15EI302L-MICROCONTROLLER BASED SYSTEM DESIGN LABORATORY

MANUAL

Department of Electronics and Instrumentation Engineering

Faculty of Engineering and Technology
Department of Electronics and Instrumentation engineering
SRM Institute of Science & Technology
SRM Nagar , Kattankulathur – 603203,
Kancheepuram District
Tamil Nadu

CONTENTS

1. MARK ASSESSMENT DETAILS

ALLOTMENT OF MARKS:

Internal assessment	=	60 marks
Practical examination	=	40 marks
Total	=	100 marks

INTERNAL ASSESSMENT (60 MARKS)

Split up of internal marks

Record	5 marks
Model exam	10 marks
Quiz/Viva	5 marks
Experiments	40 marks
Total	60 marks

PRACTICAL EXAMINATION (40 MARKS)

Split up of practical examination marks

Aim and Procedure	25 marks
Circuit Diagram	30 marks
Tabulation	30 marks
Result	05 marks
Viva voce	10 marks
Total	100 marks

2. GENERAL INSTRUCTIONS FOR LABORATORY CLASSES

1. . Enter the Lab with CLOSED TOE SHOES.
2. Students should wear lab coat.
3. The HAIR should be protected, let it not be loose.
4. Students should come with observation and record note book to the laboratory.
5. Students should maintain silence inside the laboratory.
6. TOOLS, APPARATUS and COMPONENT sets are to be returned before leaving the lab.
7. HEADINGS and DETAILS should be neatly written
    i. Aim of the experiment
    ii. Apparatus / Tools / Instruments required
    iii. Theory
    iv. Procedure / Algorithm / Program
    v. Model Calculations/ Design calculations
    vi. Block Diagram / Flow charts/ Circuit diagram
    vii. Tabulations/ Waveforms/ Graph
    viii. Result / discussions
8. Experiment number and date should be written in the appropriate place.
9. After completing the experiment, the answer to pre lab viva-voce questions should be neatly written in the workbook.
10. Be REGULAR, SYSTEMATIC, PATIENT, AND STEADY

15EI302L	Microcontroller based System Design Laboratory			L	T	P	C
				0	0	2	1
Co-requisite:	15EI302						
Prerequisite:	NIL						
Data Book / Codes/Standards	NIL						
Course Category	P	PROFESSIONAL CORE	ELECTRONICS ENGINEERING				
Course designed by	Department of Electronics and Instrumentation Engineering						
Approval	32nd Academic Council Meeting held on 23rd July, 2016						

PURPOSE	To develop skills in programming and interfacing applications of microprocessors and microcontrollers.

INSTRUCTIONAL OBJECTIVES		**STUDENT OUTCOMES**						
At the end of the course, student will be able to								
1.	Improve their ability in their programming skills	b						
2.	Equip themselves familiar with interfacing concepts of microprocessors	b	c	d				
3.	Equip themselves familiar with interfacing concepts of microcontrollers	b	c	d				

Session	Description of Topic	Conduct hours	C-D-I-O	IOs	Reference
	General Purpose Programming Exercises Using 8086				
1	Addition, Subtraction, Multiplication and Division	3	C,I	1,2	1,2
2	Finding the maximum value in an array.	2	C,I	1,2	1,2
3	Sorting of data.	1	C,I	1,2	1,2
4	BCD-to-Hex conversion and Hex-to-BCD conversion.	3	C,I	1,2	1,2
5	Block data transfer (forward and reverse)	3	C,I	1,2	1,2
	Interfacing with Application Boards				
6	Traffic light control using 8051	3	C,D,I	1,3	1,3

Session	Description of Topic	Conduct hours	C-D-I-O	IOs	Reference
7	Stepper motor control using 8051 controller	3	C,D,I	1,3	1,3
8	Temperature control system using 8051	3	C,D,I	1,3	1,3
9	LCD Display using 8051/ Nu-LB-NUC140 controller	3			
10	8 bit ADC and 8 bit DAC. using nuvoTon (NUC140) board	3			
11	Seven segment display using nuvoTon (NUC140) board				
		30			

LEARNING RESOURCES	
Sl. No.	**REFERENCES**
1.	*Laboratory Manual*
2.	*N. Senthil Kumar, M. Saravanan and S. Jeevananthan, "Microprocessors and Microcontrollers", Oxford Publishers,2010..*
3.	*nuvoTon Cortex M0 (Nu-LB-NUC100/140) Driver and Processor Reference Manual; www.nuvoton.com*

Course nature				**Practical**			
Assessment Method (Weightage 100%)							
In-semester	**Assessment tool**	Experiments	Record	MCQ/Quiz/Viva Voce	Model examination	**Total**	Experiments
	Weightage	**40%**	**5%**	**5%**	**10%**	**60%**	**40%**
					End semester examination Weightage :		**40%**

4. Introduvction to the laboratory

AIM:

To study about the basic architecture of 8086 microprocessor.

APPARATUS:

8086 micro processor kit
Keyboard

THEORY:

A microprocessor is a multipurpose, programmed clock driven register based system which takes the input in binary process (arithmetic, logical) and gives the output.
The 8086 is the first 16-bit microprocessor released by Intel which can execute 2.5 million instructions per second. It has a 20-bit address bus.
The main feature of 8086 which makes it better than 8085 is the presence of a six-byte instruction queue in which the instructions fetched from the memory are placed before they are executed.

Architecture of 8086

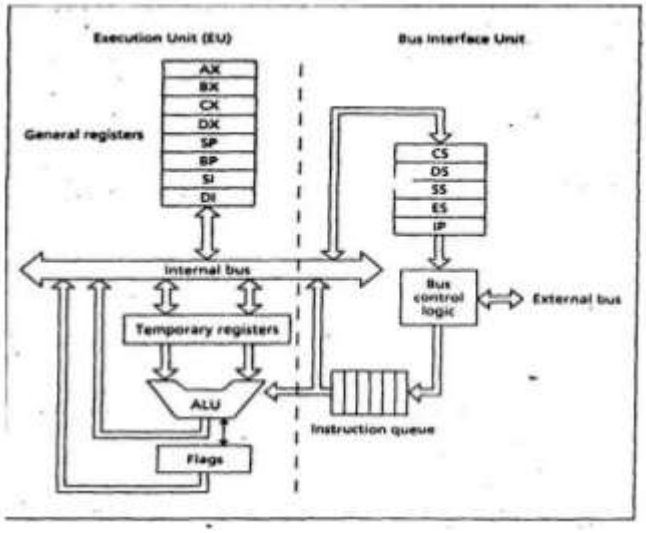


Image - From Microprocessors and Microcontrollers by N.Senthil Kumar

1. Execution Unit – It includes the ALU (Arithmetic Logical Unit), eight 16-bit general-

purpose registers, 16-bit flag register and a control unit.

Register Organisation ->
8086 consists of 2 types of register: general purpose and special purpose registers.
General purpose register is used for holding data, variables and intermediate results temporarily and can also be used as counters.
The Execution Unit consists of eight 16-bit general purpose registers – AX, BX, CX, DX, SP, BP, SI and DI. Among these registers, AX, BX, CX, DX can be divided into two 8-bit registers – AH and AL, BH and BL, CH and CL and DH and DL. The general purpose registers can be used to store 8-bit or 16-bit data during program execution.

General Purpose Registers --

(i) AX/AL: It is used as the accumulator with the lower 8-bit stored in AL and the higher 8-bits stored in AH. It is used in the multiply, divide and input/output operations.
(ii) BX: The BX register holds the offset address of a location in the memory. It is also used to refer to the data in the memory using look-up table technique using XLAT instruction.
(iii) CX/CL: It is used as default counter in string and loop instructions
(iv) DX: It is used to hold a part of the result during a multiplication operation and a part of the dividend before a division operation.

Pointers and Index Registers –
The index registers are used as general purpose registers as well as for offset storage.

(i) SP: The stack pointer is used to hold the offset address of the data stored at the top of the stack segment. It is used with the SS to decide the address at which the data is to be pushed or popped.
(ii) BP: It is also called base pointer. It is also used to hold the offset address of the data to be read from or written into stack segment.
(iii) SI: It is also called as source index register. It is used to hold the offset address of the source data in the data segment, while executing string instructions.
(iv) DI: It is also called as destination index. It is used to hold the offset address of the destination data in the extra segment, while executing string instructions.

Flag Registers –
The flags in the flag register can be classified into status flags and control flags. The flags CF, PF, AF, ZF, SF and OF are called status flags, as they indicate the status of the result that is obtained after the execution of an arithmetic or logic instruction. The flags DF, IF, and TF are called control flags, as they control the operation of the CPU.

(i) CF: The carry flag holds the carry after an 8-bit or 16-bit addition or the borrow after an 8-bit or 16-but subtraction operation.
(ii) PF: If the lower eight bits of the result have an odd parity (i.e., odd number of 1s), parity flag is set is to 0. Otherwise, it is set to 1.
(iii) AF: The auxiliary flag holds the carry after addition or the borrow after subtraction of the bits in the bit position 3.
(iv) ZF: The zero flag indicates that the result of an arithmetic or logic operation is

zero. If z=1, the result is zero and if z=0, the result is not zero.

(v) SF: Sign flag holds the arithmetic sign of the result after an arithmetic or logical instruction is executed.

(vi) TF: Trap flag is used to debug a program using the single-step technique. If TF=1, the 8086 gets interrupted after the execution of each instruction in the program.

(vii) DF: Direction flag selects either the increment or decrement mode for the DI and/or SI, during the execution of string instructions.

(viii) IF: Interrupt flag controls the operation of the INTR interrupt pin of the 8086.

(ix) OF: An overflow flag indicates that the result has exceeded the capacity of the machine.

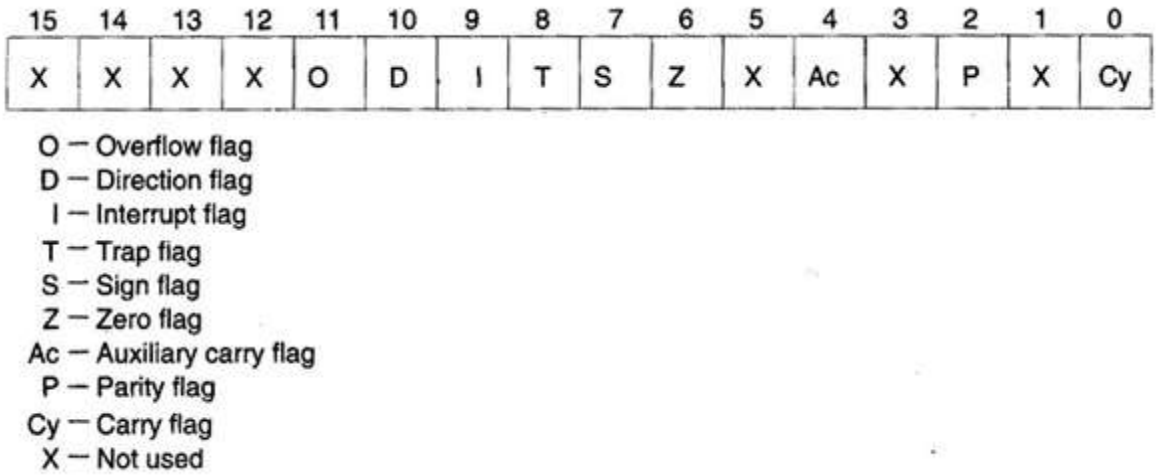


Image – Flag register in a 8086 microprocessor

2. Bus Interface Unit – It includes the adder for address calculations, four 16-bit registers (CS, DS, SS, ES) and a 16-bit instruction pointer, a six-byte instruction queue and bus control logic.

   Special purpose registers are used as segment registers, pointers, index registers or as offset storage registers for particular address modes.
   The memory consists of 4 types of registers,
   Code segment registers (CS), data segment register (DS), stack segment registers (SS) and extra segment registers (ES). CS stores the executable program, DS stores the data. The SS holds the stack of the program, which is needed while executing the CALL and RET instructions and also to handle interrupts.
   The CPU uses the stack for temporarily storing the important data. While addressing any memory location, the physical address is detected from 2 parts, the first is segment address and the second is offset address.

<u>Pin Diagram and Explanation of 8086</u>

(i) AD15-AD0: These pins act as the multiplexed address and data bus of the microprocessor. Whenever the ALE (address latch enable) pin is HIGH, these pins

carry the address, and whenever it is LOW, these pins carry data.

(ii) A19/S6-A16/S3: These pins are multiplexed to provide the address signals A19-A16 and the status bits S6-S3. When ALE =1, these pins carry the address and ALE=0, they carry the status lines.

(iii) NMI: The non-maskable interrupt input is a hardware interrupt. It cannot be disabled by software. It is a positive edge-triggered interrupt and when it occurs, the type 2 interrupt occurs in the 8086.

(iv) INTR: The interrupt request is a level triggered hardware interrupt, which depends on the status of IF. When IF=1, INTR is held HIGH, the 8086 gets interrupted.

(v) CLK: The clock signal must have a duty cycle of 33% to provide a proper internal timing for the 8086.

(vi) $\overline{\boldsymbol{BHE}}$/S7: The Bus High Enable pin is used in the 8086 to enable the most significant data bus during a read/write operation. The state of the status line S7 is always logic 1.

(vii) MN/$\overline{\boldsymbol{MX}}$: This pin is used to select either the minimum mode or the maximum mode operation in the 8086.

(viii) $\overline{\boldsymbol{RD}}$: Whenever the read signal is at logic 0, the 8086 reads the data from the memory or I/O device through the data bus.

(ix) READY: This input is used to insert waits states into the timing cycle of the 8086. If the READY pin is at logic 1, it has no effect on the operation of the microprocessor. If it is at logic 0, the 8086 enters the wait state and remains idle.

(x) M/$\overline{\boldsymbol{IO}}$: This pin indicates whether the 8086 is performing memory read/write operation (M/$\overline{\boldsymbol{IO}} = \mathbf{1}$) or I/O read/write operation (M/$\overline{\boldsymbol{IO}} = \mathbf{0}$).

(xi) HOLD: The Hold input requests a direct memory access and is generated by the DMA controller. If the Hold signal is at logic 1, the 8086 completes the execution of the current execution and places its address, data and control buses in the high impedance state. If the Hold signal is at logic 0, the 8086 executes instructions normally.

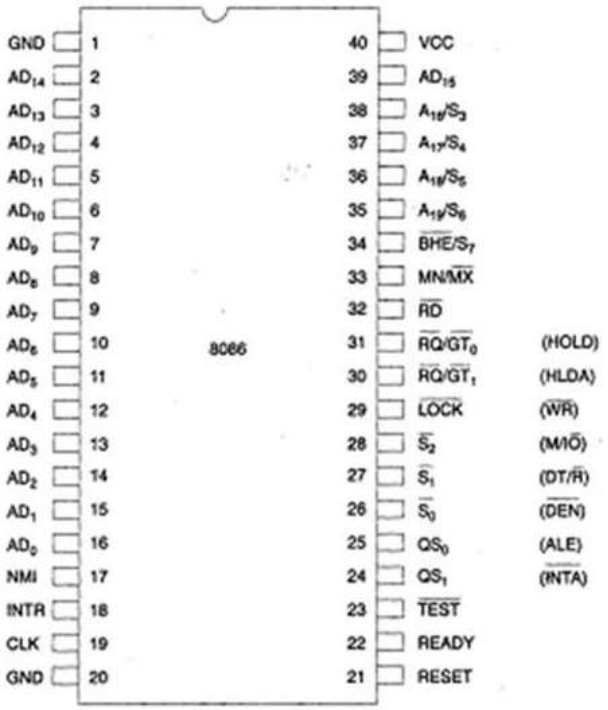


Image – Pin diagram of 8086 microprocessor

RESULT:

The assembly language program for 8 bit addition of two numbers was executed successfully by using 8085 micro processing kit.

Prepared by
Dr. K. A. Sunitha

Exercise Number 1a

Title of the Experiment: ADDITION OF TWO 8-BIT NUMBERS
Date of the Exercise:

AIM:

(a) To write and execute an assemble language program to add two 8-bit data.
(b) To write and execute an assemble language program to subtract two 8-bit data.

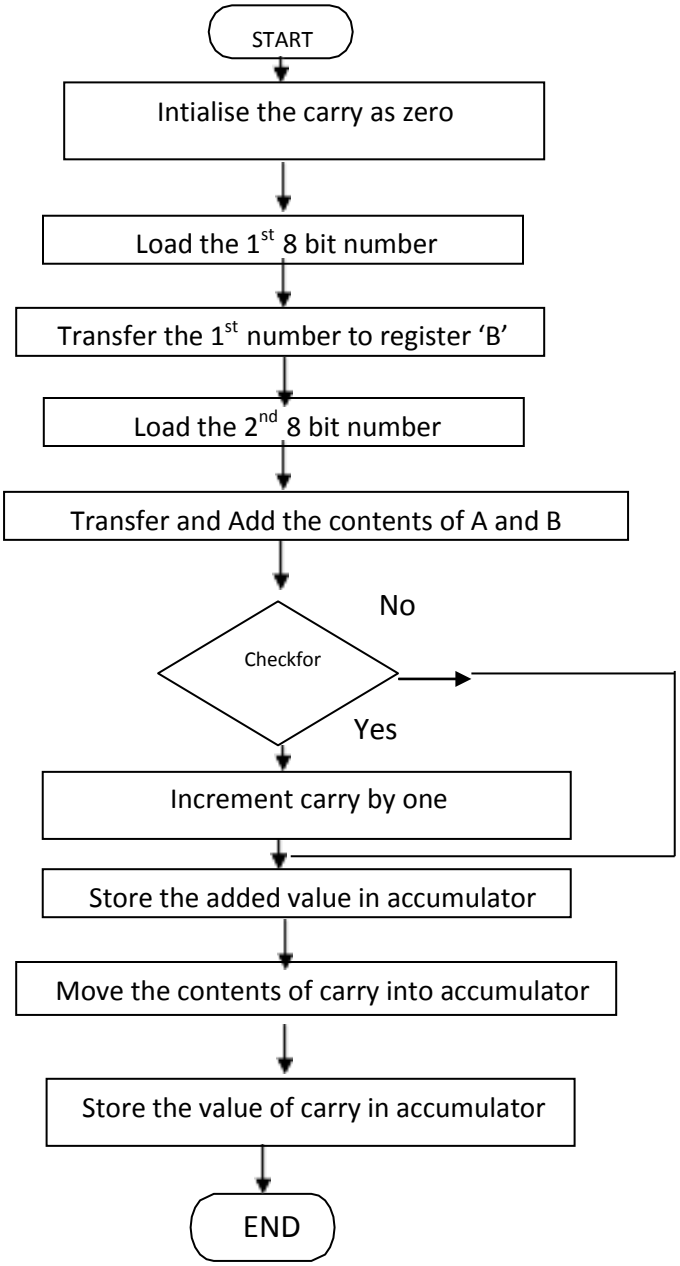
APPARATUS REQUIRED:

1. Microprocessor 8086 kit.
2. Keyboard

ALGORITHM:

Step 1	:	Start the microprocessor
Step 2	:	Initialize the carry as‘Zero’
Step 3	:	Load the first 8 bit data into the accumulator
Step 4	:	Copy the contents of accumulator into the register ‘B’
Step 5	:	Load the second 8 bit data into the accumulator.
Step 6	:	Add the 2 - 8 bit data and check for carry.
Step 7	:	Jump on if no carry
Step 8	:	Increment carry if there is
Step 9	:	Store the added requesting accumulator
Step 10	:	More the carry value to accumulator
Step 11	:	Store the carry value in accumulator
Step 12	:	Stop the program execution.

FLOW CHART


START
Intialise the carry as zero
Load the 1st 8 bit number
Transfer the 1st number to register 'B'
Load the 2nd 8 bit number
Transfer and Add the contents of A and B
Checkfor
No
Yes
Increment carry by one
Store the added value in accumulator
Move the contents of carry into accumulator
Store the value of carry in accumulator
END

PROGRAM:

Address	Mnemonics	Comments
4100	MOV AL,01	Initialize AL
4102	MOV BL, 02	Initialise BL
4105	MOV CL, 00	Initialise CL for carry
4106	ADD AL,BL	Add AX and BX
4109	JNC100C	Checks for Carry at 100C
410A	INC CL	Increment CL
410D	MOV[1300],AL	Move AL to address
410E	MOV AL,CL	Move CL to AL
4111	MOV[1301],AL	Move Al to 1301 location
4112	INT3	End program.

Input Without carry

Input Address	Value
4300	04
4301	02

Output

Output Address	Value
4302	06
4303	00 (carry)

With carry

Input Address	Value
4300	FF
4301	FF

Output Address	Value
4302	FE
4303	01 (carry)

Calculation 1111 1111
1111 1111
---
(1) 1111 1110
========
= F E

RESULT:

The assembly language program for 8 bit addition of two numbers was executed successfully by using 8085 micro processing kit.

Prepared by
Dr. K. A. Sunitha

Exercise Number 1b

Title of the Experiment: SUBTRACTION OF TWO 8-BIT NUMBERS
Date of the Exercise:

AIM:

To write an assembly language for Subtracting two 8 bit numbers by using microprocessor kit.

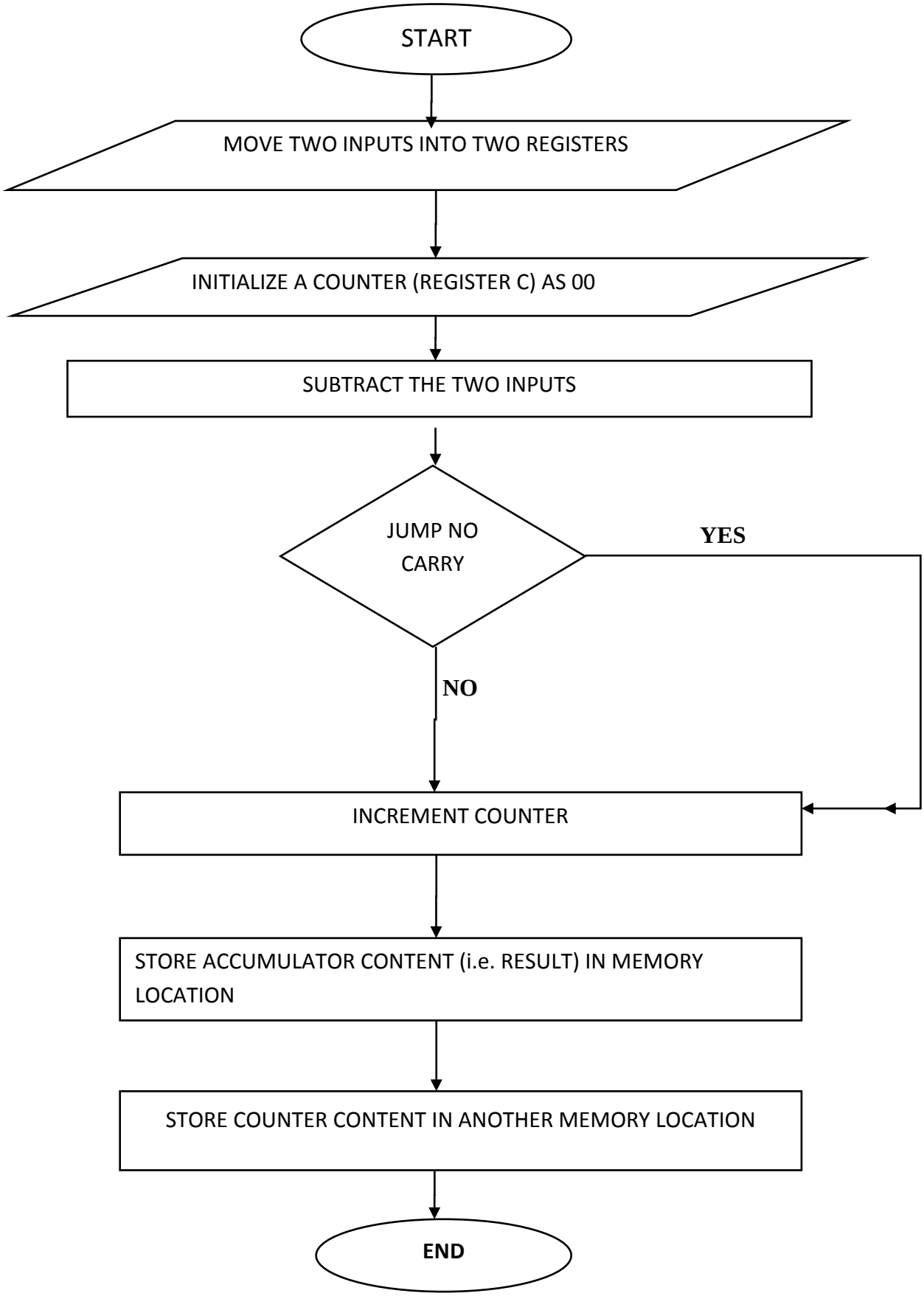
APPARATUS:

8086 microprocessor
(05V) DC battery

ALGORITHM:

1. Load the first data from memory to accumulator.
2. Move the accumulator content to B register.
3. Load the second data from memory to accumulator.
4. Clear C register.
5. Subtract the content of B register to accumulator.
6. Check carry flag if carry is set go to step7 else to step8.
7. Increment C register.
8. Store the answer in memory.
9. Move the C register content to accumulator and store it in memory.
10. Stop.

FLOW CHART


START
MOVE TWO INPUTS INTO TWO REGISTERS
INITIALIZE A COUNTER (REGISTER C) AS 00
SUBTRACT THE TWO INPUTS
JUMP NO CARRY
YES
NO
INCREMENT COUNTER
STORE ACCUMULATOR CONTENT (i.e. RESULT) IN MEMORY LOCATION
STORE COUNTER CONTENT IN ANOTHER MEMORY LOCATION
END

TABULATION:

ADDRESS	MNEMONICS	COMMENT
8000	MOV AX, 0080	;MOVE 80 TO REGISTER AX
8001		
8002	MOV BX, 0080	;MOVE 80 TO REGISTER BX
8003		
8004	MOV CL,00	;MOVE 00 TO REGISTER CX
8005		
8006	SUB AX,BX	SUBTRACT BX FROM AX
8007	JNC100C	;JUMP NO CARRY TO 100C
8008		
8009		
800A	INR CL	;INCREMENT REGISTER CL
800B	MOV[1500],AL	MOVE AX TO [1500]
800C		
800D		
800E	MOVAX,CX	;MOVE CONTENTS OF CX TO AX
800F	MOV[1501],AL	MOVE AX VALUE TO 1501
8010		
8011		
8012	INT3	TERMINATE

RESULT:

SUBTRACTION OF TWO 8-BIT DATA			
INPUT		OUTPUT	
MEMORY LOCATION	CONTENT	MEMORY LOCATION	CONTENT
8001	08	1500	00
8003	09	1501	01

Thus subtraction of two 8-bit numbers was done and results were verified.

Prepared by
Dr. K. A. Sunitha

Exercise Number: 1c

Title of the Experiment: MULTIPLICATION OF TWO 8-BIT NUMBERS
Date of the Exercise:

AIM:

To write and execute an assemble language program to multiply two 8-bit data.

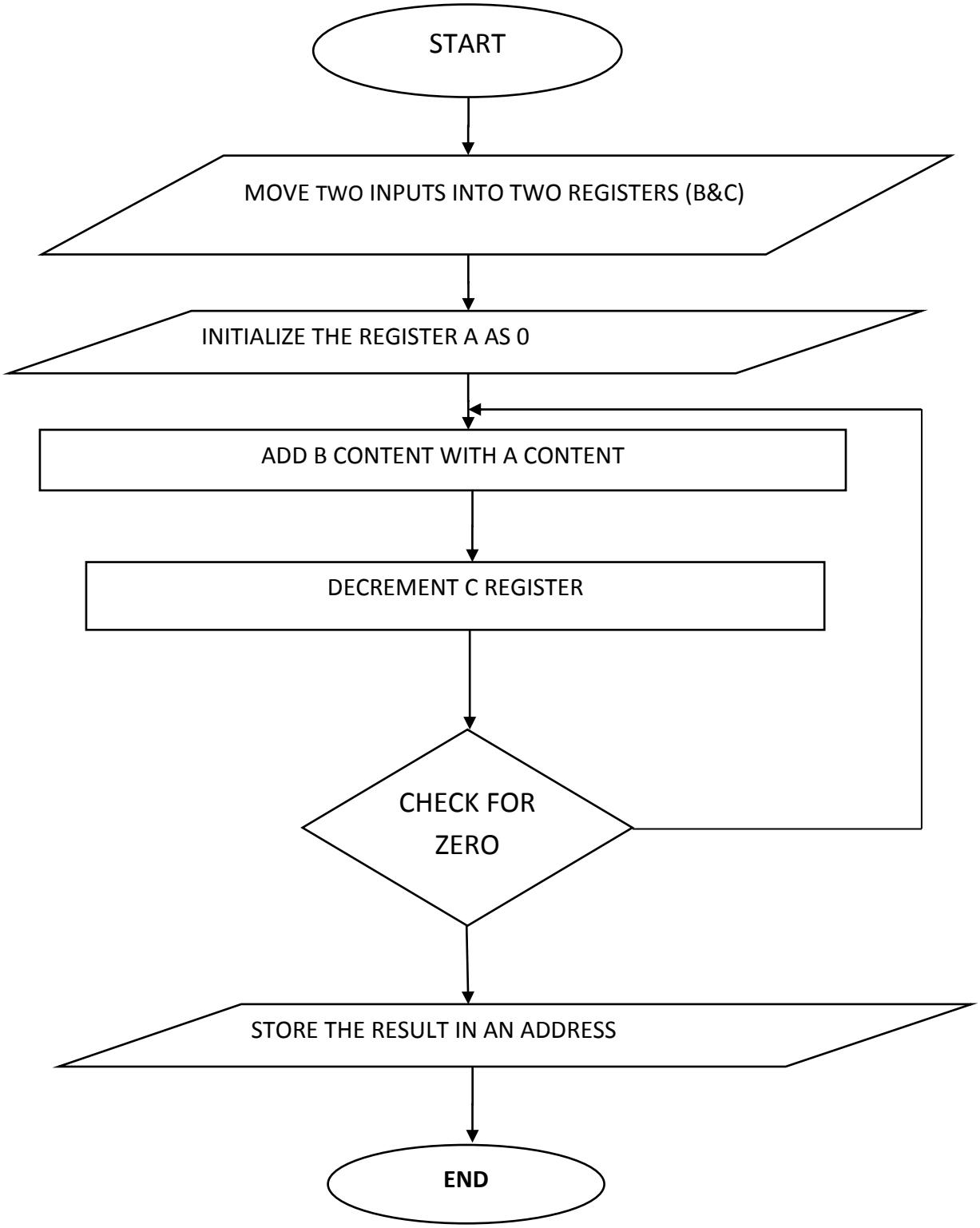
APPARATUS REQUIRED:

1. Microprocessor 8085 kit.
2. Manual.

ALGORITHM:

1. Load the first data from memory to B register.
2. Load the second data from memory to C register.
3. Initialize A register with zero.
4. Add the content of B register to accumulator.
6. Decrement C register.
7. Check zero flag if zero is set go to step8 else to step5.
8. Store the sum in memory.
9. End the program and execute it to display the result.

FLOW CHART


START
MOVE TWO INPUTS INTO TWO REGISTERS (B&C)
INITIALIZE THE REGISTER A AS 0
ADD B CONTENT WITH A CONTENT
DECREMENT C REGISTER
CHECK FOR ZERO
STORE THE RESULT IN AN ADDRESS
END

TABULATION:

ADDRESS	MNEMONICS	COMMENT
8000	MOV CL,00	CLEAR CL REGISTER FOR CARRY
8001		
8002	MOV AL,01	MOV
8003		
8004	MVI A,00	;MOVE 00 TO REGISTER C
8005		
8006	ADD B	;ADD (A)+(B) AND STORE IN A
8007	DCR C	;DECREMENT REGITER C
8008	JNZ	;JUMP NO ZERO TO 8006
8009	06	
800A	80	;INCREMENT REGISTER C
800B	STA 8200	;STORE (A) IN 8200
800C	00	
800D	82	
800E	HLT	;HALT

RESULT:

ADDITION OF TWO 8-BIT DATA			
INPUT		OUTPUT	
MEMORY LOCATION	CONTENT	MEMORY LOCATION	CONTENT
8001	03	8200	06
8003	02		

Thus multiplication of two 8-bit numbers was done and result was verified.

Prepared by
Dr. K. A. Sunitha

Exercise Number: 1d

Title of the Experiment:DIVISION OF TWO 8-BIT NUMBERS
Date of the Exercise:

AIM: To write and execute an assemble language program to divide two 8-bit data.

APPARATUS REQUIRED:

1. Microprocessor 8085 kit.
2. Manual.
3. Op code sheet.

ALGORITHM:

1. Load the first data from memory to accumulator.

2. Move the accumulator content to B register.

3. Load the second data from memory to accumulator.

4. Initialize C register with zero.

5. Compare the numbers to check for carry.

6. Subtract the two numbers

7. Increment C register if there is a carry.

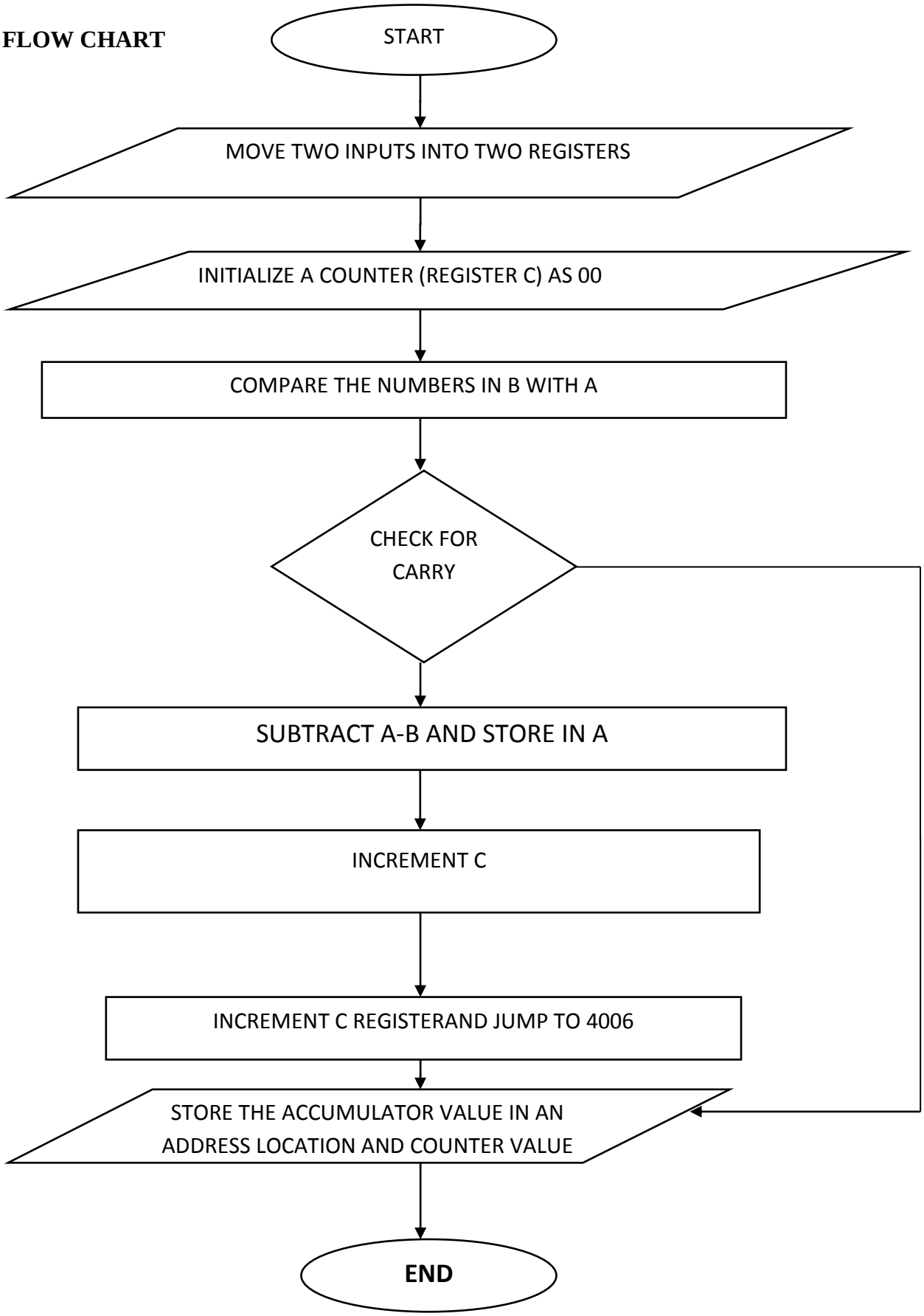
8. Check whether repeated subtraction is over.

9. Store the difference in memory.

10. Move the C register content to accumulator and store it in memory.

11. Stop.

FLOW CHART


START
MOVE TWO INPUTS INTO TWO REGISTERS
INITIALIZE A COUNTER (REGISTER C) AS 00
COMPARE THE NUMBERS IN B WITH A
CHECK FOR CARRY
SUBTRACT A-B AND STORE IN A
INCREMENT C
INCREMENT C REGISTERAND JUMP TO 4006
STORE THE ACCUMULATOR VALUE IN AN ADDRESS LOCATION AND COUNTER VALUE
END

TABULATION:

ADDRESS	LABEL	MNEMONICS	OP-CODE	COMMENT
4000		MVI A,04	3E	MOVE 04 TO ACCUMULATOR
4001			04	
4002		MVI B,02	06	MOVE 02 TO B REGISTER
4003			02	
4004		MVI C,00	0E	MOVE 00 TO REGISTER C
4005			00	
4006		CMP B	B8	COMPARE THE DATA IN B WITH A
4007		JC 400F	DA	JUMP TO 400F IF CARRY IS PRESENT
4008			0F	
4009			40	
400A		SUB B	90	SUBTRACT A-B
400B		INR C	0C	INCREMENT C REGISTER
400C		JMP4006	C3	JMP TO 4006
400D			06	
400E			40	
400F		STA 4100	32	STORE THE DIFFERENCE IN 4100
4010			00	
4011			41	
4012		MOV A,C	79	MOVE DATA IN C TOA
4013		STA 4101	32	STORE DATA IN A TO 4101
4014			01	
4015			41	
4016		HLT	76	STOP

RESULT:

DIVISION OF TWO 8-BIT DATA			
INPUT		OUTPUT	
MEMORY LOCATION	CONTENT	MEMORY LOCATION	CONTENT
4001	04	4100	00
4003	02	4101	02

Thus division of two 8-bit numbers was done and results were verified.

Prepared by
Dr. K. A. Sunitha

Exercise Number 2

Title of the Experiment: FINDING THE MAXIMUM VALUE OF AN ARRAY
Date of the Exercise:

AIM: To sort the numbers in 8086 and obtain

a) smallest number
b) largest number

APPARATUS REQUIRED:

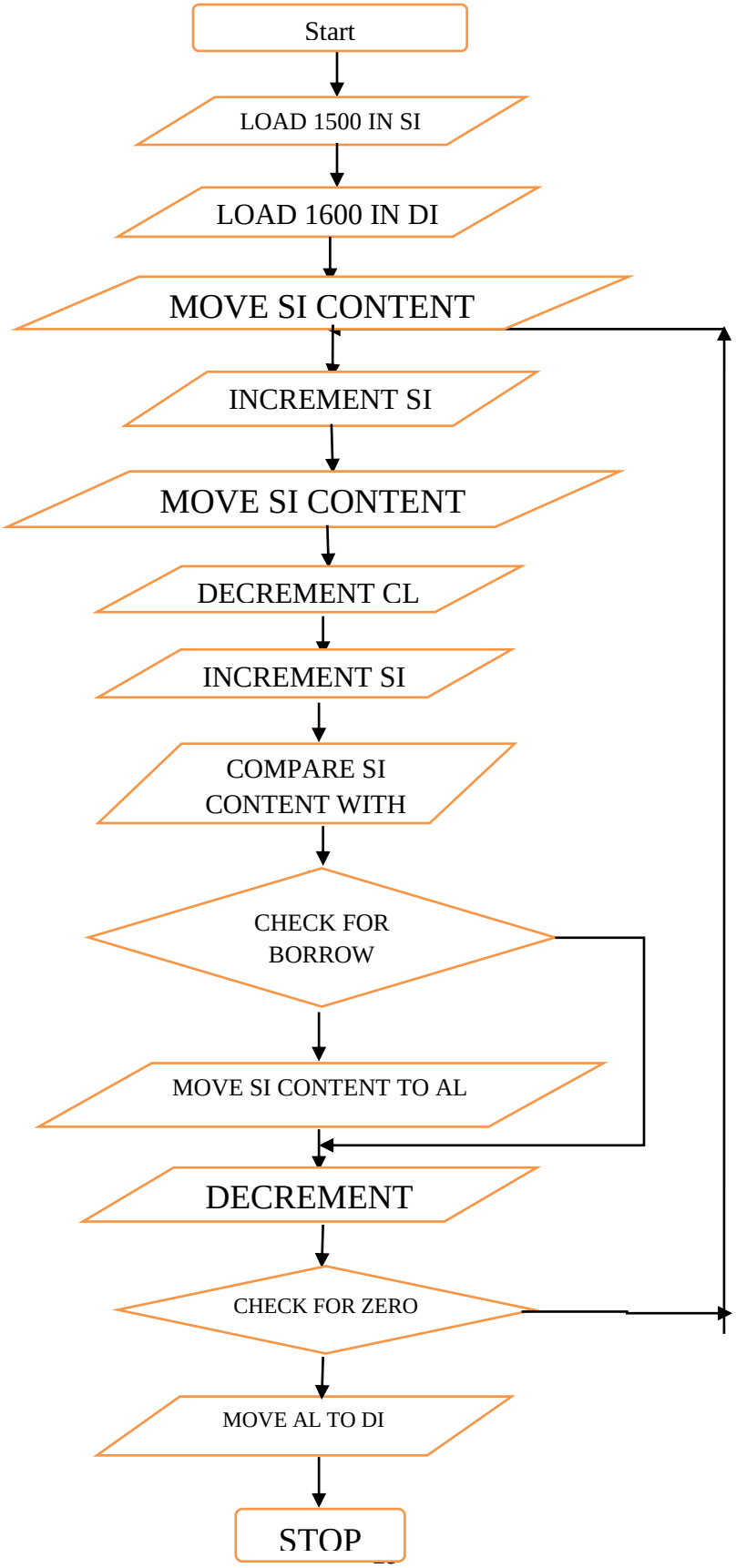
1. 8086 Trainer Kit
2. Keyboard

A) Smallest Number:-

ALGORITHM:

1. Set the SI and DI values.
2. Set CL as counter.
3. Increment SI.
4. Move content to AL.
5. Decrement CL.
6. Increment SI.
7. Compare the content with that of AL.
8. If no borrow, **JUMP**.
9. If content present, move content to AL.
10. Decrement CL.
11. Jump if no zero.
12. On obtaining largest number, move to DI.
13. Stop the program.

FLOW CHART


Start
LOAD 1500 IN SI
LOAD 1600 IN DI
MOVE SI CONTENT
INCREMENT SI
MOVE SI CONTENT
DECREMENT CL
INCREMENT SI
COMPARE SI CONTENT WITH
CHECK FOR BORROW
MOVE SI CONTENT TO AL
DECREMENT
CHECK FOR ZERO
MOVE AL TO DI
STOP

;

PROGRAM:

Address	Label	Mnemonics	Comments
1200		MOV SI, 1580	SI becomes 1508
1204		MOV DI, 1680	DI becomes 1608
1207		MOV CL, [SI]	Move [1508] to CL
1209		INC SI	Increment SI
120A		MOV AL, [SI]	Move [1509] to AL
120C		DEC CL	Decrement CL
120D	Loop 2	INC SI	Increment SI
120E		CMP AL, [SI]	Compare
1210		JNB	Jump if no borrows
1213		MOV AL, [SI]	Move [150A] to AL
1215	Loop 1	DEC CL	Decrement CL
1216		JNZ	Jump if no zero
1219		MOV[DI], AL	Move AL to [DI]
121B		INT 3	Stop

RESULT:

Input		Output	
Address	**Input**	**Address**	**Output**
1580	05	1600	09
1581	01		
1583	03		
1584	08		
1585	09		

Thus a set of numbers were sorted and the largest number was obtained.

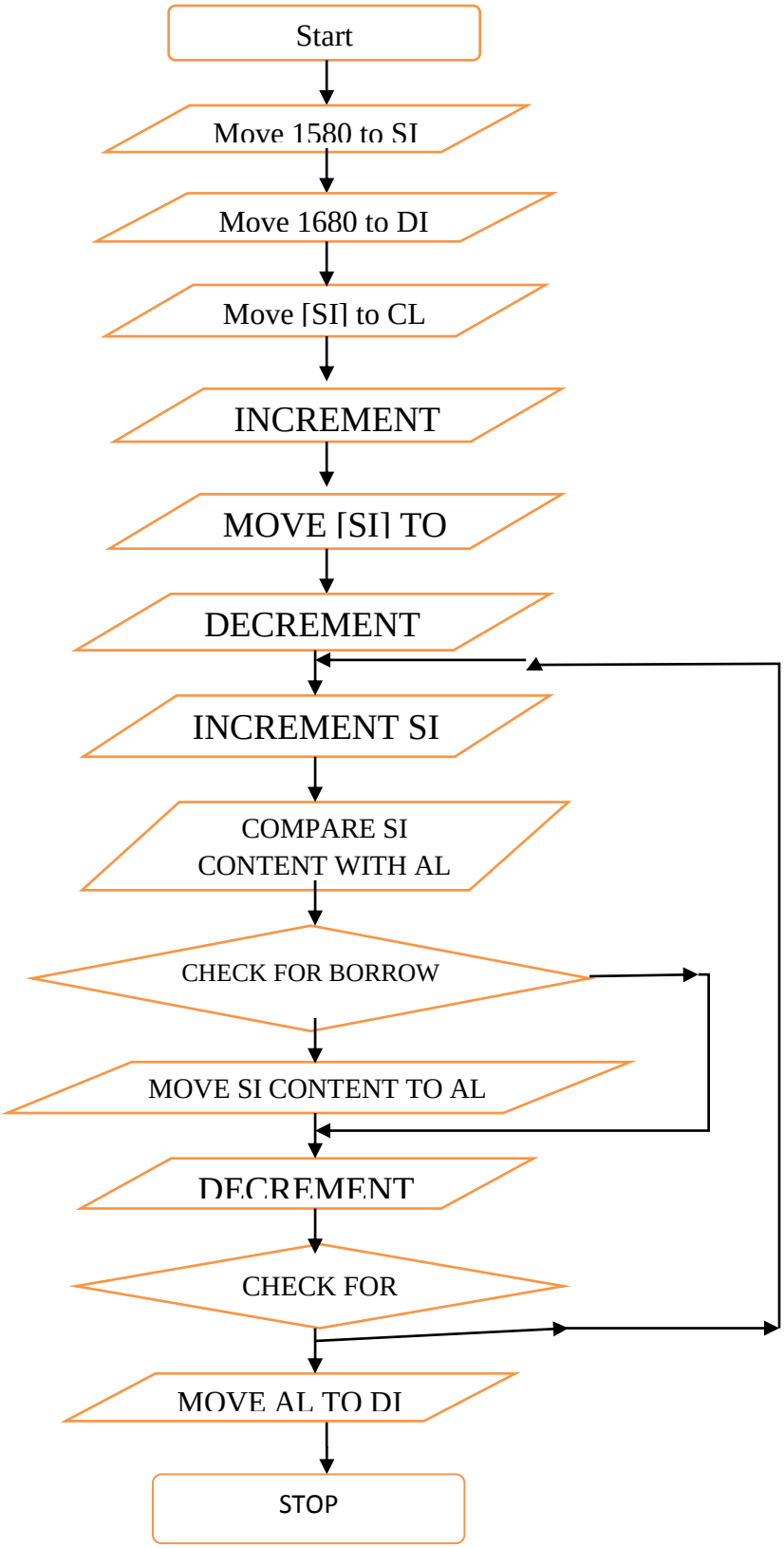
Prepared by
Vibha.k

B) Largest Number:-

ALGORITHM:

1. Load SI with 1500 and DI with 1600.
2. Move SI content to CL and increment it.
3. Move SI content to AL.
4. Decrement CL.
5. Increment SI.
6. Compare the content of SI with that in AL.
7. Jump if borrow, to 1016.
8. Move the SI content in AL.
9. Decrement CL.
10. Jump if no zero.
11. Move the AL content in the address of DI.
12. Stop the program.

FLOW CHART


Start
Move 1580 to SI
Move 1680 to DI
Move [SI] to CL
INCREMENT
MOVE [SI] TO
DECREMENT
INCREMENT SI
COMPARE SI
CONTENT WITH AL
CHECK FOR BORROW
MOVE SI CONTENT TO AL
DECREMENT
CHECK FOR
MOVE AL TO DI
STOP

PROGRAM:

Address	Label	Mnemonics	Comments
1000		MOV SI, 1500	Load SI with 1500
1003		MOV DI,1600	Load DI with 1600
1006		MOV CL, [SI]	Move SI content in CL.
1008		INC SI	Increment SI
100A		MOV AL, [SI]	Move SI content in AL
100C		DEC CL	Decrement CL
100D	Loop 2	INC SI	Increment SI
100F		CMP AL, [SI]	**Compare SI content with AL**
1011		JB 1016 (loop 1)	Jump if no borrows
1014		MOV AL, [SI]	Move SI content in AL
1016	Loop 1	DEC CL	Decrement CL
1017		JNZ 100D(loop 2)	Jump if no zero
101A		MOV[DI], AL	Move AL in DI
101C		INT 3	Stop

RESULT:

Input		Output	
Address	**Input**	**Address**	**Output**
1500	05	1600	01
1501	01		
1502	02		
1503	03		
1504	04		
1505	08		

Thus a set of numbers were sorted and the smallest number was obtained.

Prepared by
Vibha.k

Exercise Number: 3

Title of the Experiment: SORTING OF DATA
Date of the Exercise:

AIM:

To arrange the given set of numbers in ascending and descending order.

APPARATUS REQUIRED:

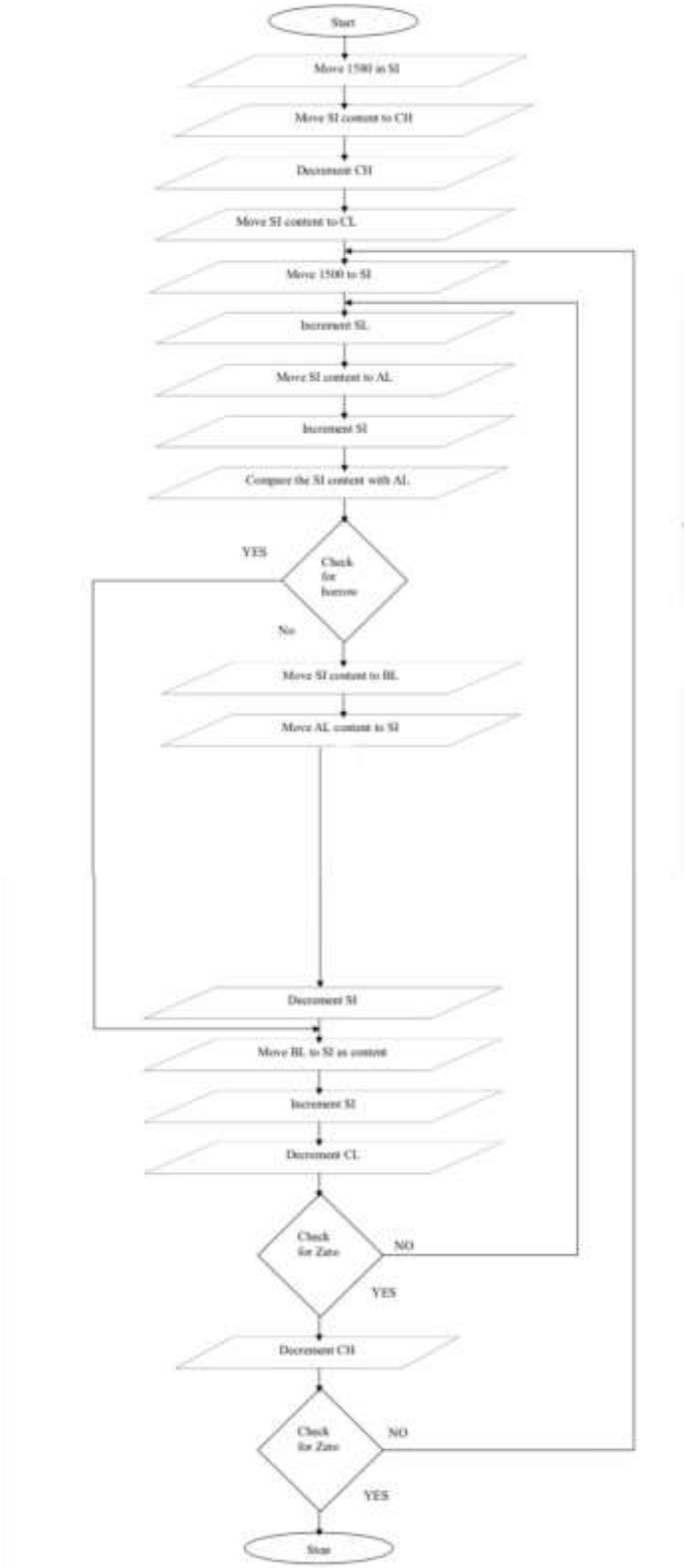
8086 kit
Keyboard

ASCENDING ORDER :-

ALGORITHM:

Step 1 : Set 1580 as SI
Step 2 : Move SI content to CH counter
Step 3 : Decrement CH
Step 4 : Move SI content to CL
Step 5 : Move 1500 to SI
Step 6 : Increment SI
Step 7 : Move SI content to AL
Step 8 : Increment SI
Step 9 : Compare the SI content with AL
Step 10 : If borrow jump
Step 11 : Move SI content to BL
Step 12 : Decrement SI
Step 13 : Move BL content to SI
Step 14 : Increment SI
Step 15 : Decrement CL
Step 16 : Jump if no zero
Step 17 : Decrement CH
Step 18 : Jump if no zero
Step 19 : Stop the program

FLOW CHART


Start
Move 1500 to SI
Move SI content to CH
Decrement CH
Move SI content to CL
Move 1500 to SI
Increment SI
Move SI content to AL
Increment SI
Compare the SI content with AL
YES
Check for borrow
No
Move SI content to BL
Move AL content to SI
Decrement SI
Move BL to SI as content
Increment SI
Decrement CL
Check for Zero
NO
YES
Decrement CH
Check for Zero
NO
YES
Stop

TABULATION :

ADDRESS	LABEL	MNEMONICS	COMMENTS
1001		MOV SI, 1580	Move 1580 to SI
1004		MOV CH, [SI]	Move [SI] in CH
1006		DEC CH	Decrement CH
1007		MOV CL, [SI]	Move [SI] in CL
1009	Loop 3	MOV SI,1500	Move 1500 in SI
100C		INC SI	Increment SI
100D	Loop 2	MOV AL, [SI]	Move [SI] in AL
100F		INC SI	Increment SI
1012		CMP AL, [SI]	Compare [SI] with AL
1014		JB 101C	Jump if borrow
1017		MOV BL, [SI]	Move [SI] to BL
1018		MOV [SI], AL	Move AL in [SI]
1019		DEC SI	Decrement SI
101A		MOV [SI], BL	Move BL in [SI]
101B		INC SI	Increment SI
101C	Loop1	DEC CL	Decrement CL
101D		JNZ 100D	Jump if no zero
1020		DEC CH	Decrement CH
1021		JNZ	Jump if no zero
1024		INT 3	Stop the program

RESULT:

INPUT		OUTPUT	
ADDRESS	INPUT	ADDRESS	OUTPUT

Thus sorting of numbers was done in ascending order successfully.

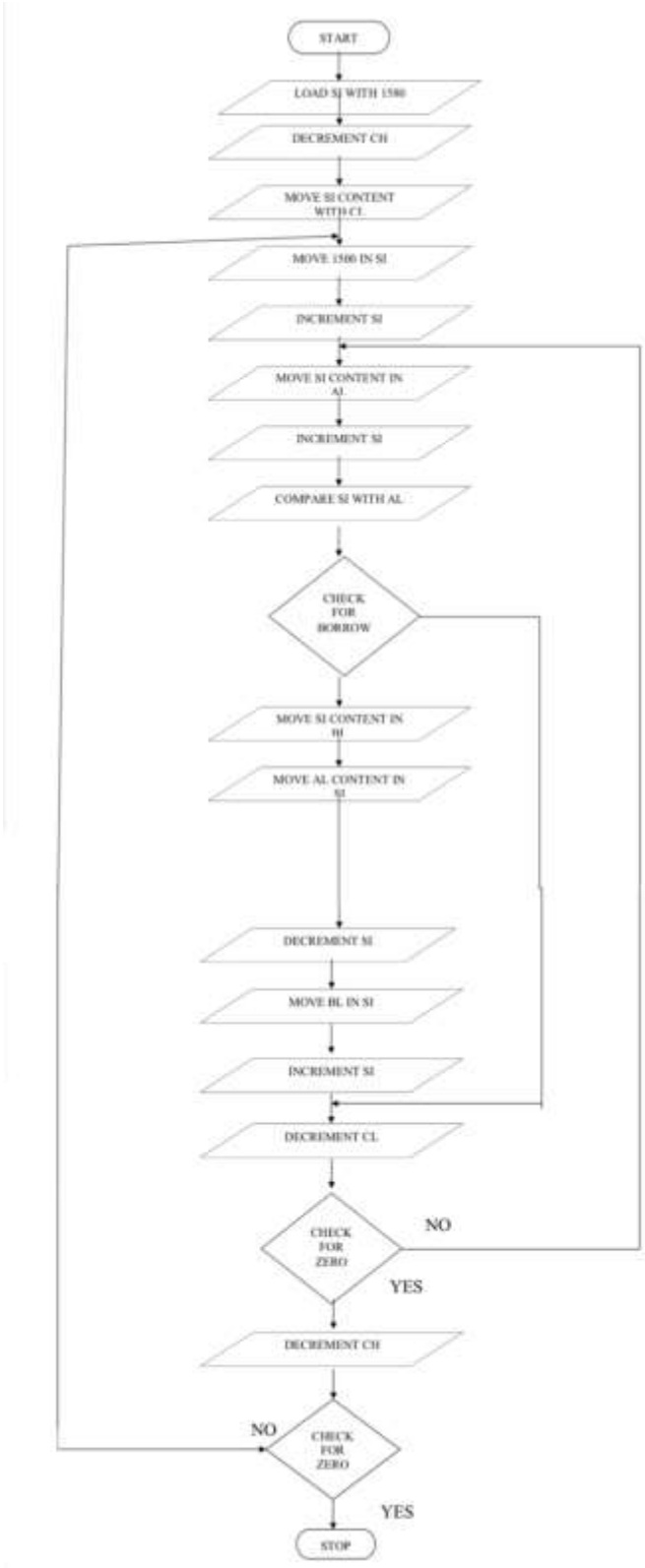
Prepared by
Vibha.k

DESCENDING ORDER :-

ALGORITHM:

- Move 1580 to SI
- Move the content of SI to CH
- Decrement CH
- Move SI content to CL
- Move 1500 to SI
- Increment SI
- Move SI content to AL
- Increment SI
- Compare the SI content with that in AL
- Jump if no borrow
- Move AL content to SI
- Decrement SI
- Increment SI
- Decrement CL
- Jump if no zero
- Decrement CH
- Jump if no zero
- Stop the program

FLOW CHART


START
LOAD SI WITH 1500
DECREMENT CH
MOVE SI CONTENT WITH CL
MOVE 1500 IN SI
INCREMENT SI
MOVE SI CONTENT IN AL
INCREMENT SI
COMPARE SI WITH AL
CHECK FOR BORROW
MOVE SI CONTENT IN BL
MOVE AL CONTENT IN SI
DECREMENT SI
MOVE BL IN SI
INCREMENT SI
DECREMENT CL
CHECK FOR ZERO
NO
YES
DECREMENT CH
NO
CHECK FOR ZERO
YES
STOP

TABULATION:

Address	Label	Mnemonics	Comments
1001		MOV SI,1580	Move 1580 to SI
1004		MOV CH,[SI]	Move [SI] to CH
1006		DEC CH	Decrement CH
1007		MOV CL,[SI]	Move [SI] to CL
1009	Loop3	MOV SI,1500	Move 1500 to SI
100C		INC SI	Increment SI
100D	Loop2	MOV AL,[SI]	Move [SI] to AL
100F		INC SI	Increment SI
1012		CMP AL,[SI]	Compare AL with [SI]
1014		JB 101C	Jump if borrow
1017		MOV BL,[SI]	Move [SI] to BL
1018		MOV [SI],AL	Move AL to [SI]
1019		DEC SI	Decrement SI
101A		MOV [SI],BL	Move BL to [SI]
101B		INC SI	Increment SI
101C	Loop1	DEC CL	Decrement CL
101D		JNZ 100D	Jump if no zero
1020		DEC CH	Decrement CH
1021		JNZ 1009	Jump if no zero
1024		INT3	Terminate

RESULT:

INPUT		OUTPUT	
ADDRESS	INPUT	ADDRESS	OUTPUT

Thus sorting of numbers in descending order was done successfully.

Prepared by
Vibha.k

Exercise Number: 4

Title of the Experiment: CODE CONVERSIONS USING 8086
Date of the Exercise:

AIM:

(a)To write and execute an assemble language program to perform code conversion for BCD to BINARY
(b)To write and execute an assemble language program to perform code conversion for BINARY TO BCD

APPARATUS REQUIRED:

4. Microprocessor 8086 kit.
5. Manual.

(a) CODE CONVERSION FOR BCD TO BINARY:-

ALGORITHM:

Step 1 : Start the microprocessor.

Step 2 : Move data ax and bcd data to dx and ax

Step 3: Perform and operation between o7 and bcd input

Step 4 : Move data ax and bcd data bx and ax

Step 5 : Perform multiplication by 10

Step 6 : Add the content of B register to accumulator.

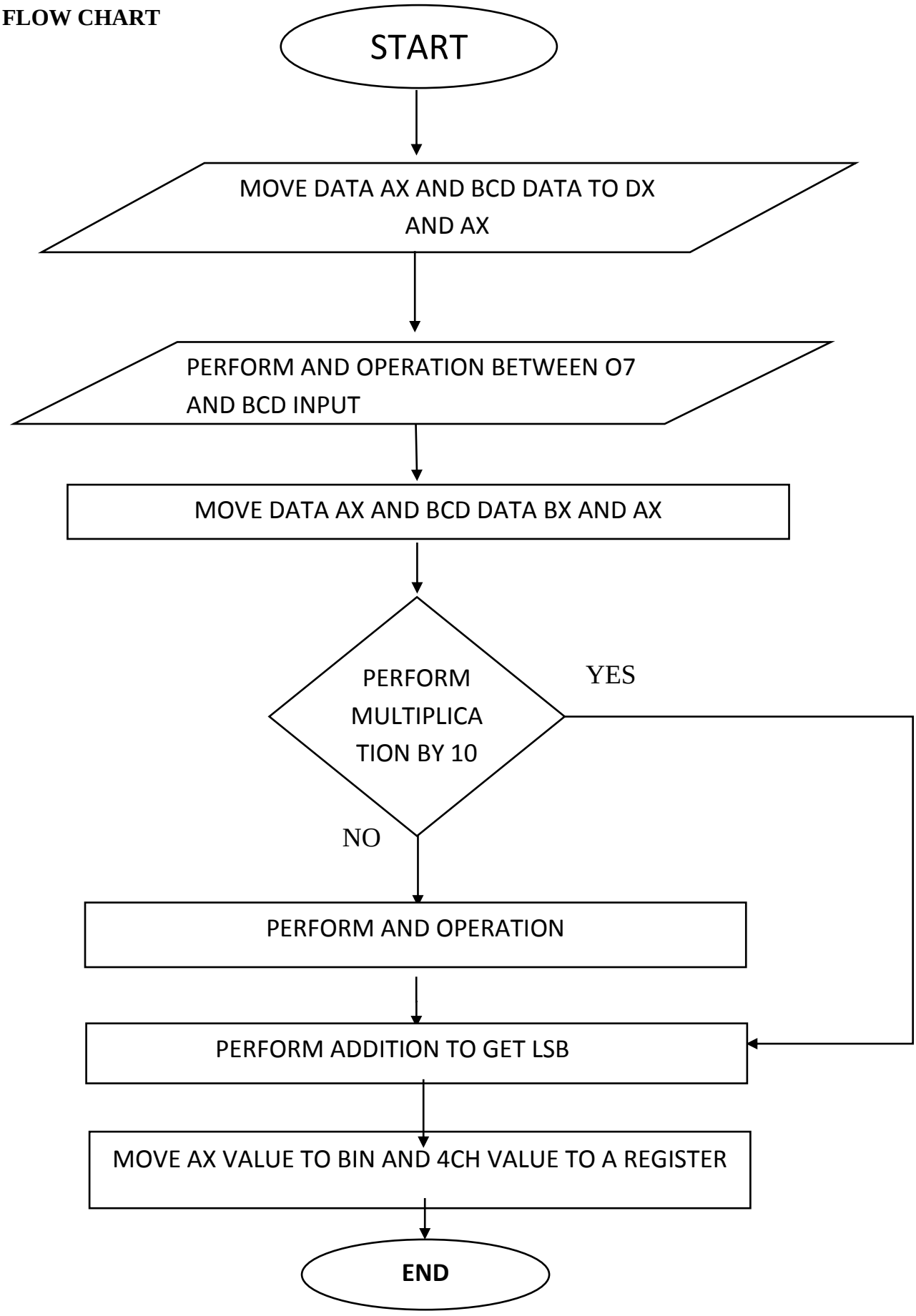
Step 7: Perform and operation

Step 8: Perform addition to get lsb

Step 9: Move ax value to bin and 4ch value to a register

Step10: Sto

FLOW CHART


START
MOVE DATA AX AND BCD DATA TO DX AND AX
PERFORM AND OPERATION BETWEEN O7 AND BCD INPUT
MOVE DATA AX AND BCD DATA BX AND AX
PERFORM MULTIPLICATION BY 10
YES
NO
PERFORM AND OPERATION
PERFORM ADDITION TO GET LSB
MOVE AX VALUE TO BIN AND 4CH VALUE TO A REGISTER
END

TABULATION:

ADDRESS	MNEMONICS	COMMENT
1000	MOV AX,DATA	STORE DATA AX
1001		
1003	MOV DX,AX	MOVE BCD DATA TO AX
1003		
1005	MOV AX,BCD	PERFORM AND OPERATION BETWEEN
1005		
1007	AND AX,07H	07 AND INPUT BCD
1009	MOV BX,AX	MOVE DATA TO BX
1008		
1009		
100B	MOV AX,BX	MOVE THE BCD DATA TO AX
100D	AND AX,0F0	PERFORM THE AND WITH 0F0H FOR SHIFTING OPERATION
100C		
100D		
100F	MOV CX,0AH	10 IN DECIMAL
1012	MUL CX	PERFROM MULTIPILICATION BY 10
1010		
1011		
1013	ADD AX,BX	PERFROM ADD OPERATION TO GET LSB
1015	MOV BIN,AX	MOV RESULT TO BINARY
1017	MOV AH,4CH	MOVE THE VALUE
1019	INT 21H	STOP THE EXECUTION

RESULT:

BCD to Binary			
INPUT		OUTPUT	
MEMORY LOCATION	CONTENT	MEMORY LOCATION	CONTENT
1001	80	1300	00
1003	80	1301	01

Thus BCD to BINARY conversion was verified.

Prepared by
Vibha.k

(b) CODE CONVERSION FOR BINARY TO BCD:-

ALGORITHM:

Step 1 : Start the microprocessor.

Step 2: Move data to ax.

Step3: Move data of hexadecimal to binary cl register.

Step4: Move data ax and bcd data cl and ax.

Step5: Perform divison for register.

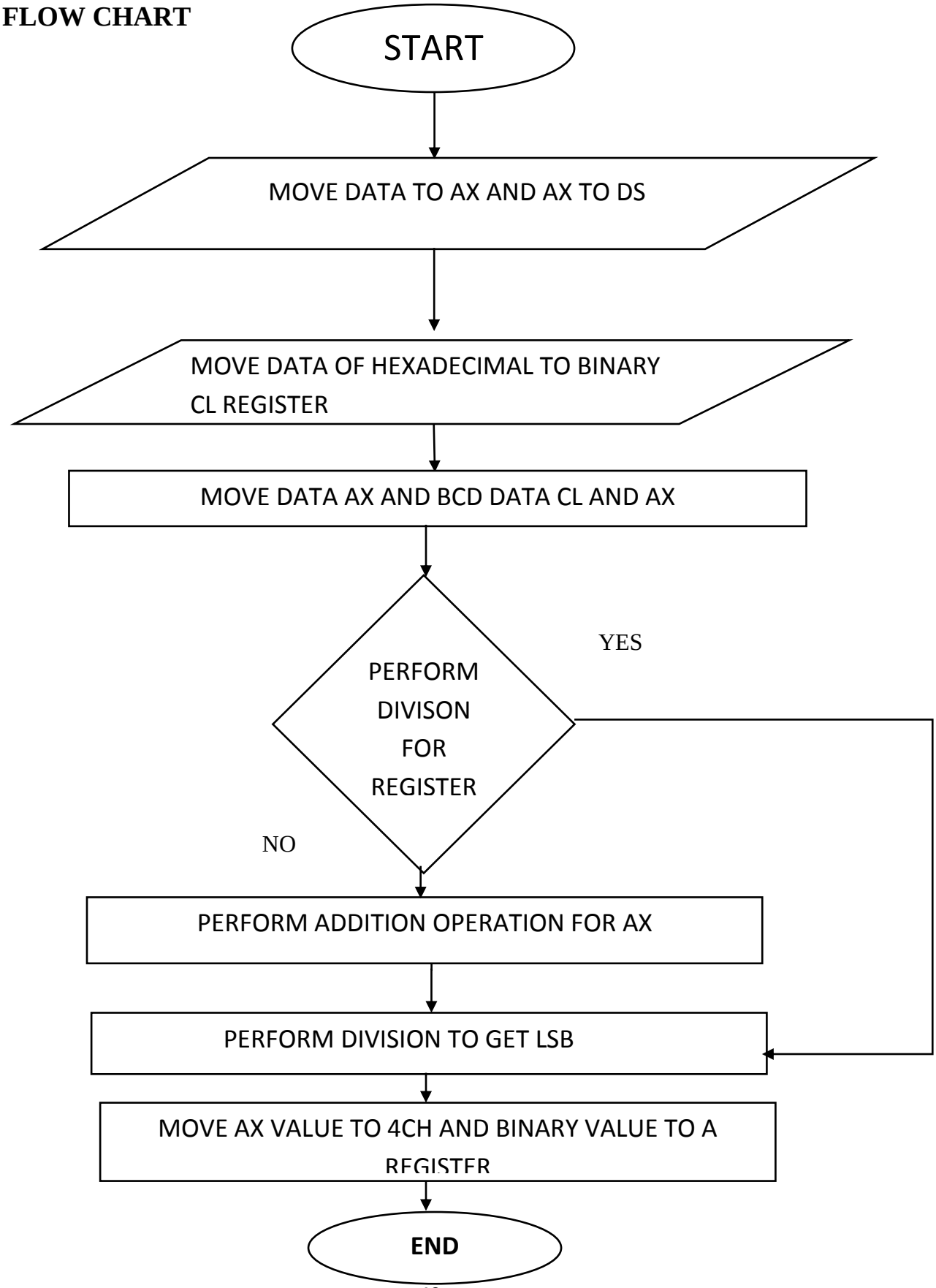
Step6: Perform addition operation for ax.

Step7: Perform division to get lsb.

Step8: Move ax value to 4ch and binary value to a register

Step9: Stop

FLOW CHART


START
MOVE DATA TO AX AND AX TO DS
MOVE DATA OF HEXADECIMAL TO BINARY
CL REGISTER
MOVE DATA AX AND BCD DATA CL AND AX
PERFORM
DIVISON
FOR
REGISTER
YES
NO
PERFORM ADDITION OPERATION FOR AX
PERFORM DIVISION TO GET LSB
MOVE AX VALUE TO 4CH AND BINARY VALUE TO A
REGISTER
END

TABULATION:

ADDRESS	MNEMONICS	COMMENT
1000	MOV AX,DATA	MOVE DATA TO AX REGISTER
1002		
1003	MOV DS,AX	MOVE AX DATA TO DS
1004		
1005	MOV AX,BIN	MOVE THE BINARY TO AX
1005		
1007	MOV CL,64H	MOVE THE DATA OF HEXADECIMAL TO CL REGISTER
1009	DIV CL	PERFORM DIVISON OPERATION
1008		
1009		
100B	MOV BCD+1,AL	MOVE DATA TO ACCUMULATOR
100D	MOV AL,AH	
100C		
100D		
100F	MOV AH,00H	MOVE THE DATA TO AH REGISTER
1012	MOV CL,0AH	MOVE OAH TO CL REGISTER
1010		
1011		
1014	DIV CL	PERFORM DIVISON OPERATION FOR CL
1015	MOV CL,04	MOVE 04 TO CL REGISTER
1017	ROR AL,CL	ROTATE RIGHT CL TO AL REGISTER
1019	ADD AL,AH	PERFORM ADD OPERATION
1021	MOV AH,4CH	MOVE HEXADECIMAL TO AH REGISTER
1025	INT 21H	STOP

RESULT:

Binary to BCD			
INPUT		OUTPUT	
MEMORY LOCATION	CONTENT	MEMORY LOCATION	CONTENT
1001	80	1300	00
1003	80	1301	01

Thus Binary to BCD conversion was verified

Prepared by
Vibha.k

Exercise Number: 5

Title of the Experiment: BLOCK DATA TRANSFER IN 8086
Date of the Exercise:

AIM:

(a) To write and execute an assemble language program to block transfer in forward order.

(b) To write and execute an assemble language program to block transfer in reverse order.

APPARATUS REQUIRED:

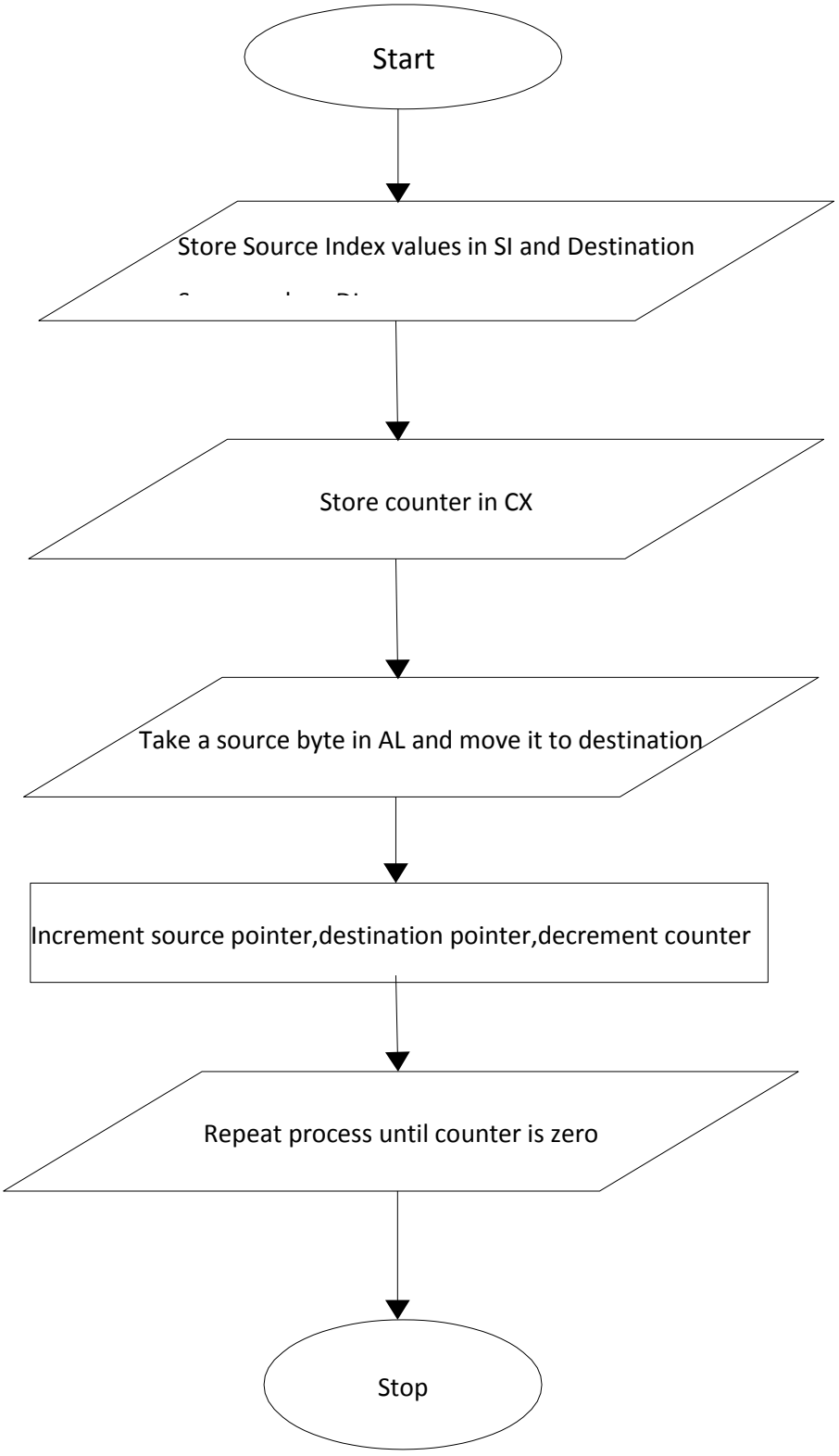
1. Microprocessor 8086 kit.
2. Manual.

a) BLOCK TRANSFER IN FORWARD ORDER

ALGORITHM:

1. Start the program
2. Initialization of segment register, counter and program
3. Move a byte from source to destination
4. Upgrade pointer, decrement counter
5. Continue the loop if count is not zero
6. Stop if count is not zero

FLOW CHART


Start
Store Source Index values in SI and Destination
Store counter in CX
Take a source byte in AL and move it to destination
Increment source pointer,destination pointer,decrement counter
Repeat process until counter is zero
Stop

TABULATION:

ADDRESS	MNEMONICS	COMMENT
1000	MOV SI,1500	Move pointer to string source
1003	MOV DI,1600	Move the pointer to destination string
1006	MOV CX,05	Count for length of string
1008	MOV AL,[SI]	Take a source byte in AL
100A	MOV [DI],AL	Move it to destination
100C	INC SI	Increment source pointer
100D	INC DI	Increment destination pointer
100E	DEC CX	Decrement count by 1
100F	JNZ	Continue if count is not zero
1010	INT3	Stop if count is zero

RESULT:

DATA TRANSFER			
INPUT		OUTPUT	
MEMORY LOCATION	CONTENT	MEMORY LOCATION	CONTENT
1500	01	1600	01
1501	02	1601	02
1502	03	1602	03
1503	04	1603	04
1504	05	1604	05

Thus the block data transfer in forward order is done.

b) BLOCK TRANSFER IN REVERSE ORDER

ALGORITHM:

1. Start the program and store the source address in the source index.
2. Store the destination address in destination index.
3. Set the counter value.
4. Set directional flag to 1 for decrement order.
5. Store counter value in CX.
6. Repeat the process until CX=0.
7. END of the program.

TABULATION:

ADDRESS	**MNEMONICS**	**COMMENT**
1000	MOV SI	Store the source address in source index
1001	00	
1002	11	
1003	MOV DI	Store the destination in DI
1004	04	
1005	12	
1006	STD	Set direction flag for decrement
1007	01	
1008	MOV CX	Store the value in CX register
1009	05	
100A	REPE	Repeat the process until CX=0
100B	MOVSB	
100C	INT3	End the program

RESULT:

DATA TRANSFER			
INPUT		OUTPUT	
MEMORY LOCATION	CONTENT	MEMORY LOCATION	CONTENT
1100	01	1200	05
1101	02	1201	04
1102	03	1202	03
1103	04	1203	02
1104	05	1204	01

Thus the block data transfer in reverse order is done.

Prepared by
K.A.Sunitha

Exercise Number: 6

Title of the Experiment: TRAFFIC LIGHT CONTROL USING 8051
Date of the Exercise:

Aim:

To interface traffic light controller with 8051 microcontroller using 8255.

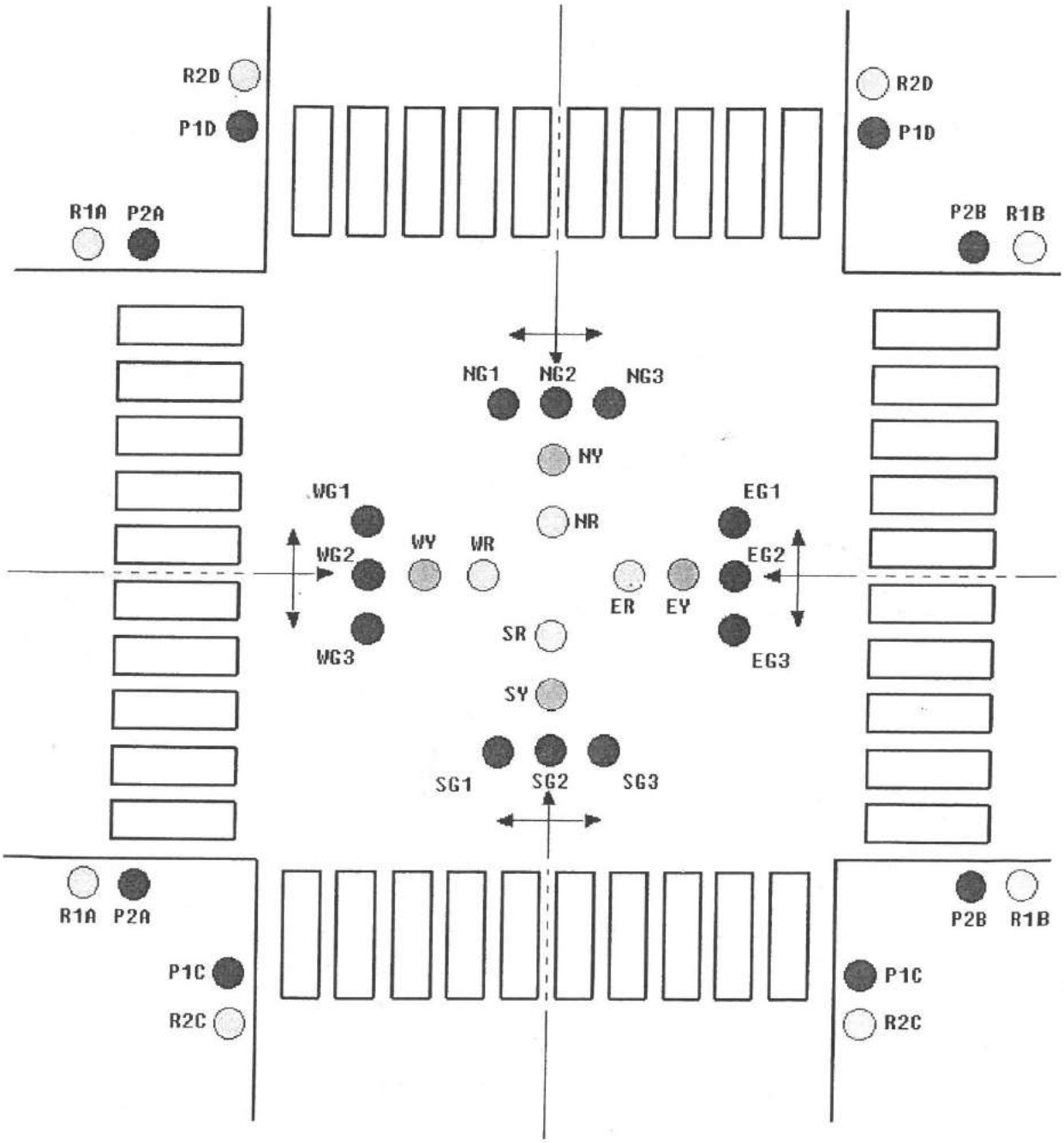
Apparatus required:

Microcontroller 8051 kit
8255 peripheral device
Traffic Light Controller

Procedure:

Step 1	:	Connect the 5V supply to the trainer kit.
Step 2	:	Connect the 26-pin FRC from the kit.
Step 3	:	Switch ON the power supply.
Step 4	:	Assemble the program.
Step 5	:	Execute it and output display by LED.
Step 6	:	Switch OFF the power supply and remove the connectors.

FRONT PANEL
TRAFFIC LIGHT CONTROLLER



Main Program:

Address	Label	Mnemonics	Hex-Code	Comment
9000		MOV A, #80H	74 80	ALL PORTS AS O/P
9002		MOV DPTR,#PPI+3	90 60 03	
9005		MOVX@DPTR,A	F0	

For Starting Vehicles N-Direct(St) & Pedest Stopping:

Address	Label	Mnemonics	Hex-Code	Comment
9006		MOV A, #0FH	74 0F	FOR PEDESTRIAN
9008		MOV DPTR,#PPI+1	90 60 01	SIGNAL
900B		MOVX@DPTR,A	F0	
900C		MOV A, #4DH	74 4D	FOR GREEN LEDS IN N-S
900E		MOV DPTR,#PPI	90 60 00	DIRECTION
9011		MOVX@DPTR,A	F0	
9012		LCALL DELAY	12 90 7E	SEQUENCE DELAY
9015		LCALL AMBER	12 90 72	AMBER DELAY

For Stop Vehicles In N-S Direct & Start In E-W Direction:

Address	Label	Mnemonics	Hex-Code	Comment
9018		MOV DPTR,#PPI	90 60 00	FOR STOPPING N-S SIDES
901B		MOV A, #8BH	74 8B	STARTING E-W SIDES
901D		MOVX@DPTR,A	F0	
901E		LCALL DELAY	12 90 7E	SEQUENCY DELAY
9021		LCALL AMBER	12 90 72	AMBER DELAY

For Straight Right Turn In N-S Sides & Stoping E-W Sides:

Address	Label	Mnemonics	Hex-Code	Comment
9024		MOV A, #49H	74 49	FOR FREE LEFT IN ALL
9026		MOV DPTR,#PPI	90 60 00	SIDES & STOPPING
9029		MOVX@DPTR,A	F0	IN E-W SIDES
902A		MOV DPTR,#PPI+2	90 60 02	FOR RIGHT TURN IN N-S
902D		MOV A, #1H	74 01	SIDES
902F		MOVX@DPTR,A	F0	
9030		LCALL DELAY	12 90 7E	SEQUENCE DELAY
9033		MOV A, #0H	74 00	FOR AMBER
9035		MOVX@DPTR,A	F0	SIGNAL
9036		LCALL AMBER	12 90 72	FOR AMBER DELAY

Stop Right Turn In N-S Sides & Start Right Turn In E-W:

Address	Label	Mnemonics	Hex-Code	Comment
9039		MOV A, #89H	74 89	FOR STOPPING VEHICLES
903B		MOV DPTR,#PPI	90 60 00	IN N-S SIDES
903E		MOVX@DPTR,A	F0	
903F		MOV DPTR,#PPI+2	90 60 02	FOR RIGHT TURN IN
9042		MOV A, #2H	74 02	E-W SIDES
9044		MOVX@DPTR,A	F0	
9045		LCALL DELAY	12 90 7E	SEQUENCE DELAY
9048		MOV A, #0H	74 00	
904A		MOVX@DPTR,A	F0	
904B		MOVA, #30H	74 30	
904D		MOV DPTR,#PPI	90 60 00	
9050		MOVX@DPTR,A	F0	
9051		MOV R1,#4H	79 04	
9053		LCALL DELAYSUB	12 9084	FOR AMBER DELAY

For Starting Pedestrain:

Address	Label	Mnemonics	Hex-Code	Comment
9056		MOV A, #0C00H	74 C0	FOR STOPPING VEHICLE
9058		MOV DPTR,#PPI	90 60 00	IN ALL SIDES
905B		MOVX@DPTR,A	F0	
905C		INC DPTR	A3	GREEN SIGNAL FOR
905D		MOV A, #0F00H	74 F0	PEDESTRAIN
905F		MOVX@DPTR,A	F0	
9060		MOV R1,#10H	79 10	DELAY FOR PEDESTRAIN
9062		LCALL DELAYSUB	12 90 84	
9065		MOVA, #30H	74 30	
9067		MOV DPTR,#PPI	90 60 00	
906A		MOVX@DPTR,A	F0	
906B		MOV R1,#8H	79 08	
906D		LCALL DELAYSUB	12 90 84	AMBER DELAY
9070		AJMP CONTINUE	01 06	

Amber:

Address	Label	Mnemonics	Hex-Code	Comment
9072		MOV DPTR,#PPI	90 60 00	FOR AMBER SIGNAL
9075		MOV A, #39H	74 39	IN ALL SIDES
9077		MOVX@DPTR,A	F0	
9078		MOV R1,#8H	79 08	DELAY COUNT
907A		LCALL DELAYSUB	12 90 84	
907D		RET	22	

Delay:

Address	Label	Mnemonics	Hex-Code	Comment
907E		MOV R1,#040H	79 40	DELAY COUNT FOR
9080		LCALL DELAYSUB	12 90 84	GREEN & RED SIGNALS
9083		RET	22	

Delay Subroutine:

Address	Label	Mnemonics	Hex-Code	Comment
9084		MOV R2,#0FFH	7A FF	
9086		MOV A,#0FFH	74 FF	
9088		NOP	00	
9089		DEC A	14	
908A		JNZ BACK	70 FC	
908C		DEC R2	1A	
908D		MOV A,R2	EA	
908E		JNZ BACK1	70 F6	
9090		MOV A,R1	E9	
9091		JZ OUT	60 03	
9093		DEC R1	19	
9094		JNZ BACK2	70 EE	
9096		RET	22	

Result:

Hence the Traffic Light Controller is interfaced with 8051 microcontroller.

Prepared by
G.Y.Rajaa Vikhram

Exercise Number: 7

Title of the Experiment: STEPPER MOTOR CONTROL USING 8051 CONTROLLER
Date of the Exercise:

Aim:

To interface stepper motor with 8051 microcontroller

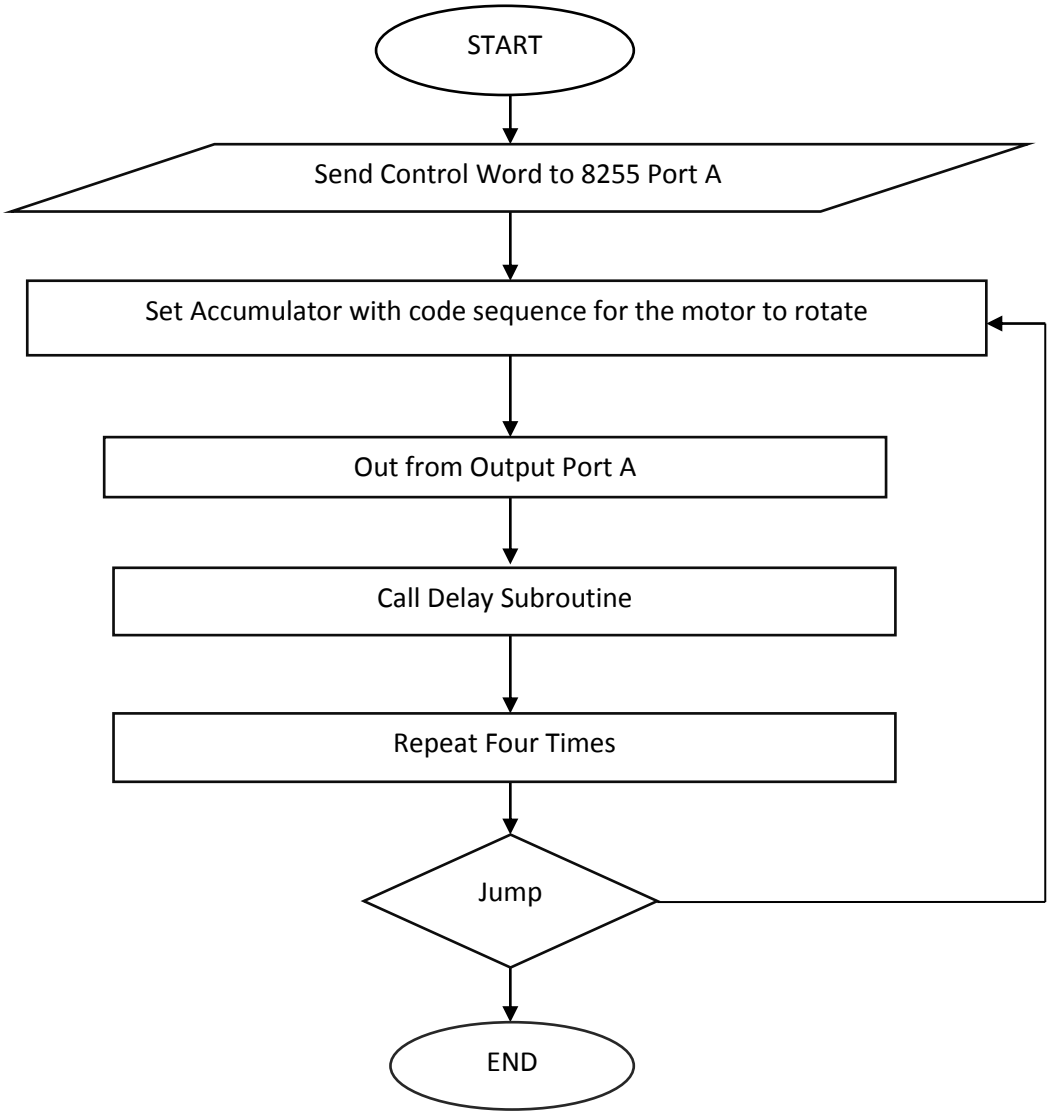
Apparatus required:

Microcontroller 8051 kit
8255 peripheral device
Stepper motor

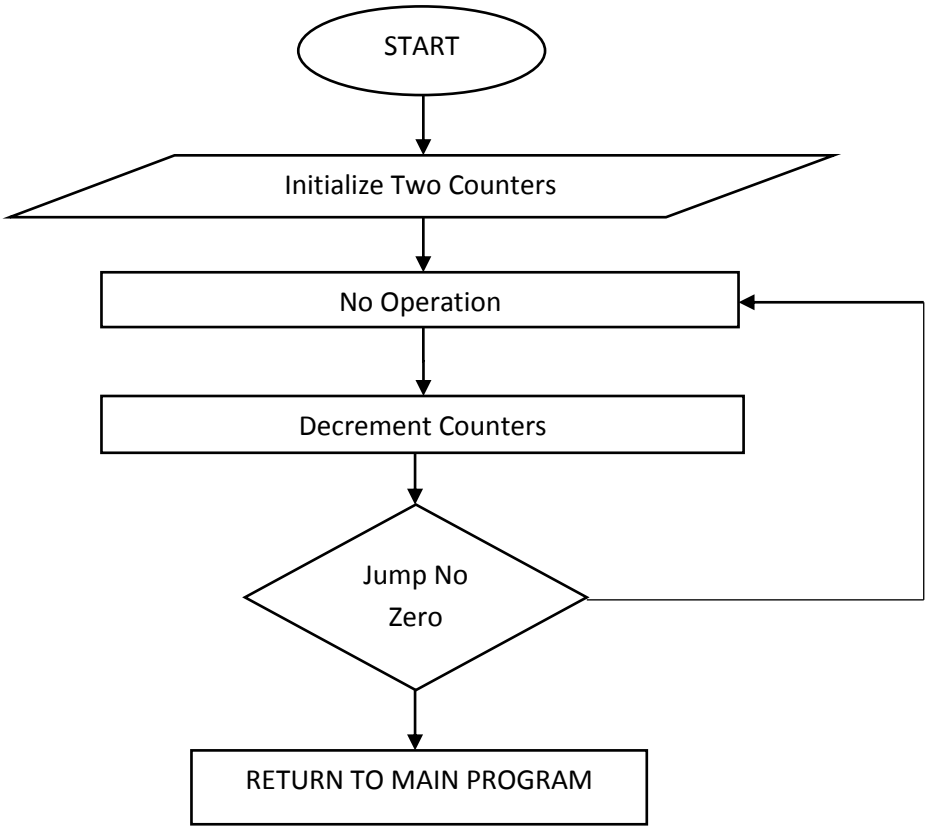
Algorithm:

Step 1	:	Start the microcontroller.
Step 2	:	Input the key control words 0A, 06, 03, 09.
Step 3	:	Move to the accumulator and given through output port.
Step 4	:	Introduce the time delay routine.
Step 5	:	Using jump condition continue from the beginning to get continuous rotation.
Step 6	:	Stop.

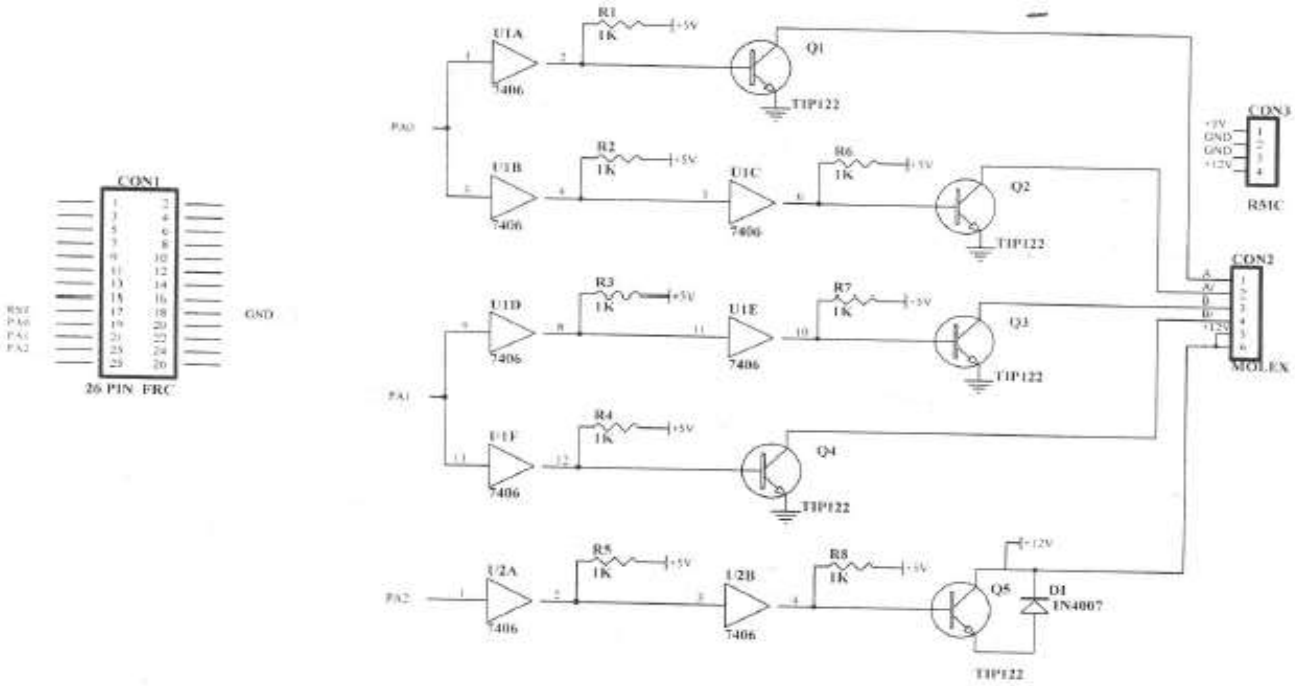
Flow Chart: **(**Main Program)



Flow Chart: **(**Subroutine Program)



STEPPER MOTOR INTERFACE



Main Program:

Clockwise Direction:

Address	Label	Mnemonics	Hex-Code	Comment
9000		MOV DPTR, #6003H	906003	Control Port of 8255
9003		MOV A, #80H	7480	
9005		MOVX@DPTR,A	F0	All bits outputs
9006		MOV DPTR, #6000H	906000	
9009	LOOP1	MOV A, #05H	7405	First Step Sequence
900B		MOVX@DPTR, A	F0	
900C		LCALL 9024(DELAY)	129024	Call delay
900F		MOV A, #07H	7407	Second step sequence
9011		MOVX@DPTR,A	F0	
9012		LCALL 9024(DELAY)	129024	Delay
9015		MOV A, #06H	7406	Third step sequence
9017		MOVX@DPTR,A	F0	
9018		LCALL 9024(DELAY)	129024	Delay
901B		MOV A, #04H	7404	Fourth sequence
901D		MOVX@DPTR,A	F0	
901E		LCALL 9024(DELAY)	129024	Delay
9021		LJMP 9009(LOOP1)	029009	Repeat

Anti Clockwise Direction:

Address	Label	Mnemonics	Hex-Code	Comment
9100		MOV DPTR, #6003H	906003	Control Port of 8255
9103		MOV A, #80H	7480	
9105		MOVX@DPTR,A	F0	All bits outputs
9106		MOV DPTR, #6000H	906000	
9109	LOOP2	MOV A, #04H	7404	First Step Sequence
910B		MOVX@DPTR, A	F0	
910C		LCALL 9024(DELAY)	129024	Call delay
910F		MOV A, #06H	7406	Second step sequence
9111		MOVX@DPTR,A	F0	
9112		LCALL 9024(DELAY)	129024	Delay
9115		MOV A, #07H	7407	Third step sequence
9117		MOVX@DPTR,A	F0	
9118		LCALL 9024(DELAY)	129024	Delay
911B		MOV A, #05H	7405	Fourth sequence
911D		MOVX@DPTR,A	F0	
911E		LCALL 9024(DELAY)	129024	Delay
9121		LJMP 9109(LOOP2)	029109	Repeat

Delay Subroutine:

Address	Label	Mnemonics	Hex-Code	Comment
9024		MOV R1, #0AH	790A	Changing the speed of rotation
9026	LOOP3	MOV A, #40H	7440	
9028	LOOP4	NOP	00	
9029		NOP	00	
902A		NOP	00	
902B		NOP	00	
902C		DEC A	14	
902D		JNZ 9028(LOOP4)	70F9	
902F		DJNZ R1,9026(LOOP3)	D9F5	
9031		RET	22	

Result:

Hence the control of stepper motor using 8051 microcontroller is successfully submitted and verified.

Prepared by
G.Y.Rajaa Vikhram

Exercise Number: 8

Title of the Experiment: TEMPERATURE CONTROL SYSTEM USING 8051
Date of the Exercise:

Exercise Number: 9

Title of the Experiment: LCD DISPLAY USING 8051
Date of the Exercise:

Aim:

To interface LCD with 8051 microcontroller using 8255.

Apparatus required:

8051 micro controller kit
1 & 10K TRIM POT
26 PIN FRC
16 PIN RMC
2 PIN MOLEX

Procedure:

Step 1	:	Connect the 5V power supply to the LCD Interface card.
Step 2	:	Connect the 26-pin FRC from the kit to the LCD card.
Step 3	:	Switch ON the power supply.
Step 4	:	Assemble the program.
Step 5	:	Execute it and view the display in the LCD.
Step 6	:	Switch OFF the power supply and remove all the connections.

Main Program:

Address	Label	Mnemonics	Hex Code	Comments
8900		LCALL INIT	12 89 42	
8903		MOV DPTR,#6001H	90 60 01	PORT B
8906		MOV A,#01H	74 01	
8908		MOVX @DPTR,A	F0	
8909		LCALL LCDENA	12 89 73	
890C		MOV DPTR,#6001H	90 60 01	PORT B
890F		MOV A,#41H	74 41	CHAR “A”
8911		MOVX @DPTR,A	F0	
8912		LCALL LCD ENA DATA	12 89 83	
8915		MOV A,#42H	74 42	CHAR “B”
8917		MOVX @DPTR,A	F0	
8918		LCALL LCD ENA DATA	12 89 83	
891B		MOV A,#43H	74 43	CHAR “C”
891D		MOVX @DPTR,A	F0	
891E		LCALL LCD ENA DATA	12 89 83	

Second line:

Address	Label	Mnemonics	Hex Code	Comments
8921		MOV DPTR,#6001H	90 60 01	PORT B
8924		MOV A,#0C0H	74 C0	2nd LINE ADD
8926		MOVX @DPTR,A	F0	
8927		LCALL LCDENA	12 89 73	
892A		MOV DPTR,#6001H	90 60 01	PORT B
892D		MOV A,#31H	74 31	NUM “1”
892F		MOVX @DPTR,A	F0	
8930		LCALL LCD ENA DATA	12 89 83	
8933		MOV A,#32H	74 32	NUM “2”
8935		MOVX @DPTR,A	F0	
8936		LCALL LCD ENA DATA	12 89 83	
8939		MOV A,#33H	74 33	NUM “3”
893B		MOVX @DPTR,A	F0	
893C		LCALL LCD ENA DATA	12 89 83	
893F		LCALL 00BBH	12 00 BB	

LCD Initialisation:

Address	Label	Mnemonics	Hex-Code	Comment
8942		MOV DPTR, #6003H	90 60 03	CWR FOR 8255
8945		MOV A, #80H	74 80	
8947		MOVX@DPTR,A	F0	
8948		MOV DPTR, #6001H	90 60 01	PORT B DATA
894B		MOV A, #38H	74 38	FUNCTION TEST
894D		MOVX@DPTR, A	F0	
894E		LCALL LCDENA	12 89 73	
8951		MOV A, #38H	74 38	FUNCTION TEST
8953		MOVX@DPTR,A	F0	
8954		LCALL LCDENA	12 89 73	
8957		MOV A, #38H	74 38	FUNCTION TEST
8959		MOVX@DPTR,A	F0	
895A		LCALL LCDENA	12 89 73	
895D		MOV A, #0FH	74 0F	DIS ON/OFF
895F		MOVX@DPTR,A	F0	
8960		LCALL LCDENA	12 89 73	
8963		MOV A, #06H	74 06	ENTRY MODE
8965		MOVX@DPTR,A	F0	
8966		LCALL LCDENA	12 89 73	
8969		MOV A, #01H	74 01	CLEAR SCREEN
896B		MOVX@DPTR,A	F0	
896C		LCALL LCDENA	12 89 73	
896F		LCALL LCDENA	12 89 73	
8972		RET	22	

LCDENA:

Address	Label	Mnemonics	Hex-Code	Comment
8973		MOV DPTR, #6002H	90 60 02	PORT C
8976		MOV A, #02H	74 02	LCD ENA
8978		MOVX@DPTR,A	F0	
8979		MOV A, #00H	74 00	LCD DISA
897B		MOVX@DPTR,A	F0	
897C		LCALL DELAY	12 89 93	
897F		MOV DPTR,#6001H	90 60 01	PORT B
8982		RET	22	

LCDENADATA:

Address	Label	Mnemonics	Hex-Code	Comment
8983		MOV DPTR, #6002H	90 60 02	PORT C
8986		MOV A, #03H	74 03	ENA DATA
8988		MOVX@DPTR,A	F0	
8989		MOV A, #00H	74 00	DISA DATA
898B		MOVX@DPTR,A	F0	
898C		LCALL DELAY	12 89 93	
898F		MOV DPTR,#6001H	90 60 01	PORT B
8992		RET	22	

DELAY ROUTINE:

Address	Label	Mnemonics	Hex-Code	Comment
8993		MOV R5, #3FH	7D 3F	
8995	D2	MOV A,#FFH	74 FF	
8997	D1	NOP	00	
8998		NOP	00	
8999		NOP	00	
899A		NOP	00	
899B		NOP	00	
899C		DEC A	14	
899D		JNZ D1	70 F8	
899F		DJNZ, D2	DD F4	
89A1		RET	22	

Result:

Hence the LCD display is interfaced with 8051 microcontroller.

Prepared by
G.Y.Rajaa Vikhram

Exercise Number: 10

Title of the Experiment: SEVEN SEGMENT DISPLAY USING nuvoTON BOARD
Date of the Exercise:

AIM: To interface a seven segment display with a NUC140 nuvoTON board.

APPARATUS REQUIRED: NUC140 nuvoTON board, 7 segment display

PROCEDURE:

CODE:

```
#include
"NUC1xx.h"
void clr_segment(void)
{
unsigned char i;
for(i=0; i<4; i++)
{
GPIOC->DOUT &= ~(1<<(4+i));
}
}
void show_segment(unsigned char n, unsigned char number)
{
unsigned char i;
unsigned char number__[10] = {0x82 ,0xEE, 0x07, 0x46, 0x6A, 0x52, 0x12,
0xE6, 0x02, 0x62};
for(i=0; i<8; i++)
{
if((number__[number_]&0x01)==0x01)
GPIOE->DOUT |= (1<<i);
else
GPIOE->DOUT &= ~(1<<i);
number__[number_]=number__[number_]>>1;
}
GPIOC->DOUT |= (1<<(4+n));
}
```

Prepared by
K. A. Sunitha