8051 PROGRAMS

1. Sum of 8-bit Numbers Stored in Memory

```
ORG 00H
MOV R0,#50H   ;get memory location in memory pointer R0
MOV R1,#51H   ;get memory location on memory pointer register R1
MOV A,@R0     ;get content of memory location 50H to accumulator
ADD A,@R1        ;add content of A with content of memory location 51H and store
result in A
MOV R0,#52H    ;get 52H to memory pointer R0
MOV@R0,A        ;copy content of A to memory location 52H
END
```

2. Add 16-bit Numbers

```
ORG 00H
MOV DPTR,#2040H  ; get 2040H into DPTR
MOV A,#2BH            ;get lower byte of second 16-bit number on accumulator
MOV R0,#20H           ;get higher byte of second 16-bit number on accumulator
ADD A,82H                ;[A]+[DPL]
MOV 82H,A               ;save result of lower byte addition
MOV A,R0                  ;get higher byte of second number in A
ADDC A,83H               ;[A]+[DPH]
MOV 83H,A               ;Save result of higher byte addition
END
```

3. Add 16-bit Numbers

```
MOV R0,#34H //LOWER NIBBLE OF NO.1
MOV R1,#12H //HIGHER NIBBLE OF NO.1
MOV R2,#0DCH //LOWER NIBBLE OF NO.2
MOV R3,#0FEH //HIGHER NIBBLE OF NO.2
CLR C
MOV A,R0
ADD A,R2
MOV 22H,A
MOV A,R1
ADDC A,R3
MOV 21H,A
MOV 00H,C
END
```

4. Multiplication and Division

```
MOV A,51H      ;get content of memory location 51H to accumulator
MOV 0F0H,52H;get content of memory location 52H to B register
MUL AB             ;multiply content of A with content of B
MOV 53H,A        ;get lower order byte of product in memory location 53H
MOV 54H,0F0H  ;get higher order byte of product in memory location in  54H
MOV A,51H         ;get content of memory location 51H to accumulator
MOV 0F0H,52H  ;get content of memory location 52H to register B
DIV AB                ;divide content of register A with register B
MOV 55H,A          ;Copy quotient of result to memory location 55H
MOV 56H,0F0H   ;copy remainder of result to memory location 56H
END
```

ORG 00H appears before the listing above:

5. Find Largest Number

```
              ORG 00H
         MOV DPTR,#2000H;initialize pointer to memory where numbers are stored
          MOV R0,#0AH        ; initialize counter
          MOV R3,#00H      ;maximum=0
AGAIN:    MOV A,@DPTR      ;get the number from memory
          CJNE A,R3,NE        ;compare number wi maximum number
          AJMP SKIP            ;if equal go to SKIP
    NE:   JC SKIP                  ;if not equal check for carry, if carry go to skip
          MOV R3,A             ;otherwise maximum=[[DPTR]]
  SKIP:   INC DPTR             ; Increment memory pointer
          DJNZ R0,AGAIN   ; Decrement count, if count=0 stop otherwise go to AGAIN
          END
```

6. Exchange the content of FFh and FF00h

```
MOV DPTR, #FF00H     ; TAKE THE ADDRESS IN DPTR

MOVX A, @DPTR           ; GET THE CONTENT OF FF0H IN A

MOV R0, 0FFH             ; SAVE THE CONTENT OF FFH IN R0

MOV 0FFH, A               ; MOVE A TO 50H
```

```
        MOV A, R0               ; GET CONTENT OF 50H IN A

        MOVX @DPTR, A           ; MOVE IT TO 0050H
```

7. Transfer the block of data from 20h to 30h to external location 1020h to 1030h.

```
    MOV R7, #0AH            ; INITIALIZE COUNTER BY 10D

        MOV R0, #20H            ; GET INITIAL SOURCE LOCATION

        MOV DPTR, #1020H        ; GET INITIAL DESTINATION LOCATION

    NXT:    MOV A, @R0          ; GET FIRST CONTENT IN ACC

        MOVX @DPTR, A           ; MOVE IT TO EXTERNAL LOCATION

        INC R0                  ; INCREMENT SOURCE LOCATION

        INC DPTR                ; INCREASE DESTINATION LOCATION

    DJNZ R7, NXT      ; DECREASE R7. IF ZERO THEN OVER OTHERWISE MOVE
    NEXT
```

8. Write an 8051 program to copy a block of 10 bytes of data from RAM locations starting at 35h to RAM locations starting at 60h.

```
MOV R0, #35h ;  Source pointer

MOV R1, #60h ; destination pointer

MOV R3, #0Ah ; counter

BACK: MOV A,@R0

MOV @R1, A

INC R0

INC R1

DJNZ R3, BACK

HERE: SJMP HERE

END
```

9. Write a program to check if the character string of length 7, stored in RAM locations 50H onwards is a Palindrome. If it is, output ‘Y’ to P1.

Solution:

A Palindrome is a string in which the characters are the same whether the string is read in the forward or backward direction. Example, ‘MADAM’, ‘RADAR’.

```
MOV R2, #03 ; take half the string length as counter value

MOV R0, #50H ; take R0 as pointer to the forward reading

MOV R1, #56H ; take R1 as pointer for the backward reading Of the string

Back: MOV A, @R0 ; move into A the character pointed by R0

MOV B, @R1 ; move into B the character  pointed by R1 24

CJNE A, B, NEXT ;compare it with the character pointed by R1
```

```
INC R0 ; increment the forward counter
DEC R1 ; decrement the backward counter
DJNZ R2, BACK ; repeat until all characters are compared
MOV P1, #’Y’ ; since the string is a Palindrome output ‘Y’
NEXT: NOP ; if not equal, do nothing since it is not a Palindrome
END
```

10. W rite the sequence of 8051 instructions to store any two numbers at two consecutive locations 70H and 71H, multiply them and store the result in location 72H.

```
      MOV R0, #70H;set source address 20H to R0
       MOV R1, #72H;set destination address 30H to R1
        MOV A, @R0;take the first operand from source to register A
        INC R0; Point to the next location
        MOV B, @R0 ;take the second operand from source to register B
        MUL A B  ;Multiply A and B
        MOV @R1, B; Store higher order byte to 30H
        INC R1; Increase R1 to point to the next location
        MOV @R1, A ;Store lower order byte to 31H
HALT:   SJMP HALT ; Stop the program
```

11. Write an 8051 program to count the number of 1s in the binary representation of a given number.

```
MOV DPTR,#9000H ;LOAD DPTR WITH 9000H
MOVX A,@DPTR ;MOVE DATA FROM EXTERNAL MEMORY LOCATION TO A
MOV R0,#0H ;LOAD R0 WITH 0
MOV R1,#8H ;LOAD R1 WITH 8
CLR C ;CLEAR CARRY BIT
UP:RLC A ;ROTATE A LEFT THROUGH CARRY
JNC NEXT ;IF NO CARRY, JUMP TO LABEL NEXT
INC R0 ;INCREMENT R0
NEXT:DJNZ R1,UP ;DECREMENT R1, AND JUMP TO LABEL NEXT, IF R1≠0
INC DPTR ;INCREMENT DPTR
MOV A,R0 ;MOVE DATA FROM R0 TO A
MOVX @DPTR,A ;MOVE DATA FROM A TO EXTERNAL MEMORY LOCATION
HERE:SJMP HERE
END
```

12. Write an assembly language program to sort an array of N =____ h bytes of data in

ascending/descending order stored from location 9000h. (Using bubble sort algorithm)
LET N = 06H

```
 MOV R0,#05H //COUNT (N-1) ARRAY SIZE = N
LOOP1: MOV DPTR, #9000H //ARRAY STORED FROM ADDRESS 9000H
 MOV R1,#05H //INITIALIZE EXCHANGE COUNTER
 LOOP2: MOVX A, @DPTR //GET NUMBER FROM ARRAY AND STORE IN REGISTER
 MOV B, A
 INC DPTR
 MOVX A, @DPTR //NEXT NUMBER IN THE ARRAY
 CLR C //RESET BORROW FLAG
 MOV R2, A //STORE IN R2
 SUBB A, B //2ND-1 ST NO, SINCE NO COMPARE INSTRUCTION IN 8051
 JNC NOEXCHG // JC - FOR DESCENDING ORDER
 MOV A,B //EXCHANGE THE 2 NOS IN THE ARRAY
 MOVX @DPTR,A
 DEC DPL //DEC DPTR - INSTRUCTION NOT PRESENT
 MOV A,R2
 MOVX @DPTR,A
 INC DPTR
NOEXCHG: DJNZ R1,LOOP2 //DECREMENT COMPARE COUNTER
 DJNZ R0,LOOP1 //DECREMENT PASS COUNTER
 END
```

WRITE AN ASSEMBLY LANGUAGE PROGRAM TO FIND THE SQUARE OF A GIVEN NUMBER N.

LET N = 05

```
MOV A,#05 // A=N=05
MOV B,A
MUL AB
MOV 30H,A // RESULT IS STORED IN 30H AND 31H
MOV 31H,B
END
```

13. Write an assembly language program to count number of ones and zeros in a eight bit number.

```
    MOV R1,#00H // TO COUNT NUMBER OF 0S
```

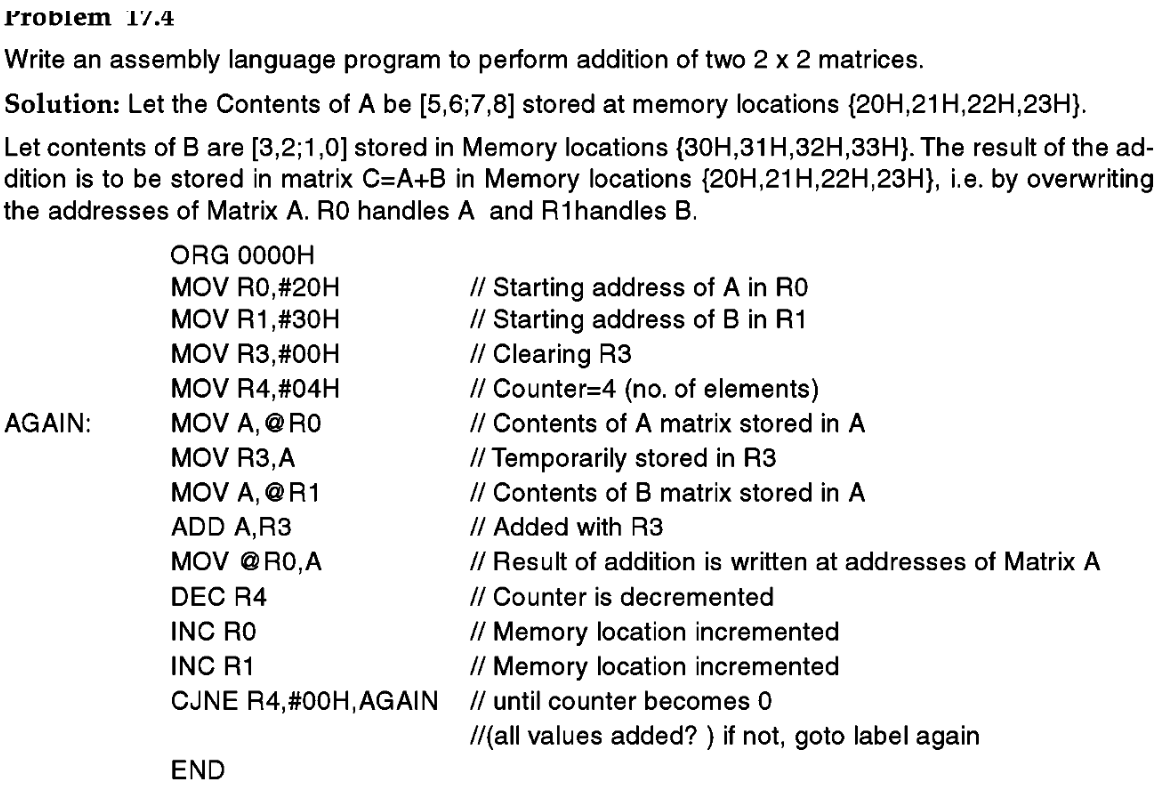
```
MOV R2,#00H // TO COUNT NUMBER OF 1S
MOV R7,#08H // COUNTER FOR 8-BITS
MOV A,#97H // DATA TO COUNT NUMBER OF 1S AND 0S
AGAIN: RLC A
JC NEXT
INC R1
SJMP HERE
NEXT: INC R2
HERE: DJNZ R7,AGAIN
END
```

14. Write an ALP to compare two eight bit numbers NUM1 and NUM2 stored in external memory locations 8000h and 8001h respectively. Reflect your result as: If NUM1<NUM2, SET LSB of data RAM location 2FH (bitaddress 78H). If NUM1>NUM2, SET MSB of location 2FH (bit address7FH). If NUM1 = NUM2, then Clear both LSB & MSB of bit addressable memory location 2FH.

```
MOV DPTR,#8000H
MOVX A,@DPTR
MOV R0,A
INC DPTR
MOVX A,@DPTR
CLR C
SUB A,R0
JZ EQUAL
JNC SMALL
SETB 7FH
SJMP END1
SMALL: SETB 78H
SJMP END1
 EQUAL: CLR 78H
CLR 7FH
 END1:
END
```

15. Write an assembly language program to perform logical operations AND, OR, XOR on two eight bit numbers stored in internal RAM locations 21h, 22h.

```
MOV A, 21H //DO NOT USE #, AS DATA RAM 21H IS TO BE ACCESSED
ANL A, 22H //LOGICAL AND OPERATION
MOV 30H, A //AND OPERATION RESULT STORED IN 30H
MOV A, 21H
ORL A,22H //LOGICAL OR OPERATION
MOV 31H, A //OR OPERATION RESULT STORED IN 31H
MOV A,21H
XRL A,22H //LOGICAL XOR OPERATION
MOV 32H,A // XOR OPERATION RESULT STORED IN 32H
 END
```



Problem 17.4

Write an assembly language program to perform addition of two 2 x 2 matrices.

Solution: Let the Contents of A be [5,6;7,8] stored at memory locations {20H,21H,22H,23H}.

Let contents of B are [3,2;1,0] stored in Memory locations {30H,31H,32H,33H}. The result of the addition is to be stored in matrix C=A+B in Memory locations {20H,21H,22H,23H}, i.e. by overwriting the addresses of Matrix A. R0 handles A and R1handles B.

```
        ORG 0000H
        MOV R0,#20H          // Starting address of A in R0
        MOV R1,#30H          // Starting address of B in R1
        MOV R3,#00H          // Clearing R3
        MOV R4,#04H          // Counter=4 (no. of elements)
AGAIN:  MOV A,@R0            // Contents of A matrix stored in A
        MOV R3,A             // Temporarily stored in R3
        MOV A,@R1            // Contents of B matrix stored in A
        ADD A,R3             // Added with R3
        MOV @R0,A            // Result of addition is written at addresses of Matrix A
        DEC R4               // Counter is decremented
        INC R0               // Memory location incremented
        INC R1               // Memory location incremented
        CJNE R4,#00H,AGAIN   // until counter becomes 0
                             //(all values added? ) if not, goto label again
        END
```