MICRO PROCESSSOR AND MICRO CONTROLLER

LABORATORY MANUAL

R 19

III/ IV B.E (ECE) II–Semester

DEPT. OF ELECTRONICS AND COMMUNICATION ENGINEERING

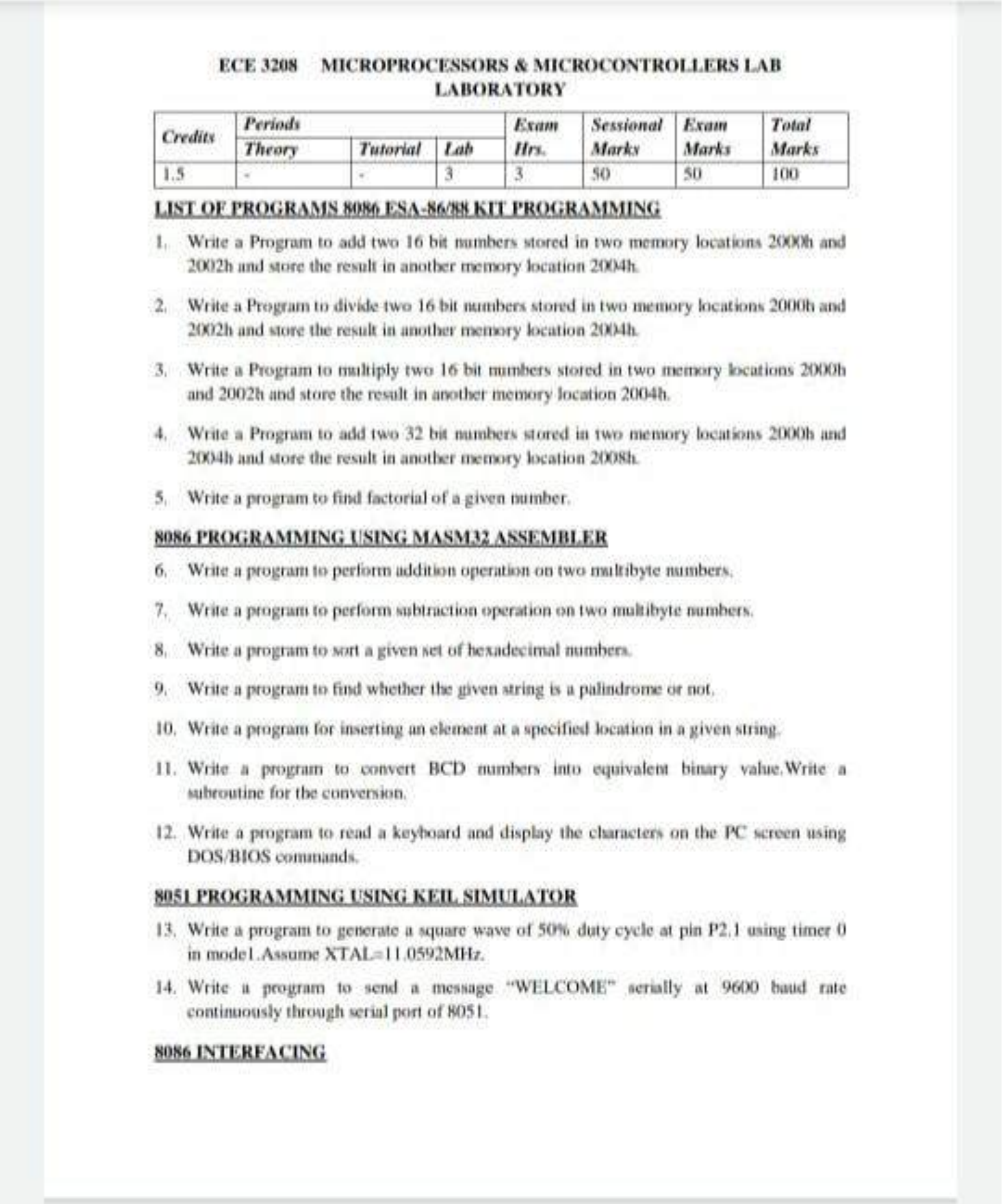
SIR C.R.REDDY COLLEGE OF ENGINEERING
ELURU-534007

MICRO PROCESSOR AND MICRO CONTROLLER LAB

LABORATORY MANUAL

LIST OF EXPERMENTS

S. No	Name Of The Experiment	Pg. No
	Part - A: 8086 PROGRAMS	
1	**A. Multibyte addition**	
	B. Multibyte subtraction	
2	**A. Multiplication of 16-bit**	
	B. Division of 16-bit	
3	**A. Sorting in Ascending order**	
	B. Sorting in descending order	
4	**Array of BCD addition**	
5	**Factorial of numbers**	
	PART –B : 8086 INTERFACE	
1	**Sawtooth wave**	
2	**Square wave**	
3	**Triangular wave generation**	
4	**Seven segment display**	
5	**Stepper motor**	
	Part - C: 8051 PROGRAMS	
1	**Even sum in array of data**	
2	**Counting no of 1’s and 0’s**	
3	**Sorting in 8051**	
4	**Average of array numbers**	
	Part - D: 8051 INTERFACE	
1	**Sawtooth wave**	
2	**Square wave**	
3	**Seven segment display**	
4	**Stepper motor**	
	Add On Experiments	
1		
2		
3		



ECE 3208 MICROPROCESSORS & MICROCONTROLLERS LAB LABORATORY

Credits	*Periods*			*Exam*	*Sessional*	*Exam*	*Total*
	Theory	*Tutorial*	*Lab*	*Hrs.*	*Marks*	*Marks*	*Marks*
1.5	-	-	3	3	50	50	100

LIST OF PROGRAMS 8086 ESA-86/88 KIT PROGRAMMING

1. Write a Program to add two 16 bit numbers stored in two memory locations 2000h and 2002h and store the result in another memory location 2004h.
2. Write a Program to divide two 16 bit numbers stored in two memory locations 2000h and 2002h and store the result in another memory location 2004h.
3. Write a Program to multiply two 16 bit numbers stored in two memory locations 2000h and 2002h and store the result in another memory location 2004h.
4. Write a Program to add two 32 bit numbers stored in two memory locations 2000h and 2004h and store the result in another memory location 2008h.
5. Write a program to find factorial of a given number.

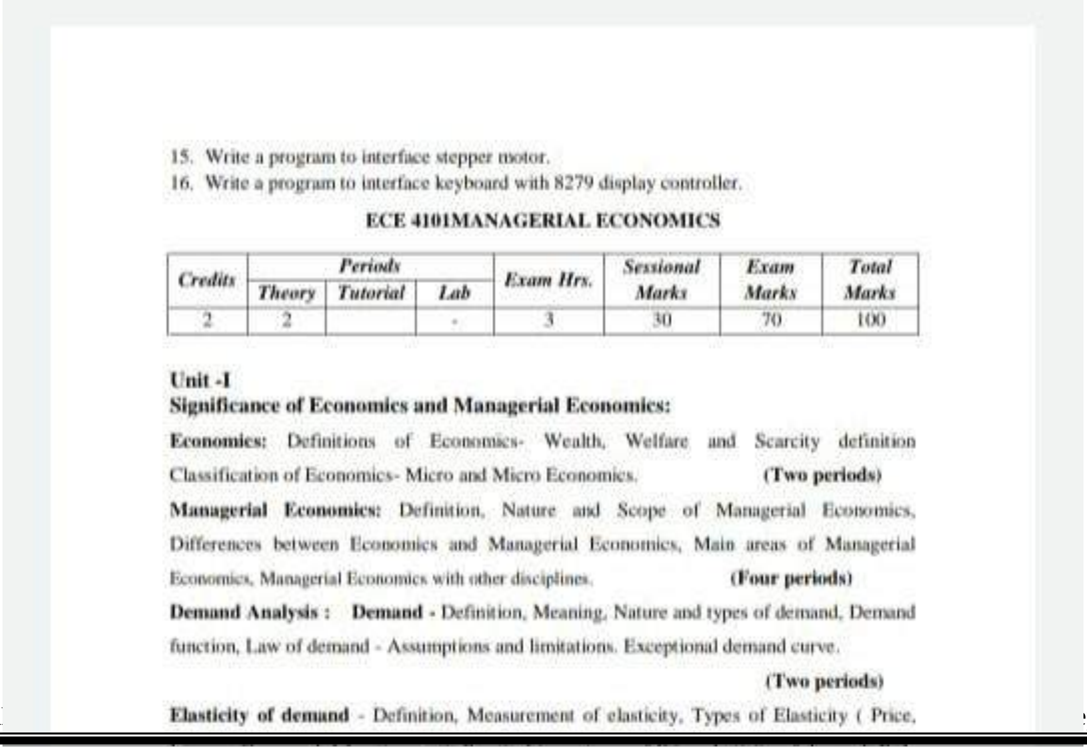
8086 PROGRAMMING USING MASM32 ASSEMBLER

6. Write a program to perform addition operation on two multibyte numbers.
7. Write a program to perform subtraction operation on two multibyte numbers.
8. Write a program to sort a given set of hexadecimal numbers.
9. Write a program to find whether the given string is a palindrome or not.
10. Write a program for inserting an element at a specified location in a given string.
11. Write a program to convert BCD numbers into equivalent binary value.Write a subroutine for the conversion.
12. Write a program to read a keyboard and display the characters on the PC screen using DOS/BIOS commands.

8051 PROGRAMMING USING KEIL SIMULATOR

13. Write a program to generate a square wave of 50% duty cycle at pin P2.1 using timer 0 in mode1.Assume XTAL=11.0592MHz.
14. Write a program to send a message "WELCOME" serially at 9600 baud rate continuously through serial port of 8051.

8086 INTERFACING



15. Write a program to interface stepper motor.
16. Write a program to interface keyboard with 8279 display controller.

ECE 4101MANAGERIAL ECONOMICS

Credits	*Periods*			*Exam Hrs.*	*Sessional Marks*	*Exam Marks*	*Total Marks*
	Theory	*Tutorial*	*Lab*				
2	2		-	3	30	70	100

Unit -I

Significance of Economics and Managerial Economics:

Economics: Definitions of Economics- Wealth, Welfare and Scarcity definition Classification of Economics- Micro and Micro Economics. **(Two periods)**

Managerial Economics: Definition, Nature and Scope of Managerial Economics, Differences between Economics and Managerial Economics, Main areas of Managerial Economics, Managerial Economics with other disciplines. **(Four periods)**

Demand Analysis : Demand - Definition, Meaning, Nature and types of demand, Demand function, Law of demand - Assumptions and limitations. Exceptional demand curve.

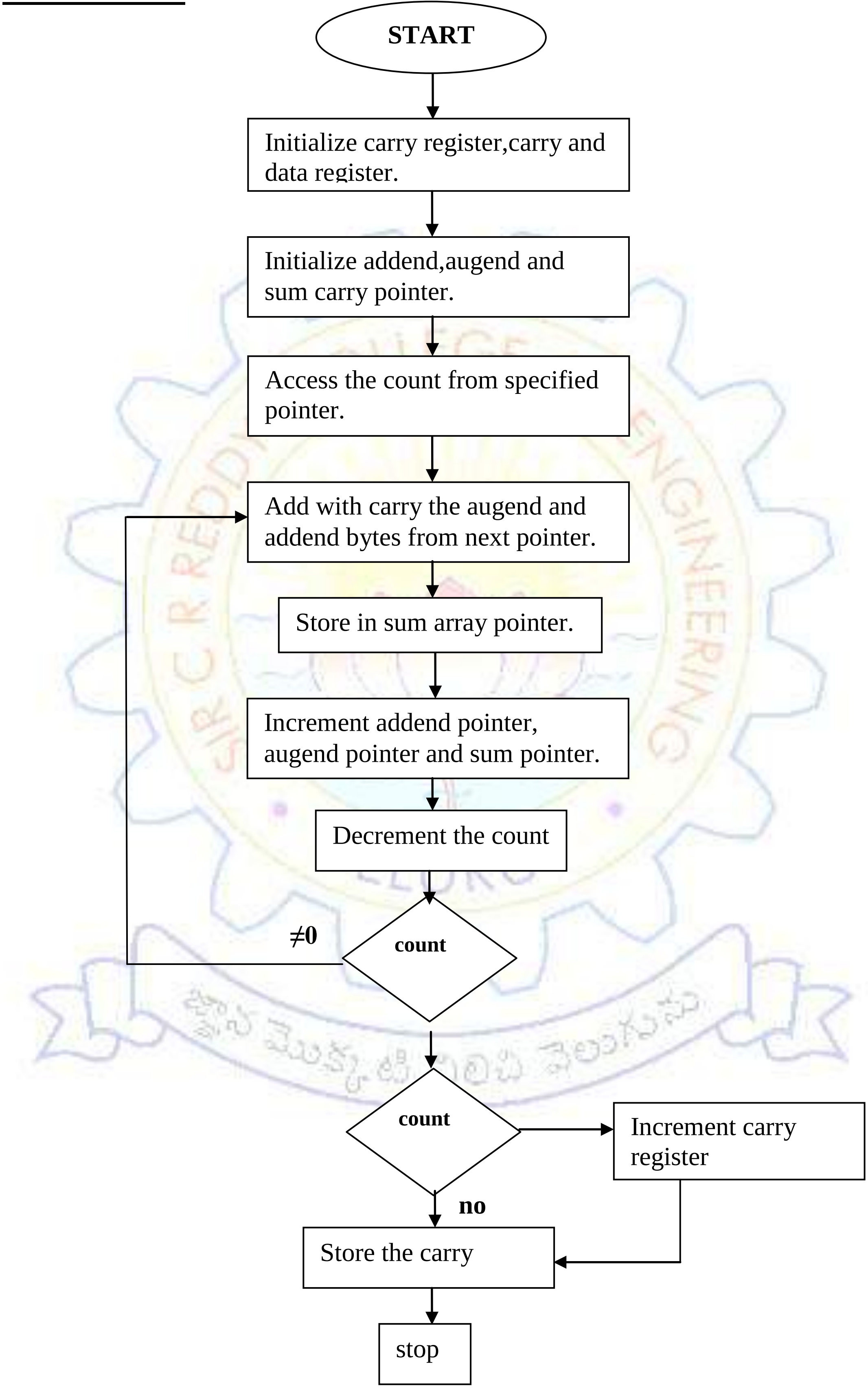
(Two periods)

Elasticity of demand - Definition, Measurement of elasticity, Types of Elasticity (Price,

8086 PROGRAMS

1(a).MULTIBYTE ADDITION

FLOWCHART:


START
Initialize carry register,carry and data register.
Initialize addend,augend and sum carry pointer.
Access the count from specified pointer.
Add with carry the augend and addend bytes from next pointer.
Store in sum array pointer.
Increment addend pointer, augend pointer and sum pointer.
Decrement the count
count
≠0
count
Increment carry register
no
Store the carry
stop

1(a).MULTIBYTE ADDITION

DATE:

EXP NO:

AIM:

Write An Assembly Language Programme For Perform The Multibyte Number Addition.

APPARATUS:

MASM 32 Assembler, ESA-86/88 Kit.

ALGORITHM:

Step1: Set SI Register As Pointer For Data.

Step2: Clear The Carry Register (Cl)

Step3: Initialize The Augend And Sum Array Pointer.

Step4: Access The Count Value From The Pointer.

Step5: Access The Augend And Addend Data From Next Pointer.

Step6: Perform The Byte Addition And Store In Sum Pointer.

Step7: Increment The Augend Pointer, Adder Pointer And Sum Pointer.

Step8: Decrement The Count Value.

Step9: Check The Count If It Zero Go To Next Step If Not Go To Step5.

Step10: Check The Carry If No Go To Step12 Else Go To Next Step.

Step11: Increment Carry Register Values.

Step12: Store The Carry.

ASSEMBLY LANGUAGE PROGRAM:

Address in hexa	Opcode in hexa	mnemonic	operand	comments
1000	B86A07	MOV	AX,076A	Initialization
1003	8ED8	MOV	DS,AX	Data Segment
1005	F8	CLC		Clear The Carry Flag
1006	BE0020	MOV	SI, 2000	Initialize Augend Carry Pointer
1009	BB0021	MOV	BX,2100	Initialize Addend Array Pointer
100C	BF0022	MOV	DI, 2200	Initialize Sum Array Pointer
100F	B100	MOV	CL,100	Clear The Carry Register
1011	8A2C	MOV	CH,[SI]	Load The Count
1013	46	INC	SI	Increment Augend Pointer
1014	8A04	MOV	AL,[SI]	Load The Augend Byte From Pointer
1016	1207	ADC	AL,[BX]	Add The Addend Byte From Pointer With Carry
1018	8805	MOV	[SI],AL	Store The Sum Byte Into Pointer
101A	47	INC	DI	Increment For Next Sum Pointer
101B	43	INC	BX	Increment For Next Addressed Pointer
101C	FECD	DEC	CH	Decrement Count
101E	75F3	JNZ	1013	If It Not Zero Repeat The Addition If Zero Check For Carry
1020	7302	INC	1024	Check For Carry If No Store The Carry.
1022	FEC1	INC	CL	Increment The Carry Register.
1024	880D	MOV	[DI],CL	Store The Multibyte Sum
1026	CC	INT	03	Stop The Programme

OUTPUT:

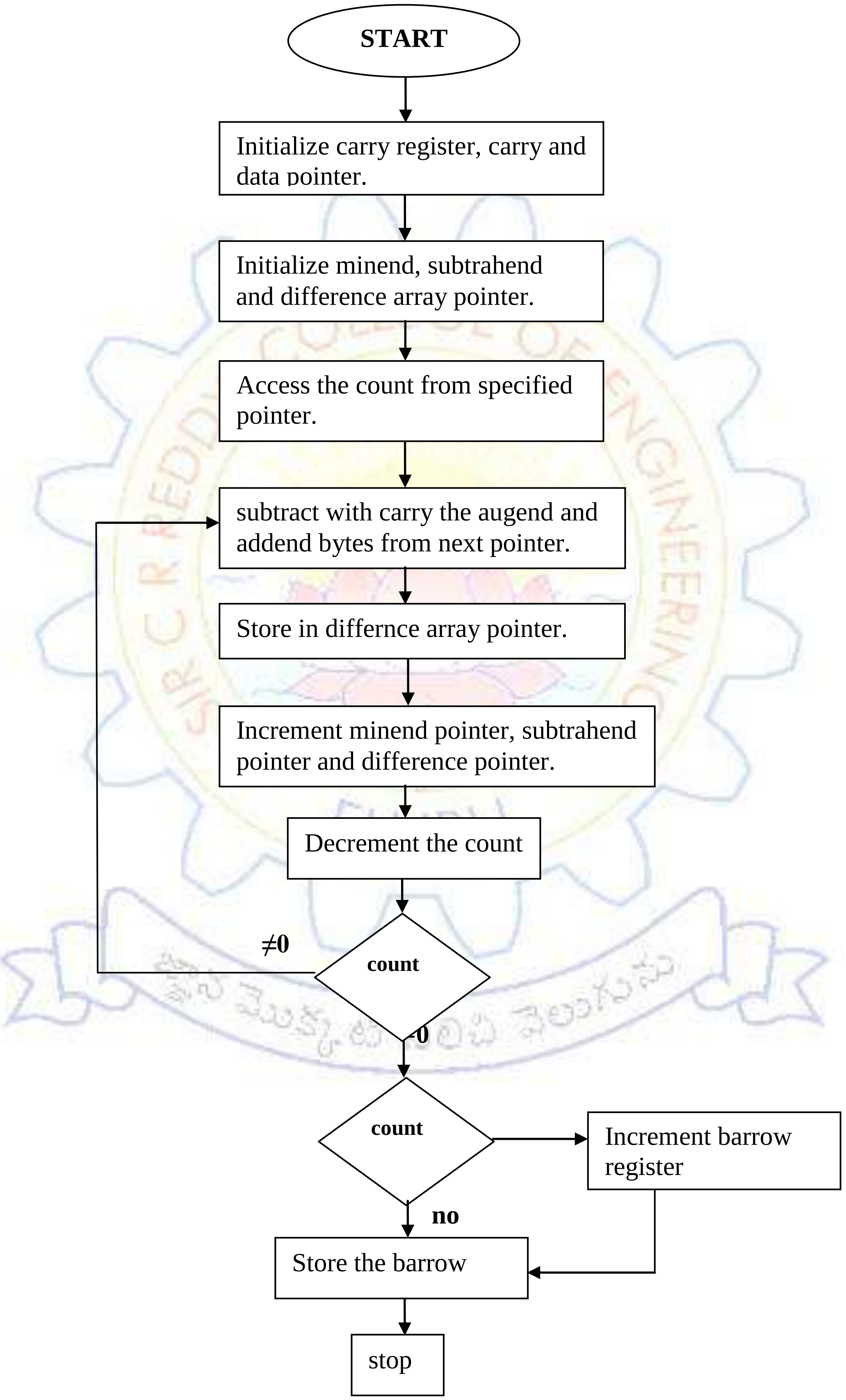
For N= 4					
2000	2001H 2100 2200	2002H 2101 2201	2003H 2102 2202	2004H 2103 2203	augend addend 2204 sum/carry

For N= 6							
2000	2001 2100 2200	2002 2101 2201	2003 2102 2202	2004 2103 2203	2005 2104 2204	2006 2105 2205	2206

RESULT:

1(b).MULTIBYTE SUBTRACTION

FLOWCHART:


START
Initialize carry register, carry and data pointer.
Initialize minend, subtrahend and difference array pointer.
Access the count from specified pointer.
subtract with carry the augend and addend bytes from next pointer.
Store in differnce array pointer.
Increment minend pointer, subtrahend pointer and difference pointer.
Decrement the count
≠0
count
=0
count
Increment barrow register
no
Store the barrow
stop

1(b).MULTIBYTE SUBTRACTION

DATE:

EXP.NO:

AIM:

Write An Assembly Language To Perform To An Multibyte Numbers Subtraction.

APPARATUS:

MASM 32 Assembler, ESA-86/88 Kit.

ALGORITHM:

Step1: Set SI Register As Pointer For Data.

Step2: Clear The Barrow Register(Cl).

Step3: Initialize The ,Minerend, Subtrahend And Difference Array Pointer.

Step4: Access The Count Value From The Pointer.

Step5: Access The Minerend And Subtrahend Data From Next Pointer.

Step6: Perform The Byte Subtraction And Store The Difference Pointer.

Step7: Increment The Subtrahend Pointer, Minend Pointer Nd Difference Pointer.

Step8: Decrement The Count Value.

Step9: Check The Count If It Zero Go To Next Step, If Not Go To Step5.

Step10: Check The Carry If No Go To Step12 Else To Next Step.

Step11: Increment Barrow Register Values.

Step12: Store The Barrow.

Step13: Stop The Programme.

ASSEMBLY LANGUAGE PROGRAM:

OFFSET CODE: 076AH

address in hexa	Opcode in hexa	mnemonic	operand	comments
1000	B86A07	MOV	AX,076A	Initialization Of Data Segment
1003	8ED8	MOV	DS,AX	
1005	E8	CLC		Clear Carry Flag
1006	BE0020	MOV	SI,2000	Initialize Subtrahend Array Pointer
100C	BF0022	MOV	DI,2200	Initialize Difference Array Pointer
100F	B100	MOV	CL 100	Clear The Carry Register
1011	8A2C	MOV	CH,[SI]	Load The Count
1013	46	INC	SI	Increment For Minuend Pointer
1014	8A04	MOV	AL,[SI]	Load The Minuend Byte From Pointer
1016	1A07	SBB	AL,[BX]	Subtract Subtrahend From
1018	8805	MOV	SI,[AL]	Store Difference In Pointer
101A	47	INC	DI	Increment Difference Pointer
101B	43	INC	BX	Increment Subtrahend Array Pointer
101C	FECD	DEC	CH	Decrement Cunt
101E	75F3	JNZ	1013	If It Not Zero Repeat The Subtractionif Zero Check Barrow
1020	7302	INC	1024	Check For Barrow If No Store The Barrow
1022	FECI	INC	CL	Increment The Carry Register.
1024	880D	MOV	[DI],CL	Store Multibyte Difference
1026	CC	INT	03	Stop The Programme

OUTPUT:

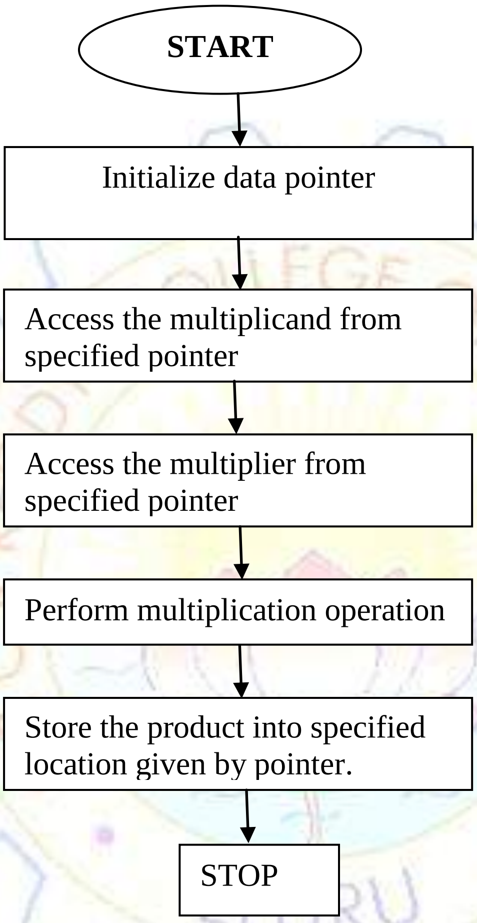
For N= 4					
2000	2001 2100 2200	2002 2101 2201	2003 2102 2202	2004 2103 2203	2204 sum/carry

For N= 6							
2000	2001 2100 2200	2002 2101 2201	2003 2102 2202	2004 2103 2203	2005 2104 2204	2006 2105 2205	2206

RESULT:

2(a).16 BIT MULTIPLICATION

FLOWCHART:


START
Initialize data pointer
Access the multiplicand from specified pointer
Access the multiplier from specified pointer
Perform multiplication operation
Store the product into specified location given by pointer.
STOP

2(a).16-BIT MULTIPLICATION

Date:

Exp No:

Aim:

Write an assembly language programme to multiply the two 8-Bit Numbers stored in 2000H & 2002H memory locations and store the result in 2004H and 2006H memory locations

Apparatus:

MASM 32 Assembler, ESA-86/88 Kit

Algorithm:

STEP1: Set SI register as pointer for data

STEP2: Get multiplier 16-Bit data into AX-Register.

STEP3: Get the multiplicand 16-Bit data and into BX register.

STEP4: Perform the 16- Bit multiplication operation.

STEP5: Store the product in memory.

STEP6: Stop.

ASSEMBLY LANGUAGE PROGRAM:

DATA SEGMENT OFFSET : 076AH

Address	Machine code	Mnemonic	Operand	Comments
1000	B86A07	MOV	AX,076A	Initialization of data segment
1003	8ED8	MOV	DS,AX	
1005	BE0020	MOV	SI,2000H	Set SI as a pointer for Data
1008	8BC4	MOV	AX,[SI]	Get the multiplicand 16 bit data into AX
100A	8B5002	MOV	BX,[SI+4]	Load the multiplicate 16 bit data into BX
100D	F7E3	MUL	BX	Perform 16 bit multipliction
100F	894404	MOV	[SI+6],AX	Store the product lower wordin memory by specified pointer
1012		MOV	[SI+08],DX	Store the product higher word in memory by specified pointer
1015		INT	03	Break the programme

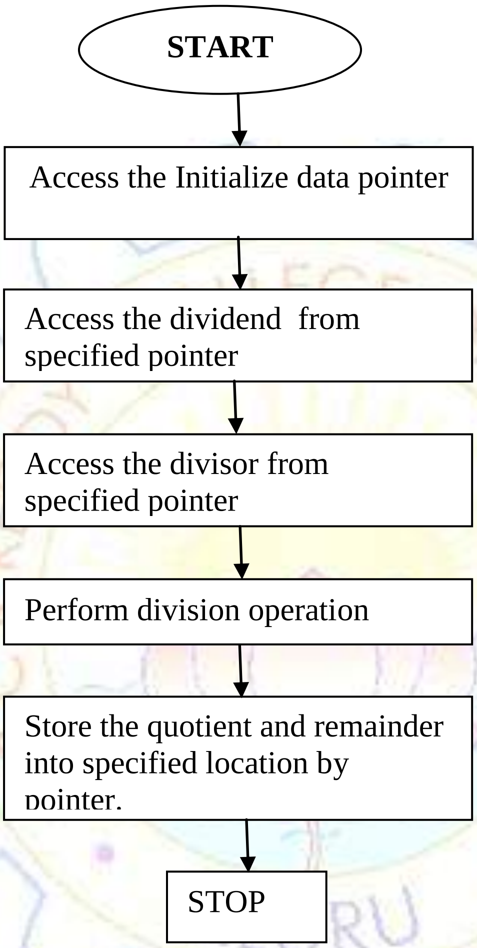
OUTPUT:

DATA	Multiplicand 16bit data		Multiplier 16 bit data		Higher 16bit		Lower 16-bit	
	2003H	2002H	2001H	2000H	2007H	2006H	2005H	2004H
Data1								
Data2								
Data3								
Data4								
Data5								

RESULT:

2(b).DIVISION OF 16-BIT NUMBERS

FLOWCHART:


START
Access the Initialize data pointer
Access the dividend from specified pointer
Access the divisor from specified pointer
Perform division operation
Store the quotient and remainder into specified location by pointer.
STOP

2(b).DIVISION OF 16-BIT NUMBERS

DATE:
EXP.NO:

AIM:

Write An Assembly Language Programme To Divide The Two 16-Bit Numbers Stored In 2000h And 2002h Memory Locatins And Store The Result In 2004h And 2006h Memory Loctions.

APPARATUS:

MASM 32 Assembler, ESA-86/88 Kit

ALGORITHM:

Step1: Set The SI Register As Pointer For Data.

Step2: Get Dividend Lower 16-Bit Data Into AX Register.

Step3: Get The Dividend Hogher 16-Bit Data Into DX Register.

Step4: Get The Divisor 16-Bit Data Into BX Register.

Step5: Perform The 16-Bit Division Operation.

Step6: Store The Quotient In Memory.

Step7: Store The Remainder In Memory.

Step8: Stop.

ASSEMBLY LANGUAGE PROGRAM:

DATA SEGMENT OFFSET: 076AH

Address in hexa	Opcode in hexa	mnemonic	operand	comments
1000	B86A07	MOV	AX,076A	Initialization Of Data Segment
1003	8ED8	MOV	DS,AX	
1005	BE0020	MOV	SI,2000H	Set Si As Pointer For Data
1008	8B04	MOV	AX,[SI]	Get The Dividend Lower 16 Bit Data Into Ax.
100A	8B5402	MOV	DX,[SI+2]	Load The Dividend Higher 16-Bit Data Into Dx
100D	8B5C04	MOV	BX,[SI+4]	Load The Divisor 16-Bit Data Into Bx.
1010	F7F3	DIV	BX	Perform 16-Bit Division
1012	894406	MOV	[SI+6],AX	Store The Quotient In Memory By Specified Pointer.
1015	895408	MOV	[SI+8],DX	Store The Remainder In Memory By Specified Pointer.
1018	CC	INT	03	Break The Programme

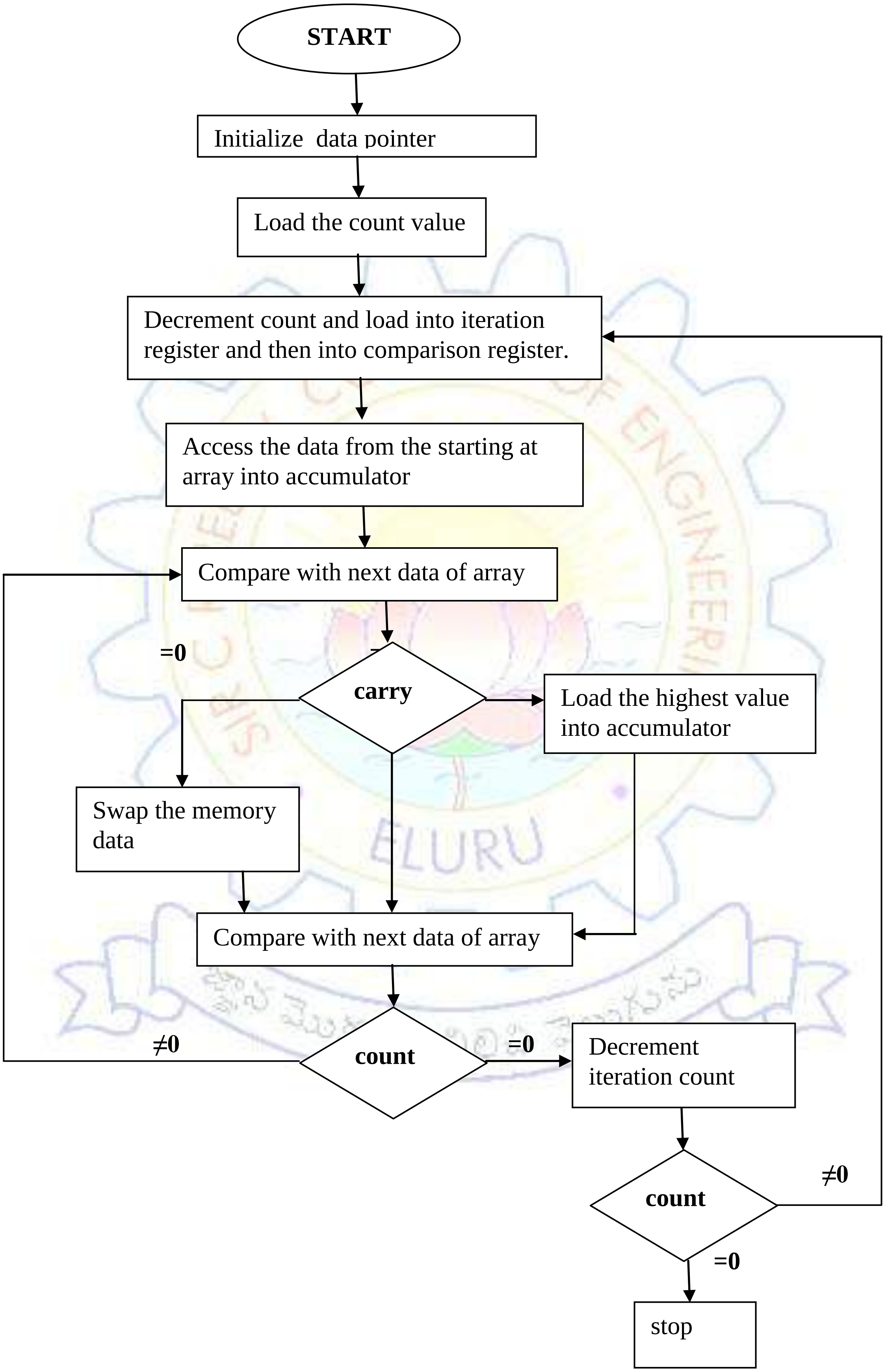
OUTPUT:

DATA	**Multiplicand 16bit data**		**Multiplier 16 bit data**		**Higher 16bit**		**Lower 16-bit**	
	2003H	**2002H**	**2001H**	**2000H**	**2007H**	**2006H**	**2005H**	**2004H**
Data1								
Data2								
Data3								
Data4								
Data5								

RESULT:

3(a).SORTING THE DATA

FLOWCHART:


START
Initialize data pointer
Load the count value
Decrement count and load into iteration register and then into comparison register.
Access the data from the starting at array into accumulator
Compare with next data of array
=0
carry
Load the highest value into accumulator
Swap the memory data
Compare with next data of array
≠0
count
=0
Decrement iteration count
count
≠0
=0
stop

3(a).SORTING THE DATA

DATE:

EXP.NO:

Aim:

Write an assembly language progmme to perform the sorting of an array.

Apparatus:

MASM 32 assembler

ESA -82/86 kit

Algorithm:

Step 1: Set SI register as pointer for data.

Step 2: Load the count value

Step 3: Decrement the count value

Step 4: Load it into iteration register then into comparison register

Step 5: Access the data from the storing of arrays into accumulator.

Step 6: Compare with next data of the array pointer.

Step 7: Check array if array exist store highest value into accumulator then go to step 9, if carry doesn't exist go to next step.

Step 8: Swap the memory contents.

Step 9: Decrement the comparison count if it zero go to next step. If it does not zero

go to step6.

Step 10: Decrement the iteration count.

Step 11: If it non zero go to step 4. If it zero go to next step.

Step 12: Stop the program.

ASSEMBLY LANGUAGE PROGRAM:

ascending order

DATASEGMENT OFFSET : 076AH

Address in hexa	Opcode in hexa	label	mnemonic	operand	comments
1000	B86A07		MOV	AX,076A	Initialization of data Segment
1003	8E02		MOV	DS,AX	
1005	BE0020		MOV	SI,2000H	Set SI as pointer for data
1008	8A3C		MOV	BH,[SI]	Load the count
100A	FECF		DEC	BH	Decrement for iterations and comparison
100C	8ADF	LOOP3	MOV	BL,BH	Load the iteration no. to comparisons register
100E	BE0120		MOV	SI,2001H	Initialize data pointer for sorting
1011	8A04	LOOP2	MOV	AL,[SI]	Load the data into acc
1013	46		INC	SI	Increment pointer for next data
1014	3A04		CMP	AL,[SI]	Compare acc and memory data
1016	7205		INC	1010H	If acc is less go for decrement then compare value, if else go for swap
1018	8604		XCHG	AL,[SI]	Swapping
101A	8644FE		XCHG	AL,[SI-1]	
101D	FECB	LOOP1	DEC	BL	Decrement comparison number
101F	7SF0		JNZ	1011,H	If it non zero load theproceed for next comparison

1021	FECF		DEC	BH	Decrement the count number
1023	75E7		JNZ	100C H	It is non-zero go for next iterations
1025	CC		INT	3	Stop the program

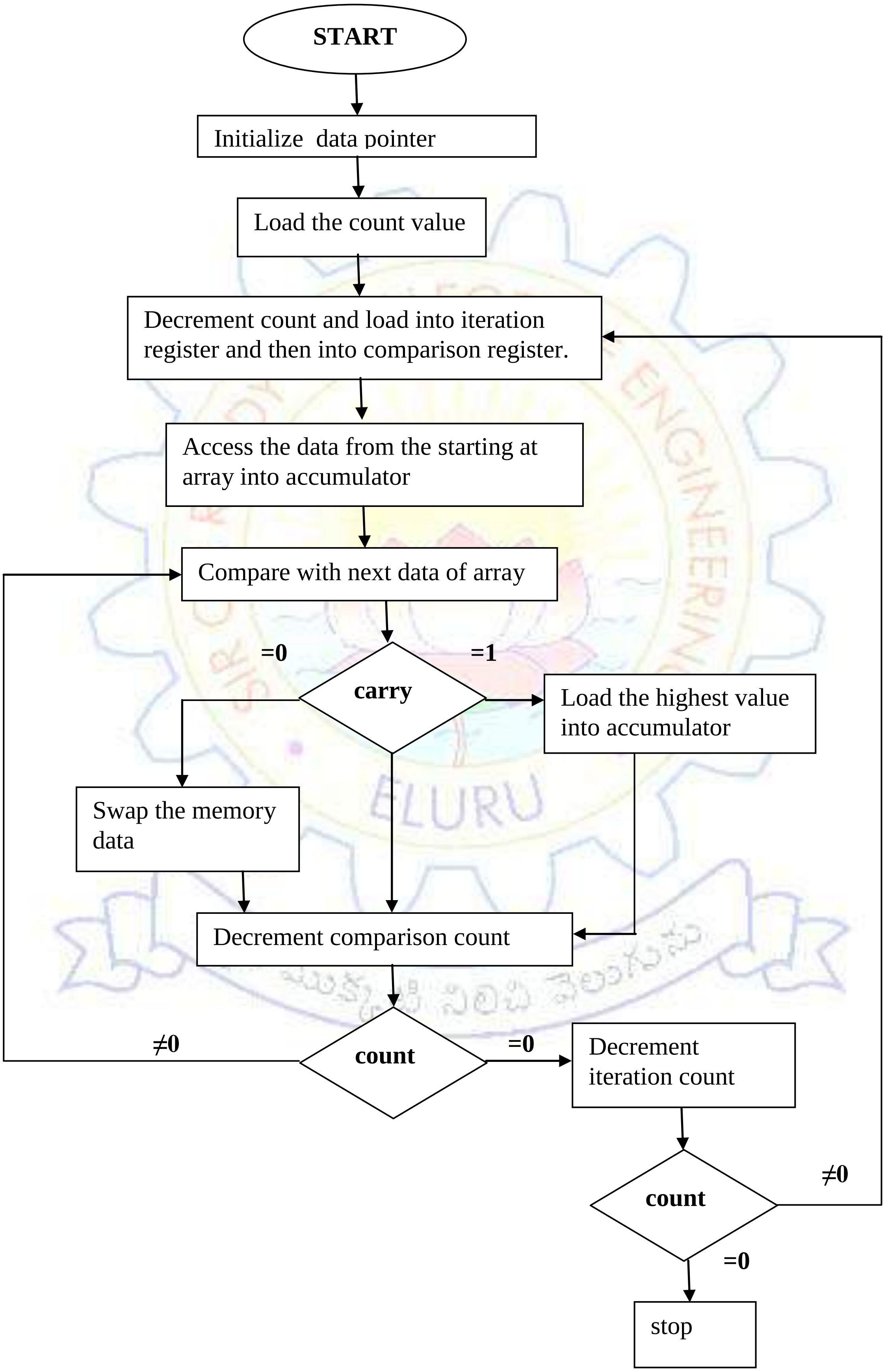
OUTPUT:

Data no.	**Count 2000**	**2001**	**2002**	**2003**	**2004**	**2005**	**2006**

Result:

3(b).SORTING DATA IN DESCENDING ORDER

FLOWCHART:


START
Initialize data pointer
Load the count value
Decrement count and load into iteration register and then into comparison register.
Access the data from the starting at array into accumulator
Compare with next data of array
=0
=1
carry
Load the highest value into accumulator
Swap the memory data
Decrement comparison count
≠0
count
=0
Decrement iteration count
count
≠0
=0
stop

3(b).SORTING DATA IN DESCENDING ORDER

DATE:

EXP.NO:

AIM:

Write an assembly language programme to perform the sorting of array in descending order.

APPARATUS:

MASM 32 Assembler, ESA-86/88 Kit

ALGORITHM:

Step1: set SI register as pointer for data

Step2: load the count value.

Step3: decrement the count value.

Step4: load it into iteration register then into comparison register.

Step5: access the data from the starting of array into accumulator.

Step6: compare with the next data of the array pointer

Step7: check the carry if carry exists store highest value into accumulator then go to step9,if carry doesn’t exists next step.

Step8: swap the memory contents

Step 9: Decrement the comparison count if it zero go to next step. If it not zero go to step6.

Step 10: Decrement the iteration count.

Step 11: If it non zero go to step 4. If it zero go to next step.

Step 12: Stop the program.

ASSEMBLY LANGUAGE PROGRAM:

DATA SEGMENT OFFSET : 076AH

address **n hexa**	**Opcode** **in hexa**	**label**	**mnemonic**	**operand**	**comments**
1000	B86A07		MOV	AX,076A	Initialization of data Segment
1003	8E08		MOV	DS,AX	
1005	BE0020		MOV	SI,2000H	Set SI as pointer for data
1008	8A3C		MOV	BH,[SI]	Load the count
100A	FECF		DEC	BH	Decrement for iterations and comparison
100C	8ADF	LOOP3	MOV	BL,BH	Load the iteration no. to comparisons register
100E	BE0120		MOV	SI,2001H	Initialize data pointer for sorting
1011	8A04	LOOP2	MOV	AL,[SI]	Load the data into acc
1013	46		INC	SI	Increment pointer for next data
1014	3A04		CMP	AL,[SI]	Compare acc and memory data
1016	7205		JC	1010H	If acc is less go for decrement then compare value, if else go for swap
1018	8604		XCHG	AL,[SI]	Swapping
101A	8644FE		XCHG	AL,[SI-1]	
101D	FECB	LOOP1	DEC	BL	Decrement comparison number
101F	7SF0		JNZ	1011,H	If it non zero load thehighest no. into acc. And proceed for next comparison
1021	FECF		DEC	BH	Decrement the count number

1023	75E7		JNZ	100C H	It is non-zero go for next iterations
1025	CC		INT	3	Stop the program

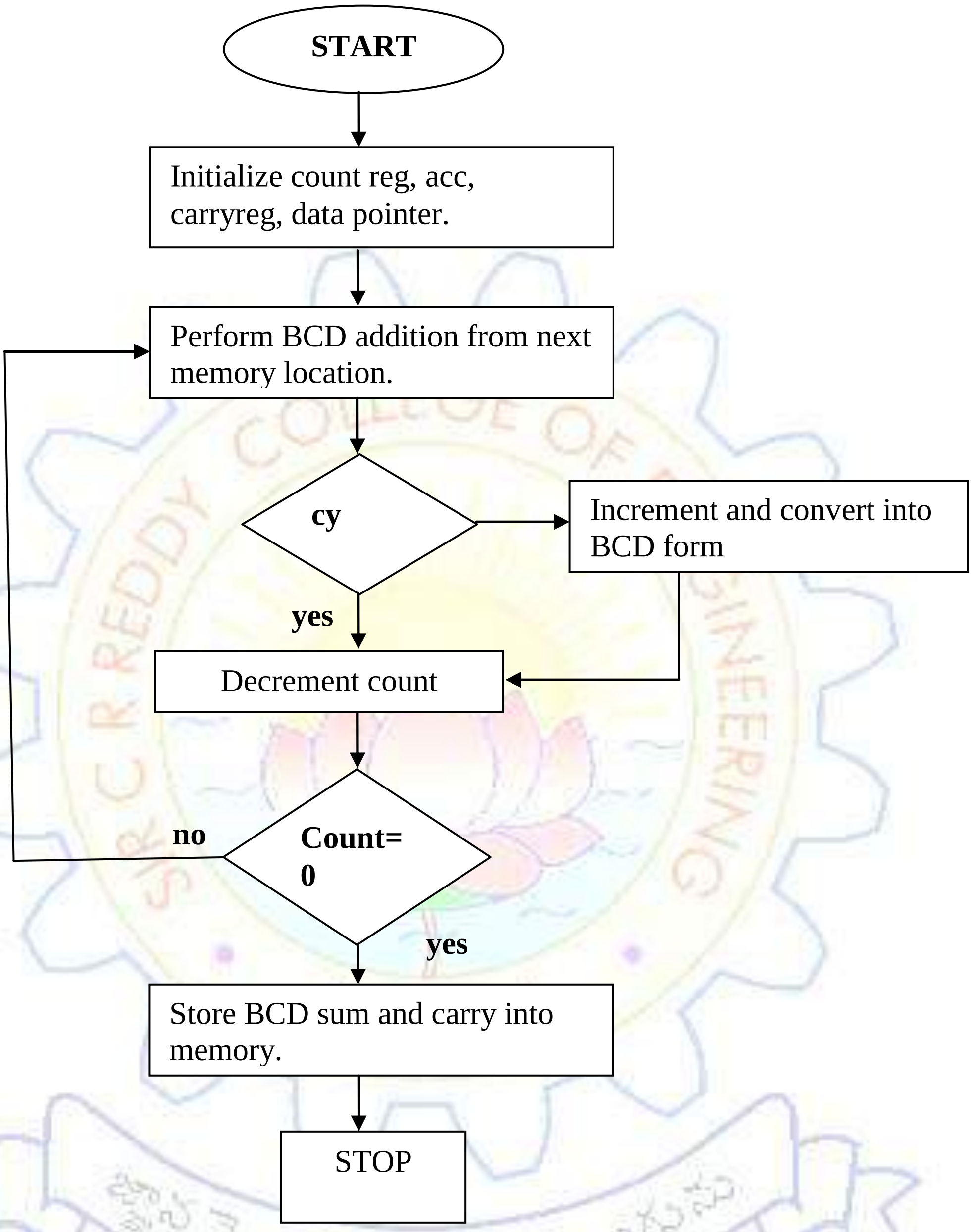
OUTPUT:

Data	**Count**						
no.	**2000**	**2001**	**2002**	**2003**	**2004**	**2005**	**2006**

RESULT:

4.ARRAY OF BCD ADDITION

FLOWCHART:


START
Initialize count reg, acc, carryreg, data pointer.
Perform BCD addition from next memory location.
cy
Increment and convert into BCD form
yes
Decrement count
no
Count= 0
yes
Store BCD sum and carry into memory.
STOP

4.ARRAY OF BCD ADDITION

DATE:

EXP.NO:

AIM:

Write An Assembly Language Programme For an array of BCD addition and the result will be stored in some memory locations

APPARATUS:

MASM 32 Assembler, ESA-86/88 Kit.

ALGORITHM:

Step1: SI with 2000H

Step2: initialize Acc. With 00.

Step3: Initialize carry register with 00.

Step4: move Count Value From data Pointer into count registers.

Step5: increment SI value.

Step6: add memory content with accumulator.

Step7: convert sum value from hexa decimal to decimal.

Step8:check the carry. If not go to step13, if yes go to next step.

Step9:exchange carry into acc.

Step10:increment accumulator.

Step11:count acc.from hexadecimal to decimal.

Step12: exchange carry register and acc.

Step13: decrement count register

Step14: check count register if zero go to step15 if it not zero go to step5.

Step15: store the BCD into memory.

Step16: store the carry into carry.

Step17: stop the program.

ASSEMBLY LANGUAGE PROGRAM:

DATA SEGMENT OFFSET : 076AH

Address in hexa	Opcode in hexa	mnemonic	operand	comments
1000	B86A07	MOV	AX,076A	Initialization of data
1003	8ED8	MOV	DS,AX	Segment
1005	B000	MOV	AL,00	Clear the acc. Register
1007	B100	MOV	CL,00	Clear the carry register
1009	BE0020	MOV	SI,2000	Initialize the acc. Register
100C	8E2C	MOV	CH,[SI]	Load the count
100E	46	INC	SI	Increment the data pointer
100F	0204	ADD	AL,[SI]	Add the byte from data pointer
1011	27	DAA		Apply the decimal adjust after addition
1012	7307	JNC	101B	Check for carry if no store the carry
1014	86C1	XCHG	AL,CL	Exchange acc. And carry
1016	0401	ADD	AL,01	Add the 01 from pointer
1018	27	DAA		Apply the decimal adjust after addition
1019	86C1	XCHG	AL,CL	Exchange the carry and acc.
101B	FECD	DEC	CH	Decrement count
101D	75EF	JNZ	100E	If it not zero repeat addition if zero check for carry
101F	884401	MOV	[SI+01],AL	Store the sum in memory by specified pointer
1025	CC	INT	O3	Stop programme.

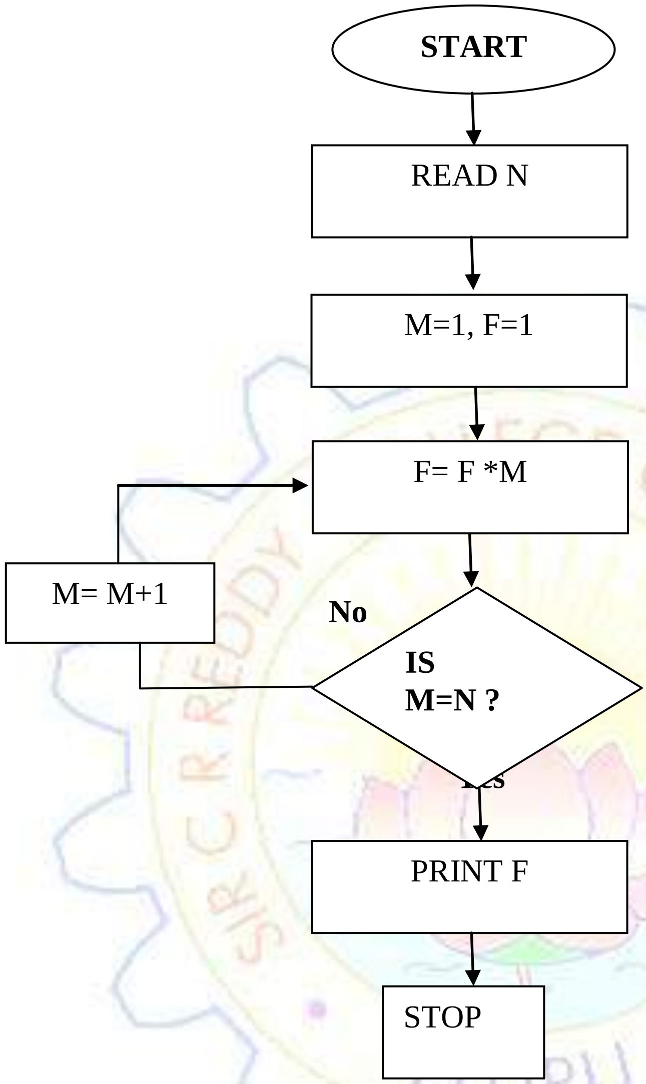
OUTPUT:

S.NO	1	2	3	4	5

RESULT:

5. FACTORIAL OF NUMBER

FLOWCHART:



5.FACTORIAL OF NUMBER

DATE:

EXP.NO:

AIM:

Write An Assembly Language Programmeto find the factorial of a given number stored in 2000H memory location and store the resulting in memory location.

APPARATUS:

MASM 32 Assembler, ESA-86/88 Kit.

ALGORITHM:

Step1: set SI register as pointer for data

Step2: get given data(N)

Step3: Initialize the multiplicand with 1

Step4: Initialize the multiplier with 1.

Step5: perform the multiplication.

Step6: compare with multiplier with given number, if true go to step 9, if not go to STEP7

Step7: increment multiplier valve with 1

Step8: go to step 5.

Step9:store the factorial in memory.

Step10:stop

ASSEMBLY LAGUAGE PROGRAM:

DATA SEGMENT OFFSET: 076AH

Address in hexa	Opcode in hexa	mnemonic	operand	comments
1000	B86A07	MOV	AX,076A	Initialization of data
1003	8ED8	MOV	DS,AX	Segment
1005	BE0020	MOV	SI,2000H	Set SI as pointer for data
1008	8B1C	MOV	BX,[SI]	Access the number from memory which factorial to be find
100A	83FB00	CMP	BX,+00	Compare with given no.
100D	740A	JZ	101D	If it is equal store the result
100F	B80100	MOV	AX,0001	Initialize multiplicand with 1
1012	8BC8	MOV	CX,AX	Initialize multiplier with 1
1014	F7E1	MUL	CX	Perform 8 bit multiplication
1016	3BCB	CMP	CX,BX	Compare with multiplier at a given number
1018	7406	JZ	1020	If it equal store result in memory
101A	41	INC	CX	If not equal increment multiplier
101B	EBF7	JMP	1014	Go to perform next multiplication
101D	B80100	MOV	AX,0001	Store the result
101E	CC	INT	03	Stop the programme

OUTPUT:

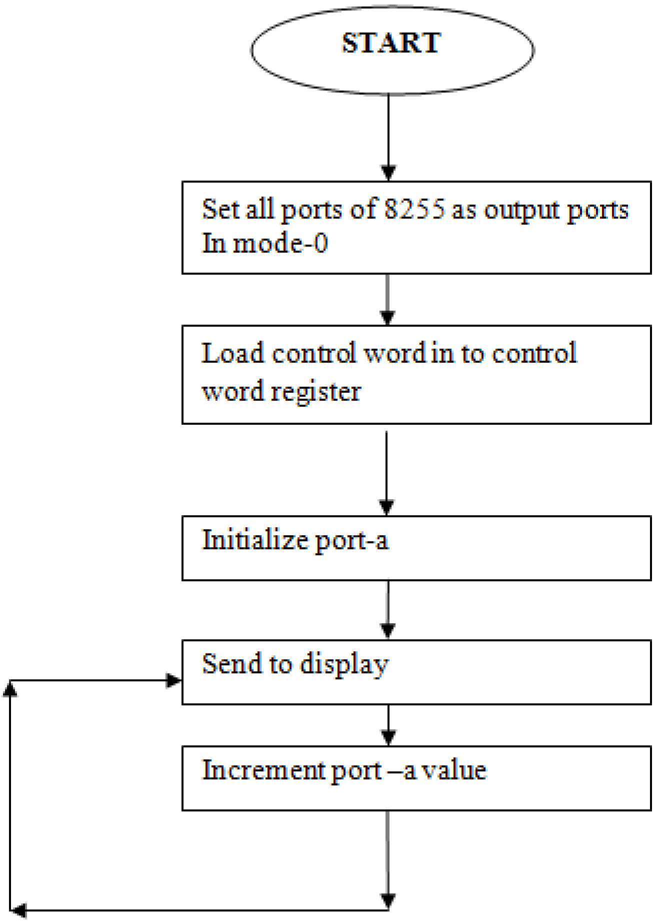
Data no.	Number	Factorial			
	Address 2000H	Higher word		Lower word	
		Higher byte address 2004H	lower byte address 2003H	Higher byte address 2002H	lower byte address 2001H

RESULT:

8086 INTERFACING

1.SAWTOOTH WAVE

FLOW CHART :


START
Set all ports of 8255 as output ports In mode-0
Load control word in to control word register
Initialize port-a
Send to display
Increment port –a value

1.SAWTOOTH WAVE

DATE:-
EXP.NO:-

AIM:- Write an Assembly Language Program to generate sawtooth wave using DAC through 8255 PPI

APPARATUS :-

MASM 32 ASSEMBLER , ESA-86/88 KIT, DAC card and CRO

THEORY:-

In this circuit the 8086 processor is interfaced with 8255 in mode-0 and set all the ports are set to output. The output of port-A is connected to DAC which converts the digital input to corresponding analog output. The is send to CRO to display. Initially the port-A is loaded with 00 and the corresponding analog output is send to CRO. And increment port-A value continuously until the maximum value. If the maximum value is 0FF no need to compare. Once it is reached to maximum value then it will reached to initial value. And repeated the same. If the maximum is not FF then for each and every increment we should compare with maximum value if is equal or less than we should send to port-A to display. After that again start from 00 and repeat. The wave amplitude and frequency are depends on maximum count value to send to Port-A.

ALGORITHM :-

step1: Set all ports as output of 8255 in mode-0

Step2: Load control word into controlword register.

Step3: Initialize port-a with 00 and output to port-a

Step4: send to display through DAC

Step5: Increment the port-a value and go to step 4

ASSEMBLY LANGUAGE PROGRAM

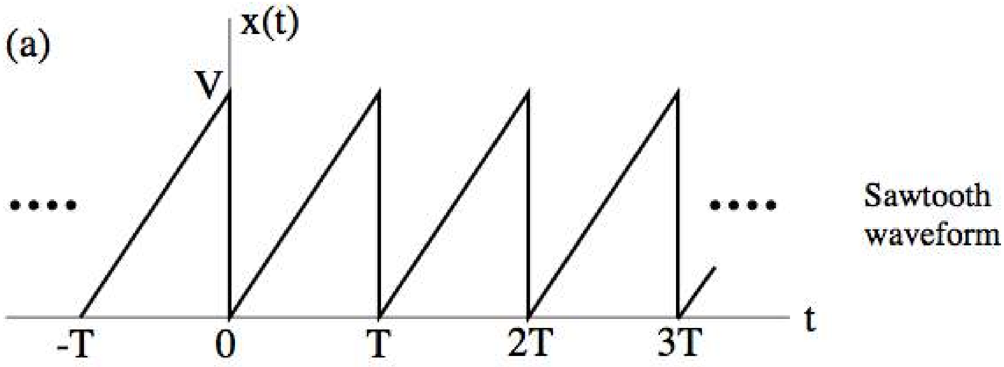
```
          MOV    DX,0FFE6H
          MOV     AL,80H
          OUT     DX,AL
          MOV   DX,0FFE0H
          MOV    AL,00H
      LOOP1:    OUT    DX,AL
          INC     A
          JMP     LOOP1
```

ASSEMBLY LANGUAGE PROGRAM FOR SAWTOOH WAVE

DATA SEGMENT OFFSET : 076AH

ADDRESS IN HEXA	OP CODE IN HEXA	MNEMONIC	OPERAND	COMMENT
2000	BA,E6,FF	MOV	DX,0FFE6	Set all ports of 8255 as output In mode -0
2003	B0,80	MOV	AL,80H	
2005	EE	OUT	DX,AL	Load control word in to contro Word register
2006	BA,E0,FF	MOV	DX,0FFE0	Out the initial word into port-A
2009	B0,00	MOV	AL,00	
200B	EE	OUT	DX,AL	Send to display through DAC
200C	FE,C0	INC	AL	Increment continuously
200E	EB,FB	JMP	200B	Display continuously

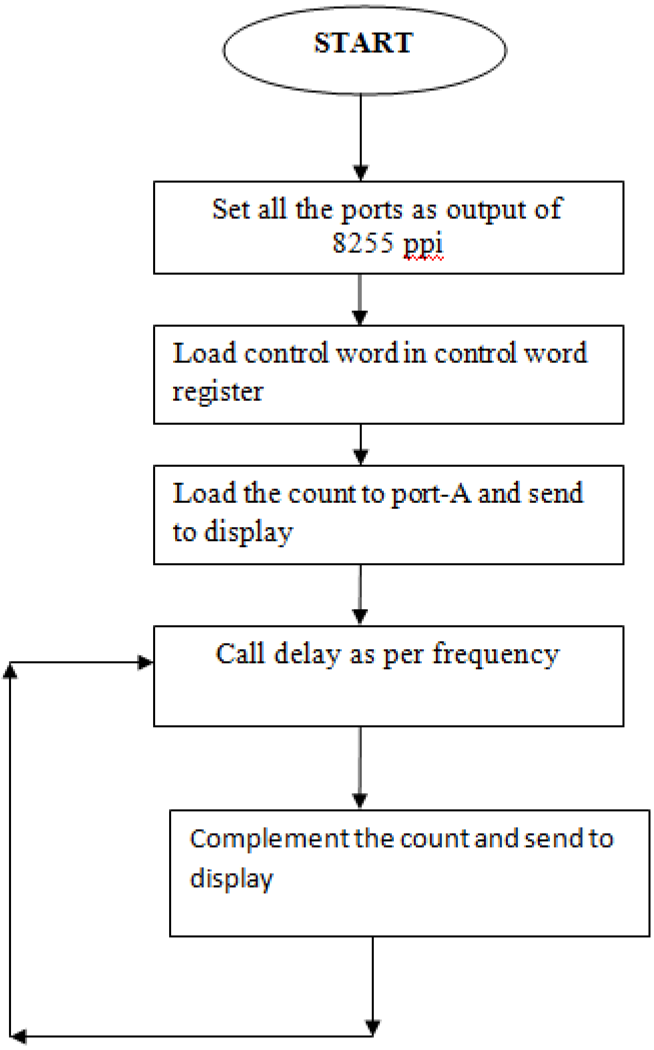
RESULT:-



Result:- Generating the sawtooth wave with different amplitudes and frequencies.

2.SQUARE WAVE GENERATION

FLOW CHART:


START
Set all the ports as output of 8255 ppi
Load control word in control word register
Load the count to port-A and send to display
Call delay as per frequency
Complement the count and send to display

2.SQUAREWAVE

DATE:-

EXP.NO:-

AIM:-

Write an Assembly Language Program to generate the square wave using 8255 ppi in mode-0

APPARATUS :-

ESA-86/88 KIT, CRO, DAC

THEORY:-

In this circuit the 8086 processor is interfaced with 8255 in mode-0 and set all the ports are set to output. The output of port-A is connected to DAC which converts the digital input to corresponding analog output. The is send to CRO to display. Initially the port-A is loaded with FF and the corresponding analog output is send to CRO. And call the delay as per frequency requirement for on time. For off time complement the count and then send to display. Repeat the above continuously. The square wave having duty cycle 50%. So ontime and off time are equal , for this we are calling same delay routine.

ALGORITHM :-

step1: Set all ports as output of 8255 in mode-0

Step2: Load control word into control word register.

Step3: Initialize port-a with count and output to port-A.

Step4: Call Delay.

Step5: Complement the count and output to port-A

Step6: Goto step 4

ASSEMBLY LANGUAGE PROGRAM:

```
          MOV   DX,0FFE6H
          MOV    AL,80H
          OUT    DX,AL
          MOV   DX,0FFE0H
          MOV    AL,FFH
    LOOP1:    OUT    DX,AL
          CALL  2050H(DELAY)
          NOT   AL
          JMP   LOOP1
```

ASSEMBLY LANGUAGE PROGRAM FOR SQUARE WAVE

DATA SEGMENT OFFSET : 076AH

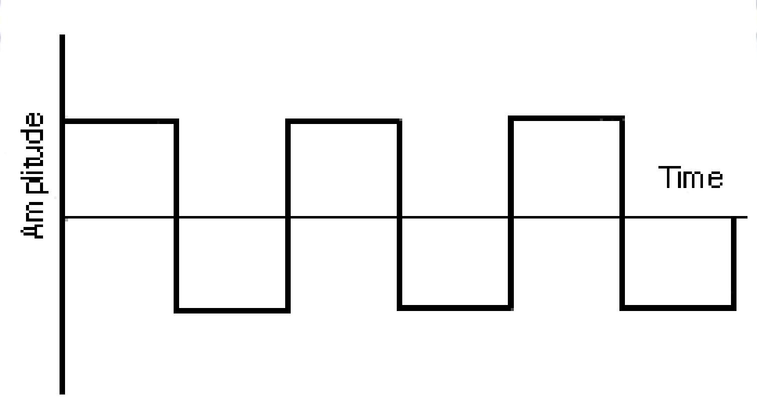
ADDRESS IN HEXA	OP CODE IN HEXA	MNEMONIC	OPERAND	COMMENT
2000	BA,E6,FF	MOV	DX,0FFE6	Set all ports of 8255 as output In mode -0
2003	B0,80	MOV	AL,80H	
2005	EE	OUT	DX,AL	Load control word in to contro Word register
2006	BA,E0,FF	MOV	DX,0FFE0	Out the initial word into port-A
2009	B0,FF	MOV	AL,0FF	
200B	EE	OUT	DX,AL	Send to display through DAC
200C	E8,06,00	CALL	2041 (DELAY)	Call delay program for on/off time
200F	F6,D0	NOT	AL	Complement for off time
2011	EB,FB	JMP	200B	Send to display for off time

DELAY PROGRAM

2041	B9,0F,00		MOV	CX,00FF	
2044	BA,FF,FF	LOOP2	MOV	DX,0FFFF	
2047	4A	LOOP1	DEC	DX	
2048	75,FD		JNZ	LOOP1 (2047)	
204A	49		DEC	CX	
204B	75,F7		JNZ	LOOP2 (2044)	
204D	C3		RET		

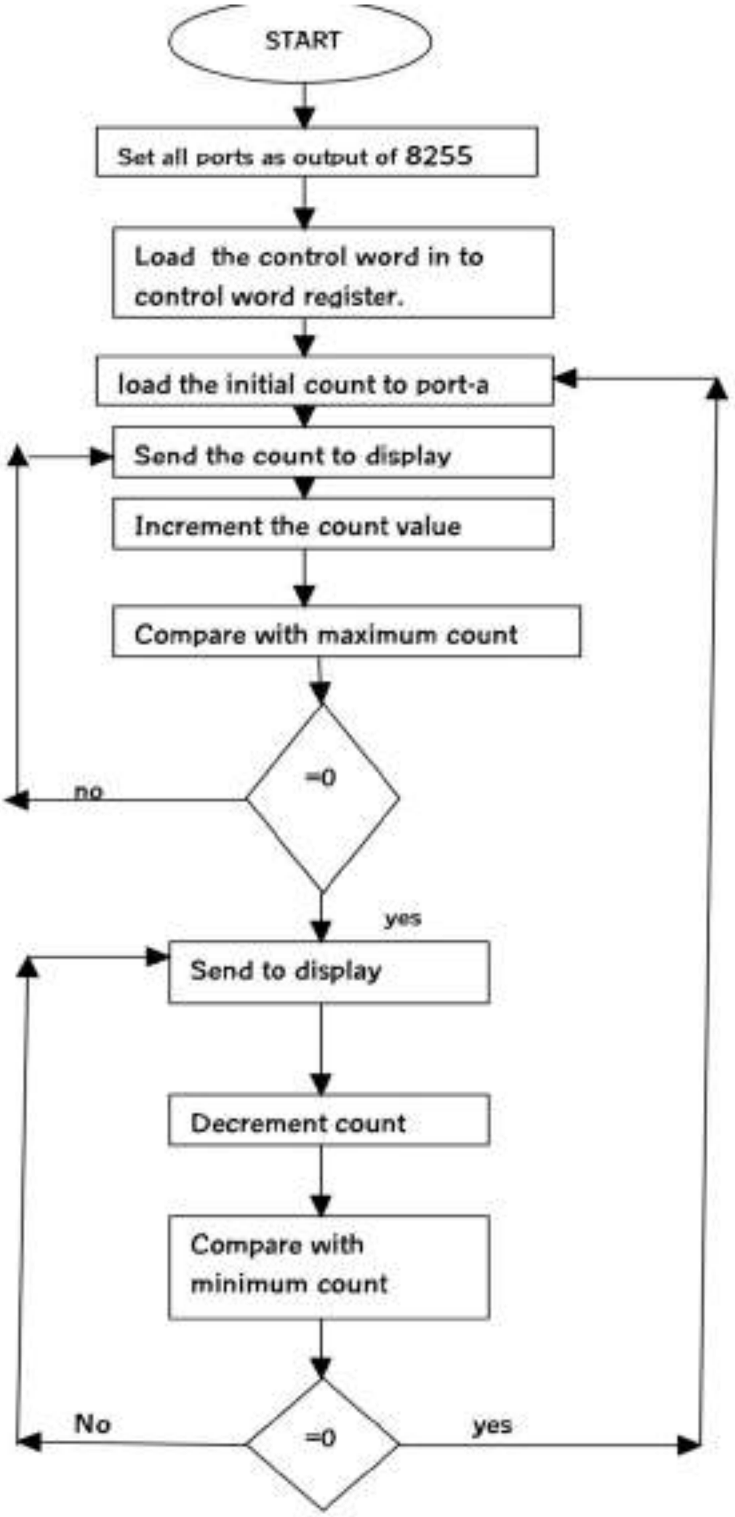
RESULT :-

Generating the different square wave with different frequencies.



3.TRIANGULAR WAVE GENERATION

FLOWCHART


START
Set all ports as output of 8255
Load the control word in to control word register.
load the initial count to port-a
Send the count to display
Increment the count value
Compare with maximum count
=0
no
yes
Send to display
Decrement count
Compare with minimum count
=0
No
yes

5.TRIANGULAR WAVE GENERATION

DATE:-

EXPNO:-

AIM:-

write an assembly language program to generate triangular wave using DAC through 8255 PPI

APPARATUS:-

ESA -86 training kit, DAC card and CRO

THEORY:-

In this circuit the 8086 processor is interfaced with 8255 in mode-0 and set all the ports are set to output. The output of port-a is connected to DAC which converts the digital input to corresponding analog output. This send to CRO to display. Initially the port-A is loaded with 00 and the corresponding analog output is send to CRO. And increment count and then compare to maximum value. If it is less than the maximum count then send to display. After that the count value is decremented, after decrement compare with minimum value. If it is greater than minimum value then send to display. Once it is reached to minimum then again start increment the count . this process is repeat .

ALGORITHM:-

Step1: set all ports as output of 8255 in mode-0

Step2: Load control word into control word register.

Step3: initialize port-A with 00 and output to port-A

Step4: send to display through DAC

Step5: Increment the port-A value and compare with maximum count

Step6: If it is less than the maximum count go to step4 if no goto next step

Step7: send to display count through DAC

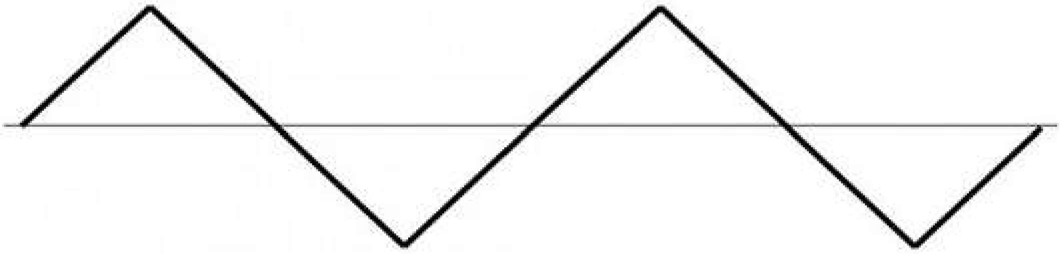
Step8: Decrement count and compare to minimum count

Step9: If it is greater than the minimum count goto step 7, if no goto step 3

ASSEMBLY LANGUAGE PROGRAM FOR TRIANGULAR WAVE

ADDRESS	OP CODE	MNEMONIC	OPERAND	COMMENT
2000	BA, E6, FF	MOV	DX,0FFE6	Set all the ports As output in mode-0
2003	B0,80	MOV	AL,80	Send control Word to control Word register.
2005	EE	OUT	DX,AL	
2006	BA,E0,FF	MOV	DX,0FFE0	Send initial count to port-A To display
2009	B0,00	MOV	AL,00	
200B	EE	OUT	DX,AL	
200C	FE,C0	INC	AL	Is reach to maximum
200E	3C,FF	CMP	AL,0FF	If no out display
2010	75,F9	JNZ	200B	
2012	EE	OUT	DX,AL	Decrement count is count reaches to minimum if no out to display
2013	FE,C8	DEC	AL	
2015	3C,00	CMP	AL,00	
2017	75,F9	JNZ	2012	
2019	EB,EE	JMP	2009	Go to initial count

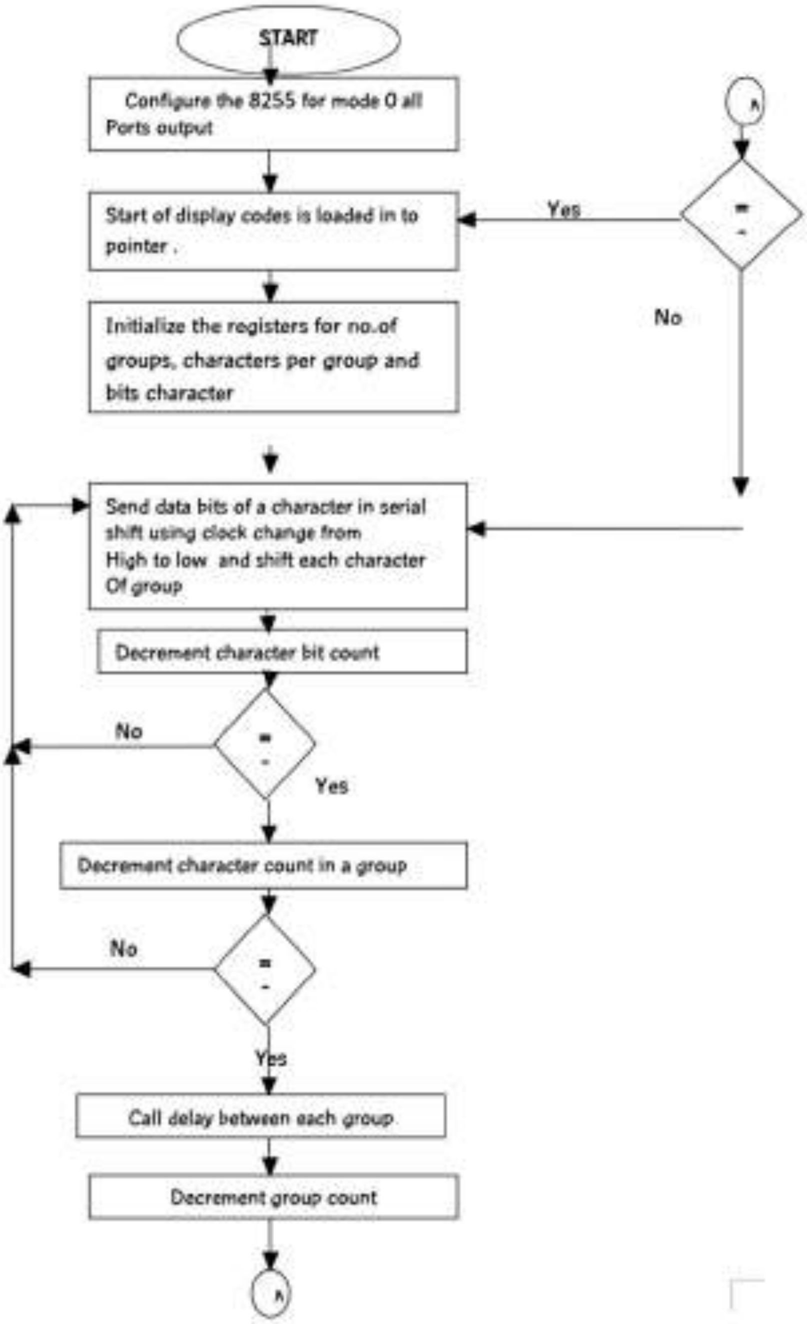
RESULT :-Triangular wave is generated with different frequencies with different amplitudes



Note:- The amplitude and frequency depends upon maximum and minimum count to be loaded into port-A

4.SEVEN SEGMENT DISPLAY

FLOW CHART


START
Configure the 8255 for mode 0 all Ports output
Start of display codes is loaded in to pointer .
Initialize the registers for no.of groups, characters per group and bits character
Send data bits of a character in serial shift using clock change from High to low and shift each character Of group
Decrement character bit count
No
Yes
Decrement character count in a group
No
Yes
Call delay between each group
Decrement group count
A
A
Yes
No

4.SEVEN SEGMENT DISPLAY

DATE:-
EXP.NO:-

AIM:-

Write an Assembly Language Program to interface the sevensegment display and print the required characters using 8086 through 8255

APPARATUS :-

MASM 32 ASSEMBLER , ESA-86/88 KIT

ALGORITHM :-

step1: Set SI register as pointer for data.

Step2: Initialize the group counter register.

Step3: Initialize the character count register .

Step4: Initialize the bit count register

Step5: Load the character from the memory specified by pointer.

Step6: Increment the memory pointer for next character.

Step7: Find the next bit of character.

Step8: Shift that bit to specific port(PB).

Step9:set clock and send to specific port(PC).

Step10:reset the clock and send to specific port(PC).

Step11: Decrement bit count register, check, if it zero goto next step, if not goto step7.

Step 12: Decrement character count register, check, if it zero go to next count, if not goto Step4.

Step 13: Call delay program between each group

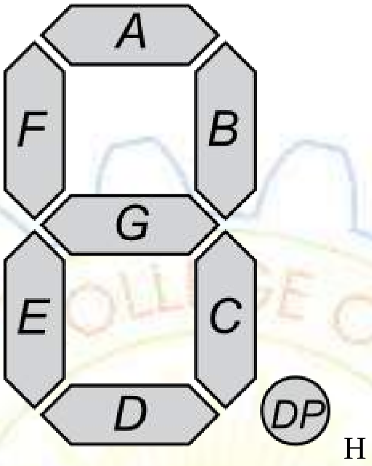
Step14: Decrement group counter , check, if it zero goto next step , if not gotogoto step 3

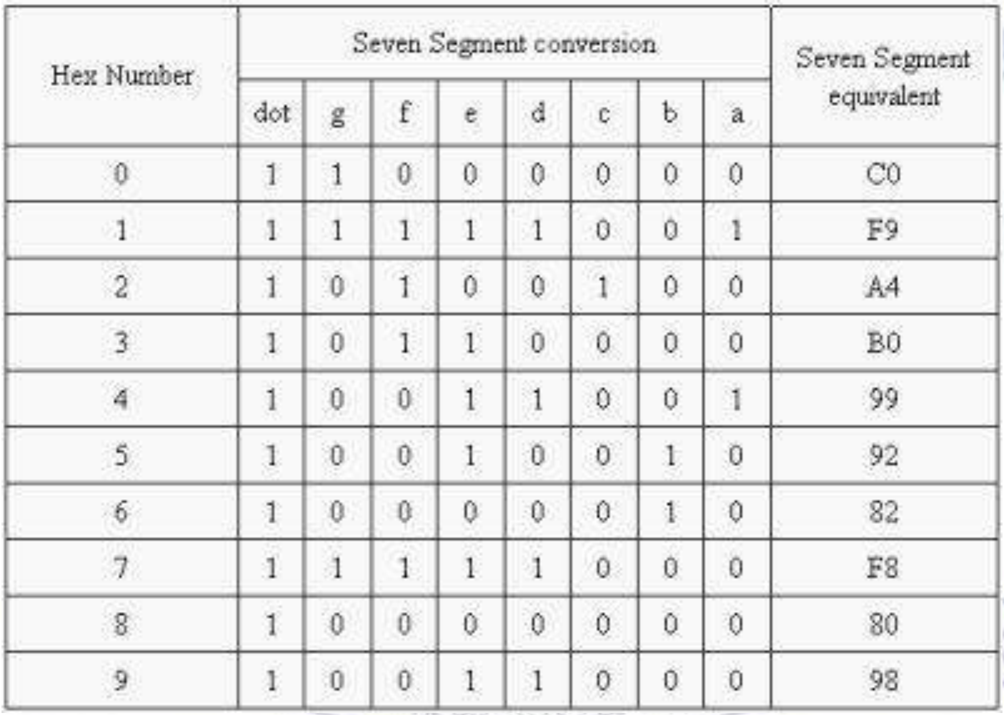
Step 15:go to step 1.

THEORY:

There are four digit 7 segment display driven by the outputs of four cascaded serial-in-parallel-out shift registers. Data to be displayer is transmitted serially, bit by bit, to the interface over the port line PB0. Each bit is clocked into the shift registers by providing a common clock through the port line PC0. Thus , information for all the four digits is provided by 32 bits clocked into the shift registers serially.

Display Codes: since the outputs of shift registers are connected to the cathode sides of LED segments, low input must be given to the segments for making them glow and high inputs for making them blank. Each display has 7 bar segments and a dot as in shown in figure below. For displaying any character its corresponding segments must be given blow inputs.





Hex Number	Seven Segment conversion								Seven Segment equivalent
	dot	g	f	e	d	c	b	a	
0	1	1	0	0	0	0	0	0	C0
1	1	1	1	1	1	0	0	1	F9
2	1	0	1	0	0	1	0	0	A4
3	1	0	1	1	0	0	0	0	B0
4	1	0	0	1	1	0	0	1	99
5	1	0	0	1	0	0	1	0	92
6	1	0	0	0	0	0	1	0	82
7	1	1	1	1	1	0	0	0	F8
8	1	0	0	0	0	0	0	0	80
9	1	0	0	1	1	0	0	0	98

ASSEMBLY LANGUAGE PROGRAM:-

```
          MOV     DX,0FFE6H
          MOV     AL,80H
          OUT     DX,AL
LOOP4:    MOV     SI,2050H
          MOV     CL,05
LOOP3:    MOV     CH,04
LOOP2:    MOV     BL,08
:         MOV     AL,[SI]
          INC     SI
LOOP1:    ROL     AL,1
```

```
          MOV       DX,0FFE2H
          OUT       DX,AL
          MOV       AH,AL
          MOV       AL,1
          MOV       DX,0FFE4H
          OUT       DX,AL
          DEC       AL
          OUT       DX,AL
          MOV       AL,AH
          DEC       BL
          JNZ       LOOP1
          DEC       CH
          JNZ       LOOP2
          CALL      DELAY
          DEC       CL
          JNZ       LOOP3
          JMP       LOOP4
```

ADDRESS	OPCODE	LABEL	MNEMONIC	OPERANDS	COMMENTS
2000	BA,E7,FF		MOV	DX, 0FFE6	Configure 8255 All ports output
2003	B0,80		MOV	AL,80H	Control word to set all ports output
2005	EE		OUT	DX,AL	Load the control word in To control word register
2006	BE,20,50	LOOP 4	MOV	SI,2050H	Start of display code
2009	B1,05		MOV	CL,5	5 groups to display
200B	B5,04	LOOP3	MOV	CH,4	4 Characters per group
200D	B3,08	LOOP2	MOV	BL,8	8 Bits per character
200F	8A,04		MOV	AL,[SI]	Character get the display Code
2011	46		INC	SI	Increment pointer for next Character.
2012	DO,CO	LOOP1	ROL	AL,1	Get one data bit
2014	BA,E3,FF		MOV	DX,0FFE2	Port B initialization
2017	EE		OUT	DX,AL	Data bit output to port B
2018	88, C4		MOV	AH,AL	Store temporarily the acc. In to AH
201A	B0,1		MOV	AL,1	Output the clock
201C	BA,E5,FF		MOV	DX,0FFE4	Instillation the port c
201F	EE		OUT	DX,AL	Output the clock through Port c
2020	FE,C8		DEC	AL	To shift register
2022	EE		OUT	DX,AL	Output the clock
2023	88,E0		MOV	AL,AH	Load temporary stored data Into AL
2025	FE,CB		DEC	BL	All bits are over?
2027	75,E9		JNZ	LOOP1 (2012)	No continue
2029	FE,CD		DEC	CH	All characters over?
202B	75,E0		JNZ	LOOP2	No continue

				(200D)	
202D	E8,06,00		CALL	DELAY (2040)	Introduce delay
2030	FE,C9		DEC	CL	All groups are over.
2032	75,D7		JNZ	LOOP3 (200B)	No to continue
2034	EB,D0		JMP	LOOP4 (2006)	Yes start from beginning

DELAY PROGRAM

2040	51	DELAY	PUSH	CX	Delay subroutine
			PUSH	DX	
2041	B9,FF,00		MOV	CX,00FF	
2044	BA,FF,FF	LOOP2	MOV	DX,0FFFF	
2047	4A	LOOP1	DEC	DX	
2048	75,FD		JNZ	LOOP1 (2047)	
204A	49		DEC	CX	
204B	75,F7		JNZ	LOOP2 (2044)	
204D	5A		POP	DX	
204E	59		POP	CX	
204F	C3		RET		

STRING

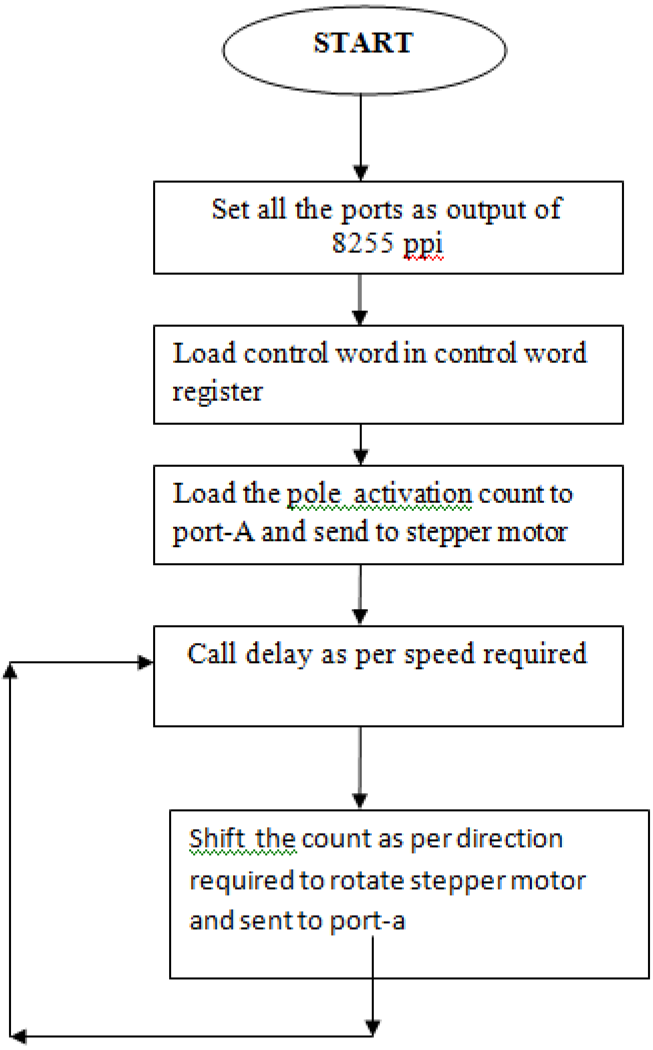
2050	0BF	0CC	0CC	0C6
2054	0BF	86	0C0	0C6
2058	0C6	86	92	88
205C	0BF	0CC	0C7	86
2060	0F8	0C0	0C0	0C0

RESULT:- The output is displayed as follows according to above code

```
C     r     r     --

C     O     E     --

A     S     E     C

E     L     r     --

O     O     O     7
```

5.STEPPER MOTOR

FLOW CHART


START
Set all the ports as output of 8255 ppi
Load control word in control word register
Load the pole activation count to port-A and send to stepper motor
Call delay as per speed required
Shift the count as per direction required to rotate stepper motor and sent to port-a

5.STEPPER MOTOR

DATE:-

EXP.NO:-

AIM:-

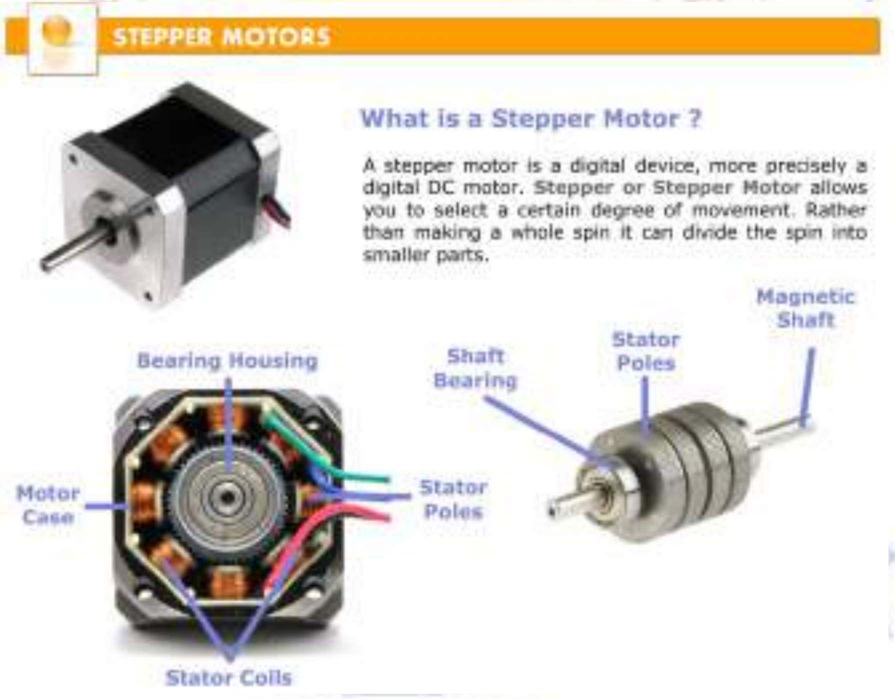
Write an Assembly Language Program to rotate the stepper motor using 8255 ppi in mode-0

APPARATUS :-

ESA-86/88 KIT, CRO, DAC

THEORY:-

In this circuit the 8086 processor is interfaced with 8255 in mode-0 and set all the ports are set to output. The output of port-A is connected to stepper motor And call the delay as per speed requirement. As per direction rotate stepper and shift the pole action with speed requirement. And repeat the same for continuously rotating the motor.



What is a Stepper Motor ?

A stepper motor is a digital device, more precisely a digital DC motor. Stepper or Stepper Motor allows you to select a certain degree of movement. Rather than making a whole spin it can divide the spin into smaller parts.

ALGORITHM :-

step1: Set all ports as output of 8255 in mode-0

Step2: Load control word into control word register.

Step3: Initialize port-a with pole activation count and output to port-A.

Step4: Call Delay according to speed.

Step5: rotate as per direction shift the poleaction count and output to port-A

Step6: Goto step 4

ASSEMBLY LANGUAGE PROGRAM:

```
            MOV    DX,0FFE6H
            MOV     AL,80H
            OUT     DX,AL
            MOV   DX,0FFE0H
            MOV    AL,88H
      LOOP1:    OUT    DX,AL
            CALL  2050H(DELAY)
            ROR    AL,1
            JMP    LOOP1
```

ASSEMBLY LANGUAGE PROGRAM FOR SQUARE WAVE

DATA SEGMENT OFFSET : 076AH

ADDRESS IN HEXA	OP CODE IN HEXA	MNEMONIC	OPERAND	COMMENT
2000	BA,E6,FF	MOV	DX,0FFE6	Set all ports of 8255 as output In mode -0
2003	B0,80	MOV	AL,80H	
2005	EE	OUT	DX,AL	Load control word in to contro Word register
2006	BA,E0,FF	MOV	DX,0FFE0	Out the initial word into port-A
2009	B0,88	MOV	AL,88	
200B	EE	OUT	DX,AL	Send to display through DAC
200C	E8,06,00	CALL	2041 (DELAY)	Call delay program for on/off time
200F	D0,C8	ROR	AL,1	Complement for off time
2011	EB,FB	JMP	200B	Send to display for off time

DELAY PROGRAM

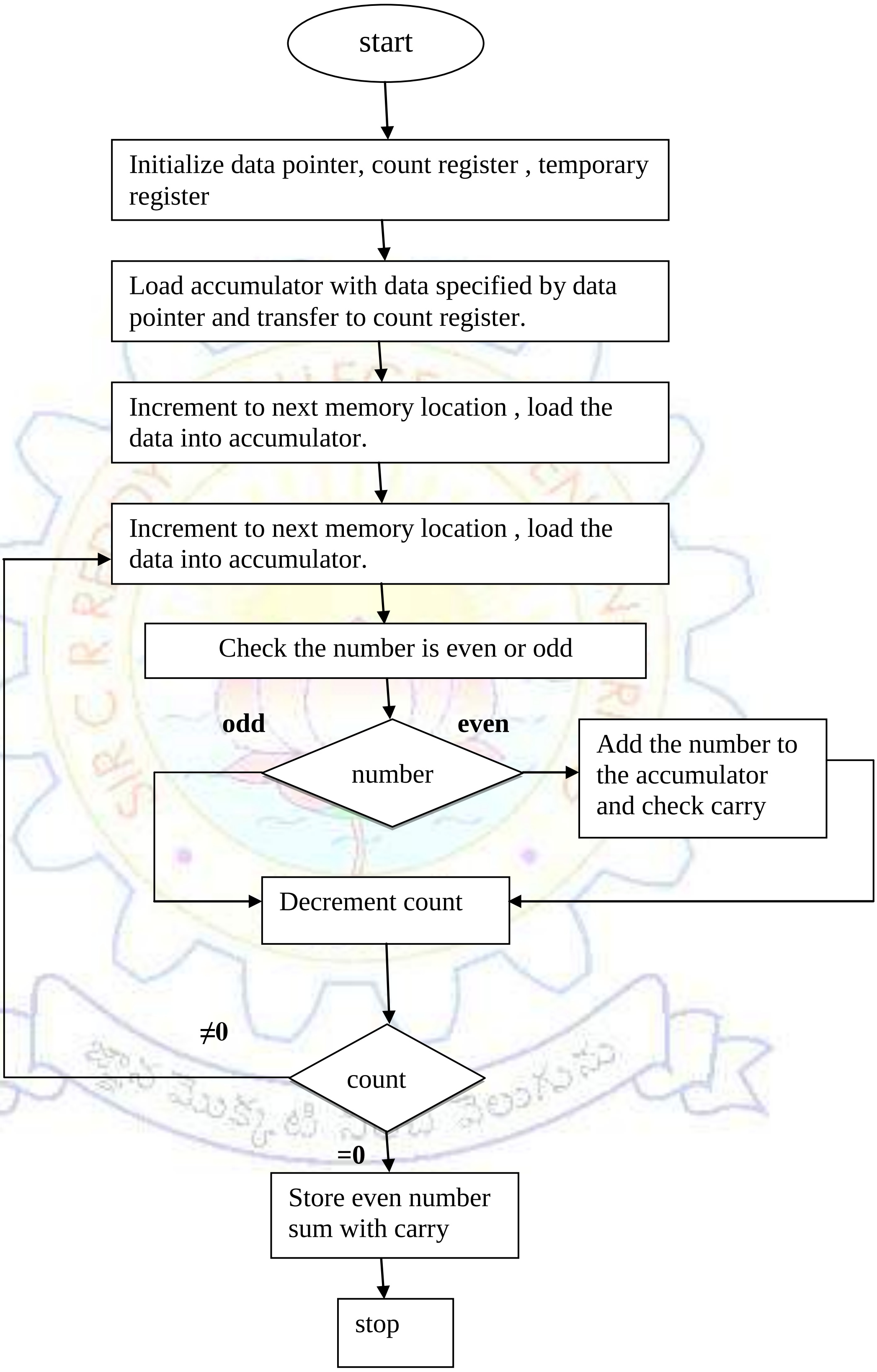
2041	B9,0F,00		MOV	CX,00FF	
2044	BA,FF,FF	LOOP2	MOV	DX,0FFFF	
2047	4A	LOOP1	DEC	DX	
2048	75,FD		JNZ	LOOP1 (2047)	
204A	49		DEC	CX	
204B	75,F7		JNZ	LOOP2 (2044)	
204D	C3		RET		

RESULT :-

Rotating the stepper motor with different directions and with different speeds.

8051 PROGRAMS

1.EVEN SUM IN ARRAY OF DATA


start
Initialize data pointer, count register , temporary register
Load accumulator with data specified by data pointer and transfer to count register.
Increment to next memory location , load the data into accumulator.
Increment to next memory location , load the data into accumulator.
Check the number is even or odd
odd
even
number
Add the number to the accumulator and check carry
Decrement count
≠0
count
=0
Store even number sum with carry
stop

1. EVEN SUM IN ARRAY OF DATA

EXP NO:
DATE:

AIM: To find the sum of even numbers in the given array of data.

APPARATUS:

KEIL µVISION

ALGORITHM:

Step1: Initialize the count register(r_1).

Step2: Initialize the data pointer (40h).

Step3: Initialize the temporary registers(r_2).

Step4: Load the accumulator with data address specified by data pointer and transfer to count register.

Step5: Load the next data from memory into accumulator.

Step6: Rotate right through carry, the accumulator to check even or odd.,

Step7: Rotate left the accumulator for the given data.

Step8: Add accumulator with temporary register.

Step9: Store sum into temporary register.

Step10: Increment count.

Step11: If count =0, Go to next step.

If count ≠0, Go to step5.

Step12: Store the sum of even numbers into memory.

Step13: Store carry of even numbers into memory.

Step14: End the programme.

ASSEMBLY LANGUAGE PROGRAM

DATA SEGMENT OFFSET : 076AH

ADDRESS IN HEXA	OP CODE IN HEXA	MNEMONICS & OPERAND	COMMENTS
0000	C3	CLR C	Clear Carry
0001	7B00	MOV $R_{3,}$ #00H	Initialize Register
0003	7840	MOV R_0,#40H	Initialize The Data Pointer
0005	E6	MOV A,@R_0	Load The Data Into Accumulator
0006	F9	MOV R_1,A	Load Data Into Count Reg.
0007	7A00	MOV $R_{2,\#00}$	Initialize Temprory Register
0009	08	LOOP2 INC R_0	Increment Next Memory Location
000A	E6	MOV A,@R_0	Load The Data Into Acc.
000B	B	RRC A	Rotate Acc. Right Through Carry.
000C	4006	JC LOOP1	If Carry Go To Loop
000E	33	RLC A	If There Is No Carry Rotate Left
000F	2A	ADD A,R_2	Add Acc Data & Temp Register
0010	FA	MOV $R_{2,}$A	Store Added Data In R2 Register
0011	5001	JNC LOOP1	Jump If Not Zero Loop1
0013	0B	INC R_3	Increment Or Store Data R3 Register
0014	D9F3	LOOP1 DJNZ R_1, LOOP2	Decrement Count And Repeat The
0016	08	INC R_0	Increment Data Pointer
0017	EA	MOV A,R_2	Load Temporary Register Data
0018	F6	MOV @R_0,A	Move Accumulator Data Into Data Pointer.
0019	08	INC R_0	Increment
001A	E8	MOV A,R_3	Load Sum Of Even Numbers Into Accumulator
001B	F6	MOV @$R_{0,}$ A	Store The Sum Of Even Numbers
		END	End Of The Program

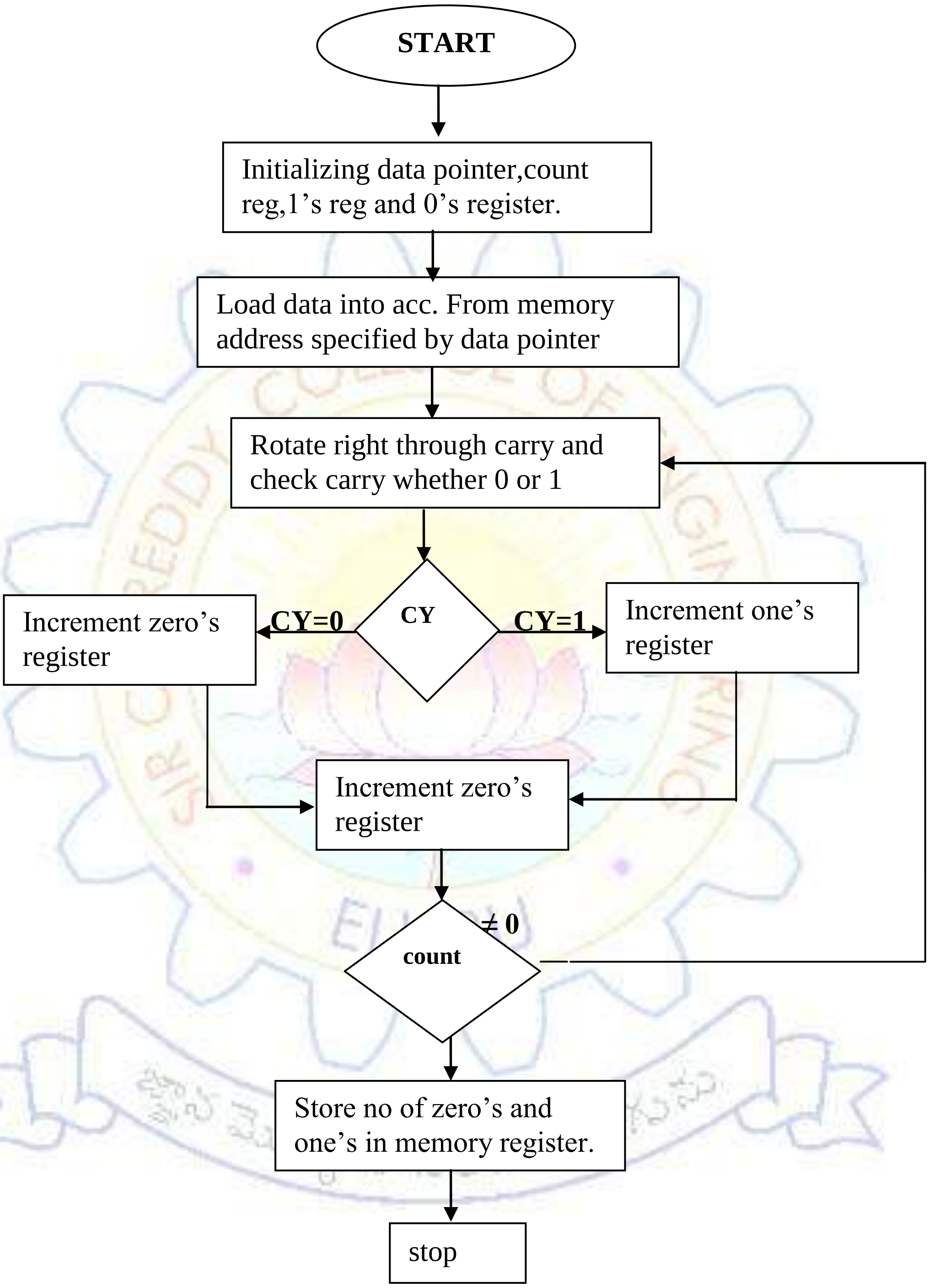
OUTPUT:

S.NO	COUNT(40H)	41H	42H	43H	44H	45H	46H	47H	48H

RESULT:

2.COUNTING NO.OF ZEROS & ONES

FLOWCHART:


START
Initializing data pointer,count reg,1's reg and 0's register.
Load data into acc. From memory address specified by data pointer
Rotate right through carry and check carry whether 0 or 1
CY
CY=0
CY=1
Increment zero's register
Increment one's register
Increment zero's register
count
≠ 0
Store no of zero's and one's in memory register.
stop

2.COUNTING NO.OF ZEROS & ONES

EXP NO:
DATE:

AIM:

To Find The Number Of Zero's And Number Of One's In The Given Data

APPARATUS:

KEIL µVISION

ALGORITHM:

STEP1: Initialize The Data Pointer (R_0).

STEP2: Initialize The Count Register (R_1).

STEP3: Initialize The One's Register (R_2).

STEP4: Initialize The Zero's Register (R_3).

STEP5: Load Data Into Accumulator From Data Pointer (40h).

STEP6: Rotate Accumulator Right Through Carry.

STEP7: Check The Carry Flag

If Cy=1 Then Go To Step10

If Cy=0 Go To Next Step.

STEP8: Increment Zero's Register R_3 By '1' .

STEP9:S Jump To Step11.

STEP10: Increment The One's Register(R_2).

STEP11: Decrement The Count Register(R_1).

STEP12: Check The Count

If Count = 0 Go To Next Step

If Count≠ 0 Go To Step6.

STEP13:Store One's Register Into Memory Specified By Data Pointer.

STEP14: Store Zero's Register Into Memory Specified By Data Pointer.

STEP15: End The Program.

ASSEMBLY LANGUAGE PROGRAM:

DATA SEGMENT OFFSET : 076AH

ADDRESS IN HEXA	OP CODE IN HEXA	MNEMONICS & OPERAND	COMMENTS
0000	7840	MOV R0, #40H	Initialization Of Data Pointer.
0002	E6	MOV A,@R0	Load The Data Into Accumulator.
0003	7900	MOVR1,#00	Initialization Of One’s Count Register.
0005	7A00	MOV R2,#00	Initialization Of Zero’s Count
0007	7B08	MOV R3, #08	Initializing Count Register.
0009	B	LOOP3 RRC A	Locate The Acc Right Through carry
000A .	4003	LOOP1 JC	If Carry Exists Go To Loop1 To Inc Check 0 Or 1.
000C	0A	INC R2	Increment One’s Register.
000D	001 loop2	SJMP LOOP2	After Checking Condition Go To
.000F	O9	LOOP1 INC R1	Increment One’s Register
0010	DBF3	LOOP2 R3, LOOP3	If Count To Repeat Loop3.
0012	8A41	MOV 41H, R1	Put No Of One’s In R1 Of 42h memory
0014	8A42	MOV 42H, R2	Put No Of Zero’s In R2 Of 42h memory
		END	End Of The Program

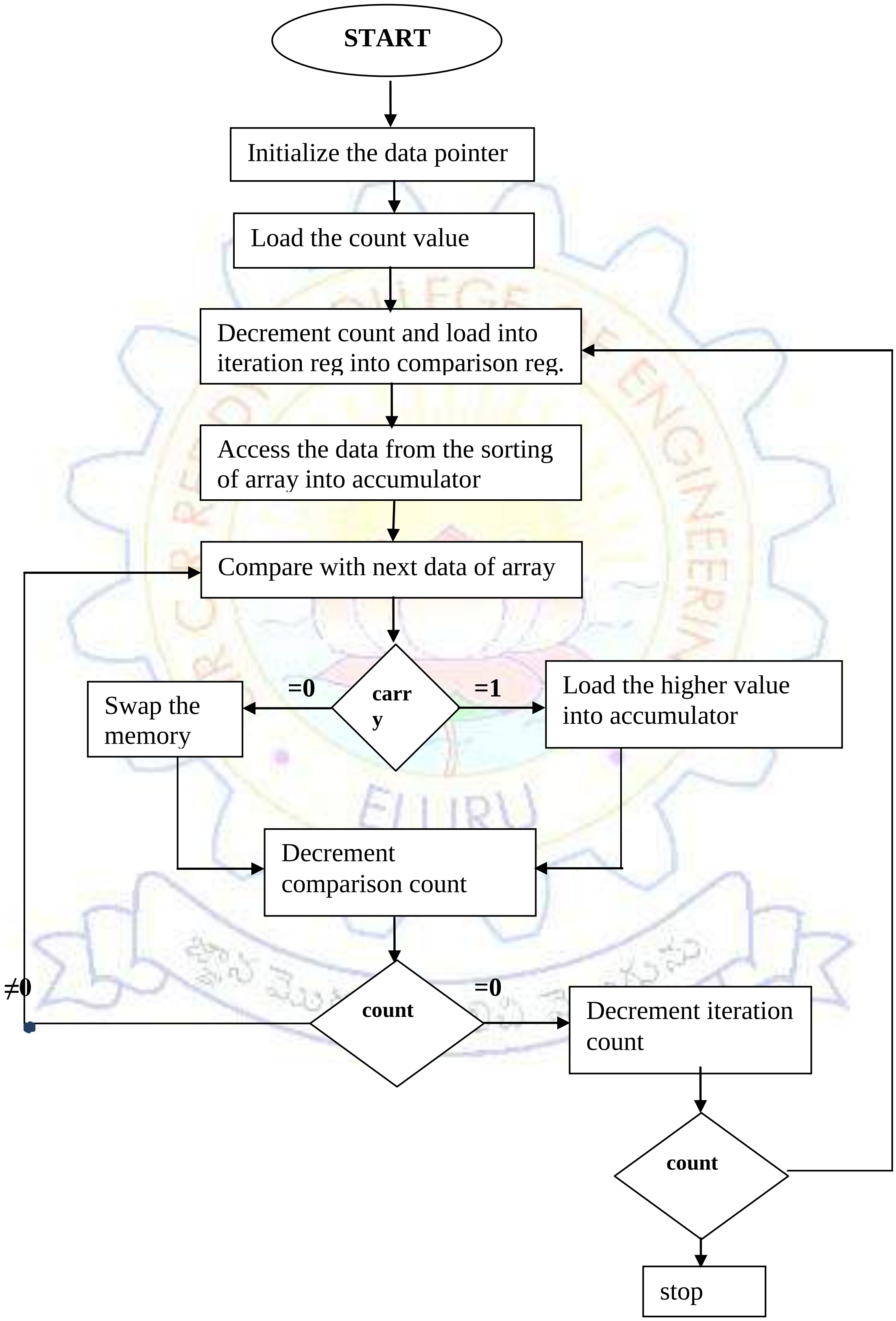
OUTPUT:

S.NO	INPUT(8BIT DATA) (40H)	NO.OF 0’S IN 0’S REGISTER(41H)	NO.OF ONE’S IN 1’S REGISTER (42H)

RESULT:

3.SORTING IN 8051

FLOWCHART:


START
Initialize the data pointer
Load the count value
Decrement count and load into iteration reg into comparison reg.
Access the data from the sorting of array into accumulator
Compare with next data of array
=0
carr y
=1
Swap the memory
Load the higher value into accumulator
Decrement comparison count
≠0
count
=0
Decrement iteration count
count
stop

3.SORTING IN 8051

EXP NO:
DATE:

AIM:

Write an assembly language program for 8051 to perform sorting of the array.

APPARATUS:

1. KEIL µVISION

ALGORITHM:

STEP1: SET SI register as pointer for data.

STEP2: load the count value.

STEP3: Decrement The Count Value.

STEP4: Load If Into Iteration Register Then Into Comparison Register.

STEP5: Access The Data From The Sorting Of The Array Into Accumulator.

STEP6: Compare With The Next Data Of The Array Pointer.

STEP7: Check The Carry If Carry Exists Stores Highest Value Into The Accumulator Then Go To Next Step9. If Carry Does Not Exists Go To Next Step.

STEP8: Swap The Memory Contents.

STEP9: Decrement The Comparison Count
If It Is Zero Go To Next Step
If It Not Zero Go To Step 6

STEP10: decrement the iteration count

STEP11: If It Non-Zero Go To Step4, If It Is Zero Go To Next Step.

STEP12: Stop The Program.

ASSEMBLY LANGUAGE PROGRAM:

ASCENDING ORDER

DATA SEGMENT OFFSET : 076AH

Address in hexa	Opcode in hexa	Mnemonic and operand	comments
0000	7940	MOV R_1, #40H	Initialize The Data Count
0002	E540	MOV A, 40H	Load The Data In Accumulator.
0004	14	DEC A	Decrement Accumulator.
0005	FA	MOV R_2, A	Load The Iteration Onto Register From
0006	EA	LOOP4 MOV A, R_2	Load The Value Of R_2in Accumulator.
0007	FB	MOV R_3, A	Load The Accumulator Value
0008	7941	MOV R_1, #41H	Initialize Data Pointer.
000A	E7	loop3MOV A, @R_1	Load Data Into Accumulator
000B	09	INC R_1	Increment R_1 Register.
000C	87F0	MOVA,@R_1	Load The Increment Data Into B Location
000E	B5F0000	CJNC a,b,loop1	Compare A And B If Not Equal Decrement Data Pointer.
0011	4005	LOOP1 JC LOOP2	Carry Exist Go To Swap If Not Go To Next Step.
0013	F7	MOV @R_1, A	Load The Data Into R_1register From Accumulator.
0014	19	DEC R_1	Decrement count value
0015	A7F0	MOV @R_1,B	Move or load the data in R_1 register from b
0017	09	INC R_1	Increment count value
0018	DBF0	LOOP2 DJNZ R_3	Decrement the count if not Zero
001A	DAEA	DJNZ R_2	If non zero, the value decrement else Go to next iteration
		End	Stop execution

ASSEMBLY LANGUAGE PROGRAM:DESCENDING ORDER

DATA SEGMENT OFFSET : 076AH

Address in hexa	**Opcode in hexa**	**Mnemonic and operand**	**comments**
0000	7940	MOV $R_{1,}$ #40H	Initialize The Data Count
0002	E540	MOV A, 40H	Load The Data In Accumulator.
0004	14	DEC A	Decrement Accumulator.
0005	FA	MOV $R_{2,}$ A	LoadThe Iteration Onto Register From Acc.
0006	EA	LOOP4 MOV A, R_2	Load The Value Of R_2in Accumulator.
0007	FB	MOV R_3, A	Load The Accumulator Value Into reg.R3
0008	7941	MOV $R_{1,}$ #41H	Initialize Data Pointer.
000A	E7	LOOP3 MOV A, @R_1	Load Data Into Accumulator
000B	09	INC R_1	increment R_1 Register.
000C	87F0	MOVA,@R_1	Load The Increment Data Into B Location
000E	B5F0000	CJNC A,B,LOOP1	Compare A And B If Not Equal Decrement Data Pointer.
0011	4005	LOOP1 JNC	Carry Exist Go To next iteration if not swap
0013	F7	MOV @$R_{1,}$ A	Load The Data Into R_1register From Acc.
0014	19	DEC R_1	Decrement count value
0015	A7F0	MOV @R_1,B	Move or load the data in R_1 register from b
0017	09	INC R_1	Increment count value
0018	DBF0	LOOP2DJNZ R_3	Decrement the count if not Zero
001A	DAEA	DJNZ R_2	If non zero, the value decrement else Go to next iteration
		End	Stop execution

OBSERVATIONS:ASCENDING

S.NO	COUNT(40H)	41H	42H	43H	44H	45H	46H	47H	48H

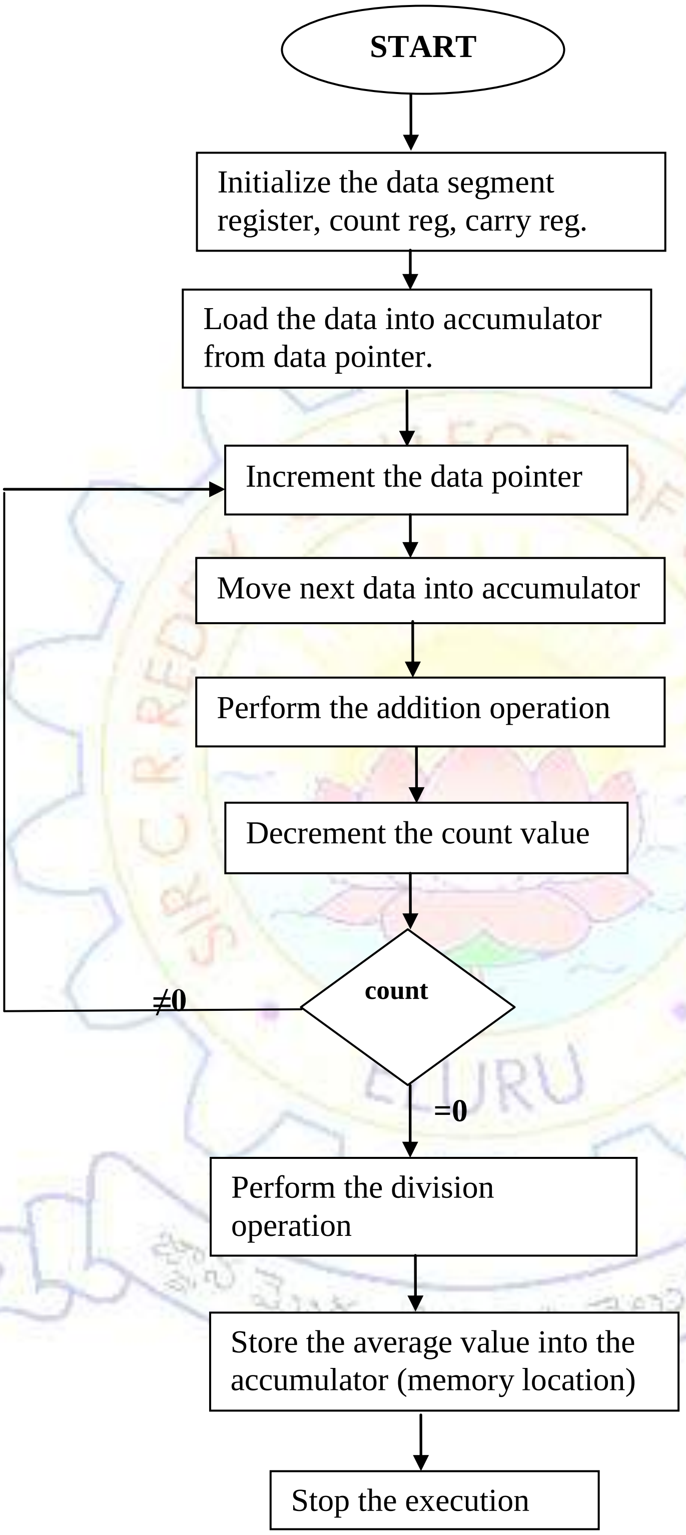
DESCENDING

S.NO	COUNT(40H)	41H	42H	43H	44H	45H	46H	47H	48H

RESULT:

4.AVERAGE OF ARRAY OF NUMBERS

FLOWCHART:


START
Initialize the data segment register, count reg, carry reg.
Load the data into accumulator from data pointer.
Increment the data pointer
Move next data into accumulator
Perform the addition operation
Decrement the count value
count
≠0
=0
Perform the division operation
Store the average value into the accumulator (memory location)
Stop the execution

4.AVERAGE OF ARRAY OF NUMBERS

DATE:

EXP.NO:

AIM:
To write the assembly language program to find the average of given numbers.

APPARATUS:

KEIL µVISION

PROCEDURE:

STEP1: Initialize the data pointer, count register and carry register.

STEP2: Load the data from the memory location into accumulator.

STEP3: Move the data into register B.

STEP4: Increment the value in the accumulator by one and move it onto the register.

STEP5: Initialize the register R_2 with ‘0’.

STEP6: Increment data pointer and load the data into accumulator.

STEP7: Add the register to data and accumulator next data.

STEP8: Check the count register, If 0 go to next step else go to step 6 .

STEP9: Divide the accumulator with count register.

STEP10: Store the sum and carry in the memory location.

STEP11: End program.

ASSEMBLY LANGUAGE PROGRAM:

DATA SEGMENT offset : 076AH

Address in hexa	**Opcode in hexa**	**Mnemonic and operand**	**comments**
0000	7840	MOV R_0,#40H	Initialize data pointer
0002	E6	MOV A,@R_0	load the data from data pointer to Acc.
0003	F9	MOV R_1,A	Load the data into count register
0004	F5F0	MOV B,A	Load the accumulator data into B register
0006	7A00	MOV R_2,#00	Initialize carry register
0008	E8	LOOP1 INC R_0	Increment data pointer
0009	E6	MOV A,@R_0	Load the data into accumulator
000A	2A	APP A,R_2	Load the carry register data into accumulator
000B	FA	MOV R_2,A	Store sum into R_2
000C	D9FA	DJNZ R_1,LOOP1	Check the count register if it is non zero go to Loop1
000E	EA	MOVA,R_2	Load data into accumulator
000F	84	DIV A,B	Divide the ccumulator data with count data
0010	08	INC R_0	Increment data pointer
0011	F6	MOV @R_0,A	Store the sum in the register from the Accumulator
0012	08	INC R_0	Increment the data pointer register R_0
0013	A6F0	MOV @R_0,B	Store the carry in the register from B register
		End	Stop program

OUTPUT:

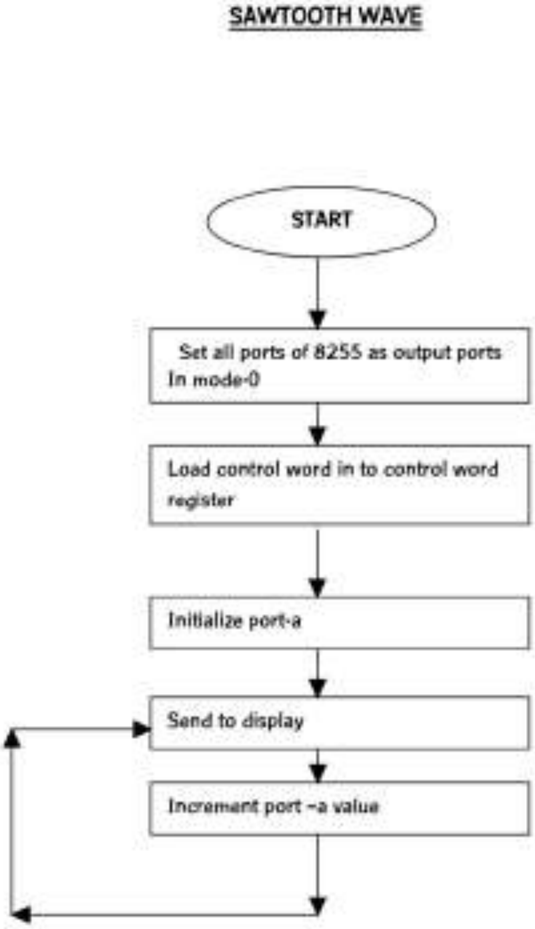
S.NO	DATA	COUNT(40H)	41H	42H	43H	44H	45H	46H sum	47H carry

RESULT:

8051 INTERFACING

1.SAWTOOTH WAVE

FLOW CHART


SAWTOOTH WAVE
START
Set all ports of 8255 as output ports In mode-0
Load control word in to control word register
Initialize port-a
Send to display
Increment port –a value

1.SAWTOOTH WAVE

AIM:-

Write an Assembly Language Program to generate sawtooth wave using DAC through 8255 PPI

APPARATUS :-

ESA-8051 KIT, DAC card and CRO

THEORY:-

In this circuit the 8051 controller is interfaced with 8255 in mode-0 and set all the ports are set to output.The output of port-A is connected to DAC which converts the digital input to corresponding analog output. The is send to CRO to display. Initially the port-A is loaded with 00 and the corresponding analog output is send to CRO. And increment port-A value continuously until the maximum value. If the maximum value is 0FF no need to compare. Once it is reached to maximum value then it will reached to initial value. And repeated the same. If the maximum is not FF then for each and every increment we should compare with maximum value if is equal or less than we should send to port-A to display. After that again start from 00 and repeat. The wave amplitude and frequency are depends on maximum count value to send to Port-A.

ALGORITHM :-

Step1: Set all ports as output of 8255 in mode-0

Step2: Load control word into controlword register.

Step3: Initialize port-a with 00 and output to port-a

Step4: Send To Display Through DAC

Step5: Increment the port-a value and go to step 4

ASSEMBLY LANGUAGE PROGRAM

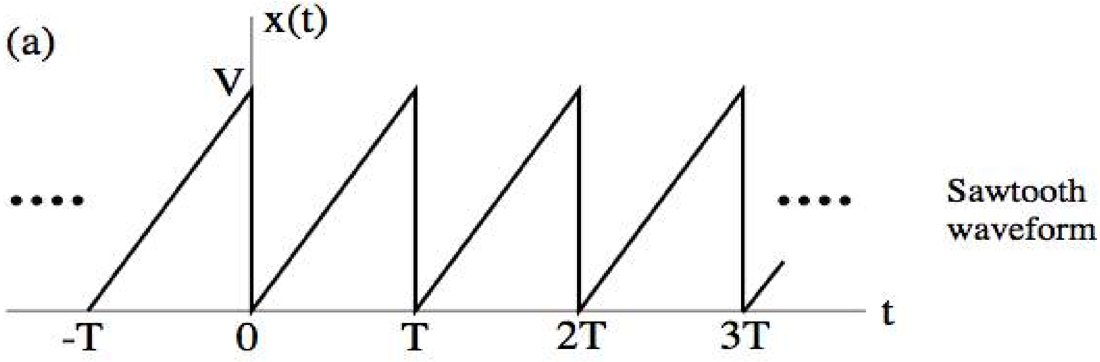
```
MOV         0A0,#0E8
MOV          R0,#03
MOV        A,#80
MOVX      @R0,A
MOV         A,#00H
MOV        R0,#00
LOOP1:    MOVX      @R0,A
INC          A
SJMP        LOOP1
```

ASSEMBLY LANGUAGE PROGRAM FOR SAWTOOH WAVE

DATA SEGMENT OFFSET : 076AH

ADDRESS IN HEXA	OP CODE IN HEXA	MNEMONICS	OPERAND	COMMENT
8000	75,A0,E8	MOV	0A0,#0E8	Set all ports of 8255 as output In mode -0
8003	78,03	MOV	R0,#03	
8005	74,80	MOV	A,#80	Load control word in to contro Word register
8007	F2	MOVX	@R0,A	Out the initial word into port-A
8008	74,00	MOV	A,#00	
800A	78,00	MOV	R0,#00	Send to display through DAC
800C	F2	MOVX	@R0,A	
800D	04	INC	A	Increment continuously
800E	80,FC	SJMP	800C	Continue to display

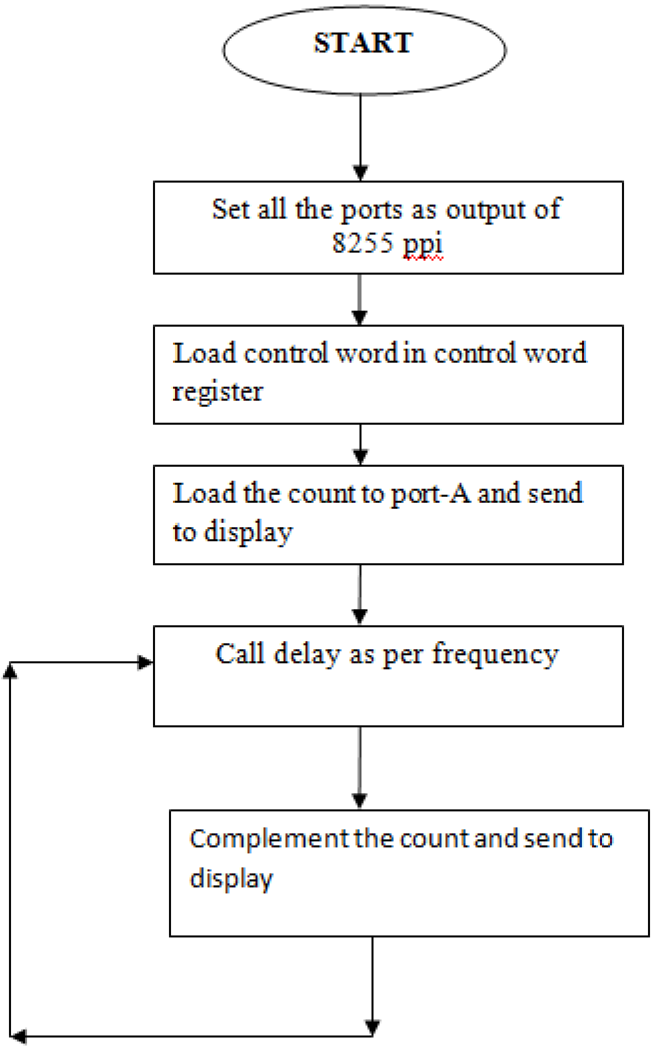
RESULT:-



Result:- Generating the sawtooth wave with different amplitudes and frequencies.

2.SQUARE WAVE GENERATION

FLOW CHART


START
Set all the ports as output of 8255 ppi
Load control word in control word register
Load the count to port-A and send to display
Call delay as per frequency
Complement the count and send to display

2.SQUAREWAVE

DATE:-
EXP.NO:-

AIM:-

Write an Assembly Language Program to generate the square wave using 8255 ppi in
mode-0

APPARATUS :-

ESA-86/88 KIT, CRO, DAC

THEORY:-

In this circuit the 8051 controller is interfaced with 8255 in mode-0 and set all the ports are set to output. The output of port-A is connected to DAC which converts the digital input to corresponding analog output. The is send to CRO to display. Initially the port-A is loaded with FF and the corresponding analog output is send to CRO. And call the delay as per frequency requirement for on time. For off time complement the count and then send to display. Repeat the above continuously. The square wave having duty cycle 50%. So ontime and off time are equal , for this we are calling same delay routine.

ALGORITHM :-

Step1: Set all ports as output of 8255 in mode-0

Step2: Load control word into control word register.

Step3: Initialize port-a with count and output to port-A.

Step4: Call Delay.

Step5: Complement the count and output to port-A

Step6: Goto step 4

ASSEMBLY LANGUAGE PROGRAM:

```
MOV        0A0,#0E8          DELAY:   MOV        R1,#0FF
MOV         R0,#03           LOOP2:      MOV      R2,#0FF
MOV       A,#80                          HERE:      DJNZ       R2,HERE
MOVX      @R0,A                               DJNZ      R1,LOOP2
MOV       R0,#00                                        RET
MOV        A,#0FF
LOOP1:    MOVX     @R0,A
LCALL     8014H(DELAY)
CPL       A
LJMP      LOOP1
```

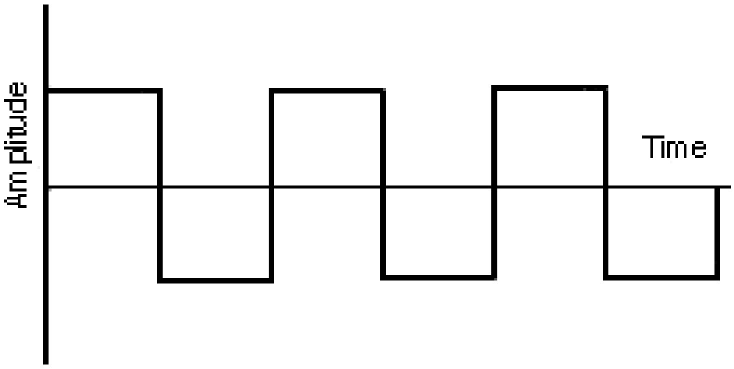
ASSEMBLY LANGUAGE PROGRAM FOR SQUARE WAVE

DATA SEGMENT OFFSET : 076AH

ADDRESS IN HEXA	OP CODE IN HEXA	MNEMONIC	OPERAND	COMMENT
8000	75,A0,E8	MOV	0A0,#0E8	Set all ports of 8255 as output In mode -0
8003	78,03	MOV	R0,#03	
8005	74,80	MOV	A,#80	
8007	F2	MOVX	@R0,A	
8008	78,00	MOV	R0,#0	Out the initial word into port-A
800A	74,FF	MOV	A,#0FF	
800C	F2	MOVX	@R0,A	
800D	12,80,14	LCALL	8014	Call delay program for on/off Time
8010	F4	CPL	A	Complement for off /on time
8011	02,80,0C	LJMP	8014	Call delay program for on/off Time
8014	79,0F	MOV	R1,#0F	Delay program
8016	7A,0FF	MOV	R2,#0FF	
8018	DA,FE	DJNZ	R2,8018	
801A	D9,FA	DJNZ	R1,8016	
801C	22	RET		

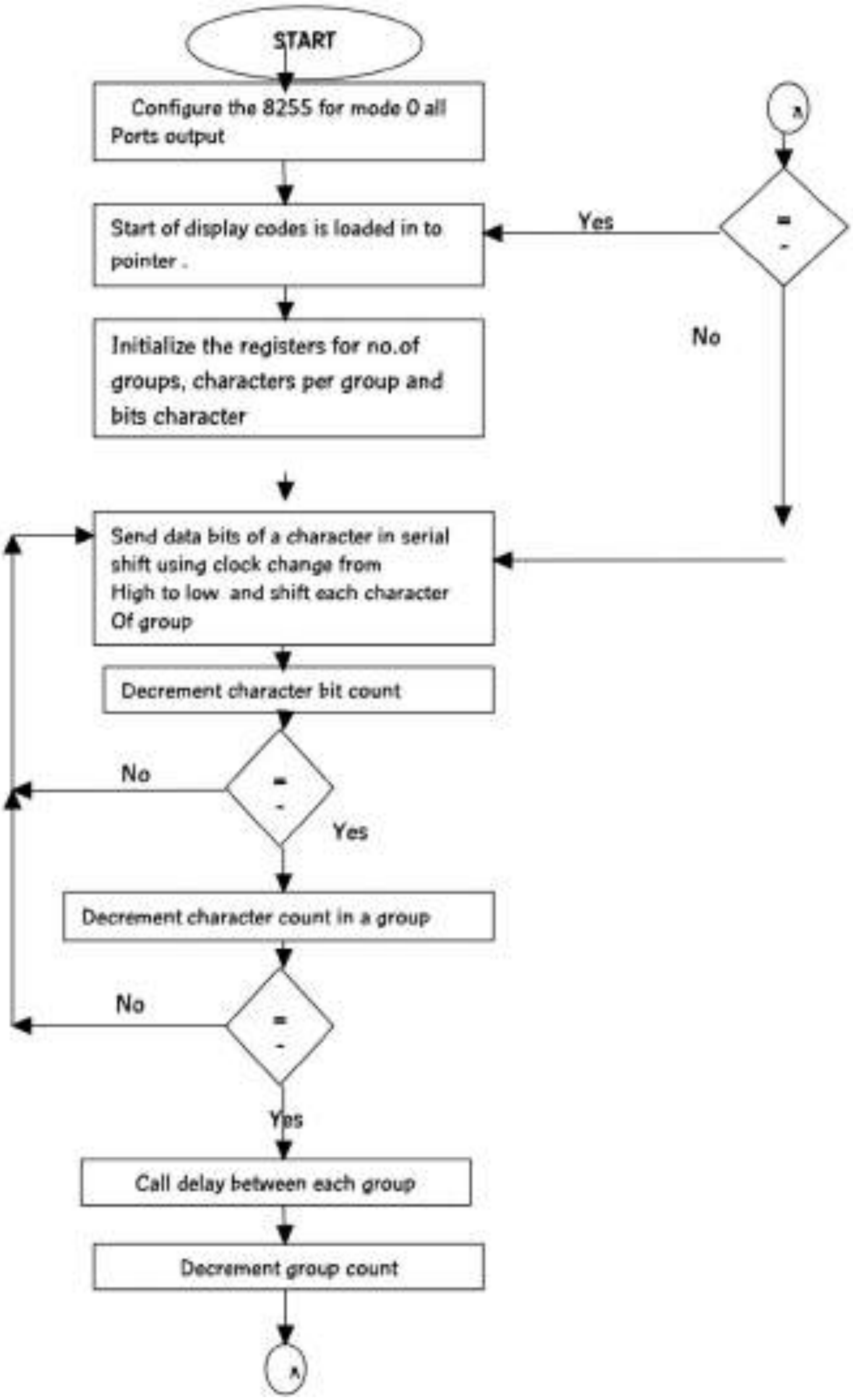
RESULT :-

Generating the different square wave with different frequencies.



3.SEVEN SEGMENT DISPLAY

FLOW CHART


START
Configure the 8255 for mode 0 all Ports output
Start of display codes is loaded in to pointer .
Initialize the registers for no.of groups, characters per group and bits character
Send data bits of a character in serial shift using clock change from High to low and shift each character Of group
Decrement character bit count
No
Yes
Decrement character count in a group
No
Yes
Call delay between each group
Decrement group count
A
A
Yes
No

3.SEVEN SEGMENT DISPLAY

DATE:-
EXP.NO:-

AIM:-

Write an Assembly Language Program to interface the seven segment display and print the required characters using 8086 through 8255

APPARATUS :-

ESA-51 KIT,7-segment card

ALGORITHM :-

step1: Set SI register as pointer for data.

Step2: Initialize the group counter register.

Step3: Initialize the character count register .

Step4: Initialize the bit count register

Step5: Load the character from the memory specified by pointer.

Step6: Increment the memory pointer for next character.

Step7: Find the next bit of character.

Step8: Shift that bit to specific port(PB).

Step9: set clock and send to specific port(PC).

Step10: reset the clock and send to specific port(PC).

Step11: Decrement bit count register, check, if it zero goto next step, if not goto step7.

Step 12: Decrement character count register, check, if it zero go to next count, if not goto Step4.

Step 13: Call delay program between each group

Step14: Decrement group counter , check, if it zero goto next step , if not gotogoto step 3

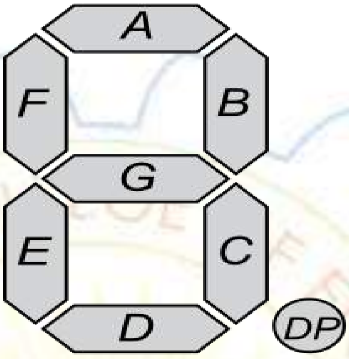
Step 15: go to step 1.

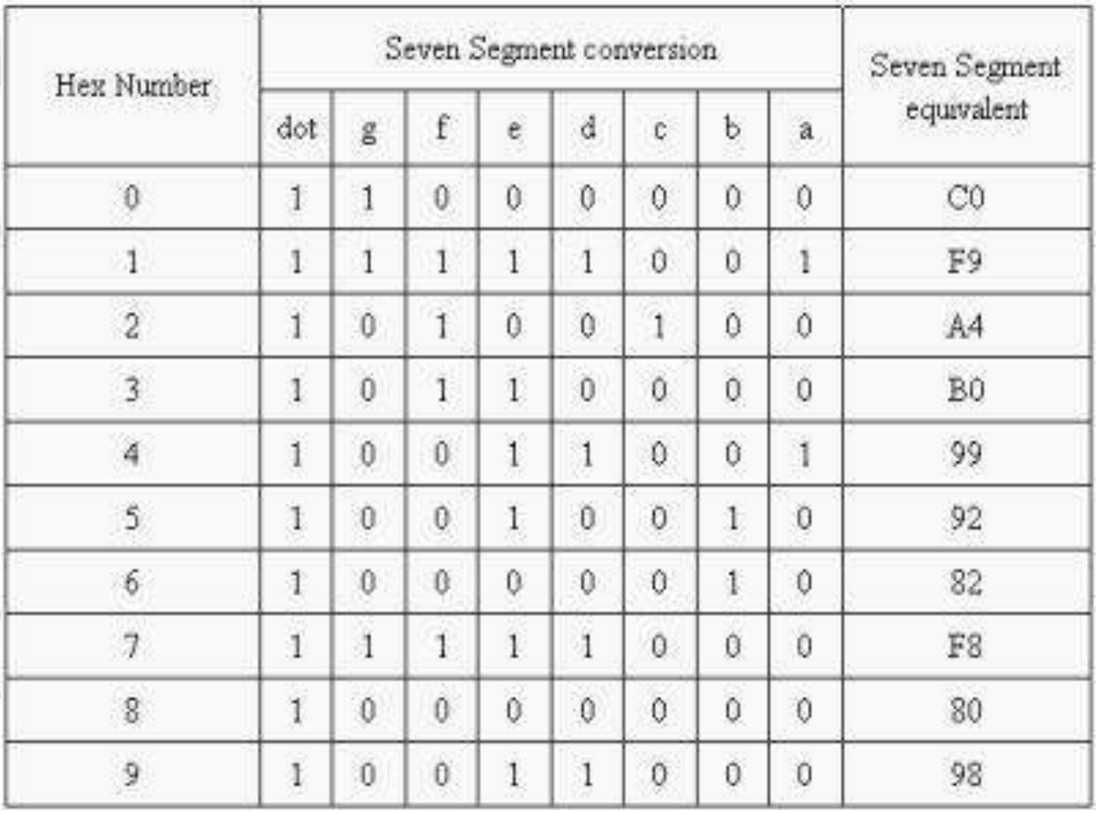
THEORY:

There are four digit 7 segment display driven by the outputs of four cascaded serial-in-parallel-out shift registers. Data to be displayer is transmitted serially, bit by bit, to the interface over the port line PB0. Each bit is clocked into the shift registers by providing a common clock through the port line PC0. Thus , information for all the four digits is provided by 32 bits clocked into the shift registers serially.

Display Codes: since the outputs of shift registers are connected to the cathode sides of LED segments, low input must be given to the segments for making them glow and high inputs for making them blank. Each display has 7 bar segments and a dot as in shown in figure below. For displaying any character its corresponding segments must be given blow inputs.

H





Hex Number	Seven Segment conversion								Seven Segment equivalent
	dot	g	f	e	d	c	b	a	
0	1	1	0	0	0	0	0	0	C0
1	1	1	1	1	1	0	0	1	F9
2	1	0	1	0	0	1	0	0	A4
3	1	0	1	1	0	0	0	0	B0
4	1	0	0	1	1	0	0	1	99
5	1	0	0	1	0	0	1	0	92
6	1	0	0	0	0	0	1	0	82
7	1	1	1	1	1	0	0	0	F8
8	1	0	0	0	0	0	0	0	80
9	1	0	0	1	1	0	0	0	98

ASSEMBLY LANGUAGE PROGRAM:-

```
        MOV     0A0,#0E8
        MOV     R0,#3
```

```
                MOV         A,#80
                MOVX        @R0,A
LOOP4:                  MOV         DPTR,#8050
                MOV         R3,#05
LOOP3:                  MOV         R1,#04
LOOP2:                  MOV         R2,#08
                MOVX        A,@DPTR
                INC         DPTR
LOOP1:                  RL          A
                MOV         R4,A
                MOV         R0,#1
                MOVX        @R0,A
                MOV         A,#01
                MOV         R0,#2
                MOVX        @R0,A
                DEC         A
                MOVX        @R0,A
                MOV         A,R4
                DEC         R2
                CJNE        R2,#0,LOOP1(8013)
                DEC         R1
                CJNE        R1,#0, LOOP2(800F)
                LCALL       8040(DELAY)
                DEC         R3
                CJNE        R3,#0,LOOP3(800D)
                SJMP        LOOP4(8008)
```

ADDRESS	OPCODE	MNEMONIC	OPERANDS	COMMENTS
8000	75,A0,E8	MOV	0A0,#0E8	Configure 8255 All ports output
8003	78,03	MOV	R0,#03	Control word to set all ports output
8005	74,80	MOV	A,#80	Load the control word in
8007	F2	MOVX	@RO,A	To control word register
8008	90,80,50	MOV	DPTR,#8050	Start of display code
800B	7B,05	MOV	R3,#5	5 groups to display
800D	79,04	MOV	R1,#4	4 Characters per group
800F	7A,08	MOV	R2,#8	8 Bits per character
8011	E0	,MOVX	A,@DPTR	Character get the display Code
8012	A3	INC	DPTR	Increment pointer for next Character.
8013	23	RL	A	Get one data bit
8014	FC	MOV	R4,A	Port B initialization
8015	78,01	MOV	R0,#1	Data bit output to port B
8017	F2	MOVX	@R0,A	Store temporarily the acc. In to AH
8018	74,01	MOV	A,#1	Output the clock
801A	78,02	MOV	R0,#2	Instillation the port c
801C	F2	MOVX	@R0,A	Output the clock through

				Port c
801D	14	DEC	A	To shift register
801E	F2	MOVX	@R0,A	Output the clock
801F	EC	MOV	A,R4	Load temporary stored data Into AL
8020	1A	DEC	R2	All bits are over?
8021	BA,00,EF	CJNE	R2,#0,8013	No continue
8024	19	DEC	R1	All characters over?
8025	B9,00,E7	CJNE	R1,#0,800F	No continue
8028	12,80,35	LCALL	8040	Introduce delay
802B	1B	DEC	R3	All groups are over.
802C	BB,00,DE	CJNE	R3,#0,800D	No to continue
802F	80,D7	SJMP	8008	Yes start from beginning
8040	7D,10	MOV	R5,#10	DELAY PROGRAM
8042	7F,0FF	MOV	R6,#0FF	
8044	7F,0FF	MOV	R7,#0FF	
8046	DF,FE	DJNZ	R7,8046	
8048	DE,FA	DJNZ	R6,8044	
804A	DD,F6	DJNZ	R5,8042	
804C	22	RET		

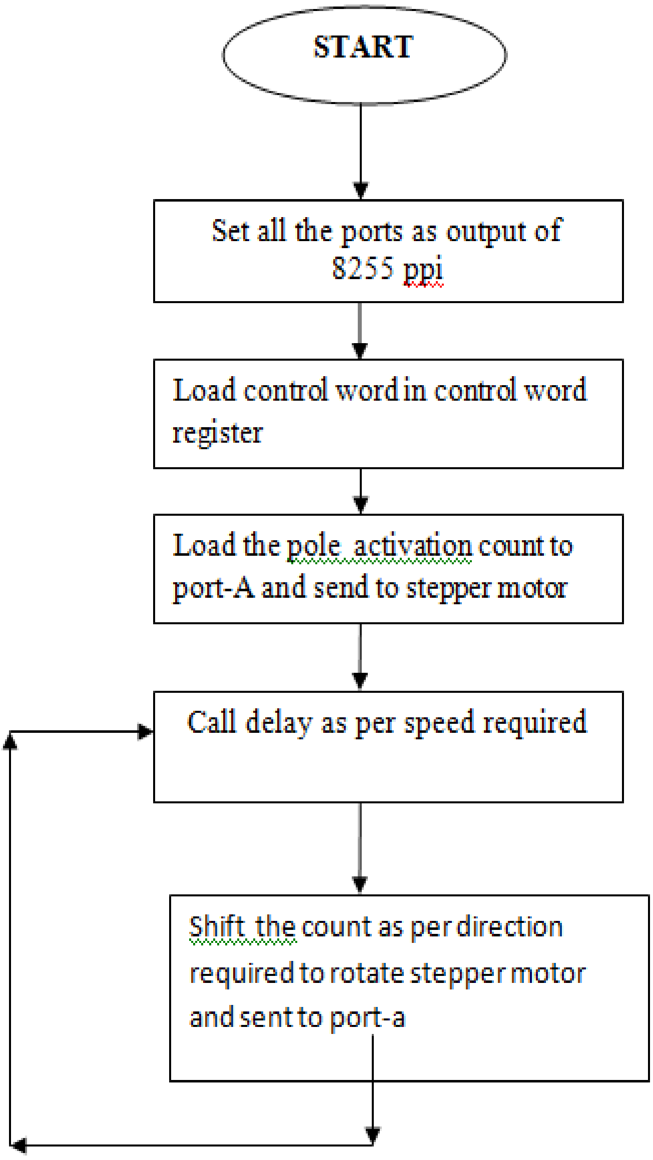
STRING

8050	0BF	0CC	0CC	0C6
8054	0BF	86	0C0	0C6
8058	0C6	86	92	88
805C	0BF	0CC	0C7	86
8060	0F8	0C0	0C0	0C0

RESULT:- The output is displayed as follows according to above code

4.STEPPER MOTOR

FLOW CHART


START
Set all the ports as output of 8255 ppi
Load control word in control word register
Load the pole activation count to port-A and send to stepper motor
Call delay as per speed required
Shift the count as per direction required to rotate stepper motor and sent to port-a

4.STEPPER MOTOR

DATE:-
EXP.NO:-

AIM:-

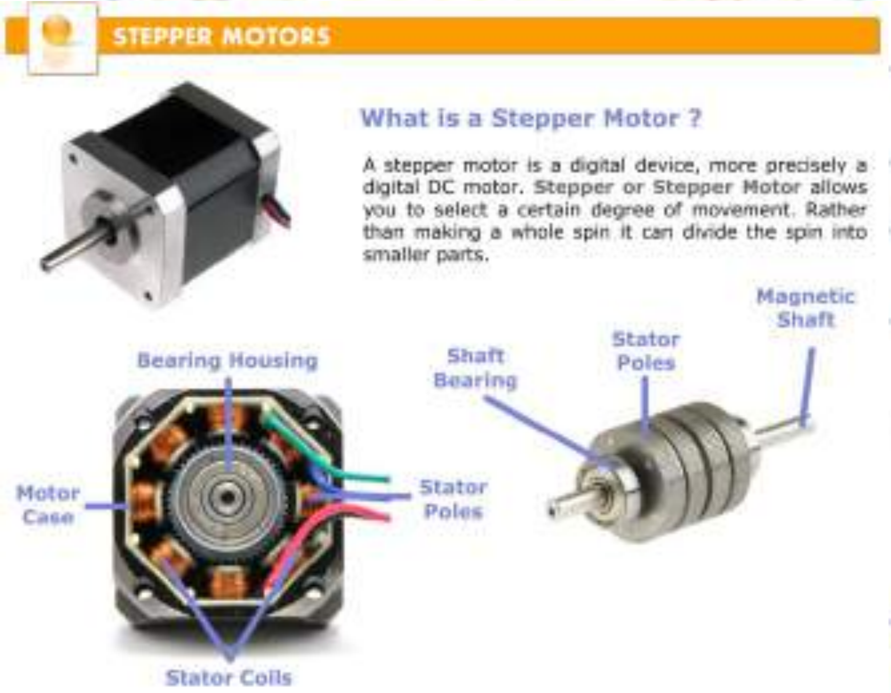
Write an Assembly Language Program to rotate the stepper motor using 8255 ppi in mode-0

APPARATUS :-

8051 KIT, Stepper Motor

THEORY:-

In this circuit the 8051controller is interfaced with 8255 in mode-0 and set all the ports are set to output. The output of port-A is connected to stepper motor And call the delay as per speed requirement. As per direction rotate stepper and shift the pole action with speed requirement. And repeat the same for continuously rotating the motor.



STEPPER MOTORS

What is a Stepper Motor ?

A stepper motor is a digital device, more precisely a digital DC motor. Stepper or Stepper Motor allows you to select a certain degree of movement. Rather than making a whole spin it can divide the spin into smaller parts.

ALGORITHM :-

Step1: Set all ports as output of 8255 in mode-0

Step2: Load control word into control word register.

Step3: Initialize port-a with pole activation count and output to port-A.

Step4: Call Delay according to speed.

Step5: rotate as per direction shift the poleaction count and output to port-A

Step6: Goto step 4

ASSEMBLY LANGUAGE PROGRAM:

```
               MOV    0A0,#0E8
               MOV   R0,#03
               MOV     A,#80H
               MOVX           @R0,A
               MOV   A,#88
          LOOP1:    MOV    R0,#00
          :    MOVX           @R0,A
               LCALL          8013H(DELAY)
               R R      A,1
               SJMP   LOOP1
```

ASSEMBLY LANGUAGE PROGRAM FOR SQUARE WAVE

DATA SEGMENT OFFSET : 076AH

ADDRESS IN HEXA	OP CODE IN HEXA	MNEMONICS	OPERAND	COMMENT
8000	85,E8,A0	MOV	0A0,#0E8	Set all ports of 8255 as output In mode -0
8003	78,03	MOV	R0,#03	
8005	74,80	MOV	A,#80	Load control word in to contro Word register
8007	F2	MOVX	@R0,A	
8008	74,88	MOV	A,#88	Out the initial word into port-A
800A	78,00	MOV	R0,#00	
800C	F2	MOVX	@R0,A	Send to display through DAC
800D	12,80,13	LCALL	8013	Call delay program for on/off Time
8010	03	RR	A	Complement for off time
8011	80,F9	SJMP	800A	Send to display for off time
8013	7B,0FF	MOV	R3,#0FF	DELAY PROGRAM
8015	7C,FF	MOV	R4,#0FF	
8017	DC,FE	DJNZ	R4,8017	
8019	DB,FA	DJNZ	R3,8015	
801B	22	RET		

RESULT :-

Rotating the stepper motor with different directions and with different speeds.