4

8051 ASSEMBLY PROGRAMMING

Learning Objectives

After you have completed this chapter, you should be able to

- Explain the structure of an assembly language program
- Use assembler directives to allocate memory blocks, define constants, etc.
- Write assembly programs to perform simple arithmetic operations
- Write assembly programs to set up time delays and to calculate time delays in a given loop

4.1 ASSEMBLY LANGUAGE PROGRAMS

In the previous chapter, we have discussed architecture, addressing modes and the instruction set of the 8051 microcontroller. In this chapter, we will write assembly language programs. The 8051 assembly program consists of two sections, namely, assembly language program and assembler directives. An *assembly language program* consists of a sequence of statements that tell the microcontroller to perform the desired operations. *Assembler directives* instruct the assembler on how to process subsequent assembly language instructions. Each assembler uses various directives and the assembler directives discussed here correspond to 8051 Assembler.

4.2 ASSEMBLER DIRECTIVES

Assembler directives appear just like instructions in an assembly language program, but they tell the assembler to do something other than creating the machine code for an instruction. In assembly language programming, the assembler directives instruct the assembler to

- process subsequent assembly language instructions
- define program constants
- reserve space for variables

Each assembler uses various directives. The following are the widely used 8051 assembler directives.

ORG (origin)

The ORG directive is used to indicate the starting address. It can be used only when the program counter needs to be changed. The number that comes after ORG can be either in hex or in decimal.

EXAMPLE 4.1

```
ORG 2000H                ; Set program counter 2000
```

EQU and SET

The EQU and SET directives assign numerical value or register name to the specified symbol name. EQU is used to define a constant without storing information in the memory. The symbol defined with EQU should not be redefined, whereas the SET directive allows redefinition of symbols at a later stage.

EXAMPLE 4.2

```
Pointer  SET  R1          ; use R1 as pointer
Counter EQU R3            ; use R3 as counter
N  EQU 35H
MOV R3, #N                ; 35h is  stored in R3
```

DB (define byte)

The DB directive is used to define an 8 bit data. DB directive initialises memory with 8 bit values. The numbers can be in decimal, binary, hex or in ASCII formats. For decimal, the 'D' after the decimal number is optional, but for binary and hexadecimal, 'B' and 'H' are required. For ASCII, the number is written in quotation marks (' ').

EXAMPLE 4.3

```
DATA1: DB   40H           ; hex
DATA2: DB   01011100B     ; binary
DATA3: DB   48            ; decimal
DATA4: DB   'HELLOW'      ; ASCII
```

END

The END directive signals the end of the assembly module. It indicates the end of the program to the assembler. Any text in the assembly file that appears after the END directive is ignored. If the END statement is missing, the assembler will generate an error message.

4.3 ASSEMBLY LANGUAGE PROGRAMS

EXAMPLE 4.4

Write a program to add the values of locations 50H and 51H and store the result in locations 52H and 53H.

ALGORITHM

```
Step 1: Load the memory contents 50H into A
Step 2: ADD the memory contents 51H with contents of A
Step 3: Store the contents of A in 52H
Step 4: Store the contents of carry flag in 53H.
The program is as follows
ORG 0000H   ; Set program counter 0000H
MOV A,50H   ; Load the contents of memory location 50H into A
ADD A,51H   ; Add the contents of memory location 51H with contents of A
MOV 52H,A   ; Save the least significant byte of the result in location
              52H
MOV A, #00  ; Load 00H into A
ADDC A, #00 ; Add the immediate data and the contents of carry flag to A
MOV 53H,A   ; Save the most significant byte of the result in location 53
END
```

EXAMPLE 4.5

Write a program to subtract the values of locations 51H from 50H and store the result in location 52H. If the result is positive, store 00H, else store 01H in 53H.

ALGORITHM

```
Step 1: Load the memory contents 50H into A A
Step 2: Clear the carry flag
Step 3: Subtract the memory contents 51H from the contents of A A
Step 4: Store the contents of A A in 52H
Step 5: Check the contents of carry flag, if the contents of carry flag
        is 0 (result is positive) store 00H, else if the contents of
        carry flag is 1 (result is negative) store 01H in 53H.
```

```
ORG 0000H      ; Set program counter 0000H
MOV  A,50H     ; Load the contents of memory location 50H into A
CLR C          ; Clear the borrow flag
SUBB  A,51H    ; Subtract the contents of memory location 51H from
                 content of A
MOV 52H,A      ; Store the result  in location 52H
MOV A,#00      ; Load 00H into A
ADDC A,#00     ; Add the immediate data and the contents of carry flag to A
MOV 53H,A      ; Save the most significant byte of the result in location 53
END
```

EXAMPLE 4.6

Write a program to store data FFH into RAM memory locations 50H to 58H using direct addressing mode.

```
ORG 0000H      ; Set program counter 0000H
MOV A, #0FFH   ; Load FFH into A
MOV 50H,A      ; Store contents of A in location 50H
MOV 51H,A      ; Store contents of A in location 51H
MOV 52H,A      ; Store contents of A in location 52H
MOV 53H,A      ; Store contents of A in location 53H
MOV 54H,A      ; Store contents of A in location 54H
MOV 55H,A      ; Store contents of A in location 55H
MOV 56H,A      ; Store contents of A in location 56H
MOV 57H,A      ; Store contents of A in location 57H
MOV 58H,A      ; Store contents of A in location 58H
END
```

EXAMPLE 4.7

Write a program to store data FFH into RAM memory locations 50H to 58H using indirect addressing mode.

```
        ORG 0000H        ; Set program counter 0000H
        MOV A, #0FFH     ; Load FFH into A
        MOV R0, #50H     ; Load pointer, R0=50H
        MOV R5, #08H     ; Load counter, R5=08H
Start:  MOV @R0,A        ; Copy contents of A to internal data RAM pointed
                           by R0
        INC R0           ; Increment pointer
        DJNZ R5, start   ; Repeat until R5 is zero
        END
```

EXAMPLE 4.8

Write a program to add two 16 bit numbers stored at locations 51H–52H and 55H–56H and store the result in locations 40H, 41H and 42H. Assume that the least significant byte of data and the result is stored in low address and the most significant byte of data or the result is stored in high address.

The program is as follows

```
ORG 0000H    ; Set program counter 0000H
MOV  A,51H   ; Load the contents of memory location 51H into A
ADD  A,55H   ; Add the contents of memory location 55H with contents of A
MOV 40H,A    ; Save the least significant byte of the result in location
               40H
MOV  A,52H   ; Load the contents of memory location 52H into A
ADDC  A,56H ; Add the contents of 56H and cy flag with contents of A
MOV 41H,A    ; Save the second byte of the result in location 41H
MOV A,#00    ; Load 00H into A
ADDC A,#00   ; Add the immediate data 00H and the contents of carry flag
               to A
MOV 42H,A    ; Save the most significant byte of the result in location 42H
END
```

EXAMPLE 4.9

Write a program to subtract a 16 bit number stored at locations 51H–52H from 55H–56H and store the result in locations 40H and 41H. Assume that the least significant byte of data or the result is stored in low address. If the result is positive, then store 00H, else store 01H in 42H.

The program is as follows

```
ORG 0000H    ; Set program counter 0000H
MOV  A,55H   ; Load the contents of memory location 55H into A
CLR C        ; Clear the borrow flag
SUBB  A,51H ; Sub the contents of memory location 51H from contents of A
MOV 40H,A    ; Save the least significant byte of the result in location 40H
MOV  A,56H   ; Load the contents of memory location 56H into A
SUBB  A,52H ; Sub the content of memory location 52H from the contents of A
MOV 41H,A    ; Save the most significant byte of the result  in location 41H
MOV A, #00   ; Load 00H into A
ADDC A, #00 ; Add the immediate data and the contents of carry flag to A
MOV 42H, A   ; If result is positive, store 00H, else store 01H in location 42H
END
```

EXAMPLE 4.10

Write a program to add two Binary Coded Decimal (BCD) numbers stored at locations 60H and 61H and store the result in BCD at memory locations 52H and 53H. Assume that the least significant byte of the result is stored in low address.

The program is as follows

```
ORG 0000H   ; Set program counter 0000H
MOV  A,60H  ; Load the contents of memory location 60H into A
ADD  A,61H  ; Add the contents of memory location 61H with contents of A
DA A        ; Decimal adjustment of the sum in A
MOV 52H, A  ; Save the least significant byte of the result in location 52H
MOV A,#00   ; Load 00H into A
ADDC A,#00  ; Add the immediate data and the contents of carry flag to A
MOV 53H,A   ; Save the most significant byte of the result in location 53
END
```

EXAMPLE 4.11

Write a program to clear 10 RAM locations starting at RAM address 1000H.

The program is as follows

```
      ORG 0000H          ; Set program counter 0000H
      MOV  DPTR, #1000H  ; Copy address 1000H to DPTR
      CLR A              ; Clear A
      MOV R6, #0AH       ; Load 0AH to R6
again:MOVX @DPTR,A       ; Clear RAM location pointed by DPTR
      INC DPTR           ; Increment DPTR
      DJNZ R6, again     ; Loop until counter R6 = 0
      END
```

EXAMPLE 4.12

Write a program to clear 10 RAM locations starting at RAM address 10H.

The program is as follows

```
      ORG 0000H       ; Set program counter 0000H
      MOV  R0, #10H   ; Copy address 10H to R0
      CLR A           ; Clear A
      MOV R6,#0AH     ; Load 0AH to R6
```

```
again: MOV @R0,A          ; Clear RAM location pointed by R0
       INC R0             ; Increment R0
       DJNZ R6, again     ; Loop until counter R6 = 0
       END
```

EXAMPLE 4.13

Write a program to compute 1 + 2 + ... + N (say 15) and save the sum at 70H

```
The program is as follows
    ORG 0000H              ; Set program counter 0000H
    N EQU 15
    MOV R0, #00            ; Clear R0
    CLR A                  ; Clear A
again: INC R0              ; Increment R0
    ADD A, R0              ; Add the contents of R0 with contents of A
    CJNE R0, #N, again     ; Loop until counter, R0 = N
    MOV 70H,A              ; Save the result in location 70H
    END
```

EXAMPLE 4.14

Write a program to multiply two 8 bit numbers stored at locations 70H and 71H and store the result at memory locations 52H and 53H. Assume that the least significant byte of the result is stored in low address.

```
The program is as follows
ORG 0000H ; Set program counter 0000H
MOV A,70H ; Load the contents of memory location 70H into A
MOV B,71H ; Load the contents of memory location 71H into B
MUL AB    ; Perform multiplication
MOV 52H,A ; Save the least significant byte of the result in location 52H
MOV 53H,B ; Save the most significant byte of the result in location 53
END
```

EXAMPLE 4.15

Write a program to divide contents of 70H from contents of 71H (Assume contents of 70H is greater or equal to contents of 71H). Store the remainder at memory location 53H and the quotient at memory location 52H.

```
The program is as follows
ORG 0000H      ; Set program counter 0000H
MOV A,70H      ; Load the contents of memory location 70H into A
MOV B,71H      ; Load the contents of memory location 71H into B
DIV AB         ; Perform division
MOV 52H,A      ; Save the quotient in location 52H
MOV 53H,B      ; Save the remainder in location 53H
END
```

EXAMPLE 4.16

Ten 8 bit numbers are stored in internal data memory from location 50H. Write a program to increment the data.

SOLUTION

Assume that ten 8 bit numbers are stored in internal data memory from location 50H, hence R0 or R1 must be used as a pointer.

```
The program is as follows
      ORG 0000H       ; Set program counter 0000H
      MOV R0,#50H     ; Load pointer, R0=50H
      MOV R3,#0AH     ; Load counter, R3=0AH
Loop1: INC @R0        ; Increment contents of internal data RAM pointed
                        by R0
      INC R0          ; Increment pointer
      DJNZ R3, loop1  ; Repeat until R3 is zero
      END
```

EXAMPLE 4.17

Write a program to store four 8 bit numbers in internal data RAM. Add these numbers and store the result in 55H and 56H. Assume that the least significant byte of the result is stored in low address.

```
The program is as follows
      ORG 0000H     ; Set program counter 0000H
      MOV 41H,#55H      ;     Store the first number in location 41H
      MOV 42H,#76H ; Store the second number in location 42H
      MOV 43H,#1AH ; Store the third number in location 43H
      MOV 44H,#9FH ; Store the fourth number in location 44H
      MOV R0,#41H  ; Store the first number  address 41H in R0
      MOV R5,#04H  ; Store the number 04H in R5
      CLR C        ; Clear the carry flag
      MOV 56H,#00H ; Store the number 00H in location 56H
```

```
      MOV 55H,#00H ; Store the number 00H in location 55
Loop: MOV A,55H    ; Store the contents of location 55H in A
      ADD A,@R0    ; Add contents of memory with the contents of A
      MOV 55H,A    ; Store the contents of A in location 55H
      MOV A,#00H   ; Store the number 00H in A
      ADDC A,56H   ; Add contents of 56H and the contents of carry flag
                     to A
      MOV 56H,A    ; Store the contents of A in location 56H
      INC R0       ; Increment contents of register R0
      DJNZ R5,Loop ; Decrement R5, if it is not zero, branch to loop
      END
```

EXAMPLE 4.18

Write a program to find the average of five 8 bit numbers. Store the result in 55H. (Assume that after adding five 8 bit numbers, the result is 8 bit only)

The program is as follows

```
      ORG 0000H    ; Set program counter 0000H
      MOV 40H,#05H ; Store the first number in location 40H
      MOV 41H,#55H ; Store the second number in location 41H
      MOV 42H,#06H ; Store the third number in location 42H
      MOV 43H,#1AH ; Store the fourth number in location 43H
      MOV 44H,#09H ; Store the fifth number in location 44H
      MOV R0,#40H  ; Store the first number address 40H in R0
      MOV R5,#05H  ; Store the number 05H in R5
      MOV B,R5     ; Store the number 05H in B
      CLR A        ; Clear the A
Loop: ADD A,@R0    ; Add contents of memory with the contents of A
      INC R0       ; Increment contents of register R0
      DJNZ R5,Loop ; Decrement R5, if it is not zero, branch to loop
      DIV AB       ; Divide the result by 5 (A)/(B)
      MOV 55H,A    ; Save the quotient  in location 55H
      END
```

EXAMPLE 4.19

Write a program to find the cube of an 8 bit number.

The program is as follows

```
ORG 0000H    ;  Set program counter 0000H
MOV R1,#N    ;  Load number to R1
MOV A,R1     ;  Load number to A
MOV B,R1     ;  Load number to B
```

```
MUL AB        ;  Square is computed
MOV R2,B      ;  Copy Contents of B(MSB of square) to R2
MOV B,R1      ;  Load number to B
MUL AB        ;  Multiply LSB of square with B
MOV 50,A      ;  Store result in 51H and 50H
MOV 51,B
MOV A,R2      ;  Copy MSB of square to A
MOV B,R1      ;  Copy number to B
MUL AB        ;  Multiply MSB of square with B
ADD A,51H     ;  Add LSB of result to location 51H
MOV 51H,A     ;  Store result in 52H and 51H
MOV 52H,B
MOV A,#00H    ;  Clear A
ADDC A,52H    ;  Add contents of CF to contents of 52H
MOV 52H,A     ;  Cube is stored at locations 52, 51 and 50
END
```

EXAMPLE 4.20

Write a program to multiply two 16 bit unsigned numbers and store the result at 60H–63H. Numbers are stored at 50H–51H and 52H–53H respectively. Assume that the least significant byte of data is stored in low address.

The 8051 provides only one multiplication instruction—MUL AB that multiplies the unsigned 8 bit integers of registers A and B. As shown in Fig. 4.1, to multiply two 16 bit unsigned numbers, the multiplier and the multiplicand must be broken down into 8 bit numbers as follows.

$$M = MH\ ML \text{ and } N = NH\ NL$$

where MH, ML, NH, and NL are the upper and lower 8 bit of M and N respectively. Four 8 bit multiplications must be performed and then partial products are added together.

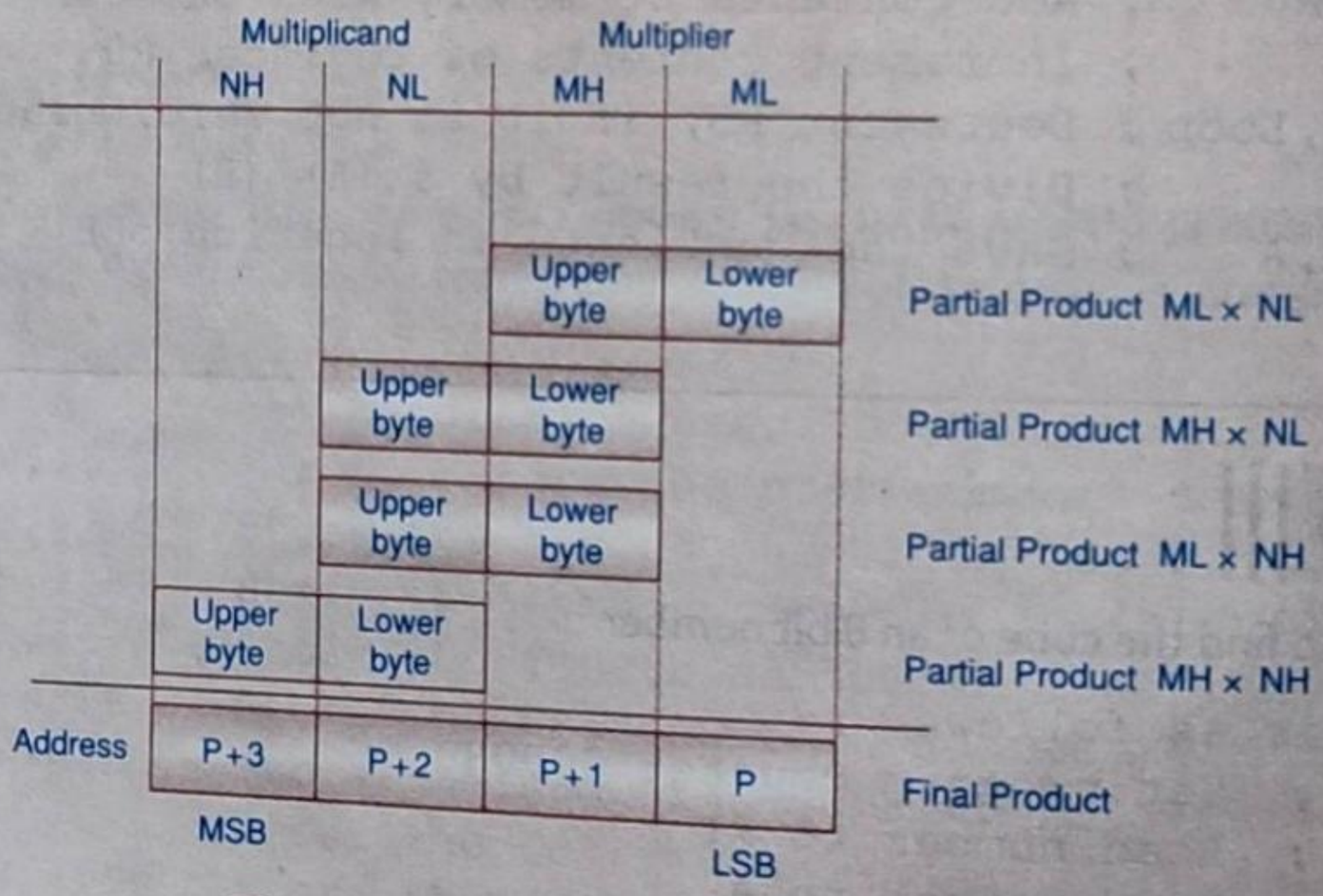

Figure 4.1 *16 bit by 16 bit multiplication*

The program is as follows

```
ORG 0000H       ; Set program counter 0000H
MOV A,50H       ; Place LSB of Multiplier in A
MOV B,52H       ; Place LSB of Multiplicand in B
MUL AB          ; Compute product
MOV 60H,A       ; Store LSB of the result
MOV 61H,B       ; Store MSB of the result
MOV A,51H       ; Place MSB of Multiplier in A
MOV B,53H       ; Place MSB of Multiplicand in B
MUL AB          ; Compute product
MOV 62H,A       ; Store LSB of the result
MOV 63H,B       ; Store MSB of the result
MOV A,51H       ; Place MSB of Multiplier in A
MOV B,52H       ; Place LSB of Multiplicand in B
MUL AB          ; Compute product
ADD A,61H       ; Add the result to P+1
MOV 61H,A       ; Store LSB of the result
MOV A,B
ADDC A,62H      ; Add the result to P+2
MOV 62H,A
MOV A,63H
ADDC A,#0H
MOV 63H,A       ; Store MSB of the result in P+3
MOV A,50H       ; Place LSB of Multiplier in A
MOV B,53H       ; Place MSB of Multiplicand in B
MUL AB          ; Compute product
ADD A,61H       ; Add the result to P+1
MOV 61H,A       ; Store LSB of result in P+1
MOV A,B
ADDC A,62H      ; Add the result to P+2
MOV 62H,A       ; Store the result in P+2
MOV A,63H
ADDC A,#0H      ; Add the result to P+3
MOV 63H,A       ; Store the result in P+3
END
```

EXAMPLE 4.21

Write a program to exchange the lower nibble of data present in external memory 6000H and 6001H.

The program is as follows

```
ORG 0000H          ; Set program counter 0000H
MOV DPTR, #6000H   ; Copy address 6000H to DPTR
MOVX A, @DPTR      ; Copy contents of 6000H to A
MOV R0,#45H        ; Load pointer, R0=45H
MOV @R0,A          ; Copy contents of A to internal data RAM pointed by R0
INC DPL            ; Increment pointer
MOVX A,@DPTR       ; Copy contents of 6001H to A
XCHD A,@R0         ; Exchange lower nibble of A with RAM pointed by R0
MOVX @DPTR,A       ; Copy contents of A to 6001H
DEC DPL            ; Decrement pointer
MOV A,@R0          ; Copy contents of internal data RAM pointed by R0 to A
MOVX @DPTR,A       ; Copy contents of A to data RAM pointed by DPTR
END
```

EXAMPLE 4.22

Write a program to count the number of 1's and 0's of 8 bit data stored in location 6000H.

The program is as follows

```
        ORG 0000H          ; Set program counter 0000H
        MOV DPTR, #6000H   ; Copy address 6000H to DPTR
        MOVX A, @DPTR      ; Copy number to A
        MOV R0,#08         ; Copy 08 in R0
        MOV R2,#00         ; Copy 00 in R2
        MOV R3,#00         ; Copy 00 in R3
        CLR C              ; Clear carry flag
BACK:   RLC A              ; Rotate contents of A through carry flag
        JC NEXT            ; If CF = 1, branch to next
        INC R2             ; If CF = 0, increment R2
        AJMP NEXT2
NEXT:   INC R3             ; If CF = 1, increment R3
NEXT2:  DJNZ R0,BACK       ; Repeat until R0 is zero
        END
```

EXAMPLE 4.23

An 8 bit code word is stored in location 1000H of external data memory. Code word is valid, if three MSBs are zero and it contains two ones in the remaining five bits. If code word is valid, store FF, else store 00 in 1001H.

SOLUTION

Code word is valid if three MSBs are zero and it contains two ones in the remaining five bits, for example–00010010, then code word is valid.

The program is as follows

```
        ORG 0000H                ; Set program counter 0000H
        MOV DPTR, #1000H         ; Copy address 1000H to DPTR
        MOVX A, @DPTR            ; Copy number to A
        MOV R1,A                 ; Copy number to R1
        INC DPTR
        MOV R2,#05               ; Copy 05 in R2
        MOV R3,#00               ; Copy 00 in R3
        MOV R4,#00               ; Copy 00 in R4
        CLR C                    ; Clear carry flag
        ANL A,#0EOH              ; Mask lower 5 bit
        CJNE A,#00,LOOP1         ; If first condition fails, branch to LOOP1
        MOV A,R1                 ; Check the second condition
BACK:   RRC A
        JNC LOOP2
        INC R3
LOOP2:  DNZ R2,BACK
        MOV A,R3
        CJNE A, #02H, LOOP1 ; If second condition fails, branch to LOOP1
                                 and store 00H
        DEC R4                   ; Code word is valid, to store FF, decrement R4
LOOP1:  MOV A,R4
        MOVX @DPTR,A
        END
```

EXAMPLE 4.24

Ten 8 bit numbers are stored in external data memory from location 5000H. Write a program to transfer a block of data to location 6000H.

SOLUTION

Assume ten 8 bit numbers are stored in external data memory from location 5000H, hence DPTR must be used as a pointer.

```
The program is as follows
        ORG 0000H      ; Set program counter 0000H
        MOV R4,#00H    ; Load 00H to R4
        MOV R5,#50H    ; Load 50H to R4
        MOV R2,#00H    ; Load 00H to R2
        MOV R3,#60H    ; Load 60H to R4
        MOV R6,#0AH    ; Load counter, R6=0AH
Loop1:  MOV DPL,R4     ; Copy contents of R5 and R4 to DPTR
        MOV DPH,R5
        MOVX A,@DPTR   ; Copy contents of data RAM pointed by DPTR to A
        MOV DPL,R2     ; Copy contents of R3 and R2 to DPTR
        MOV DPH,R3
        MOVX @DPTR,A   ; Copy contents of A to data RAM pointed by
DPTR
        INC R4         ; Increment R4
        INC R2         ; Increment R2
        DJNZ R6,Loop1  ; Repeat until R6 is zero
        END
```

EXAMPLE 4.25

Write a program to shift a 24 bit number stored at 57H–55H to the left logically four places. Assume that the least significant byte of data is stored in lower address.

```
The program is as follows
        ORG 0000H      ;  Set program counter 0000H
        MOV R1,#04     ;  Set up loop count to 4
again:  MOV A,55H      ;  Place the least significant byte of data in A
        CLR C          ;  Clear the carry flag
        RLC A          ;  Rotate contents of A (55H) left by one position
                          through carry
        MOV 55H,A
        MOV A,56H
        RLC A          ;  Rotate contents of A (56H) left by one position
                          through carry
        MOV 56H,A
        MOV A,57H
        RLC A          ;  Rotate contents of A (57H) left by one
                          position through carry
        MOV 57H,A
        DJNZ R1, again ;  Repeat until R1 is zero
        END
```

EXAMPLE 4.26

Write a program to find the smallest number of an array of N 8 bit unsigned numbers (N is an 8 bit number). The starting address of the array is at 2000H and stores the result in 2500H.

SOLUTION

N 8 bit numbers are stored in external data memory from location 2000H, hence DPTR must be used as a pointer.

ALGORITHM

```
Step 1: Initialise data in memory
Step 2: Initialise R4 with N-1
Step 3: Initialise DPTR with address of the array
Step 4: Load the element pointed by DPTR into A and increment DPTR
Step 5: Load the next element to register TEMP
Step 6: Compare contents of A with contents of TEMP
Step 7: If A is greater, then copy the next elements into A
Step 8: Increment the pointer and decrement R5 register
Step 9: Check if all the elements have been compared in the array
        (i.e. R5 = 0?)
Step 10: No, go to step 5
Step 11: If R4 is zero, terminate the program
```

The program is as follows

```
        ORG 0000H              ; Set program counter 0000H
        TEMP EQU 40H
           N EQU 04H           ; Array count
        MOV R4,#N-1            ; Load N-1 to R4
        MOV DPTR,#2000H        ; Store the starting address of the array in DPTR
        MOVX A,@DPTR           ; Copy first number to A
LOOP1:  MOV R1,A               ; Copy contents of A to R1
again:  INC DPTR               ; Increment DPTR
        MOVX A,@DPTR           ; Get the next number to A
        MOV TEMP,A             ; Copy the next number to TEMP
        MOV A,R1
        CJNE A,TEMP, LOOP2     ; (A)≠(TEMP) branch to LOOP2
        SJMP LOOP3             ; (A)=(TEMP) branch to LOOP3
LOOP2:  JC LOOP3               ; (A)<(TEMP) branch to LOOP3
        MOV A,TEMP             ; (A)>(TEMP) copy contents of TEMP to A
LOOP3:  DJNZ R4,LOOP1          ; Repeat until R4 is zero
        END
```