CHERAN COLLEGE OF ENGINEERING

Cheran Nagar, K.Paramthi, Karur – 639 111.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

LAB MANUAL

EC8681 MICROPROCESSOR AND MICROCONTROLLER LAB

III ECE/CSE – VI/V SEM

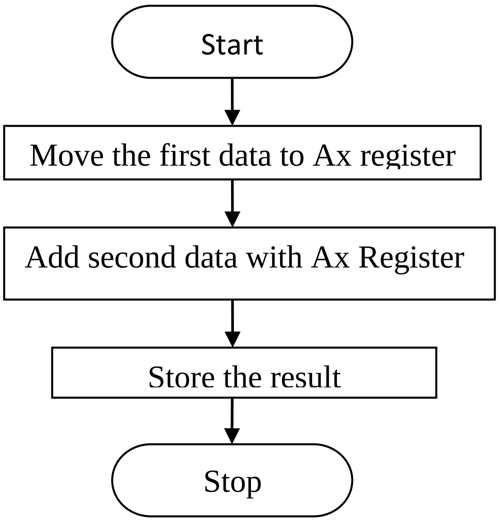
INDEX

INDEX

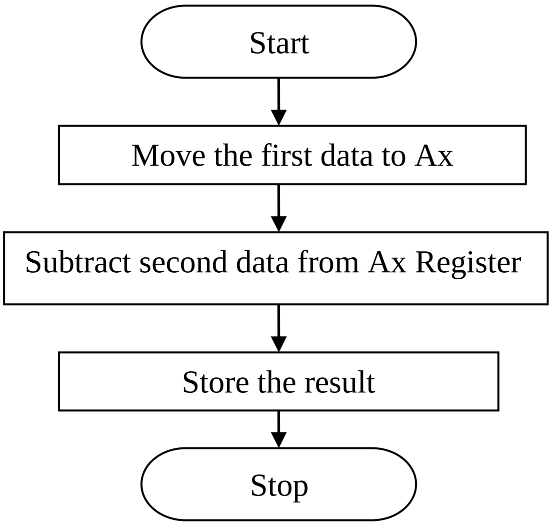
S. No	Date	Name of the Experiments	Page No	Marks	Signature of Staff
1.					
2.					
3.					
4.					
5.					
6.					
7.					
8.					
9.					
10.					
11.					
12.					
13.					
14.					
15.					
16.					
17.					
18.					
19.					
20.					

FLOWCHART:

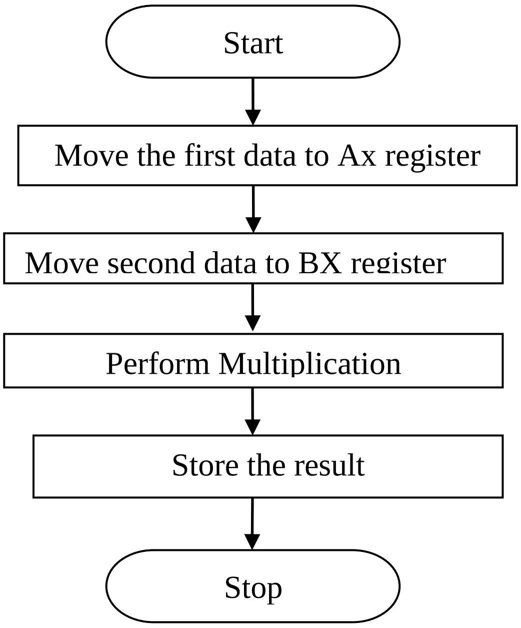
ADDITION


Start
Move the first data to Ax register
Add second data with Ax Register
Store the result
Stop

SUBTRACTION


Start
Move the first data to Ax
Subtract second data from Ax Register
Store the result
Stop

MULTIPLICATION


Start
Move the first data to Ax register
Move second data to BX register
Perform Multiplication
Store the result
Stop

Ex. No 1 ARITHMETIC AND LOGICAL OPERATIONS USING 8086

AIM:

To write an assembly language program to perform the arithmetic and logical operations using 8086 Microprocessor kit.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Power supply	1

ALGORITHM:

ADDITION

Step 1: Start the process
Step 2: Move the 16 bit data to Ax register.
Step 3: Perform the addition between accumulator content and second data
Step 4: Store the result.
Step 5: Stop the process

SUBTRACTION

Step 1: Start the process
Step 2: Move the 16 bit data to Ax register.
Step 3: Perform the subtraction between accumulator content and second data
Step 4: Store the result.
Step 5: Stop the process

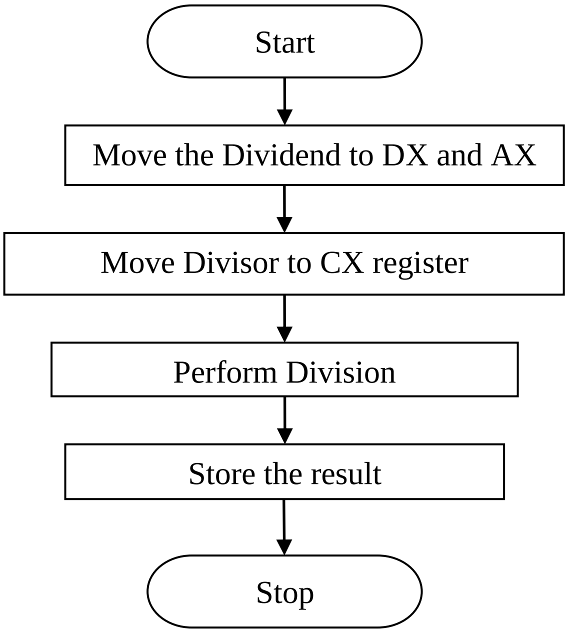
MULTIPLICATION

Step 1: Start the process
Step 2: Move the 16 bit data to Ax register.
Step 3: Move the second 16 bit data to Bx register.
Step 4: Perform the Multiplication
Step 5: Store the result.
Step 6: Stop the process

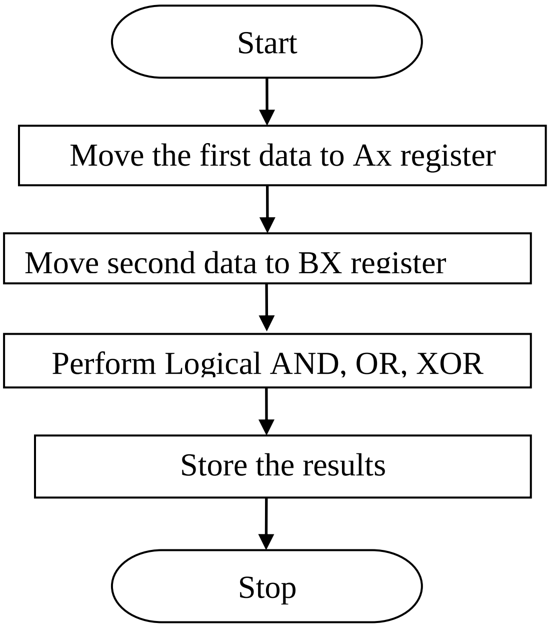
DIVISION

Step 1: Start the process
Step 2: Move the dividend to DX and AX register.
Step 3: Move the divisor to CX register.
Step 4: Perform the Division
Step 5: Store the result.
Step 6: Stop the process

DIVISION



LOGICAL OPERATION


Start
Move the first data to Ax register
Move second data to BX register
Perform Logical AND, OR, XOR
Store the results
Stop

PROGRAM:
ADDITION

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
1000		MOV AX,[1100]	8B 06 00 11	Move the content to Ax reg
1004		ADD AX, [1102]	03 06 02 11	Add second data
1008		MOV [1200],AX	89 06 00 12	Store the result
100C		HLT	F4	Stop the program

SUBTRACTION

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
1000		MOV AX,[1100]	8B 06 00 11	Move the content to Ax reg
1004		SUB AX, [1102]	2B 06 02 11	Subtract second data
1008		MOV [1200],AX	89 06 00 12	Store the result
100C		HLT	F4	Stop the program

MULTIPLICATION

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
1000		MOV AX,[1100]	8B 06 00 11	Move the content to AX reg
1004		MOV BX, [1102]	8B IE 02 11	Move the content to BX reg
1008		MUL BX	F7 E3	Perform Multiplication
100A		MOV [1200],DX	89 16 00 12	Store the MSW result
100E		MOV [1202],AX	89 06 02 12	Store the LSW
1012		HLT	F4	Stop the program

DIVISION

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
1000		MOV DX,[1100]	8B 16 00 11	Move the MSW of dividend in to DX reg
1004		MOV AX, [1102]	8B 06 02 11	Move the LSW of dividend in to AX reg
1008		MOV CX, [1102]	8B 0E 04 11	Move the divisor to CX reg

100C		DIV CX	F7 11	Perform Division
100E		MOV [1200],AX	89 06 00 12	Store the quotient
1012		MOV [1202],DX	89 16 02 12	Store the reminder
1016		HLT	F4	Stop the program

SAMPLE INPUT AND OUTPUT:

ADDITION

BEFORE EXECUTION:

1100	1234
1102	5678

AFTER EXECUTION:

1200	68AC

SUBTRACTION

BEFORE EXECUTION:

1100	5678
1102	1234

AFTER EXECUTION:

1200	4444

MULTIPLICATION

BEFORE EXECUTION:

1100	1234
1102	1234

AFTER EXECUTION:

1200	014B
1202	5A90

DIVISION

BEFORE EXECUTION:

1100	ABCD
1102	1234

AFTER EXECUTION:

1200	014B
1202	5A90

LOGICAL OPERATION

BEFORE EXECUTION:

1100	5678
1102	ABCD

AFTER EXECUTION:

1200	0248
1202	FFFD
1204	FDB5

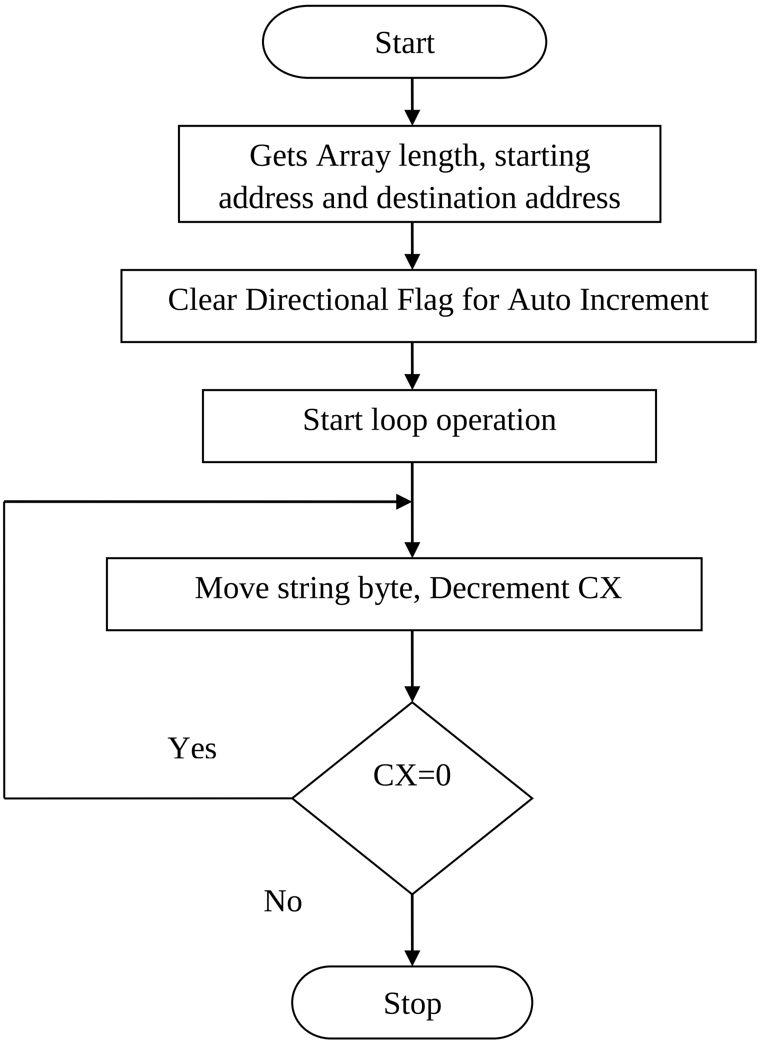
LOGICAL OPERATION

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
1000		MOV AX,[1100]	8B 06 00 11	Move the content to AX reg
1004		MOV BX, [1102]	8B IE 02 11	Move the content to BX reg
1008		AND AX,BX	21 D8	Perform Logical AND
100A		MOV [1200],AX	89 06 00 12	Store the result
100E		MOV AX,[1100]	8B 06 00 11	Move the content to AX reg
1012		OR AX,BX	09 D8	Perform Logical OR
1014		MOV [1202],AX	89 06 00 12	Store the result
1018		MOV AX,[1100]	8B 06 00 11	Move the content to AX reg
101C		XOR AX,BX	31 D8	Perform Logical XOR
101E		MOV [1204],AX	89 06 00 12	Store the result
1022		HLT	F4	Stop the program

RESULT

Thus the assembly language program to perform the arithmetic and logical operations was executed successfully and the result was verified by using 8086 Microprocessor kit.

FLOWCHART


Start
Gets Array length, starting address and destination address
Clear Directional Flag for Auto Increment
Start loop operation
Move string byte, Decrement CX
Yes
CX=0
No
Stop

SAMPLE INPUT AND OUTPUT:

BEFORE EXECUTION: **AFTER EXECUTION:**

2000	**00**
2001	**0E**
2002	**11**
2003	**1D**
2004	**14**
2005	**3C**

2000	**00**
2001	**0E**
2002	**11**
2003	**1D**
2004	**14**
2005	**3C**

Ex. No 2 MOVE A DATA BLOCK WITHOUT OVERLAP

AIM:

To write an assembly language program to move a data block without overlap using 8086 Microprocessor kit.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Power supply	1

ALGORITHM:

Step 1: Start the process
Step 2: Load the count value.
Step 3: Load the starting address of source and destination.
Step 4: Load the data from source and store to destination.
Step 5: increment the source and destination address
Step 6: Decrement the counter value
Step 7: Repeat the e Step 4, 5 and 6 until the counter value reaches zero
Step 8: Stop the process

PROGRAM:

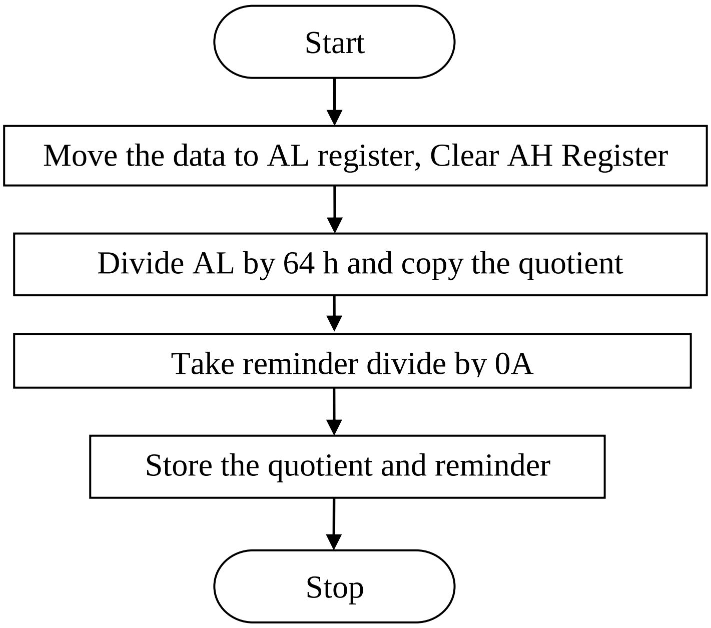
ADDRESS	LABLE	MNEMONICS	OP CODE	COMMENTS
1000		MOV CL,05H	C6 C1 05	Set counter the value
1002		MOV SI,2000H	C7 C6 00 20	Initialize the source value
1006		MOV DI,3000H	C7 C7 00 30	Initialize the destination value
100A	AA	MOV AL,[SI]	8A 04	Move the SI value to Al
100C		MOV [DI],AL	88 05	Move the Al value to DI
100E		ADD SI,0001H	81 C6 01 00	Increment the SI by 1
1002		ADD DI,0001H	81 C7 01 00	Increment the DI by 1
1006		LOOP AA	E2 F2	Check the CX value is 0000H
1008		HLT	F4	Stop the program

RESULT:

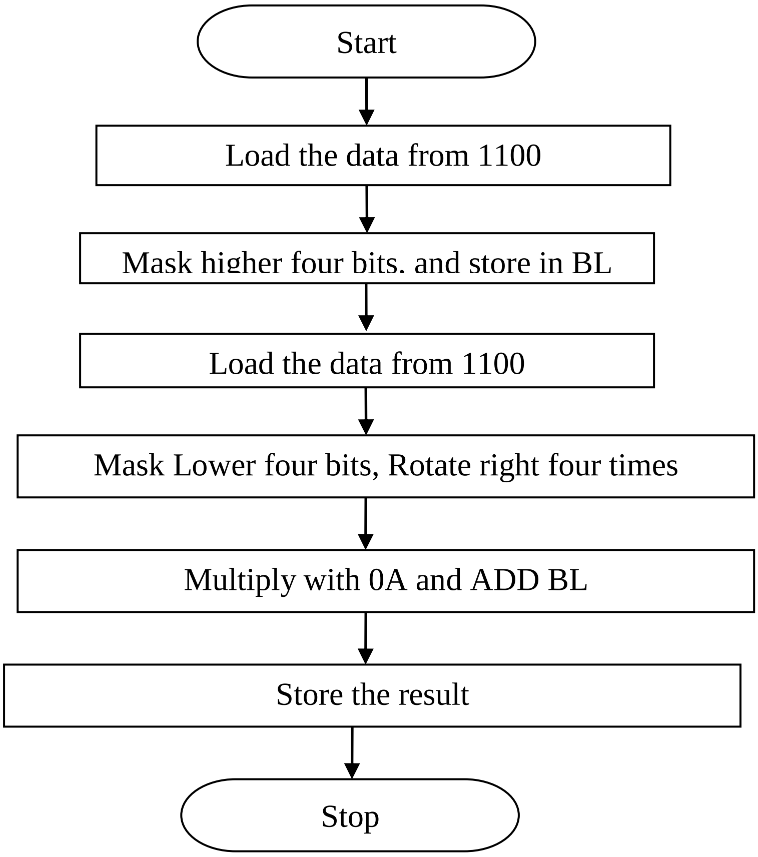
Thus the assembly language program to move a data block without overlap was executed successfully and the result was verified by using 8086 Microprocessor.

FLOW CHART

BI NARY TO DECIMAL


Start
Move the data to AL register, Clear AH Register
Divide AL by 64 h and copy the quotient
Take reminder divide by 0A
Store the quotient and reminder
Stop

DECIMAL TO BI NARY


Start
Load the data from 1100
Mask higher four bits, and store in BL
Load the data from 1100
Mask Lower four bits, Rotate right four times
Multiply with 0A and ADD BL
Store the result
Stop

Ex. No 3A

BI NARY TO DECIMAL AND DECIMAL TO BI NARY CODE CONVERSION

AIM:

To write an assembly language program to convert binary number into decimal and decimal number into binary using 8086 Microprocessor kit.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Power supply	1

ALGORITHM:

BI NARY TO DECIMAL

Step 1: Start the process
Step 2: Load the data from 1100.
Step 3: Clear the AH register.
Step 4: Move the content 64h to CL
Step 5: Divide AL/CL
Step 6: Store the quotient as hundreds.
Step 7: Move reminder to AL
Step 8: Move the content 0Ah to CL
Step 9: Divide AL/CL
Step 10: Store the quotient as tens.
Step 11: Store the quotient as ones.
Step 12: Stop the process

DECIMAL TO BI NARY

Step 1: Start the process
Step 2: Load the data from 1100.
Step 3: Perform AL and with 0F.
Step 4: Move the result to BL
Step 5: Load the data from 1100.
Step 6: Perform AL and with F0
Step7: Rotate the result four time right.
Step8: Multiply the result with 0A
Step9: Add BL with result.
Step 10: Store the result
Step 11: Stop the process

SAMPLE INPUT AND OUTPUT:

BI NARY TO DECIMAL

INPUT	
1100	**FF**

OUTPUT	
1101	**02**
1102	**05**
1103	**05**

DECIMAL TO BI NARY

INPUT	
1100	**25**

OUTPUT	
1101	**19**

PROGRAM:

BI NARY TO DECIMAL

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
1000		MOV AL,[1100]	8A 06 00 11	Load the data from 1100
1004		MOV AH,00	C6 C4 00	Clear AH Register
1007		MOV CL, 64	C6 C1 64	Move the 64h to CL register
100A		DIV CL	F6 F1	Divide the number
100C		MOV [1101],AL	88 06 01 11	Store the no of hundreds
1010		MOV AL, AH	88 E0	Move AH to AL
1012		MOV AH,00	C6 C4 00	Clear AH Register
1015		MOV CL,0A	C6 C1 0A	Move the 0Ah to CL register
1018		DIV CL	F6 F1	Divide the number
101A		MOV [1102],AX	89 06 02 11	Store the no of tens and ones
101E		HLT	F4	Stop the program

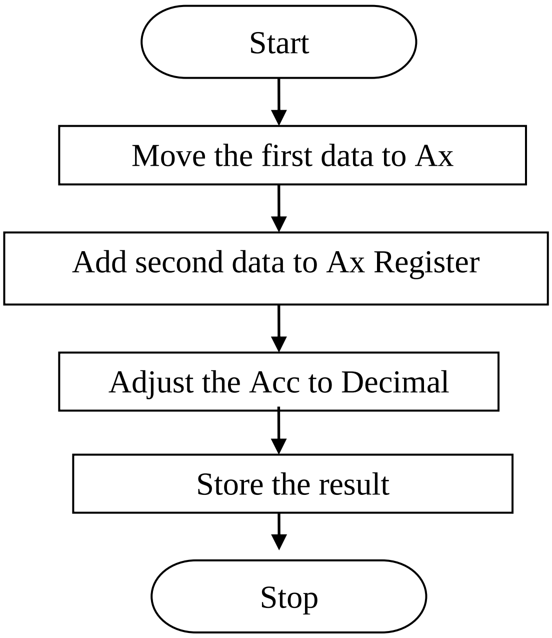
DECIMAL TO BI NARY

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
1000		MOV AL,[1100]	8A 06 00 11	Load the data from 1100
1004		AND AL,0F	80 E0 0F	Perform AL and 0F
1007		MOV BL, AL	88 C3	Move the AL to BL
1009		MOV AL,[1100]	8A 06 00 11	Load the data
100D		AND AL,0F0	80 E0 F0	Perform AL and F0
1010		MOV CL,04	C6 C1 04	Move CL to 04
1013		ROR AL,CL	D2 C8	Rotate AL four times
1015		MOV CL,0A	C6 C1 0A	Move CL to 0A
1018		MUL CL	F6 E1	Multiply 0Ah with CL register
101A		ADD AL,BL	00 D8	Add AL with BL
101C		MOV [1101],AL	88 06 01 11	Store the result
1020		HLT	F4	Stop the program

RESULT:

Thus the assembly language program to convert binary number into decimal and decimal number into binary was executed successfully and the result was verified by using 8086 Microprocessor.

FLOWCHART


Start
Move the first data to Ax
Add second data to Ax Register
Adjust the Acc to Decimal
Store the result
Stop

SAMPLE INPUT AND OUTPUT:

INPUT	
1100	**25**
1101	**27**

OUTPUT	
1200	**52**

Ex. No 3B DECIMAL ADDITION

AIM:

To write an assembly language program to add to decimal numbers using 8086 Microprocessor kit.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Power supply	1

ALGORITHM:

Step 1: Start the process
Step 2: Set the first data.
Step 3: Add second data to AL
Step 4: Decimal Adjust after addition
Step 5: Store the result.
Step 6: Stop the process

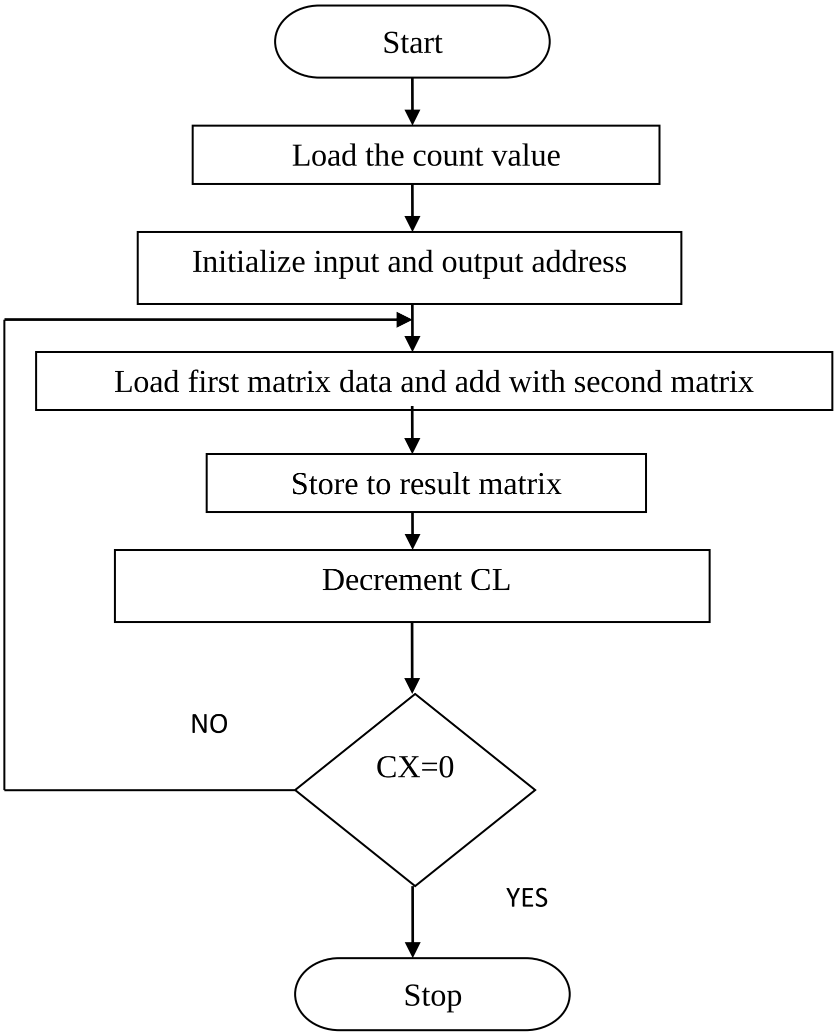
PROGRAM

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
1000		MOV AL,[1100]	8A 06 00 11	Move the content to AL reg
1004		ADD AL, [1101]	02 06 01 11	Add second data
1008		DAA	27	Decimal adjustment Accumulator
1009		MOV [1200],AL	88 06 00 12	Store the result
100D		HLT	F4	Stop the program

RESULT:

Thus the assembly language program to add to decimal numbers was executed successfully and the result was verified by using 8086 Microprocessor.

FLOWCHART


Start
Load the count value
Initialize input and output address
Load first matrix data and add with second matrix
Store to result matrix
Decrement CL
NO
CX=0
YES
Stop

SAMPLE INPUT AND OUTPUT:

INPUT MATRIX 1		INPUT MATRIX 2	
1100	25	1200	05
1101	25	1201	06
1102	25	1202	07
1103	25	1203	08
1104	25	1204	09
1105	25	1205	0A
1106	25	1206	0B
1107	25	1207	0C
1108	25	1208	0D

OUTPUT MATRIX	
1300	2A
1301	2B
1302	2C
1303	2D
1304	2E
1305	2F
1306	30
1307	31
1308	32

Ex. No 3C MATRIX ADDITION

AIM:

To write an assembly language program to add two 3 X 3 matrix using 8086 Microprocessor kit.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Power supply	1

ALGORITHM:

Step 1: Start the process
Step 2: Load the count value.
Step 3: Initialize the first matrix
Step 4: Initialize the second matrix
Step 5: Initialize the result matrix
Step 6: Get the first matrix data
Step 7: Add with second matrix data
Step 8: Store to result matrix
Step 9: Decrement count till count reach zero Repeat step 6 to step 8.
Step 6: Stop the process

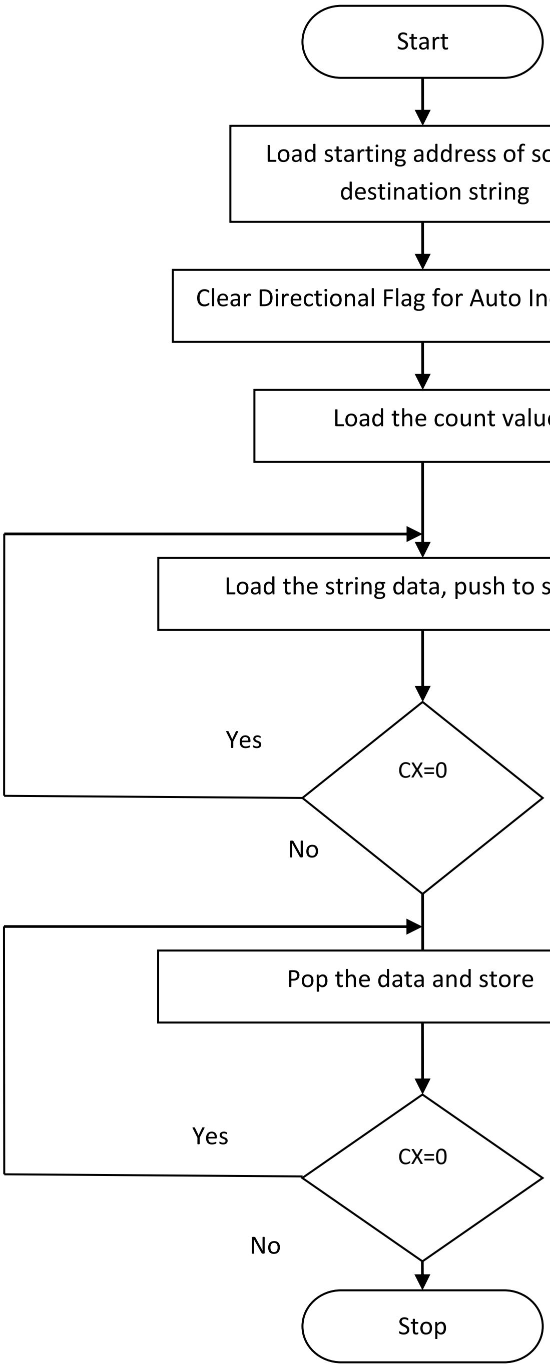
PROGRAM

ADD	OPCODE	LABLE	MNEMONICS	COMMENTS
1000	C6 C1 09		MOV CL,09	Move size of matrix to AL register
1003	C7 C6 00 11		MOV SI,1100	Move starting address of 1st Matrix
1007	C7 C7 00 12		MOV DI, 1200	Move starting address of 2nd Matrix
100B	C7 C5 00 13		MOV BP,1300	Move starting address of result Matrix
100F	8A 04	XX	MOV AL,[SI]	Get the first matrix data
1011	02 05		ADD AL,[DI]	Add the second matrix data
1013	88 46 00		MOV [BP],AL	Store the matrix result
1016	46		INC SI	Increment SI
1017	47		INC DI	Increment DI
1018	45		INC BP	Increment BP
1019	E2 F4		LOOP XX	If CX ≠ 0; Jump to XX
101B	F4		HLT	Stop the program

RESULT:

Thus the assembly language program to add two 3 X 3 matrix was executed successfully and the result was verified by using 8086 Microprocessor.

FLOWCHART


Start
Load starting address of source destination string
Clear Directional Flag for Auto Increment
Load the count value
Load the string data, push to stack
Yes
CX=0
No
Pop the data and store
Yes
CX=0
No
Stop

SAMPLE INPUT AND OUTPUT:

1600	**2222**
1602	**1234**
1604	**5555**
1606	**2525**
1608	**4545**
160A	**6363**

1700	**6363**
1702	**4545**
1704	**2525**
1706	**5555**
1708	**1234**
170A	**2222**

Ex. No 4A STRING OPERATION - REVERSE A STRING

AIM:

To write an assembly language program to perform reverse a string operation using 8086 Microprocessor kit.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Power supply	1

ALGORITHM:

Step 1: Start the process
Step 2: Set the counter value.
Step 3: Read the starting memory locations
Step 4: Load the source data and push to stack
Step 5: Repeat the step 4 until count reach zero.
Step 6: Load the count value again
Step 7: Pop the stack and store to Destination data.
Step 8: Repeat the step 4 until count reach zero.
Step 9: Stop the process

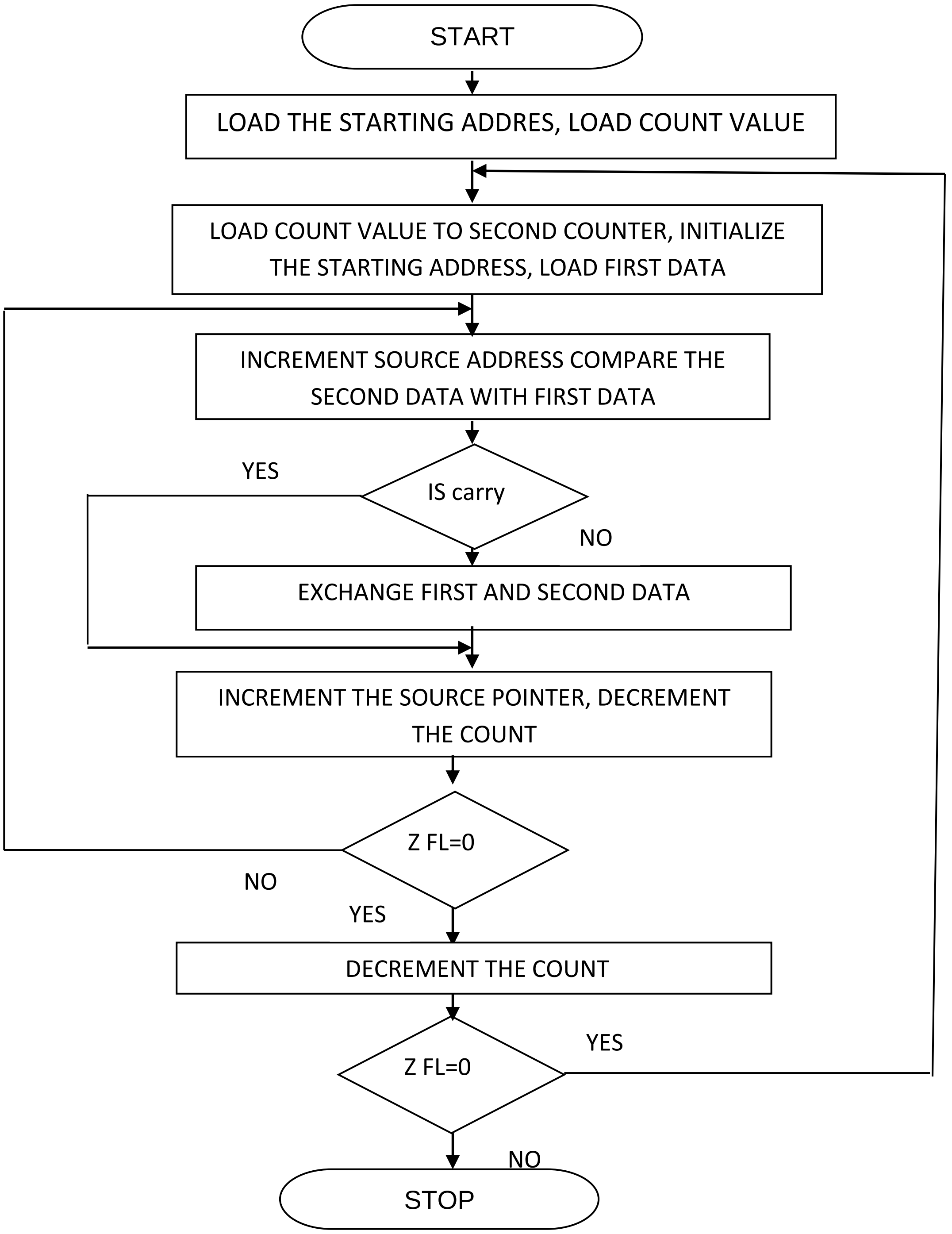
PROGRAM

Address	Opcode	Instruction	Comments
1500	C7C20A06	MOV DX,0006	Move 0005 to DX
1504	FC	CLD	Clear Directional Flag
1505	89D1	MOV CX,DX	Move DX to CX
1507	C7C60016	MOV SI,1600	Initialize source stating address
150B	C7C70017	MOV DI,1700	Initialize destination stating address
150F	AD	LODSW	Load string byte
1510	50	PUSH AX	Push AX Value
1511	E2FC	LOOP 150F	Repeat Load and Push till CX=0
1513	89D1	MOV CX,DX	Move DX to CX
1515	58	POP AX	Pop AX Value
1516	AB	STOSW	Store string byte
1517	E2FC	LOOP 1515	Repeat Load and Push till CX=0
1519	F4	HLT	Halt

RESULT:

Thus the assembly language program to perform reverse a string operation was executed successfully and the result was verified by using 8086 Microprocessor.

FLOWCHART


START
LOAD THE STARTING ADDRES, LOAD COUNT VALUE
LOAD COUNT VALUE TO SECOND COUNTER, INITIALIZE THE STARTING ADDRESS, LOAD FIRST DATA
INCREMENT SOURCE ADDRESS COMPARE THE SECOND DATA WITH FIRST DATA
YES
IS carry
NO
EXCHANGE FIRST AND SECOND DATA
INCREMENT THE SOURCE POINTER, DECREMENT THE COUNT
Z FL=0
NO
YES
DECREMENT THE COUNT
YES
Z FL=0
NO
STOP

BEFORE EXECUTION:

1100	**4444**
1102	**3333**
1104	**2222**
1106	**6666**
1108	**1111**

AFTER EXECUTION:

ASCENDING		**DESCENDING**	
1100	**1111**	**1100**	**6666**
1102	**2222**	**1102**	**4444**
1104	**3333**	**1104**	**3333**
1106	**4444**	**1106**	**2222**
1108	**6666**	**1108**	**1111**

Ex. No 4B ASCENDING / DESCENDING ORDER

AIM:

To write an assembly language program to sort the array of numbers in ascending/ descending order by using 8086 processor kit.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	Microprocessor 8086	1
2.	Power Supply	1

ALGORITHM:

STEP 1: Load the count value and starting address of the array.
STEP 2: Load the second loop count value and starting address of the array
STEP 3: Increment address register.
STEP 4: Load the first data to AX register.
STEP 5: Compare the AX register with next data.
STEP 6: If second data less than first data swap the data.
STEP 7: Increment address register and decrement the count.
STEP 8: Repeat the step 5,6 and 7 until count reach zero.
STEP 9: Decrement the first count.
STEP 10: Repeat the step 2 to 9 until count reach zero.
STEP 11: Stop the execution.

PROGRAM:

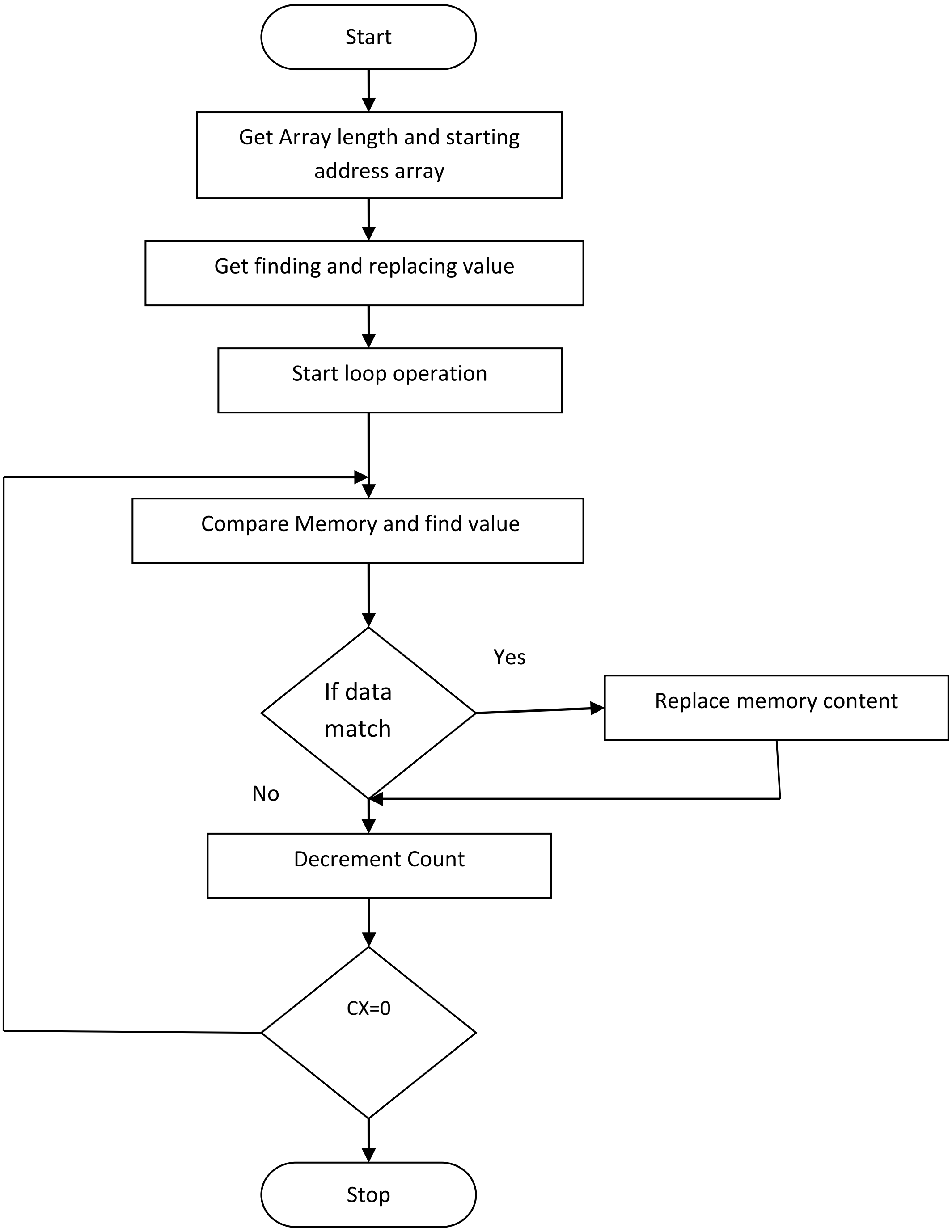
ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1000	C7 C2 05 00		MOV DX, 05	Load count to DX
1004	89 D1	LOOP2:	MOV CX, DX	Load count DX in to CX
1006	C7 C6 00 11		MOV SI, 1100	Move starting address to SI
100A	8B 04	AGAIN:	MOV AX, [SI]	Load the data from SI
100C	3B 44 02		CMP AX, [SI+2]	Compare with next data
100F	72 05		JC/JNC LOOP1	Carry/No Carry jump to Loop1
1011	87 44 02		XCHG [SI +2], AX	Exchange second data with AX
1014	87 04		XCHG [SI], AX	Exchange first data with AX
1016	81 C6 02 00	LOOP1:	ADD SI, 02	Increment SI twice
101A	E2 EE		LOOP AGAIN	If CX not zero jump to AGAIN

101C	4A		DEC DX	Decrement DX
101D	75 E5		JNZ LOOP2	If DX not zero jump to LOOP2
101F	F4		HLT	Halt

RESULT:

Thus the assembly language program to sort the array of number in ascending / descending order was executed successfully and the result was verified by using Microprocessor kit.

FLOWCHART


Start
Get Array length and starting address array
Get finding and replacing value
Start loop operation
Compare Memory and find value
If data match
Yes
Replace memory content
No
Decrement Count
CX=0
Stop

BEFORE EXECUTION:

2000	00
2001	0E
2002	22
2003	1D
2004	22
2005	3C

AFTER EXECUTION:

2000	00
2001	0E
2002	45
2003	1D
2004	45
2005	3C

Ex. No 4C FIND AND REPLACE

AIM:

To write an assembly language program to find a number and replace with another number using 8086 Microprocessor kit.

APPARATUS REQUIRED

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Power supply	1

ALGORITHM:

Step 1: Start the process
Step 2: Set the counter value to cx
Step 3: Set the starting locations of Source address to SI reg
Step 4: Move the finding value to BL reg
Step 5: Move the replace value to DL reg
Step 6: Compare BL reg with Memory content
Step 7: If zero flag not set go to Step 9
Step 8: Store the replace value
Step 9: Increment SI for next data
Step 10: repeat step 6 to step 9 till Cx =0
Step 11: Stop the process

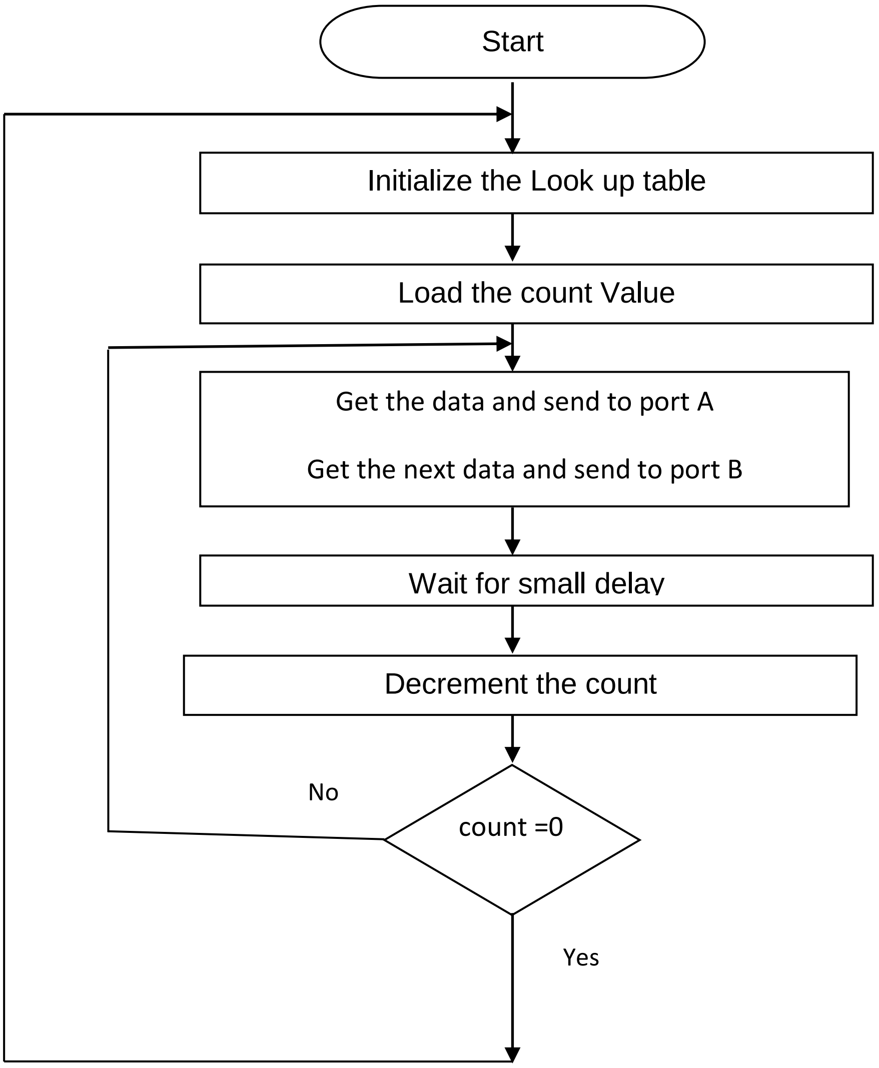
PROGRAM

ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1000	C6 C1 06		MOV CL, 06H	Set the counter value
1003	C7 C6 00 20		MOV SI, 2000H	Initialize the source values
1007	C6 C3 22		MOV BL, 22H	Assign the searching value to BL
100A	C6 C2 45		MOV DL, 45H	Assign the replace value to DL
100D	3A 1C	AA	CMP BL,[SI]	Compare the BL value with SI content
100F	75 02		JNE BB	Jump to loop when both No are not equal
1011	88 14		MOV [SI], DL	If equal, move the DL to SI memory pointer
1013	46	BB	INC SI	Increment the SI by 1
1014	E2 F7		LOOP AA	Check the CX value is 0000H
1016	F4		HLT	Stop the program

RESULT:

Thus the assembly language program to find a number and replace with another number was executed successfully and the result was verified by using 8086 Microprocessor.

FLOWCHART:


Start
Initialize the Look up table
Load the count Value
Get the data and send to port A
Get the next data and send to port B
Wait for small delay
Decrement the count
No
count =0
Yes

Ex. No 5 TRAFFIC LIGHT CONTROLLER USING 8086

AIM:

To write an assembly language program to interface a Traffic light controller using 8086 microprocessor.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Traffic light controller interface board	1
3.	Power Supply	1

ALGORITHM:

STEP 1: Initialize look up table starting address and count value.

STEP 2: Initialize all port as output port.

STEP 3: Do the following steps until the counter reaches 0.

1. Load the first element in the accumulator
2. Send the value to port A.
3. Load the next element in the accumulator
4. Send the value to port B.
5. Load the next element in the accumulator
6. Send the value to port C.
7. Decrement the count
8. Call delay.

STEP 4: Repeat the step three until count reaches zero.

STEP 5: Repeat the step three and four until kit reset.

D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0			
GL	Y	R	GS	W	W	W	W	GR	GL	Y	R	GS	GR	GL	Y	R	GS	GR	GL	Y	R	GS	GR	A	B	C
S	S	S	S	S	W	N	E	S	W	W	W	W	W	N	N	N	N	N	E	E	E	E	E			
1	0	1	0	1	1	1	1	0	1	0	1	0	0	1	0	1	0	0	1	0	0	1	1	AF	52	93
1	0	1	0	1	1	1	1	0	1	0	1	0	0	1	0	1	0	0	1	1	0	0	0	AF	52	98
1	0	0	1	1	1	1	1	1	1	0	1	0	0	1	0	1	0	0	1	0	1	0	0	9F	D2	94
1	1	0	0	1	1	1	1	0	1	0	1	0	0	1	0	1	0	0	1	0	1	0	0	CF	52	94
1	0	1	0	1	1	1	1	0	1	0	0	1	1	1	0	1	0	0	1	0	1	0	0	AF	4E	94
1	0	1	0	1	1	1	1	0	1	1	0	0	0	1	0	1	0	0	1	0	1	0	0	AF	62	94
1	0	1	0	1	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	1	0	0	AF	52	74
1	0	1	0	1	1	1	1	0	1	0	1	0	0	1	1	0	0	0	1	0	1	0	0	AF	53	14
0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	20	10	84

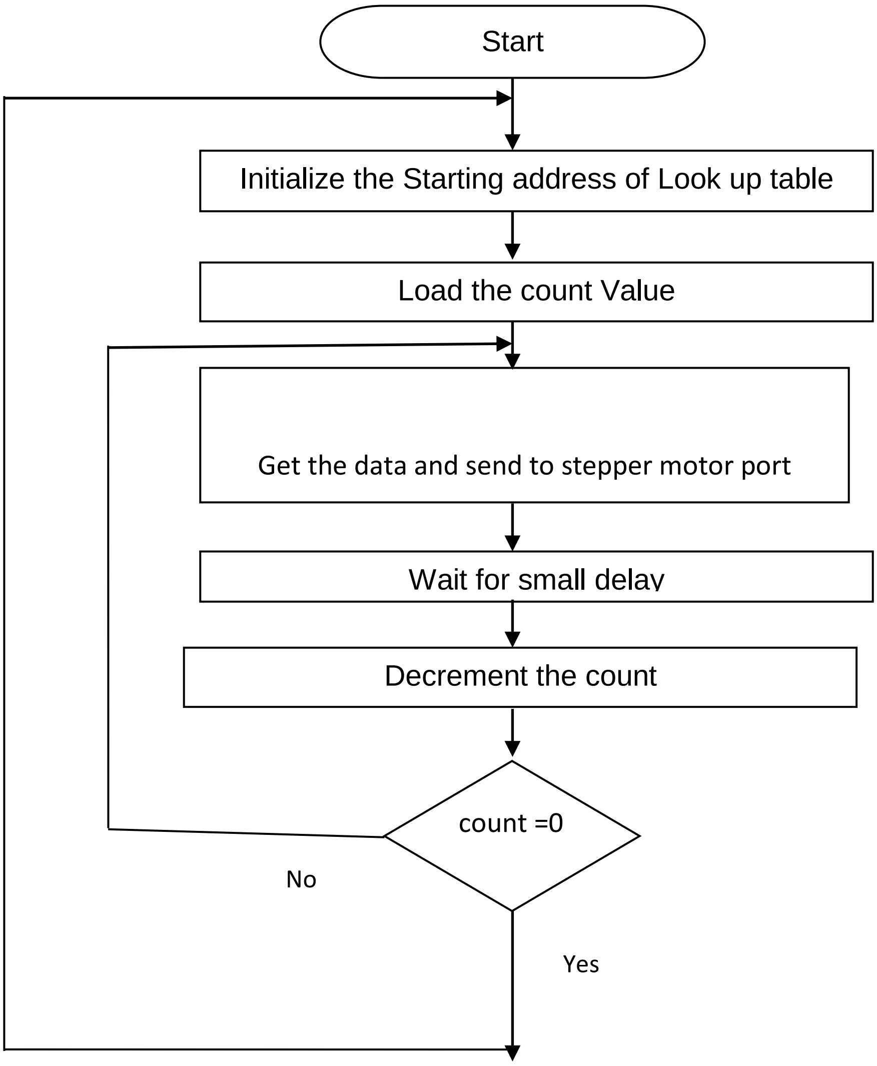
PROGRAM:

ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1000	C7 C3 00 11	START	MOV BX, 1100	Initialize Look up table address
1004	C7 C1 0C 00		MOV CX,000C	Move Count Value to CX
1008	8A 07		MOV AL,[BX]	Load the Data
100A	E6 26		OUT 26,AL	Send to Control word
100C	43		INC BX	Increment memory location
100D	8A 07	NEXT	MOV AL,[BX]	Load the Data
100F	E6 20		OUT 20,AL	Send to Port A
1011	43	BB	INC BX	Increment memory location
1012	8A 07		MOV AL,[BX]	Load the Data
1014	E6 22		OUT 22,AL	Send to Port B
1016	43	BB	INC BX	Increment memory location
1017	8A 07		MOV AL,[BX]	Load the Data
1019	E6 24		OUT 24,AL	Send to Port C
101B	E8 06 00		CALL DELAY	Call Delay
101E	43		INC BX	Increment memory location
101F	E2 EC		LOOP NEXT	If CX ≠ 0; Jump to NEXT
1021	E9 DC FF		JMP START	Jump to START
1024	51	DELAY	PUSH CX	Push CX to stack
1025	C7 C1 05 00		MOV CX,0005	Move the Data 05 to CX register
1029	C7 C2 FF FF	REPEAT	MOV DX,0FFFF	Move the Data FFFF to DX register
102D	4A	LOOP2	DEC DX	Decrement DX
102E	75 FD		JNZ LOOP 2	If DX ≠ 0; Jump to LOOP2
1030	E2 F7		LOOP REPEAT	If CX ≠ 0; Jump to REPEAT
1032	59		POP CX	Pop CX from stack
1033	C3		RET	Return

RESULT:

Thus a traffic light was interfaced with 8086 and result was verified.

FLOWCHART:


Start
Initialize the Starting address of Look up table
Load the count Value
Get the data and send to stepper motor port
Wait for small delay
Decrement the count
count =0
No
Yes

Ex. No 6 STEPPER MOTOR INTERFACING WITH 8086

AIM:

To write an assembly language program to interface a stepper motor with 8086 microprocessor and operate it.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Stepper Motor interface board	1
3.	Stepper Motor	1
4.	Power Supply	1

ALGORITHM:

```
    STEP 1: Initialize look up table starting address and count value.
    STEP 2: Do the following steps until the counter reaches 0.
    1.    Load the first element in the accumulator
    2.    Send the value to stepper motor port.
    3.    Decrement the count
    4.    Call delay.
  STEP 3: Repeat the step one, two until reset the kit.
```

THEORY:

A motor in which the rotor is able to assume only discrete stationary angular position is a stepper motor. The rotary motion occurs in a step-wise manner from one equilibrium position to the next. Stepper Motors are used very wisely in position control systems like printers, disk drives, process control machine tools, etc.

The basic two-phase stepper motor consists of two pairs of stator poles. Each of the four poles has its own winding. The excitation of any one winding generates a North Pole. A South Pole gets induced at the diametrically opposite side. The rotor magnetic system has two end faces. It is a permanent magnet with one face as South Pole and the other as North Pole.

The Stepper Motor windings A1, A2, B1, B2 are cyclically excited with a DC current to run the motor in clockwise direction. By reversing the phase sequence as A1, B2, A2, B1, anticlockwise stepping can be obtained.

2-PHASE SWITCHING SCHEME:

In this scheme, any two adjacent stator windings are energized. The switching scheme is table. This scheme produces more torque.

ENERGIZING SCHEME TABLE:

ANTICLOCKWISE						CLOCKWISE					
STEP	**A1**	**A2**	**B1**	**B2**	**DATA**	**STEP**	**A1**	**A2**	**B1**	**B2**	**DATA**
1	**1**	**0**	**0**	**1**	**9h**	**1**	**1**	**0**	**1**	**0**	**Ah**
2	**0**	**1**	**0**	**1**	**5h**	**2**	**0**	**1**	**1**	**0**	**6h**
3	**0**	**1**	**1**	**0**	**6h**	**3**	**0**	**1**	**0**	**1**	**5h**
4	**1**	**0**	**1**	**0**	**Ah**	**4**	**1**	**0**	**0**	**1**	**9h**

PROCEDURE:

Enter the above program starting from location 1000.and execute the same. The stepper motor rotates. Varying the count at CX vary the speed. Entering the data in the look-up TABLE in the reverse order can vary direction of rotation.

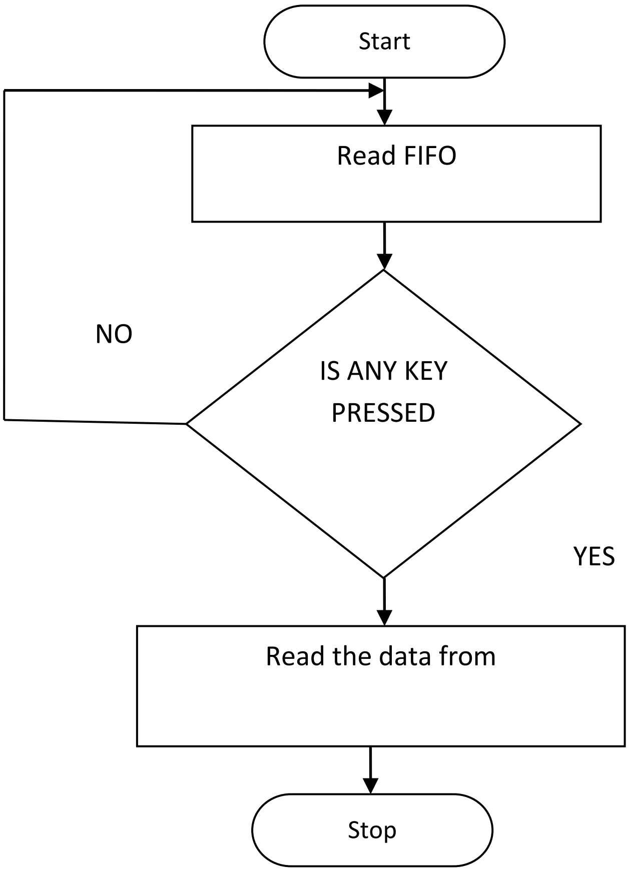
PROGRAM :

ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1000	C7 C7 14 10	START	MOV DI, 1014	Initialize Look up table address
1004	C6 C1 04		MOV CL,04	Move Count Value to CX
1007	8A 05	LOOP1	MOV AL,[DI]	Load the first data
1009	E6 C0		OUT C0,AL	Send the data to Port
100B	C7 C2 10 10		MOV DX,1010	Move the Data 1010 to DX register
100F	4A	DELAY	DEC DX	Decrement DX
1010	75 FD		JNZ DELAY	If DX ≠ 0; Jump to DELAY
1012	47		INC DI	Increment DI
1013	E2 F2		LOOP LOOP1	If CX ≠ 0; Jump to LOOP1
1015	E9 E8 FF		JMP START	Jump to START
1018	09 05 06 0A			Look up Table data

RESULT:

Thus a stepper motor was interfaced with 8085 and the stepper motor rotation in forward and reverse directions at various speeds was verified.

FLOWCHART



SAMPLE INPUT AND OUTPUT:

4200	02

Ex. No 7A

INTERFACING 8279 WITH 8086 AND

READ A KEY FROM KEYBOARD

AIM:

To write an assembly language program to Read a key from the key keyboard and store it using 8279.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Power supply	1
3.	8279 Interface card	1

ALGORITHM:

Step 1: Start the program.
Step 2: Select display/ keyboard mode.
Step 3: Send the value to control word register.
Step 4: Is any key pressed go to next step else wait here.
Step 5: Read the key and store it in memory.
Step 6: Stop the program.

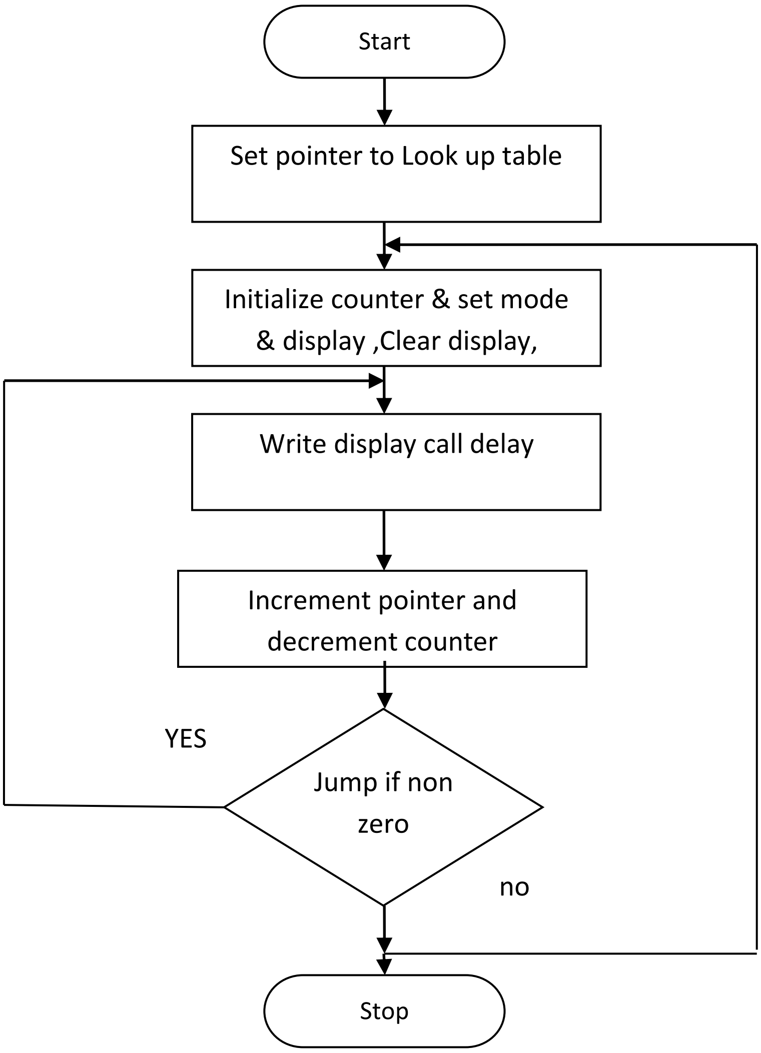
PROGRAM:

ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1000	C7 C3 00 11	START	MOV BX, 1100	Move 1100 into BX
1004	E4 C2	LOOP1	IN AL,C2	Read port C2
1006	F6 C0 07		TEST AL,07	Test AL with 07
1009	74 F8		JZ LOOP1	Z=1,jump to loop1
100B	C6 C0 40		MOV AL,40	Move 40 into AL
100E	E6 C2	DELAY	OUT C2, AL	Send to port C2
1010	E4 C0		IN AL, C0	Read port C0
1012	88 07		MOV [BX],AL	Store to 1100
1014	F4		HLT	halt

RESULT:

Thus the assembly language program to read a key and store into memory location was executed successfully using 8279.

FLOWCHART


Start
Set pointer to Look up table
Initialize counter & set mode & display ,Clear display,
Write display call delay
Increment pointer and decrement counter
YES
Jump if non zero
no
Stop

SEVEN SEGMENT DISPLAY

Char	D7	D6	D5	D4	D3	D2	D1	D0	hex
	d	c	b	a	dp	g	f	e	
H	1	0	0	1	1	0	0	0	98
E	0	1	1	0	1	0	0	0	68
L	0	1	1	1	1	1	0	0	7C
P	1	1	0	0	1	0	0	0	C8
	1	1	1	1	1	1	1	1	FF
U	0	0	0	1	1	1	0	0	1C
S	0	0	1	0	1	0	0	1	29

AIM:

To write an assembly language program to display the rolling message “HELP US” in the seven segments displays.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Power supply	1
3.	8279 Interface card	1

ALGORITHM:

Step 1: Start the program.
Step 2: Set the pointer to starting address of look up table
Step 3: initialize the counter
Step 4: Set mode and display
Step 5: Send the value to control word register.
Step 6: Clear the display.
Step 7: Write the data to be displayed.
Step 8: increment the pointer and decrement the counter.
Step 9: If count is not zero go to step 7
Step 10: Repeat the step 2 step 9.

PROGRAM

ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1000	C7 C6 00 12	START	MOV SI, 1200	Initialize SI with 1200
1004	C7 C1 0F 00		MOV CX,000F	Load count value
1008	C6 C0 10		MOV AL,10	Load AL with 10
100B	E6 C2		OUT C2, AL	Send to port C2
100D	C6 C0 CC		MOV AL,CC	Load AL with cc
1010	E6 C2		OUT C2, AL	Send to port C2
1012	C6 C0 90		MOV AL,90	Load AL with 90
1015	E6 C2		OUT C2,AL	Send to port C2
1017	8A 04	NXT	MOV AL,[SI]	Load Look up table data
1019	E6 CO		OUT C0, AL	Send to port C0
101B	E8 E7 04		CALL DELAY	Call delay
101E	46		INC SI	Increment SI
101F	E2 F6		LOOP NXT	If CX not zero jump to NXT

1021	EB E2		JMP START	Jump START

ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1500	C7 C2 FF A0	DELAY	MOV DX, 0A0FF	Load DX with A0FF
1504	4A	LOOP 1	DEC DX	Decrement DX
1505	73 FD		JNZ LOOP 1	If DX not zero jump to loop1
1507	C3		RET	Return

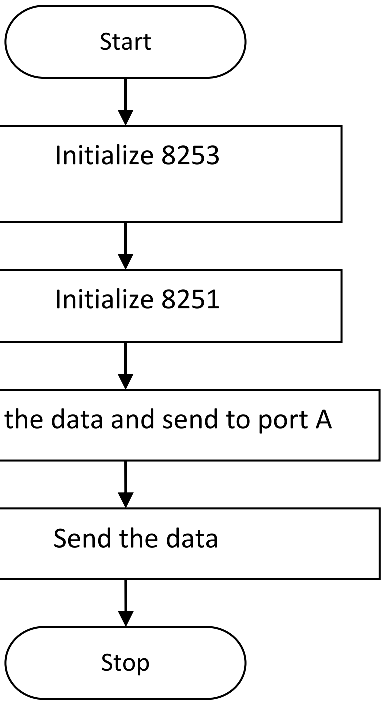
ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1200	FF FF FF FF FF FF FF FF			Look up table
1208	98 68 7C C8 FF 1C 29 FF			

RESULT:

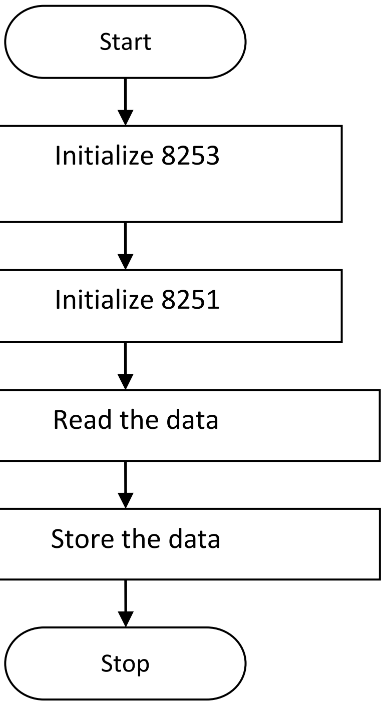
Thus the assembly language program to display the rolling message “HELP US” executed successfully.

FLOWCHART:

TRANSMITTER:



RECEIVER:



Ex. No 8A SERIAL COMMUNICATION USING 8086

AIM:

To write an assembly language program to interface 8251 with 8086 and perform serial communication.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Power supply	1
3.	8255 Interface card	1

ALGORITHM:

TRANSMITTER:

1. Initialize the 8253.
2. Move the mode command word (4E) to Accumulator
3. Output the accumulator to port address C2
4. Move the command instruction word (37) to Accumulator.
5. Output the accumulator to port address C2
6. Move the data to be transmitted to accumulator
7. Output the accumulator to port address C0
8. Reset the system.

RECEIVER:

1. Initialize the 8253.
2. Move the mode command word (4E) to Accumulator
3. Output the accumulator to port address C2
4. Move the command instruction word (37) to Accumulator.
5. Output the accumulator to port address C2
6. Read the data from port address C0
7. Store the received data.
8. Reset the system.

PROCEDURE:

1. Connect two 8086 kits using 9 PIN D type cable through 8251.
2. Load Transmitter program in One kit and Receiver in the other kit
3. Execute Receiver and then Transmitter and again Receiver.
4. Verify the result at 1500.

SAMPLE INPUT AND OUTPUT:

OUTPUT

1500	41

PROGRAM:

TRANSMITTER:

ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1000	C6 C0 36	START	MOV AL, 36	Move the Data 36 to AL register
1003	E6 CE		OUT CE, AL	Send the data to Port CE
1005	C6 C0 10		MOV AL,10	Move the Data 10 to AL register
1008	E6 C8		OUT C8,AL	Send the data to Port C8
100A	C6 C0 00		MOV AL,00	Move the Data 00 to AL register
100D	E6 C8		OUT C8,AL	Send the data to Port C8
100F	C6 C0 4E		MOV AL,4E	Move the Data 4E to AL register
1012	E6 C2		OUT C2,AL	Send the data to Port C2
1014	C6 C0 37		MOV AL,37	Move the Data 37 to AL register
1017	E6 C2		OUT C2,AL	Send the data to Port C2
1019	E4 C2	LOOP1	IN AL,C2	Read the port C2
101B	80 E0 04		AND AL,04	A and with 04
101E	74 F9		JZ LOOP1	If CX ≠ 0; Jump to LOOP1
1020	C6 C0 41		MOV AL,41	Move the Data 41 to AL register
1023	E6 C0		OUT C0, AL	Send the data to Port C0
1025	CD 02		INT 2	Interrupt

RECEIVER:

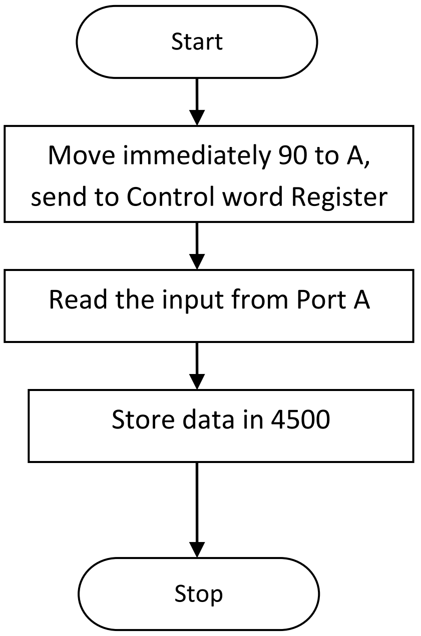
ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1200	C6 C0 36	START	MOV AL, 36	Move the Data 36 to AL register
1203	E6 CE		OUT CE, AL	Send the data to Port CE
1205	C6 C0 10		MOV AL,10	Move the Data 10 to AL register
1208	E6 C8		OUT C8,AL	Send the data to Port C8
120A	C6 C0 00		MOV AL,00	Move the Data 00 to AL register
120D	E6 C8		OUT C8,AL	Send the data to Port C8

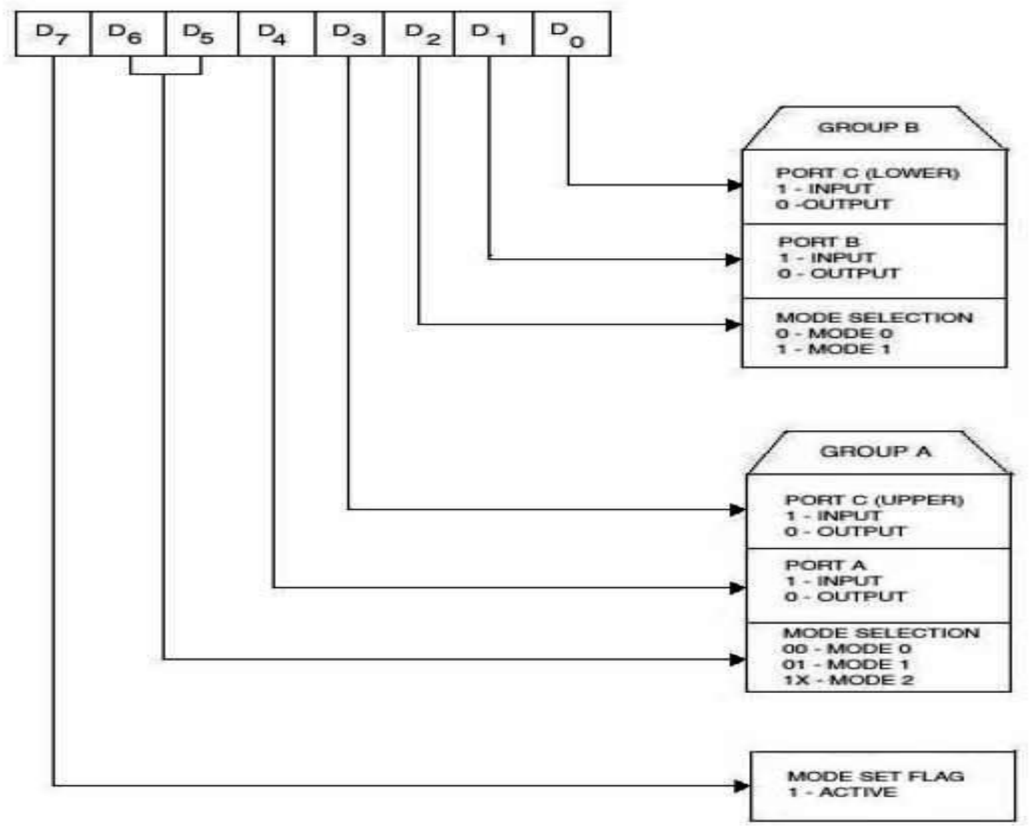
120F	C6 C0 4E		MOV AL,4E	Move the Data 4E to AL register
1212	E6 C2		OUT C2,AL	Send the data to Port C2
1214	C6 C0 37		MOV AL,37	Move the Data 37 to AL register
1217	E6 C2		OUT C2,AL	Send the data to Port C2
1219	E4 C2	LOOP1	IN AL,C2	Read from port C2
121B	80 E0 02		AND AL,02	A and with 02
121E	74 F9		JZ LOOP1	If CX ≠ 0; Jump to LOOP1
1220	E4 C0		IN AL,C0	Read from port C0
1222	C7 C3 00 15		MOV BX,1500	Move the Data 1500 to BX register
1226	88 07		MOV [BX],AL	Store data to 1500
1228	CD 02		INT 2	Interrupt

RESULT:

Thus the assembly language program to interface 8251 with 8086 and perform serial communication was executed and result was verified.

FLOWCHART


Start
Move immediately 90 to A, send to Control word Register
Read the input from Port A
Store data in 4500
Stop



SAMPLE INPUT AND OUTPUT:

INPUT	**OUTPUT**
Port A: 1101 1010	4500: DA

Ex. No 8B INTERFACING 8255 WITH 8086 IN MODE 0

AIM:

To write an assembly language program to interface 8255 with 8086 in mode 0.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Power supply	1
3.	8255 Interface card	1

ALGORITHM:

Step 1: Start the program.
Step 2: Move immediate data 90 to A.
Step 3: Selecting output from control register.
Step 4: Read the input from port.
Step 5: Store the data in 1500.
Step 6: Stop the program.

Step 1: Start the program.
Step 2: Move immediate data 90 to A.
Step 3: Selecting output from control register.
Step 4: Read the input from port.
Step 5: Send to Port B.
Step 6: Stop the program.

PROGRAM:

ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1000	C7 C6 00 15	START	MOV SI, 1500	
1004	C6 C0 90		MOV AL,90	
1007	E6 C6		OUT C6,AL	
1009	E4 C0		IN AL,C0	
100B	88 04		MOV [SI],AL	
100D	F4		HLT	

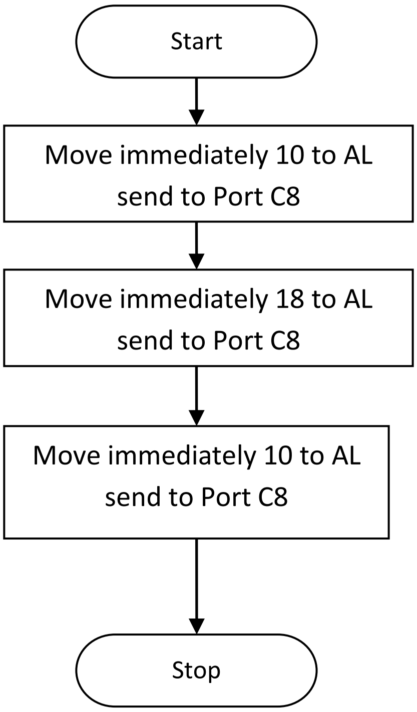
ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1000	C6 C0 90		MOV AL,90	
1003	E6 C6		OUT C6,AL	

1005	E4 C0		IN AL,C0	
1007	E6 C2		OUT C2,AL	
1009	E9 F9 FF		JMP 1000	

RESULT:

Thus the assembly language program to interface 8255 with 8086 in mode 0.

FLOWCHART


Start
Move immediately 10 to AL send to Port C8
Move immediately 18 to AL send to Port C8
Move immediately 10 to AL send to Port C8
Stop

Jumper J2 in B position

Jumper J5 in A position.

Ex. No 9A INTERFACING ADC WITH 8086

AIM:

To write an assembly language program to interface ADC with 8086.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Power supply	1
3.	ADC Interface card	1

ALGORITHM:

Step 1: Start the program.
Step 2: Move immediate data 10 to AL.
Step 3: Send the data to Port C8.
Step 4: Move immediate data 18 to AL.
Step 5: Send the data to Port C8.
Step 6: Move immediate data 10 to AL.
Step 7: Send the data to Port C8.
Step 8: Stop the program.

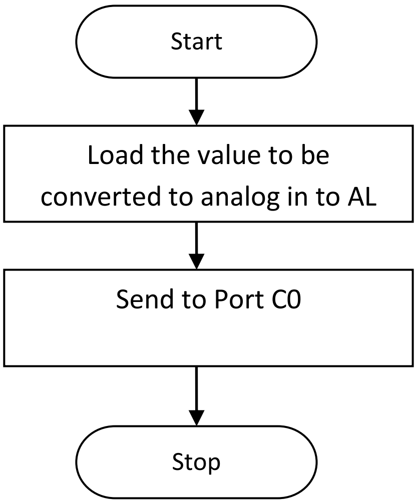
PROGRAM:

ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1000	C6 CO 10		MOV AL,10	Move the Data 10 to AL register
1003	E6 C8		OUT C8,AL	Send the data to Port C8
1005	C6 C0 18		MOV AL,18	Move the Data 18 to AL register
1008	E6 C8		OUT C8,AL	Send the data to Port C8
100A	C6 C0 10		MOV AL,10	Move the Data 10 to AL register
100D	E6 C8		OUT C8,AL	Send the data to Port C8
100F	F4		HLT	Halt the program

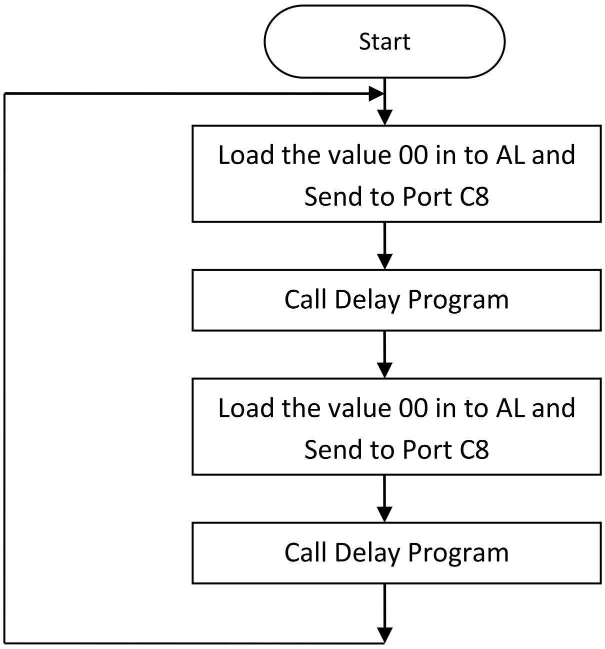
RESULT:

Thus the assembly language program to interface ADC with 8086 was executed and result was verified.

FLOWCHART
DAC INTERFACE



SQUARE WAVEFORM


Start
Load the value 00 in to AL and Send to Port C8
Call Delay Program
Load the value 00 in to AL and Send to Port C8
Call Delay Program

Ex. No 9B INTERFACING DAC WITH 8086 AND WAVEFORM GENERATION

AIM:

To write an assembly language program to interface DAC with 8086 and generate square and saw tooth waveforms.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Power supply	1
3.	DAC Interface card	1
4.	CRO	1

ALGORITHM:

DAC INTERFACE

Step 1: Start the program.
Step 2: Move immediate to be converted to analog in to AL.
Step 3: Send the data to Port C0.
Step 4: Stop the program.

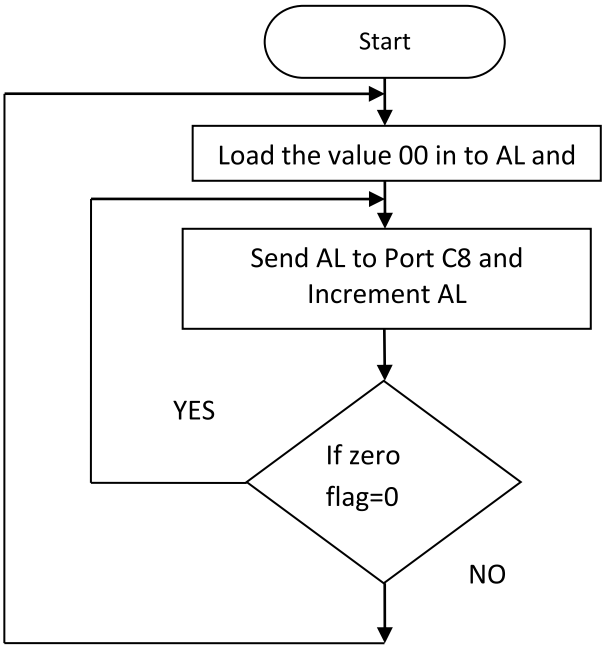
SQUARE WAVEFORM

Step 1: Start the program.
Step 2: Move immediate value 00 in to AL.
Step 3: Send the data to Port C8.
Step 4: Call delay program.
Step 5: Move immediate value FF in to AL.
Step 6: Send the data to Port C8.
Step 7: Call delay program.
Step 8: Repeated step 2 to 7.
Step 9: Stop the program.

SAW TOOTH WAVEFORM

Step 1: Start the program.
Step 2: Move immediate value 00 in to AL.
Step 3: Send the data to Port C8.
Step 4: Increment AL
Step 5: If zero flag not set send to step 3.
Step 6: Repeated step 2 to 5.
Step 7: Stop the program.

SAW TOOTH WAVEFORM


Start
Load the value 00 in to AL and
Send AL to Port C8 and Increment AL
YES
If zero flag=0
NO

Wave form	Amplitude	Time period
SQUARE		
SAW TOOTH		

PROGRAM:

DAC INTERFACE

ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1000	C6 C0 7F		MOV AL,7F	Move the Data 7F to AL register
1003	E6 C0		OUT C0,AL	Send the data to Port C0
1005	F4		HLT	Halt the program

SQUARE WAVEFORM

ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1000	C6 C0 00	START	MOV AL,00	Move the Data 00 to AL register
1003	E6 C8		OUT C8,AL	Send the data to Port C8
1005	E8 08 00		CALL 1010	Call the delay subroutine
1008	C6 C0 FF		MOV AL,FF	Move the Data FF to AL register
100B	E6 C8		OUT C8,AL	Send the data to Port C8
100D	E8 00 00		CALL 1010	Call the delay subroutine
1010	E9 ED FF		JMP 1000	Jump to start
1013	C7 C1 FF 05	XX	MOV CX,05FF	Move the Data 05FF to CX register
1017	E2 FF		LOOP 1013	If CX ≠ 0; Jump to XX
1019	C3		RET	Return

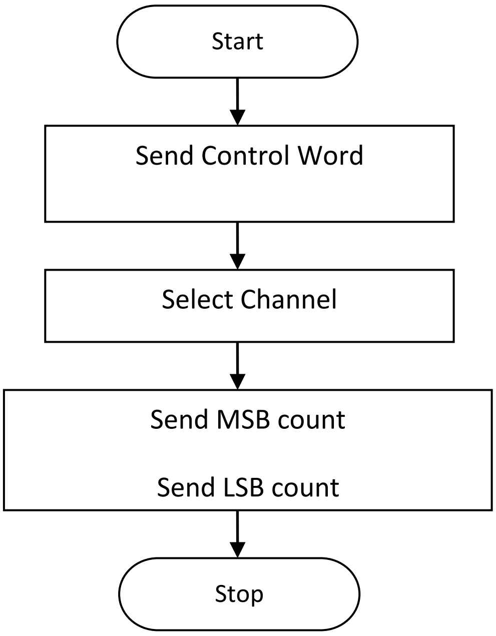
SAW TOOTH WAVEFORM

ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1100	C6 C0 00	START	MOV AL,00	Move the Data 00 to AL register
1103	E6 C0	XX	OUT C0,AL	Send the data to Port C0
1105	FE C0		INC AL	Increment AL
1107	75 FA		JNZ 1002	If Zero flag not set Jump to XX
1108	E9 F4 FF		JMP 1000	Jump to start

RESULT:

Thus the assembly language program to interface DAC with 8086 was executed and square and saw tooth waveforms are generated.

FLOWCHART:



DIFFERENT MODES OF OPERATION:

Mode 0 – Interrupt on terminal count:

The output will be initially low after mode set operations. After loading the counter, the output will be remaining low while counting and on terminal count; the output will become high, until reloaded again.

Let us set the channel 0 in mode 0. Connect the CLK 0 to the debounce circuit by changing the jumper J3 and then execute the following program.

It is observed in CRO that the output of Channel 0 is initially LOW. After giving six clock pulses, the output goes HIGH.

Mode 1 – Programmable ONE-SHOT:

After loading the counter, the output will remain low following the rising edge of the gate input. The output will go high on the terminal count. It is re triggerable hence the output will remain low for the full count, after any rising edge of the gate input.

Example:

The following program initializes channel 0 of 8253 in Mode 1 and also initiates triggering of Gate 0. OUT 0 goes low, as clock pulse after triggering the goes back to high level after 5 clock pulses. Execute the program, give clock pulses through the debounce logic and verify using CRO.

Ex. No 10 **INTERFACING 8253 TIMER / COUNTER WITH 8086**

AIM:

To write an assembly language program to interface 8253 timer with 8086.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Power supply	1
3.	8253 Interface card	1
4.	CRO	1

ALGORITHM:

Step 1: Start the program.
Step 2: Send Control Word.
Step 3: Selecting Channel.
Step 4: Send MSB and LSB count.
Step 5: Stop the program.

PROGRAM:

MODE 0

Address	Op codes	Label	Mnemonic	Comments
1000	C6 C0 30	START:	MOV AL,30	Channel 0 in mode 0
1003	E6 CE		OUT CE,AL	Send Mode Control word
1005	C6 C0 05		MOV AL,05	LSB of count
1008	E6 C8		OUT C8,AL	Write count to register
100A	C6 C0 00		MOV AL,00	MSB of count
100D	E6 C8		OUT C8,AL	Write count to register
100F	F4		HLT	

MODE 1

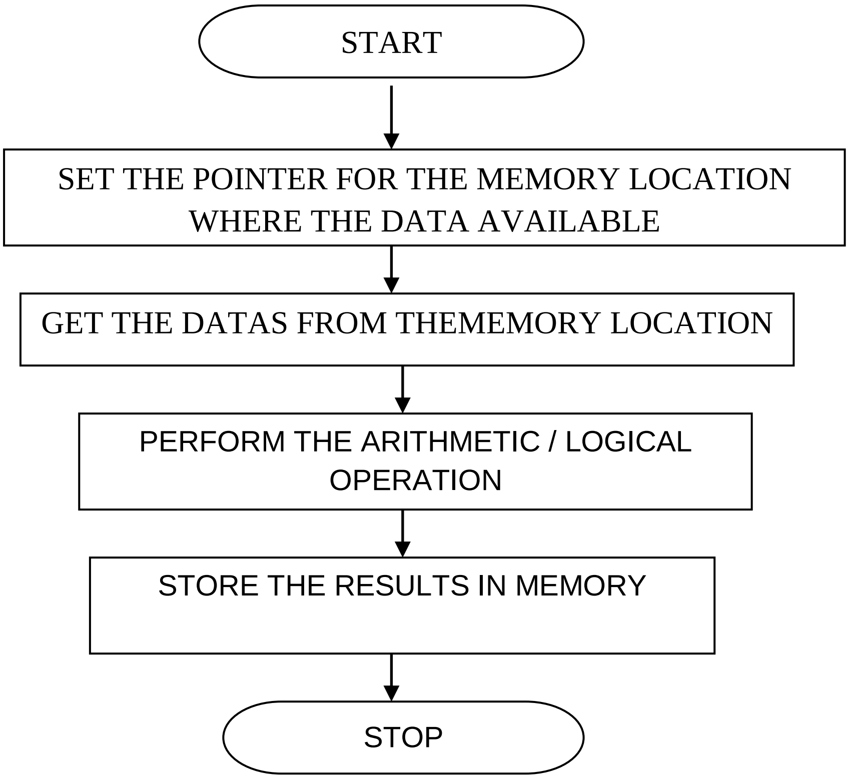
Address	Op codes	Label	Mnemonic	Comments
1100	C6 C0 32	START:	MOV AL,32	Channel 0 in mode 1

1103	E6 CE		OUT CE,AL	Send Mode Control word
1105	C6 C0 05		MOV AL,05	LSB of count
1108	E6 C8		OUT C8,AL	Write count to register
110A	C6 C0 00		MOV AL,00	MSB of count
110D	E6 C8		OUT C8,AL	Write count to register
110F	E6 D0		OUT D0,AL	Trigger Gate0
1111	F4		HLT	

RESULT:

Thus the assembly language program to interface 8253 Timer with 8086.

FLOWCHART:


START
SET THE POINTER FOR THE MEMORY LOCATION WHERE THE DATA AVAILABLE
GET THE DATAS FROM THEMEMORY LOCATION
PERFORM THE ARITHMETIC / LOGICAL OPERATION
STORE THE RESULTS IN MEMORY
STOP

Ex. No 11 ARITHMETIC / LOGICAL OPERATIONS USING 8051

AIM:

To write an assembly language program to perform Arithmetic / Logical operations using 8051 Microcontroller kit.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8051 Microcontroller kit	1
2.	Power supply	1

ALGORITHM:

ADDITION

Step 1: Set the pointer for the memory location where the data available
Step 2: Load the first data to accumulator
Step 3: Move the first data from accumulator to B register.
Step 4: Load the Second data to accumulator
Step 5: Add the B register to accumulator.
Step 6: Store the result in next memory location
Step 7: Stop the process.

SUBTRACTION

Step 1: Set the pointer for the memory location where the data available
Step 2: Load the first data to accumulator
Step 3: Move the first data from accumulator to B register.
Step 4: Load the Second data to accumulator
Step 5: Clear the carry flag
Step 6: Subtract B register from accumulator.
Step 7: Store the result in next memory location
Step 8: Stop the process.

MULTIPLICATION

Step 1: Set the pointer for the memory location where the data available
Step 2: Load the first data to accumulator
Step 3: Move the first data from accumulator to B register.
Step 4: Load the Second data to accumulator
Step 5: Multiply B register with accumulator.
Step 6: Store the result in next memory locations
Step 7: Stop the process

DIVISION

Step 1: Set the pointer for the memory location where the data available
Step 2: Load the first data to accumulator
Step 3: Move the first data from accumulator to B register.
Step 4: Load the Second data to accumulator
Step 5: Divide Accumulator by B register.
Step 6: Store the result in next memory locations

Step 7: Stop the process

SAMPLE INPUT AND OUTPUT:

ADDITION

INPUT	
4200	25
4201	66

OUTPUT	
4202	8B

SUBTRACTION

INPUT	
4200	66
4201	23

OUTPUT	
4202	43

MULTIPLICATION

INPUT	
4200	25
4201	25

OUTPUT	
4202	05
4203	59

DIVISION

INPUT	
4200	25
4201	05

OUTPUT	
4202	
4203	

LOGICAL OPERATIONS

INPUT	
4200	78
4201	CD

OUTPUT	
4202	48/FD/B5

LOGICAL OPERATIONS

Step 1: Set the pointer for the memory location where the data available
Step 2: Load the first data to accumulator
Step 3: Move the first data from accumulator to B register.
Step 4: Load the Second data to accumulator
Step 5: Perform the Logical AND/ OR/ XOR with B register and accumulator.
Step 6: Store the result in next memory location
Step 7: Stop the process.

PROGRAM

ADDITION

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
4100		MOV DPTR,#4200	90 42 00	Move 4200 to DPTR
4103		MOVX A, @DPTR	E0	Get the Data from 4200
4104		MOV B, A	F5 F0	Move data from A to B
4106		INC DPTR	A3	Increment DPTR
4107		MOVX A, @DPTR	E0	Get the Data from 4201
4108		ADD A,B	25 F0	Add A and B
410A		INC DPTR	A3	Increment DPTR
410B		MOVX @DPTR,A	F0	Store the result at 4202
410C	HERE	SJMP HERE	80 FE	Stop the program

SUBTRACTION

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
4100		MOV DPTR,#4200	90 42 00	Move 4200 to DPTR
4103		MOVX A, @DPTR	E0	Get the Data from 4200
4104		MOV B, A	F5 F0	Move data from A to B
4106		INC DPTR	A3	Increment DPTR
4107		MOVX A, @DPTR	E0	Get the Data from 4201
4108		CLR C	C3	Clear Carry Flag
4109		SUBB A,B	95 F0	Subtract B from A
410B		INC DPTR	A3	Increment DPTR
410C		MOVX @DPTR,A	F0	Store the result at 4202
410D	HERE	SJMP HERE	80 FE	Stop the program

MULTIPLICATION

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
4100		MOV DPTR,#4200	90 42 00	Move 4200 to DPTR
4103		MOVX A, @DPTR	E0	Get the Data from 4200
4104		MOV B, A	F5 F0	Move data from A to B
4106		INC DPTR	A3	Increment DPTR
4107		MOVX A, @DPTR	E0	Get the Data from 4201
4108		MUL AB	A4	Multiply A and B
4109		INC DPTR	A3	Increment DPTR
410A		MOVX @DPTR,A	F0	Store the result at 4202
410B		MOV A,B	E5 F0	Move B to A
410D		INC DPTR	A3	Increment DPTR
410E		MOVX @DPTR,A	F0	Store the result at 4203
410F	HERE	SJMP HERE	80 FE	Stop the program

DIVISION

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
4100		MOV DPTR,#4200	90 42 00	Move 4200 to DPTR
4103		MOVX A, @DPTR	E0	Get the Data from 4200
4104		MOV B, A	F5 F0	Move data from A to B
4106		INC DPTR	A3	Increment DPTR
4107		MOVX A, @DPTR	E0	Get the Data from 4201
4108		DIV AB	84	Divide A and B
4109		INC DPTR	A3	Increment DPTR
410A		MOVX @DPTR,A	F0	Store the result at 4202
410B		MOV A,B	E5 F0	Move B to A
410D		INC DPTR	A3	Increment DPTR
410E		MOVX @DPTR,A	F0	Store the result at 4203
410F	HERE	SJMP HERE	80 FE	Stop the program

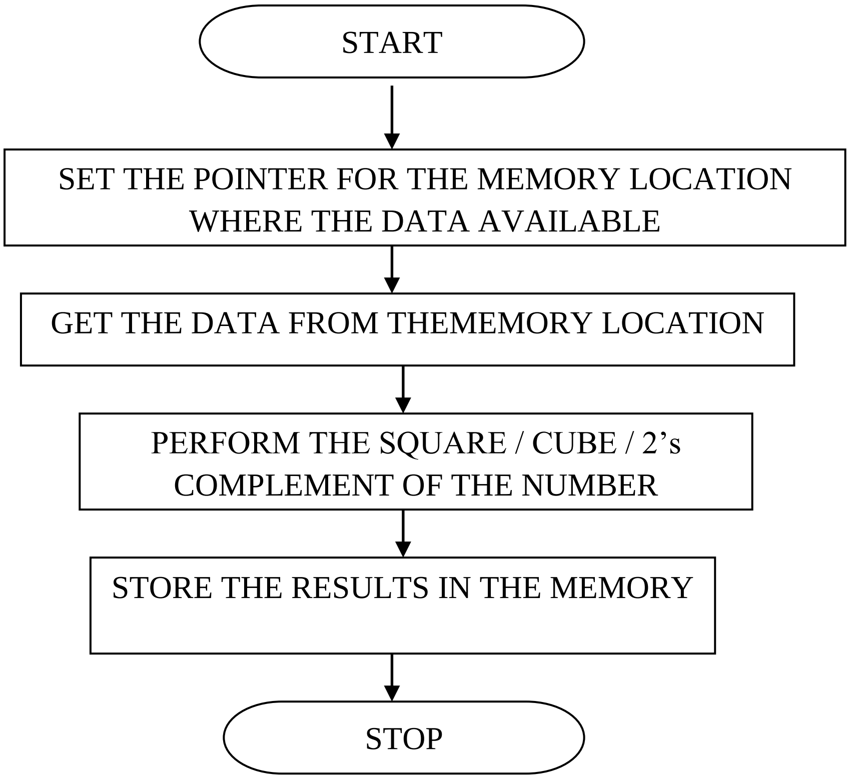
LOGICAL OPERATIONS

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
4100		MOV DPTR,#4200	90 42 00	Move 4200 to DPTR
4103		MOVX A, @DPTR	E0	Get the Data from 4200
4104		MOV B, A	F5 F0	Move data from A to B
4106		INC DPTR	A3	Increment DPTR
4107		MOVX A, @DPTR	E0	Get the Data from 4201
4108		ANL A,B / ORL A,B / XRL A,B	55 F0 45 F0 65 F0	A Logical AND with B A Logical OR with B A Logical XOR with B
410A		INC DPTR	A3	Increment DPTR
410B		MOVX @DPTR,A	F0	Store the result at 4202
410C	HERE	SJMP HERE	80 FE	Stop the program

RESULT:

Thus the assembly language program to perform Arithmetic / Logical operations was executed successfully and the result was verified by using 8051 Microcontroller kit.

FLOWCHART


START
SET THE POINTER FOR THE MEMORY LOCATION WHERE THE DATA AVAILABLE
GET THE DATA FROM THEMEMORY LOCATION
PERFORM THE SQUARE / CUBE / 2’s COMPLEMENT OF THE NUMBER
STORE THE RESULTS IN THE MEMORY
STOP

Ex. No 12A SQUARE / CUBE / 2’S COMPLEMENT OF A NUMBER USING 8051

AIM:

To write an assembly language program to find Square, Cube and 2’s complement of a number using 8051 Microcontroller kit.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8051 Microcontroller kit	1
2.	Power supply	1

ALGORITHM:

SQUARE

Step 1: Set the pointer for the memory location where the data available
Step 2: Load the data to accumulator
Step 3: Move the data from accumulator to B register.
Step 4: Multiply B register with accumulator.
Step 5: Store the result in next memory locations
Step 6: Stop the process.

CUBE

Step 1: Set the pointer for the memory location where the data available
Step 2: Load the data to accumulator
Step 3: Move the data from accumulator to B register.
Step 4: Multiply B register with accumulator.
Step 5: Multiply B register with accumulator.
Step 6: Store the result in next memory locations
Step 7: Stop the process.

2’s COMPLEMENT

Step 1: Set the pointer for the memory location where the data available
Step 2: Load the data to accumulator
Step 3: Complement the Accumulator.
Step 4: Increment the Accumulator.
Step 5: Store the result in next memory location.
Step 6: Stop the process

SAMPLE INPUT AND OUTPUT:

SQUARE

INPUT	
4200	25

OUTPUT	
4201	05
4202	59

CUBE

INPUT	
4200	25

OUTPUT	
4201	C5
4202	DD

2’s COMPLEMENT

INPUT	
4200	25

OUTPUT	
4202	DB

PROGRAM
SQUARE

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
4100		MOV DPTR,#4200	90 42 00	Move 4200 to DPTR
4103		MOVX A, @DPTR	E0	Get the Data from 4200
4104		MOV B, A	F5 F0	Move data from A to B
4106		MUL AB	A4	Multiply A and B
4107		INC DPTR	A3	Increment DPTR
4108		MOVX @DPTR,A	F0	Store the result at 4201
4109		MOV A,B	E5 F0	Move B to A
410B		INC DPTR	A3	Increment DPTR
410C		MOVX @DPTR,A	F0	Store the result at 4202
410D	HERE	SJMP HERE	80 FE	Stop the program

CUBE

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
4100		MOV DPTR,#4200	90 42 00	Move 4200 to DPTR
4103		MOVX A, @DPTR	E0	Get the Data from 4200
4104		MOV B, A	F5 F0	Move data from A to B
4106		MOV R0,A	F8	
4107		MUL AB	A4	Multiply A and B
4108		MOV B,R0	88 F0	
410A		MUL AB	A4	Multiply A and B
410B		INC DPTR	A3	Increment DPTR
410C		MOVX @DPTR,A	F0	Store the result at 4201
410D		MOV A,B	E5 F0	Move B to A
410F		INC DPTR	A3	Increment DPTR
4110		MOVX @DPTR,A	F0	Store the result at 4202
4111	HERE	SJMP HERE	80 FE	Stop the program

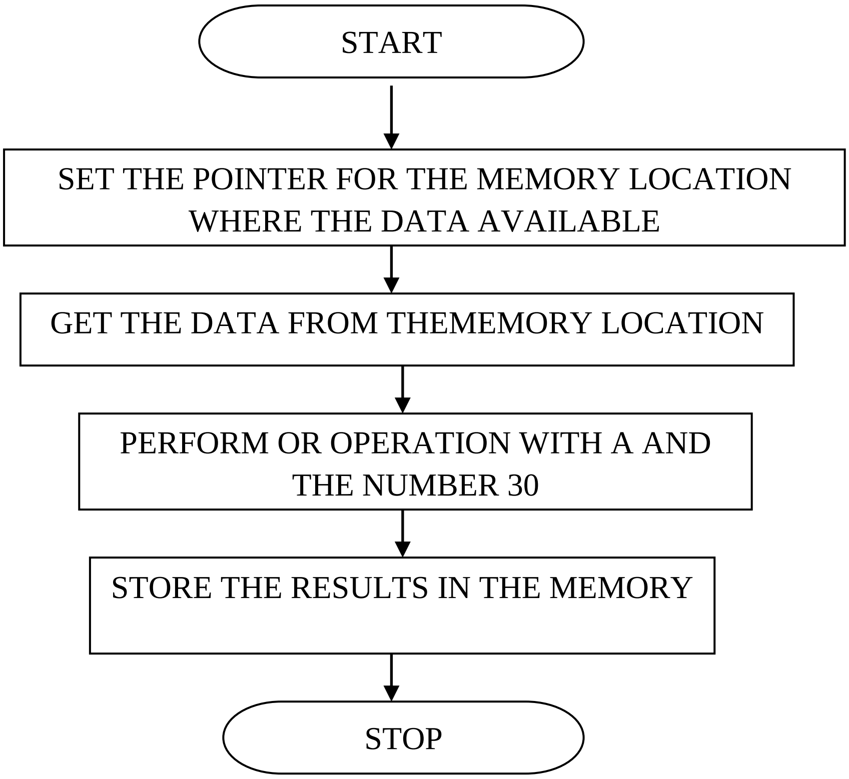
2’s COMPLEMENT

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
4100		MOV DPTR,#4200	90 42 00	Move 4200 to DPTR
4103		MOVX A, @DPTR	E0	Get the Data from 4200
4104		CPL A	F4	Complement A
4105		INC A	04	Increment A
4106		INC DPTR	A3	Increment DPTR
4107		MOVX @DPTR,A	F0	Store the result at 4201
4108	HERE	SJMP HERE	80 FE	Stop the program

RESULT:

Thus the assembly language program to find Square, Cube and 2’s complement of a number was executed successfully and the result was verified by using 8051 Microcontroller kit.

FLOWCHART:


START
SET THE POINTER FOR THE MEMORY LOCATION WHERE THE DATA AVAILABLE
GET THE DATA FROM THEMEMORY LOCATION
PERFORM OR OPERATION WITH A AND THE NUMBER 30
STORE THE RESULTS IN THE MEMORY
STOP

SAMPLE INPUT AND OUTPUT:

INPUT	
4200	**05**

OUTPUT	
4202	**35**

Ex. No 12B UNPACKED BCD TO ASCII USING 8051

AIM:

To write an assembly language program Unpacked BCD to ASCII code using 8051 Microcontroller kit.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8051 Microcontroller kit	1
2.	Power supply	1

ALGORITHM:

Step 1: Set the pointer for the memory location where the data available
Step 2: Load the data to accumulator
Step 3: Perform OR operation to Accumulator with Immediate Value 30.
Step 4: Store the result in next memory location.
Step 5: Stop the process.

PROGRAM

ADDRESS	LABLE	MNEMONICS	OPCODE	COMMENTS
4100		MOV DPTR,#4200	90 42 00	Move 4200 to DPTR
4103		MOVX A, @DPTR	E0	Get the Data from 4200
4104		ORL A,#30	44 30	Perform the OR A with 30
4106		INC DPTR	A3	Increment DPTR
4107		MOVX @DPTR,A	F0	Store the result at 4201
4108	HERE	SJMP HERE	80 FE	Stop the program

RESULT:

Thus the assembly language program to perform 8 bit multiplication was executed successfully and the result was verified by using 8051 Microcontroller kit.

AIM:

To write an assembly language program to display the digital clock by displaying the hours, minutes and seconds using 8086 kits.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microprocessor kit	1
2.	Power supply	1

PROGRAM:

ADDRESS	OP CODE	LABLE	MNENONICS	COMMENTS
1000	E8 75 00	START	CALL CONV	Call Convert function
1003	E8 62 00		CALL DISP	Call Display function
1006	C6 C0 B0		MOV AL,B0	Move B0 in AL
1009	E6 16		OUT 16,AL	Send to port 16
100B	C6 C1 07		MOV CL,07	Move 07 in CL
100E	C6 C0 88	YY	MOV AL,88	Move 88 in AL
1011	E6 14		OUT 14,AL	Send to port 14
1013	C6 C0 80		MOV AL,80	Move 80 in AL
1016	E6 14		OUT 14,AL	Send to port 14
1018	C6 C0 80	XX	MOV AL,80	Move 80 in AL
101B	E6 16		OUT 16,AL	Send to port 16
101D	90		NOP	No operation
101E	90		NOP	No operation
101F	90		NOP	No operation
1020	90		NOP	No operation
1021	E4 14		IN AL,14	Read data from port 14
1023	88 C2		MOV DL,AL	Move AL into DL
1025	E4 14		IN AL,14	Read data from port 14

1026	08 D0		OR AL, DL	OR AL with DL
1029	75 ED		JNZ XX	No zero jump to XX
102B	FE C9		DEC CL	Decrement CL
102D	75 DF		JNZ YY	No zero jump to YY
102F	C7 C6 00 15		MOV SI,1500	Move 1500 in SI
1033	8A 04		MOV AL,[SI]	Get data from SI
1035	FE C0		INC AL	Increment AL
1037	88 04		MOV [SI],AL	Store to SI
1039	80 F83C		CMP AL,3C	Compare AL with 3C
103C	75 C2		JNZ START	No zero jump to START
103E	C6 C0 00		MOV AL,00	Move 00 in AL
1041	88 04		MOV[SI],AL	Store to SI
1043	46		INC SI	Increment SI
1044	8A 04		MOV AL,[SI]	Get data from SI
1046	FE C0		INC AL	Increment AL
1048	88 04		MOV [SI],AL	Store to SI
104A	46		CMP AL,3C	Compare AL with 3C
104D	75 A0		JNZ START	No zero jump to START
104F	C6 C0 00		MOV AL,00	Move 00 in AL

SAMPLE INPUT:

1500	Seconds
1501	Minutes
1502	Hours

1052	88 04		MOV[SI],AL	Store to SI
1054	46		INC SI	Increment SI
1055	8A 04		MOV AL,[SI]	Get data from SI
1057	FE C0		INC AL	Increment AL
1059	88 04		MOV [SI],AL	Store to SI
105B	80 F8 18		CMP AL,18	Compare AL with 18
105E	75 A0		JNZ START	No zero jump to START
1060	C6 C0 00		MOV AL,00	Move 00 in to AL
1063	88 04		MOV[SI],AL	Store to SI
1065	E9 98 FF		JMP START	No zero jump to START
1068	C6 C4 06	DISP	MOV AH,06	Move 06 in to AH
106B	C7 C2 00 16		MOV DX,1600	Move 1600 in to FX
106F	C6 C501		MOV CH,01	Move 01 in to CH
1072	C6 C1 00		MOV CL,00	Move 00 in to CL
1075	CD 05		INT 5	Interrupt 5
1077	C3		RET	Return
1078	C7 C6 00 15	CONV	MOV SI,1500	Move 1500 in to SI
107C	C7 C3 08 16		MOV BX,1608	Move 1608 in to BX
1080	C6 C0 24		MOV AL,24	Move 24 in to AL
1083	88 07		MOV [BX],AL	Store AL to BX location
1085	8A 04	SEC	MOV AL,[SI]	Get data from SI
1087	C6 C4 00		MOV AH,00	Move 00 in to AH
108A	C6 C6 0A		MOV DH,0A	Move 0A in to DH
108D	F6 F6		DIV DH	Divide AX/DH
108F	80 C4 30		ADD AH,30	Add AH with 30
1092	4B		DEC BX	Decrement BX
1093	88 27		MOV [BX],AH	Store AH to BX location
1095	4B		DEC BX	Decrement BX

1096	80 C0 30		ADD AL,30	Add AL with 30
1099	88 07		MOV [BX],AL	Store AL to BX location
109B	4B		DEC BX	Decrement BX
109C	C6 C0 3A		MOV AL,3A	Move 3A in to AL
109F	88 07		MOV [BX],AL	Store AL to BX location
10A1	4B		DEC BX	Decrement BX
10A2	46	MIN	INC SI	Increment SI
10A3	8A 04		MOV AL,[SI]	Load AL from SI location
10A5	C6 C4 00		MOV AH,00	Move 00 in to AH
10A8	C6 C6 0A		MOV DH,0A	Move 0A in to DH
10AB	F6 F6		DIV DH	Divide AX/DH
10AD	80 C4 30		ADD AH,30	Add AH with 30
10B0	88 27		MOV [BX],AH	Store AH to BX location
10B2	4B		DEC BX	Decrement BX
10B3	80 C0 30		ADD AL,30	Add AL with 30
10B6	88 07		MOV [BX],AL	Store AL to BX location
10B8	4B		DEC BX	Decrement BX
10B9	C6 C0 3A		MOV AL,3A	Move 3A in to AL
10BC	88 07		MOV [BX],AL	Store AL to BX location
10BE	4B		DEC BX	Decrement BX
10BF	46	HOUR	INC SI	Increment SI
10C0	8A 04		MOV AL,[SI]	Store AL to SI location
10C2	C6 C4 00		MOV AH,00	Move 00 in to AH
10C5	C6 C6 0A		MOV DH,0A	Move 0A in to DH
10C8	F6 F6		DIV DH	Divide AX/DH

10CA	80 C4 30		ADD AH,30	Add AH with 30
10CD	88 27		MOV [BX],AH	Store AH to BX location
10CF	4B		DEC BX	Decrement BX
10D0	80 C0 30		ADD AL,30	Add AL with 30
10D3	88 07		MOV [BX],AL	Store AL to BX location
10D5	C3		RET	Return
10D6	C3		RET	Return
10D7	E4 02	ZZ	IN AL,02	Read Port 02
10D9	80 E0 FF		AND AL,0FF	And AL with FF
10DC	80 F8 F0		CMP AL,0F0	Compare AL with F0
10DF	75 F6		JNE ZZ	No zero jump to ZZ

RESULT:

Thus the assembly language program to digital clock by displaying the hours, minutes and seconds using 8086 was executed and result was verified.

Ex. No 14 PASSWORD CHECKING

AIM:

To write an assembly language program to verify the password using MASM.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microcontroller kit	1
2.	Power supply	
3.	PC with MASM	1

PROGRAM

```
ASSUME CS:CODE,DS:DATA,ES:EXTRA

DATA SEGMENT

        STRING1 DB 'MRCET'

        STRLEN EQU ($-STRING1)

        SANE DB 'STRINGS ARE UNEQUAL$'

        SAE DB 'STRINGS ARE EQUAL$'

DATA ENDS

EXTRA SEGMENT

        STRING2 DB 'MRCET'

EXTRA ENDS

CODE SEGMENT

START:  MOV AX,DATA

                MOV DS,AX

                MOV AX,EXTRA

                MOV ES,AX

                MOV SI,OFFSET STRING1

                MOV DI,OFFSET STRING2

                CLD
```

```
            MOV CX,STRLEN

            REP CMPSB

            JZ GO

            MOV AH,09H

            MOV DX,OFFSET SANE

            INT 21H

            JMP EXITP
GO:         MOV AH,09H

            MOV DX,OFFSET SAE

            INT 21H

EXITP:  INT 03H

CODE ENDS

END START
```

RESULT:

Thus the assembly language program to 8086 processor to verify the password was executed successfully and the result was verified.

Ex. No 15 PRINTER STATUS

AIM:

To write an assembly language program in 8086 processor to display the status of Printer using MASM.

APPARATUS REQUIRED:

S. No	Apparatus	Qty
1.	8086 Microcontroller kit	1
2.	Power supply	
3.	PC with MASM	1

PROGRAM

```
name printmsg
        page 60,80
title program to send a message to printer
        .model small
        .stack 64
        .data
msg db 'If this is Printed on paper',0dh,0ah
    db 'Then Program is Working',0dh,0ah
len equ $-msg
errmsg db 'Error! Printer is not connected or switched off',0dh,0ah,'$'
        .code
main:
        mov ax,@data
        mov ds,ax
        mov ah,02h  ;get printer status
        mov dx,0    ;printer 0
        int 17h     ;returns with ah=status
        rol ah,01   ;if ah7=1 then printer is ready | mov ah7 to carry flag
        jc online
offline:
        lea dx,errmsg
        mov ah,09h    ;displays errmsg
        int 21h
        jmp exit
online:
        mov cx,len
        mov si,00h
        mov ah,05h   ;prints the char in dl on printer
again:
        mov dl,msg[si]
        int 21h
        inc si
        loop again   ;dec cx,until cx=0
exit:
        mov ah,4ch
        int 21h
end main
```

RESULT:

Thus the assembly language program in 8086 processor to print Printer status was executed successfully and the result was verified.