Lab-1

1) Write a ALP to multiply content located at 20H, 21H and stone im 22H and 23H.

```
0000H    MOV A, 20H
0002H    MOV B, 21H
0005H    MUL AB
0006H    MOV 22H, A
0008H    MOV 23H, B
000BH    HERE: SJMP HERE
```

Address	Data
20H	FF
21H	05
22H	FB
23H	04

2) Write a ALP to divide 75H by 12H and store quotient im R0 and remainder im R1 of RB3.

```
0000H    MOV A, #75H
0002H    MOV B, #12H
0005H    DIV AB
0006H    SETB PSW.3
0008H    SETB PSW.4
000AH    MOV R0, A
000BH    MOV R1, B
000DH    HERE: SJMP HERE
```

Address	Data
20H	FF
21H	05
R0	06H
R1	09H

3) Write a ALP to add content of 20H, 21H address and store at 22H and 23H.

```
0000H    MOV  A, 20H
0002H    MOV  R0, 21H
0004H    ADD  A, R0
0005H    MOV  22H, A
0007H    JNC  HERE
0009H    MOV  23H, #01H
000CH    HERE : SJMP HERE
```

Address	Data
20H	FF
21H	05
22H	04
23H	01

4) Add content of location 2000H and 2001H and store the result at 2002H and carry at 2003H.

```
0000     MOV  DPTR, #2000H
0003     MOV  A, @DPTR
0004     MOV  B, A
0006     INC  DPTR
0007     MOVX A, @DPTR
0008     ADD  A, B
000A     INC  DPTR
000B     MOVX @DPTR, A
000C     INC  DPTR
000D     ADDC A, #00H
000F     MOVX @DPTR, A
0010     HERE : SJMP HERE.
```

Address	Data.
2000	10
2001	FF
2002	0F
2003	01

5) Transfer block of 10 byte to 20H to 30H.

```
0000        MOV  R0, #20H
0002        MOV  R1, #30H
0004        MOV  R2, #10
            TRANSFER_LOOP:
0006        MOV  A, @R0
0007        MOV  @R1, A
0008        INC  R0
0009        INC  R1
000A        DJNZ R2, TRANSFER_LOOP
000C        HERE : SJMP HERE
```

	0	1	2	3	4	5	6	7	8	9
20	1	2	3	4	5	6	7	8	9	10
30	1	2	3	4	5	6	7	8	9	10

INT RAM

20	14
21	44
22	59
23	41
24	73
25	54
26	69
27	83
28	71
29	94

EXT RAM

2000	14
2001	44
2002	59
2003	41
2004	73
2005	54
2006	69
2007	83
2008	71
2009	94

20	79
21	14
22	44
23	59
24	41
25	73
26	54
27	69
28	83
29	94

2000	79
2001	14
2002	44
2003	59
2004	41
2005	73
2006	54
2007	69
2008	83
2009	94

Lab-2

1. Write an ALP in 8051 to transfer a block of 10 bytes from INT RAM (starting with location 20H) to EXT RAM (starting with location 2000H).

```
MOV R0, #20H
MOV DPTR, #2000H
MOV R1, #0AH
NEXT: MOV A, @R0
MOVX @DPTR, A
INC R0
INC DPTR
DJNZ R1, ~~LOOP~~ NEXT
HERE: SJMP HERE
END
```

2. Write an ALP in 8051 to transfer a block of 10 bytes from EXT RAM (storing with location 2000H) to INT RAM (storing with location 20H).

```
MOV R0, #20H
MOV DPTR, #2000H
MOV R1, #0AH
NEXT: MOVX A, @DPTR
MOV @R0, A
INC R0
INC DPTR
DJNZ R1, NEXT
HERE: SJMP HERE
END
```

1579	30
157A	51
157B	67
157C	19
157D	73
157E	55
157F	79
1580	43
1581	56
1582	78

3513	30
351A	51
351B	67
351C	19
351D	73
351E	55
351F	79
3520	43
3521	56
3522	78

3. Write ALP to transfer a block of 10 bytes from EXT RAM location starting with 1579H to EXT RAM location starting with 3579 H.

```
MOV  R1, #15H
MOV  R0, #79H
MOV  R3, #35H
MOV  R2, #79H

MOV  R5, #0AH

NEXT: MOV  DPH, R1
MOV  DPL, R0
MOVX  A, @DPTR
INC  DPTR
MOV  R1, DPH
MOV  R0, DPL

MOV  DPH, R3
MOV  DPL, R2
MOVX  @DPTR, A
INC  DPTR
MOV  R3, DPH
MOV  R2, DPL

DJNZ  R5, NEXT
HERE : SJMP  HERE.
```

ROM		RAM	
0250 H	4D	40 H	54
0251 H	4E	41 H	49
0252 H	4E	42 H	4E
0253 H	49	43 H	4E
0254 H	54	44 H	4D
0255 H	00		
0256 H	-		

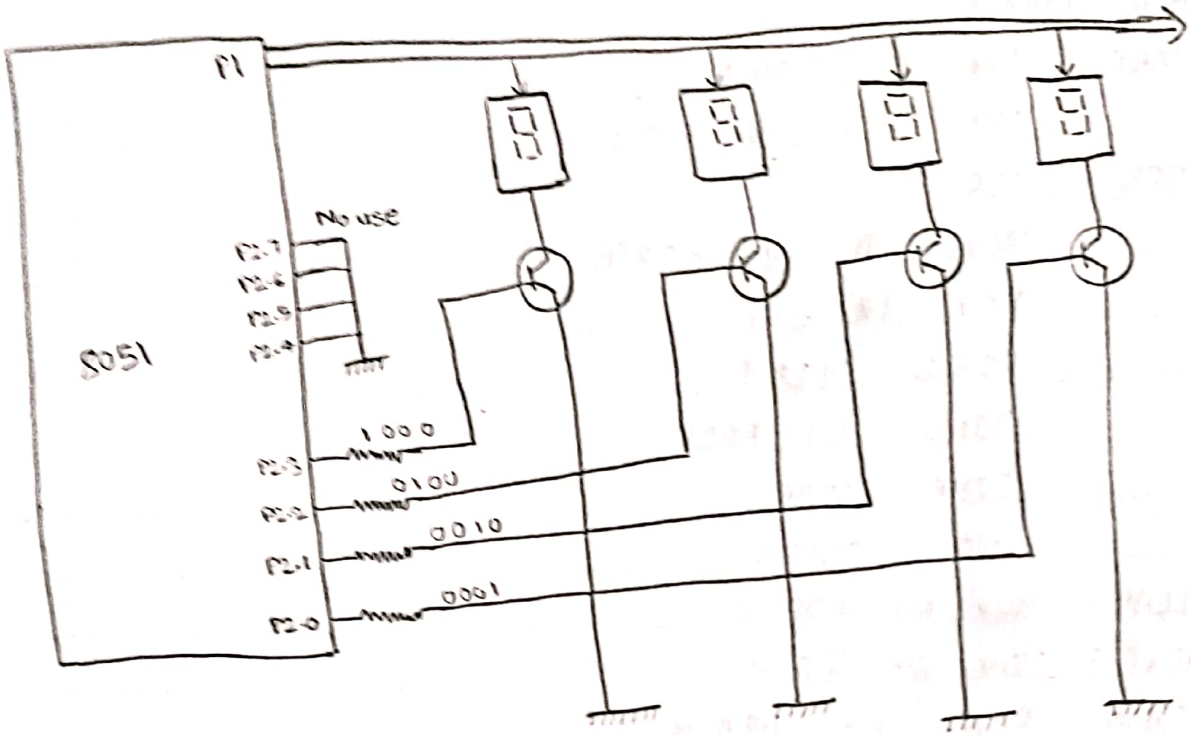
Lab - 3

1) Assume that ROM space 0250H contains "MNNIT". Write an ALP to transfer the byte in reverse order into RAM location starting at 40H. [Note: You are not allowed to use any register as loop counter.]

```
ORG 0000H
MOV R0, #40H
MOV DPTR, #0256H
LOOP1: CLR A
MOVC A, @A+ DPTR
INC DPTR
INC R0
CJNE A, #00H, LOOP1

DEC R0
DEC R0
MOV DPTR, #0250H

LOOP2 : CLR A
MOVC A, @A+ DPTR
MOV @R0, A
DEC R0
INC DPTR
CJNE A, #00H, LOOP2
HERE : SJMP HERE

ORG 0250H
DB 'MNNIT\0'
END
```

2) Find out the common cathode "seven segment" code for the numbers 0 to 9 (in the form of lookup table).

```
ORG   0000H
START: MOV   R7, #0AH
       MOV   DPTR, #0400H
BACK : CLR   A
       MOVC  A, @A+DPTR
       MOV   P1, A
       ACALL DELAY
       DJNZ  R7, BACK
       SJMP  START
       ORG   0300H
DELAY: MOV   R5, #0FFH
NEXT : MOV   R4, #05H
AGAIN: DJNZ  R4, AGAIN
       DJNZ  R5, NEXT
       RET
ORG   0400H
DB    3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 6FH
END
```



Lab-4

Q.1) Write an assembly program to display a constant 4 digit number on 4 seven segment display.

```
ORG   0000H
MOV   DPTR , #0400H
LOOP: MOV   A , #03H
MOVC  A , @A + DPTR
MOV   P1 , A
MOV   P2 , #08H
ACALL DELAY
MOV   A , #00H
MOVC  A , @A + DPTR
MOV   P1 , A
MOV   P2 , #04H
MOV   DELAY
MOV   A , #05H
MOVC  A , @A + DPTR
MOV   P1 , A
MOV   P2 , #02H
ACALL DELAY
MOV   A , #07
MOVC  A , @A + DPTR
MOV   P1 , A
MOV   P2 , #01H
ACALL DELAY
SJMP  LOOP
```

30H	1	87	01
31 H	4		
32 H	0		
33 H	4		
34 H	1		

```
ORG 0300H
DELAY : MOV R4, #0FFH
NEXT : DJNZ R4, NEXT
RET

ORG 0400H
DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 6FH
END
```

2. Check if number is palindrome or not.

```
ORG 0000H
MOV DPTR, #30H   ; point to the start of the number in memory
MOV R0, #30H     ; Pointer to the first digit
MOV R1, #34H     ; Pointer to the last digit

CHECK_LOOP:
        MOVX A, @R0
        MOVX B, @R1
        CJNE A, B, NOT_PALINDROME
        INC R0
        DEC R1
        CJNE R0, R1, CHECK_LOOP

PALINDROME :
        MOV R7, #01H    ; Set result to 01H (palindrome)
        SJMP END

NOT_PALINDROME : MOV R7, #00H    ; not a palindrome
END : SJMP END
END
```

Lab-5

1. Check if a string is palindrome or not.

```
ORG   0000H

MOV   R7, #00H
MOV   DPTR, #0300H
AGAIN: MOV   A, R7
       MOVC  A, @A+DPTR
       JZ    HERE
       INC   R7
       SJMP  AGAIN

HERE: MOV  R0, #00H
      MOV  DPTR, #0300H
      MOV  A, R7
      MOV  B, #02H
      DIV  AB
      MOV  R1, A

BACK: DEC  R7
      MOV  A, R0
      MOVC A, @A+DPTR
      MOV  B, A
      MOV  A, R7
      MOVC A, @A+DPTR
      CJNE A, B, NOT_PALINDROME
      INC  R0
      DJNZ R1, BACK
      MOV  A, #0FFH
      SJMP SKIP
```

11

```
NOT_PALINDROME: MOV A, #00H

SKIP: SJMP SKIP

ORG 0300H
DB 'RACECAR', 0
END
```

A FFH

2. Write an ALP in 8051 to take the count from location 32H and store the fibonacci series from 33H onwards.

```
ORG 0000H

MOV R3, 032H
MOV R1, #00H
MOV R2, #01H
MOV R0, #33H

NEXT: MOV A, R1
MOV @R0, A
INC R0
MOV A, R1
ADD A, R2
MOV B, R2
MOV R1, B
MOV R2, A
DJNZ R3, NEXT

HERE: SJMP HERE
END.
```

32H	05
33H	00
34H	01
35H	01
36H	02
37H	03

3. Series of 5 is stored in location starting from 20H, Write an ALP to sort them in ascending order (use bubble sort).

20H	05		20H	01
21H	04	→	21H	02
22H	03		22H	03
23H	02		23H	04
24H	01		24H	05

```
ORG 0000H
MOV R3, #05H
DEC R3

UP1 : MOV R2, #05H
      DEC R2
      MOV R0, #20H
UP :  MOV A, @R0
      INC R0
      MOV B, @R0
      CJNE A, 0F0H, AGAIN
AGAIN : JC DOWN
        XCH A, @R0
        DEC R0
        MOV @R0, A
        INC R0
DOWN : DJNZ R2, UP
       DJNZ R3, UP1
H: SJMP H
END
```

Lab - 6

1. a) Addition of 2 8-bit numbers.

```
ORG 0000H
MOV A, #08H
MOV R0, #08H
ADD A, R0
MOV P2, A
HERE: SJMP HERE
```

b) Addition of 2 16-bit numbers and displaying result on port P0 and P2.

```
; num1 = 1234H , num2 = 7856H
ORG 0000H

MOV R0, #41H
MOV R1, #51H

MOV A, @R0
ADD A, @R1
MOV R3, A
DEC R0
DEC R1
MOV A, @R0
ADDC A, @R1
MOV R4, A
MOV A, #00H
ADDC A, #00H
MOV R5, A
```

```
MOV  R0, #60H
MOV  A , R5
MOV  @R0, A
INC  R0
MOV  A, R4
MOV  @R0, A
INC  R0
MOV  A, R3
MOV  @R0, A

HERE: SJMP HERE
END
```

40H – 12H
41H – 34H
50H – 78H
51H – 56H

60H – 00H
61H – 8AH
62H – 8AH

Q.2) White am ALP to toggle the port P2 with the delay of 1 secomd. Visualize the program usimg proteus. (XTAL = 12 MHz).

```
ORG  0000H

L1: MOV  P2, #55H
ACALL  DELAY
MOV  P2, #0AAH
ACALL  DELAY
SJMP  L1
```

```
DELAY: MOV R7, #0C8H
MOV TMOD, #01H
L2: MOV TH0, #0DBH
MOV TL0, #0FFH
SETB TCON.4
L3: JNB TCON.5, L3
CLR TCON.4
CLR TCON.5
DJNZ R7, L2
RET
```

3. 2 16-bit numbers are stored in ROM space consecutively starting from location 0250H. Perform 16-bit multiplication of these 2 numbers and store the result in RAM space starting from location 50H.

```
ORG 0H

MOV 30H, #0FFH
MOV 31H, #0FFH
MOV 40H, #0FFH
MOV 41H, #0FFH

; Q X S
MOV A, 31H
MOV B, 41H
MUL AB
MOV 53H, A
MOV 52H, B
```

30H	FFH
31H	FFH
40H	FFH
41H	FFH
50H	FFH
51H	FEH
52H	00H
53H	01H

P[31] Q[30]
R[41] S[40]

```
; PxR
MOV  A, 30H
MOV  B, 40H
MUL  AB
MOV 51H, A
MOV 50H, B

; PxS
MOV  A, 30H
MOV  B, 41H
MUL  AB
ADD  A, 52H
MOV 52H, A
MOV  A, B
ADDC  A, 51H
MOV  51H, A
JNC  SKIP1
INC  50H
SKIP1 :

; QxR
MOV  A, 31H
MOV  B, 40H
MUL  AB
ADD  A, 52H
MOV 52H, A
MOV A, B
ADDC  A, 51H
MOV 51H, A
JNC  SKIP2
INC  50H
SKIP2 :  SJMP SKIP2
END
```