ST.ANNE'S

COLLEGE OF ENGINEERING AND TECHNOLOGY

(Approved by AICTE, New Delhi. Affiliated to Anna University, Chennai)
(An ISO 9001: 2015 Certified Institution)
ANGUCHETTYPALAYAM, PANRUTI – 607 106.

.

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL

AUGUST 2022– NOV 2022 / ODD SEMESTER

SUBJECT CODE/NAME: EC 8681/Microprocessor and Microcontroller Laboratory

YEAR/SEM: III/V **BATCH**: 2020- 2024

AS PER ANNA UNIVERSITY, CHENNAI REGULATION 2017

LIST OF EXPERIMENTS

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 controller
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

INDEX

Ex.No.	Date	Title	Marks	Staff Sign.

Ex.No.	Date	Title	Marks	Staff Sign.

EX.NO.1. PROGRAMS FOR BASIC ARITHMETIC AND LOGICAL OPERATIONS (USING 8086)

AIM:

To write an assembly language program to perform arithmetic operations using 8086 Microprocessor.

ALGORITHM:-

a) Addition:-

(i) Start the process
(ii) Initialize the count value
(iii) Get the two data.
(iv) Add the two data values
(v) If carry exists increment the count value.
(vi) Store the result.
(vii) Stop the process.

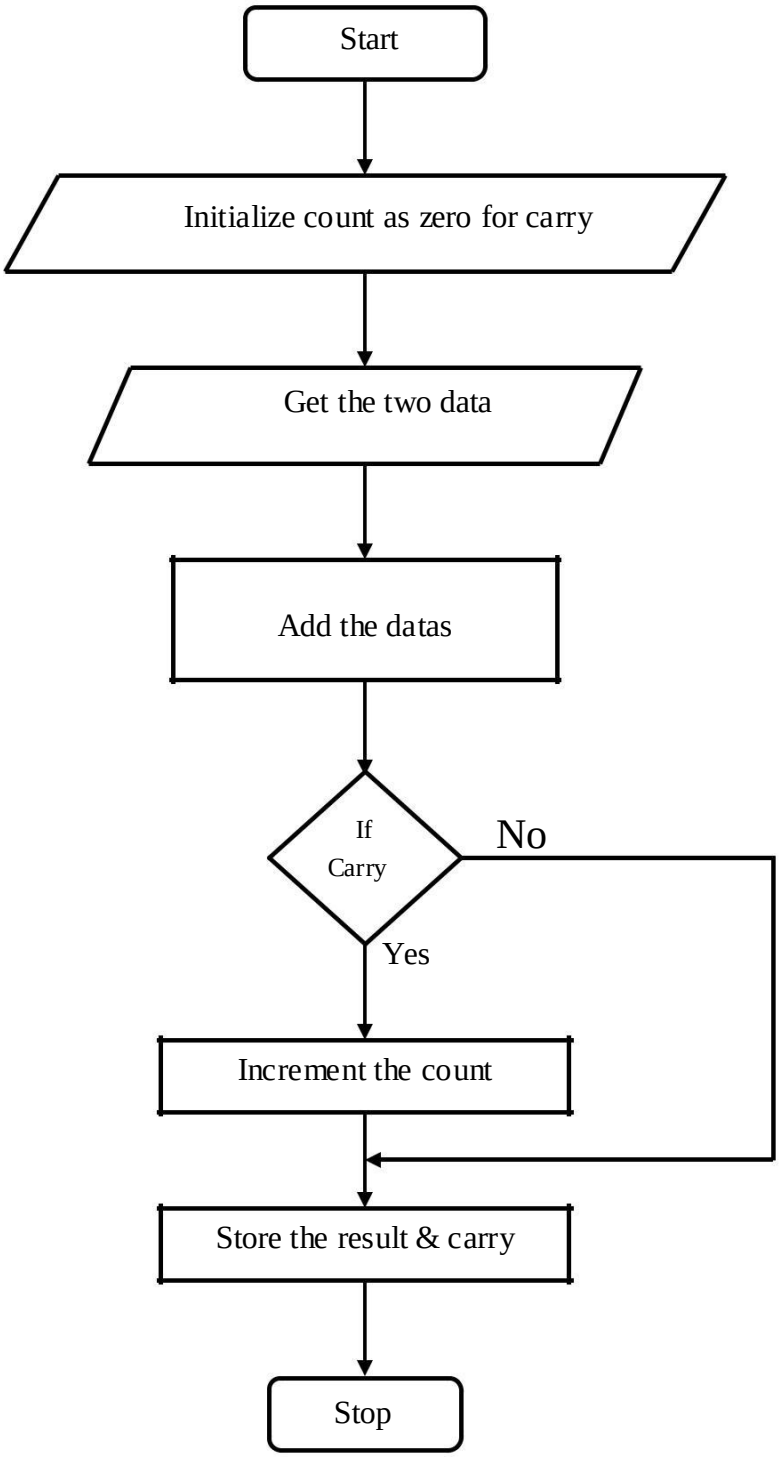
PROGRAM

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
	1000	MOV	CL , 00	C6, C1, 00	; Initialize the count
	1003	MOV	AX, 0F0C	C7, C0, 0C, 0F	; Move1st data to accumulator
	1007	MOV	BX, 111F	C7, C3, 1F, 11	; Move2nd data to register
	100B	ADD	BX, AX	01, C3	; Add the two data
	100D	JNC	LOOP1	73, 02	; Jump on no carry
	100F	INC	CL	FE, C1	; Increment the counter
LOOP1:	1011	MOV	[1100], BX	89,1E, 00, 11	; Store the result
	1015	MOV	[1102], CL	88, 0E, 02,11	; Store the carry
	1019	HLT		F4	; Stop the process

OUTPUT
16 – BIT ADDITION

Address	Output
1100	**2B**
1101	**20**
1102	**00**

16 BIT ADDITION



1 B) 16 BIT SUBTRACTION

ALGORITHM:-

(i) Start the process

(ii) Initialize the count value

(iii) Get the two data and subtract it.

(iv) If carry exists, get the 2's complement of the value.

(v) Store the result and carry value.

(vi) Stop the process.

PROGRAM

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
	1000	MOV	CL, 00	C6, C1, 00	; Initialize the count
	1003	MOV	AX, [1100]	8B, 06, 00, 11	; Move1st data to accumulator
	1007	MOV	BX, [1102]	8B, 1E, 02, 11	; Move 2nd data to 'B' register
	100B	SUB	BX, AX	29, C3	; Subtract the two datas
	100D	JNC	LOOP1	73, 05	; Jump on no carry
	100F	INC	CL	FE, C1	; Increment the counter
	1011	NOT	BX	F7, D3	; Get the complement value
	1013	INC	BX	43	; Increment the value
LOOP1:	1014	MOV	[1104], BX	89, 1E, 04, 11	; Store the result
	1018	MOV	[1106], CL	88, 0E, 06, 11	; Store the carry
	101C	HLT		F4	; Stop the process

OUTPUT

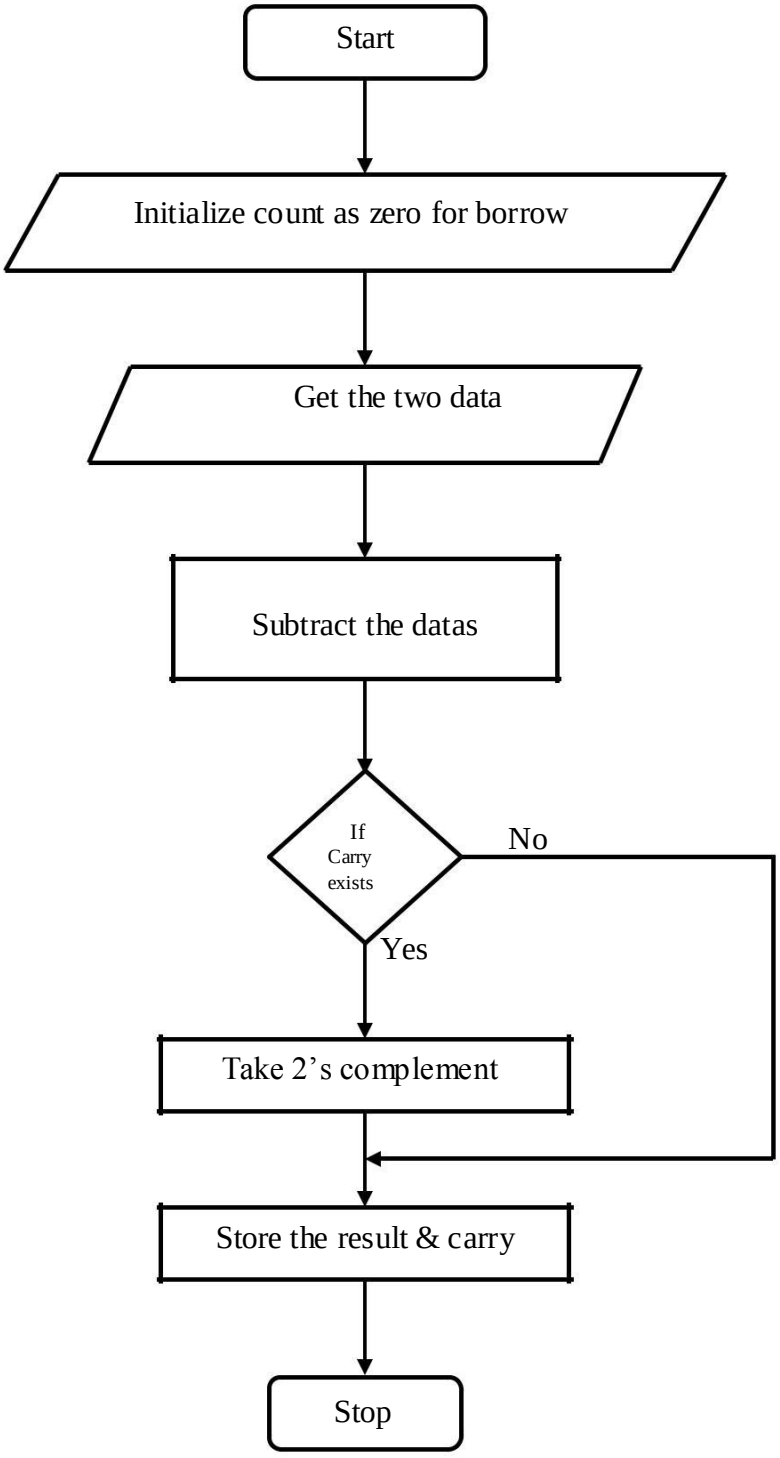
16 – BIT SUBTRACTION

Address	Input
1100	**76**
1101	**86**
1102	**45**
1103	**81**

Address	Output
1104	**31**
1105	**65**
1106	**00**

FLOWCHART:-

Subtraction:-



1.C) 16 BIT MULTIPLICATION

ALGORITHM:-

(i) Start the process

(ii) Get the two values

(iii) Multiply the two values.

(iv) Store the result and overflow

(v) Stop the process.

PROGRAM

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
	1000	MOV	SI, 1100	C7, C6,00, 11	; Move the source index
	1004	MOV	AX, [SI]	8B, 04	value
	1006	MOV	BX, [SI + 02]	8B, 54, 02	; Move the first data
	1009	MUL	BX	F7, E3	; Get the second data
	100B	MOV	[SI + 04],	89, 44, 04	; Multiply the data
	100E	MOV	AX	89, 54, 0b	; Store the result
	1011	HLT	[SI + 06],	F4	; Store the over flow
			DX		; Stop the process

INPUT

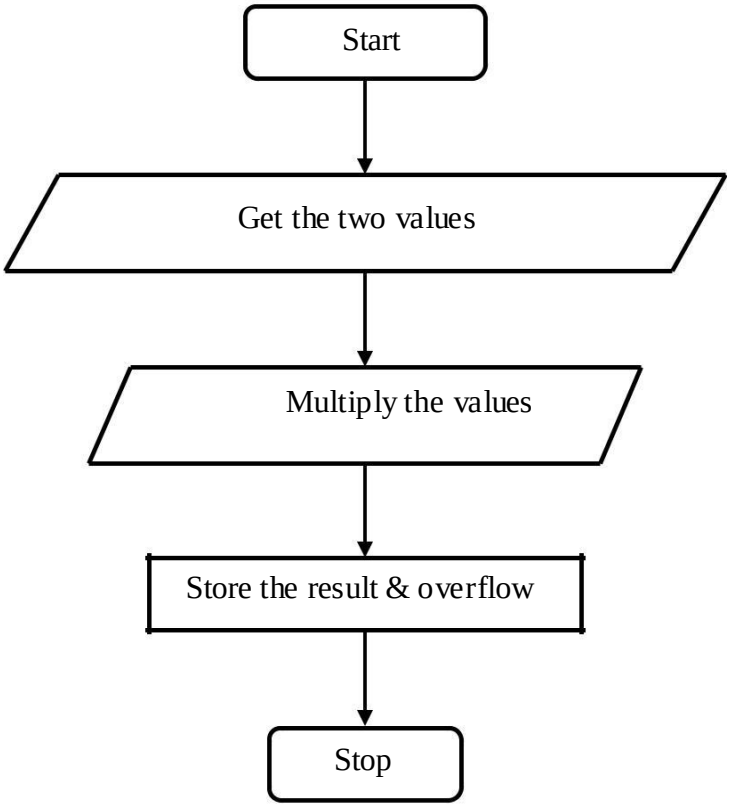
Address	Input
1100	**11**
1101	**11**
1102	**00**
1103	**11**

OUTPUT

Address	Output
1104	**00**
1105	**21**
1106	**22**
1107	**01**

FLOW CHART:-

Multiplication:-



D) 16 BIT DIVISION

AIM:

To perform division of a 32 bit number by a 16 bit number and store the quotient and remainder in memory

ALGORITHM:-

(i) Start the process

(ii) Get the two values

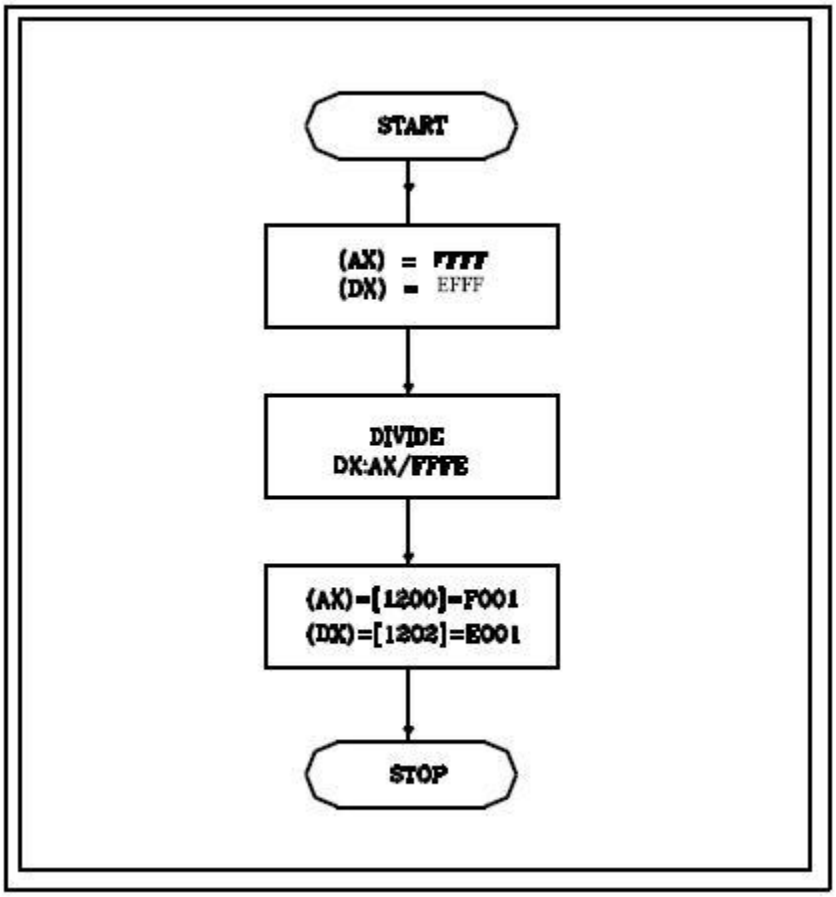
(iii) Initialize 'DX' register as zero

(iv) Divide the values

(v) Store the quotient and remainder

(vi) Stop the process.

FLOWCHART:



D) 16 BIT DIVISION

PROGRAM

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
	1000	MOV	SI, 1100	C7, C6,00, 11	; Get the source index value
	1004	MOV	Ax, [SI]	8B, 04	; Get the first data
	1006	MOV	DX, [SI +	8B, 54, 02	; Initialize 'DX' register
	1009	MOV	02]	8B, 5C, 04	value
	100C	DIV	BX, [SI + 04]	F7, E3	; Get the dividend value
	100E	MOV	BX	89, 44, 06	; Divide the value
	1011	MOV	[SI + 06],	89, 54, 08	; Move the quotient
	1014	HLT	AX	F4	; Move the remainder & store
			[SI + 08],		; Stop the process
			DX		

16 – BIT DIVISIION

Address	Input
1100	42 (DIVIDEND)
1101	24
1102	00
1103	00
1104	02(DIVISOR)
1105	00

Address	Output
1106	21(QUOTIENT)
1107	12
1108	00(REMAINDER)
1109	00

RESULT:-

Thus the assembly language program for 16 Bit Arithmetic and Logical operations has been done and verified.

VIVA QUESTIONS AND ANSWERS

1. What is a Microprocessor?

Microprocessor is a CPU fabricated on a single chip, program-controlled device, which fetches the instructions from memory, decodes and executes the instructions.

2. What is Instruction Set?

It is the set of the instructions that the Microprocessor can execute.

3. What is Clock Speed?

Clock speed is measured in the MHz and it determines that how many instructions a processor can processed. The speed of the microprocessor is measured in the MHz or GHz.

4. What are the features of Intel 8086?

Features:

Released by Intel in 1978
Produced from 1978 to 1990s
A 16-bit microprocessor chip.
Max. CPU clock rate:5 MHz to 10 MHz
Instruction set: x86-16

5. What are the flags in 8086?

In 8086 carry flag, Parity flag, Auxiliary carry flag, Zero flag, Overflow flag, Trace flag, Interrupt flag, Direction flag, and Sign flag.

6. **What is assembly language?**

The language in which the mnemonics (short -hand form of instructions) are used to write a program is called assembly language. The manufacturers of microprocessor give the mnemonics.

7. What are machine language and assembly language programs?

The software developed using 1's and 0's are called machine language, programs. The software developed using mnemonics are called assembly language programs.

8. What is the drawback in machine language and assembly language, programs?

The machine language and assembly language programs are machine dependent. The programs developed using these languages for a particular machine cannot be directly run on another machine.

9. Define bit, byte and word.

A digit of the binary number or code is called bit. Also, the bit is the fundamental storage unit of computer memory.

The 8-bit (8-digit) binary number or code is called byte and 16-bit binary number or code is called word. (Some microprocessor manufactures refer the basic data size operated by the processor as word).

10. What is a bus?

Bus is a group of conducting lines that carries data, address and control signals.

2. PROGRAM FOR SEARCHING AND SORTING OF AN ARRAY USING 8086

2a. SORTING AN ARRAY IN ASCENDING ORDER

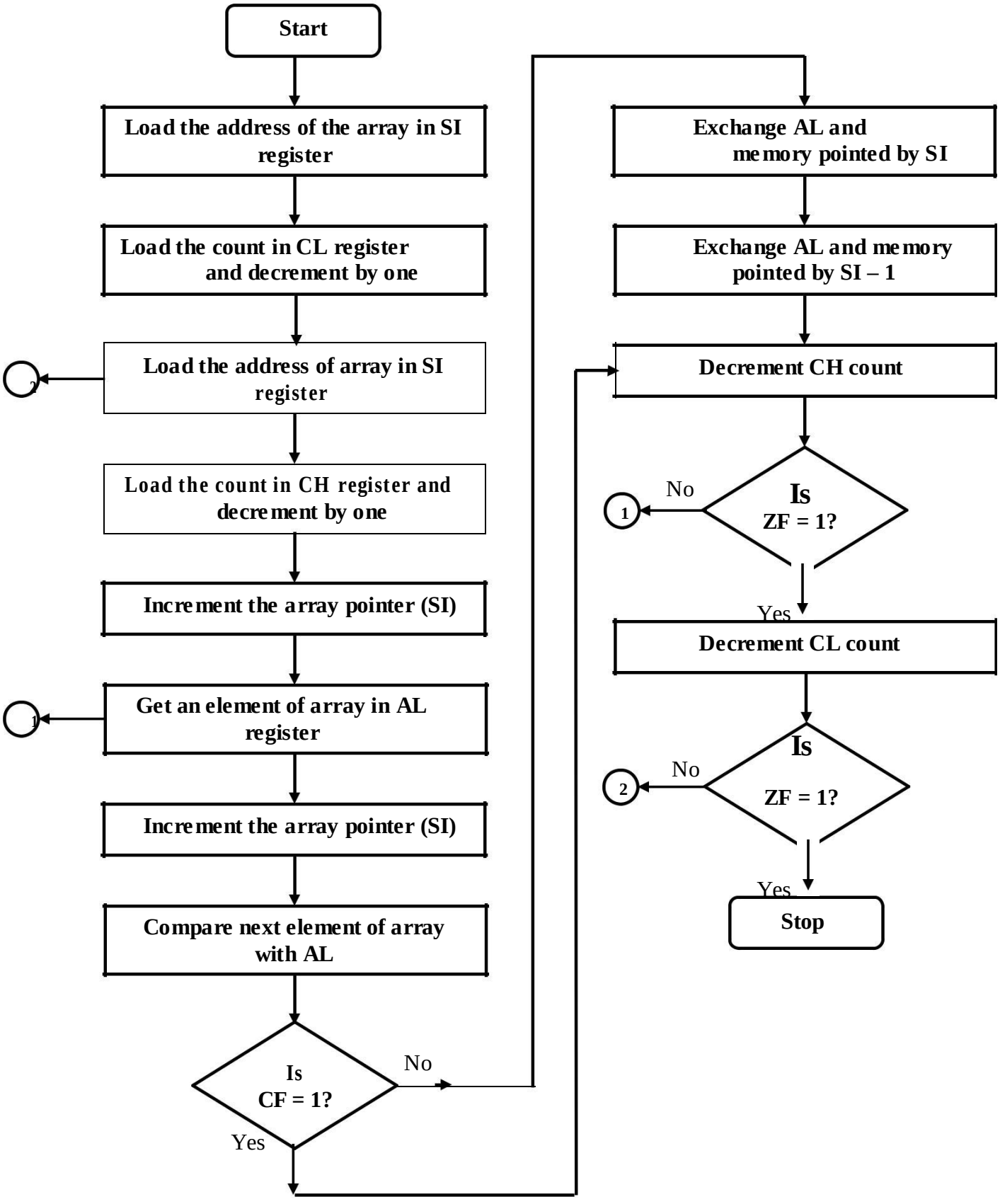
AIM:-

Write an assembly language program to sort an array of data in ascending order.

ALGORITHM:-

1. Set SI register as pointer for array.
2. Set CL register as count for N – 1 repetitions.
3. Initialize array pointer.
4. Set CH as count for N – 1 comparisons.
5. Increment the array pointer.
6. Get an element of array AL register.
7. Increment the array pointer.
8. Compare the next element of the array with AL.
9. Checks carry flag. If carry flag is set then go to step -12, otherwise go to next step.
10. Exchange the content of memory pointed by SI and the content of previous memory location
11. Decrement the count for comparisons (CH register).
12. Check zero flag. If zero flag is reset then go to step-6, otherwise go to next step.
13. Decrement the count for repetitions (CL register).
14. Check zero flag. If zero flag is reset then go to step-3, otherwise go to next step.
15. Stop.

SORTING IN ASCENDING ORDER



PROGRAM

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	1000	MOV	SI, 1100H	C7 C6 00 11	; Set SI register as pointer for array
	1004	MOV	CL, [SI}	8A 0C	; Set CL as count for N – 1 repetitions
	1006	DEC	CL	FE C9	
REPEAT	1008	MOV	SI, 1100H	C7 C6 00 11	; Initialize pointer
	100C	MOV	CH, [SI]	8A 2C	: Set CH as count for N – 1 comparisons
	100E	DEC	CH	FE CD	
	1010	INC	SI	46	; Decrement the count
REPCOM	1011	MOV	AL, [SI]	8A 04	; Get an element of array in AL register
	1013	INC	SI	46	
	1014	CMP	AL, [SI]	3A 04	; Compare with next element of array ; in memory
	1016	JC	AHEAD	72 05	; It AL register is lesser than memory, ‘then go to AHEAD
	1018	XCHG	AL, [SI]	86 04	; If AL is less than memory then ; exchange
	101A	XCHG	AL, [SI –1]	86 44 FF	; the content of memory pointed by ; SI and the previous memory location
AHEAD	101D	DEC	CH	FE CD	; Decrement the count for comparisons
	101F	JNZ	REPCOM	75 F0	; Repeat comparisons until CH count is ; zero
	1021	DEC	CL	FE C9	; Decrement the count for repetitions
	1023	JNZ	REPEAT	75 E3	; Repeat N – 1 comparisons until CL count is zero
	1025	HLT		F4	

Address	Input
1100	05 – count
1101	09
1102	49
1103	24
1104	32
1105	64

Address	Output
1100	05 – count
1101	09
1102	24
1103	32
1104	49
1105	64

RESULT:

Thus the assembly language program to sort an array of data in ascending order using 8086 has been done and verify successfully.

b. SORTING AN ARRAY IN DESCENDING ORDER

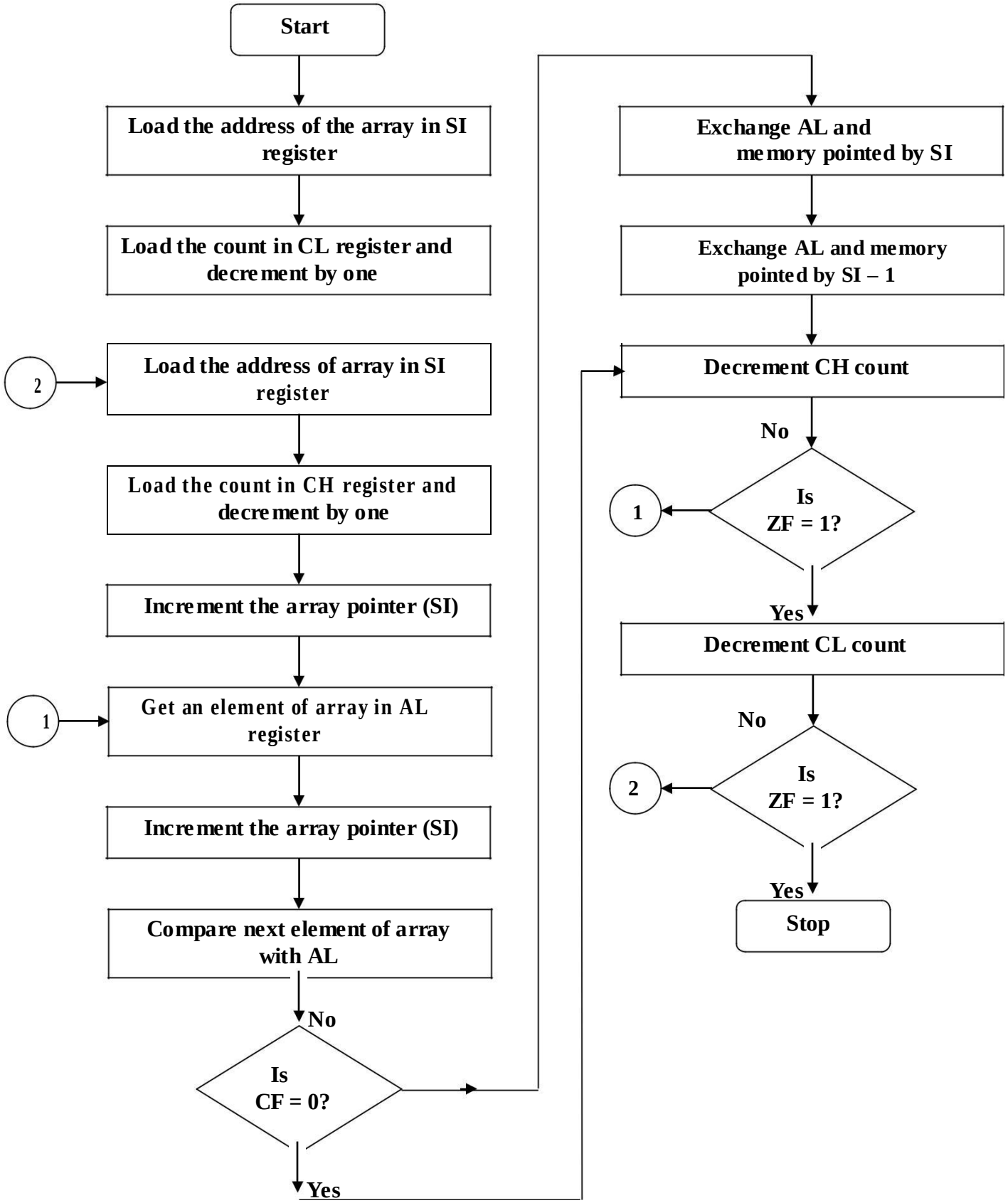
AIM:-

Write an assembly language program to sort an array of data in descending order.

ALGORITHM:-

1. Set SI register as pointer for array.
2. Set CL register as count for N – 1 repetitions.
3. Initialize array pointer.
4. Set CH as count for N – 1 comparisons.
5. Increment the array pointer.
6. Get an element of array AL register.
7. Increment the array pointer.
8. Compare the next element of the array with AL.
9. Checks carry flag. If carry flag is set then go to step -12, otherwise go to next step.
10. Exchange the content of memory pointed by SI and the content of previous memory location (For this, exchange AL and memory pointed by SI, and then exchange AL and memory pointed SI – I).
11. Decrement the count for comparisons (CH register).
12. Check zero flag. If zero flag is reset then go to step-6, otherwise go to next step.
13. Decrement the count for repetitions' (CL register).
14. Check zero flag. If zero flag is reset then go to step-3, otherwise go to next step.
15. Stop.

SORTING IN DESCENDING ORDER



PROGRAM

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	1000	MOV	SI, 1100H	C7 C6 00 11	; Set SI register as pointer for array
	1004	MOV	CL, [SI}	8A 0C	; Set CL as count for N – 1 repetitions
	1006	DEC	CL	FE C9	
REPEAT	1008	MOV	SI, 1100H	C7 C6 00 11	; Initialize pointer
	100C	MOV	CH, [SI]	8A 2C	: Set CH as count for N – 1 comparisons
	100E	DEC	CH	FE CD	
	1010	INC	SI	46	; Increment the count
REPCOM	1011	MOV	AL, [SI]	8A 04	; Get an element of array in AL register
	1013	INC	SI	46	
	1014	CMP	AL, [SI]	3A 04	; Compare with next element of array ; in memory
	1016	JNC	AHEAD	73 05	; It AL is greater than memory, then go ; to AHEAD
	1018	XCHG	AL, [SI]	86 04	; If AL is less than memory then
	101A	XCHG	AL, [SI –1]	86 44 FF	; exchange the content of memory ; pointed by SI and the previous memory ; location
AHEAD	101D	DEC	CH	FE CD	; Decrement the count for comparisons
	101F	JNZ	REPCOM	75 F0	; Repeat comparisons until CH count is zero
	1021	DEC	CL	FE C9	; Decrement the count for repetitions
	1023	JNZ	REPEAT	75 E3	; Repeat N – 1 comparisons until CL count is zero
	1025	HLT		F4	

Address	Input
1100	05 – count
1101	09
1102	49
1103	24
1104	32
1105	64

Address	Output
1100	05 – count
1101	64
1102	49
1103	32
1104	24
1105	09

RESULT:

Thus the assembly language program to sort an array of data in descending order using 8086 has been done and verify successfully.

2 C . SEARCHING FOR SMALLEST NUMBER IN AN ARRAY

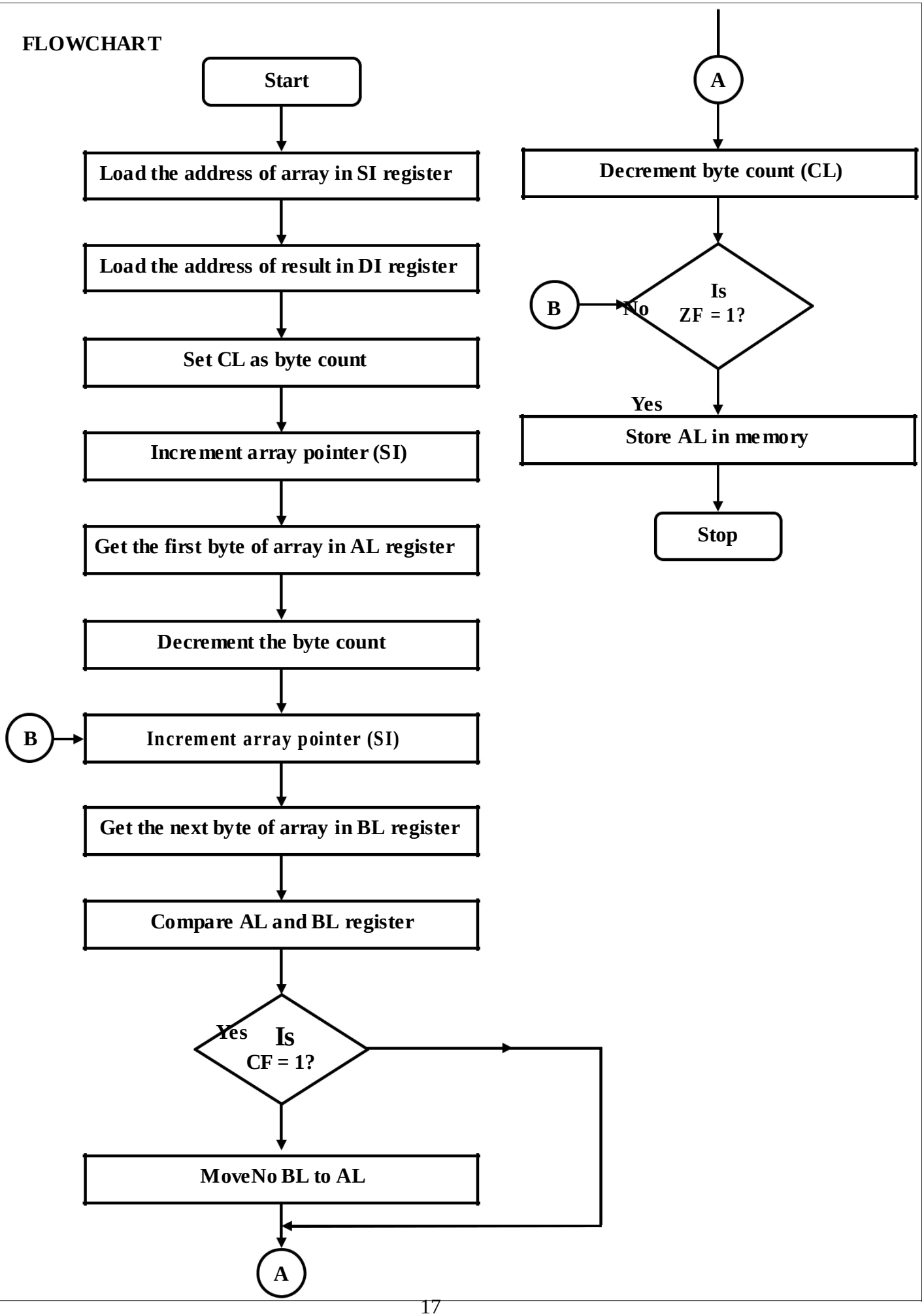
AIM:-

Write an assembly language program to search the smallest data in an array.

ALGORITHM:

1. Load the staring address of the array in SI register.
2. Load the address of the result in DI register.
3. Load the number of bytes in the array in CL register.
4. Increment the array pointer (SI register).
5. Get the first byte of the array in AL register
6. Decrement the byte count (CL register).
7. Increment the array pointer (SI register).
8. Get next byte of the array in BL register.
9. Compare current smallest (AL) and next byte (BL) if the array.
10. Check carry flag. If carry flag is set then go to step -12, otherwise go to next step.
11. Move BL to AL.
12. Decrement the byte count (CL register).
13. Check zero flag. If zero flag is reset then go to step-7, otherwise go to next step.
14. Save the smallest data in memory pointed by DI.
15. Stop.

FLOWCHART


Start
Load the address of array in SI register
Load the address of result in DI register
Set CL as byte count
Increment array pointer (SI)
Get the first byte of array in AL register
Decrement the byte count
B
Increment array pointer (SI)
Get the next byte of array in BL register
Compare AL and BL register
Yes
Is
CF = 1?
MoveNo BL to AL
A
A
Decrement byte count (CL)
B
No
Is
ZF = 1?
Yes
Store AL in memory
Stop

PROGRAM

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START	1000	MOV	SI, 1100H	C7 C6 00 11	; Set SI register as pointer for array
	1004	MOV	DI, 1200H	C7 C7 00 12	; Set DI register as pointer for result
	1008	MOV	CL, [SI]	8A 0C	; Set CL as count for elements in the array
	100A	INC	SI	46	; Increment the address pointer
	100B	MOV	AL, [SI]	8A 04	; Set first data as smallest
	100D	DEC	CL	FE C9	; Decrement the count
AGAIN	100F	INC	SI	46	; Make SI to point to next data in array
	1010	MOV	BL, [SI]	8A 1C	; Get the next data in BL register
	1012	CMP	AL, BL	38 D8	; Compare current smallest data in AL ; with BL
	1014	JC	AHEAD	72 02	; If carry is set then AL is less than BL ; hence proceed to AHEAD
	1016	MOV	AL, BL	88 D8	; If carry is not set then make BL as ; current smallest
AHEAD	1018	DEC	CL	FE C9	; Decrement the count
	101A	JNZ	AGAIN	75 F3	; If count is not zero repeat search
	101C	MOV	[DI], AL	88 05	; Store the smallest data in memory
	101E	HLT		F4	

Smallest no in the array

Address	**Input**
1100	**(**05) count
1101	22
1102	AA
1103	FF
1104	45
1105	50
Address	**Output**
200	22

RESULT:

Thus the assembly language program for smallest data in an array using 8086 has been done and verify successfully.

2D). SEARCHING FOR LARGEST NUMBER IN AN ARRAY

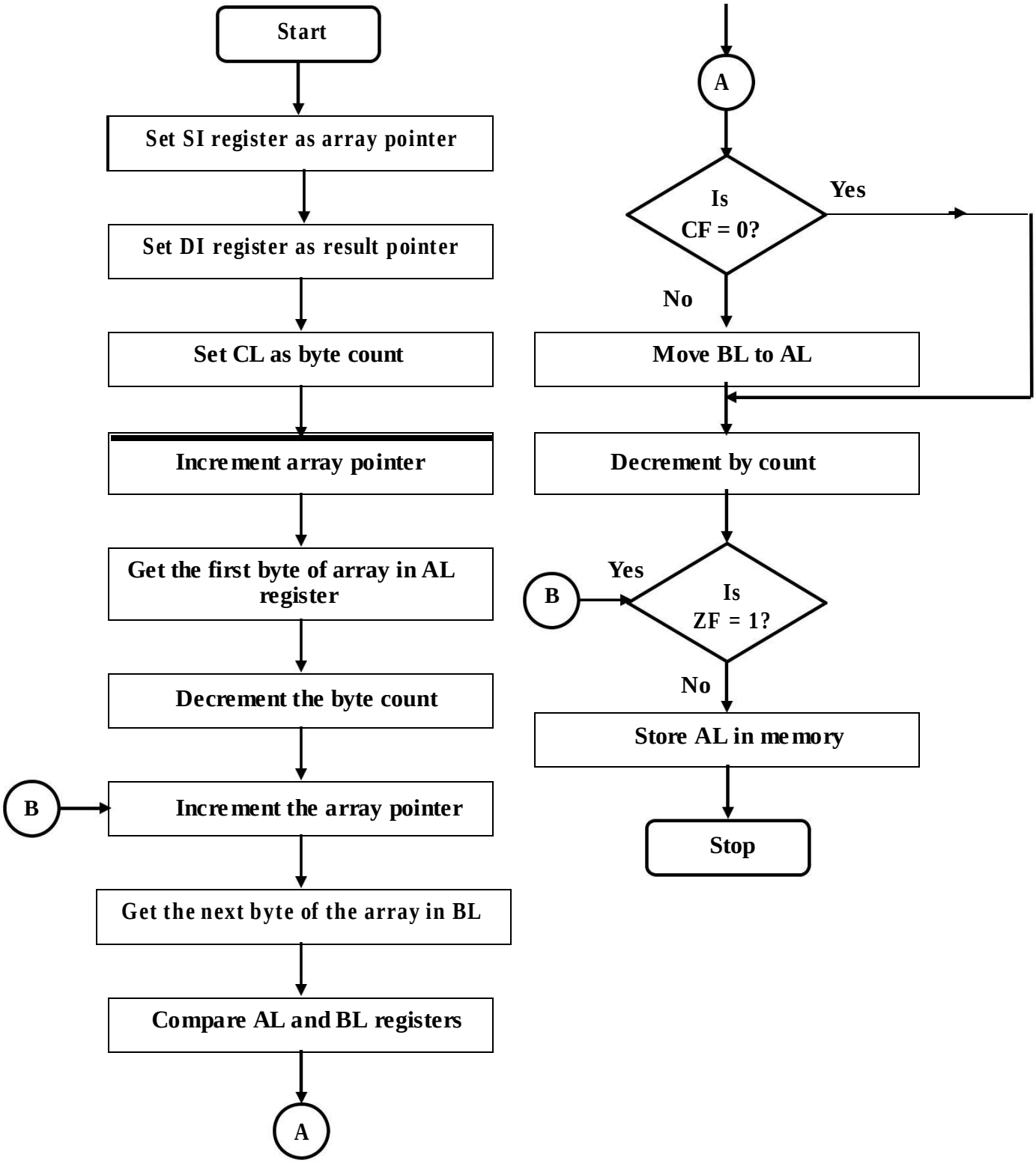
AIM:-

Write an assembly language program to search the largest data in an array.

ALGORITHM:

1. Load the staring address of the array in SI register.
2. Load the address of the result in DI register.
3. Load the number of bytes in the array in CL register.
4. Increment the array pointer (SI register).
5. Get the first byte of the array in AL register
6. Decrement the byte count (CL register).
7. Increment the array pointer (SI register).
8. Get next byte of the array in BL register.
9. Compare current smallest (AL) and next byte (BL) if the array.
10. Checks carry flag. If carry flag is set then go to step -12, otherwise go to next step.
11. Move BL to AL.
12. Decrement the byte count (CL register).
13. Check zero flag. If zero flag is reset then go to step-7, otherwise go to next step.
14. Save the largest data in memory pointed by DI.
15. Stop.

FLOWCHART:


Start
Set SI register as array pointer
Set DI register as result pointer
Set CL as byte count
Increment array pointer
Get the first byte of array in AL register
Decrement the byte count
B
Increment the array pointer
Get the next byte of the array in BL
Compare AL and BL registers
A
A
Is CF = 0?
Yes
No
Move BL to AL
Decrement by count
B
Yes
Is ZF = 1?
No
Store AL in memory
Stop

PROGRAM

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START	1000	MOV	SI, 1100H	C7 C6 00 11	; Set SI register as pointer for array
	1004	MOV	DI, 1200H	C7 C7 00 12	; Set DI register as pointer for result
	1008	MOV	CL, [SI]	8A 0C	; Set CL as count for elements in the ; array
	100A	INC	SI	46	; Increment the address pointer
	100B	MOV	AL, [SI]	8A 04	; Set first data as smallest
AGAIN	100D	DEC	CL	FE C9	; Decrement the count
	100F	INC	SI	46	; Make SI to point to next data in array
	1010	MOV	BL,[SI]	8A 1C	; Get the next data in BL register
	1012	CMP	AL, BL	38 D8	; Compare current smallest data in AL ; with BL
	1014	JNC	AHEAD	73 02	; If carry is set then AL is less than BL ; hence proceed to AHEAD
	1016	MOV	AL, BL	88 D8	; If carry is not set then make BL as
AHEAD					; current largest
	1018	DEC	CL	FE C9	; Decrement the count
	101A	JNZ	AGAIN	75 F3	; If count is not zero repeat search
	101C	MOV	[DI], AL	88 05	; Store the smallest data in memory
	101E	HLT		F4	

Largest

Address	**Input**
1100	**05 – count**
1101	**22**
1102	**AA**
1103	**FF**
1104	**45**
1105	**50**
Address	**Output**
1200	**FF**

RESULT:

Thus the assembly language program for largest data in an array using 8086 has been done and verify successfully.

VIVA QUESTIONS AND ANSWERS

1. What are the different types of Addressing Modes?

The different types of Addressing Modes are
Immediate, Direct, Register, Register Indirect, Indexed, Register Relative addressing modes

2.What are Data Copy/Transfer Instructions?

A:- Mov, Push, Pop, Xchg, In, Out, Xlat, Lea, Lds/Les, Lahf, Sahf, Pushf, Popf

3. What are Machine Control Instructions?

A:- Nop, Hlt, Wait, Lock

4) What are Flag Manipulation Instructions?

A:- Cld, Std, Cli, Sti

5) What are String Instructions?

A:- Rep, MovSB/MovSW, Cmps, Scas, Lods, Stos

6. Why data bus is bi-directional?

The microprocessor has to fetch (read) the data from memory or input device for processing and after processing, it has to store (write) the data to memory or output device. Hence the data bus is bi-directional.

7. Why address bus is unidirectional?

The address is an identification number used by the microprocessor to identify or access a memory location or I / O device. It is an output signal from the processor. Hence the address bus is unidirectional.

8. What is the function of microprocessor in a system?

The microprocessor is the master in the system, which controls all the activity of the system. It issues address and control signals and fetches the instruction and data from memory. Then it executes the instruction to take appropriate action.

3. PROGRAM FOR STRING MANIPULATION OPERATIONS USING 8086

AIM:-

To write a program for string manipulation such as fill a byte, move a string; compare the string by using 8086 microprocessor kit.

ALOGRITHM:

a) Move the string:

Step1: Start the process

Step2: Initialize the memory

Step3: Clear the direction flag

Step4: Move the value to string

Step5: Stop the process

b) Compare the string: Step1:

Start the process

Step2: Initialize the counter and carry value

Step3: Initialize the memory value

Step4: Compare two values

Step5: If the two value are equal set the carry otherwise reset

Step6: Stop the process

c) Fill a Byte:

Step1: Start the process

Step2: Clear the direction flag

Step3: Initialize the counter

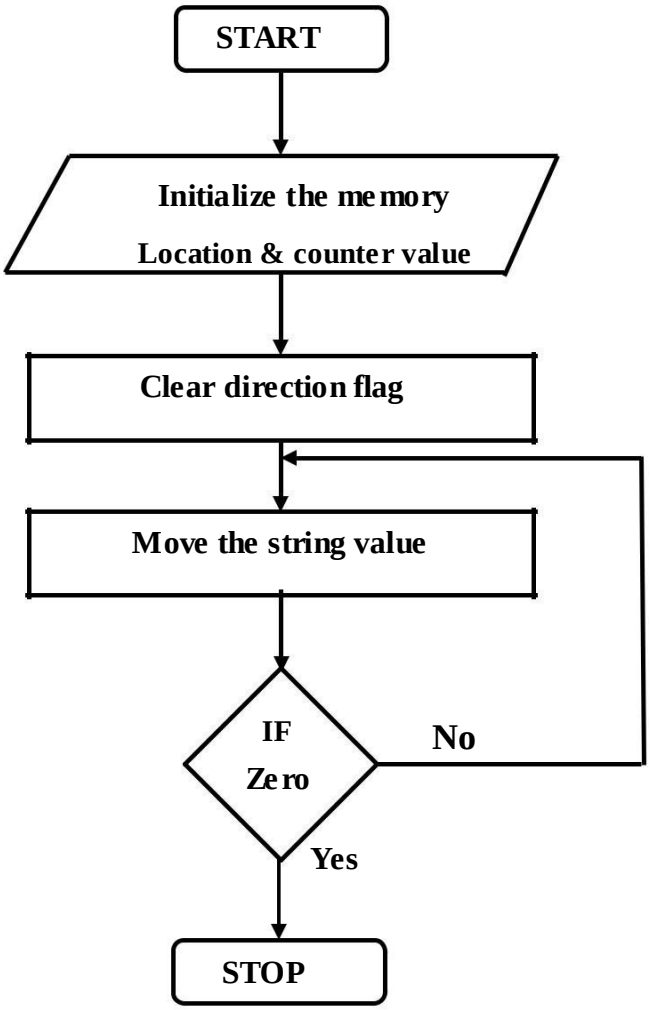
Step4: Get the value of byte

Step5: Initialize the memory

Step6: Store the value in memory

Step7: Stop the process

Move the String:



3a) Move the String:

PROGRAM:

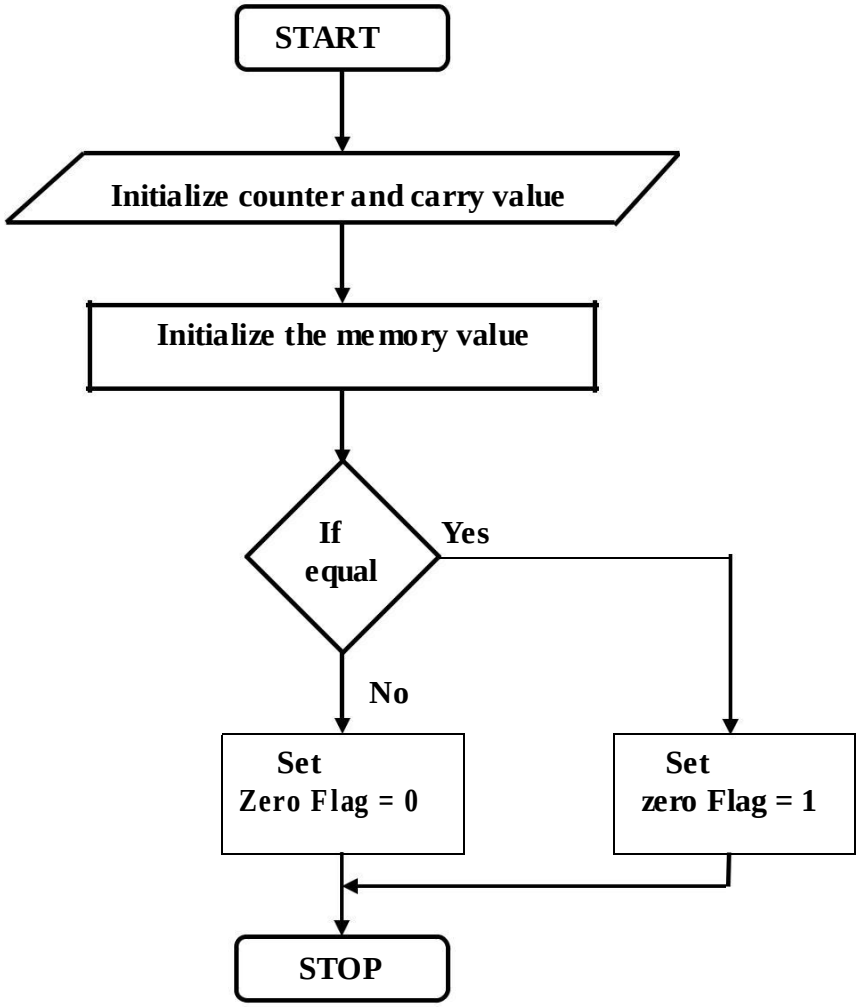
Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
	1000	MOV	SI,1100	C7, C6, 00, 11	Initialize the memory
	1004	MOV	DI,1200	C7, C7, 00, 12	Initialize the memory
	1008	MOV	CX,0005	C7, C1, 05, 00	Initialize the counter
	100C	CLD		FC	Clear the direction flag
LOOP1:	100D	MOVSB		A4	Store the result of string
	100E	LOOP	LOOP1	E2, FD	Go to LOOP L1
	1010	HLT		F4	Stop the process.

Observation:

Address	Input
1100	**11**
1101	**22**
1102	**33**
1103	**44**
1104	**55**

Address	Output
1200	**11**
1201	**22**
1202	**33**
1203	**44**
1204	**55**

Compare the String:



3b) Compare the String:

PROGRAM

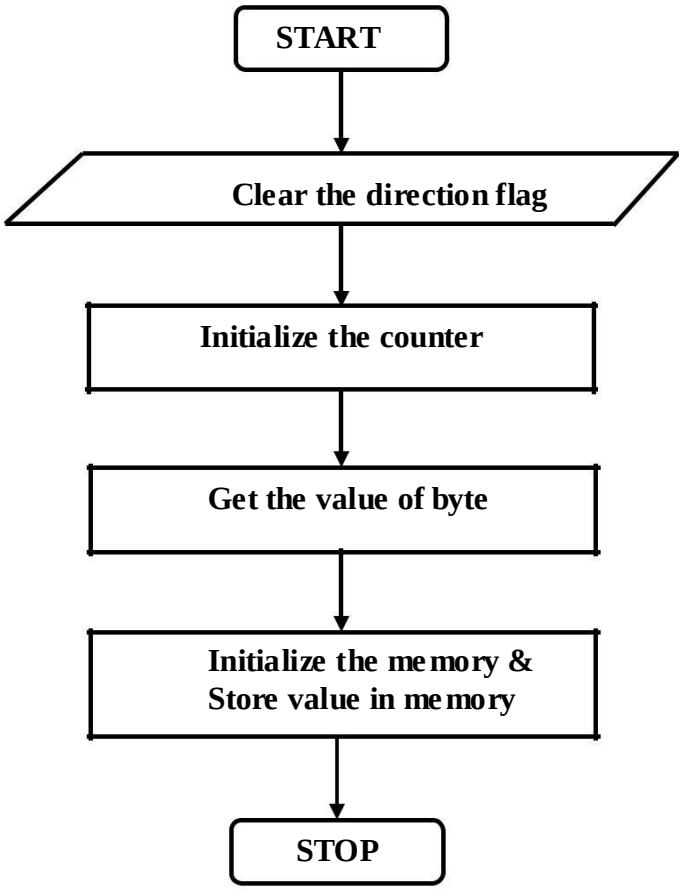
Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
	1000	CLD		FC	Clear the direction flag
	1001	MOV	DX,0000	C7, C2, 00, 00	Initialize the carry
	1005	MOV	CX,0005	C7, C1, 05, 00	Initialize the counter
	1009	MOV	SI,1200	C7, C6, 00, 12	
	100D	MOV	DI,1300	C7, C7, 00, 13	Initialize the memory
	1011	REPZ	CMPSB	F3, A6	Compare the string
	1013	JNZ	LOOP1	75, 01	If no zero to L1
	1015	INC	DX	42	Increment DX value.
LOOP1:	1016	MOV	[1400], DX	89, 16, 00, 14	Move the value in memory
	101A	HLT		F4	Stop the process

Address	Input
1200	11
1201	12
1202	13
1203	14
1204	15

Address	Input
1300	11
1301	12
1302	13
1303	14
1304	15

Address	Output
1400	01
1401	00

Fill a Byte:


START
Clear the direction flag
Initialize the counter
Get the value of byte
Initialize the memory &
Store value in memory
STOP

3c) Fill a Byte

PROGRAM

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
	1000	CLD		FC	Clear the direction flag
	1001	MOV	DX, 0005	C7, C2, 05, 00	Initialize the counter
	1005	MOV	AL,1F	C6, C0,1F	Get the value of byte
	1008	MOV	DI,1200	C7, C7, 00, 12	Initialize the memory
LOOP1:	100C	STOSB		AA	Store the value
	100D	LOOP	LOOP1	E2, FD	Loop L1
	100F	HLT		F4	Stop the process

Address	Input
1200	**1F**
1201	**1F**
1202	**1F**
1203	**1F**
1204	**1F**

Address	Output
1200	**1F**
1201	**1F**
1202	**1F**
1203	**1F**
1204	**1F**

RESULT:

Thus the operation of string manipulation is done and verified using 8086 microprocessor**.**

VIVA QUESTIONS AND ANSWERS

1. Explain the difference between a JMP and CALL instruction?

A JMP instruction permanently changes the program counter.

A CALL instruction leaves information on the stack so that the original program execution sequence can be resumed.

2. What is Assembler?

The assembler translates the assembly language program text which is given as input to the assembler to their binary equivalents known as object code.

3. What is the use of HLDA?

HLDA is the acknowledgment signal for HOLD. It indicates whether the HOLD signal is received or not.

HOLD and HLDA are used as the control signals for DMA operations.

4. Explain about "LEA"?

LEA(Load Effective Address) is used for initializing a register with an offset address.

A common use for LEA is to intialize an offset in BX, DI or SI for indexing an address in memory.

5. Difference between "Shift" and "Rotate".

Shift and Rotate commands are used to convert a number to another form where some bits are shifted or rotated.

A rotate instruction is a closed loop instruction. That is, the data moved out at one end is put back in at the other end.

6. What are the modes in which 8086 can operate?

The 8086 can operate in two modes and they are minimum (or uniprocessor) mode and maximum (or multiprocessor) mode.

7. What is the data and address size in 8086?

The 8086 can operate on either 8-bit or 16-bit data. The 8086 uses 20 bit address to access memory and 16-bit address to access 1/0 devices.

8. Explain the function of M/IO in 8086.

The signal M/IO is used to differentiate memory address and 1/0 address When the processor is accessing memory locations MI 10 is asserted high and when it is accessing 1/0 mapped devices it is asserted low.

4. CODE CONVERSION, DECIMAL ARITHMETIC AND MATRIX OPERATIONS

4a) Hexadecimal to Decimal code conversion

Aim:

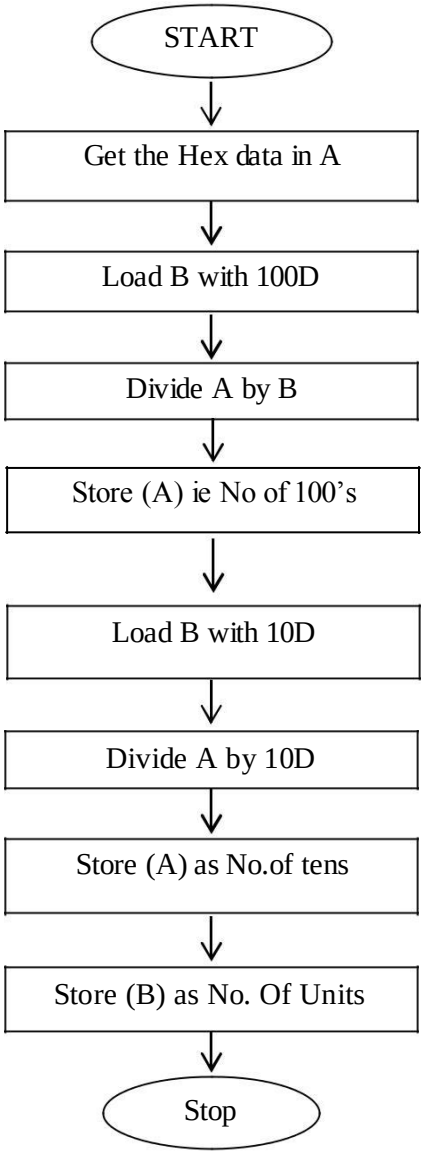
To write an assembly language program to convert hexadecimal number into decimal number

Algorithm:

1. Load the number to be converted into the accumulator.
2. If the number is less than 100 (64H), go to next step; otherwise, subtract 100 (64H) repeatedly until the remainder is less than 100 (64H). Have the count(100's value) in separate register which is the carry.
3. If the number is less than 10 (0AH), go to next step; otherwise, subtract 10 (0AH) repeatedly until the remainder is less than 10 (0AH). Have the count(ten's value) in separate register.
4. The accumulator now has the units.
5. Multiply the ten's value by 10 and add it with the units.
6. Store the result and carry in the specified memory location.

FLOWCHART:

Hexadecimal to Decimal conversion



PROGRAM

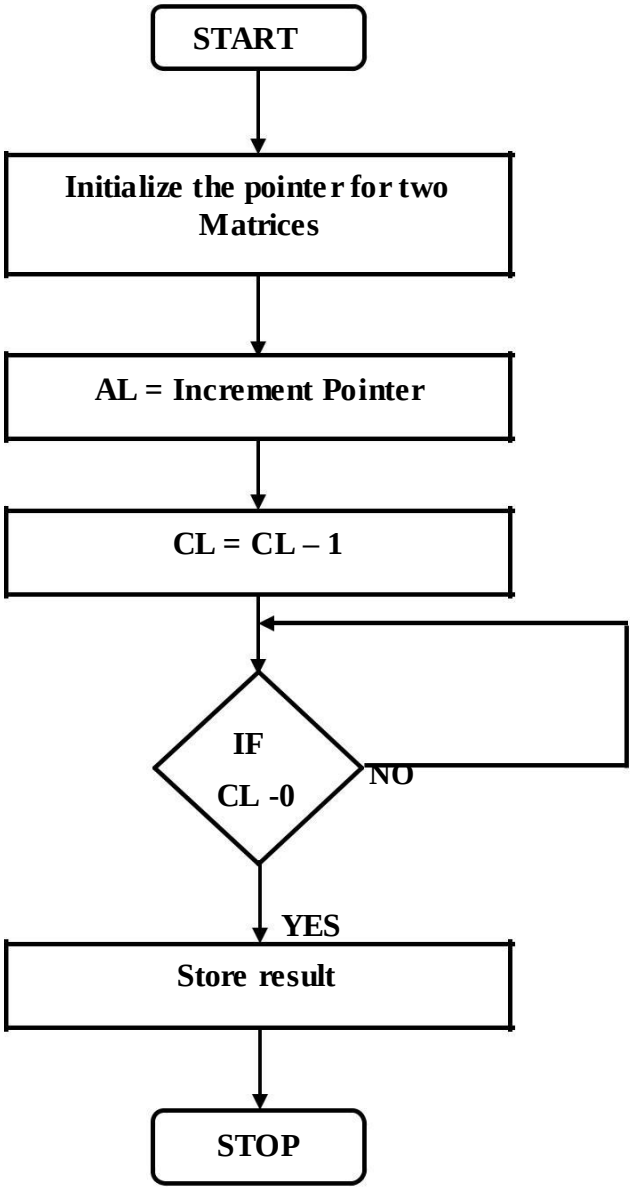
Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START	1000	MOV	SI,1100	C7 C6 00 11	; Load the input address 1100
	1004	MOV	DX,00 00	C7 C2 00 00	; Load address in SI
	1008	MOV	AX,[SI]	8B 04	; Load 64 to Count the number of 100s
	100A	MOV	BX,00 64	C7 C3 64 00	;Get the number of hundreds
	100E	DIV	BX	F7 F3	; Load number of hundreds in1102 & 1103
	1010	MOV	[SI+02],AX	89 44 02	; Move the remainder to AX
	1013	MOV	AX,DX	89 D0	; Initialize DX with 0000
	1015	MOV	DX ,00 00	C7 C2 00 00	; Load 0A to find number of tens
	1019	MOV	BX, 00 0A	C7 C3 0A 00	; Divide by 0A to get number of tens
	101D	DIV	BX	F7 F3	; Move no of tens to the address 1104 & 1105
	101F	MOV	[SI+04],AX	89 44 04	; Move no of ones to the address 1106 & 1107
	1022	MOV	[SI+06],DX	89 54 06	; Halt
	1025	HLT		F4	

Address	Input
1100	**FF**
1101	**00**

Address	Output
1102	**02**
1103	**00**
1104	**05**
1105	**00**
1106	**05**
1107	00

MATRIX OPERATION

FLOW CHART:



4b. MATRIX OPERATIONS USING 8086

AIM:

To write a program for addition of two 3x3 matrix by using 8086.

ALGORITHM:

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

PROGRAM

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START	1000	MOV	CL,09	C6 C1 09	;count for 3 x 3 matrix
	1003	MOV	SI,1200	C7 C6 00 12	; address in SI
	1007	MOV	DI,1300	C7 C7 00 13	; address in DI
LOOP	100B	MOV	AL,[SI]	8A 04	;Load AL with matrix
	100D	MOV	BL,[DI]	8A 1D	; Load BL with matrix
	100F	ADD	AL,BL	00 D8	; ADD two data
	1011	MOV	[DI],AL	88 05	;Store result
	1013	INC	DI	47	; Increment DI
	1014	INC	SI	46	; Increment SI
	1015	DEC	CL	FE C9	;Decrement CL
	1017	JNZ	LOOP	75 F2	; Loop continues until zero
	1019	INT	3	CC	; Break point

Address	Input	Address	Input
1200	01	1300	12
1201	02	1301	02
1202	03	1302	04
1203	04	1303	06
1204	05	1304	08
1205	06	1305	02
1206	07	1306	04
1207	08	1307	06
1208	09	1308	03

Address	Output
1300	13
1301	04
1302	07
1303	0A
1304	0D
1305	08
1306	0B
1307	0E
1308	0C

PROGRAM for Matrix operation using MASM assembler

```
.MODEL SMALL
.DATA
    TAB  DB 3,4,5,6,0
         DB 1,4,5,7,0
         DB 1,8,9,0,0
         DB 1,8,9,2,0
         DB 1,1,1,1,0
         DB 0,0,0,0,0
    TOTROWS DB 0
    TOTCOLS DB 0
    ROWS DB 5
    COLS DB 4
.CODE
    MOV AX,@DATA
    MOV DS,AX
    ; COUNTING TOTAL ROWS
    LEA SI,TAB
    L1: MOV CX,4
    L2: MOV AH,BYTE PTR[SI]
        ADD TOTROWS ,
         AH INC SI
```

```
        LOOP L2
        MOV AH,TOTROWS
        MOV [SI],AH
        MOV TOTROWS,0
        INC SI
        SUB ROWS,1
        CMP ROWS,0
        JG L1

; COUNTING TOTAL COLS
        LEA SI,TAB
        MOV BX,00
L3:     MOV CX,5
        LEA SI,TAB
        ADD SI,BX
L4:     MOV AH,BYTE PTR[SI]
        ADD TOTCOLS , AH
        ADD SI,5
        LOOP L4
        MOV AH,TOTCOLS
        MOV [SI],AH
        MOV TOTCOLS,0
        SUB COLS,1
        CMP COLS,0
        ADD BX,1
        JG L3
        MOV AX,4C00H
        INT 21H
        END
```

RESULT:

Thus the matrix operation and code conversion were executed and verified successfully.

VIVA QUESTIONS AND ANSWERS

1. Difference between JMP and JNC?

A:-JMP is Unconditional Branch.

JNC is Conditional Branch.

2.**What are the 4 Segments in 8086?**

A:-Code Segment Register {CS}

Data Segment Register {DS}

Extra Segment Register {ES}

Stack Segment Register {SS}

3. Distinguish between packed BCD and unpacked BCD

Packed BCD numbers are stored two digits to a byte in 4 bit groups referred as nibbles

Ex:86 in unpacked BCD there is only one digit per byte Ex: 08, 06

4. Describe CBW and CWD instructions

The CBW and CWDE mnemonics reference the same opcode. The CBW instruction is intended for use when the operand-size attribute is 16 and the CWDE instruction for when the operand-size attribute is 32. The CWDE instruction is different from the CWD (convert word to double) instruction. The CWD instruction uses the DX:AX register pair as a destination operand; whereas, the CWDE instruction uses the EAX register as a destination.

5. Describe about MUL, IMUL, DIV, IDIV instructions

MUL (multiply) instruction is used for unsigned multiplication. This instruction multiplies bytes or words.

IMUL (Integer multiply) instruction is used for signed multiplication. This instruction multiply bytes or words.

The DIV instruction is to divide unsigned data. We can divide a byte by byte, a word by byte, double word by word.

The IDIV instruction is to divide signed data. We can divide a byte by byte, a word by byte, double word by word and the operations are just like DIV instructions

6.Describe about LOOP instructions

The LOOP instruction is a combination of a decrement of CX and a conditional jump. In the 8086, LOOP decrements CX and if CX is not equal to zero, it jumps to the address indicated by the label. If CX becomes a 0, the next sequential instruction executes.

5. MOVE A DATA BLOCK WITHOUT OVERLAP

AIM:

To convert a given Move a data block without overlap using u086 MASM assembler and 8086 kit.

ALGORITHM:

1. Initialize the memory location to the data pointer.
2. Increment B register.
3. Increment accumulator by 1 and adjust it to decimal every time.
4. Compare the given decimal number with accumulator value.
5. When both matches, the equivalent hexadecimal value is in B register.
6. Store the resultant in memory location.

Move data block without overlap using 8086 kit			
1000			ORG 1000H
1000	B8 0000		MOV AX,0000H
1003	8E D8		MOV DS,AX
1005	B9 0005		MOV CX,0005
1008	BF 3000		MOV DI,3000H
100B	BE 1200		MOV SI,1200H
100E	8B 04	L1	MOV AX,[SI]
1010	89 05		MOV [DI],AX
1012	46		INC SI
1013	47		INC DI
1014	49		DEC CX
1015	8B C1		MOV AX,CX
1017	75 F5		JNZ L1
1019	B4 4C		MOV AH,4CH
101B	CD 21		INT 21H

OBSERVATION:

INPUT:

1200 = 14H
1201 = 35H
1202 = 18H
1203 = 36H
1204 = 54H

OUTPUT:

1300 = 14H
1301 = 35H
1302 = 18H
1303 = 36H
1304 = 54H

PROGRAM

Move data block without overlap using 8086 MASAM Assembler

```
DATA SEGMENT X DB 01H,02H,03H,04H,05H     ;Initialize Data Segments Memory Locations
Y DB 05 DUP (0)
DATA ENDS
CODE SEGMENT ASSUME CS: CODE, DS: DATA
START:
MOV AX, DATA          ; Initialize DS to point to start of the memory
MOV DS, AX            ; set aside for storing of data
MOV CX, 05H           ; Load counter
LEA SI, X+04          ; SI pointer pointed to top of the memory
LEA DI, X+04+03       ; 03 is displacement of over lapping, DI pointed to; the top of the
                        destination block
CODE ENDS
END START
```

RESULT:

Thus the program for moving the data block without overlap was executed and verified using 8086 MASM assembler and 8086 kit.

VIVA QUESTIONS AND ANSWERS

1. Give examples of conditional branch instructions

In a loop if there are different jump instructions with a condition or counter called conditional loop and instructions in that loop are called unconditional branch instructions.

2. Give examples of unconditional branch instructions

In a loop if there are different jump instructions with no condition it is called unconditional loop and instructions in that loop are called unconditional branch instructions.

3. What are flag manipulation instructions ? Give examples

Flag manipulation instructions. STC, CLC, CMC. Set, clear, complement carry flag. STD, CLD. Set, clear direction flag

4.Explain about DAA instruction

decimal adjust addition result

DAA

The daa instruction is used to adjust the content of the AL register after that register is used to perform the addition of two packed BCDs.

5. **Explain about CALL and RETURN instructions**

CALL 16-bit memory address of a subroutine

It is a 3-byte instruction that transfers the program sequence to a subroutine

RETURN instruction in the subroutine. The return instruction is used either to return a function value or to terminate the execution of a function.

6. PASSWORD CHECKING, PRINT RAM SIZE AND SYSTEM DATE

AIM:

To write an 8086 MASM assembler program for performing password checking, Print RAM size and system date.

APPARATUS REQUIRED:

SL.NO	ITEM	QUANTITY
1.	**8086 Microprocessor kit**	1
2.	Intel Desktop systems with MASM	1
3.	RTC Interface board	1

PROGRAM:

6 A) PASSWORD CHECKING

```
; PASSWORD IS MASM1234
DATA SEGMENT
PASSWORD DB 'MASM1234'
LEN EQU ($-PASSWORD)
MSG1 DB 10,13,'ENTER YOUR PASSWORD: $'
MSG2 DB 10,13,'WELCOME TO ELECTRONICS WORLD!!$'
MSG3 DB 10,13,'INCORRECT PASSWORD!$'
NEW DB 10,13,'$'
INST DB 10 DUP(0)
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS:
DATA START:
MOV AX,DATA
MOV DS,AX
LEA DX,MSG1
MOV AH,09H
INT 21H
MOV SI,00
UP1:

MOV AH,08H
INT 21H

CMP AL,0DH
```

```
JE DOWN
MOV [INST+SI],AL
MOV DL,'*'
MOV AH,02H
INT 21H
INC SI
JMP UP1
DOWN:
MOV BX,00
MOV CX,LEN
CHECK:
MOV AL,[INST+BX]
MOV DL,[PASSWORD+BX]
CMP AL,DL
JNE FAIL
INC BX
LOOP CHECK
LEA DX,MSG2
MOV AH,09H
INT 21H
JMP FINISH
FAIL:
LEA DX,MSG3
MOV AH,009H
INT 21H
FINISH:
INT 3
CODE ENDS
END START
END
```

****

Password set Input

1240:16 (1)

1241:1E (2)

1242:26 (3)

1243:25(4)

1244:2E(5)

Output:

Enter the Password:

Type 12345

Message: 'WELCOME TO ELECTRONICS WORLD!!$'

>>

6 B)DISPLAY MONTH/DAY/YEAR

```
.MODEL SMALL
        .STACK 64
        .DATA
        Today
        SAVEDAY   DB    ?
        SAVEMON   DB    ?
        TEN       DB    10
        ELEVEN    DB    11
        TWELVE    DB    12
        DAYSTAB   DB    'SUNDAY,$ ', 'MONDAY,$'
                  DB    'TUESDAY,$ ', 'WEDNESDAY,$'
                  DB    'THURSDAY,$ ', 'FRIDAY,$'
                  DB    'SATURDAY,$ '
        MONTAB    DB    'JANUARY $','FEBUARY $','MARCH $'
                  DB    'APRIL $','MAY $','JUNE $'
                  DB    'JULY $','AUGUST $','SEPTEMBER $'
                  DB    'OCTOBER $','NOVEMBER $','DECEMBER $'
                  CODE
        BEGIN     PROC  FAR
                  MOV   AX,@DATA
                  MOV   DS,AX
                  MOV   ES,AX
                  MOV   AX,0600H
                  CALL  Q10SCR
                  CALL  Q20CURS
                  MOV   AH,2AH
                  INT   21H
                  MOV   SAVEMON,DH
                  MOV   SAVEDAY,DL
                  CALL  B10DAYWK
                  CALL  C10MONTH
                  CALL  D10DAYMO
                  CALL  E10INPT
                  CALL  Q10SCR
```

```
                MOV     AX,4C00H
                INT     21H
BEGIN           ENDP
B10DAYWK        PROC    NEAR
                MUL     TWELVE
                LEA     DX,DAYSTAB
                ADD     DX,AX
                MOV     AH,09H
                INT     21H
                RET
B10DAYWK        ENDP
C10MONTH        PROC    NEAR
                MOV     AL,SAVEMON
                DEC     AL
                MUL     ELEVEN
                LEA     DX,MONTAB
                ADD     DX,AX
                MOV     AH,09H
                INT     21H
                RET
C10MONTH        ENDP
.386
D10DAYMO        PROC    NEAR
                MOVZX   AX,SAVEDAY
                DIV     TEN
                OR      AX,3030H
                MOV     BX,AX
                MOV     AH,02H
                MOV     DL,BL
                INT     21H
                MOV     AH,02H
                MOV     DL,BH
                INT     21H
                RET
```

```
**************************
D10DAYMO            ENDP

E10INPT             PROC    NEAR
                    MOV     AH,10H
                    INT     16H
                    RET
E10INPT             ENDP

Q10SCR              PROC    NEAR
                    MOV     AX,0600H
                    MOV     BH,17H
                    MOV     CX,0000
                    MOV     DX,184FH
                    INT     10H
                    RET
Q10SCR              ENDP
Q20CURS             PROC    NEAR
                    MOV     AH,02H
                    MOV     BH,00
                    MOV     DH,10
                    MOV     DL,24
                    INT     10H
                    RET
Q20CURS             ENDP
                    END     BEGIN
```

Observation:

Input

Set time: 1200 : 05 LSB

1201: 05 MSB(seconds)

1202: 09

1203:05 (Minutes)

1204:03

1205: 02(Hours)

Set Date:

1206: 05 LSB

1207: 02 MSB(Date)

1208: 01LSB

1209: 00 MSB (Month)

120A:06 LSB

120B : 01 MSB (year)

Output:

The time is displayed as ; 23:59:55

The date is displayed as ; 25:01:16

After 5 seconds the date is displayed as

Date: 26:01:15 in VBMB 8 Kit

>>

6 C) RAM SIZE

```
ORG 0000H
CLR
CLR
CPL A
ADD A, #01H
MOV A,R3
AGAIN: SJMP AGAIN
**********
```

Observation:

OUTPUT

"RAM SIZE IS 16 KB" is displayed in the LCD.

RESULT:

Thus the output for the Password checking, Print RAM size and system date was executed and verified using MASM assembler successfully

VIVA QUESTIONS AND ANSWERS

1. How do you read and write characters on to screen using interrupts?

An interrupt is a condition that causes the microprocessor to temporarily work on a different task, and then later return to its previous task. Interrupts can be internal or external.

2. What is the significance of LEA instruction?

LEA(Load Effective Address) is used for initializing a register with an offset address.

A common use for LEA is to intialize an offset in BX, DI or SI for indexing an address in memory.

An equivalent operation to LEA is MOV with the OFFSET operator, which generates slightly shorter machine code.

3. What is an assembler directive?

An assembler directive is a direct command to microprocessor to perform certain operations.

4. How the assembler process is carried out in 8086?

A microprocessor executes a collection of machine instructions that tell the processor what to do is known as assembly process.

5. How a procedure is represented in assembler directive?

Procedures are a group of instructions stored as a separate program in memory and it is called from the main program whenever required. The type of procedure depends on where the procedures are stored in memory. If it is in the same code segment as that of the main program then it is a near procedure otherwise it is a far procedure.

7. COUNTERS AND TIME DELAY

AIM:

To write an assembly language program for up counter using 8086 kit and 8086 MASM assembler.

APPARATUS REQUIRED:

SL.NO	ITEM	SPECIFICATION	QUANTITY
1.	Microprocessor kit	8086 kit	1
2.	Power Supply	+5 V, dc,+12V dc	1
3.	RTC Interface board	–	–

PROCEDURE:

1. Enter the program into the kit
2. Execute the program
3. The counter value displayed in the LCD, he value starts from 00H T0 99H

PROGRAM

UP COUNTER using 8086 kit

```
1000    EB 2F 10    START:    CALL CONVERT
1003    E8 00 1D              CALL DISPLAY
1006    B9 00 00    DELAY:    MOV CX,0000H
1009    41          L1:       INC CX
100A    81 F9 FF FF           CMP CX,0FFFFH
100E    75 F9                 JNZ L1
1010    BE 00 15              MOV SI,1500H
1013    8A 04                 MOV AL,[SI]
1015    FE C0                      INC AL
1017    88 04                 MOV [SI],AL
1019    3C 64                 CMP AL,064H
101B    75 E3                 JNZ START
101D    B0 00                 MOV AL,00H
101F    88 04                 MOV [SI],AL
1021    EB DD                      JMP START
1023    B4 06       DISPLAY:  MOV AH,06H
1025    BA 00 16              MOV DX,1600H
1028    B5 01                 MOV CH,01H
102A    B1,00                 MOV CL,00H
102C    CD 05                      INT 5
102E    C3                    RET
102F    BE 00 15    CONVERT:  MOV [SI],1500H
1032    BB 02 16              MOV BX,1602H
1035    B0 24                 MOV AL,24H
1037    88 07                 MOV [BX],AL
1039    8A 04                 MOV AL,[SI]
103B    B4 00                 MOV AH,00H
103D    B6 0A                 MOV DH,0AH
103F    F6 F6                 DIV DH
1041    80 C4 30              ADD AH,30H
1044    4B                    DEC BX
1045    88 27                 MOV[BX],AH
1047    4B                    DEC BX
1048    04 30                 ADD AL,30H
104A    88 07                 MOV [BX],AL
104C    4B                    DEC BX
104D    C3                    RET
104E    E4 02       GETC:         IN AL,02H
1050    24 FF                 AND AL,0FFH
1052    3C F0                 CMP AL,0F0H
1054    75 F8                 JNE GETC
1055    F4                    HLT
```

UP COUNTER using 8086 MASM assembler

```
MODEL SMALL
STACK 100H
DATA
PROMPT DB 'The counting from 0 to 9 is : $'
CODE
MAIN PROC
MOV AX, @DATA          ; initialize DS
MOV DS, AX
LEA DX, PROMPT         ; load and print PROMPT
MOV AH, 9
INT 21H
MOV CX, 10             ; initialize CX
MOV AH, 2               ; set output function
MOV DL, 48              ; set DL with 0
@LOOP:                 ; loop label
 INT 21H               ; print character
INC DL                  ; increment DL to next ASCII character
DEC CX                 ; decrement CX
JNZ @LOOP               ; jump to label @LOOP if CX is 0
MOV AH, 4CH            ; return control to DOS
INT 21H
MAIN ENDP
END MAIN
```

RESULT:

Thus the program for up counter using 8086 MASM assembler was executed and verified successfully

VIVA QUESTIONS AND ANSWERS

1. **What is a RAM?**
   RAM is a random access memory which is used to store data temporarily.

2. **What are the types of RAM?**
   Static RAM, Dynamic RAM

3. **How many 32kB RAMs can be interfaced with 8086?**
   4 32kB RAMs can be interfaced with 8086

4. **What is the necessity of RAM in processor?**
   RAM is necessary to hold the data temporarily when a processor is executing any program.

5. **Differentiate EPROM and EEPROM.**
   EPROM and EEPROM both are erasable and can be reprogrammed, but the basic difference between them is that **EPROM** is erased using **Ultra violet rays** whereas, **EEPROM** can be erased using **electric signals**. Let us discuss the differences between EPROM and EEPROM with the help of comparison chart shown below.

8. TRAFFIC LIGHT CONTROL

AIM:-

To write an assembly program for Traffic Light Control using 8086 LCD Microprocessor Kit.

PROGRAM:

```
CNTRL     EQU     26H
PORT A    EQU     20H
PORT B    EQU     22H
PORT C    EQU     24H
```

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START	1000	MOV	AL,80H	C6 C0 80	
	1003	OUT	(CNTRL)26,AL	E6 26	
REPEAT	1005	MOV	BX,LOOK UP	C7 C3 73 10	
	1009	MOV	SI,LABEL	C7 C6 7F 10	
	100D	CALL	OUT	E8 33 00	
	1010	MOV	AL,[SI]	8A 04	
	1012	OUT	(PORTA)20,AL	E6 20	
	1014	CALL	DELAY 1	E8 4D 00	
	1017	INC	SI	46	
	1018	INC	BX	43	
	1019	CALL	OUT	E8 27 00	
	101C	MOV	AL,[SI]	8A 04	
	101E	OUT	(PORTB)22,AL	E6 22	
	1020	CALL	DELAY 1	E8 41 00	
	1023	INC	SI	46	
	1024	INC	BX	43	
	1025	CALL	OUT	E8 1B 00	
	1028	MOV	AL,[SI]	8A 04	
	102A	OUT	(PORTC)24,AL	E6 24	
	102C	CALL	DELAY 1	E8 35 00	
	102F	INC	SI	46	
	1030	INC	BX	43	

	1031	CALL	OUT	E8 0F 00	
	1034	MOV	AL,[SI]	8A 04	
	1036	OUT	(PORTC)24,AL	E6 24	
	1038	INC	SI	46	
	1039	MOV	AL,[SI]	8A 04	
	103B	OUT	(PORTA)20,,AL	E6 26	
OUT:	103D	CALL	DELAY 1	E8 24 00	
	1040	JMP	REPEAT	E9 C2 FF	
	1043	MOV	AL,[BX]	8A 07	
	1045	OUT	(PORTC)24,AL	E6 24	
	1047	INC	BX	43	
	1048	MOV	AL,[BX]	8A 07	
	104A	OUT	(PORTB)22,AL	E6 22	
	104C	INC	BX	43	
	104D	MOV	AL,[BX]	8A 07	
	104F	OUT	(PORTA)20,AL	E6 20	
DELAY:	1051	CALL	DELAY	E8 01 00	
A:	1054	RET		C3	
A1:	1055	MOV	DI,00040H	C7 C7 40 00	
	1059	MOV	DX,0FFFFH	C7 C2 FF FF	
	105D	DEC	DX	4A	
	105E	JNZ	**A1**	75 FD	
	1060	DEC	DI	4F	
DELAY1:	1061	JNZ	**A**	75 F6	
B:	1063	RET		C3	
B1:	1064	MOV	DI,00015H	C7 C7 15 00	
	1068	MOV	DX,0FFFFH	C7 C2 FF FF	
	106C	DEC	DX	4A	
	106D	JNZ	**B1**	75 FD	
	106F	DEC	DI	4F	
LOOK UP:	1070	JNZ	**B**	75 F6	
	1072	RET		C3	

LABEL:	1073	DB	12H,27H,44H,10H		
	1077		2BH,92H,10H,9DH		
	107B		84H,48H,2EH,84H		
	107F	DB	48H,6BH,20H,49H		
	1083		04		

VIVA QUESTIONS AND ANSWERS

1. Give the sequence of operation in traffic light controller.

The typical sequence is as follows:
Green (safe to proceed)
Amber (slow down, red light soon)
Red (stop)
Red / amber (stay stopped but just letting you know the light turns green soon)

2. What is the name of the peripheral device used to interface traffic light controller with microprocessor?

8255 PPI(Programmable peripheral Interface)

3. What is 8255?

It is PPI- Programmable Peripheral Interface. it is used to connect I/O devices to microprocessor and supports parallel communication.

4. How many input and output ports are in PPI?

The port is a buffered I/O, which is used to hold the data transmitted from the processor to I/O device or vice-versa

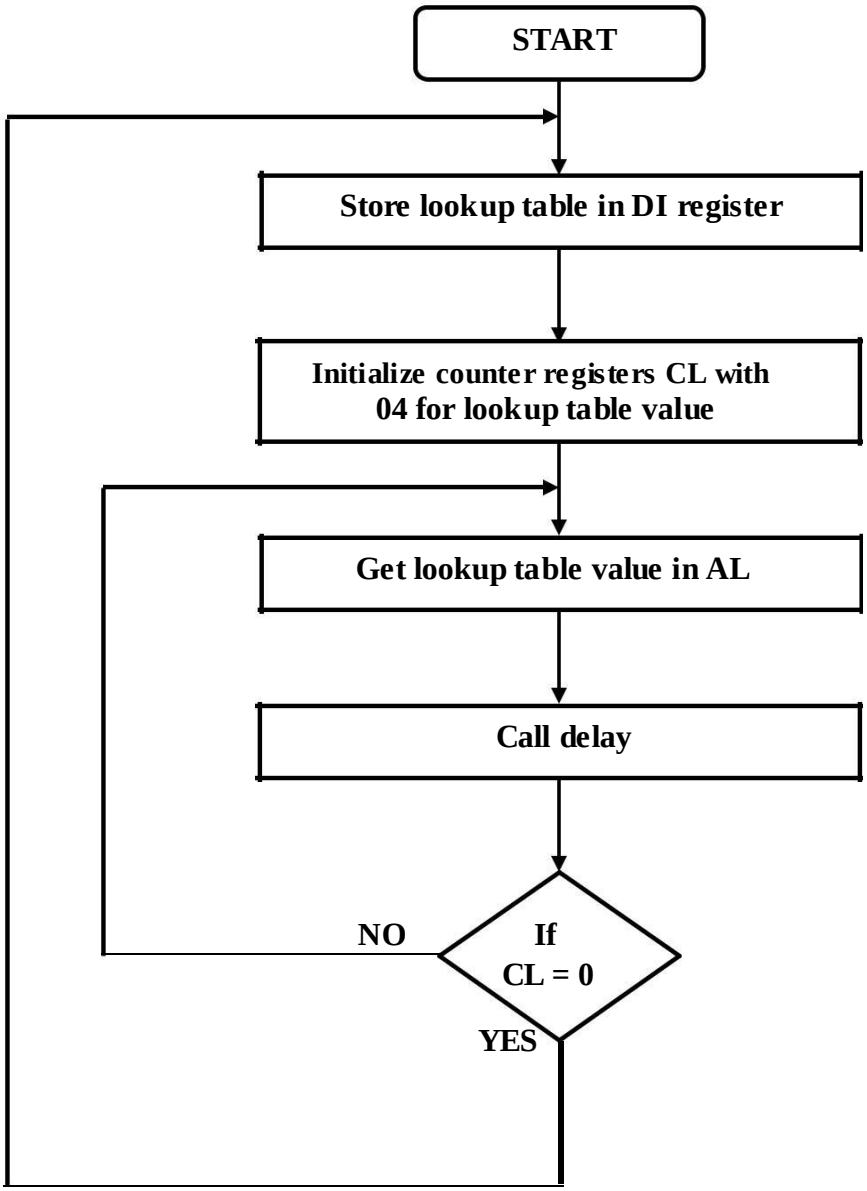
5. What is BSR mode?

Bit set or reset mode, If BSR=1,bit is set,if BSR=0,it is reset.

RESULT:

Thus the assembly language program for Traffic Light Control was executed and verified using 8086 Microprocessor kit.

FLOW CHART:


START
Store lookup table in DI register
Initialize counter registers CL with 04 for lookup table value
Get lookup table value in AL
Call delay
If CL = 0
NO
YES

9. STEPPER MOTOR CONTROL

AIM:-

To write an assembly language program to control the speed of stepper motor in both directions using 8086 Microprocessor kit.

APPARATUS REQUIRED:

i. Microprocessor kit
ii. Stepper Motor Interface Card
iii. Stepper motor

ALGORITHM:-

a. Start the program
b. Store lookup table value in DI register
c. Initialize counter register CL with 04_H for lookup table value.
d. Get lookup table value in CL.
e. Call delay
f. If CL = 0, go to step1 otherwise get next lookup table value.

Lookup table:-

(Anti clockwise direction)	**(Clockwise direction)**
1200 : 09	1200 : 0A
1201 : 05	1201 : 06
1202 : 06	1202 : 05
1203 : 0A	1203 : 09

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START	1000	MOV	DI,1200	C7,C7,00,12	; Initialize lookup table
	1004	MOV	CL,04	C6,C1,04	;Initialize count value
REPEAT	1007	MOV	AL,[DI]	8A 05	Get lookup table value
	1009	OUT	C0,AL	E6 C0	;Sent it to output port
	100B	MOV	DX,1010H	C7 C2 10 10	;Delay program
DELAY	100F	DEC	DX	4A	
	1010	JNZ	DELAY	75 FD	
	1012	INC	DI	47	;Increment [DI]
	1013	LOOP	REPEAT	E2 F2	;if CX ≠ 0, go to Repeat
	1015	JMP	START	E9 E8 FF	;Repeat to start

VIVA QUESTIONS AND ANSWERS

1. What are the applications of stepper motor

Used in tape drives, floppy disc drives printers and electric watches. The **stepper motor** also use in X-Y plotter and robotics

2. Discuss the salient features of stepper motor

The rotation angle of the **motor** is proportional to the input pulse.

The **motor** has full torque at standstill. Precise positioning and repeatability of movement since good **stepper motors** have an accuracy of 3 – 5% of a step and this error is non cumulative from one step to the next.

3. What are the schemes used in stepper motor

A microcontroller or stepper motor controller can be used to activate the drive .Various drive techniques have been developed to better approximate a sinusoidal drive waveform: these are half stepping and micro stepping.

4. Write the calculation for step size.

Let N_r be the number of rotor teeth and m be the number of stacks or phases.Hence, Tooth pitch is represented by the

$$\tau_p = \frac{360^\circ}{N_r} \ldots\ldots\ldots (1)$$

Therefore,

$$\text{Step angle} = \frac{360^\circ}{m\, N_r} \ldots\ldots\ldots (2)$$

5. How can the speed of stepper motor can be controlled?

To control the speed of a stepper motor, you control the time between steps. And as long as there is enough excess torque to keep up, you can control the position, speed, and acceleration.

RESULT:

Thus the assembly language program for speed control of stepper motor was executed and verified using 8086 Microprocessor kit.

10. DIGITAL CLOCK

AIM:-

To display the digital clock specifically by displaying the hours, minutes and seconds using 8086 kits

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	1000	MOV	AL,05H	C6 C0 05	
	1003	OUT	DE,AL	E6 DE	
	1005	MOV	AL,04H	C6 C0 04	
	1008	OUT	DE,AL	E6 DE	
	100A	MOV	SI,1310H	C7 C6 10 13	
	100E	MOV	AL,[SI]	8A 04	
	1010	OUT	C0,AL	E6 C0	
	1012	INC	SI	46	
	1013	MOV	AL,[SI]	8A 04	
	1015	OUT	D0,AL	E6 D0	
	1017	INC	SI	46	
	1018	MOV	AL,[SI]	8A 04	
	101A	OUT	C2,AL	E6 C2	
	101C	INC	SI	46	
	101D	MOV	AL,[SI]	8A 04	
	101F	OUT	D2,AL	E6 D2	
	1021	INC	SI	46	
	1022	MOV	AL,[SI]	8A 04	
	1024	OUT	C4,AL	E6 C4	
	1026	INC	SI	46	
	1027	MOV	AL,[SI]	8A 04	
	1029	OUT	D4,AL	E6 D4	
L1:	102B	MOV	SI,1320H	C7 C6 20 13	
	102F	IN	AL,D4H	E4 D4	
	1031	AND	AL,0FH	80 E0 0F	
	1034	MOV	[SI],AL	88 04	
	1036	IN	AL, C4H	E4 C4	
	1038	AND	AL,0FH	80 E0 0F	
	103B	INC	SI	46	
	103C	MOV	[SI],AL	88 04	
	103E	IN	AL, D2H	E4 D2	
	1040	AND	AL,0FH	80 E0 0F	
	1043	INC	SI	46	
	1044	MOV	[SI],AL	88 04	
	1046	IN	AL, C2H	E4 C2	
	1048	AND	AL,0FH	80 E0 0F	
	104B	INC	SI	46	
	104C	MOV	[SI],AL	88 04	

	104E	IN	AL, D0H	E4 D0	
	1050	AND	AL,0FH	80 E0 0F	
	1053	INC	SI	46	
	1054	MOV	[SI],AL	88 04	
	1056	IN	AL, C0H	E4 C0	
	1058	AND	AL,0FH	80 E0 0F	
	105B	INC	SI	46	
	105C	MOV	[SI],AL	88 04	
OUT_CHECK:	105E	MOV	SI,1320H	C7 C6 20 13	
	1062	MOV	AL,[SI]	8A 04	
	1064	OUT	E0,AL	E6 E0	
	1066	INC	SI	46	
	1067	MOV	AL,[SI]	8A 04	
	1069	OUT	F0,AL	E6 F0	
	106B	INC	SI	46	
	106C	MOV	AL,[SI]	8A 04	
	106E	OUT	E2,AL	E6 E2	
	1070	INC	SI	46	
	1071	MOV	AL,[SI]	8A 04	
	1073	OUT	F2,AL	E6 F2	
	1075	INC	SI	46	
	1076	MOV	AL,[SI]	8A 04	
	1078	OUT	E4,AL	E6 E4	
	107A	INC	SI	46	
	107B	MOV	AL,[SI]	8A 04	
	107D	OUT	F4,AL	E6 F4	
	107F	JMP	L1	E9 A9 FF	
	1082	ENDS			

Observation:

Input

1200	00
1201	00
1202	00
1203	00
1204	00

Output:

Time is displayed in the RTC board as

	Hour		Minutes		seconds	
X	0	0	0	5	9	
X	0	0	1	0	0	

RESULT:

Thus the digital clock program has been written and executed using 8086 microprocessor kit and the output of digital clock was displayed as [hours: minutes: seconds] successfully.

VIVA QUESTIONS AND ANSWERS

1. **What type of RTC kit is used?**
   DS1307

2. **What is the format of time being displayed?**
   HH:MM:SS

3. **What are the different functionalities of RTC kit?**
   The purpose of an **RTC** or a **real time clock** is to provide precise time and date which can be used for various applications

4. **Whether 7 segment display used here is common anode or common cathode type.**

   common anode type 7 segment display

5. What are the addresses of hour, minute and seconds register?

Bit 6 of the hours register is defined as the 12- or 24-hour mode select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24-hour mode, bit 5 is the second 10 hour bit (20- 23 hours).

11. KEY BOARD AND DISPLAY

AIM:-

To write an assembly language program to interfacing of 8279 with 8086.

APPARATUS REQUIRED:-

- 8086 Microprocessor kit
- 8279 interface board

ALGORITHM:-

(a) Rolling Display

Step1: Start the process

Step2: Initialize lookup table pointer, counter of keyboard display mode of 8279.

Step3: Initialize the prescalar counter and clear the display.

Step4: Get the seven segment display & carried it, in display RAM.

Step5: Increment the look up table pointer.

Step6: Decrement the counter until it becomes zero.

Step7: Stop the process.

(b) Accept a key and display it using 8279

Step1: Start the process

Step2: Set the data to set mode & display

Step3: Initialize the counter and clear the display RAM.

Step4: Write the display RAM command.

Step5: Clear the display RAM.

Step6: Decrement the counter value until it becomes zero.

Step7: Get the key data to be displayed.

Step8: Set the memory to need the FIFO RAM.

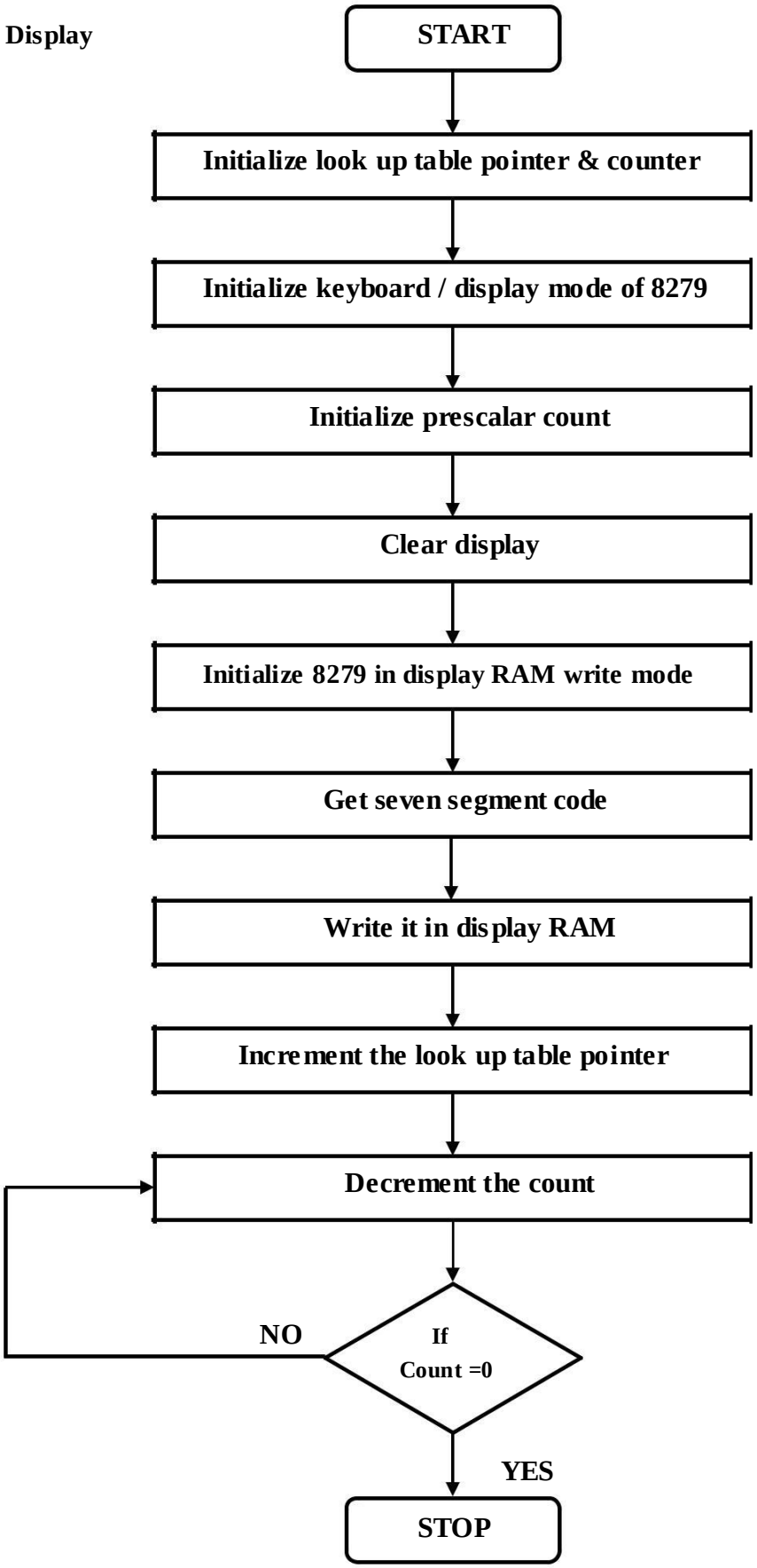
Step9: Get the corresponding code from look up table.

Step10: Store it is necessary.

Step11: Stop the process.

FLOW CHART:

(a) Rolling Display



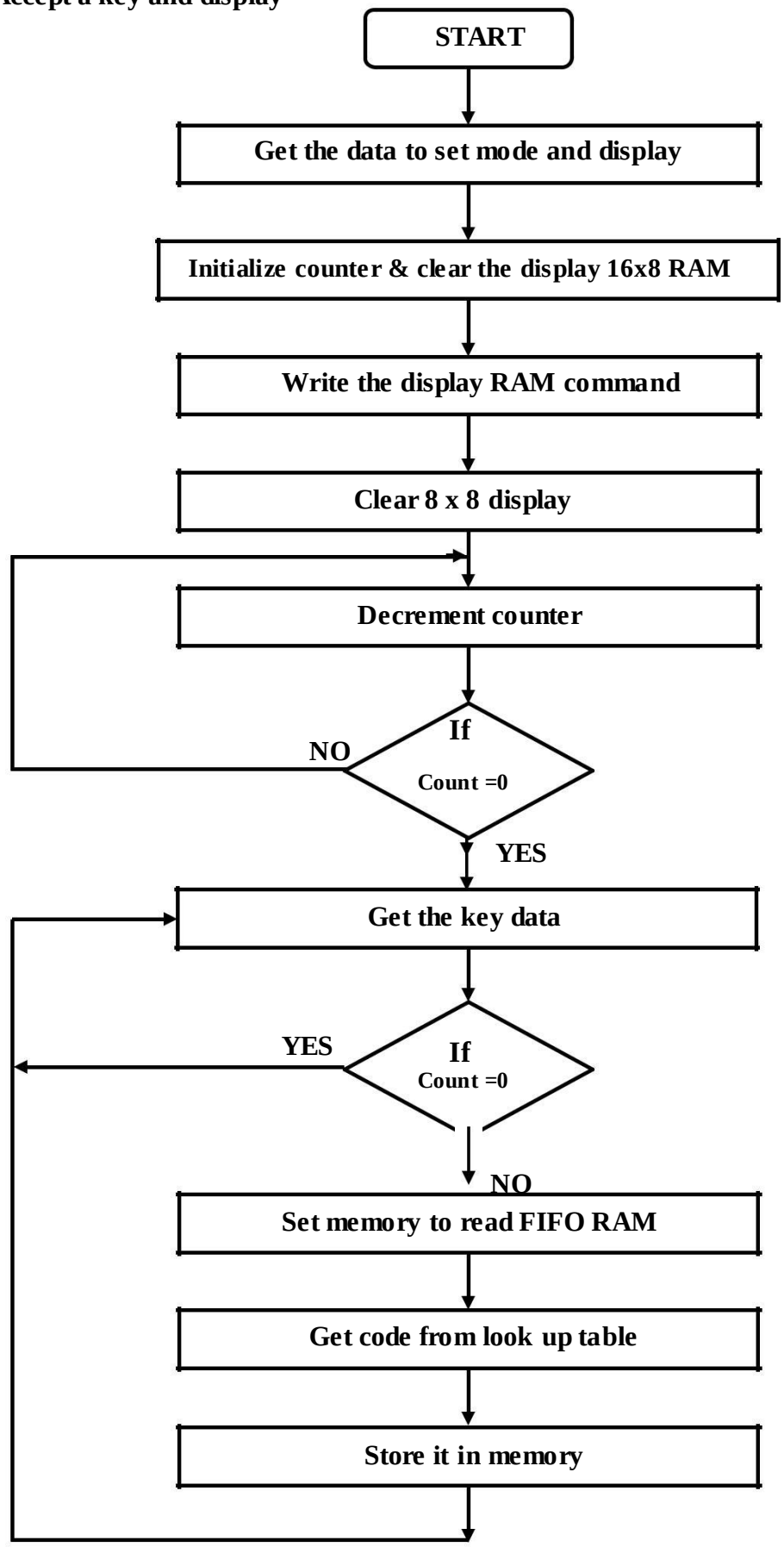
PROGRAM:-

To Display 'A'

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START	1000	MOV	AL,00	C6 C0 00	; Display & keyboard mode set
	1003	OUT	C2,AL	E6 C2	
	1005	MOV	AL,0CC	C6 C0 CC	; Clear Display
	1008	OUT	C2,AL	E6 C2	
	100A	MOV	AL,90	C6 C0 90	; Write display RAM
	100D	OUT	C2,AL	E6 C2	
	100F	MOV	AL,88	C6 C0 88	; Get character
	1012	OUT	C0,AL	E6 C0	
	1014	MOV	AL,0FF	C6, C0 FF	; Blank unused
	1017	MOV	CX,0005	C7 C1 05 00	7segment LED's
NEXT	101B	OUT	C0,AL	E6 C0	
	101D	LOOP	NEXT	E2 FC	
	101F	HLT		F4	; Stop the program

FLOW CHART:-

(b) **Accept a key and display**



PROGRAM:- To Rolling Display (Display message is' HELP US')

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START	1000	MOV	SI,1200	C7 C6 00 12	; load lookup table
	1004	MOV	CX,000F	C7 C1 0F 00	
	1008	MOV	AL,10	C6C010	;Display / keyboard
	100B	OUT	C2,AL	E6 C2	mode set
	100D	MOV	AL,0CC	C6 C0 CC	; Clear Display
	1010	OUT	C2,AL	E6 C2	
	1012	MOV	AL,90	C6 C0 90	; Write display RAM
	1015	OUT	C2,AL	E6 C2	
NEXT:	1017	MOV	AL,[SI]	8A 04	; Get to be displayed
	1019	OUT	C0,AL	E6 C0	character
	101B	CALL	DELAY	E8 E2 04	;Call display program
	101E	INC	SI	46	
	101F	LOOP	NEXT	E2 F6	
	1021	JMP	START	E9 DC FF	;Repeat
DELAY	1500	MOV	DX,0A0FF	C7 C2 FF A0	;Delay program
LOOP1:	1504	DEC	DX	4A	
	1505	JNZ	LOOP1	75 FD	
	1507	RET		C3	

LOOK – UP – TABLE ("HELP US")					
1200	**1201**	**1202**	**1203**	**1204**	**1205**
FF	FF	FF	FF	FF	FF
1206	**1207**	**1208**	**1209**	**120A**	**120B**
FF	FF	98	68	7C	C8
120C	**120D**	**120E**	**120F**		
FF	1C	29	FF		

RESULT:-

Thus the assembly language program for interfacing 8279 keyboard and display controller with 8086 microprocessor trainer kit was executed and successfully verified.

VIVA QUESTIONS AND ANSWERS

1. Give some examples of input devices to microprocessor-based system.
The input devices used in the microprocessor-based system are Keyboards, DIP switches, ADC, Floppy disc, etc.

2. What are the tasks involved in keyboard interface?
The tasks involved in keyboard interfacing are sensing a key actuation, debouncing the key and generating key codes (Decoding the key). These task are performed software if the keyboard is interfaced through ports and they are performed by hardware if the keyboard is interfaced through 8279.

3. How a keyboard matrix is formed in keyboard interface using 8279?
The return lines, RLo to RL7 of 8279 are used to form the columns of keyboard matrix. In decoded scan the scan lines SLo to SL3 of 8279 are used to form the rows of keyboard matrix. In encoded scan mode, the output lines of external decoder are used as rows of keyboard matrix.

4. What is scanning in keyboard and what is scan time?
The process of sending a zero to each row of a keyboard matrix and reading the columns for key actuation is called scanning. The scan time is the time taken by the processor to scan all the rows one by one starting from first row and coming back to the first row again.

5. What is scanning in display and what is the scan time?
In display devices, the process of sending display codes to 7 -segment LEDs to display the LEDs one by one is called scanning (or multiplexed display). The scan time is the time taken to display all the 7-segment LEDs one by one, starting from first LED and coming back to the first LED again.

12. PRINTER STATUS

AIM:

To write an assembly language program to print a message in printer using VBMB – 005

APPARATUS REQUIRED:

1. 8086 Microprocessor kit,
2. Power supply,
3. VBMB005 interfacing board.
4. Printer

(LOOK – UP - TABLE) ROUTINE TO INITIALISE PRINTER						
1500	1501	1502	1503	1504	1505	1506
1B	**47**	**09**	**09**	**09**	**1B**	**0E**
1507	1508	1509	150A	150B	150C	150D
56	**69**	**20**	**4D**	**69**	**63**	**72**
150E	150F	1510	1511	1512	1513	1514
6F	**73**	**79**	**73**	**74**	**65**	**6D**
1515	1516	1517	1518	1519	151A	151B
73	**0A**	**0A**	**09**	**09**	**09**	**09**
151C	151D	151E	151F	1520	1521	1522
1B	**78**	**01**	**44**	**45**	**4D**	**4F**
1523	1524	1525	1526	1527	1528	1529
20	**4F**	**46**	**0A**	**0A**	**09**	**09**
152A	152B	152C	152D	152E	152F	1530
09	**1B**	**78**	**00**	**1B**	**45**	**1B**
1531	1532	1533	1534	1535	1536	1537
47	**43**	**45**	**4E**	**54**	**52**	**4F**
1538	1539	153A	153B	153C	153D	153E
4E	**49**	**43**	**53**	**20**	**50**	**52**
153F	1540	1541	1542	1543	1544	1545
49	**4E**	**54**	**45**	**52**	**20**	**49**
1546	1547	1548	1549	154A	154B	154C
00	**4E**	**54**	**45**	**52**	**46**	**41**
154D	154E	154F	1550	1551	1552	1553
43	**45**	**20**	**42**	**4F**	**41**	**52**
1554	1555	1556	1557	1558	1559	155A
44	**2E**	**1B**	**48**	**1B**	**46**	**END**

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	1000	MOV	CL,59H	C6 C1 59	
	1003	MOV	SI,1500H	C7 C6 00 15	
	1007	MOV	AL,05	C6 C0 05	
	100A	OUT	(CONTL)D0,AL	E6 D0	
	100C	IN	AL,(STAT)C0	E4 C0	
	100E	AND	AL,20H	80 E0 20	
	1011	CMP	AL,20H	80 F8 20	
	1014	JNZ	ERR	75 3B	
PROCEED:	1016				
	1016	MOV	AL,[SI]	8A 04	
	1018	CALL	PRINT ;	E8 0B 00	
	101B	INC	SI	46	
	101C	DEC	CL	FE C9	
	101E	JNZ	PROCEED	75 F6	
	1020	MOV	AL,0AH	C6 C0 6A	
	1023	CALL	PRINT	E8 00 00	
PRINT:	1026				
	1026	MOV	BL,AL	88 C3	
	1028	CALL	CHECK	E8 12 00	
STATUS:	102B				
	102B	MOV	AL,BL	88 D8	
	102D	OUT	(DATA)C8,AL	E6 C8	
	102F	MOV	AL,01	C6 C0 01	
	1032	OUT	(CONTL)D0,AL	E6 D0	
	1034	NOP		90	
	1035	NOP		90	
	1036	NOP		90	
	1037	MOV	AL,05H	C6 C6 05	
	103A	OUT	(CONTL)D0,AL	E6 D0	
	103C	RET		C3	
CHECK:	103D				
	103D	IN	AL,(STAT)C0	E4 C0	
	103F	AND	AL,20H	80 E0 20	
	1042	JZ	CHECK	74 F9	
	1044	IN	AL,(STAT)C0	E4 C0	
	1046	AND	AL,80H	80 E0 80	
	1049	CMP	AL,80H	80 F8 80	
	104C	JNZ	STATUS	75 DD	
	104E	JMP	CHECK	E9 EC FF	
ERR:	1051				
	1051	INT	2	CD 02	

RESULT:

Thus the given message was printed in the printer using 8086 Microprocessor kit and VBMB – 005.

VIVA QUESTIONS AND ANSWERS

1. **Which interrupt subroutine is used to return printer status?**
   We use the INT (interrupt) instruction to call system routines; on completion, an interrupt routine executes an IRET (interrupt return) ... to the printer; Function 1: initializes a printer port; Function 2: gets printer status

2. **Explain ROL instruction**
   Rotate accumulator left

3. **Explain Printer Port**.
   Printer port. A printer port is a female connector, or port, on the back of a computer that allows it to interact with a printer. These ports enable users to send documents and pictures to a printer.

4. **What is meant by Return Instruction**?
   The Return instruction is used to return to the Main Program from a Subroutine Program or Interrupt Program. The Return instruction can be Conditional or Unconditional

5. **Differentiate CMP and SUB Instructions.**
   The main difference between cmp and sub is that cmp does not store the result of the subtract operation; it performs subtraction only to set the status flags.

13. SERIAL INTERFACE AND PARALLEL INTERFACE

13a) SERIAL INTERFACE

AIM:

To write a program to send byte value from one microprocessor kit to other kit in serial method by using 8251.

PROCEDURE:

1. Take two no of 8086 microprocessor kits.
2. Enter the Transmit program in Transmitter kit.
3. Enter the receive program in receiver kit.
4. Interface the two kits with 9-9 serial cable in the serial port of the microprocessor kits.
5. (LCD kit means pc-pc cable; LED kit means kit-kit cable)
6. Enter the Baud rate in Transmitter and the receiver kit
7. Enter the data in Transmitter kit use the memory location 1500.
8. Execute the receiver kit.
9. Execute the Transmitter kit.
10. Result will be available in receiver kit memory location 1500.

PROGRAM: TRANSMITTER SECTION:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
	1000	MOV	SI,1500H	C7,C6,00,15	: MOVE 1500 to SI register
	1004	MOV	AL,36H	C6,C0,36	: MOV 36, to AL register
	1007	OUT	16H,AL	E6,16	
	1009	MOV	AL,40H	C6,C0,40	: MOVE 40 to AI register
	100C	OUT	10H,AL	E6,16	: MOV 01 to AL
	100E	MOV	AL,01H	C6,C0,01	
	1011	OUT	10H,AL	E6,10	
RELOAD	1013	MOV	CL,05H	C6,C1,05	: MOV 05, to CL register
CHECK	1016	IN	AL,0AH	E4,04	
	1018	AND	AL,04H	80,E0,04	
	101B	JZ	CHECK	74,79	:JUMP Check
	101D	MOV	AL,[SI]	8A,04	:MOV SI to AL
	101F	OUT	08H,AL	E6,08	
	1021	INC	SI	46	
	1022	CMP	AL,3FH	80,F8,8F	
	1025	JNZ	RELOAD	75,F0	: JUMP on No – zero reload
	1027	DEC	CL	FE,C9	
	1029	JNZ	CHECK	75,EB	:JUMP Check
	102B	INT	02	CD,02	:INT the 02

Transmitter	
Address	**Input**
1500	**01**
1501	**02**
1502	**03**
1503	**04**
1504	**05**

Receiver	
Address	**Output**
1500	**01**
1501	**02**
1502	**03**
1503	**04**
1504	**05**

RECEIVER SECTION:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
	1000	MOV	SI,1500H	C7,C6,00,15	: MOVE 1500 to SI register
	1004	MOV	AL,36H	C6,C0,36	
	1007	OUT	16H,AL	E6,16	: MOV AL to 16H
	1009	MOV	AL,40H	C6,C0,40	
	100C	OUT	10H,AL	E6,16	
	100E	MOV	AL,01H	C6,C0,01	: MOVE 01 to AI register
	1011	OUT	10H,AL	E6,10	
RELOAD	1013	MOV	CL,05H	C6,C1,05	: MOV 05, to CL register
CHECK	1016	IN	AL,0AH	E4,04	
	1018	AND	AL,02H	80,E0,02	:AND the Alto 02
	101B	JZ	CHECK	74,79	:JUMP the Check
	101D	IN	AL,08	E4,08	
	101F	MOV	[SI],AL	88,04	
	1021	INC	SI	46	:Increment the SI
	1022	CMP	AL,3FH	80,F8,8F	
	1025	JNZ	RELOAD	75,EC	: JUMP reload
	1027	DEC	CL	FE,C9	:Decrement CL
	1029	JNZ	CHECK	75,EB	:JUMP Check
	102B	INT	02	CD,02	
	102D	INT		CD,02	

RESULT:

Thus the program to send byte value from one microprocessor kit to other kit in serial method by using 8251has been successfully verified.

VIVA QUESTIONS AND ANSWERS

1. What is baud rate?

The baud rate is the rate at which the serial data are transmitted. Baud rate is defined as l /(The time for a bit cell). In some systems one bit cell has one data bit, then the baud rate and bits/sec are same.

2. What is USART?

The device which can be programmed to perform Synchronous or Asynchronous serial communication is called USART (Universal Synchronous Asynchronous Receiver Transmitter). The INTEL 8251A is an example of USART.

3. What are the functions performed by INTEL 8251A?

The INTEL 8251lA is used for converting parallel data to serial or vice versa. The data transmission or reception can be either asynchronously or synchronously. The 8251A can be used to interface MODEM and establish serial communication through MODEM over telephone lines.

4. What is an Interrupt?

Interrupt is a signal send by an external device to the processor so as to request the processor to perform a particular task or work.

5. What are the control words of 8251A and what are its functions?

The control words of 8251A are Mode word and Command word.

The mode word informs 8251 about the baud rate, character length, parity and stop bits. The command word can be send to enable the data transmission and reception.

6. What are the information that can be obtained from the status word of 8251?

The status word can be read by the CPU to check the readiness of the transmitter or receiver and to check the character synchronization in synchronous reception. It also provides information regarding various errors in the data received. The various error conditions that can be checked from the status word are parity error, overrun error and framing error.

13b). PARALLEL COMMUNICATION BETWEEN TWO 8086 MICROPROCESSORS KITS

AIM:

To write a program to send data from one microprocessor kit to other kit in parallel method by using mode1 and mode2 of 8255.

PROCEDURE:

1. Take two 8086 microprocessor kits.
2. Enter the transmitter program in transmitter kit.
3. Enter the receiver program in receiver kit.
4. Interface the two kits with 26-core cable on PPI-1.
5. Execute the receiver kit.
6. Execute the transmitter kit.
7. Go and see the memory location 1200 in receiver your getting 8 same data's.
8. Data is available in transmitter kit the memory location is 100f.
9. We will change the data & execute the following procedure & get the result in receiver kit.

PROGRAM: TRANSMITTER PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
	1000	MOV	AL,82H	C7,C0,82	: MOVE 82 to AL register
	1003	OUT	26H,AL	E6,26	
	1005	MOV	AL,3FH	C6,C0,3F	: MOV 3F to AL register
	1008	OUT	20H,AL	E6,20	
LOOP	100A	IN	AL,22H	E4,22	: MOVE 22H to AI register
	100C	SUB	AL,3FH	80,E8,3F	
	100F	JNZ	LOOP	75 F9	:JUMP LOOP
	1011	MOV	AL,24H	C6,C0,24	
	1014	OUT	20H,AL	E6,20	
	1016	CALL	DELAY	E8,02,00	:Call delay
	1019	INT	02	C0,02	
DELAY	101B	MOV	BL,05H	C6,C3,FF	
LION	101E	MOV	DL,0FFH	C6,C2,FF	
LOOP2	1021	DEC	DL	FF,CA	:Decrement CL
	1023	JNZ	LOOP2	75,F9	:JUMP Loop2
	1025	DEC	BL	FE,CB	
	1027	JNZ	LION	75,F4	:Jump on no zero to lion
	1029	RET		C3	

Receiver	
Address	**Output**
1200	**24**
1201	**24**
1202	**24**
1203	**24**
1204	**24**
1205	**24**
1206	**24**
1207	**24**

RECEIVER PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
	1000	MOV	AL,90H	C7,C0,90	: MOVE 90 to AL register
	1003	OUT	26H,AL	E6,26	: MOV 20 to AL register
CHECK	1005	IN	Al,20H	E4,20	
	1007	SUB	AL,3FH	80,E8,3F	: MOVE 3FH to AI register
	100A	JNZ	CHECK	75F9	
	100C	MOV	AL,3FH	C6,C0,3F	
	100F	OUT	22H,AL	E6,22	
	1011	MOV	Cl, 08	C6,C1,08	: MOVE 08H to AI register
	1014	CALL	DELAY	E8,12,00	
	1017	MOV	SI, 1200	C7,C8,00,12	
LOOP1	101B	IN	AL,20H	E4,20	
	101D	MOV	[SI], AL	88,04	
	100F	CALL	DELAY	E8,07,00	:Call delay
	1022	INC	SI	46	
	1023	DEC	CL	FE,C9	
	1025	JNZ	LOOP1	75,F4	
	1027	INT	02	CD,02	
DELAY	1029	MOV	BL, 05H	C6,C3,05	
LION	102C	MOV	DL, 0FFH	C6,C2,FF	
LOOP2	102F	DEC	DL	FE,CA	:Decrement CL
	1031	JNZ	LOOP2	75,FC	;JUMP LOOP2
	1033	DEC	BL	FE,CB	:Decrement BL
	1035	JNZ	LION	75,F5	:JUMP lion
	1037	RET		C3	
	1038	RET		C3	

RESULT;

Thus the program to sent data in parallel method from one microprocessor kit to another using 8255 has been verified successfully.

VIVA QUESTIONS AND ANSWERS

1. Give some examples of port devices used in 8085 microprocessor based system?

The various INTEL I/O port devices used in 8085 microprocessor based system are 8212, 8155, 8156, 8255, 8355 and 8755.

2. Write a short note on INTEL 8255?

The INTEL 8255 is a I/O port device consisting of 3 numbers of 8 –bit parallel I/O ports. The ports can be programmed to function either as a input port or as a output port in different operating modes. It requires 4 internal addresses and has one logic LOW chip select pin.

3. What is the drawback in memory mapped I/0?

When I/O devices are memory mapped, some of the addresses are allotted to I/O devices and so the full address space cannot be used for addressing memory (i.e., physical memory address space will be reduced). Hence memory mapping is useful only for small systems, where the memory requirement is less.

4. How DMA is initiated?

When the I/O device needs a DMA transfer, it will send a DMA request signal to DMA controller. The DMA controller in turn sends a HOLD request to the processor. When the processor receives a HOLD request, it will drive its tri-stated pins to high impedance state at the end of current instruction execution and send an acknowledge signal to DMA controller. Now the DMA controller will perform DMA transfer.

5. What is processor cycle (Machine cycle)?

The processor cycle or machine cycle is the basic operation performed by the processor. To execute an instruction, the processor will run one or more machine cycles in a particular order.

6. What is Instruction cycle?

The sequence of operations that a processor has to carry out while executing the instruction is called Instruction cycle. Each instruction cycle of a processor indium consists of a number of machine cycles.

14. A/D AND D/A INTERFACE AND WAVEFORM GENERATION

14a) A/D INTERFACE WITH 8086

AIM:-

To write an assembly language program for interfacing of ADC with 8086.

ALGORITHM:-

(ii) Start the program

(iii) Select channel 0 (CH – 0)

(iv) Make ALE low (ALE = 10)

(v) Make ALE high (ALE = 18)

(vi) Male ALE low (ALE = 10)

(vii) Stop the program

PROCEDURE:

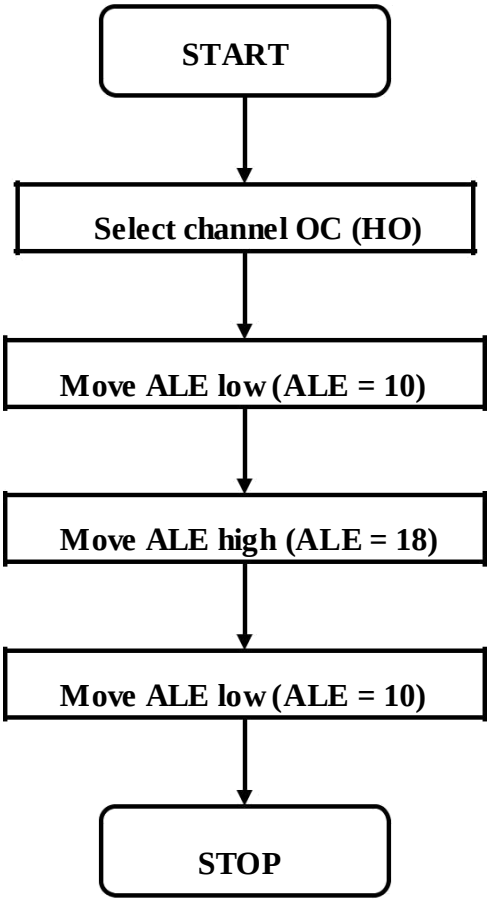
(i) Place jumper J2 in C position

(ii) Place jumper J5 in A position

(iii) Enter and execute the program

(iv) Vary the analog input (using trim pot) and view the corresponding digital value in LED display,

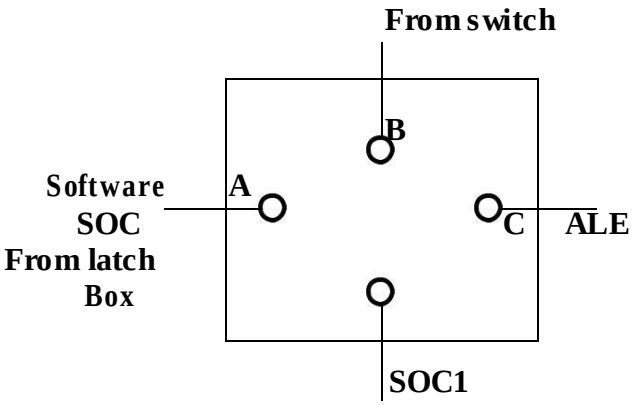
FLOW CHART:


START
Select channel OC (HO)
Move ALE low (ALE = 10)
Move ALE high (ALE = 18)
Move ALE low (ALE = 10)
STOP

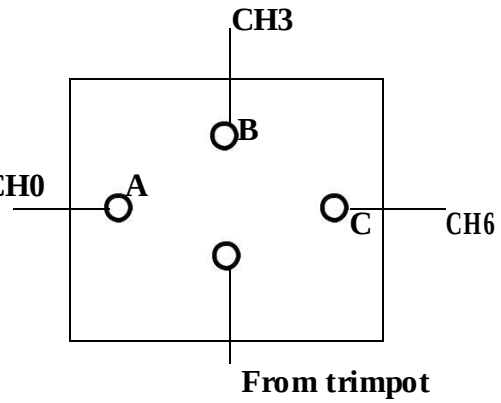
PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
	1000	MOV	AL, 10	C6 C0 10	; Channel selection
	1003	OUT	C8, AL	E6 C8	; ALE low
	1005	MOV	AL, 18	C6 C0 18	; ALE high
	1008	OUT	C8, AL	E6 C8	
	100A	MOV	AL,10	C6 C0 10	; ALE low
	100D	OUT	C8,AL	E6 C8	
	100F	HLT	F4		; Stop

Jumper Details:-



J_2 [SOC Jumper Selection for CHO – CH7]



J_5 [Provision to correct the trimpot to any of mentioned channel]

RESULT:

Thus the assembly language program for performing the interfacing of ADC with 8086 has been done verified.

14b. INTERFACING OF DAC WITH 8086

AIM:-

To write an assembly language program to generate square, triangular and saw tooth and waveform by interfacing of DAC with 8086.

ALOGRITHM:

(a) Square Wave:-

- Initially the output value is predefined high value and after some time, the delayed output value becomes lower and stays in that position for some time delay.
- Initialize the accumulator and display it.
- Using delay program, the output is displayed from '00' value.
- Increment the value up to 'FF' and display it for high value.
- Using repeat instruction the square waveform is obtained.

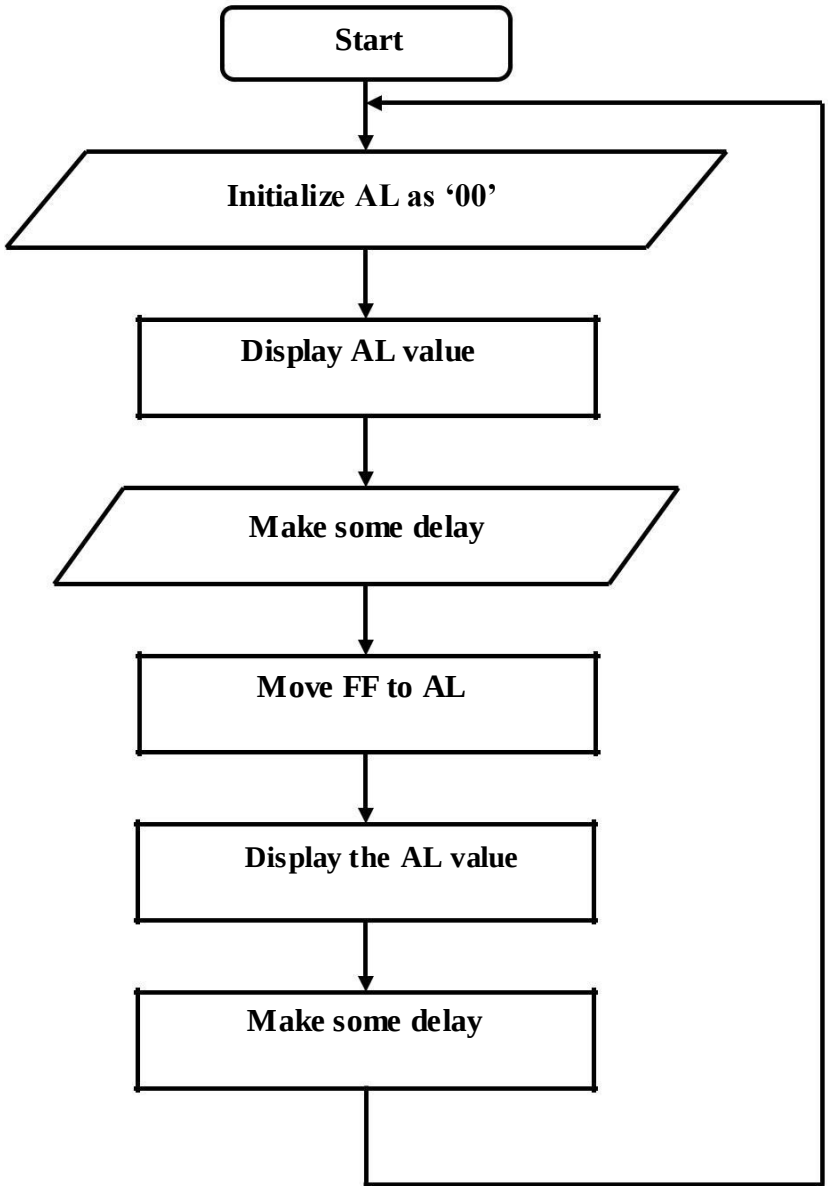
(c) Saw tooth Waveform:-

- Initialize the accumulator to'00' values.
- Display this value in C0
- Increment the accumulator up to 'FF'
- Suddenly it is sent to 00 and repeats the process.

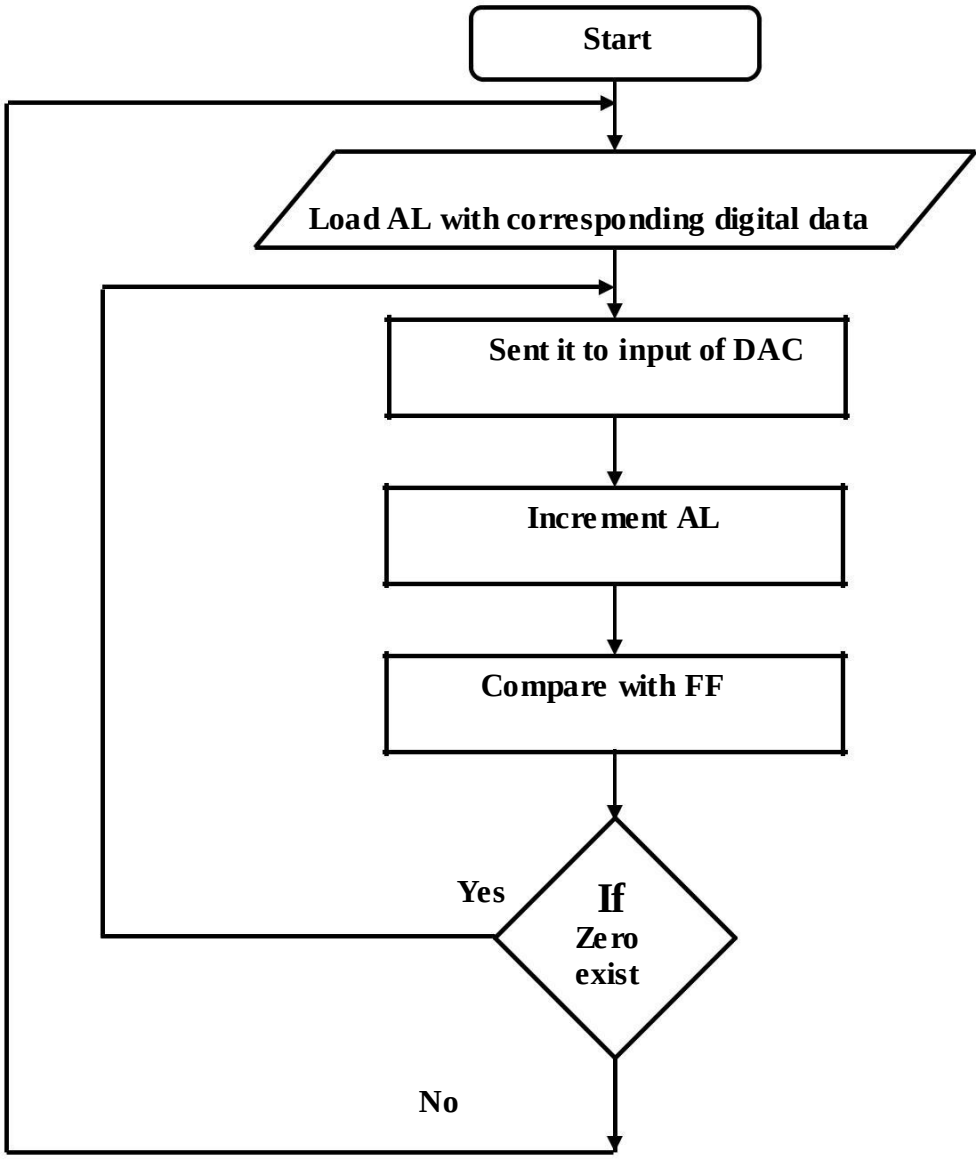
(d) Triangular Waveform:-

- Initialize the accumulator to'00'
- Out the results in 'C8'
- Increment the accumulator up to the value of FF and display it.
- Decrement the accumulator to'00' and then display it.
- Repeat the procedure for continuous waveform.

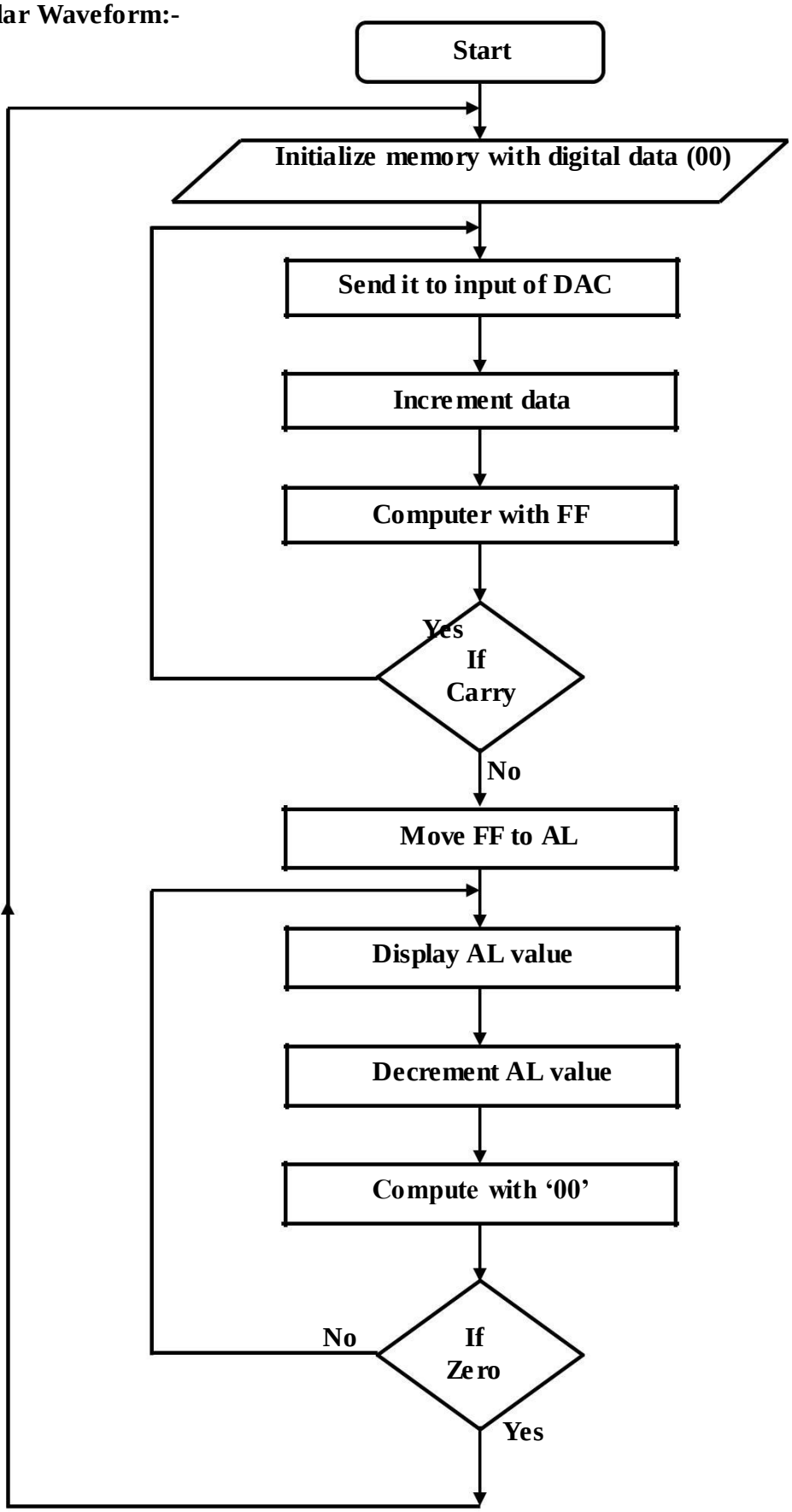
(a) Square Waveform:-



(b) Saw tooth Waveform:-



(c) Triangular Waveform:-



(a) To generate square waveform

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
LOOP2 :	1000	MOV	AL, 00	C6 C0 00	; Set Logic 0 level
	1003	OUT	C8, AL	E6 C8	
	1005	CALL	Delay	E8 0B 00	;Generate timing delay
	1008	MOV	AL,0FF	C6 C0 FF	;Set logic 1 level
	100B	OUT	C8, AL	E6 C8	
	100D	CALL	Delay	E8 03 00	; Generate timing delay
	1010	JMP	LOOP2	E9 ED FF	:Repeat to generates Square Wave
Delay:	1013	MOV	CX, 05FF	C7 C1 FF 05	:Delay Program
LOOP3:	1017	LOOP	LOOP3	E2 FE	
	1019	RET		C3	

(b) To generate saw tooth wave

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
LOOP2 :	1000	MOV	AL, 00	C6 C0 00	; Set logic 0 level
LOOP1:	1003	OUT	C0, AL	E6 C0	
	1005	INC	AL	FE C0	;Increment Logic0 toLogic1
	1007	JNZ	LOOP1	75 FA	;If ZF=0, jump to next
	1009	JMP	LOOP2	E9 F4 FF	;Repeat

(c) To generate triangular waveform

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
LOOP3 :	1000	MOV	BL, 00	C6 C3 00	;Set logic 0
LOOP1:	1003	MOV	AL, BL	88 D8	;copy logic 0
	1005	OUT	C8, AL	E6 C8	
	1007	INC	BL	FE C3	; Increment logic0 to logic1
	1009	JNZ	LOOP1:	75 F8	; If ZF=0, jump to next
	100B	MOV	BL, 0FF	C6 C3 FF	Set logic 1
LOOP2:	100E	MOV	AL, BL	88 D8	;copy logic 1
	1010	OUT	C8, AL	E6 C8	
	1012	DEC	BL	FE CB	; Decrement logic0 tologic1
	1014	JNZ	LOOP2	75 F8	; If ZF=0, jump to next
	1016	JMP	LOOP3	E9 E7 FF	;Repeat

RESULT:

Thus an assembly language program to generate square, triangular and saw tooth waveform was done using DAC Interface and 8086 microprocessor kit.

VIVA QUESTIONS AND ANSWERS

1. What are the internal devices of a typical DAC?

The internal devices of a DAC are R/2R resistive network, an internal latch and current to voltage converting amplifier.

2. What is settling or conversion time in DAC?

The time taken by the DAC to convert a given digital data to corresponding analog signal is called conversion time.

3. What are the different types of ADC?

The different types of ADC are successive approximation ADC, counter type ADC flash type ADC, integrator converters and voltage- to-frequency converters.

4. Define stack

Stack is a sequence of RAM memory locations defined by the programmer.

5. What is program counter? How is it useful in program execution?

The program counter keeps track of program execution. To execute a program the starting address of the program is loaded in program counter. The PC sends out an address to fetch a byte of instruction from memory and increments its content automatically.

6. How the microprocessor is synchronized with peripherals?

The timing and control unit synchronizes all the microprocessor operations with clock and generates control signals necessary for communication between the microprocessor and peripherals.

15. BASIC ARITHMETIC AND LOGIAL OPERATIONS USING 8051

A. 8 BIT ADDITION

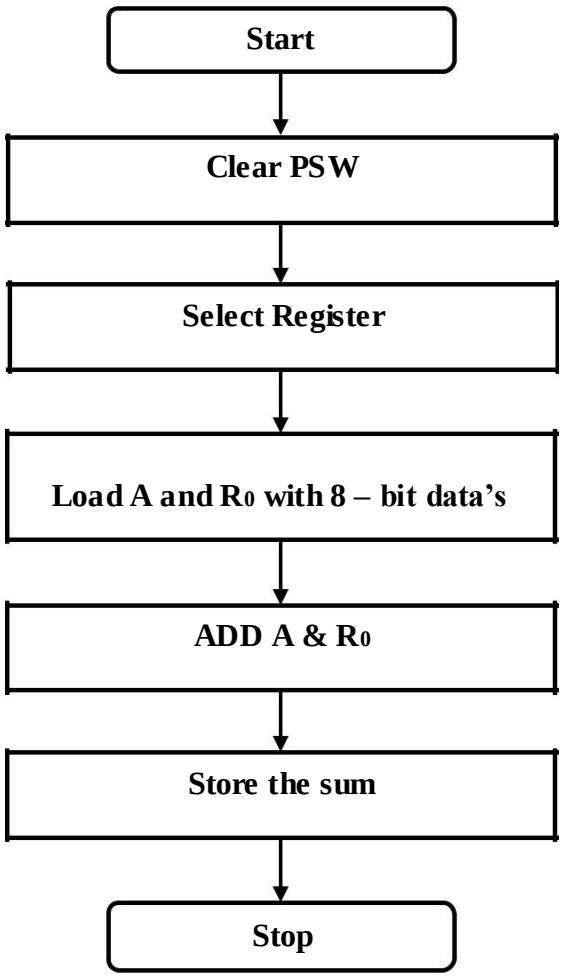
AIM:

To write a program to add two 8-bit numbers using 8051 microcontroller.

ALGORITHM:

1. Clear Program Status Word.
2. Select Register bank by giving proper values to RS1 & RS0 of PSW.
3. Load accumulator A with any desired 8-bit data.
4. Load the register R 0 with the second 8- bit data.
5. Add these two 8-bit numbers.
6. Store the result.
7. Stop the program.

FLOW CHART:


Start
Clear PSW
Select Register
Load A and R0 with 8 – bit data's
ADD A & R0
Store the sum
Stop

PROGRAM:

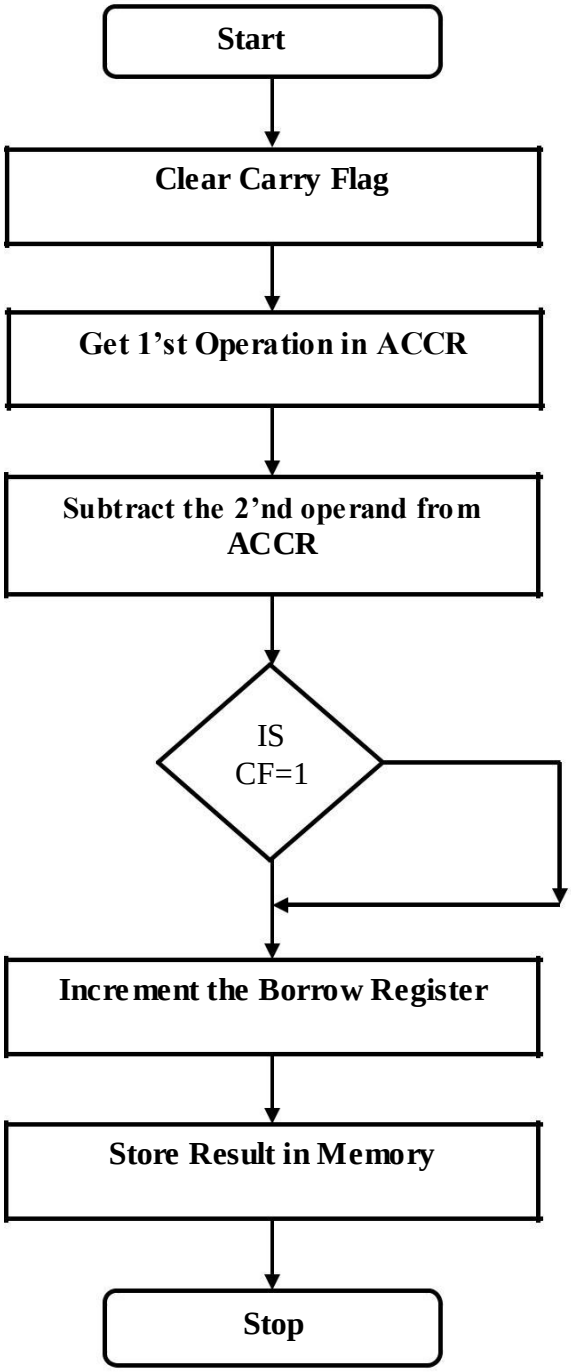
Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	4100	CLR	C	C3	Clear CY Flag
	4101	MOV	A,#0A	74 0A	Get the data1 in Accumulator
	4103	ADDC	A,#10	34 10	Add the data1 with data 2
	4105	MOV	DPTR,#4500	90 45 00	Initialize the memory location
	4108	MOVX	@DPTR,A	F0	Store the result in memory location
L1	4109	SJMP	L1	80 FE	Stop the program

Address	**Output**
4500	**1A(LSB)**
4501	**00(MSB)**

RESULT:

Thus the 8051 Assembly Language Program for addition of two 8 bit numbers was executed.

FLOW CHART:


Start
Clear Carry Flag
Get 1'st Operation in ACCR
Subtract the 2'nd operand from ACCR
IS CF=1
Increment the Borrow Register
Store Result in Memory
Stop

15B. 8 BIT SUBTRACTION

AIM:

To perform subtraction of two 8 bit data and store the result in memory.

ALGORITHM:

1. Clear the carry flag.
2. Initialize the register for borrow.
3. Get the first operand into the accumulator.
4. Subtract the second operand from the accumulator.
5. If a borrow results increment the carry register.
6. Store the result in memory.

PROGRAM:

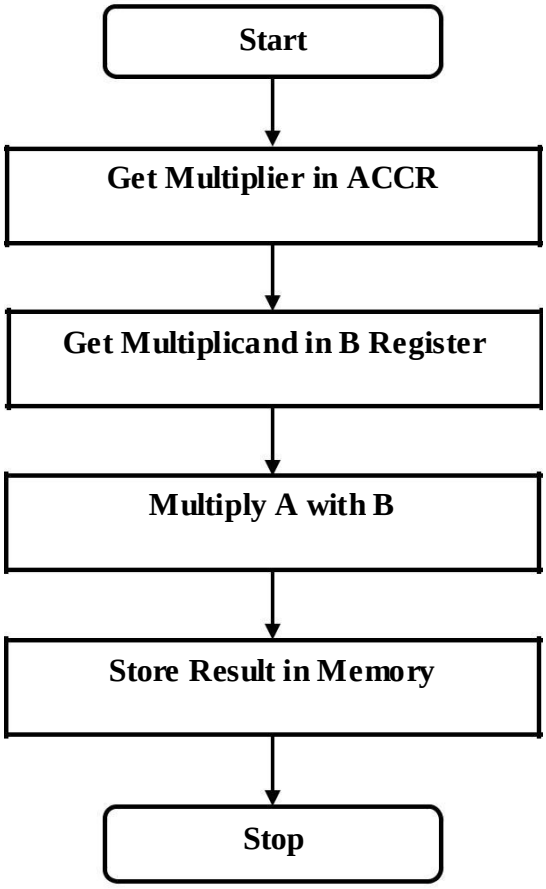
Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	4100	CLR	C	C3	Clear CY Flag
	4101	MOV	A,#0A	74 0A	Get the data1 in Accumulator
	4103	SUBB	A,#05	94 05	Subtract data2 from data1
	4105	MOV	DPTR,#4500	90 45 00	Initialize memory location
	4108	MOVX	@DPTR,A	F0	Store the difference in memory location
L1	4109	SJMP	L1	80 FE	Stop the program

Address	Output
4500	05

RESULT:

Thus the 8051 Assembly Language Program for subtraction of two 8 bit numbers was executed.

FLOW CHART:


Start
Get Multiplier in ACCR
Get Multiplicand in B Register
Multiply A with B
Store Result in Memory
Stop

15 C. 8 BIT MULTIPLICATION

AIM:

To perform multiplication of two 8 bit data and store the result in memory.

ALGORITHM:

1. Get the multiplier in the accumulator.
2. Get the multiplicand in the B register.
3. Multiply A with B.

Store the product in memory

PROGRAM:

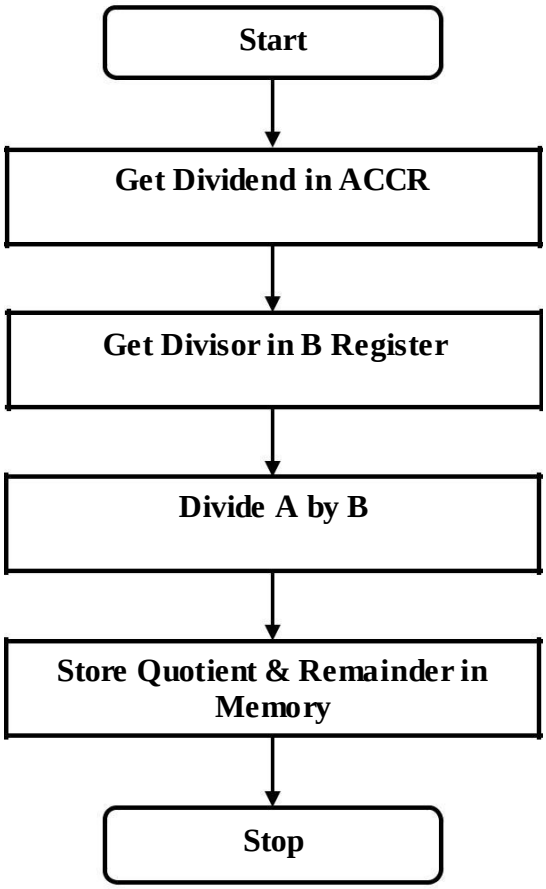
Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	4100	MOV	A,#05	74 05	Store data1 in accumulator
	4102	MOV	B,#03	75 F0 03	Store data2 in B register
	4105	MUL	AB	A4	Multiply both
	4106	MOV	DPTR,#4500	90 45 00	Initialize memory location
	4109	MOVX	@DPTR,A	F0	Store lower order result
	410A	INC	DPTR	A3	Go to next memory location
	410B	MOV	A,B	E5 F0	Store higher order result
	410D	MOVX	@DPTR,A	F0	
L1	410E	SJMP	L1	80 FE	Stop the program

Address	Output
4500	**0F(LSB)**
4501	**00(MSB)**

RESULT:

Thus the 8051Assembly Language Program for multiplication of two 8 bit numbers was executed.

FLOW CHART:


Start
Get Dividend in ACCR
Get Divisor in B Register
Divide A by B
Store Quotient & Remainder in Memory
Stop

15 D. 8 BIT DIVISION

AIM:

To perform division of two 8 bit data and store the result in memory.

ALGORITHM:

1. Get the Dividend in the accumulator.
2. Get the Divisor in the B register.
3. Divide A by B.

Store the Quotient and Remainder in memory

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	4100	MOV	A,#15	74 15	Store data1 in accumulator
	4102	MOV	B,#03	75 F0 03	Store data2 in B register
	4105	DIV	AB	84	Divide
	4106	MOV	DPTR,#4500	90 45 00	Initialize memory location
	4109	MOVX	@DPTR,A	F0	Store remainder
	410A	INC	DPTR	A3	Go to next memory location
	410B	MOV	A,B	E5 F0	Store quotient
	410D	MOVX	@DPTR,A	F0	
L1	410E	SJMP	L1	80 FE	Stop the program

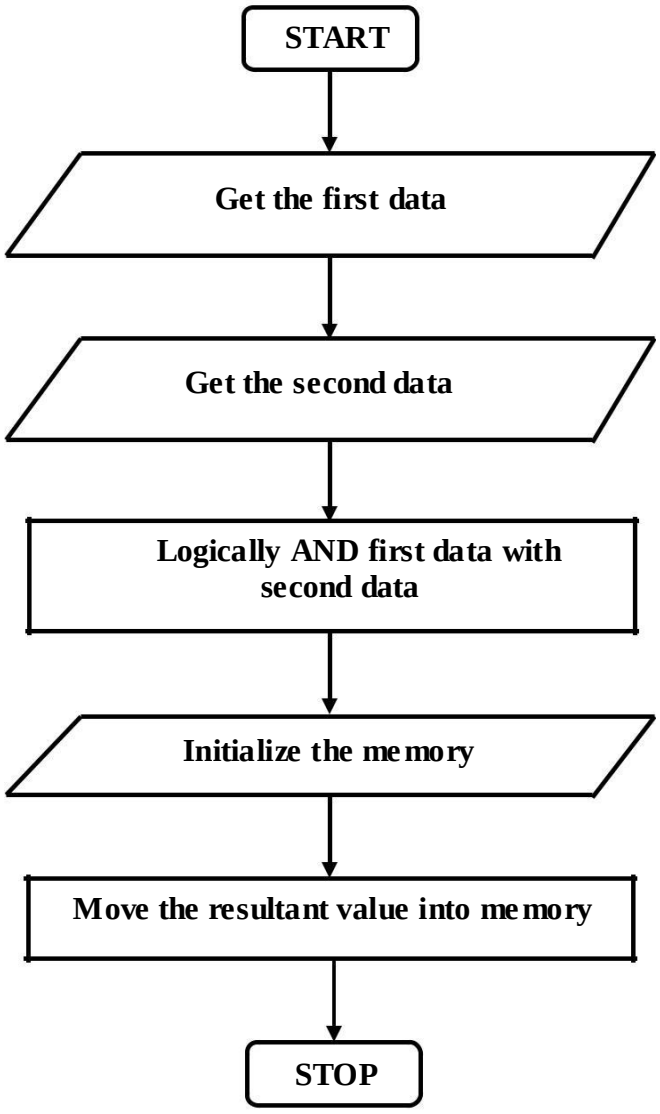
Input	
Memory Location	**Data**
4500 (dividend)	**0F**
4501 (divisor)	**03**

Output	
Memory Location	**Data**
4502 (remainder)	**05**
4503 (quotient)	**00**

RESULT:

Thus the 8051 8051Assembly Language Program for division of two 8 bit numbers was executed.

FLOW CHART:


START
Get the first data
Get the second data
Logically AND first data with second data
Initialize the memory
Move the resultant value into memory
STOP

15 D. MASKING BITS IN AN 8 – BIT NUMBER

AIM:

To write an assembly language program to mask bits o and 7 of an 8 – bit number and store the result in memory using 8051 microcontroller.

APPARATUS REQUIRED:

8051 microcontroller kit

ALGORITHM:

Masking bits in a 8 bit number

- Start the process
- Get the two data values
- Get the second data
- Logically 'AND' the two data values.
- Initialize the memory value and store the result in memory.
- Start the process

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START	4100	MOV	A,#87	74 87	
	4102	ANL	A,#7E	54 7E	
	4104	MOV	DPTR,#4500	90 45 00	
	4107	MOVX	@DPTR,A	F0	
L1	4108	SJMP	L1	80 FE	

Output	
Memory Location	**Data**
4500	**06**

RESULT:

Thus the 8051assembly language program for masking bits was executed and verified.

VIVA QUESTIONS AND ANSWERS

1. **What is DPTR?**

DPTR is a data pointer register in 8051 , which is of 2 bytes(16bits)....DPH,DPL

2. What is the difference between Microprocessor and Microcontroller?

Key difference in both of them is presence of external peripheral, where microcontrollers have RAM, ROM, EEPROM embedded in it while we have to use external circuits in case of microprocessors. ... As all the peripheral of microcontroller are on single chip it is compact while microprocessor is bulky.

3.What is the use of stack pointer?

Stack pointer points the top of stack.

4. What is the use of SJMP?

SJMP jumps unconditionally to the address specified reladdr. Reladdr must be within -128 or +127 bytes of the instruction that follows the SJMP instruction.

5. What is meant by Register Bank?

The 8051 microcontroller has a total of 128 bytes of RAM.
We will discuss about the allocation of these 128 bytes of RAM and examine their usage as stack and register.
... These 32 bytes are divided into four register banks in which each bank has 8 registers, R0–R7.

6.What is the use of Accumulator?

An accumulator is a register for short-term, intermediate storage of arithmetic and logic data in a computer's CPU (central processing unit).

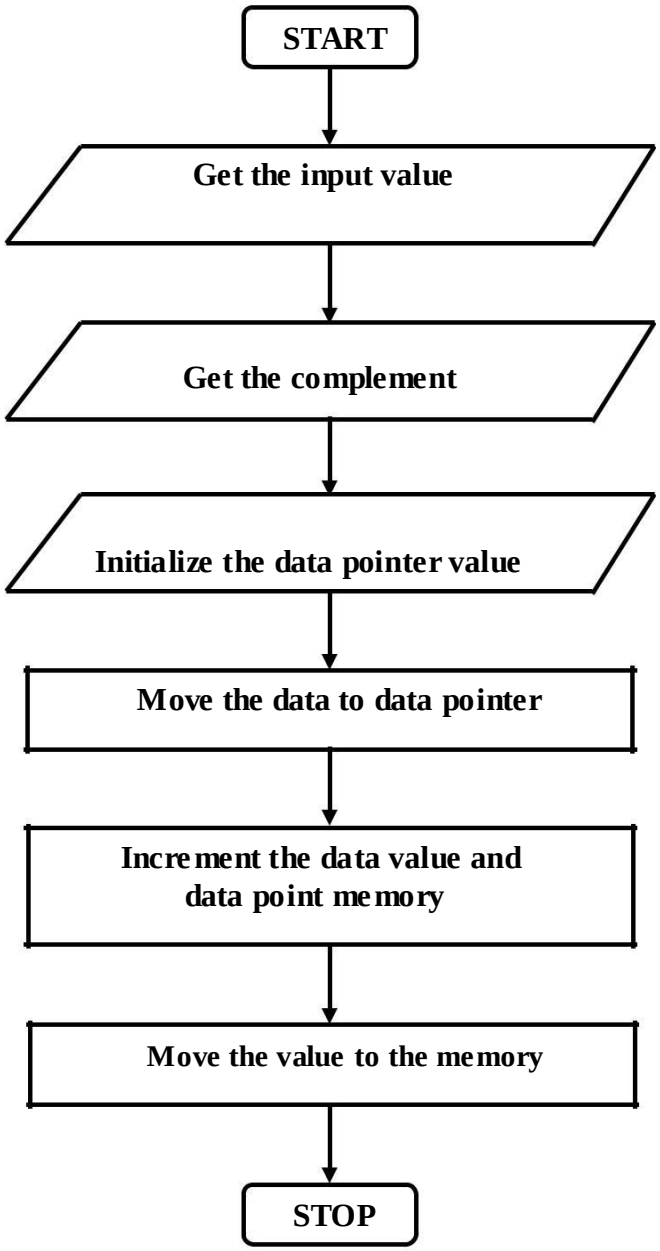
7. What is the use of Interrupt?

An interrupt is a condition that causes the microprocessor to temporarily work on a different task, and then later return to its previous task. Interrupts can be internal or external.

8. What is meant by Port?

The port is a buffered I/O, which is used to hold the data transmitted from the microprocessor to I/O device or vice-versa.

a) 1's and 2's complement



16. SQUARE AND CUBE PROGRAM, FIND 2'S COMPLIMENT OF A NUMBER

AIM:-

To write an assembly language to perform arithmetic, logical and bit manipulation instruction using 8051.

ALOGRITHM:

a) 1's and 2's complement

- Get the value
- Get the complement value of data.
- Initialize the data pointer value as memory.
- Move the complemented value to memory of data pointer.
- Increment the value and memory.
- Store the result in memory.
- Stop the process.

a) 1's and 2's complement

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
	4100	MOV	A, #02	74, 02	Get the initial value
	4102	CPL	A	F4	Complement the value
	4103	MOV	DPTR, # 4200	90, 42, 00	Initialize the memory
	4106	MOVX	@ DPTR, A	F0	Move the data to memory
	4107	INC	A	04	Increment Accumulator
	4108	INC	DPTR	A3	Increment the memory
	4109	MOVX	@ DPTR, A	F0	Move the value to memory
ECE:	410A	SJMP	ECE	80, FE	Continue the process.

1’s and 2’s complement

Output	
Memory Location	**Data**
4200	**FD (1’s complement**
4201	**FE(2’S Complement)**

Square of a number

Input		Output	
Memory Location	**Data**	**Memory Location**	**Data**
4200	**89**	**4201**	**51**
		4202	**49**

b) **SQUARE PROGRAM FOR 8051**

```
$MOD51
ORG 4100H
MOV DPTR,#4200H
MOVX A,@DPTR
MOV B,A
MUL AB
INC DPTR
MOVX @DPTR,A
INC DPTR
MOV A,B
MOVX @DPTR,A
L:SJMP L
```

****

C). CUBE PROGRAM FOR 8051

```
$MOD51
 ORG 4100H
 MOV DPTR,#4200H
 MOVX A,@DPTR
 MOV B,A
 MOV R7,A
 MUL AB
 MOV R0,A
 MOV R1,B
 MOV A,R7
 ANL A,#0FH
 MOV B,A
 MOV A,R0
 MUL AB
 MOV R2,A
 MOV R3,B
 MOV A,R1
 MOV B,A
 MOV A,R7
 ANL A,#0FH
 MUL AB
 MOV R4,A
 MOV R5,B
 MOV A,R3
 MOV B,R4
 ADD A,B
 MOV R3,A
```

```
MOV A,R0
MOV B,A
MOV A,R7
ANL A,#0F0H
SWAP A
MUL AB
MOV R4,A
MOV R6,B
MOV A,R7
ANL A,#0F0H
SWAP A
MOV B,R1
MUL AB
MOV R0,A
MOV R1,B
MOV A,R6
MOV B,R0
ADD A,B
MOV R0,A
MOV R7,A
MOV A,R4

SWAP A
ANL A,#0F0H
MOV R6,A
MOV A,R0
SWAP A
ANL A,#0F0H
MOV R0,A
MOV A,R4
SWAP A
ANL A,#0FH
MOV R4,A
MOV B,R0
ADD A,B
MOV R4,A
MOV A,R1
SWAP A
MOV B,A
MOV A,R7
SWAP A
ANL A,#0FH
ADD A,B
MOV R0,A
MOV A,R6
MOV B,R2
ADD A,B
MOV R6,A
MOV A,R3
MOV B,R4
ADDC A,B
MOV R3,A
```

```
MOV A,R0
MOV B,R5
ADDC A,B
MOV R0,A
MOV DPTR,#4500H
MOV A,R6
MOVX @DPTR,A
INC DPTR
MOV A,R3
MOVX @DPTR,A
INC DPTR
MOV A,R0
MOVX @DPTR,A
L:SJMP L
```

Cube of a number

Input		Output	
Memory Location	**Data**	**Memory Location**	**Data**
4200	**89**	**4500**	**56**
		4501	**3C**
		4502	**27**

RESULT:

Thus the assembly language program to find 2's complement, Square and cube of a number was executed and verified successfully using 8051 microcontroller

VIVA QUESTIONS AND ANSWERS

1. Explain ANL Instruction.

The ANL instruction performs a bitwise logical AND operation between the specified byte or bit operands and stores the result in the destination operand.

2. **Explain Swap Instruction.**

Swap instruction swaps the contents of two registers

3. What is meant by CPL instruction?

CPL performs complement operation.It converts 0's to 1's and vice versa.

4. What is the difference between SJMP and LJMP?

SJMP, LJMP, AJMP. Short jump, relative address is 8 bit it support 127 location forward, Long jump range is 64 kb, Absolute jump to anywhere ...

5. What is the use of MOVX instruction?

The MOVX instruction transfers data between the accumulator and external data memory. External memory may be addressed via 16-bits in the DPTR register or via 8-bits in the R0 or R1 registers. When using 8-bit addressing, Port 2 must contain the high-order byte of the address

6. **What is meant by watch dog timer?**

A **watchdog timer** (WDT) is a hardware **timer** that automatically generates a system reset if the main program neglects to periodically service it. It is often used to automatically reset an embedded device that hangs because of a software or hardware fault.

17. UNPACKED BCD TO ASCII

AIM:

To convert BCD number into ASCII by using 8051 micro controller

RESOURCES REQUIERED:

- 8051 microcontroller kit
- Keyboard
- Power supply

Key	ASCII (hex)	Binary	BCD (unpacked)
0	30	011 0000	0000 0000
1	31	011 0001	0000 0001
2	32	011 0010	0000 0010
3	33	011 0011	0000 0011
4	34	011 0100	0000 0100
5	35	011 0101	0000 0101
6	36	011 0110	0000 0110
7	37	011 0111	0000 0111
8	38	011 1000	0000 1000
9	39	011 1001	0000 1001

For example

Packed BCD	*Unpacked BCD*	*ASCII*
29H	02H & 09H	32H & 39H
0010 1001	0000 0010 &	0011 0010 &
	0000 1001	0011 1001

Input		Output	
Memory Location	**Data**	**Memory Location**	**Data**
4200	**89**	**4201** **4202**	**39** **38**

Unpacked bcd to ascii conversion program for 8051

```
$MOD51
ORG 4100H
MOV DPTR,#4200H
MOVX A,@DPTR
MOV B,A
ANL A,#0FH
ADD A,#30H
MOV R0,A
MOV A,B
SWAP A
ANL A,#0FH
ADD A,#30H
MOV R1,A
INC DPTR
MOV A,R0
MOVX @DPTR,A
INC DPTR
MOV A,R1
MOVX @DPTR,A
L:SJMP L
```

RESULT:

Thus the assembly language program to convert unpacked BCD to ASCII was executed and verified successfully using 8051 microcontroller

VIVA QUESTIONS AND ANSWERES

1. What is meant by 'packed BCD' number?

Packed BCD is the first and second **number** is represented as the first 4 bits and last 4 bits in a byte. The **number** 75 in **packed BCD** would be 01110101. Unpacked **BCD** is each **number** is represented by its own byte. The **number** 75 in unpacked **BCD** would be 00000111 and 00000101

2. Differentiate between ANL and ORL instruction.

The different ways of pointing out an operand's location (source and destination) are.. For the ANL (AND) and ORL (oR) bit oriented operations, the source bit may use it.

3. What does SWAP A instruction do?

The SWAP instruction exchanges the low-order and high-order nibbles within the accumulator. No flags are affected by this instruction. See Also: XCH, XCHD ...

4. **Differentiate between Packed and Unpacked BCD numbers**.

Packed BCD is **the** first and second **number** are represented as **the** first **4** bits and last **4** bits in a byte. **The number** 75 in **packed BCD** would be 01110101. **Unpacked BCD** is each **number** is represented by its own byte. **The number** 75 in **unpacked BCD** would be 00000111 and 00000101

5. What is meant by ASCII code?

ASCII stands for American Standard Code for Information Interchange. Below is the **ASCII** character table, including descriptions of the first 32 characters.