RAJALAKSHMI ENGINEERING COLLEGE

Thandalam, Chennai – 602 105.

Department of Electronics and Communication Engineering

CS 6412 - MICROPROCESSORS & MICROCONTROLLERS LABORATORY

IV SEM
COMPUTER SCIENCE ENGINEEERING

NAME:	
ROLL NO.:	
REG. NO.:	
SEC:	

CS 6412 & IT 6411-MICROPROCESSORS & MICROCONTROLLERS LAB
IV Semester CSE & IT

Syllabus

8086 Programs using kits and MASM

1. Basic arithmetic and Logical operations
2. Move a data block without overlap
3. Code conversion, decimal arithmetic and Matrix operations.
4. Floating point operations, string manipulations, sorting and searching
5. Password checking, Print RAM size and system date
6. Counters and Time Delay

Peripherals and Interfacing Experiments

7. Traffic light control
8. Stepper motor control
9. Digital clock
10. Key board and Display
11. Printer status
12. Serial interface and Parallel interface
13. A/D and D/A interface and Waveform Generation

8051 Experiments using kits and MASM

14. Basic arithmetic and Logical operations
15. Square and Cube program, Find 2's complement of a number
16. Unpacked BCD to ASCII

CONTENT BEYOND SYLLABUS

- **Program using PIC MICROCONTROLLER – FLOW CODE**
- **8253 – TIMER INTERFACING**

LAB PLAN

S.No.	Cycle of Experiments	Experiments
		8086
1		Arithmetic and Logical operations
2		Move a data block with out overlap, Decimal arithmetic
3		Code conversion, Matrix operation
4		String Manipulations, Searching: a) Largest b) Smallest, Sorting: a) Ascending b) Descending
		MASM
5	**Cycle I**	Password checking, Print RAM size, System date
		INTERFACING
6		8279 - Keyboard/Display Controller
7		8255-PPI, 8251-USART
8	**Cycle II**	A/D and D/A interface and Waveform Generation
		8051
9		Arithmetic and Logical operations
10		Square, Cube, &2's complement of a number
11		Unpacked BCD to ASCII
		INTERFACING
12		Stepper motor
13	**Cycle III**	Traffic light controller

CONTENT BEYOND SYLLABUS

14. 8253 – TIMER INTERFACING
15. Program using PIC MICROCONTROLLER – FLOW CODE

WORKING PROCEDURE FOR 8086 MICROPROCESSOR (SCIENTECH)- Starting address 400-700:

Step: 1- press the Reset button.
Step: 2- For giving the starting address press A & type the starting address.
Step: 3- Then type the entire program.

Input:

Step: 4- To specify the input, Press Reset & type D then give address of the input data & specify the data.

Execution:

Step: 5- For executing the program, Press Reset & type G then specify the starting address & press Enter. (The program will be executed)

Output:

Step:6- To see the output, Press Reset & type D then specify the output address of data then press ENTER (It will shows the output).

To check entire prog. With opcode:

Reset & type U & type the starting address and then enter. Press down arrow to view the entire program.

WORKING PROCEDURE FOR 8051 MICROCONTROLLER (SCIENTECH KIT))- Starting address 4000-5000:

Step1: Reset & Press A. Then type ASM ORG and specify the starting address.
Step2: Press Enter & type the entire program.

Input:

Step3: To enter the input data, Reset & press M. Type the address then specify the data.

Execution:

Step4: To execute the program, Type GO space starting address & press enter (it will get executed).

Output:

Step5: To see the output, Reset & press M. Type the output address (where the output is stored).

To check entire prog. With opcode:

Reset & select d enter starting address to ending address.

WORKING PROCEDURE FOR MASM (MACRO ASSEMBLER)

Start
Run
cmd
Enter
c:cd\8086
c :8086>edit
Command box appears. Click escape and type the program
Save the file as filename.asm
Exit
To execute:
c:8086>masm filename.asm
Enter (4 times)
(Object file will be created)
c:8086>link filename.obj
Enter (4 times)
(Executable file will be created)
c:8086>debug filename.exe
g (go execute) – To execute entire program
t (to see step by step output)
d (specify the address range to see the output. Say, d 2000 2001)

CYCLE 1
8086
PROGRAMS

ADDITION				ADDITION			
	ADDRESS	DATA			ADDRESS	DATA	
INPUT			INPUT 1	INPUT			INPUT1
			INPUT 2				INPUT2
OUTPUT			SUM	OUTPUT			SUM
			CARRY				CARRY

MANUAL CALCULATION:-

SUBTRACTION				SUBTRACTION			
	ADDRESS	DATA			ADDRESS	DATA	
INPUT			INPUT 1	INPUT			INPUT1
			INPUT 2				INPUT2
OUTPUT			DIFFRENCE	OUTPUT			DIFFERENCE
			BORROW				BORROW

MANUAL CALCULATION:-

EXP. NO. 1 BASIC ARITHMETIC AND LOGICAL OPERATIONS

AIM:

To write a program to perform arithmetic and logical operations using 8086.

ALGORITHM:

1. Initialize input memory with input datas.
2. Get the first number from memory and store in Register.
3. Get the second number in memory and store it to the Register.
4. Perform arithmetic and logical operations on the datas and store it to the output memory.

PROGRAM:
ARITHMETIC OPERATIONS

ADDRESS	a.ADDITION	ADDRESS	b.SUBTRACTION

	MOV CX, 0000
	MOV SI, 0500
	MOV AX, [SI]
	ADD SI, 0002
	MOV BX, [SI]
	ADD AX, BX
	JNC L1
	INC CX
	L1: MOV DI, 0600
	MOV [DI], AX
	ADD DI, 0002
	MOV [DI], CX
	HLT
			MOV CX, 0000
			MOV SI, 0500
			MOV AX, [SI]
			ADD SI, 0002
			MOV BX, [SI]
			SUB AX, BX
			JNC L1
			INC CX
			L1: MOV DI, 0600
			MOV [DI], AX
			ADD DI, 0002
			MOV [DI], CX
			HLT

ADDRESS	**c.MULTIPLICATION**	**ADDRESS**	**d.DIVISION**

	MOV DX, 0000
	MOV SI, 0500
	MOV AX, [SI]
	ADD SI, 0002
	MOV BX, [SI]
	MUL BX
	MOV DI, 0600
	MOV [DI], AX
	ADD DI, 0002
	MOV [DI], DX
	HLT
			MOV DX, 0000
			MOV SI, 0500
			MOV DI, 0600
			MOV AX, [SI]
			ADD SI, 0002
			MOV BX, [SI]
			DIV BX
			MOV [DI], AX
			ADD DI, 0002
			MOV [DI], DX
			HLT

MULTIPLICATION				DIVISION			
	ADDRESS	DATA			ADDRESS	DATA	
INPUT			INPUT 1 (16-BIT)	INPUT			INPUT1
			INPUT 2 (16-BIT)				INPUT2
OUTPUT			PRODUCT (LSB 16 BIT)	OUTPUT			QUOTIENT
			PRODUCT (MSB 16 BIT)				REMAINDER

OR			AND		
	ADDRESS	DATA		ADDRESS	DATA
OUTPUT			OUTPUT		
XOR			NOT		
OUTPUT			OUTPUT		

MANUAL CALCULATION:-

LOGICAL OPERATIONS

ADDRESS	e.AND OPERATION	ADDRESS	f.OR OPERATION

	MOV AX, 0004
	MOV BX, 0002
	AND AX, BX
	MOV DI, 0500
	MOV [DI], AX
	HLT
			MOV AX, 0004
			MOV BX, 0002
			OR AX, BX
			MOV DI, 0500
			MOV [DI], AX
			HLT

ADDRESS	**g.XOR OPERATION**	**ADDRESS**	**h.NOT OPERATION**

	MOV AX, 0004
	MOV BX, 0002
	XOR AX, BX
	MOV DI, 0500
	MOV [DI], AX
	HLT
			MOV AX, 0004
			NOT AX
			MOV DI, 0500
			MOV [DI], AX
			HLT

RESULT:-

INPUT			OUTPUT	
ADDRESS	**DATA**		**ADDRESS**	**DATA**
		COUNT		
		DATA 1		
		DATA 2		
		DATA 3		
		DATA 4		

EXP. NO. 2 MOVE A DATA BLOCK WITHOUT OVERLAP AND DECIMAL ARITHMETIC

2a.AIM:

To write and execute an assembly language program to move a block of data without overlap using 8086

ALGORITHM:

1. Initialize counter
2. Initialize source block pointer
3. Initialize destination block pointer
4. Get the byte from source block
5. Store the byte in the destination block
6. Increment source, destination pointers and decrement counter
7. Repeat steps 4,5 and 6 unit counter equal to zero
8. Stop

PROGRAM:

ADDRESS	BLOCK DATA TRANSFER WITHOUT OVERLAP

	MOV SI, 0500
	MOV DI, 0600
	MOV CX, [SI]
	L1: ADD SI, 0002
	MOV AX, [SI]
	MOV [DI], AX
	ADD DI, 0002
	DEC CX
	JNZ L1
	HLT

RESULT:-

	ADDRESS	**DATA**	
INPUT			**INPUT 1**
			INPUT 2
OUTPUT			**SUM**
			CARRY

2b. AIM:

To write and execute an assembly language program for performing the decimal arithmetic.

ALGORITHM:

1. Initialize CX register and load the input address to index register.
2. Move the first data into the accumulator.
3. Increment twice the index register.
4. Move the second data in some register.
5. Move the second data in some register
6. If carry is generated increment the carry register.
7. Store the contents of accumulator and carry register in memory.
8. Stop

PROGRAM:

ADDRESS	DECIMAL ARITHMETIC
	MOV CX, 0000
	MOV SI,0500
	MOV AX, [SI]
	ADD SI, 0002
	MOV BX, [SI]
	ADD AX, BX
	DAA
	JNC L1
	INC CX
	L1: MOV DI, 0600
	MOV [DI], AX
	ADD DI, 0002
	MOV [DI], CX
	HLT

RESULT:-

EXP. NO. 3A **CODE CONVERSION**

AIM:

To write a program to perform code conversion.

ALGORITHM:

(i) BCD to Hexadecimal

1. Initialize the index register with input data
2. Load it to the accumulator
3. Mask the LSB nibbles and rotate it four times
4. Multiply it with 0AH
5. Mask the MSB nibbles
6. Add it with the result of step 4
7. Store the output in a memory location

(ii)Hexadecimal to BCD

1. Initialize the index register with input data
2. Load it to the accumulator
3. Separate hundredth digit by comparing with 64H
4. Increment a register for number of hundreds and subtract by 64H
5. Separate tens digit by comparing with 0AH
6. Increment a register for number of tens and subtract by 0AH
7. The remaining unit digit is stored to index register
8. Result of step 5 is rotated four times
9. Add it with the index register
10. Store the output in memory location

(iii)ASCII to Hexadecimal

1. Initialize the index register with the input data.
2. Load it to the accumulator
3. Mask the MSB nibbles
4. Compare with OAH
5. If carry generated add 30H, if no carry add 37H
6. Mask LSB nibbles
7. Repeat step 3 & 4
8. Store the output to a memory location.

(iv)Hexadecimal to ASCII

1. Initialize the index register with the input data.
2. Load it to the accumulator
3. Subtract 30H
4. Compare with 0AH
5. If no carry subtract 07H and store the result
6. If carry store the result of step 3 to a memory location.

.

BCD TO HEXADECIMAL			HEXADECIMAL TO BCD		
	ADDRESS	DATA		ADDRESS	DATA
INPUT			INPUT		
OUTPUT			OUTPUT		

	BCD to Hexadecimal		**Hexadecimal to BCD**
	MOV SI,1200		XOR BX,BX
	MOV AL,[SI]		XOR CX,CX
	AND AL,0F0		MOV SI,1200
	MOV CL,01		MOV AL,[SI]
	ROL AL,CL		HU: CMP AL,64
	ROL AL,CL		JC TE
	ROL AL,CL		SUB AL,64
	ROL AL,CL		INC BX
	MOV CL,0A		JMP HU
	MUL CL		TE: CMP AL,0A
	MOV BL,[SI]		JC UN
	AND BL,OF		SUB AL,0A
	ADD AL,BL		INC CH
	MOV DI,1300		JMP TE
	MOV [DI],AL		UN: MOV [SI],AL
	HLT		MOV AL,CH
			MOV CL,01
			ROL AL,CL
			ROL AL,CL
			ROL AL,CL
			ROL AL,CL
			ADD AL,[SI]
			MOV DI,1300
			MOV [DI],AL
			MOV DI,1301
			MOV [DI],BX
			HLT

HEXADECIMAL TO ASCII			ASCII TO HEXADECIMAL		
INPUT			**INPUT**		
OUTPUT			**OUTPUT**		

	Hexadecimal to ASCII		**ASCII to Hexadecimal**

	MOV SI,600
	MOV AL,[SI]
	MOV BL,AL
	AND AL,0F
	CALL L1
	MOV DI,700
	MOV [DI],AL
	MOV AL,BL
	AND AL,0F
	MOV CL,01
	ROL AL,CL
	ROL AL,CL
	ROL AL,CL
	ROL AL,CL
	CALL L1
	MOV DI,701
	MOV [DI],AL
	HLT
	L1: CMP AL,0A
	JC L2
	ADD AL,07
	L2: ADD AL,30
	RET
			MOV SI,1200
			MOV AL,[SI]
			CALL L1
			MOV DI,1300
			MOV [DI],AL
			HLT
			L1: SUB AL,30
			CMP AL,0A
			JC L2
			SUB AL,07
			L2: RET

RESULT:-

MATRIX ADDITION			
INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA

EXP. NO. 3b **MATRIX OPERATION**

AIM:

To write a program to perform matrix addition and multiplication.

ALGORITHM:

(i) Matrix addition

1. Initialize the index and pointer registers for data and result.
2. Load CL with count.
3. Add the two matrix by each element
4. Process continues until CL becomes zero.
5. Store the result.

(ii) Matrix subtraction

1. Initialize the index and pointer registers for data and result.
2. Load CL with count.
3. Subtract the two matrix by each element
4. Process continues until CL becomes zero.
5. Store the result.

ADDRESS	**MATRIX ADDITION**:
	MOV SI.1000
	MOV BP,1020
	MOV DI,1050
	MOV CL,04
	MOV AL,[SI]
	MOV BL,[BP+0]
	ADD AL,BL
	MOV [DI],AL
	ADD DI,01
	ADD SI,01
	ADD BP,01
	DEC CL
	JNZ L1
	HLT

MATRIX SUBTRACTION			
INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA

ADDRESS	MATRIX SUBTRACTION:

	MOV SI.1000
	MOV BP,1020
	MOV DI,1050
	MOV CL,04
	MOV AL,[SI]
	MOV BL,[BP+0]
	SUB AL,BL
	MOV [DI],AL
	ADD DI,01
	ADD SI,01
	ADD BP,01
	DEC CL
	JNZ L1
	HLT

RESULT:-

SCAN A STRING			COMPARE A STRING		
ADDRESS	**DATA**		**ADDRESS**	**DATA**	
		I/P STRING 1			**I/P STRING 1**
		I/P STRING 2			**I/P STRING 1**
		I/P STRING 3			**I/P STRING 2**
		I/P STRING 4			**I/P STRING 2**
		OUTPUT			**OUTPUT**

COPY A STRING					
ADDRESS	**DATA**		**ADDRESS**	**DATA**	
		I/P STRING 1			**O/P STRING 1**
		I/P STRING 2			**O/P STRING 2**
		I/P STRING 3			**O/P STRING 3**
		I/P STRING 4			**O/P STRING 4**

EXP. NO. 4a. **STRING MANIPULATIONS**

AIM:

To write a program to perform string manipulations.

ALGORITHM:

1. Initialize the source index and destination index registers.
2. Initialize the count register.
3. Repeat the string instructions to perform Scan/Copy/ Compare a string
4. Store the result of respective string manipulations.

PROGRAM:

ADDRESS	(i)SCAN A STRING	ADDRESS	(ii)COPY A STRING	ADDRESS	(iii)COMPARE A STRING
	MOV DI, 0500		MOV SI, 0500		MOV SI, 0500
	MOV AX, 0003		MOV DI, 0600		MOV DI,0600
	MOV CX, 0004		MOV CX,0004		MOV CX,0002
	CLD		CLD		CLD
	REPNE		REP		REP
	SCASW		MOVSW		CMPSW
	JNZ L1		HLT		JZ L1
	MOV BX, 0001				MOV AX, 0000
	JMP L2				JMP L2
	L1: MOV BX,0000				L1: MOV AX, 0001
	L2: MOV DI, 0600				L2: MOV DI, 0650
	MOV [DI],BX				MOV [DI], AX
	HLT				HLT

RESULT:-

SMALLEST NUMBER:-

INPUT			OUTPUT	
ADDRESS	**DATA**		**ADDRESS**	**DATA**
	03	**COUNT**		
	00			
		DATA 1		
		DATA 2		
		DATA 3		

LARGEST NUMBER:-

INPUT			OUTPUT	
ADDRESS	**DATA**		**ADDRESS**	**DATA**
	03	**COUNT**		
	00			
		DATA 1		
		DATA 2		
		DATA 3		

EXP. NO. 4b. **SEARCHING A NUMBER**

AIM:

To write a program to perform searching in an array.

ALGORITHM:

1. Load the memory address to index register.
2. Load the count value to count regirster
3. Increment the index register
4. Move the content to accumulator
5. Compare the data with the next data in index register.
6. Check the carry flag.[No carry for largest and Carry for smallest]
7. Increment the index register and decrement the count register.
8. Repeat steps 5 to 7 until count becomes zero.
9. Store the output in a memory location
10. Stop

ADDRESS	LARGEST OF N NUMBERS:	ADDRESS	SMALLEST OF N NUMBERS:
	MOV SI,600		MOV SI,600
	MOV CX,[SI]		MOV CX,[SI]
	ADD SI,02		ADD SI,02
	MOV AX,[SI]		MOV AX,[SI]
	L2:CMP AX,[SI+02]		L2:CMP AX,[SI+02]
	JNC L1		JC L1
	MOV AX,[SI+02]		MOV AX,[SI+02]
	L1:ADD SI,02		L1:ADD SI,02
	LOOP L2		LOOP L2
	MOV [400],AX		MOV [400],AX
	HLT		HLT

RESULT:

ASCENDING ORDER:-

INPUT			OUTPUT	
ADDRESS	**DATA**		**ADDRESS**	**DATA**
		DATA 1		
		DATA 2		
		DATA 3		

DESCENDING ORDER:-

INPUT			OUTPUT	
ADDRESS	**DATA**		**ADDRESS**	**DATA**
		DATA 1		
		DATA 2		
		DATA 3		

EXP. NO. 4C. **SORTING AN ARRAY**

AIM:

To write a program to perform sorting in an array.

ALGORITHM:

1. Load the memory address to index register.
2. Load the count value for outer loop to register
3. Load the count value for inner to register
4. Move the content to accumulator
5. Compare the data with the next data in index register.
6. Check the carry flag.[No carry for Ascending and Carry for Descending]
7. Increment the index register and decrement the inner loop .
8. Repeat steps 3 to6 until count becomes zero.
9. Once inner loop becomes zero, decrement the outer loop
10. Repeat steps2 to 8 until count becomes zero
11. Stop

ADDRESS	**ASCENDING ORDER**:	**ADDRESS**	**DESCENDING ORDER**:
	MOV DX,0003		MOV DX,0003
	DEC DX		DEC DX
	L3:MOV CX,DX		L3:MOV CX,DX
	MOV SI,300		MOV SI,300
	L2:MOV AX,[SI]		L2:MOV AX,[SI]
	CMP [SI+02],AX		CMP [SI+02],AX
	JNC L1		JC L1
	MOV BX,[SI+02]		MOV BX,[SI+02]
	MOV [SI+02],AX		MOV [SI+02],AX
	MOV [SI],BX		MOV [SI],BX
	L1:INC SI		L1:INC SI
	INC SI		INC SI
	LOOP L2		LOOP L2
	DEC DX		DEC DX
	JNZ L3		JNZ L3
	HLT		HLT

RESULT:

PASSWORD CHECKING

OUTPUT:

MASM

EXP. NO. 5a. **PASSWORD CHECKING**

AIM:

To write a MASM program to perform password checking.

PROGRAM:

```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
PASSWORD DB 'INDIA$'
PROMPT DB 0DH, 0AH, 'ENTER PASSWORD:', 0DH, 0AH, '$'
ENTRY DB 41 DUP(?)
MSGSUC DB 0DH, 0AH, 'AUTHORIZED PERSON', 0DH, 0AH, '$'
MSGFAIL DB 0DH, 0AH, 'WRONG PASSWORD TRY AGAIN', 0DH, 0AH, '$'
DATA ENDS
CODE SEGMENT
START: MOV AX, DATA
 MOV DS, AX
 MOV ES, AX
 LEA DX, PROMPT
 MOV AH, 09H
 INT 21H
 MOV BP,0
AGAIN: MOV AH, 08H
 INT 21H
 CMP AL, 0DH
 JZ ACTION
 MOV ENTRY[BX],AL
 INC BX
 LOOP AGAIN
ACTION: MOV ENTRY[BX], '$'
 LEA SI, PASSWORD
 LEA DI, ENTRY
 MOV CX, 06
 REPE CMPSB
 JE SUCMSG
 LEA DX, MSGFAIL
 MOV AH, 09H
 INT 21H
 JMP L1
SUCMSG: LEA DX, MSGSUC
 MOV AH, 09H
 INT 21H
L1: MOV AH, 4CH
 INT 21H
CODE ENDS
END START
```

RESULT:

SYSTEM DATE
OUTPUT

EXP. NO. 5b. **SYSTEM DATE**

AIM:

To write a MASM program to display the system date.

PROGRAM:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
MESS DB 0DH,0AH,'SYSTEM DATE IS:',0DH,0AH,'$'
DATA ENDS
CODE SEGMENT
START: MOV AX,DATA
       MOV DS,AX
       MOV ES,AX
       LEA DX,MESS
       MOV AH,09H
       INT 21H
       MOV AH,2AH
       INT 21H
       PUSH CX
       MOV CX,0
       MOV CL,DL
       PUSH CX
       MOV CL,DH
       PUSH CX
; DISPLAY MONTH
       MOV DX,0
       POP AX
       MOV CX,0
       MOV BX,10
L1:    DIV BX
       PUSH DX
       ADD CX,1
       MOV DX,0
       CMP AX,0
       JNE L1
L2:    POP DX
       ADD DL,48
       MOV AH,02H
       INT 21H
       LOOP L2
       MOV DL,'/'
       MOV AH,02H
       INT 21H
;DISPLAY DAY
       MOV DX,0
       POP AX
       MOV CX,0
       MOV BX,10
L3:    DIV BX
       PUSH DX
       ADD CX,1
       MOV DX,0
       CMP AX,0
       JNE L3
```

```
L4:    POP DX
       ADD DL,48
       MOV AH,02H
       INT 21H
       LOOP L4
       MOV DL,'/'
       MOV AH,02H
       INT 21H
;DISPLAY YEAR
       MOV DX,0
       POP AX
       MOV CX,0
       MOV BX,10
L5:    DIV BX
       PUSH DX
       ADD CX,1
       MOV DX,0
       CMP AX,0
       JNE L5
L6:    POP DX
       ADD DL,48
       MOV AH,02H
       INT 21H
       LOOP L6
       MOV AL,0
       MOV AH,4CH
       INT 21H
CODE ENDS
END START
```

RESULT:

PRINT RAM SIZE
OUTPUT:

EXP. NO. 5c. **PRINT RAM SIZE**

AIM:

To write a MASM program to display the RAM size of the processor.

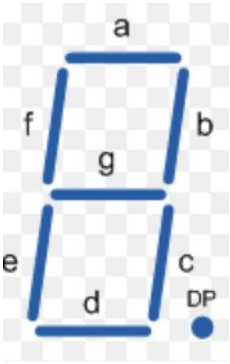
PROGRAM:

```
ASSUMECS:CODE,DS:DATA
DATA SEGMENT
MSGDB ‘MEMORY SIZE IN KILO BYTES =’
ASERESDB 4 DUP(?),0DB,0AB,’HEX’,0DB,0AB,’$’
RESDW?
HEXCODEDB ‘01234567890ABCDEF’
DATA ENDS
CODE SEGMENT
HEX_ASCPROC
        MOV DL,10
        MOV AH,0
        MOV BX,0
        DIV DL
        MOVBL,AL
        MOVDH,HEXCODE[BX]
        MOVBL,AH
        MOVDL,HEXCODE[BX]
        NET
HEX_ASCENDP
MAIN:
MOVAX,DATA
MOVDS,AX
INT 12
MOVRES,AX
MOVAL,BYTEPTR[RES]
CALLHEX_ASC
MOV ASCRES+2,DH
MOV ASCRES+3,DL
MOVAL,BYTEPTR[NEST1]
CALLHEX_ASC
MOVASCRES,DH
MOV ASCRES+1,DL
MOVDX,OFFSETMSG
MOV AH,09
INT 21
MOV AH,4C
INT 21
CODE ENDS
END MAIN
```

RESULT:

CYCLE II
INTERFACING WITH 8086

CONTROL WORD FORMAT TO DISPLAY **“CSE 2017”**



Letter	a	b	c	d	e	f	g	dp	Hex code

EXP. NO. 6 8279 – KEYBOARD AND DISPLAY CONTROLLER

AIM:

To write an assembly language program to display the given message in rolling fashion.

ALGORITHM:

1. Initialize a display and keyboard mode set.
2. Initialize read control word.
3. Initialize clear control word.
4. Initialize the count register
5. The hexadecimal values for the message to be displayed in rolling manner is found.
6. A look up table is made with those values.
7. The values are fetched and displayed with a delay until count register becomes zero.
8. Repeat step 7 for rolling display.

PROGRAM:

Address	Label	Mnemonics	Operand
		MOV	AL,18
		OUT	66,AL
		MOV	AL,DC
		OUT	66,AL
		MOV	AL,90
		OUT	66,AL
	L1	MOV	CX,0F
		MOV	SI,0600
	L2	MOV	AL,[SI]
		OUT	60,AL
		INC	SI
		CALL	DELAY
		LOOP	L2
		JMP	L1
	DELAY	MOV	DX,FFFF
	L3	DEC	DX
		JNZ	L3
		RET	

❍ *000DDMMM*

Mode set: Opcode 000.

DD sets displays mode.

MMM sets keyboard mode.

DD field selects either:

- 8- or 16-digit display
- Whether new data are entered to the rightmost or leftmost display position.

DD	*Function*
00	8-digit display with left entry
01	16-digit display with left entry
10	8-digit display with right entry
11	16-digit display with right entry

MMM field:

DD	*Function*
000	Encoded keyboard with 2-key lockout
001	Decoded keyboard with 2-key lockout
010	Encoded keyboard with N-key rollover
011	Decoded keyboard with N-key rollover
100	Encoded sensor matrix
101	Decoded sensor matrix
110	Strobed keyboard, encoded display scan
111	Strobed keyboard, decoded display scan

Z selects auto-increment for the address.

❍ *011ZAAAA*

The ***display read control word*** selects the read address of one of the display RAM positions for reading through the data port.

❍ *1100CCFA*

The ***clear control word*** clears the display, FIFO or both

Bit F clears FIFO and the display RAM status, and sets address pointer to 000.

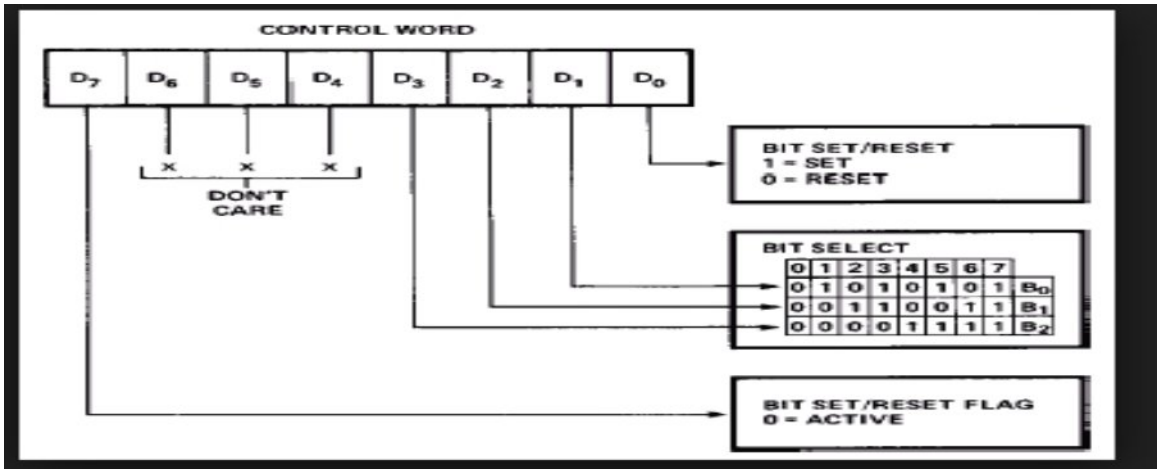
If CC are 00 or 01, all display RAM locations become 00000000.

If CC is 10, --> 00100000, if CC is 11, --> 11111111.

600	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**		
608		**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**

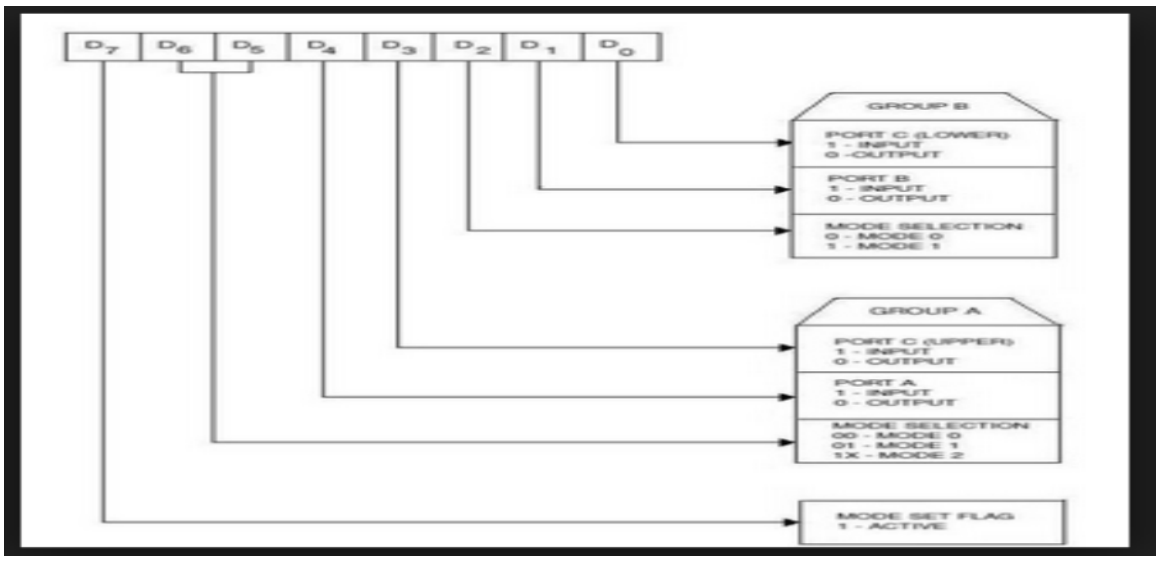
RESULT:

BSR MODE FORMAT:



OUTPUT:

I/O MODE FORMAT:



OUTPUT:

EXP. NO. 7a. **8255 – PARALLEL COMMUNICATION**

AIM:

To program the Programmable Peripheral Interface (8255) in I/O mode and BSR mode.

ALGORITHM:

1. Initialize the D7 bit for I/O mode and BSR mode.
2. Transmit an array of data from input port to output port for I/O mode.
3. Set/ Reset a particular bit of an output port for BSR mode.

PROGRAM:

I/O MODE:

Address	Label	Mnemonics	Operand
		MOV	SI,0600
		MOV	AL,80
		OUT	66,AL
		MOV	CX,0003
	L1:	MOV	AL,[SI]
		OUT	60,AL
		INC	SI
		DEC	CX
		JNZ	L1
		HLT	

BSR MODE:

Address	Label	Mnemonics	Operand
		MOV	AL,80
		OUT	66,AL
	L1:	MOV	AL,07
		OUT	66,AL
		MOV	AL,06
		OUT	66,AL
		JMP	L1

RESULT:

SERIAL COMMUNICATION:
OUTPUT:

EXP. NO. 7b. **8251 – SERIAL COMMUNICATION**

AIM:

To program the Programmable Communication Interface (8251) to transmit data serially.

PROGRAM:

```
MOV AL,4E
OUT 66,AL
MOV AL,05
OUT 66,AL
L1: IN AL,66
AND AL,02
JZ L1
IN AL,60
MOV DL,AL
NOP
L2: IN AL,66
AND AL,01
JZ L2
MOV AL,DL
OUT 60,AL
JMP L1
```

OUTPUT: After executing the 8086 program ,reset button is clicked twice after a key is pressed in the keyboard .The key pressed in the keyboard is displayed on the screen after resetting 5 times.

RESULT:

Address lines to select analog input:

Analog I/P selected	Address lines		
	C	B	A
I / P 0	0	0	0
I / P 1	0	0	1
I / P 2	0	1	0
I / P 3	0	1	1
I / P 4	1	0	0
I / P 5	1	0	1
I / P 6	1	1	0
I / P 7	1	1	1

OUTPUT:-

INPUT	**OUTPUT**
ANALOG VOLTAGE	**DIGITAL DATA ADDRESS:-0500**

EXP. NO. 8a. ANALOG TO DIGITAL CONVERTOR

AIM:

To write a program to convert an analog signal to digital signal using an ADC interface.

ALGORITHM:

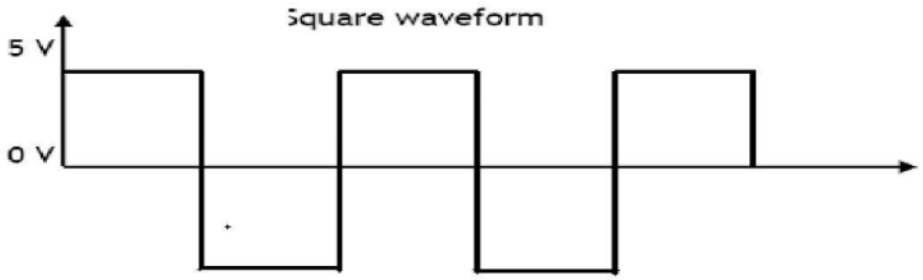
1. Initialize 8255 to set port A as input port.
2. Select an analog input.
3. Send the start conversion pulse.
4. Read EOC signal.
5. If EOC=1 continue else go to step 4.
6. Read the digital output.
7. Store it in a memory location.

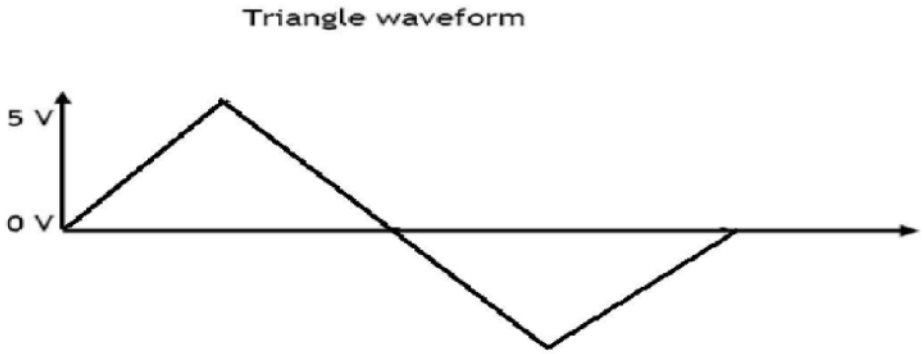
PROGRAM:

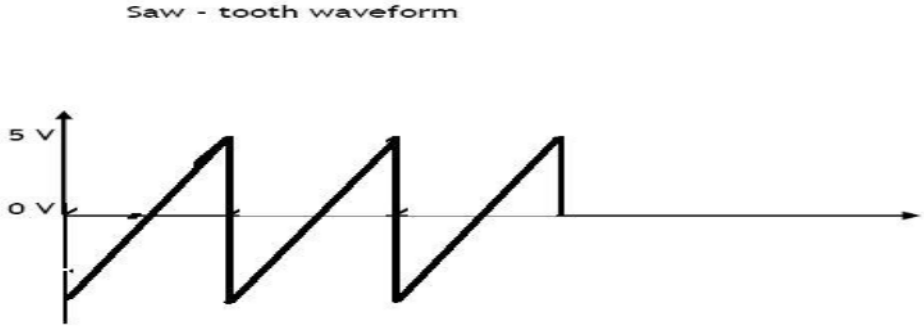
Address	Label	Mnemonics	Operand
		MOV	AL,98
		OUT	76,AL
		MOV	AL,00
		OUT	74,AL
		MOV	AL,01
		OUT	74,AL
		MOV	AL,03
		OUT	74,AL
		NOP	
		NOP	
		MOV	AL,01
		OUT	74,AL
	L1	IN	AL,74
		AND	AL,10
		JZ	L1
		IN	AL,70
		MOV	SI,0500
		MOV	[SI],AL
		HLT	

RESULT:-

MODEL GRAPH:


Square waveform
5 V
0 V


Triangle waveform
5 V
0 V


Saw - tooth waveform
5 V
0 V

EXP. NO. 8b. DIGITAL TO ANALOG CONVERTOR

AIM:

To generate different types of waveforms by interfacing a DAC card with microprocessor trainer kit.

ALGORITHM:

Measurement of analog voltage:

1. Initialize 8255.
2. Send the digital value of DAC.
3. Read the corresponding analog value of its output.

Waveform generation:

Square waveform:

1. Send a high value to DAC and introduce delay.
2. Send a low value to DAC and introduce delay.
3. Repeat the above procedure.

Sawtooth waveform:

1. Load low value to accumulator.
2. Send the value to DAC.
3. Increment the accumulator.
4. Repeat step (ii) and (iii) until accumulator value reaches FF.
5. Repeat the above procedure from step 1.

Triangular wave:

1. Load the low value in accumulator.
2. Send this to DAC
3. Increment the accumulator
4. Repeat step 2 and 3 until accumulator reaches FF,decrement the accumulator and send the content to DAC.
5. The above procedure is repeated from step (i).

PROGRAM:

SQUARE WAVE				**TRIANGULAR WAVE**			
Address	**Label**	**Mnemonics**	**Operand**	**Address**	**Label**	**Mnemonics**	**Operand**
		MOV	AL,80			MOV	AL,80
		OUT	76,AL			OUT	76,AL
	L1	MOV	AL,FF		L1	MOV	AL,00
		OUT	70,AL		L2	OUT	70,AL
		CALL	delay			INC	AL
		MOV	AL,00			JNZ	L2
		OUT	70,AL			MOV	AL,FF
		CALL	delay		L3	OUT	70,AL
		JMP	L1			DEC	AL
	delay	MOV	CL,FF			JNZ	L3
	L2	DEC	CL			JMP	L1
		JNZ	L2				
		RET					

OUTPUT:-

WAVEFORM	**AMPLITUDE in volts**	**TIME PERIOD in ms**
SAWTOOTH WAVE		
TRIANGULAR WAVE		
SQUARE WAVE		

SAWTOOTH WAVE			
Address	**Label**	**Mnemonics**	**Operand**
		MOV	AL,80
		OUT	76,AL
		MOV	AL,00
	L1	OUT	70,AL
		INC	AL
		JMP	L1

RESULT:

CYCLE III
8051
PROGRAMS

ADDITION				ADDITION			
	ADDRESS	DATA			ADDRESS	DATA	
INPUT			INPUT 1	INPUT			INPUT1
			INPUT 2				INPUT2
OUTPUT			SUM	OUTPUT			SUM
			CARRY				CARRY

MANUAL CALCULATION:-

SUBTRACTION				SUBTRACTION			
	ADDRESS	DATA			ADDRESS	DATA	
INPUT			INPUT 1	INPUT			INPUT1
			INPUT 2				INPUT2
OUTPUT			DIFFRENCE	OUTPUT			DIFFERENCE
			BORROW				BORROW

MANUAL CALCULATION:-

EXP. NO. 9 ARITHMETIC AND LOGICAL OPERATIONS

AIM:

To write a program to perform arithmetic and logical operations.

ALGORITHM:

1. Load DPTR with input address.
2. Load first data to accumulator
3. Increment DPTR
4. Load second data to another register.
5. Perform arithmetic and logical operation.
6. Store the result to a memory location

PROGRAMS:

ARITHMETIC OPERATIONS

a.ADDITION				**b.SUBTRACTION**			
Address	**Label**	**Mnemonics**	**Operand**	**Address**	**Label**	**Mnemonics**	**Operand**
		MOV	R0,#00			MOV	R0,#00
		MOV	DPTR, #3200			MOV	DPTR, #3200
		MOVX	A, @DPTR			MOVX	A, @DPTR
		INC	DPTR			INC	DPTR
		MOV	B, A			MOV	B, A
		MOVX	A, @DPTR			MOVX	A, @DPTR
		ADD	A, B			SUB	A, B
		JNC	L1			JNC	L1
		INC	R0			INC	R0
	L1	INC	DPTR			INC	DPTR
		MOVX	@DPTR, A			MOVX	@DPTR, A
		INC	DPTR			INC	DPTR
		MOV	A,R0			MOV	A,R0
		MOVX	@DPTR,A			MOVX	@DPTR,A
	LOOP	SJMP	LOOP		LOOP	SJMP	LOOP

c.MULTIPLICATION				**d.DIVISION**			
Address	**Label**	**Mnemonics**	**Operand**	**Address**	**Label**	**Mnemonics**	**Operand**
		MOV	DPTR,#3200			MOV	DPTR,#3200
		MOVX	A,@DPTR			MOVX	A,@DPTR
		INC	DPTR			INC	DPTR
		MOV	B,A			MOV	B,A
		MOVX	A,@DPTR			MOVX	A,@DPTR
		MUL	AB			DIV	AB
		INC	DPTR			INC	DPTR
		MOVX	@DPTR,A			MOVX	@DPTR,A
		INC	DPTR			INC	DPTR
		MOV	A,B			MOV	A,B
		MOVX	@DPTR,A			MOVX	@DPTR,A
	LOOP	SJMP	LOOP		LOOP	SJMP	LOOP

MULTIPLICATION				DIVISION			
	ADDRESS	DATA			ADDRESS	DATA	
INPUT			INPUT 1 (8-BIT)	INPUT			INPUT1
			INPUT 2 (8-BIT)				INPUT2
OUTPUT			PRODUCT (LSB 16 BIT)	OUTPUT			QUOTIENT
							REMINDER

OR			AND		
	ADDRESS	DATA		ADDRESS	DATA
OUTPUT			OUTPUT		
XOR			NOT		
OUTPUT			OUTPUT		

MANUAL CALCULATION:-

LOGICAL OPERATION

e.AND				**f.OR**			
Address	**Label**	**Mnemonics**	**Operand**	**Address**	**Label**	**Mnemonics**	**Operand**
		MOV	A, #02			MOV	A, #02
		MOV	B, #05			MOV	B, #05
		ANL	A, B			ORL	A, B
		MOV	DPTR, #3200			MOV	DPTR, #3200
		MOVX	@DPTR, A			MOVX	@DPTR, A
	LOOP	SJMP	LOOP		LOOP	SJMP	LOOP

g.EXOR				**h.NOT**			
Address	**Label**	**Mnemonics**	**Operand**	**Address**	**Label**	**Mnemonics**	**Operand**
		MOV	A, #02			MOV	A, #02
		MOV	B, #05			CPL	A
		XRL	A, B			MOV	DPTR, #3200
		MOV	DPTR, #3200			MOVX	@DPTR, A
		MOVX	@DPTR, A		LOOP	SJMP	LOOP
	LOOP	SJMP	LOOP				

RESULT:-

SQUARE OF A NUMBER				**CUBE OF A NUMBER**			
	ADDRESS	**DATA**			**ADDRESS**	**DATA**	
INPUT			**INPUT**	**INPUT**			**INPUT**
OUTPUT			**SQUARE**	**OUTPUT**			**CUBE**

2'S COMPLEMENT			
	ADDRESS	**DATA**	
INPUT			**INPUT 1 (8-BIT)**
OUTPUT			**2'S**

MANUAL CALCULATION:-

RESULT:-

EXP. NO.10 SQUARE, CUBE, 2'S COMPLEMENT OF A NUMBER

AIM:

To write a program to find square, cube and 2's complement of a number.

ALGORITHM:

SQUARE AND CUBE OF A NUMBER:

1. Load DPTR with input address
2. Load the first data to accumulator and move to another register
3. Load the second data to accumulator
4. Multiply twice for square and thrice for cube of a number
5. Store the result to a memory location.

2'S COMPLEMENT OF A NUMBER:

1. Load DPTR with input address
2. Load the data to accumulator and Complement the accumulator
3. Add 01H to the complemented content and Store the result to a memory location

PROGRAMS:

a. SQUARE PROGRAM:

Address	Label	Mnemonics	Operand
		MOV	DPTR,#3200
		MOVX	A,@DPTR
		MOV	B,A
		MUL	AB
		INC	DPTR
		MOVX	@DPTR,A
	LOOP	SJMP	LOOP

b. FIND 2's COMPLEMENT OF A NUMBER:

Address	Label	Mnemonics	Operand
		MOV	DPTR,#3200
		MOVX	A,@DPTR
		CPL	A
		ADD	A,#01
		INC	DPTR
		MOVX	@DPTR,A
	LOOP	SJMP	LOOP

c. CUBE PROGRAM:

Address	Label	Mnemonics	Operand
		MOV	DPTR,#3200
		MOVX	A,@DPTR
		MOV	B,A
		MOV	R0,A
		MUL	AB
		MOV	B,R0
		MUL	AB
		INC	DPTR
		MOVX	@DPTR,A
	LOOP	SJMP	LOOP

	ADDRESS	**DATA**
INPUT		
OUTPUT		

EXP. NO.11 UNPACKED BCD TO ASCII

AIM:

To write a program to convert unpacked BCD to ASCII.

ALGORITHM:

1. Initialize DPTR with input address.
2. Load the data to the accumulator
3. Mask MSB nibbles, perform OR operation with 30H and store the result.
4. Mask LSB nibbles, rotate, perform OR operation with 30H
5. Store the result

PROGRAM:

Address	Label	Mnemonics	Operand
		MOV	DPTR,#3200
		MOVX	A,@DPTR
		MOV	B,A
		MOV	A,#30
		ORL	A,B
		INC	DPTR
		MOVX	@DPTR,A
	LOOP	SJMP	LOOP

RESULT:-

8255 PORT ADDRESS

Port A-FF00H
Port B-FF01H
Port C-FF02H
Control word-FF03H

OUTPUT:-

EXP. NO. 12 **STEPPER MOTOR**

AIM:

To write a program to rotate a stepper motor in counter clockwise direction.

ALGORITHM:

1. Initialize 8255
2. Load the data to accumulator
3. Drive the steper motor circuitry and introduce a delay
4. Repeat step 2 & 3 for different datas

PROGRAM:

Address	Label	Mnemonics	Operand	Address	Label	Mnemonics	Operand
		MOV	DPTR, #0FF03			NOP	
		MOV	A,#80			NOP	
		MOVX	@DPTR,A			DJNZ	R6,L2
	start	MOV	DPTR, #0FF00			DJNZ	R7,L1
		MOV	A,#0FA			RET	
		MOVX	@DPTR,A				
		ACALL	delay				
		MOV	A,#0F6				
		MOVX	@DPTR,A				
		ACALL	delay				
		MOV	A,#0F5				
		MOVX	@DPTR,A				
		ACALL	delay				
		MOV	A,#0F9				
		MOVX	@DPTR,A				
		ACALL	delay				
		SJMP	start				
	delay	MOV	R7,#03F				
	L1	MOV	R6,#03F				
	L2	NOP					

RESULT:

OUTPUT:-

EXP. NO. 13 **TRAFFIC LIGHT CONTROLLER**

AIM:

To write a program to control traffic light using 8051.

ALGORITHM:

1. Initialize 8255
2. Assign four signals to two different ports of 8051
3. Load the respective values for which the signals should glow in all the four directions and introduce a delay

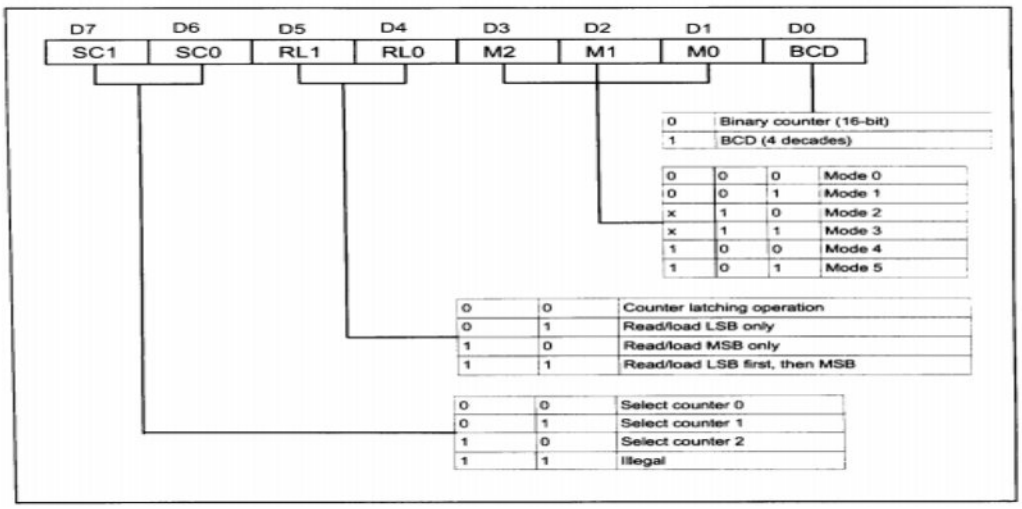
PROGRAM:

Address	Label	Mnemonics	Operand	Address	Label	Mnemonics	Operand
		MOV	DPTR, #0FF03			LCAL	L1
		MOV	A, #80H			MOV	A, #22H
		MOVX	@DPTR, A			MOVX	@DPTR, A
		MOV	A, #11H			LCAL	L2
		MOV	DPTR,#0FF00			MOV	A, #99H
		MOVX	@DPTR, A			MOVX	@DPTR, A
		MOV	DPTR, #0FF02			LCAL	L1
		MOVX	@DPTR, A			MOV	A, #22H
		LCAL	L1			MOVX	@DPTR, A
	L3	MOV	A, #44H			LCAL	L2
		MOV	DPTR, #0FF00			MOV	A, #11H
		MOVX	@DPTR, A			MOVX	@DPTR, A
		LCAL	L1			LJMP	L3
		MOV	A, #22H		L1	MOV	R1, #25H
		MOVX	@DPTR, A		L6	MOV	R2, #00H
		LCAL	L2		L5	MOV	R3, #00H
		MOV	A, #99H		L4	DJNZ	R3, L4
		MOVX	@DPTR, A			DJNZ	R2, L5
		LCAL	L1			DJNZ	R1, L6
		MOV	A, #22H			RET	
		MOVX	@DPTR, A		L2	MOV	R1, #05H
		LCAL	L2		L9	MOV	R2, #00H
		MOV	A, #11H		L8	MOV	R3, #00H
		MOVX	@DPTR, A		L7	DJNZ	R3, L7
		MOV	A, #44H			DJNZ	R2, L8
		MOV	DPTR, #0FF02			DJNZ	R1, L9
		MOVX	@DPTR, A			RET	

RESULT:

CONTENT BEYOND SYLLABUS

8253/54 Control Word



Amplitude(v)	Time(s)	Frequency(Hz)

EXP. NO.14 **8253 – TIMER**

AIM:

To write a program to generate a square wave.

ALGORITHM:

1. Initialize the control word of 8253 to square wave mode.
2. Load the MSB and LSB for the squre wave
3. Repeat step 1 & 2.

PROGRAM:

Address	Label	Mnemonics	Operand
	L1:	MOV	AL,36
		OUT	66,AL
		MOV	AL,0FF
		OUT	60,AL
		MOV	AL,00
		OOUT	60,AL
		JMP	L1

RESULT:

Output:

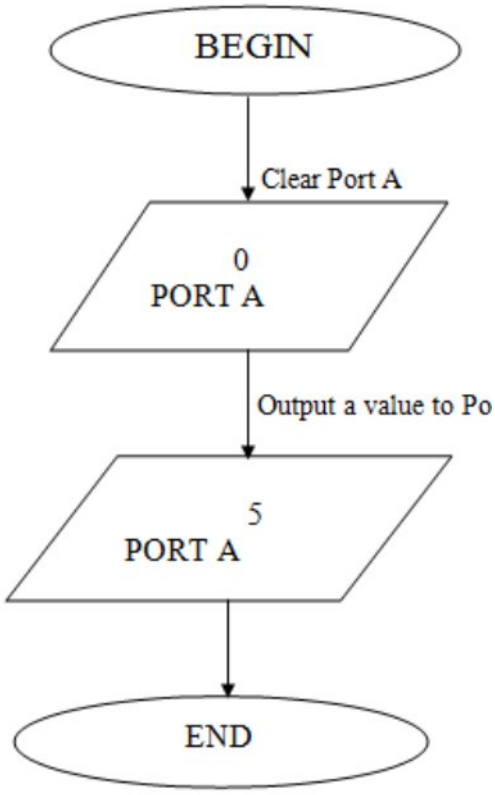
EXP. NO.15 PROGRAMMING PIC MICROCONTROLLER – FLOWCODE

AIM:

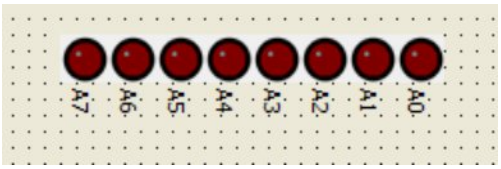
To generate a flowcode for the following a. Print a number on LED b. Display binary value from 00H to FFH.

FLOWCODE:

a. To print a number on LED



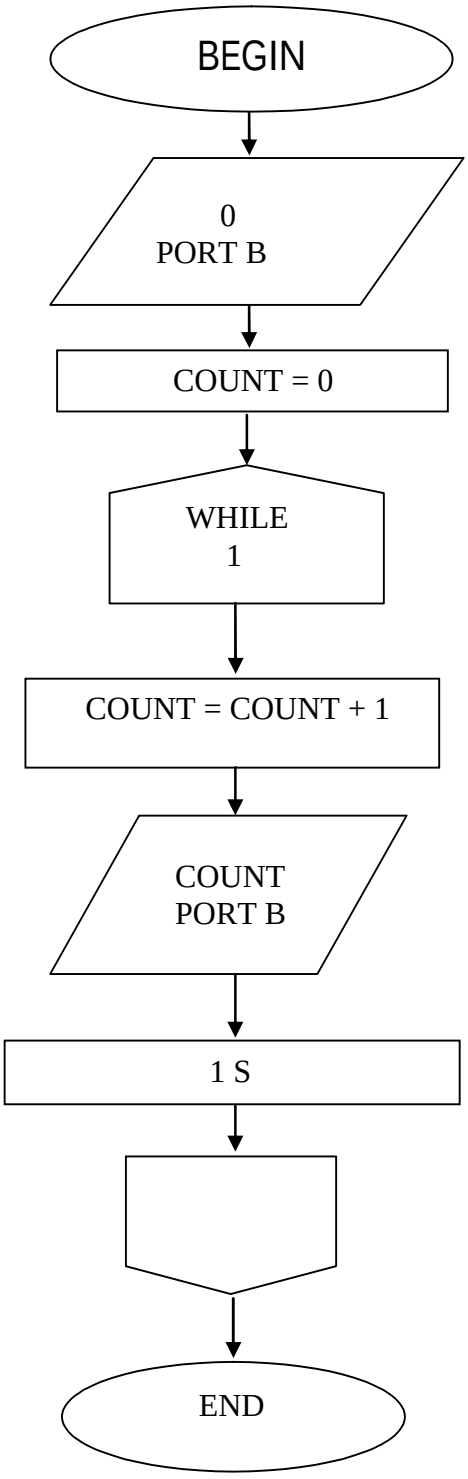
Output:



TRUTH TABLE:

B3	B2	B1	B0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

b. To display a binary value from 0 to FF



RESULT: