Program 3.1

Write a program for addition of two numbers.

Solution The following program adds two 16-bit operands. There are various methods of specifying operands depending upon the addressing modes that the programmer wants to use. Accordingly, there may be different program listings to achieve a single programming goal. A skilled programmer uses a simple logic and implements it by using a minimum number of instructions. Let us now try to explain the following program:

```
ASSUME CS:CODE, DS:DATA

DATA SEGMENT
OPR1 DW 1234H                       ; 1st operand
OPR2    DW 0002H                    ; 2nd operand
RESULT  DW 01 DUP(?)                ; A word of memory reserved for re-
                                      sult

DATA    ENDS
CODE    SEGMENT
START:  MOV AX, DATA                ; Initialize data segment
        MOV DS, AX                  ;
        MOV AX, OPR1                ; Take 1st operand in AX
        MOV BX, OPR2                ; Take 2nd operand in BX
        CLC                         ; Clear previous carry if any
        ADD AX, BX                  ; Add BX to AX
        MOV DI, OFFSET RESULT       ; Take offset of RESULT in DI
        MOV [DI], AX                ; Store the result at memory address in DI
        MOV AH, 4CH                 ; Return to DOS prompt
        INT 21H
CODE    ENDS                        ; CODE segment ends
        END START                   ; Program ends
```

Program 3.1(a) *Listings*

Program 3.2

Write a program for the addition of a series of 8-bit numbers. The series contains 100(numbers).

Solution In the first program, we have implemented the addition of two numbers. In this program, we show the addition of 100 (D) numbers. Initially, the resulting sum of the first two numbers will be stored. To this sum, the third number will be added. This procedure will be repeated till all the numbers in the series are added. A conditional jump instruction will be used to implement the counter checking logic. The comments explain the purpose of each instruction.

```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT                          ; Data segment starts
NUMLIST DB 52H, 23H,-                 ; List of byte numbers
COUNT EQU 100D                        ; Number of bytes to be added
RESULT DW 01H DUP(?)                  ; One word is reserved for result
DATA ENDS                             ; Data segment ends
CODE SEGMENT                          ; Code segment starts at relative
ORG 200H                              ; address 0200h in code segment
START:    MOV AX, DATA                ; Initialize data segment
          MOV DS, AX
          MOV CX, COUNT               ; Number of bytes to be added in CX
          XOR AX, AX                  ; Clear AX and CF
          XOR BX, BX                  ; Clear BH for converting the byte to
                                        word
          MOV SI,OFFSET NUMLIST       ; Point to the first number in the
                                        list
AGAIN:    MOV BL, [SI]                ; Take the first number in BL,BH is zero
          ADD AX, BX                  ; Add AX with BX
          INC SI                      ; Increment pointer to the byte list
          DEC CX                      ; Decrement counter
          JNZ AGAIN                   ; If all numbers are added,point to re-
                                        sult
          MOV DI, OFFSET RESULT       ; destination and store it
          MOV [DI], AX
          MOV AH, 4CH                 ; Return to DOS
          INT 21H
          CODE ENDS
END       START
```

Program 3.2 *Listings*

sion) technique in the next section.

Example 3.1

Write a program to add a data byte located at offset 0500H in 2000H segment to another data byte available at 0600H in the same segment and store the result at 0700H in the same segment.

Solution The flow chart for this problem may be drawn as shown in Fig. 3.1.

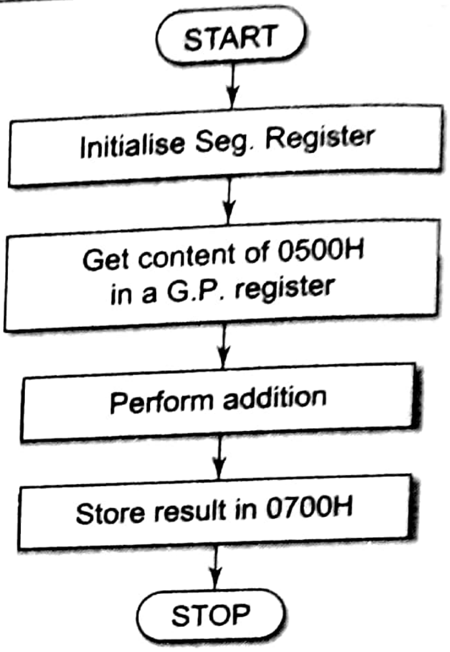

Fig. 3.1 *Flow Chart for Example 3.1*

```
MOV   AX, 2000H      ; Initialising DS with value
MOV   DS, AX         ; 2000H
MOV   AX, [500H]     ; Get first data byte from 0500H
                       offset
ADD   AX, [600H]     ; Add this to the second byte
                       from 0600H
MOV   [700H], AX     ; Store AX in 0700H (result).
HLT                  ; Stop
```

Scanned with CamScanner

Example 3.2

Write a program to move the contents of the memory location 0500H to register BX and to CX. Add immediate byte 05H to the data residing in memory location, whose address is computed using DS = 2000H and offset = 0600H. Store the result of the addition in 0700H. Assume that the data is located in the segment specified by the data segment register DS which contain 2000H.

Solution The flow chart for the program is shown in Fig. 3.2.

```
MOV   AX, 2000H
MOV   DS, AX        ; Initialize data segment register
MOV   BX, [0500H]   ; Get contents of 0500H in BX
```

```
MOV   CX, BX           ; Copy the same contents in
                         CX
ADD   [0600H], 05H;  Add byte 05H to contents
                         of 0600H
MOV   DX, [0600H]    ; Store the result in DX
MOV   [0700H], DX    ; Store the result in 0700H
HLT                  ; Stop
```

After initialising the data segment register, the content of location 0500H are moved to the BX register using MOV instruction. The same data is moved also to the CX register. For this data transfer, there may be two options as shown.

```
(a) MOV CX, BX          ; As the contents of BX will be
                        ; same as 0500H after execution
                        ; of MOV BX,[0500H].
(b) MOV CX, [0500H];  Move directly from 0500H
                        to register CX
```

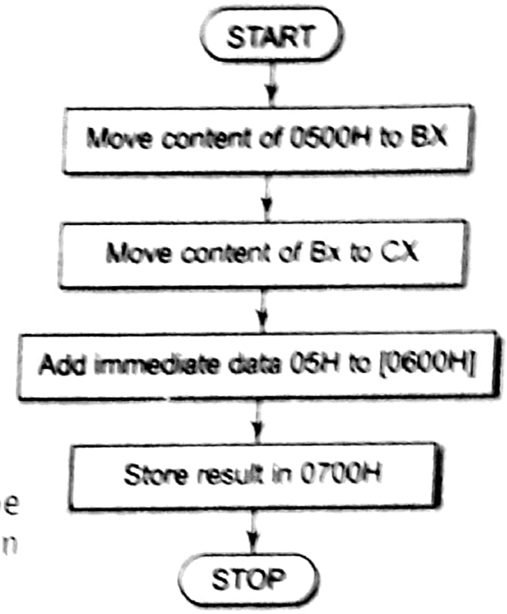

Fig. 3.2 *Flow Chart for Example 3.2*

Example 3.3

Add the contents of the memory location 2000H:0500H to contents of 3000H:0600H and store the result in 5000H:0700H.

Solution Unlike the previous example programs, this program refers to the memory locations in different segments, hence, while referring to each location, the data segment will have to be newly initialised with the required value. Figure 3.3 shows the flow chart.

The instruction sequence for the above flow chart is given along with the comments.

```
MOV   CX, 2000H     ; Initialize DS at 2000H
MOV   DS, CX
MOV   AX, [500H]    ; Get first operand in AX
MOV   CX, 3000H     ; Initialize DS at 3000H
MOV   DS, CX
MOV   BX, [0600H]   ; Get second operand in BX.
ADD   AX, BX        ; Perform addition
MOV   CX, 5000H     ; Initialize DS at 5000H
MOV   DS, CX
MOV   [0700H],AX    ; Store the result of
                      addition in
HLT                 ;  0700H and stop
```

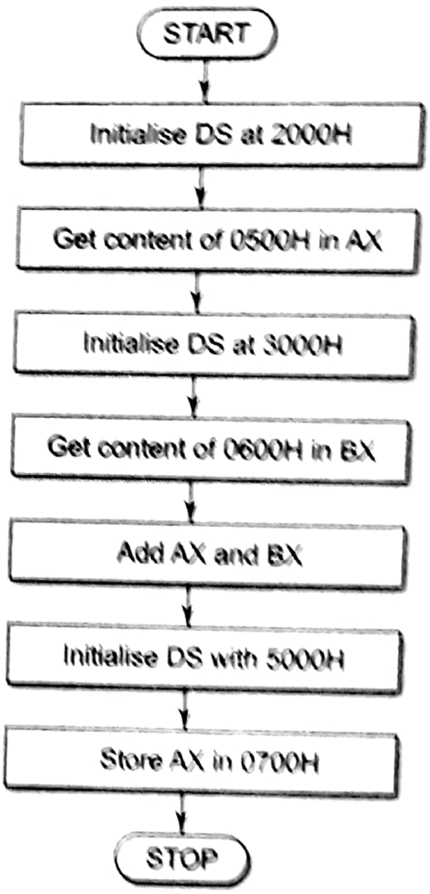

Fig. 3.3 *Flow Chart for Example 3.2*

Example 3.4

Move a byte string, 16-bytes long, from the offset 0200H to 0300H in the segment 7000H.

Solution According to the program statement, a string that is 16-bytes long is available at the offset address 0200H in the segment 7000H. The required program should move this complete string at offset 0300H, in the same segment. Let us emphasize this program in the light of comparison between 8085 and 8086 programming techniques.

An 8085 program to perform this task, is given neglecting the segment addresses.

```
         MVI   C, 010H ; Count for the length of string
         LXIH  0200H   ; Initialization of HL pair for source string
         LXID  0300H   ; Initialization of DE pair for destination
BACK  :  MOV   A, M    ; Take a byte from source in A
         STAX  D       ; Store contents of A to address pointed to by DE
                         pair
         INX   H       ; Increment source pointer
         INX   D       ; Increment destination pointer
         DCRC          ; Decrement counter
         JNZ  BACK     ; Continue if counter is not zero
         HLT           ; Stop if counter is zero
```

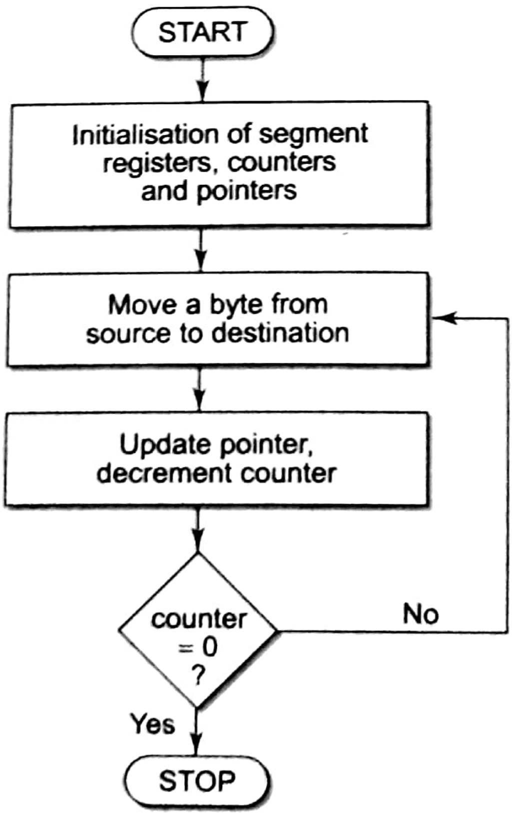

Fig. 3.4 *Flow Chart for Example 3.4*

```
        MOV   AX, 7000H
        MOV   DS, AX     ;  Data segment initialization
        MOV   SI, 0200H  ;  Pointer to source string
        MOV   DI, 0300H  ;  Pointer to destination string
        MOV   CX, 0010H  ;  Count for length of string
BACK :  MOV   AL, [SI]   ;  Take a source byte in AL
        MOV   [DI], AL   ;  Move it to destination
        INC   SI         ;  Increment source pointer
        INC   DI         ;  Increment destination pointer
        DEC   CX         ;  Decrement count by 1
        JNZ   BACK       ;  Continue if count is not 0
        HLT              ;  Stop if the count is 0
```

```
        MOV   AX, 7000H
        MOV   DS, AX          ; Data segment initialization
        MOV   SI, 0200H       ; Source pointer initialization
        MOV   DI, 0300H       ; Destination pointer initialization
        MOV   CX, 0010H       ; Counter initialization
BACK :  MOV   AL, [SI]        ; Take a byte of string from source
        MOV   [DI], AL        ; and then move it to destination
        INC   SI              ; Update source pointer
        INC   DI              ; Update destination pointer continue
        LOOP BACK             ; till CX =0.[DEC CX and JNZ BACK]
        HLT                   ; Stop if CX =0
```

```
     MOV AX, 7000H
     MOV DS, AX           ;  Source segment initialisation
     MOV ES, AX           ;  Destination segment initialisation
     MOV CX, 0010H        ;  Counter initialisation
     MOV SI, 0200H        ;  Source pointer initialisation
     MOV DI, 0300H        ;  Destination pointer initialisation
     CLD                  ;  Clear DF
REP  MOVSB                ;  Move the complete string
     HLT                  ;  Stop
```

Example 3.5

Find out the largest number from an unordered array of sixteen 8-bit numbers stored sequentially in the memory locations starting at offset 0500H in the segment 2000H.

Solution The logic for this procedure can be described as follows. The first number of the array is taken in a register, say AL. The second number of the array is then compared with the first one. If the first one is greater than the second one, it is left unchanged. However, if the second one is greater than the first, the second number replaces the first one in the AL register. The procedure is repeated for every number in the array and thus it requires 15 iterations. At the end of 15th iteration the largest number will reside in the register AL. This may be represented in terms of the flow chart as shown in Fig. 3.5. The listing is given below:

```
         MOV  CX, OF H    ; Initialize counter for number of iterations
         MOV  AX, 2000H   ; Initialize data segment
         MOV  DS, AX      ;
         MOV  SI, 0500H   ; Initialize source pointer
         MOV  AL, [SI]    ; Take first number in AL
BACK :   INC  SI          ; Increment source pointer
         CMP  AL, [SI]    ; Compare next number with the previous
         JNC  NEXT        ; If the next number is larger
         MOV  AL, [SI]    ; replace the previous one with the next
NEXT :   LOOPBACK         ; Repeat the procedure 15 times
         HLT
```

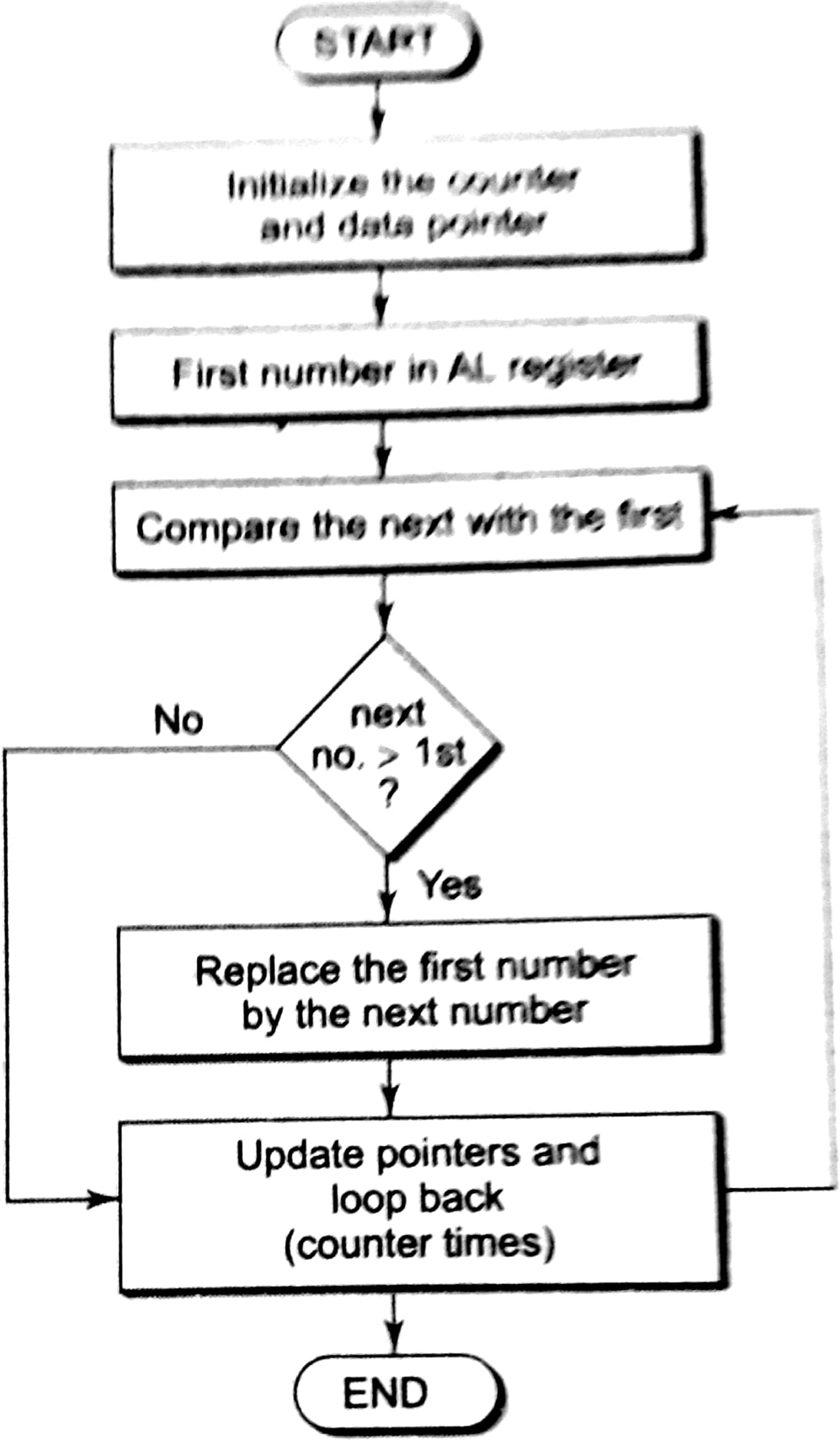

Fig. 3.5 *Flow Chart for Example 3.5*

Program 3.5

A program to find out the number of even and odd numbers from a given series of 16-bit hexadecimal numbers.

Solution The simplest logic to decide whether a binary number is even or odd, is to check the least significant bit of the number. If the bit is zero, the number is even, otherwise it is odd. Check the LSB by rotating the number through carry flag, and increment even or odd number counter.

```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
LIST DW 2357H, 0A579H, 0C322H, 0C91EH, 0C000H, 0957H
COUNT EQU 006H
DATA ENDS
CODE SEGMENT
START:      XOR BX, BX
            XOR DX, DX
            MOV AX, DATA
            MOV DS, AX
            MOV CL, COUNT
            MOV SI, OFFSET LIST
AGAIN:      MOV AX, [SI]
            ROR AX, 01
            JC ODD
            INC BX
            JMP NEXT
ODD:        INC DX
NEXT:       ADD SI, 02
            DEC CL
            JNZ AGAIN
            MOV AH, 4CH
            INT 21H
            CODE ENDS
            END START
```

Program 3.5 *Listings*

Program 3.6

Write a program to find out the number of positive numbers and negative numbers from a given series of signed numbers.

Solution Take the *i*th number in any of the registers. Rotate it left through carry. The status of carry flag, i.e. the most significant bit of the number will give the information about the sign of the number. If CF is 1, the number is negative; otherwise, it is positive.

```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
LIST DW 2579H, 0A500H, 0C009H, 0159H, 0B900H
COUNT EQU 05H
DATA ENDS
CODE SEGMENT
START:        XOR BX, BX
              XOR DX, DX
              MOV AX, DATA
              MOV DS, AX
              MOV CL, COUNT
              MOV SI, OFFSET LIST
AGAIN:        MOV AX, [SI]
              SHL AX, 01
              JC NEG
              INC BX
              JMP NEXT
NEG:          INC DX
NEXT:         ADD SI, 02
              DEC CL
              JNZ AGAIN
              MOV AH, 4CH
              INT 21H
              CODE ENDS
              END START
```

Program 3.6 *Listings*