4–19.
Figure for Analysis and Design Question 4.5.

The instruction INC SI can then be used to adjust this pointer to consecutive records.

4.7 Determine the contents of the BX and CX registers and all flags after the following program has run. Refer to Figure 4–19 for the contents of memory.

```
MOV     BX,8802H
MOV     CX,3C7AH
AND     CX,[BX]
HALT
```

Self-Test Answers

1.	Op-Code	Operand
(a)	inc	ax
(b)	wait	none
(c)	mov	dx,cx
(d)	mov	al,[bx+6]

2. (b) Can't move 16-bit immediate data into an 8-bit register
   (c) Can't copy an 8-bit register into a 16-bit register
3. DH=32
4. INC WORD PTR [0200H] (OR ADD WORD PTR [0200H],1)
5. (b) Only registers BX, BP, SI, DI can be used as memory pointers in a 64K segment
   (C) Direct memory address cannot exceed FFFFH in a 64K segment
6. AX = 5F48, BX = DC6E, SP = 0704
7. 34H
8. loop
9. 256 x 512 = 131,072 = 20000H. DX = 0002, AX = 0000
10. A560
11. E, D
12. U350 L14
13. IP
14. (a) byte, (b) data

Analysis and Design Answers

4.1 F9100H + 0350H + 12H = F9462H 'Stack segment is the default when using bp
4.2 (a) 0601H; (b) E9H; (c) 27E9H; (d) e8db27e9h
4.3 F000H - 16 bytes = EFF0H
4.4 8 * 4 = 32 = 20h. Therefore: 7E00H + 20H + 5 = 7E25H
4.5
```
mov  bx,8800h     ;Point to array base
mov  al,[bx+2]    ;Get element 2
add  al,[bx+6]    ;Add element 6
mov  [bx+12h],al  ;Store result as element 12h
```

```
4.6 mov  bx,8800h        ;BX points to base of first array
    mov  si,1            ;SI points to record one
    mov  al,[bx+si*4+3]  ;Get element three of record one
    inc  si              ;Point to next record (record two)
    add  al,[bx+si*4+3]  ;Add element three of record two
    mov  bx,a000h        ;Point BX to base of second array
    inc  si              ;Point SI to record three
    mov  [bx+si*4+2],al  ;Store result as element two of record three in the
                         ;destination array
4.7 mov  bx,8802h    ;BX=8802
    mov  cx,3c7ah    ;CX=3C7A
    and  cx,[bx]     ;   1111 0010 0011 1001  = F239
                     ; • 0011 1100 0111 1010  = 3C7A
                     ;   0011 0000 0011 1000  = 3038
    hlt
```

Flags: CF=0, PF=0 (3038 has an odd number of 1s), AF=X, ZF=0, SF=0 (bit 15 is a 0), OF=0

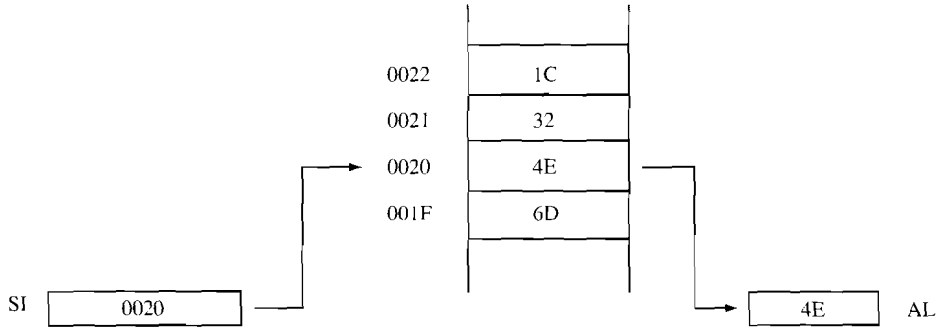


Figure 4–2.
The register indirect addressing mode uses a CPU register to specify the address of the memory operand. This example diagrams the instruction MOV AL,[SI].

The Default Memory Segment. Recall that the 80x86 processors partition memory into six segments, called the code, data, stack, extra, F, and G segments. Every instruction that references memory defaults to one of these six segments. (This was shown previously in Table 3–1). The instruction MOV AL,[SI], for example, accesses the data byte pointed to by register SI in the *data* segment. However, the similar instruction MOV AL,[SP] accesses the data byte pointed to by register SP in the *stack* segment.

Example 4.1 Assume an 8086 processor executes the instructions (a) MOV AL,[SI] and (b) MOV AL,[SP]. Compute the physical address of the memory operands, assuming SI = 0100H, SP = 0200H, SS = E010H, and DS = 1A00H.

Solution (a) The default segment is DS. The physical address, therefore, can be written as

DS:[SI] = 1A000H + 0100H = 1A100H.

(b) The default segment is SS. The physical address, therefore, can be written as

SS:[SP] = E0100H + 0200H = E0300H.

16-Bit vs. 32-Bit Instructions. In Chapter 2 we learned that the 80286 and later processors can be operated in Real or Protected Mode. Real Mode restricts the processor to 64K memory segments (16-bit offsets), while Protected Mode allows segments as large as 4 GB (32-bit offsets). Accordingly, 80x86 processors can accommodate two types of program instructions: those that use 16-bit memory addresses (*16-bit instructions*) and those that use 32-bit memory addresses (*32-bit instructions*). The first step in writing an 80x86 program, therefore, is to decide if the code is to be placed in a 16- or 32-bit memory segment.

The above paragraph might lead you to believe that none of the 32-bit registers are accessible in a 16-bit segment (Real Mode). This is not the case. Consider the following instructions, which are all valid in Real or Protected Mode:

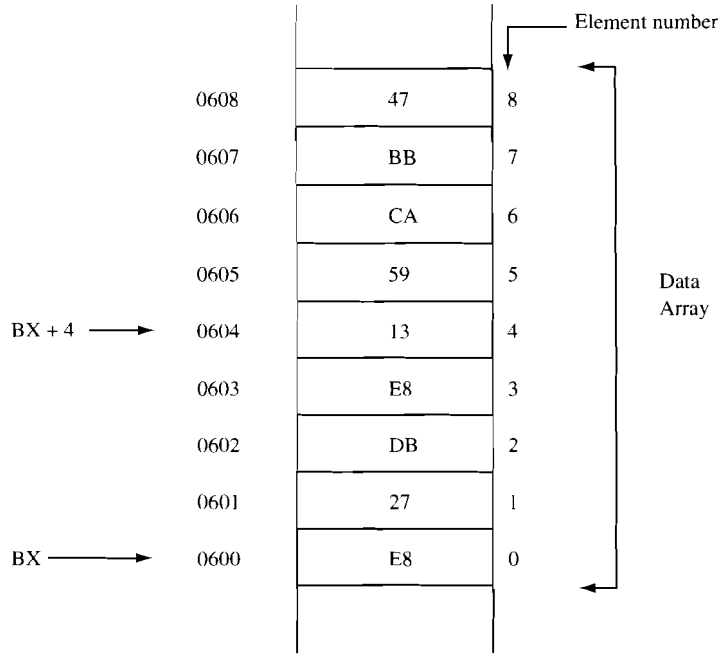


Figure 4–3.
The instruction MOV AL,[BX+4] can be used to access element four in a data array whose base address is stored in register BX.

```
MOV BX,0600H          ;BX points to array starting address
MOV SI,0010H          ;SI points to record four (4 elements/
                       record × four records = 16 = 0010)
MOV AL, [BX+SI+3]     ;Retrieve element three
```

In this example, register BX is considered the base address, register SI the index address, and three is the displacement. The sum of the three address components is:

base + index + displacement = 0600H + 0010H + 0003H = 0613H

This is called the *base plus index plus displacement addressing mode.* Note that the same 16- and 32-bit restrictions apply as given in the previous example, with the addition that register ESP is not allowed for the index address.

Scaling. Often the data stored in an array is organized as a byte, word (two bytes), doubleword (four bytes), or quad word (eight bytes). The index address of a particular record in such an array can be computed by multiplying the record number by the record size. For example, in Figure 4–4 each record stores a doubleword. The index address of record four is, therefore, 4 bytes/record × 4 (record number) = 16 = 10H. To access element three in record four, the following code can be used:

```
MOV BX,0600H          ;BX points to array starting address
MOV SI,0004H          ;SI points to record four
MOV AL,[BX+SI*4+3]    ;Retrieve element three
```