

INFORMATION TECHNOLOGY DEPARTMENT

MICROPROCESSORS LAB MANUAL



MICROPROCESSORS LAB

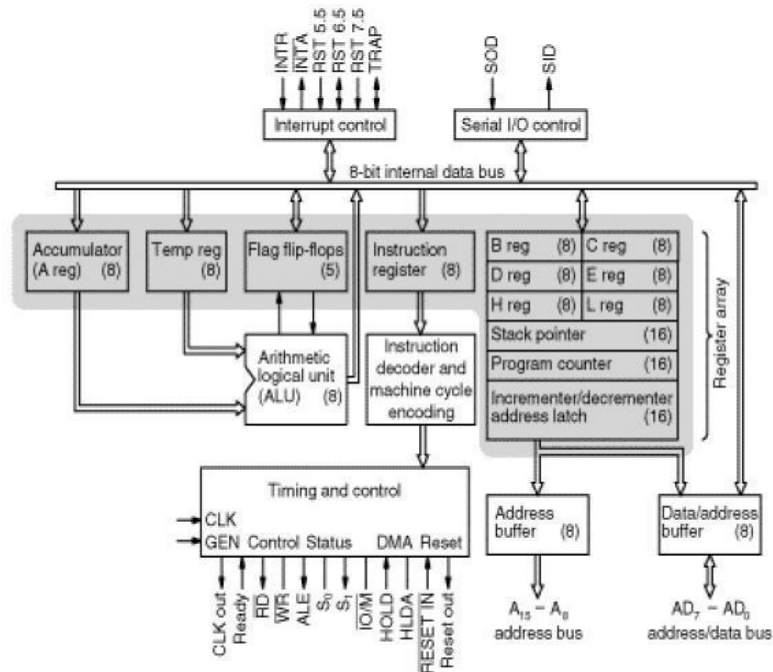
S. No.	CONTENTS	PAGE No.
1	Introduction to 8085 Microprocessor	IV
2	General Guidelines & Safety instructions	XXVII
PROGRAMS		
1	Program 1: 8- bit Subtraction	1
2	Program 2: 8- bit Division	2
3	Program 3: Palindrome	3
4	Program 4: Ascending order	4
5	Program 5: Descending order	6
6	Program 6: 16- bit Addition	8
7	Program 7: BCD to binary conversion	9
8	Program 8: Binary to BCD conversion	10
9	Program 9: Addition of a series of numbers	11
10	Program 10: 8- bit Multiplication	13
11	Program 11: Largest number in a list	14
12	Program 12: Stepper Motor	16
13	Program 13: Traffic Light	17
14	Program 14: LCD	18
15	Program 15: 7 Segment display	19
16	Program 16: Generation of Waveforms	20
17	Program 17: 8279 Interfacing	21
18	Annexure-I: Microprocessors Lab-OU Syllabus	22

1. INTRODUCTION TO 8085 MICROPROCESSOR

1.1 Introduction

The 8085 microprocessor was made by Intel in mid 1970s. It was binary compatible with 8080 microprocessor but required less supporting hardware thus leading to less expensive microprocessor systems. It is a general purpose microprocessor capable of addressing 64k of memory. The device has 40 pins, require a +5V power supply and can operate with 3 MHz single phase clock. It has also a separate address space for up to 256 I/O ports. The instruction set is backward compatible with its predecessor 8080 even though they are not pin-compatible.

1.2 8085 Internal Architecture



The 8085 has a 16 bit address bus which enables it to address 64 KB of memory, a data bus 8 bit wide and control buses that carry essential signals for various operations. It also has a built in register array which are usually labelled A(Accumulator), B, C, D, E, H, and L. Further special-purpose registers are the 16-bit Program Counter (PC), Stack Pointer (SP), and 8-bit flag register F. The microprocessor has three maskable interrupts (RST 7.5, RST 6.5 and RST 5.5), one Non-Maskable interrupt (TRAP), and one externally serviced interrupt (INTR). The RST n.5 interrupts refer to actual pins on the processor a feature which permitted simple systems to avoid the cost of a separate interrupt controller chip.

Control Unit

It generates signals within microprocessor to carry out the instruction, which has been decoded. In reality causes certain connections between blocks of the processor be opened or closed, so that data goes where it is required, and so that ALU operations occur.

Arithmetic Logic Unit

The ALU performs the actual numerical and logic operation such as „add“, „subtract“, „AND“, „OR“, etc. Uses data from memory and from Accumulator to perform arithmetic and always stores the result of operation in the Accumulator.

Registers

The 8085 microprocessor includes six registers, one accumulator, and one flag register, as shown in Fig 1. In addition, it has two 16-bit registers: the stack pointer and the program counter. The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H, and L as shown in Fig 1. They can be combined as register pairs - BC, DE, and HL - to perform some 16-bit operations. The programmer can use these registers to store or copy data into the registers by using data copy instructions.

Accumulator

The accumulator is an 8-bit register that is a part of arithmetic/logic unit (ALU). This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

Flag Registers

The ALU includes five flip-flops, which are set or reset after an operation according to data conditions of the result in the accumulator and other registers. They are called Zero(Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags. The most commonly used flags are Zero, Carry, and Sign. The microprocessor uses these flags to test data conditions.

Program Counter (PC)

This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. Memory locations have 16-bit addresses, and that is why this is a 16-bit register. The microprocessor uses this register to sequence the execution of

the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location.

Stack Pointer (SP)

The stack pointer is also a 16-bit register used as a memory pointer. It points to a memory location in R/W memory, called the stack. The beginning of the stack is defined by loading 16-bit address in the stack pointer.

Instruction Register / Decoder

This is a temporary storage for the current instruction of a program. Latest instruction is sent to here from memory prior to execution. Decoder then takes instruction and „decodes“ or interprets the instruction. Decoded instruction is then passed to next stage.

Memory Address Register (MAR)

It holds addresses received from PC for eg: of next program instruction. MAR feeds the address bus with address of the location of the program under execution.

Control Generator

It generates signals within microprocessor to carry out the instruction which has been decoded. In reality it causes certain connections between blocks of the processor to be opened or closed, so that data goes where it is required, and so that ALU operations occur.

Register Selector

This block controls the use of the register stack. Just a logic circuit which switches between different registers in the set will receive instructions from Control Unit.

8085 System Bus

The microprocessor performs four operations primarily.

- Memory Read
- Memory Write
- I/O Read
- I/O Write

All these operations are part of the communication processes between microprocessor and peripheral devices. The 8085 performs these operations using three sets of communication lines called buses - the address bus, the data bus and the control bus.

Address Bus

The address bus is a group of 16 lines. The address bus is unidirectional: bits flow only in one direction – from the 8085 to the peripheral devices. The microprocessor uses the address bus to perform the first function: identifying a peripheral or memory location. Each peripheral or memory location is identified by a 16 bit address. The 8085 with its 16 lines is capable of addressing 64 K memory locations.

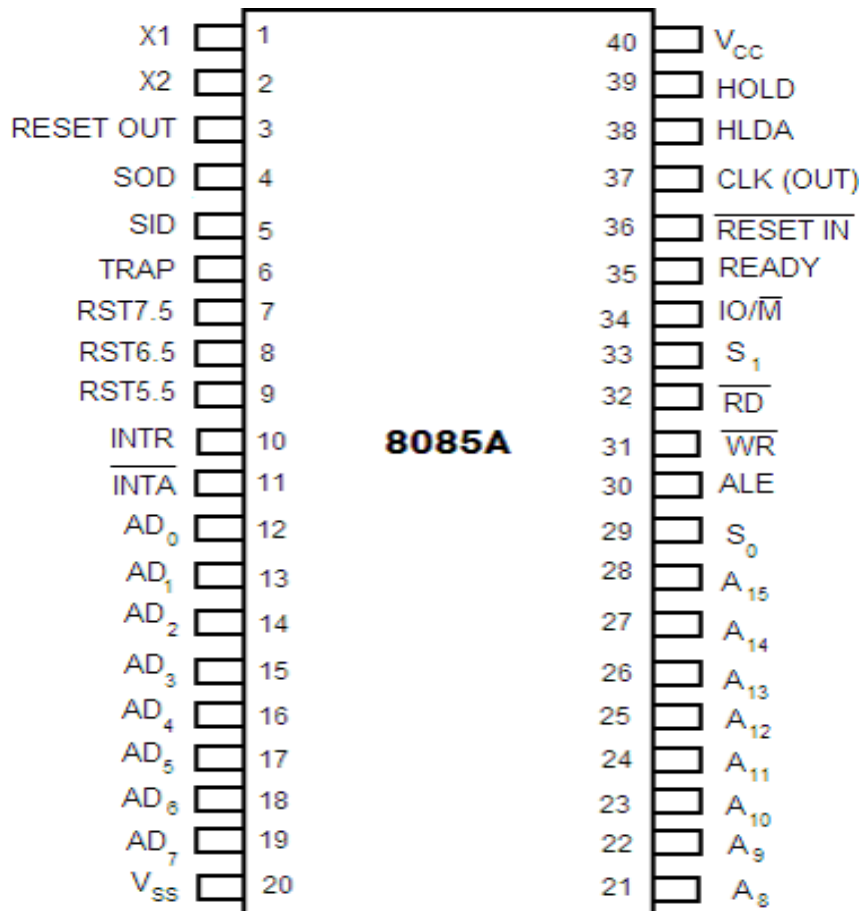
Data Bus

The data bus is a group of eight lines used for dataflow. They are bidirectional: data flows in both direction between the 8085 and memory and peripheral devices. The 8 lines enable the microprocessor to manipulate 8-bit data ranging from 00 to FF.

Control Bus

The control bus consists of various single lines that carry synchronization signals. These are not groups of lines like address of data bus but individual lines that provide a pulse to indicate an operation. The 8085 generates specific control signal for each operation it performs. These signals are used to identify a device type which the processor intends to communicate.

1.3 8085 Pin Diagram



8085 Pin Description

Properties

- f* Single + 5V Supply
- f* 4 Vectored Interrupts (One is Non Maskable)
- f* Serial In/Serial Out Port
- f* Decimal, Binary, and Double Precision Arithmetic
- f* Direct Addressing Capability to 64K bytes of memory

A8-A15 (Output 3 states)

Address Bus carries the most significant 8 bits of the memory address or the 8 bits of the I/O address; 3 stated during Hold and Halt modes.

AD0 - AD 7 (Input/Output 3state)

Multiplexed Address/Data Bus carries Lower 8 bits of the memory address (or I/O address) appear on the bus during the first clock cycle of a machine state. It then becomes the data bus during the second and third clock cycles. 3 stated during Hold and Halt modes.

ALE (Output)

Address Latch Enable occurs during the first clock cycle of a machine state and enables the address to get latched into the on chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. ALE can also be used to strobe the status information. ALE is never 3 stated.

SO, S1 (Output)

Data Bus Status: Encoded status of the bus cycle

S1	S0	
0	0	HALT
0	1	WRITE
1	0	READ
1	1	FETCH

RD (Output 3state)

READ indicates the selected memory or I/O device is to be read and that the Data Bus is available for the data transfer.

WR (Output 3state)

WRITE indicates the data on the Data Bus is to be written into the selected memory

or I/O location. Data is set up at the trailing edge of WR. 3 stated during Hold and Halt modes.

READY (Input)

If Ready is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If Ready is low, the CPU will wait for Ready to go high before completing the read or write cycle.

HOLD (Input)

HOLD indicates that another Master is requesting the use of the address and data buses. The CPU, upon receiving the Hold request, will relinquish the use of buses as soon as the completion of the current machine cycle. Internal processing can continue. The processor can regain the buses only after the Hold is removed. When the Hold is acknowledged, the Address, Data, RD, WR, and IO/M lines are 3stated.

HLDA (Output)

HOLD ACKNOWLEDGE indicates that the CPU has received the Hold request and that it will relinquish the buses in the next clock cycle. HLDA goes low after the Hold request is removed. The CPU takes the buses one half clock cycle after HLDA goes low.

INTR (Input)

INTERRUPT REQUEST is used as a general purpose interrupt. It is sampled only using the next to the last clock cycle of the instruction. If it is active, the Program Counter (PC) will be inhibited from incrementing and an INTA will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.

INTA (Output)

INTERRUPT ACKNOWLEDGE is used instead of (and has the same timing as) RD during the Instruction cycle after an INTR is accepted. It can be used to activate the 8259

Interrupt chip or some other interrupt port.

RST 5.5/ RST 6.5/ RST 7.5

RESTART INTERRUPTS have the same timing as INTR except they cause an internal RESTART to be automatically inserted.

RST 7.5 → Highest Priority

RST 6.5

RST 5.5 → Lowest Priority

The priority of these interrupts is ordered as shown above. These interrupts have a higher priority than the INTR.

TRAP (Input)

Trap interrupt is a non-maskable restart interrupt. It is recognized at the same time as INTR. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt.

RESET IN (Input)

Reset sets the Program Counter to zero and resets the Interrupt Enable and HLDA flipflops. None of the other flags or registers (except the instruction register) are affected. The CPU is held in the reset condition as long as Reset is applied.

RESET OUT (Output)

It indicates that CPU has been reset. It is used as a system RESET. The signal is synchronized to the processor clock.

X1, X2 (Input)

Crystal or R/C network connections to set the internal clock generator X1 can also be an external clock input instead of a crystal. The input frequency is divided by 2 to give the internal operating frequency.

CLK (Output)

Clock Output is used as a system clock when a crystal or R/ C network is used as an input to the CPU. The period of CLK is twice the X1, X2 input period.

IO/M (Output)

IO/M indicates whether the Read/Write is to memory or I/O. It is tristated during Hold and Halt modes.

SID (Input)

Serial input data line: The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.

SOD (output)

Serial output data line: The output SOD is set or reset as specified by the SIM instruction.

Vcc

+5V supply.

Vss

Ground Reference

1.4 8085 Addressing modes

They are mainly classified into four:

- Immediate addressing.
- Register addressing.
- Direct addressing.
- Indirect addressing.

Immediate addressing

Data is present in the instruction. Load the immediate data to the destination provided.

Example: MVI R, data

Register addressing

Data is provided through the registers.

Example: MOV Rd, Rs

Direct addressing

It is used to accept data from outside devices to store in the accumulator or send the data stored in the accumulator to the outside device. Accept the data from the port 00H and store them into the accumulator or Send the data from the accumulator to the port 01H.

Example: IN 00H or OUT 01H

Indirect Addressing

In this mode the Effective Address is calculated by the processor and the contents of the address (and the one following) are used to form a second address. The second address is where the data is stored. Note that this requires several memory accesses; two accesses to retrieve the 16-bit address and a further access (or accesses) to retrieve the data which is to be loaded into the register.

8.5. 8085 Microprocessor Trainer Kit

8.5.1 Introduction

From the 4 bit microprocessor brought out by Intel in 1971, advancement in technology have been made and now 8 bit ,16 bit , 32 bit and 64 bit microprocessors are available and 64 bit and 32 bit microprocessors are dominating the market. From the age of vacuum tubes and transistors, we are now in the age of microprocessors. Due to its adoptability and intelligence, they are used extensively. The trainer kit is a low cost 8085 based training tool developed specifically for learning the operation of today's microprocessor based systems.

8.5.2 Specifications of MPS 85-3:



MPS 85-3

A COST EFFECTIVE AND VERSATILE
MICROPROCESSOR TRAINER



MPS 85-3 is an enhanced version of our popular model MPS 85-2 based on the processor 8085. It is a versatile microprocessor trainer used as an instructional aid in colleges and universities. It is a complete single board microcomputer used for software and hardware development in research institutions and R&D labs.

MAIN FEATURES

- ★ MPS 85-3 operates on **SINGLE +5V** power supply in onboard keypad mode or from host PC through RS-232-C interface.
- ★ In both stand alone and serial modes, user can enter programs, run them at full speed and debug them through breakpoint and single step facilities.
- ★ MPS 85-3 allows program editing through **INSERT, DELETE** and **BLOCK MOVE** commands.
- ★ MPS 85-3 allows direct read/write from/to a specified I/O port through **IN BYTE** and **OUT BYTE** commands.
- ★ It has four onboard ribbon cable connectors for easy expansion and two user defined function keys.
- ★ Onboard battery backup for RAM.
- ★ It supports many types of interfaces which are common for all ESA trainers.
- ★ Built in Text editor, Assembler and Disassembler facilities in serial mode.

ACCESSORIES (OPTIONAL)

- ★ Interface Modules for training purpose : Keyboard, Elevator, Display, ADC with DAC, Dual DAC, Logic Controller, Traffic Lights, 8253 Demo, RTC, Tone Generator, Stepper Motor, Numeric Printer, 8 bit - 16 Channel ADC, 12 bit - 16 Channel ADC, Opto Isolated Input, Opto Isolated Output, LCD interface, Relay Output, DC Motor, RTD interface, etc.
- ★ Power Supply : ESA PS M2 (SMPS) ; +5V@3.0A, ±12V @ 250mA, +30V @ 100mA; required for some of the above interfaces.
- ★ Power adapter : ADP DC5 ; +5v@3.0A (SMPS)
- ★ Study Cards for 8255, 8279, 8259, 8251/8253, etc.
- ★ Cycle Step Interface.
- ★ EPROM Programmer Interface to program 2716 through 27512.
- ★ Audio Tape Interface.

SPECIFICATIONS

CPU :

8085 operated at 3.072 MHz.

MEMORY :

Three 28 pin JEDEC sockets offer 64K Bytes of memory as in the following confign.

ROM : 16K Bytes of monitor program in 27128 EPROM.

ROM / RAM : 16K Bytes user expansion using any EPROM/SRAM.

RAM : 32K Bytes supplied using 62256 CMOS static RAM with battery backup.

PERIPHERALS

8255 : (Two Nos) One 8255 is supplied to give 24 programmable I/O lines. User can have 48 I/O lines by populating an additional 8255 in expansion socket.

8253 : 3 programmable interval timers - Timer 0 is used for implementing single step facility, Timer 1 is used for generating baud clock and Timer 2 is available to the user.

8251 : For serial communication. Supports all standard bauds from 110 to 19200.

8279 : To Control 32 keys keyboard and 6-digit, half-inch size, seven segment LED display.

Bus Expansion :

Fully de-multiplexed and buffered TTL compatible bus signals brought out through two 26 pin ribbon cable connectors for expansion.

INTERFACE SIGNALS :

Parallel I/O : 48 TTL compatible lines (2x8255) brought out through two 26 pin ribbon cable connectors.

Serial I/O : RS-232-C serial interface signals through onboard 9 pin D type female connector.

INTERRUPTS :

All interrupts except TRAP are available to user. If single step function is not required, TRAP can also be used by the user.

POWER SUPPLY REQUIREMENT

+5V DC @ 900 mA Approx.

DIMENSIONS :

LxBxH : 365 x 275 x 55 mm Approx.

Weight : 1300 gms Approx.

Housed in an ABS Plastic moulded cabinet

SCOPE OF SUPPLY

1. MPS 85-3 Trainer.
2. Windows / DOS Driver CD.
3. RS-232-C Cable.
4. User's Manual.
5. Monitor Program Listing.
6. 8085 Assembly Language Reference Card.



8.6. 8085 Instruction Set Summary

Mnemonic	Description	Clock Cycles
MOV r1 r2	Move register to register	4
MOV M r	Move register to memory	7
MOV r M	Move memory to register	7
MVI r	Move immediate register	7
MVI M	Move immediate memory	10
LXI B	Load immediate register Pair B & C	10
LXI D	Load immediate register Pair D & E	10
LXI H	Load immediate register Pair H & L	10
LXI SP	Load immediate stack pointer	10
STAX B	Store A indirect	7
STAX D	Store A indirect	7
LDAX B	Load A indirect	7
LDAX D	Load A indirect	7
STA	Store A direct	13
LDA	Load A direct	13
SHLD	Store H & L direct	16
LHLD	Load H & L direct	16
XCHG	Exchange D & E H & L registers	4
PUSH B	Push register Pair B & C on stack	12
PUSH D	Push register Pair D & E on stack	12
PUSH H	Push register Pair H & L on stack	12
PUSH PSW	Push A and Flags on stack	12

MICROPROCESSORS LAB

POP B	Pop register Pair B & C off stack	10
POP D	Pop register Pair D & E off stack	10
POP H	Pop register Pair H & L off stack	10
POP PSW	Pop A and Flags off stack	10
XTHL	Exchange top of stack H & L	16
SPHL	H & L to stack pointer	6
JUMP		
JMP	Jump unconditional	10
JC	Jump on carry	7/10
JNC	Jump on no carry	7/10
JZ	Jump on zero	7/10
JNZ	Jump on no zero	7/10
JP	Jump on positive	7/10
JM	Jump on minus	7/10
JPE	Jump on parity even	7/10
JPO	Jump on parity odd	7/10
PCHL	H & L to program counter	6
CALL		
CALL	Call unconditional	18
CC	Call on carry	9/18
CNC	Call on no carry	9/18
CZ	Call on zero	9/18
CNZ	Call on no zero	9/18
CP	Call on positive	9/18

MICROPROCESSORS LAB

CM	Call on minus	9/18
CPE	Call on parity even	9/18
CPO	Call on parity odd	9/18
RET	Return	10
RC	Return on carry	6/12
RNC	Return on no carry	6/12
RZ	Return on zero	6/12
RNZ	Return on no zero	6/12
RP	Return on positive	6/12
RM	Return on minus	6/12
RPE	Return on parity even	6/12
RPO	Return on parity odd	6/12
RST	Restart	12
IN	Input	10
OUT	Output	10
INR r	Increment register	4
DCR r	Decrement register	4
INR M	Increment memory	10
DCR M	Decrement memory	10
INX B	Increment B & C registers	6
INX D	Increment D & E registers	6
INX H	Increment H & L registers	6
INX SP	Increment stack pointer	6

MICROPROCESSORS LAB

DCX B	Decrement B & C	6
DCX D	Decrement D & E	6
DCX H	Decrement H & L	6
DCX SP	Decrement stack pointer	6
ADD r	Add register to A	4
ADC r	Add register to A with carry	4
ADD M	Add memory to A	7
ADC M	Add memory to A with carry	7
ADI	Add immediate to A	7
ACI	Add immediate to A with carry	7
DAD B	Add B & C to H & L	10
DAD D	Add D & E to H & L	10
DAD H	Add H & L to H & L	10
DAD SP	Add stack pointer to H & L	10
SUB r	Subtract register from A	4
SBB r	Subtract register from A with borrow	4
SUB M	Subtract memory from A	7
SBB M	Subtract memory from A with borrow	7
SUI	Subtract immediate from A	7
SBI	Subtract immediate from A with borrow	7

MICROPROCESSORS LAB

ANA r	And register with A	4
XRA r	Exclusive Or register with A	4
ORA r	Or register with A	4
CMP r	Compare register with A	4
ANA M	And memory with A	7
XRA M	Exclusive Or Memory with A	7
ORA M	Or memory with A	7
CMP M	Compare memory with A	7
ANI	And immediate with A	7
XRI	Exclusive Or immediate with A	7
ORI	Or immediate with A	7
CPI	Compare immediate with A	7
RLC	Rotate A left	4
RRC	Rotate A right	4
RAL	Rotate A left through carry	4
RAR	Rotate A right through carry	4
CMA	Complement A	4
STC	Set carry	4
CMC	Complement carry	4
DAA	Decimal adjust A	4
EI	Enable Interrupts	4
DI	Disable Interrupts	4

NOP	No-operation	4
HLT	Halt (Power down)	5
RIM	Read Interrupt Mask	4
SIM	Set Interrupt Mask	4

8.7. Procedure to Enter & Execute a Program

Execution Procedure

- Assembly language programs for Microprocessor 8085 can be executed on its trainer kit as per the following steps:

Step 1: Storage of program machine codes.

Machine codes of the program to be executed, should be stored in the memory available on 8085 trainer kit as

RESET → EXAM. MEM. → starting address of program (2000)
 → NEXT → Now enter all the machine codes, one after the other followed by the key 'NEXT'.

Step 2: Declaration of inputs.

Input values to the program should be declared as
 RESET → EXAM. MEM. → input address → NEXT → Now enter all the input values followed by the key 'NEXT'.

Step 3: Execution of program.

The program can be executed as
 RESET → GO → starting address of program (8000) → EXEC.

Step 4: Verification of Output.

Output of the executed program can be verified as.
 RESET → EXAM. MEM. → Output address → NEXT.

8.8. A Sample Program

Aim: To multiply two 8 bit numbers.

Program Analysis: Two 8 bit numbers are stored in memory locations 8100 and 8101. They are multiplied and the results are stored in memory locations 8200 and 8201.

Program:

Memory address	Machine code	Label	Opcode	Operand	Comments
8000	AF		XRA A		Clear A
8001	A8		XRA B		Clear B
8002	A9		XRA C		Clear C
8003	21		LXI H	8100	Set HL pair as an index to source memory
8004	00				
8005	81				
8006	46		MOV B, M		Move [M] to B
8007	23		INX H		Increment HL pair
8008	86	L2	ADD M		Add [A] to [M]
8009	D2		JNC	L1	Jump if no carry to L1
800A	0D				
800B	80				
800C	0C		INR C		Increment [C]
800D	05	L1	DCR B		Decrement [B]
800E	C2		JNZ	L2	Jump if nonzero to L2
800F	08				
8010	80				
8011	32		STA	8200	Store [A] in 8200
8012	00				

8013	82				
8014	79		MOV A, C		Move [C] to A
8015	32		STA	8201	Store [A] in memory location 8201
8016	01				
8017	82				
8018	76		HLT		Stop program

Result: The program is executed and the results are stored in the memory locations 8200 and 8201.

Input: At 8100 : 03

At 8101 : 02

Output: At 8200 : 06

At 8201 : 00

8.9. 8085 Instructions & Mnemonic Codes

Hex	mnemonic		Hex	mnemonic		Hex	mnemonic		Hex	mnemonic	
CE	ACI	8-Bit	3F	CMC		2B	DCX	H	01	LXI B,	16-Bit
8F	ADC	A	BF	CMP A		3B	DCX	SP	11	LXI D,	16-Bit
88	ADC	B	B8	CMP B		F3	DI		21	LXI H,	16-Bit
89	ADC	C	B9	CMP C		FB	EI		31	LXI SP,	16-Bit
8A	ADC	D	BA	CMP D		76	HLT		7F	MOV A A	
8B	ADC	E	BB	CMP E		DB	IN	8-Bit	78	MOV A B	
8C	ADC	H	BC	CMP H		3C	INR	A	79	MOV A C	
8D	ADC	L	BD	CMP		04	INR	B	7A	MOV A D	
8E	ADC	M	BE	CMP M		0C	INR	C	7B	MOV A E	
87	ADD	A	D4	CNC	16-Bit	14	INR	D	7C	MOV A H	
80	ADD	B	C4	CNZ	16-Bit	1C	INR	E	7D	MOV A L	

MICROPROCESSORS LAB

81	ADD C	F4	CP 16-Bit	24	INR H	7E	MOV A M
82	ADD D	EC	CPE 16-Bit	2C	INR L	47	MOV B A
83	ADD E	FE	CPI 8-Bit	34	INR M	40	MOV B B
84	ADD H	E4	CPO 16-Bit	03	INX B	41	MOV B C
85	ADD L	CC	CZ 16-Bit	13	INX D	42	MOV B D
86	ADD M	27	DAA	23	INX H	43	MOV B E
C6	ADI 8-Bit	09	DAD B	33	INX SP	44	MOV B H
A7	ANA A	19	DAD D	DA	JC 16-Bit	45	MOV B L
A0	ANA B	29	DAD H	FA	JM 16-Bit	46	MOV B M
A1	ANA C	39	DAD SP	C3	JMP 16-Bit	4F	MOV C A
A2	ANA D	3D	DCR A	D2	JNC 16-Bit	48	MOV C B
A3	ANA E	05	DCR B	C2	JNC 16-Bit	49	MOV C C
A4	ANA H	0D	DCR C	F2	JP 16-Bit	4A	MOV C D
A5	ANA L	15	DCR D	EA	JPE 16-Bit	4B	MOV C E
A6	ANA M	1D	DCR E	E2	JPO 16-Bit	4C	MOV C H
E6	ANA 8-Bit	25	DCR H	CA	JZ 16-Bit	4D	MOV C L
CD	CALL 16-Bit	2D	DCR L	3A	LDA 16-Bit	4E	MOV C M
DC	CC 16-Bit	35	DCR M	0A	LDAX B	57	MOV D A
FC	CM 16-Bit	0B	DCX B	1A	LDAX D	50	MOV D B
2F	CMA	1B	DCX D	2A	LHLD 16-Bit	51	MOV D C

Hex	mnemonic	Hex	mnemonic	Hex	mnemonic	Hex	mnemonic
52	MOV D D	71	MOV M C	E5	PUSH H	9E	SBB M
53	MOV D E	72	MOV M D	F5	PUSH PSW	DE	SBI 8-Bit
54	MOV D H	73	MOV M E	17	RAL	22	SHLD 16-Bit
55	MOV D L	74	MOV M H	1F	RAR	30	SIM
56	MOV D M	75	MOV M L	D8	RC	F9	SPHL

MICROPROCESSORS LAB

5F	MOV E A	3E	MVI A 8-Bit	C9	RET	32	STA	16-Bit
58	MOV E B	06	MVI B 8-Bit	20	RIM	02	STAX B	
59	MOV E C	0E	MVI C 8-Bit	07	RLC	12	STAX D	
5A	MOV E D	16	MVI D 8-Bit	F8	RM	37	STC	
5B	MOV E E	1E	MOV E 8-Bit	D0	RNC	97	SUB A	
5C	MOV E H	26	MVI H 8-Bit	C0	RNC	90	SUB B	
5D	MOV E L	2E	MVI L 8-Bit	F0	RP	91	SUB C	
5E	MOV E M	36	MVI M 8-Bit	E8	RPE	92	SUB D	
67	MOV H A	00	NOP	E0	RPO	93	SUB E	
60	MOV H B	B7	ORA A	0F	RRC	94	SUB H	
61	MOV H C	B0	ORA B	C7	RST 0	95	SUB L	
62	MOV H D	B1	ORA C	CF	RST 1	96	SUB M	
63	MOV H E	B2	ORA D	D7	RST 2	D6	SUI	16-Bit
64	MOV H H	B3	ORA E	DF	RST 3	EB	XCHG	
65	MOV H L	B4	ORA H	E7	RST 4	AF	XRA A	
66	MOV H M	B5	ORA L	EF	RST 5	A8	XRA B	
6F	MOV L A	B6	ORA M	F7	RST 6	A9	XRA C	
68	MOV L B	F6	ORI 8-Bit	FF	RST 7	AA	XRA D	
69	MOV L C	D3	OUT 8-Bit	C8	RZ	AB	XRA E	
6A	MOV L D	E9	PCH L	9F	SBB A	AC	XRA H	
6B	MOV L E	C1	POP B	98	SBB B	AD	XRA L	
6C	MOV L H	D1	POP D	99	SBB C	AE	XRA M	
6D	MOV L L	E1	POP H	9A	SBB D	EE	XRI	8-Bit
6E	MOV L M	F1	POP PSW	9B	SBB E	E3	XTHL	
77	MOV M A	C5	PUSH B	9C	SBB H			
70	MOV M B	D5	PUSH D	9D	SBB L			

GENERAL GUIDELINES AND SAFETY INSTRUCTIONS

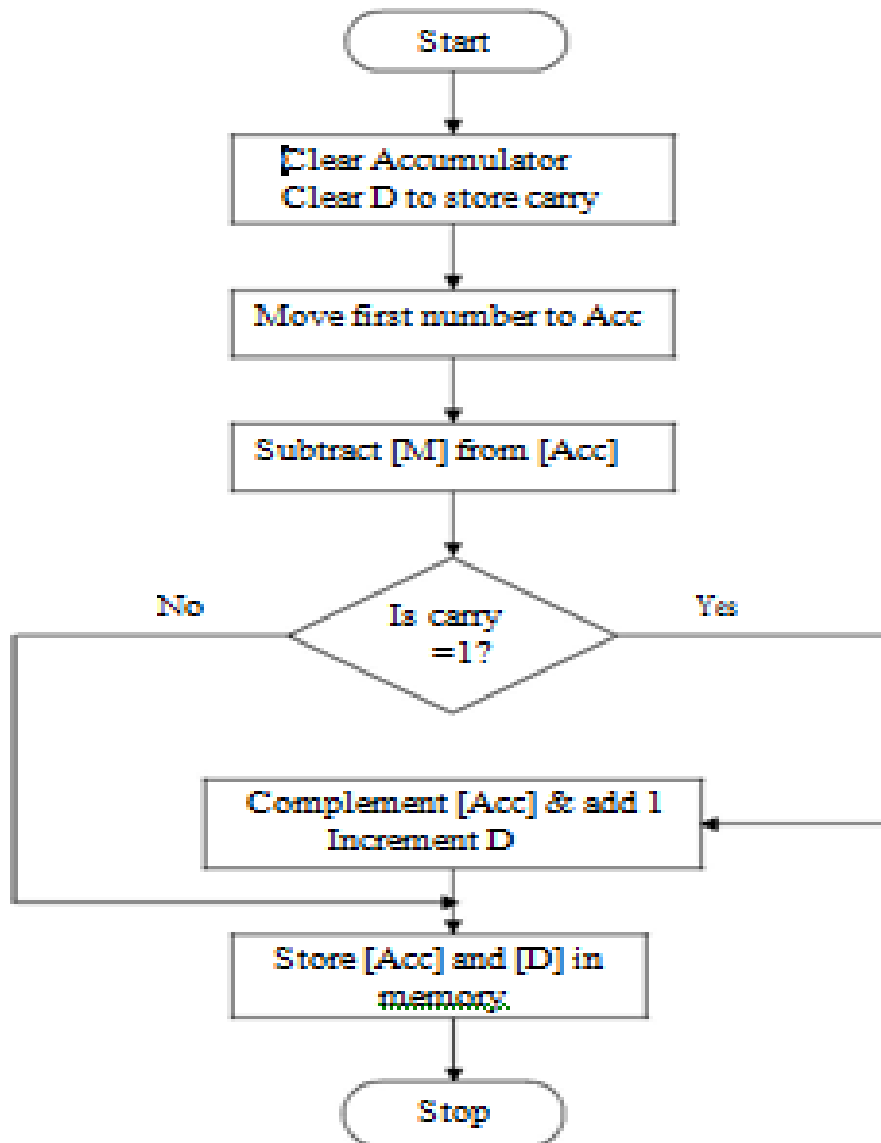
1. Sign in the log register as soon as you enter the lab and strictly observe your lab timings.
2. Strictly follow the written and verbal instructions given by the teacher / Lab Instructor. If you do not understand the instructions, the handouts and the procedures, ask the instructor or teacher.
3. It is mandatory to come to lab in a formal dress and wear your ID cards.
4. Do not wear loose-fitting clothing or jewellery in the lab. Rings and necklaces are usual excellent conductors of electricity.
5. Mobile phones should be switched off in the lab. Keep bags in the bag rack.
6. Keep the labs clean at all times, no food and drinks allowed inside the lab.
7. Intentional misconduct will lead to expulsion from the lab.
8. Do not insert connectors forcefully into the sockets.
9. **NEVER** try to experiment with the power from the wall plug.
10. Immediately report dangerous or exceptional conditions to the Lab instructor / teacher: Equipment that is not working as expected, wires or connectors are broken, the equipment that smells or “smokes”. If you are not sure what the problem is or what's going on, switch off the Emergency shutdown.
11. Never use damaged instruments, wires or connectors. Hand over these parts to the Lab instructor/Teacher.
12. Be sure of location of fire extinguishers and first aid kits in the laboratory.
13. After verification of program output, turn off power supply to the trainer kit. Do not take any item from the lab without permission.
14. Observation book and lab record should be carried to each lab. Programs of current lab session are to be written in Observation book and of previous lab session should be written in Lab record book. Both the books should be corrected by the faculty in each lab.
15. Handling of Microprocessor trainer kit: Sensitive electronic circuits and electronic components have to be handled with great care. The inappropriate handling of electronic component can damage or destroy the devices. Therefore, always handle the electronic devices as indicated by the handout, the specifications in the data sheet or other documentation.

Program 1: 8-BIT SUBTRACTION

Aim: To subtract two 8-bit numbers.

Method: The numbers to be subtracted are stored in memory locations. First number is brought to accumulator and the second number in the memory is subtracted from it. If a carry is generated, the result stored in the accumulator is complemented and a 1 is added to it. Finally, the result and carry are stored in memory locations.

Flowchart:

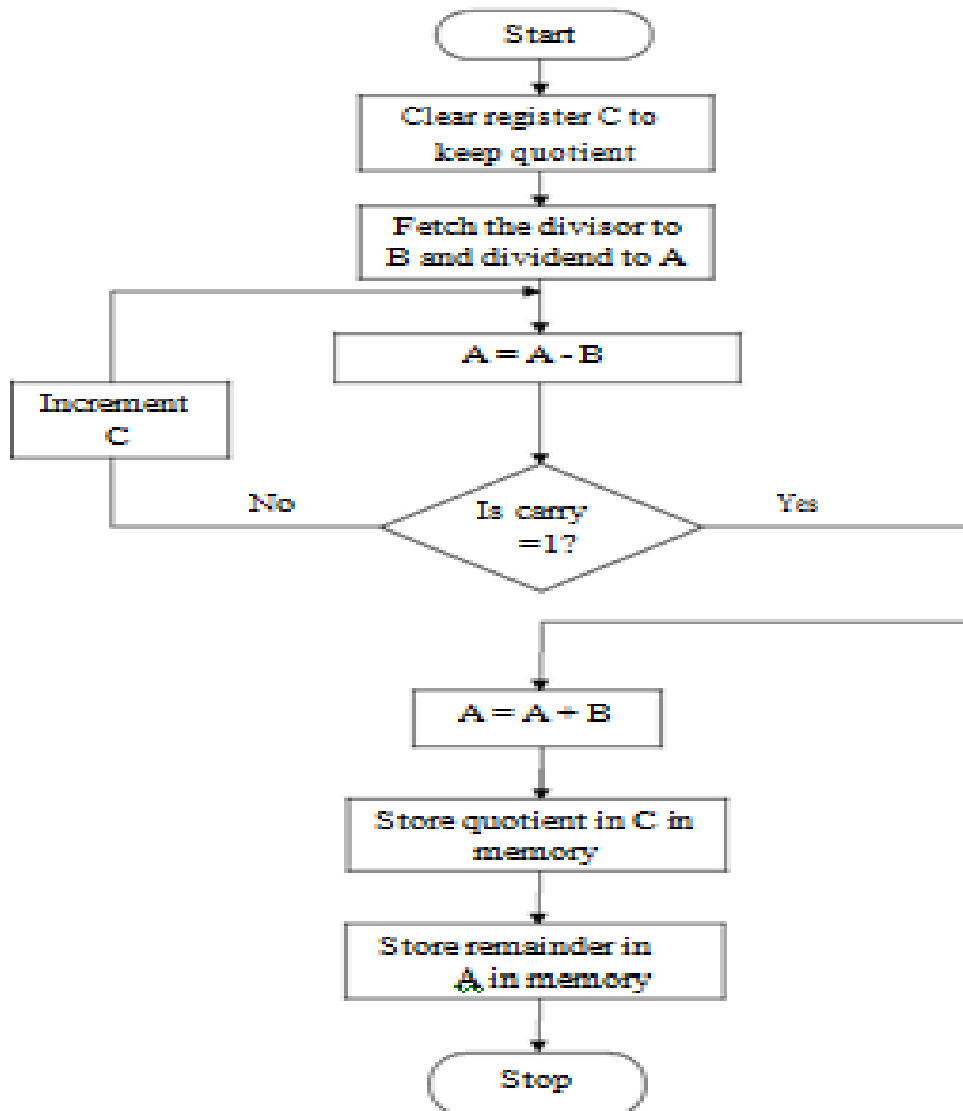


Program 2: 8-BIT DIVISION

Aim: To divide two 8-bit numbers.

Method: The numbers to be divided are stored in memory locations. The dividend is moved to accumulator. The divisor is subtracted from the accumulator content until a carry is generated. The number of times this subtraction is done will give the quotient and the remaining value in the accumulator will give the remainder of division.

Flowchart:

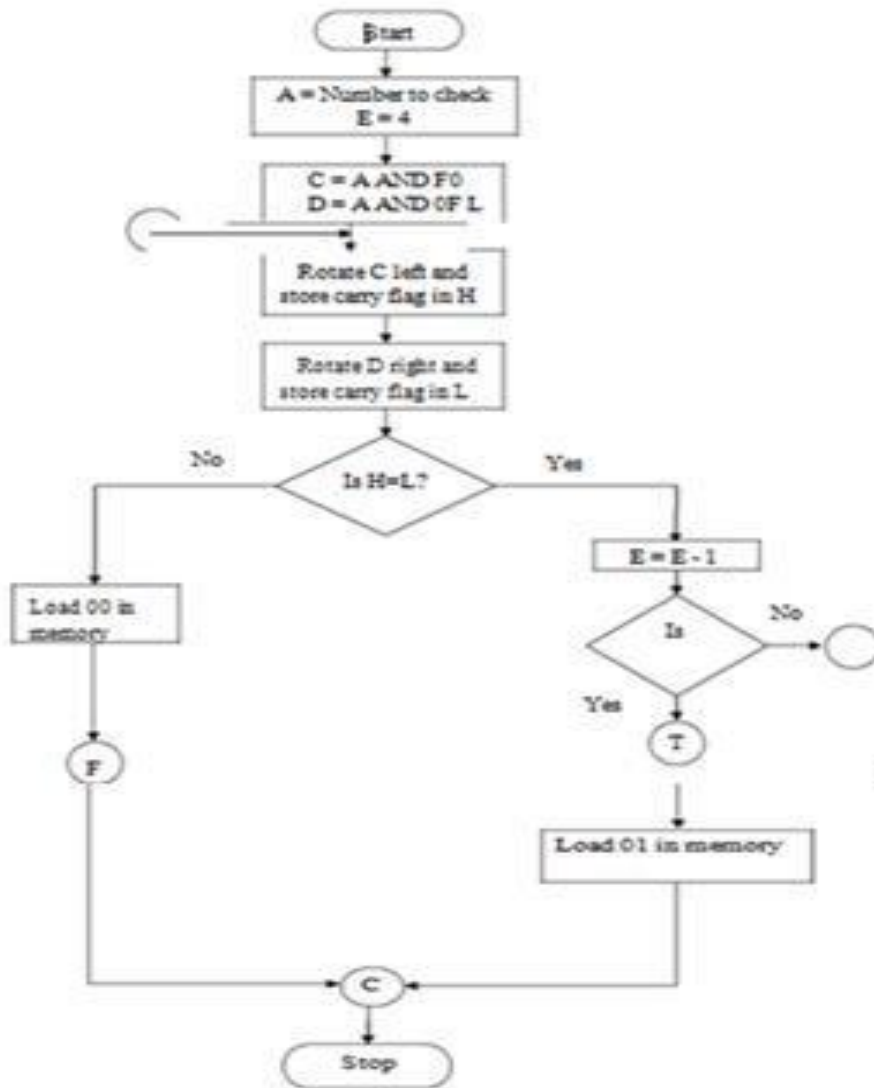


Program 3: CHECKING WHETHER A NUMBER IS PALINDROME

Aim: To check whether the given number is a palindrome or not.

Method: The number to be checked is stored in a memory location. It is fetched to a register and the first and last nibbles are separated. The first nibble is rotated left and the carry flag is checked. The last nibble is rotated right and the carry flag is again checked. If the carry flags of these two operations do not yield the same value, 00 is stored in memory location indicating that the number is not a palindrome and the program comes to a halt. But if, they yield the same result the process is repeated 4 times. If it completes 4 iterations successfully i.e. the carry flags for each nibble in an iteration remain the same, 01 is stored in memory location indicating that the number is a palindrome.

Flowchart:

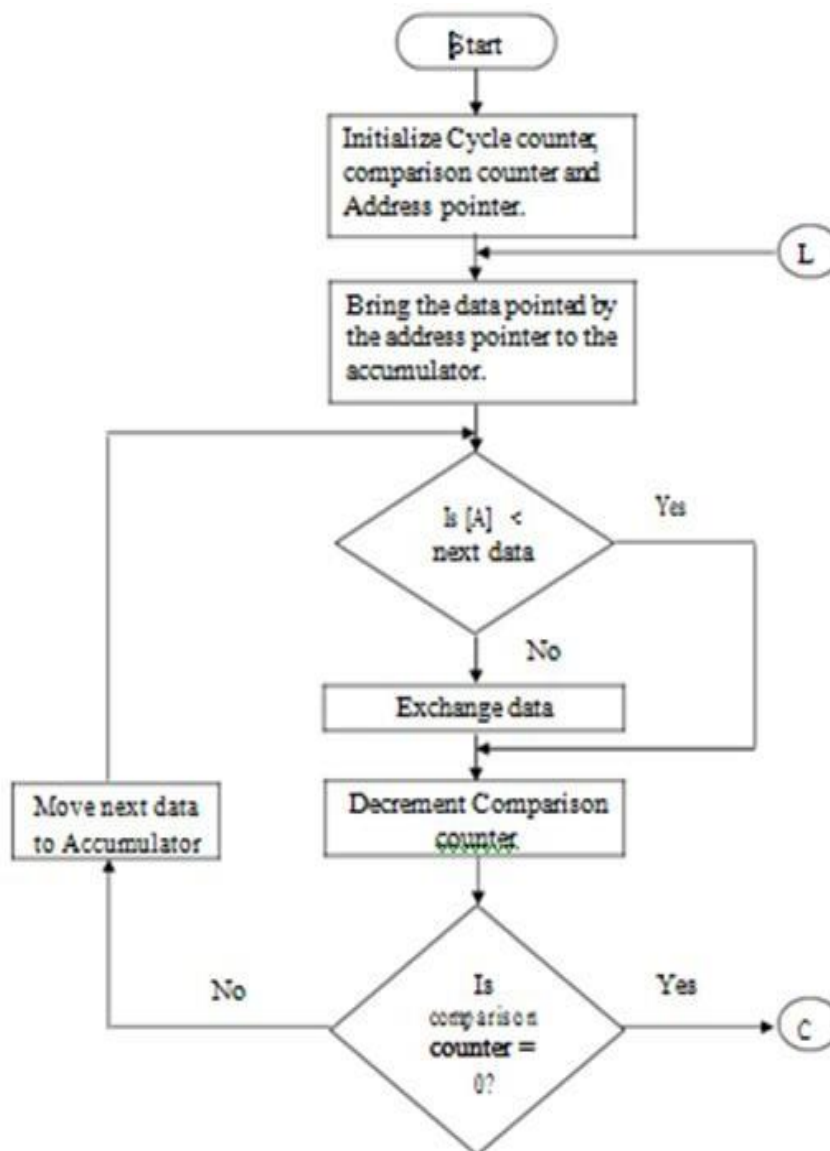


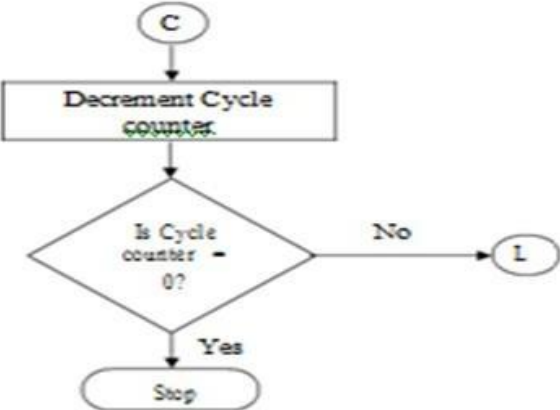
Program 4: SORTING NUMBERS IN ASCENDING ORDER

Aim: To sort 10 numbers stored in consecutive memory locations in ascending order.

Method: Initialize cycle counter, comparison counter with corresponding values and the address pointer to the location where the data is stored. Move the data pointed by the address pointer to the accumulator. Compare it with next data. If the accumulator content is less than the next data then exchange the data. Decrement comparison counter. Repeat the process with the next data until comparison counter is 0. If the comparison counter is zero then decrement cycle counter and if it is not zero increment the address pointer and repeat the whole process until cycle counter is zero.

Flowchart:



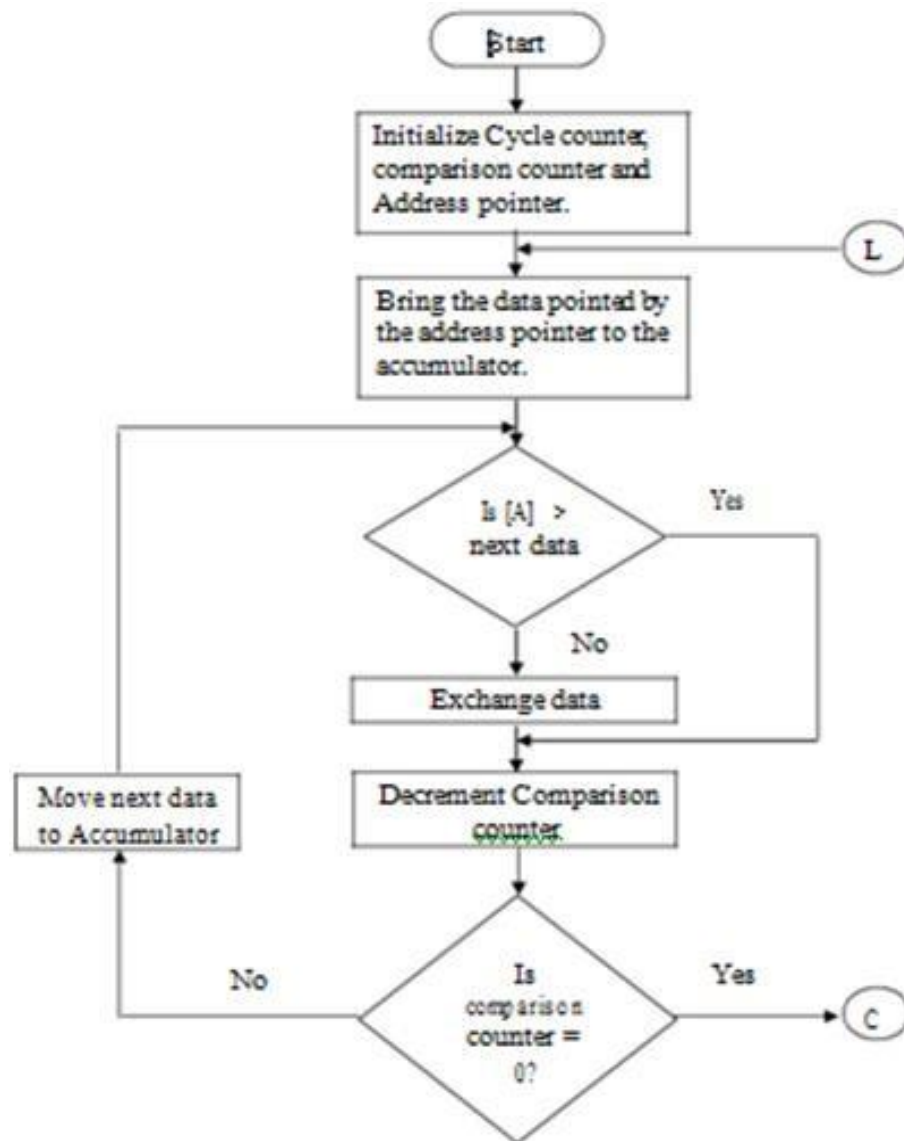


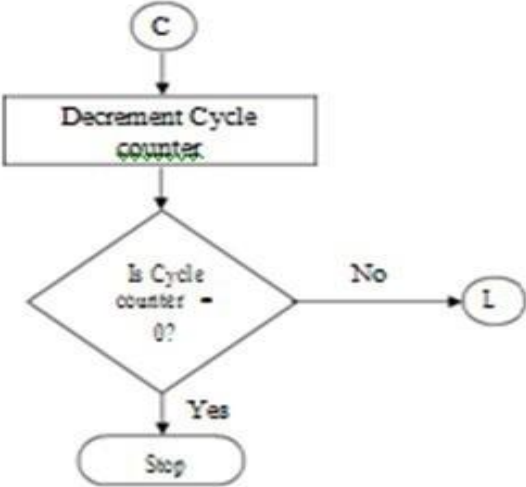
Program 5: SORTING NUMBERS IN DESCENDING ORDER

Aim: To sort 10 numbers stored in consecutive memory locations in descending order.

Method: Initialize cycle counter, comparison counter with corresponding values and the address pointer to the location where the data is stored. Move the data pointed by the address pointer to the accumulator. Compare it with next data. If the accumulator content is larger than the next data then exchange the data. Decrement comparison counter. Repeat the process with the next data until comparison counter is 0. If the comparison counter is zero then decrement cycle counter and if it is not zero increment the address pointer and repeat the whole process until cycle counter is zero.

Flowchart:



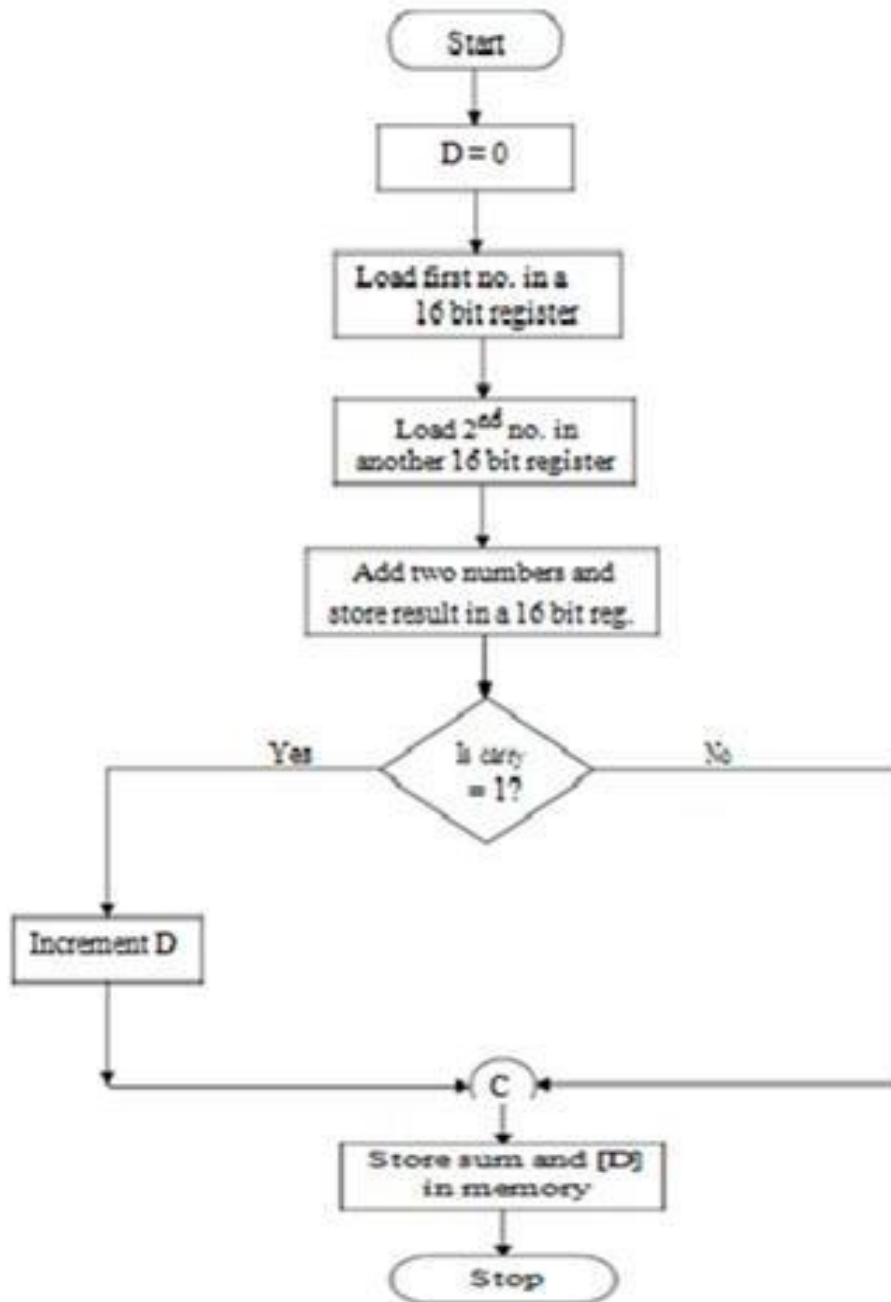


Program 6: 16-BIT ADDITION

Aim: To add two 16 bit numbers.

Method: The numbers to be added are stored in two 16 bit registers. They are added and the resultant sum and carry are stored in memory locations.

Flowchart:

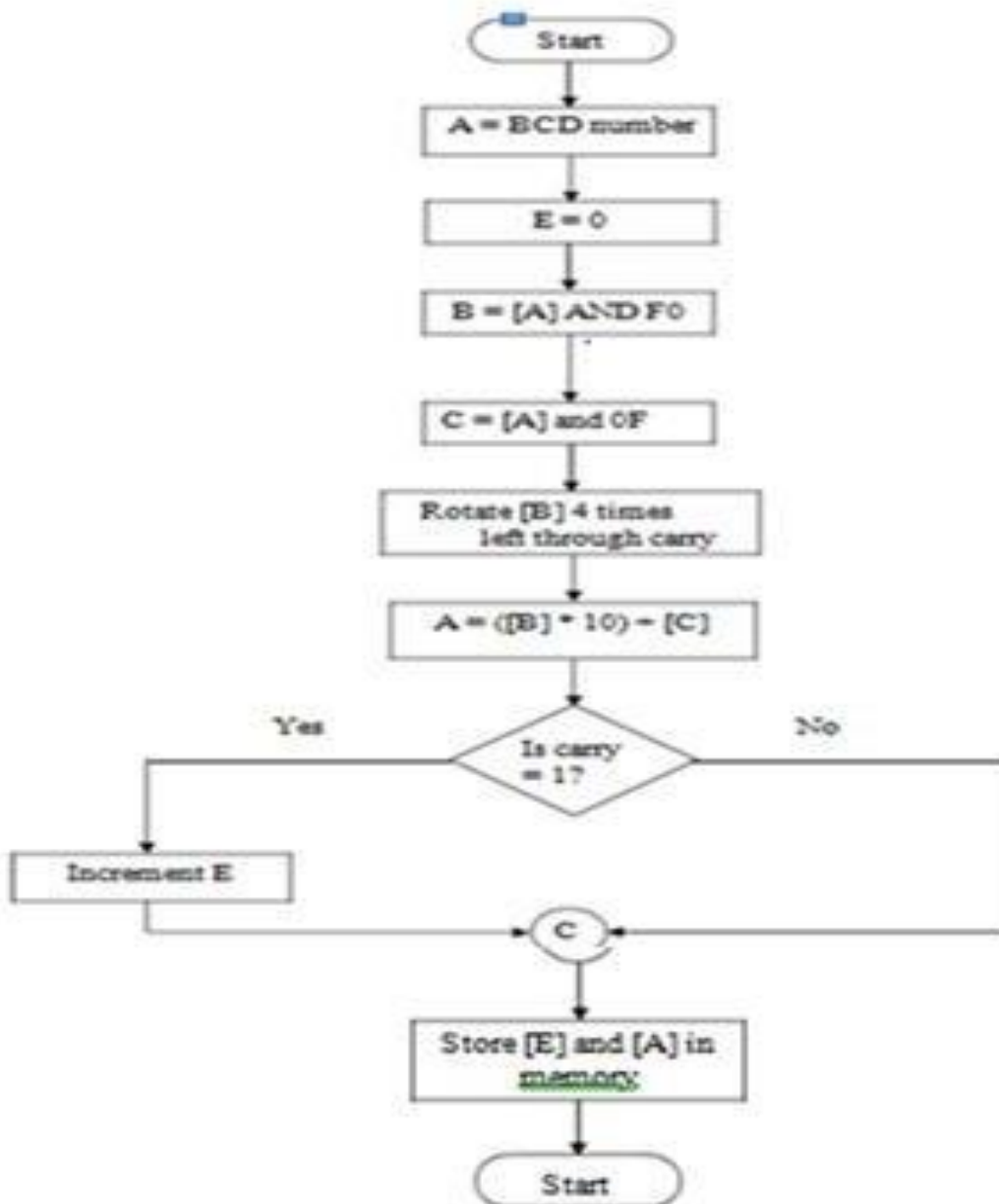


Program 7: CONVERTING BCD NUMBER TO BINARY

Aim: To convert a BCD number to a binary number.

Method: The number is ANDed with F0 to obtain the first 4 bits. Then it is rotated 4 times left through carry and the value is stored in a register (say B). The last 4 bits obtained when the BCD number ANDed with 0F is stored in another register (say C). The value in B is multiplied by 10 and then it is added with the contents of C to obtain the equivalent binary number. The carry, if any is also stored in some registers.

Flowchart:

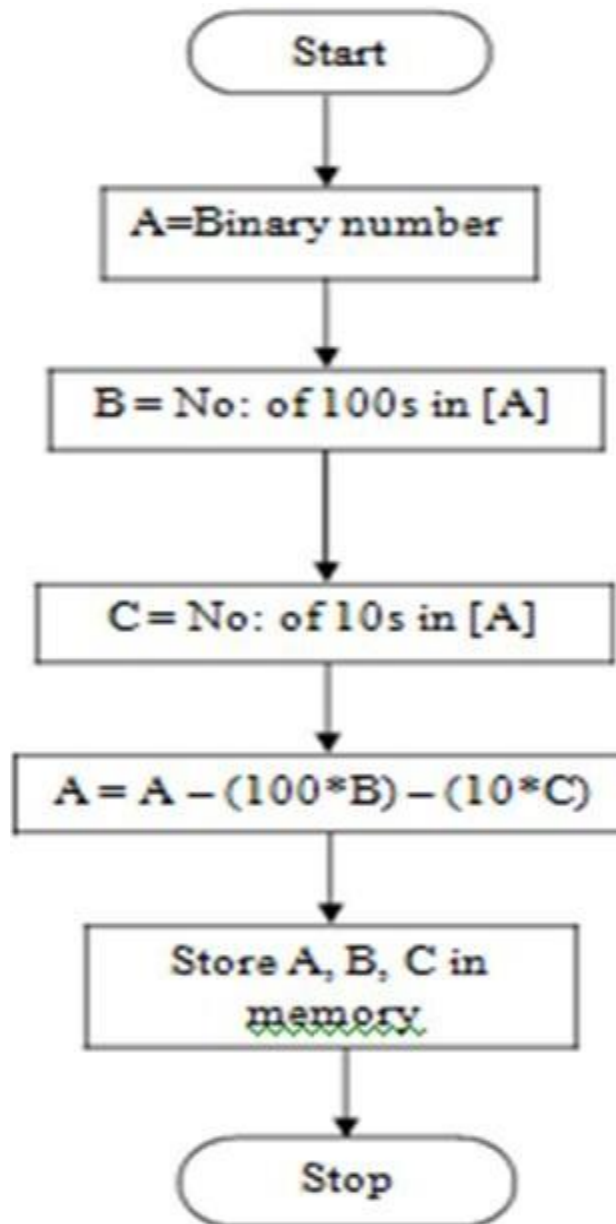


Program 8: CONVERTING BINARY NUMBER TO BCD

Aim: To convert a binary number to BCD number.

Method: The binary number is stored in a register. Count the number of 100s and store it in a register say A. Count the number of 10s in it and store it in a register say B. Subtract all 100s, 10s from the original binary number and the resulting value is stored in another register. These 3 values stored will give the equivalent BCD number.

Flowchart:

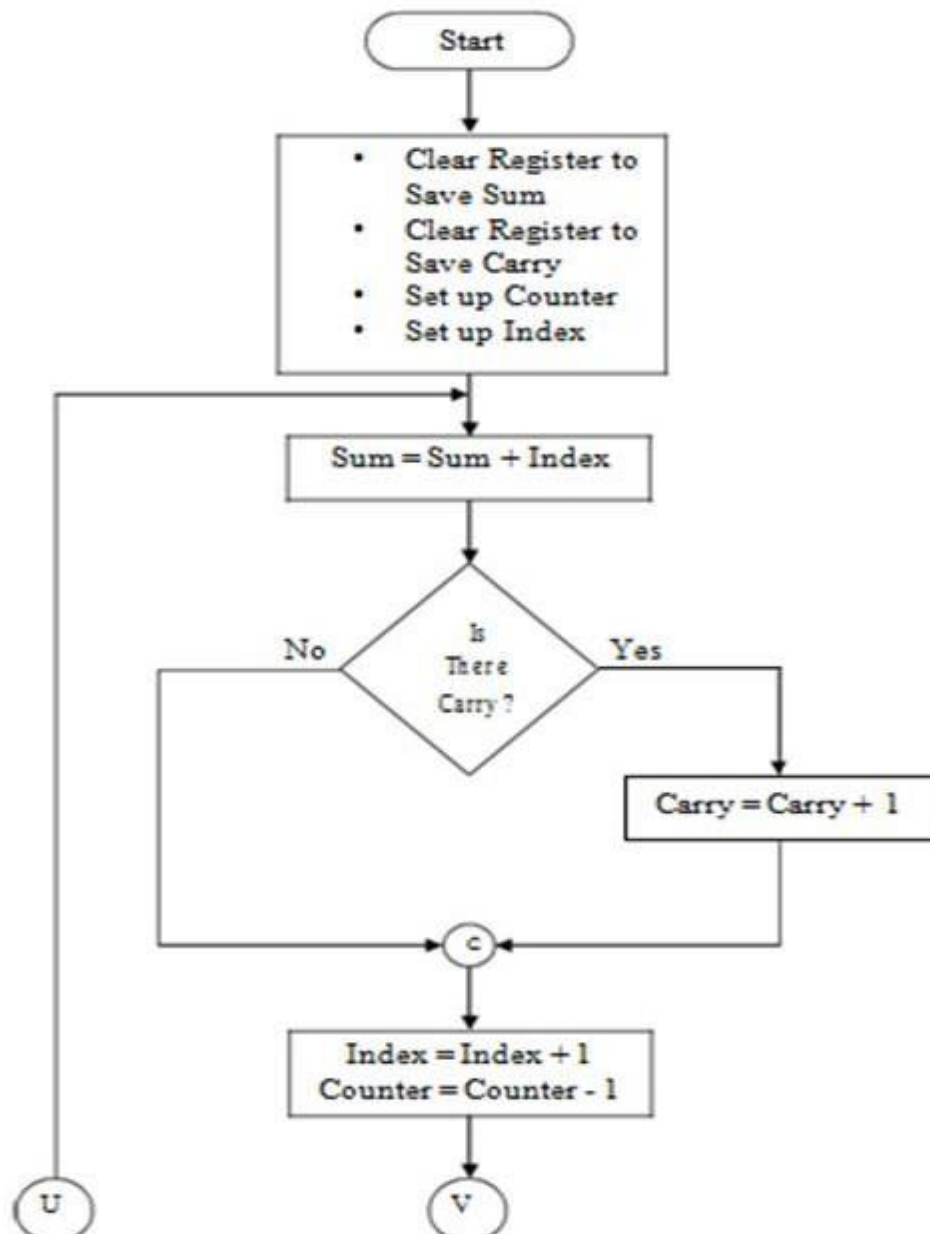


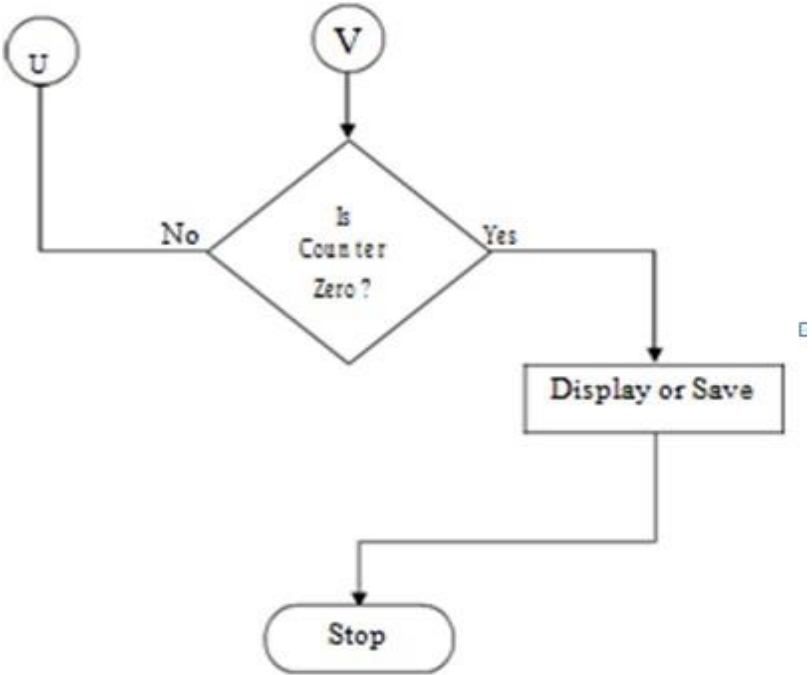
Program 9: ADDITION OF SERIES OF NUMBERS

Aim: To add ten 8 bit numbers.

Method: Move first data to accumulator. Initialize count register. Add the next data with data in the accumulator. If there is a carry increments carry register. Decrement the count register. If it is zero store the result. Else fetch the next data and add with value in the accumulator. Repeat until carry register is zero.

Flowchart:



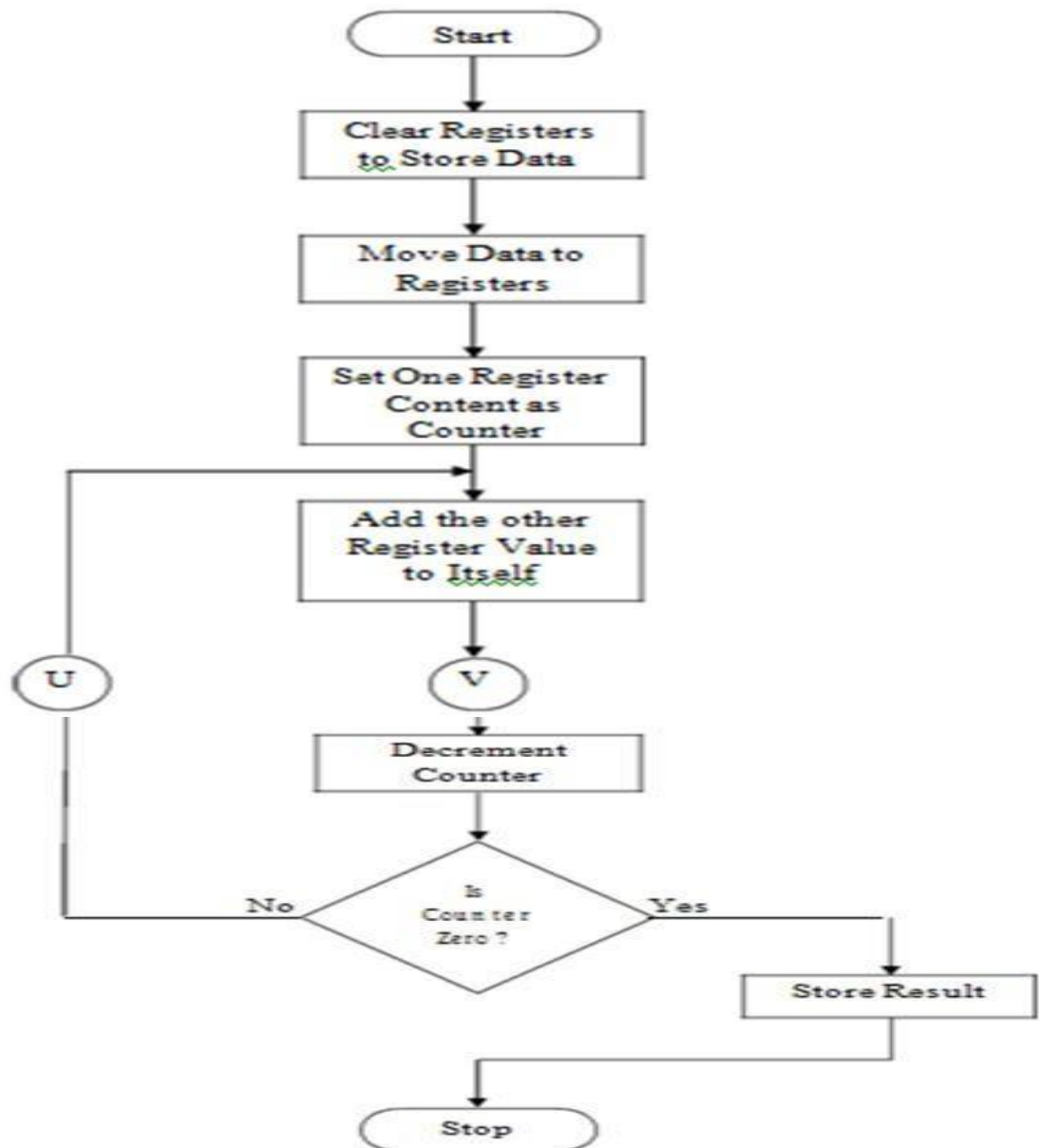


Program 10: 8-BIT MULTIPLICATION

Aim: To multiply two 8 bit numbers.

Method: Store one of the data in a register (say C register). Move the second data to accumulator. Move the accumulator content to another register (say B register). Set the data in the C register as a counter. Add the data in B register to the content of accumulator. Decrement the value in C register. Repeat the addition until the value in the counter register C is zero. The final value in the accumulator will be the product of the two values.

Flowchart:

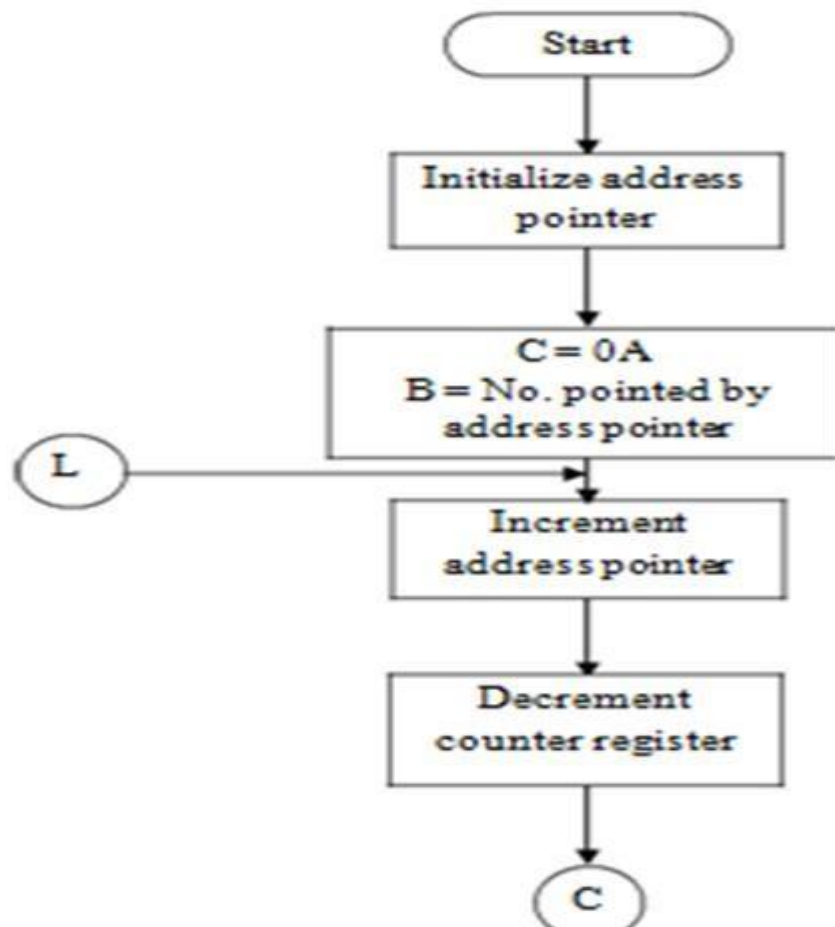


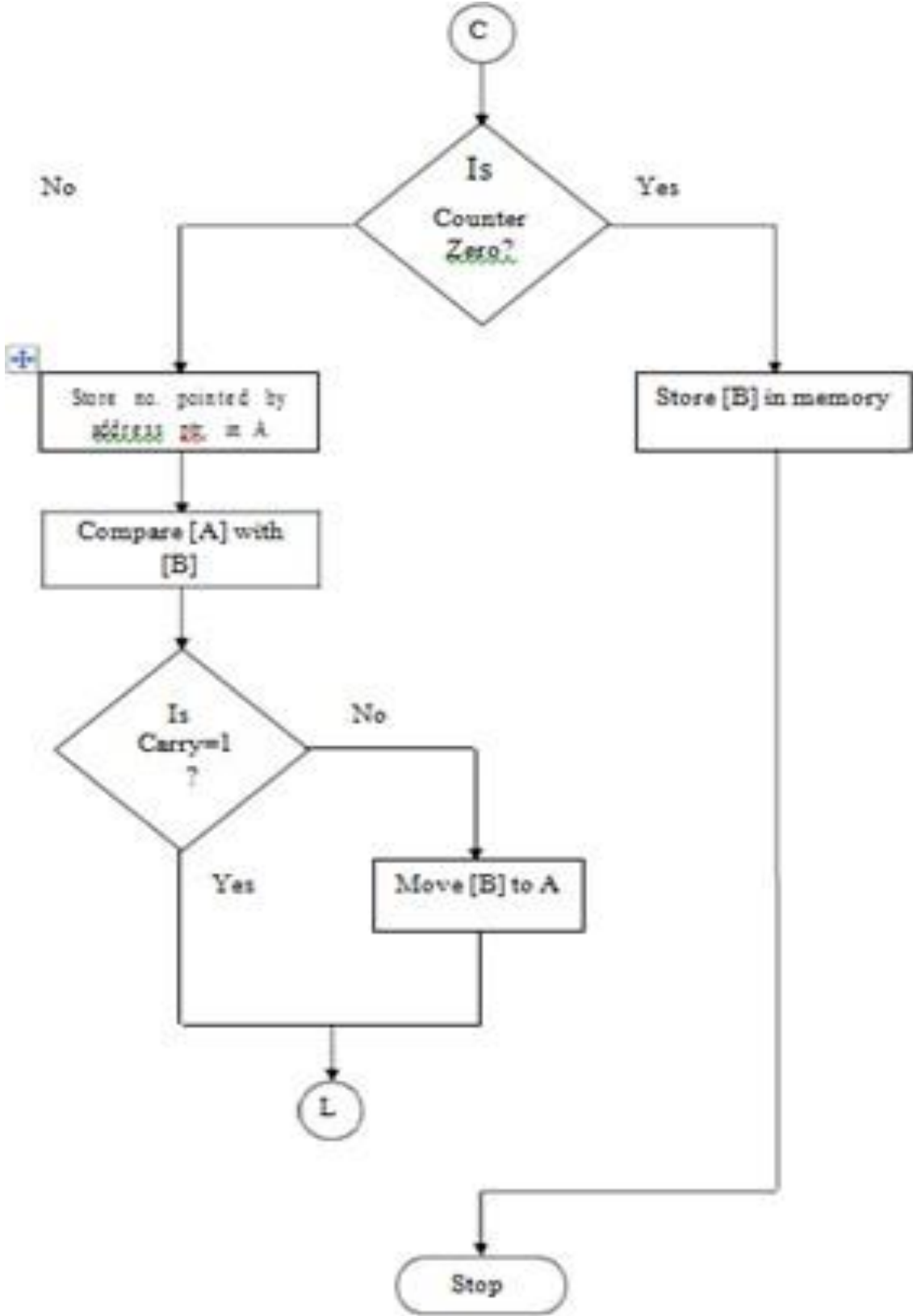
Program 11: LARGEST NUMBER IN A LIST

Aim: To find out the largest of ten 8 bit numbers.

Method: The numbers are stored in consecutive memory locations. The counter register is initialized with 0A and the address pointer points to the first number. The first number is moved to a register say B. The address pointer is incremented and counter register is decremented and the next number is fetched to accumulator. If the content of accumulator is greater than that in B, it is loaded in B. The counter register is decremented and the process is repeated until the counter register reaches to 0. The final value in B will give the largest number in the series.

Flowchart:





Program 12: STEPPER MOTOR

Aim: To rotate stepper motor in forward and reverse direction.

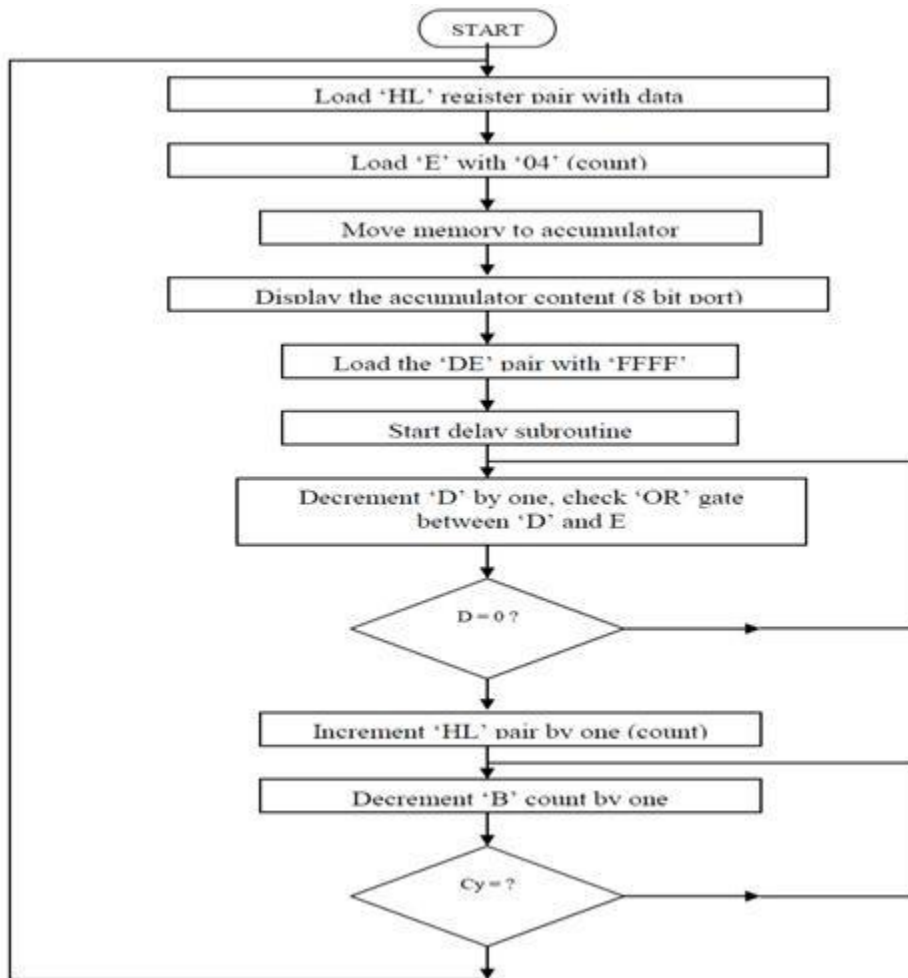
Apparatus required:

Stepper motor, 8085 microprocessor kit, (0-5V) power supply.

Algorithm:

- Step 1: Load the 'HL' pair with value from table
- Step 2: Move it to 'B' register for setting the counter
- Step 3: Move the memory value to accumulator and display it by control word
- Step 4: Load 'DE' register pair with FFFF for starting delay subroutine
- Step 5: Run the delay loop control D-register becomes zero.
- Step 6: Increment 'H' address for next value from table
- Step 7: Jump on no zero
- Step 8: When B = 0, go to start and restart the program.

Flowchart:



Program 13: TRAFFIC LIGHT CONTROL

Aim: To control the traffic light system using 8085 and 8255.

Apparatus required:

8085 Microprocessor trainer kit, Traffic light interface board, Regulated Power supply.

Description: The traffic system moves from one state to next state. By changing the delay between two signals one can change the speed of traffic.

8255 port addresses:

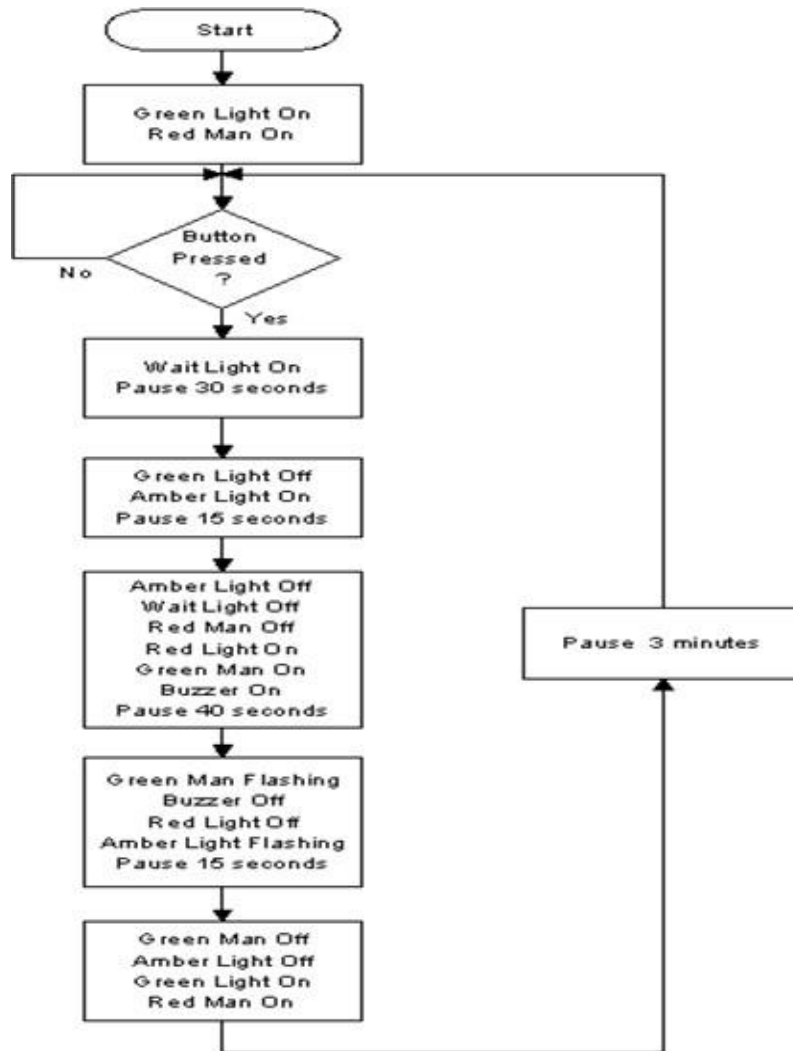
PORT A - 40H

PORT B - 41H

PORT C - 42H

Control word register -43H

Flowchart:



Program 14: DISPLAYING TEXT ON LCD

Aim: To display text on LCD using 8085 and 8255.

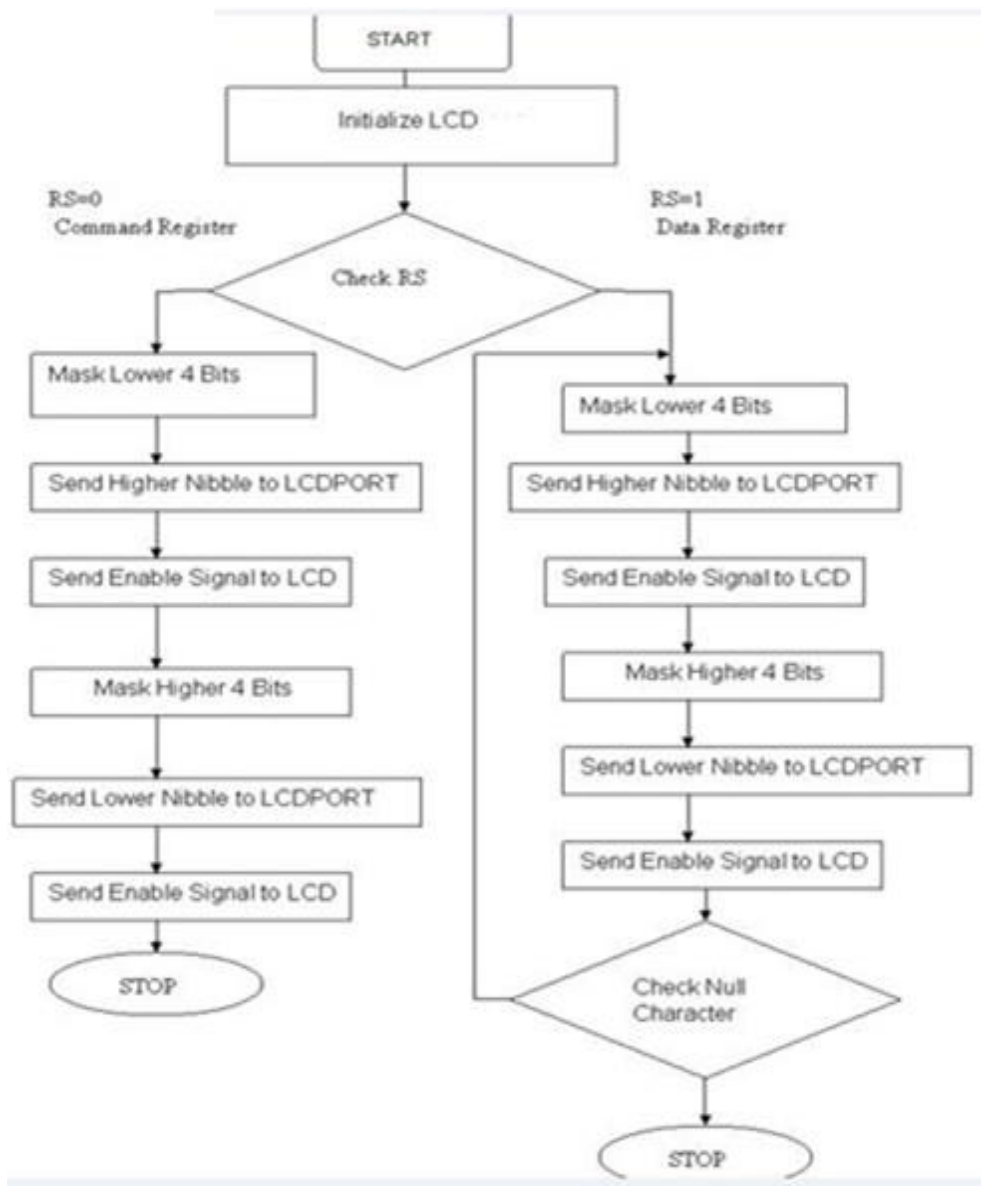
Apparatus required:

8085 Microprocessor trainer kit, LCD interface board, Regulated Power supply.

Description: LCD interface is connected over J2 of the trainer. When the trainer kit in KEYBOARD or SERIAL mode it scans system key codes.

8255 port addresses: Control word register-43H

Flowchart:



Program 15: DISPLAYING TEXT ON 7 SEGMENT

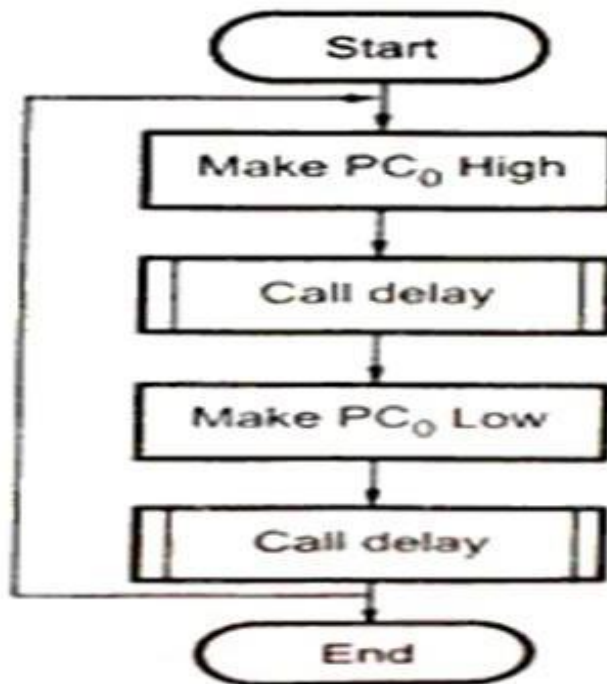
Aim: To display text on seven segment display using 8085 and 8255.

Apparatus required:

8085 Microprocessor trainer kit, seven segment display interface board, Regulated Power supply.

Description: Seven segment display interface is connected over J2 of the trainer. When the trainer kit in KEYBOARD or SERIAL mode scrolling and flashing of display can be observed.

Flowchart:



Program 16: GENERATION OF WAVEFORMS

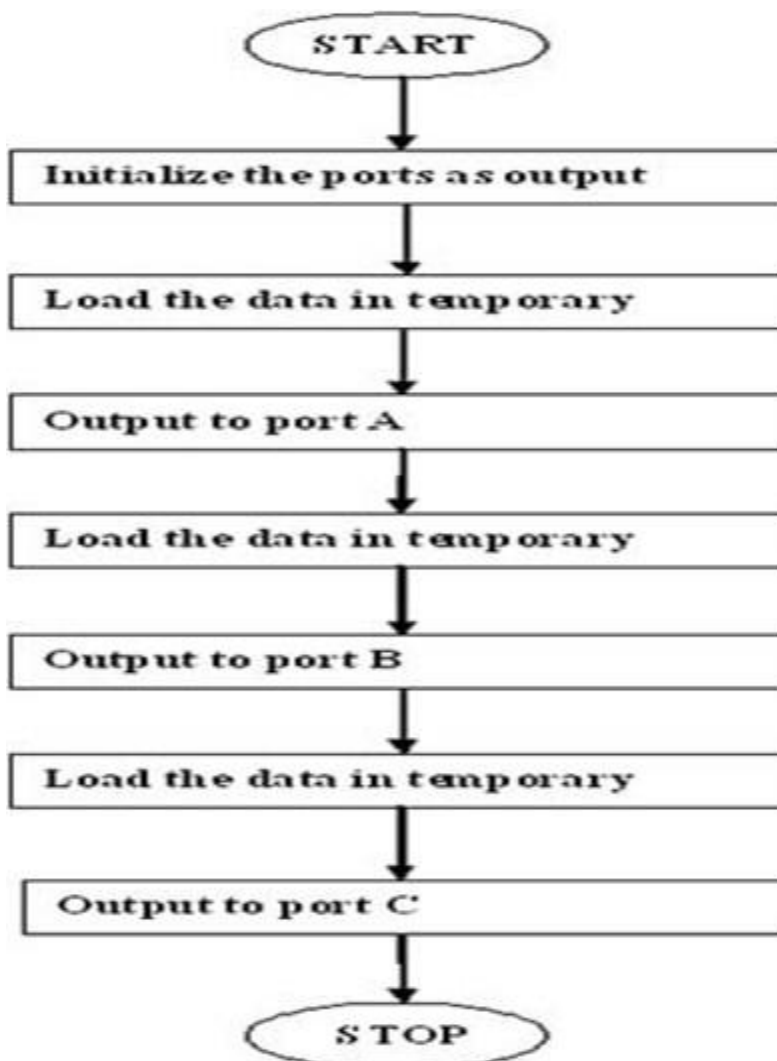
Aim: To generate different waveforms through DAC using 8085 and 8255.

Apparatus required:

8085 Microprocessor trainer kit, DAC interface board, Regulated Power supply, CRO.

Description: Seven segment display interface is connected over J2 of the trainer. When the trainer kit in KEYBOARD or SERIAL mode the square, triangular, sawtooth and staircase can be displayed on CRO.

Flowchart:



Program 17: INTERFACING OF 8279

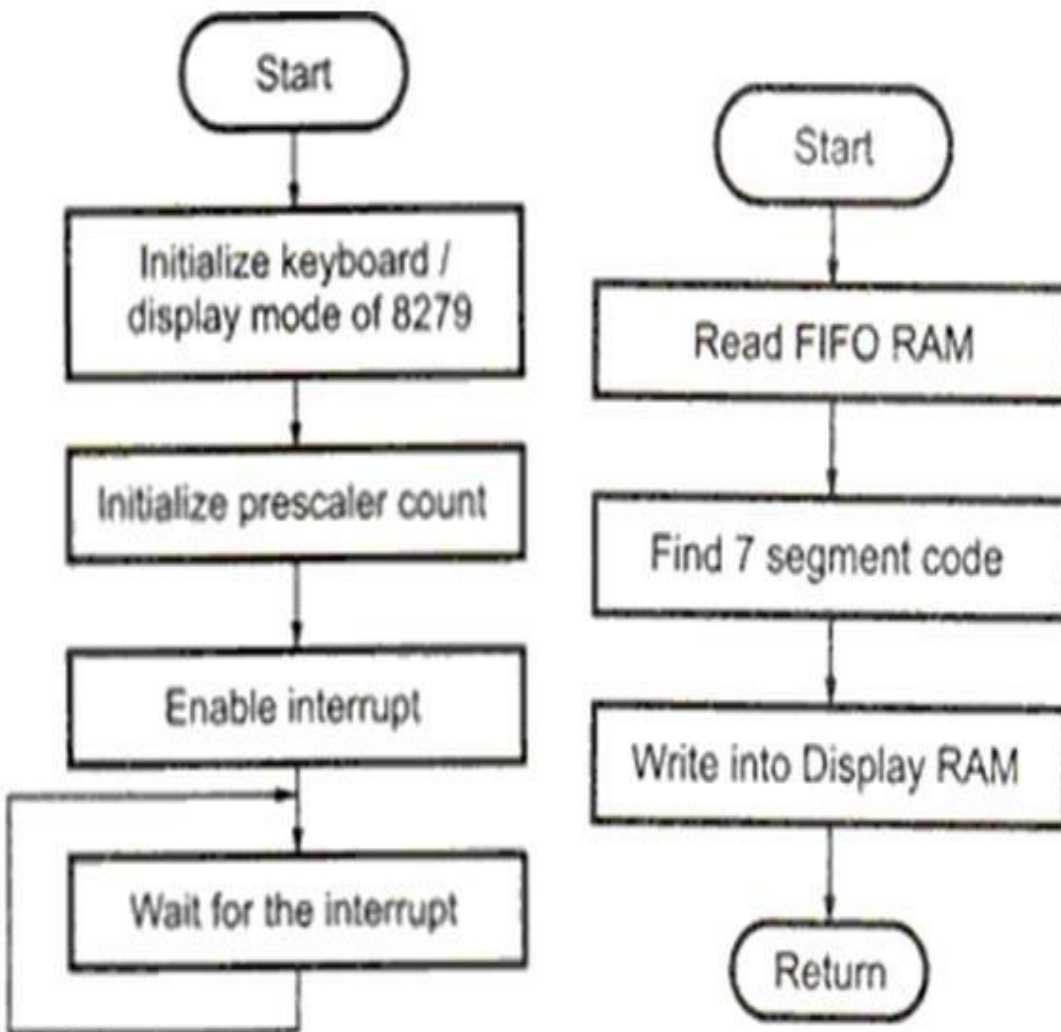
Aim: To interface 8279 to 8085 microprocessor.

Apparatus required:

8085 Microprocessor trainer kit, 8279 keyboard/display interface board, Regulated Power supply.

Description: 8279 keyboard/display interface is connected over J2 of the trainer. When the trainer kit in KEYBOARD or SERIAL mode the pressed key can be displayed on display unit.

Flowchart:



**ANNEXURE-I
UNIVERSITY SYLLABUS**

BIT 281

MICROPROCESSORS LABORATORY

Instruction	3 Periods Per week
Duration of University Examination	3 Hours
University Examination	50 Marks
Sessionals	25 Marks

Course Objectives:

1. To become familiar with the architecture and Instruction set of Intel 8085 microprocessor.
2. To provide practical hands on experience with Assembly Language Programming.
3. To familiarize the students with interfacing of various peripheral devices with 8085 microprocessor.

List of Experiments

1. Tutorials on 8085 Programming.
2. Interfacing and programming of 8255. (E.g. traffic light controller).
3. Interfacing and programming of 8254.
4. Interfacing and programming of 8279.
5. A/D and D/A converter interface.
6. Stepper motor interface.
7. Display interface.

Note: Adequate number of programs covering all the instructions of 8085 instruction set should be done on the 8085 microprocessor trainer kit.