Advance Microprocessor and Interfacing

L1(5)- Write a program that performs addition, subtraction, multiplication and division using procedure.

L2(5)- Write a program to find x^n

L3(1)- Interface 8255 with 8086 at an address 60H as a port A address. Interface 5 seven segment displays with 8255. Write ALP to display 1,2,3,4,5 over the 5 displays continuously as per their positions starting with 1 at LSB.

Group Members:

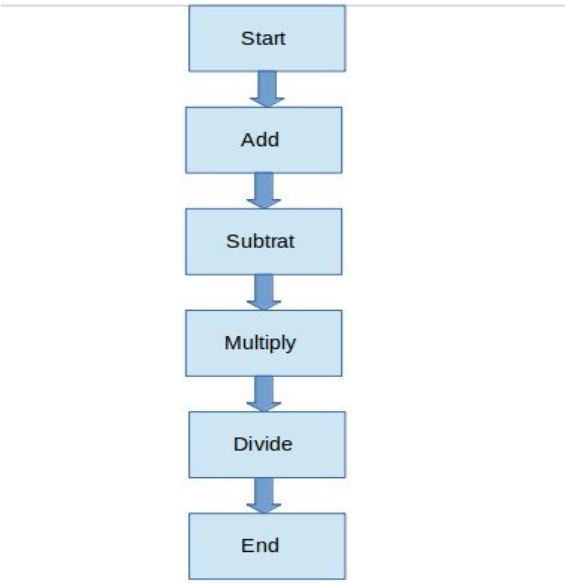
1. D Vasudha (BT17ECE014)
2. Mohammad Saad (BT17ECE048)
3. Harsh Nawandar (BT17ECE051)

Department of Electronics and Communication Engineering
Visvesvaraya National Institute of Technology, Nagpur

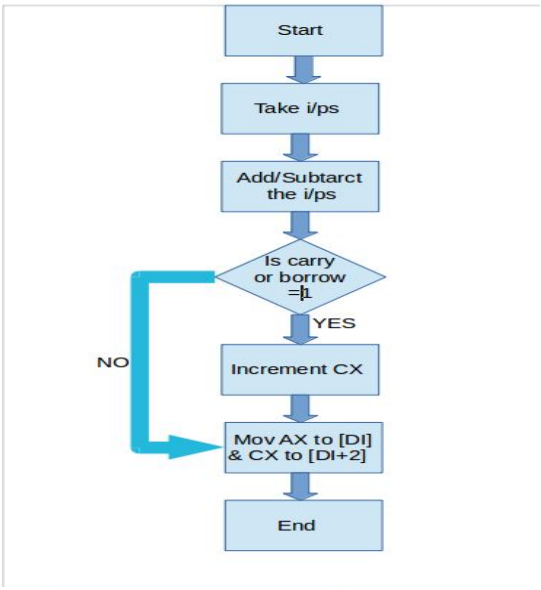
Date :15-05-2020

L1(5)- Write a program that performs addition, subtraction, multiplication and division using procedure.

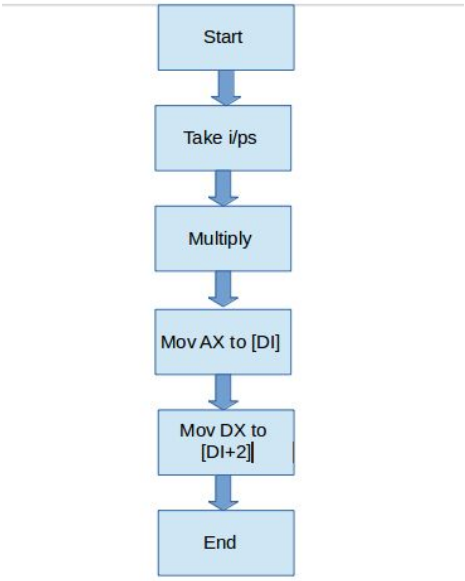
Main Program



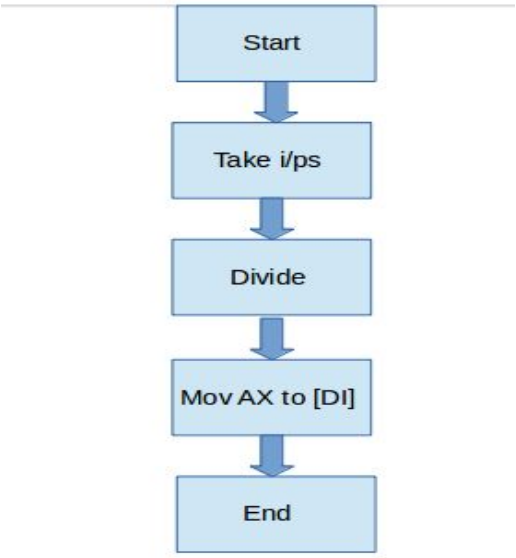
Add/ Subtract



Multiplication



Divide



Algorithm:

1. Main function consists of calling all the four procedures i.e ADD, SUBTRACT, MULTIPLY and DIVIDE.
2. More or less all the four procedures follow the same process. First we take i/p operands and perform the required operation.
3. In case of addition and subtraction, if carry or borrow is generated we increment another register and store the value.

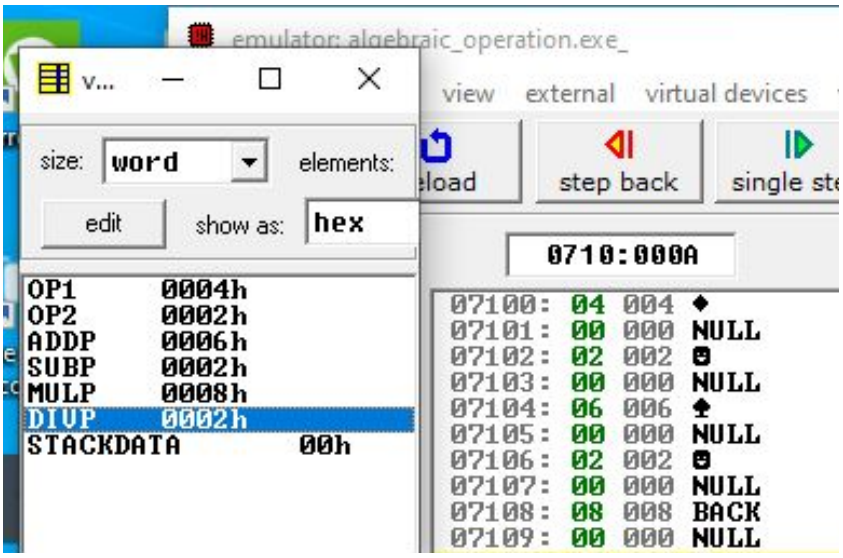
```
ASSUME CS:CODE ,DS:DATA,SS:STACK
DATA SEGMENT
        OP1 DW 00FFH
        OP2 DW 00FFH
        ADDP DW 01 DUP(?)
        SUBP DW 01 DUP(?)
        MULP DW 01 DUP(?)
        DIVP DW 01 DUP(?)
DATA ENDS

STACK SEGMENT
STACKDATA DB 100H DUP(?)
STACK ENDS

CODE SEGMENT
```

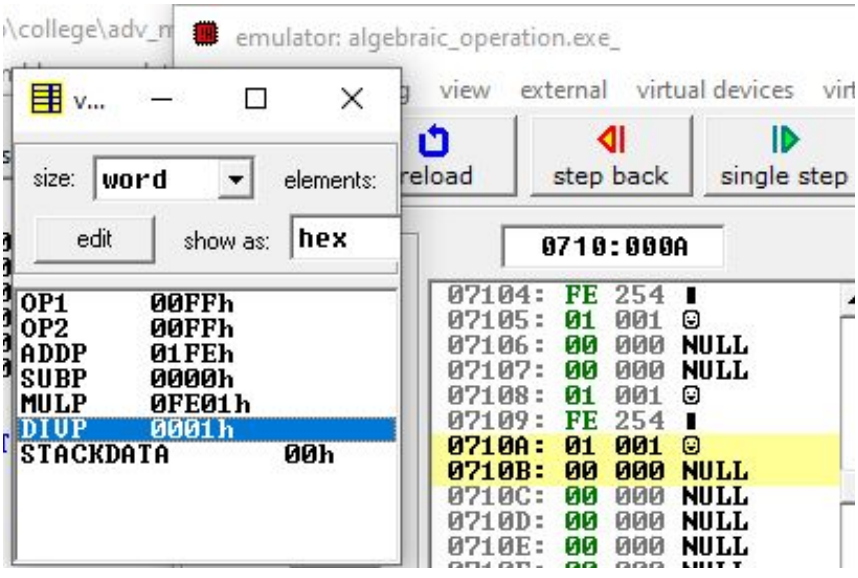
ADDRESS	**MNEMONICS**	**COMMENTS**
START	MOV AX,DATA	
	MOV DS,AX	
	MOV AX,STACK	
	MOV SS, AX	
	CALL ADD_NUMB	Calling ADD,SUB,MUL and DIV procedures
	CALL SUB_NUMB	
	CALL MUL_NUMB	

	CALL DIV_NUMB	
ADD_NUMB PROC NEAR	XOR CX,CX	
	MOV AX,OP1	
	MOV BX,OP2	
	ADD AX,BX	Add the operands
	JNC LOOP2	Check Carry
	INC CX	
LOOP2	MOV DI,OFFSET,ADDP	Move to assigned memory
	MOV [DI],AX	
	MOV [DI+2],CX	If carry exists increment the pointer and move data to it
	RET	
	ENDP	
SUB_NUMB PROC NEAR	XOR CX,CX	Clear the CX register
	MOV AX,OP1	
	MOV BX,OP2	
	SUB AX,BX	Sub the operands
	JNC LOOP1	Check Borrow
	INC CX	

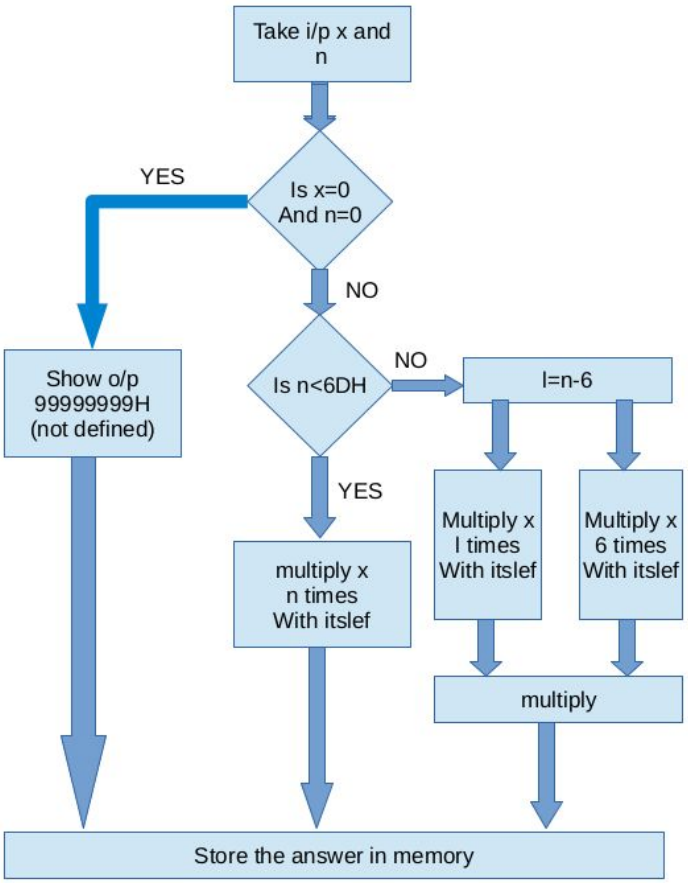
LOOP1	MOV DI,OFFSET SUBP	Move to assigned memory
	MOV [DI],AX	
	MOV [DI+2]],CX	
	RET	
	ENDP	
MUL_NUMB PROC NEAR	MOV AX,OP1	
	MOV BX,OP2	
	MUL BX	Multiply the operands
	MOV DI,OFFSET MULP	Move to assigned memory,16 bit multiplication
L1	MOV [DI],AX	
	MOV [DI+2]],DX	
	RET	
	ENDP	
DIV_NUMB PROC NEAR	MOV AX,OP1	
	MOV BX,OP2	
	XOR DX,DX	

	DIV BX	Divide the Operands
	MOV DI,OFFSET DIVP	Move to assigned memory
	MOV [DI],AX	
	RET	
	ENDP	
	CODE ENDS	
	END START	





L2-05- WRITE PROGRAM TO FIND x ^ n.



Algorithm:

1). We take the values of base (x) and power (n) from the data segment. {There is computational limitation for this mathematical expression so we take x<6D and n<12D}

2). We check whether both x and n are zero. If yes we store the answer as 99999999H as 0^0 is not defined.

3). If n<6D we simply run a loop which multiplies x n times with self in order to give x^n.

4). Else we compute x^n in two parts. We first find x^6 and then multiply it with x^(n-6) to get the final answer.

5).we store the obtained answer in the memory.

```
ASSUME  CS:CODE   DS:DATA
DATA SEGMENT
x        DB      06H     ;less than 6D
n        DB      0CH     ;less than 12D or 0C H
Op1      DW      01       DUP(?)
RESULT          DW      02        DUP(?)
DATA ENDS
```

CODE SEGMENT

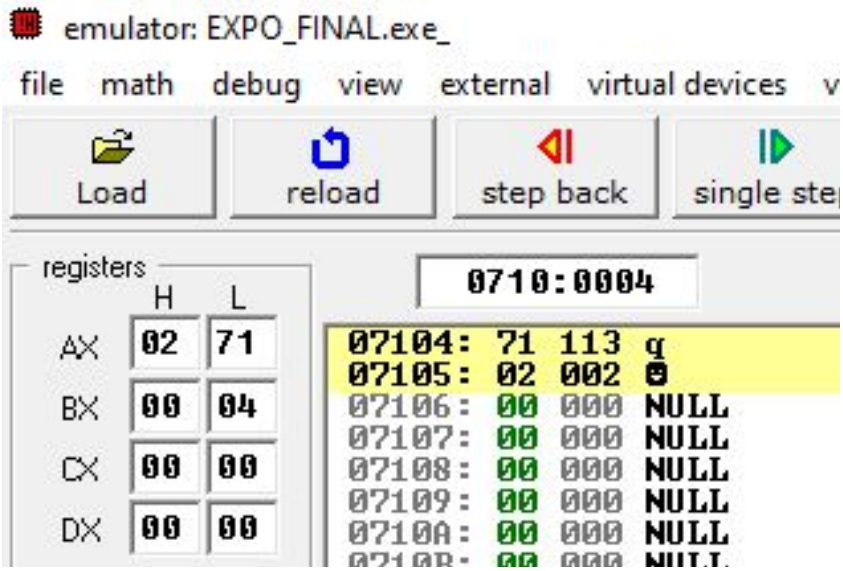
ADDRESS	MNEMONICS	COMMENTS
START	MOV AX, DATA	
	MOV DS, AX	
	MOV CL,x	Checking if x and n both are zero then jump to L7 else jump to L6
	CMP CL,00H	
	JA L6	
	MOV CL,n	
	CMP CL,01H	
	JB L7	
L6	MOV CL,n	If n is less than 6 then go to L1
	CMP CL, 06H	
	MOV BX, 0001H	
	JB L1	
	MOV CH , n	Finding l = n-6, and storing in CH
	SUB CH , 06H	
	MOV AL, x	Loading operand (x) in AL and BL
	MOV BL, x	
	XOR AH ,AH	Clearing the registers
	XOR DX ,DX	
	XOR BH ,BH	

L3	DEC CH	Multiplying x with itself l times (l = n-6)
	JZ L2	
	MUL BX	
	JMP L3	
L2	LEA DI , op1	Storing intermediate output at location op1
	MOV [DI] ,AX	
	MOV CL, 06H	Multiplying x by itself 6 times and Storing in BX
	MOV AL, 01H	
	MOV BL, x	
	XOR AH,AH	
	XOR DX,DX	
	XOR BH,BH	
L4	MUL BX	
	DEC CL	
	JNZ L4	
	MOV BX , AX	
	LEA DI, op1	Loading back intermediate output in AX
	MOV AX , [DI]	
	MUL BX	Multiply intermediate output by BX
	JMP OUT	
L1	MOV AL,01H	Loading the operand x
	MOV BL,x	
	XOR AH,AH	Clearing the registers

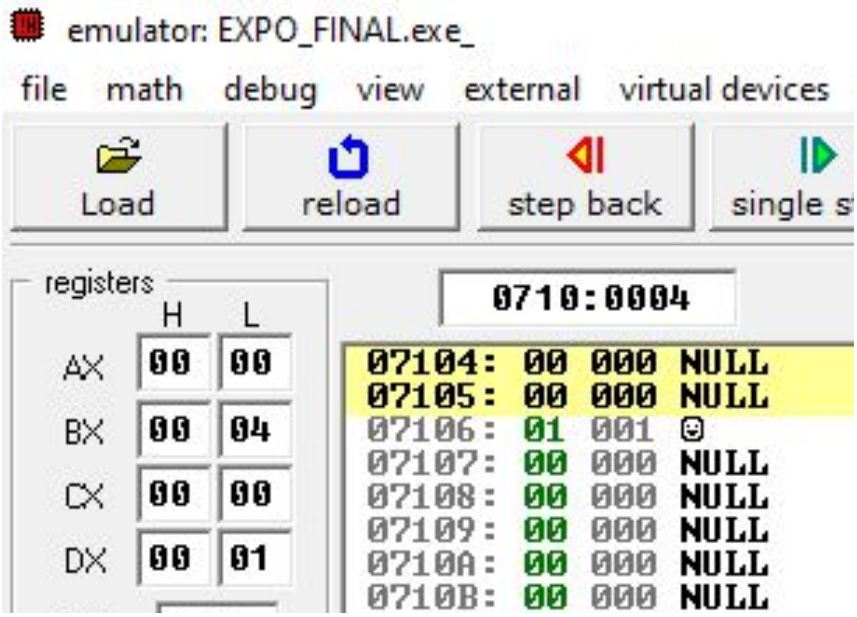
	XOR BH,BH	
L5	MUL BX	Multiplying x with itself n times
	DEC CL	
	JNZ L5	
	MOV DX , AX	
	XOR AX ,AX	
	JMP OUT	
L7	MOV AX,9999H	Storing the output as '99999999H' If both x and n are zero (i.e. invalid case)
	MOV DX,9999H	
OUT	LEA BX , RESULT	Loading the output at location RESULT
	MOV [BX],AX	
	MOV 02H[BX] , DX	
	CODE ENDS	
	END START	
	RET	

TEST CASES: (Output stored at location from 07107 to 07104)

1. x=5; n=4 (5^4 = 0625D = 00000271H)

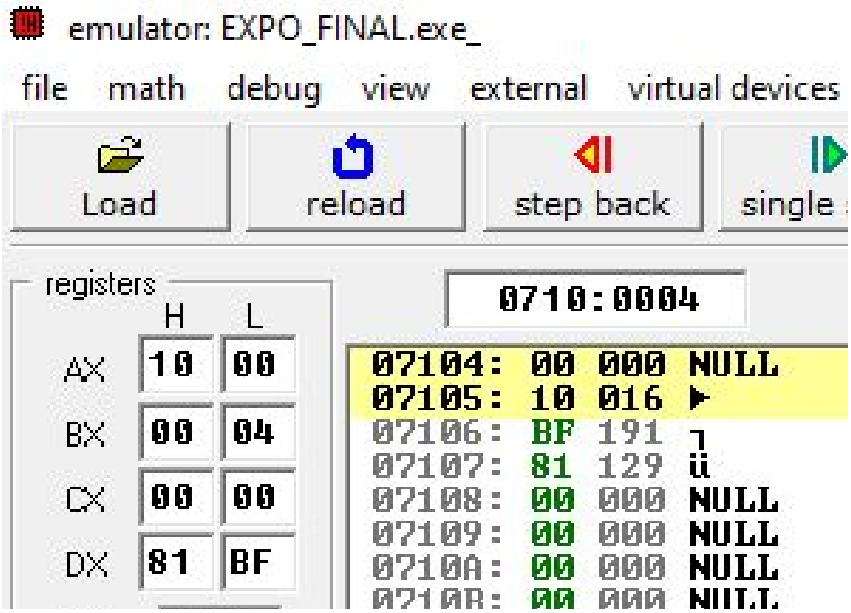


2. x=4; n=8 (4^8 = 65536D = 00010000H)



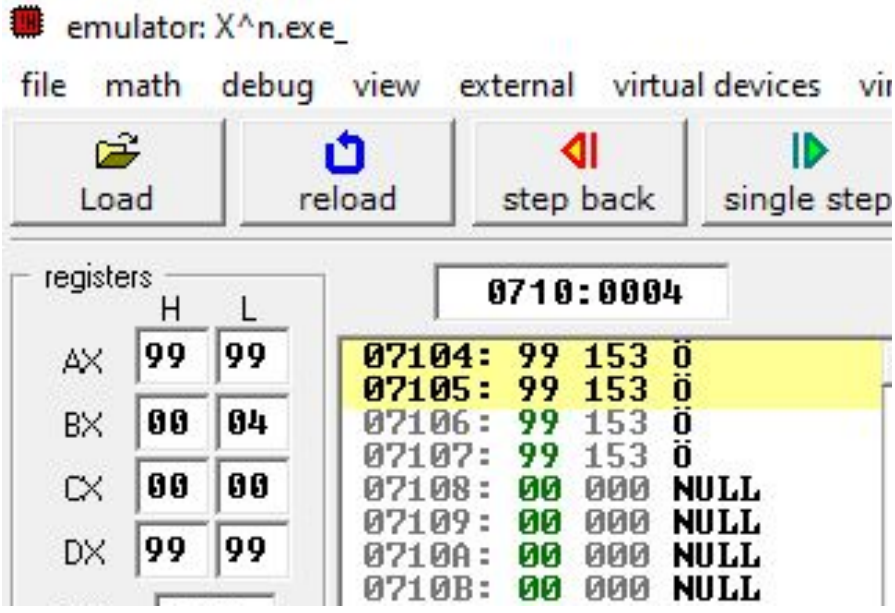
3. Extreme case

   x=6; n=12 (6^12 = 2176782336D = 81BF1000 H)

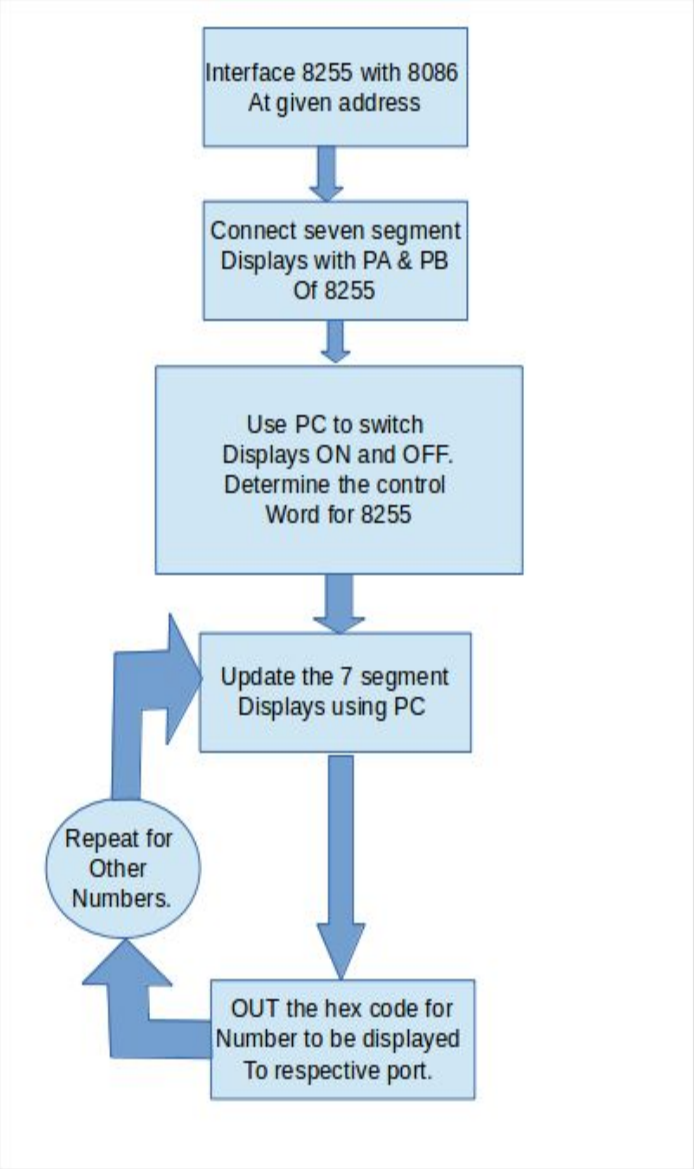


4. Invalid case

   x = 0 ; n=0 (0^0 is invalid and represented by 9999999H)



L-3-1). Interface 8255 with 8086 at an address 60H as a port A address. Interface 5 seven segment displays with 8255. Write ALP to display 1,2,3,4,5 over the 5 displays continuously as per their positions starting with 1 at LSB.



Algorithm:

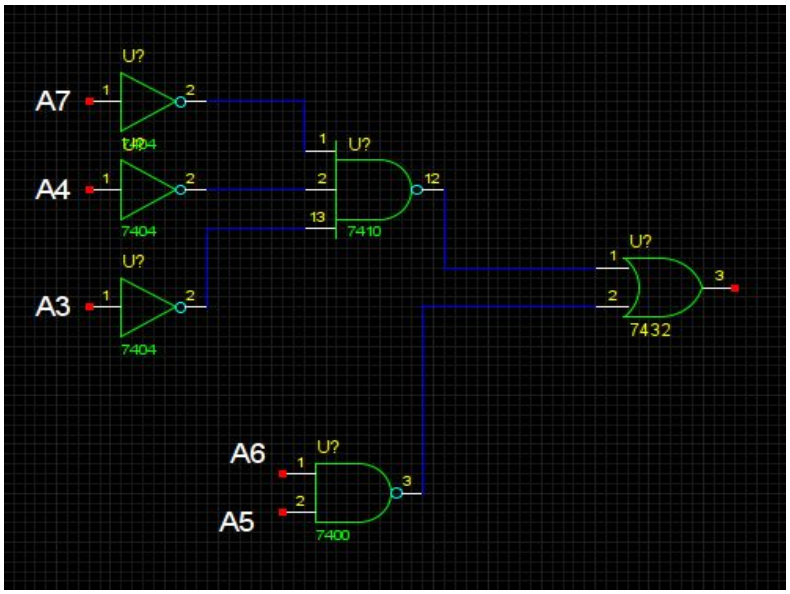
1. 8255 is interfaced with 8086 with PA address as 60H using proper chip select logic.
2. Seven segment displays(SSD) are connected with 8255 such that the SSD displaying 1,2,3 are connected to PA and SSD displaying 4,5 are connected to PB simultaneously.
3. PC is used to respectively switch SSDs ON and OFF (updating SSD in flow chart). And hence the control word is derived as 80H.
4. Now we simply OUT the HEX codes of respective numbers at the required port and repeat it.

Interfacing Description:

1). The design consists of 8086 microprocessor, 8253 Programmable Peripheral interface chip and Seven segment Display(SSD) of Common Cathode type.

2). 8255 is interfaced with 8086 in I/O mapped I/O mode. Now since we want Port A address as 60H,the Chip select logic is derived as follows:

Port	A7	A6	A5	A4	A3	A2	A1	A0	Address
Port A	0	1	1	0	0	0	0	0	60H
Port B	0	1	1	0	0	0	1	0	62H
Port C	0	1	1	0	0	1	0	0	64H
Control register	0	1	1	0	0	1	1	0	66H

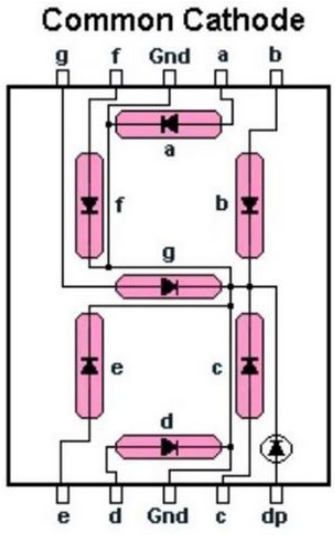


Seven segment displays(SSD) are connected with 8255 such that the SSD displaying 1,2,3 are connected to PA and SSD displaying 4,5 are connected to PB simultaneously. PC is used to respectively switch SSDs ON and OFF (updating SSD in flow chart). Hence all ports are acting as o/p ports. Since the 8255 is operated in basic i/o mode, so we derive the control word as :

1	**0**	**0**	**0**	**0**	**0**	**0**	**0**

80H.

3). Common cathode SSD:



Hence we need to send logic '1' (5V) to the LEDs that we want to be ON and in order to make display work, we send logic '0' (0V) to the GND pins.

Former is done with the help of Port A and Port B and the later is done with the help of Port C of 8255.

Seven Segment Hex Codes:

1=> 06H

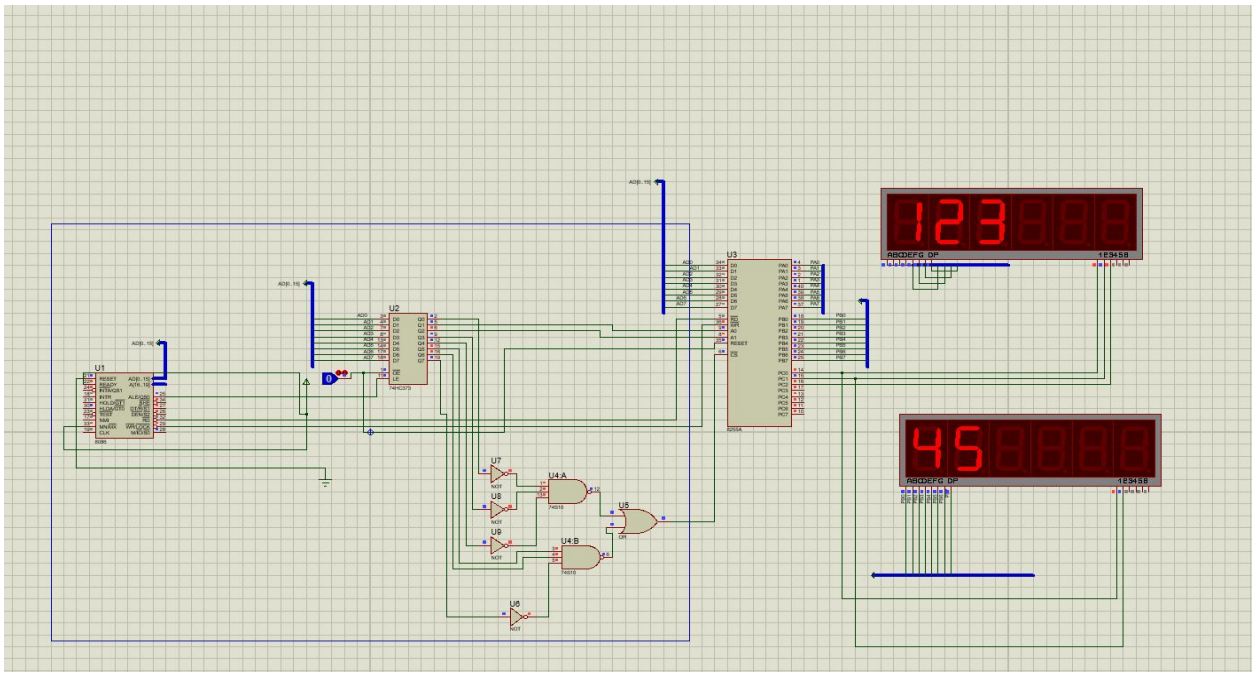
2=> 5BH
3=> 4FH
4=> 66H
5=> 6DH

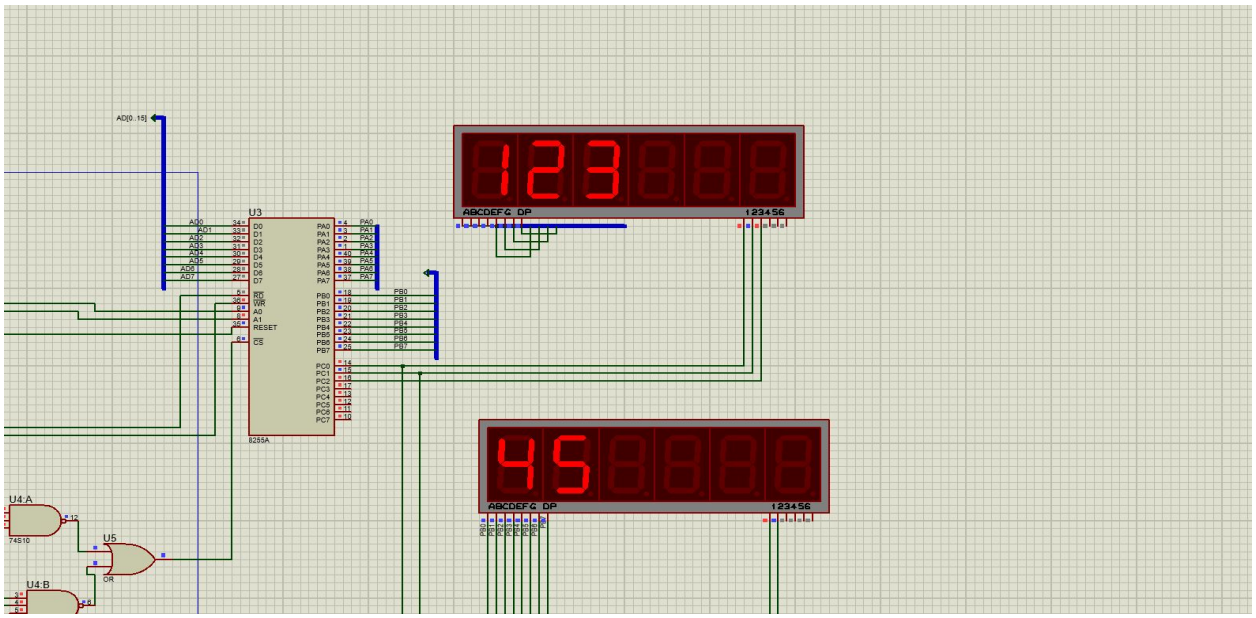
CODE:
ASSUME CS: CODE SEGMENT
CODE SEGMENT

ADDRESS	**MNEMONICS**	**COMMENTS**
START	MOV AL,80H	Control word for 8255
	OUT 66H,AL	
	MOV AL,7EH	Switching on SSD for ‘1’ and ‘4’ only using PC
	OUT 64H,AL	
	MOV AL,06H	Hex code for ‘1’
	OUT 60H,AL	Displayed using PA
	MOV AL,00H	Switching off 1st display
	OUT 60H,AL	
	MOV AL,66H	Hex code for ‘4’
	OUT 62H,AL	Displayed using PA
	MOV AL,00H	Switching off 4th display
	OUT 62H,AL	
	MOV AL,7DH	Switching on SSD for ‘2’ and ‘5’ only using PC
	OUT 64H,AL	
	MOV AL,53H	Hex code for ‘2’
	OUT 60H,AL	Displayed using PA

	MOV AL,00H	Switching off 2nd display
	OUT 60H,AL	
	MOV AL,60H	Hex code for ‘5’
	OUT 62H,AL	Displayed using PB
	MOV AL,00H	Switching off 5th display
	OUT 62H,AL	
	MOV AL,7BH	Switching on SSD for ‘3’only using PC
	OUT 64H,AL	
	MOV AL,4CH	Hex code for ‘3’
	OUT 60H,AL	Displayed using PA
	MOV AL,00H	Switching off 3rd display
	OUT 60H,AL	
	JMP START	
	COD ENDS	

SIMULATION AND PROTEUS SNAP:





Conclusion:

L1(5)- Input is taken as OP1 and OP2 , the procedures for addition , subtraction, multiplication and division is called each time and the calculations are done and stored in the assigned ADDP, SUBP, MULP, DIVP respectively

L2(5)-Program is successfully simulated to calculate x^n where x can have values less than equal to 6 and n can have values less than 12.
6^12 =2176782336D= 81BF1000H. This number is of 4 byte, hence we cant go beyond this limit as we can not multiply two numbers whose product is more than 4 bytes. Also , extreme case like 0^0 which is undefined is also considered in the code.

L3(1)-A system continuously displaying '1,2,3,4,5' was designed using 8086 processor, 8255 programmable peripheral and 5 seven segment displays. Simulation was performed on Proteus software.

References:

1. 'ADVANCED MICROPROCESSORS AND PERIPHERALS' by KM Burchandi
2. 'Microprocessor and interfacing' by D.V Hall
3. "http://www.circuitstoday.com/" for proteus

NOTE: Here is the Github link for all the codes . (Github- Repository)