INSTITUTE OF TEXTILE TECFHNOLOGY

MEASURMENT SYSTEM
MICROPROCESSOR LAB MANNUAL

BRANCH- MECHATRONICS ENGG.

5TH SEMESTER

INTISTUTE OF TEXTILE TECHNOLOGY

CHOUDWAR , CUTTACK

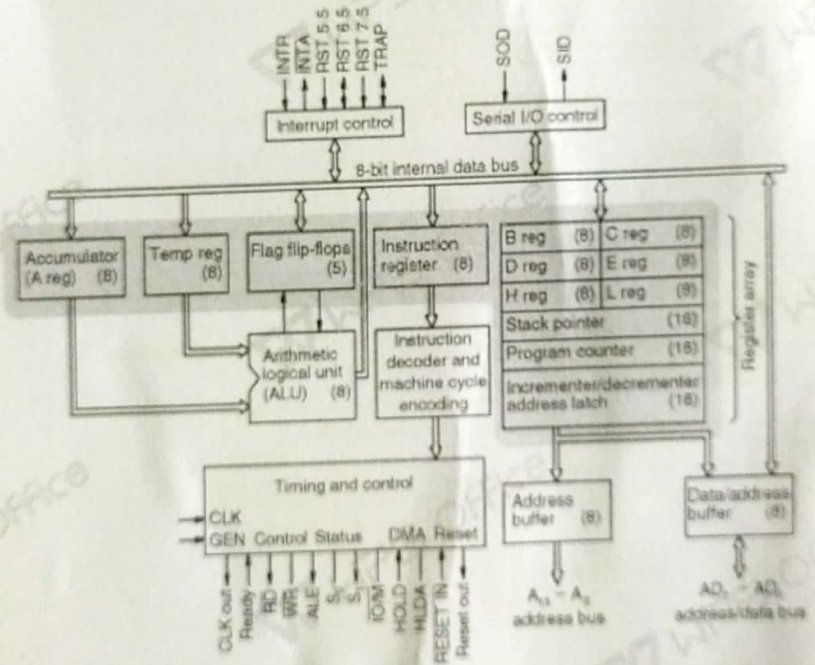
8085 MIROPROCESSOR LAB MANNUAL ,

BRANCH - MECHATRINICS ENGG. 5TH SEMESTER

EXPERIMENT NO 1

Objective: Study of architecture of microprocessor 8085.

Internal Architecture of 8085 Microprocessor



Control Unit

Generates signals within uP to carry out the instruction, which has been decoded. In reality causes certain connections between blocks of the uP to 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. Always stores result of operation in Accumulator.

Registers

The 8085/8080A-programming model includes six registers, one accumulator, and one flag register, as shown in Figure. In addition, it has two 16-bit registers: the stack pointer and the program counter. They are described briefly as follows. The 8085/8080A has six general-purpose registers to store 8-bit data; these are identified as B,C,D,E,H, and L as shown in the figure. 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.

Flags

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; they are listed in the Table and their bit positions in the flag register are shown in the Figure below. The most commonly used flags are Zero, Carry, and Sign. The microprocessor uses these flags to test data conditions. For example, after an addition of two numbers, if the sum in the accumulator id larger than eight bits, the flip-flop uses to indicate a carry -- called the Carry flag (CY) – is set to one. When an arithmetic operation results in zero, the flip-flop called the Zero(Z) flag is set to one. The first Figure shows an 8-bit register, called the flag register, adjacent to the accumulator. However, it is not used as a register; five bit positions out of eight are used to store the outputs of the five flip-flops. The flags are stored in the 8-bit register so that the programmer can examine these flags (data conditions) by accessing the register through an instruction These flags have critical importance in the decision-making process of the microprocessor. The conditions (set or reset) of the flags are tested through the software instructions. For example, the instruction JC (Jump on Carry) is implemented to change the sequence of a program when CY flag is set. The thorough understanding of flag is essential in writing assembly language programs.

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. The stack concept is explained in the chapter "Stack and Subroutines."

Instruction Register/Decoder

Temporary store for the current instruction of a program. Latest instruction sent here

from memory prior to execution. Decoder then takes instruction and 'decodes' or interprets the instruction. Decoded instruction then passed to next stage. Memory Address Register Holds address, received from PC, of next program instruction. Feeds the address bus with addresses of location of the program under execution.

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

Result: Thus the study of Architecture of 8085 microprocessor is completed.

EXPERIMENT NO 2

Objective : Addition of two 8-bit numbers.

Apparatus Required: 8085 Simulator, PC.

Program:

	MVI C 00	Initialize C register to 00
	LDA 4150	Load the value to Accumulator.
	MOV B A	Move the content of Accumulator to B register.
	LDA 4151	Load the value to Accumulator.
	ADD B	Add the value of register B to Accumulator.
	JNC LOOP	Jump on no carry.
	INR C	Increment value of register C.
LOOP:	STA 4152	Store the value of Accumulator.
	MOV A C	Move content of Register C to Accumulator
	STA 4153	Store the value of Accumulator
	HLT	

Obsevation:

Input:	80 (4150)
	80 (4251)
Output:	00 (4252)
	01 (4253)

Result: Thus the program to add two 8 bit numbers was executed.

EXPERIMENT NO 3

Objective: Addition of two 16-bit numbers.

Apparatus required: 8085 Simulator, PC.

Program:

```
LHLD 2000H      Get first 16-bit number
XCHG            Save first 16-bit number in DE
LHLD 2002H      Get second 16-bit number in HL
DAD D           Add DE and HE
SHLD 2004H      Store 16-bit result in memory locations 2004H and 2005H.
HLT             Terminate program execution
```

Observation:

Input:	12 (2000) 23 (2001) 43 (2002) 21 (2003)
Output:	55 (2004) 55 (2005)

Result: Thus the program to add two 16 bit numbers was executed.

EXPERIMENT NO 4

Objective: Subtraction of two 8-bit numbers.

Apparatus required: 8085 Simulator, PC.

Program:

	MVI C 00	Initialize C register to 00
	LDA 4150	Load the value to Accumulator.
	MOV B A	Move the content of Accumulator to B register.
	LDA 4151	Load the value to Accumulator.
	SUB B	Add the value of register B to Accumulator.
	JNC LOOP	Jump on no carry.
	CMA	Complement Accumulator Content
	INR A	Increment value of register C.
	INR C	Increment value of register C.
LOOP:	STA 4152	Store the value of Accumulator.
	MOV A C	Move content of Register C to Accumulator
	STA 4153	Store the value of Accumulator
	HLT	

Observation:

Input:	06 (4150)
	02 (4251)
Output:	04 (4252)
	01 (4253)

Result: Thus the program to subtract two 8 bit numbers was executed.

EXPERIMENT NO 5

Objective: Subtraction of two 16-bit numbers.

Apparatus required: 8085 Simulator, PC.

Program:

LXI H 9876H	load HL with 9876H data
LXI B 5432H	load BC with 9876H data
MOV A L	move the content of L reg. into Accumulator
SUB C	subtract the content of L reg. into Accumulator
STA 2000H	Store the LSB into 2000H
MOV A H	move the content of H reg. into Accumulator
SUB B	subtract the content of H reg. into Accumulator
STA 2001H	Store the LSB into 2000H
HLT	Stop

Observation:

Input:	HL = 9876H BC = 5432H
Output:	44H (2000) 44H (2001)

Result: Thus the program to subtract two 16 bit numbers was executed.

EXPERIMENT NO 6

Objective: Multiplication of two 8- bit nos. using repeated Addition.

Apparatus required: 8085 Simulator, PC.

Program:

```
        MVI D 00        Initialize D register to 00
        MVI A 00        Initialize A register to 00
        LXI H 4150
        MOV B M         Get the second number in B register
        INX H
        MOV C M         Get the second number in C register
LOOP:   ADD B           Add content of A to reg. B
        JNC  NEXT       Jump on no carry to NEXT
        INR C           Increment the content of reg. C
NEXT:   DCR C           decrement the content of reg. C
        JNZ LOOP        Jump on no zero to address
        STA 4152        Store the result in memory
        MOV A D
        STA 4153        Store the MSB of result in memory
        HLT
```

Observation:

Input:	4150(10)
	4151(02)
Output:	4152(20)
	4153(00)

Result: Thus the program to multiply two 8 bit numbers was executed.

EXPERIMENT NO 7

Objective: Division of two 8- bit nos. using repeated Subtraction.

Apparatus required: 8085 Simulator, PC.

Program:

```
        LXI H 4150      Load HL with 4150H
        MVI B M         Get the dividend in b register.
        MVI C 00        Clear C reg. for quotient.
        MOV B M         Get the second number in B register
        INX H
        MOV A M         Get the divisor in A register
NEXT:   CMP B           Compare A reg.with reg B
        JC   LOOP       Jump on no carry to NEXT
        SUB B           Increment the content of reg. C
        INR C           decrement the content of reg. C
        JMP NEXT        Jump on no zero to address
        STA 4152        Store the result in memory
        MOV A C
        STA 4153        Store the MSB of result in memory
        HLT
```

Observation:

Input:	4150(10)
	4151(02)
Output:	4152(05)
	4153(00)

Result: Thus the program to Division of two 8- bit nos. using repeated Subtraction method was executed.

EXPERIMENT NO 8

Objective: Find 1's & 2's complement of a 8 bit number

Apparatus required: 8085 Simulator, PC.

Program:

1's complement of 8 bit number

```
LDA 2200H      Get the number
CMA            Complement number
STA 2300H      Store the result
HLT            Terminate program execution
```

2's complement of a no

```
LDA 2200 H     Get the number
CMA            Complement the number
ADI 01H        ; Add one in the number
STA 2300H      Store the result
HLT            Terminate program execution
```

Result: 1's & 2's complement of a 8 bit number is executed.

EXPERIMENT NO.9

Objective: Find largest Number From an array.

Apparatus required: 8085 Simulator, PC.

Program:

```
        LXI   H, 4200   Set pointer for array
        MOV   B, M      Load the count
        INX   H
        MOV   A, M      Set 1st element as largest data
        DCR   B         Decrement the count
LOOP:   INX   H
        CMP   M         If A reg.>M go to AHEAD
        JNC   AHEAD
        MOV   A, M      Set the new value as largest
AHEAD:  DCR   B
        JNC   LOOP      Repeat comparisons till count=0
        STA   4300      Store the largest value at 4300
        HLT
```

Observation:

Input 05 (4200).....Array size
0A (4201)
F1 (4202)
1F (4203)
26 (4204)
FE (4205)

Output FE (4300)

Result: Thus the program to find largest number in an array was executed.

EXPERIMENT NO.10

Objective: Find smallest No. from an array.

Apparatus required: 8085 Simulator, PC.

Program for Smallest number:

```
        LXI H,4200      Set pointer for array
        MOV B,M         Load the Count
        INX H
        MOV A,M         Set 1st element as largest data
        DCR B           Decrement the count
LOOP:   INX H
        CMP M           If A- reg. <M go to AHEAD
        JC AHEAD
        MOV A,M         Set the new value as smallest
AHEAD:  DCR B
        JNZ LOOP        Repeat comparisons till count=0
        STA 4300        Store the largest value at 4300
        HLT
```

Observation:

Input 05 (4200).....Array size
0A (4201)
F1 (4202)
1F (4203)
26 (4204)
FE (4205)

Output 0A (4300)

Result: Thus the program to find smallest number in an array was executed.

EXPERIMENT NO 11

Objective: Transfer Block of data bytes from one memory location to another .

Apparatus required: 8085 Simulator, PC.

Program:

```
        MVI C, 0AH      Initialize counter
        LXI H, 2200H    Initialize source memory pointer
        LXI D, 2300H    Initialize destination memory pointer
BACK    MOV A, M        Get byte from source memory block
        STAX D          Store byte in the destination memory block
        INX H           Increment source memory pointer
        INX D           Increment destination memory pointer
        DCR C           Decrement counter
        JNZ  BACK       If counter  0 repeat
        HLT             Terminate program execution
```

Result: Transfer Block of data bytes from one memory location to another is executed.

EXPERIMENT NO.12

Objective: Arrange data bytes in ascending order

Apparatus required: 8085 Simulator, PC

Program

```
        LXI H, 4200
        MOV C, M
        DCR C
REPEAT: MOV D C
        LXI  H'4201
LOOP:   MOV A,M
        INX H
        CMPM
        JC SKIP
        MOV B,M
        MOV M,A
        DCX  H
        MOV M,B
        INX H
SKIP:   DCR D
        JNZ LOOP
        DCR C
        JNZ REPEAT
        HLT
```

Result: Arrange data bytes in ascending is executed.

EXPERIMENT NO.13

Objective: Arrange data bytes in descending order
Apparatus required: 8085 Simulator, PC

Program:

```
        LXI H, 4200
        MOV C, M
        DCR C
REPEAT: MOV D, C
        LXI H, 4201
LOOP:   MOVA, M
        INX H
        CPM M
        JNC SKIP
        MOV B, M
        MOV M, A
        DCX  H
        MOV M, B
        INX H
SKIP:   DCR D
        JNZ LOOP
        DCR C
        JNZ REPEAT
        HLT
```

Result: Thus the program to Arrange data bytes in descending order was executed.

EXPERIMENT NO.14

Objective: Study of IC 8255.

Apparatus required: 8085 Simulator, PC,8255 IC.

Theory: The 8255A programmable peripheral interface (PPI) implements a general-purpose I/O interface to connect peripheral equipment to a microcomputer system bus. The core's functional configuration is programmed by the system software so that external logic is not required to interface peripheral devices.

Features

- Three 8-bit Peripheral Ports - Ports A, B, and C
- Three programming modes for Peripheral Ports: Mode 0 (Basic Input/Output), Mode 1 (Strobed Input/Output), and Mode 2 (Bidirectional)
- Total of 24 programmable I/O lines
- 8-bit bidirectional system data bus with standard microprocessor interface controls

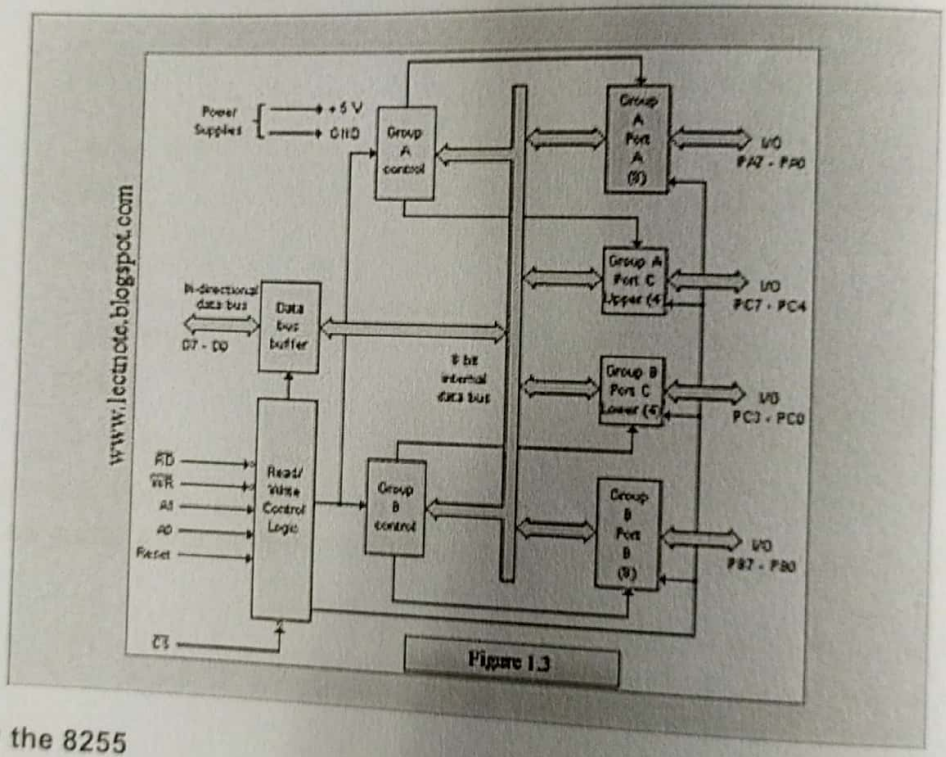


Figure 1.3

Programming the 8255

CONTROL LOGIC

$\overline{RD}$ (Read): This control signal enables the Read operation. When the signal is low, the MPU reads data fro a selected I/O Port of the 8255.

$\overline{WR}$ (Write): This control signal enables the write operation. When the signal goes low, MPU writes into a selected I/O Port or control register.

RESET (Reset): This is an active high signal; it clears the control register and sets all ports in the input mode.

$(\overline{CS})$, A0 and A1: Theses are device select signals. Chip Select is connected to a decoded address, and A0 and A1 are generally connected to MPU address lines A0 and A1 respectively

$(\overline{CS})$	A_1	A_0	Selected
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control Register
1	X	X	Not Selected

CONTROL WORD

Figure 1.5 shows a register called the control register. The contents of this register called control word. This register can be accessed to write a control word when A0 and A1 are at logic 1. This control register is not accessible for a read operation.

Bit D7 of the control register specifies either I/O function or the Bit Set/Reset function. If bit D7=1, bits D6-D0 determines I/O functions in various modes. If bit D7=0, Port C operates in the Bit Set/Reset (BSR) mode. The BSR control word does not affect the functions of Port A and Port B.

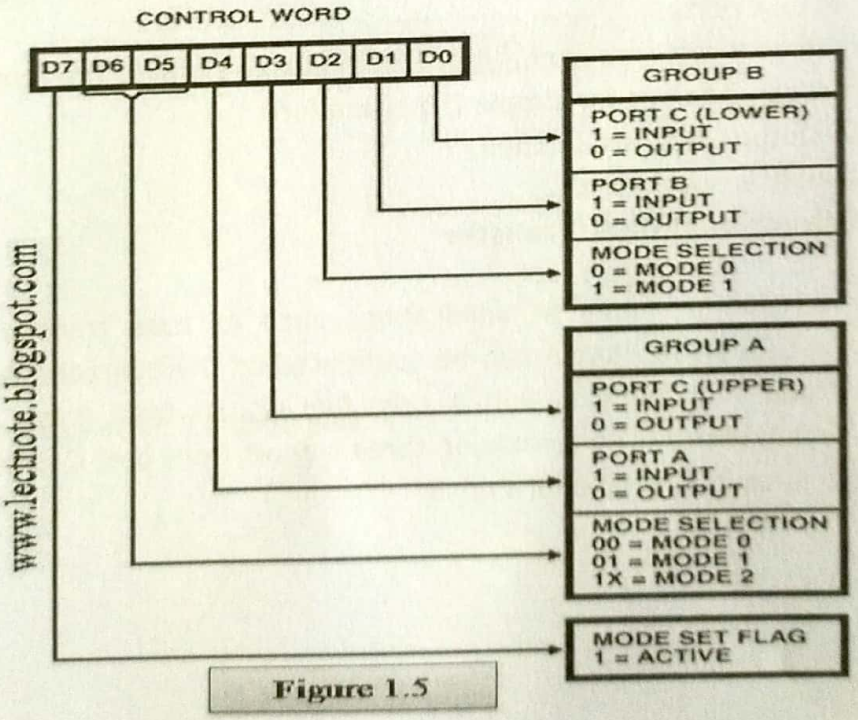


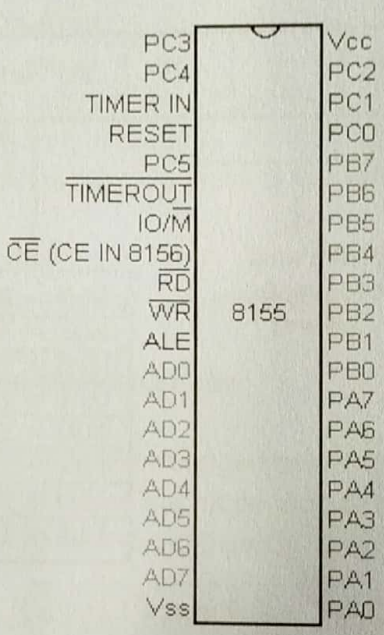
Figure 1.5

EXPERIMENT NO.15

Objective: Inter facing of IC 8155.

Apparatus required: 8085 Simulator, PC.

Theory: the 8155 is a multipurpose programmable device specially designed to be compatible with the 8085 microprocessor. The ALE, IO/M RD and WR signals from the 8085 can be connected directally to the device. This eliminates the need for external demultiplexing of the low-order bus AD_7-$Ad_{0\ and}$ generation of the control signals such as MEMR,IOR MEMW and IOW. The 8155 includes 256 bytes of R/W memory,three I/O ports,and a timer.the pin diagram 8155 is shown in figure:



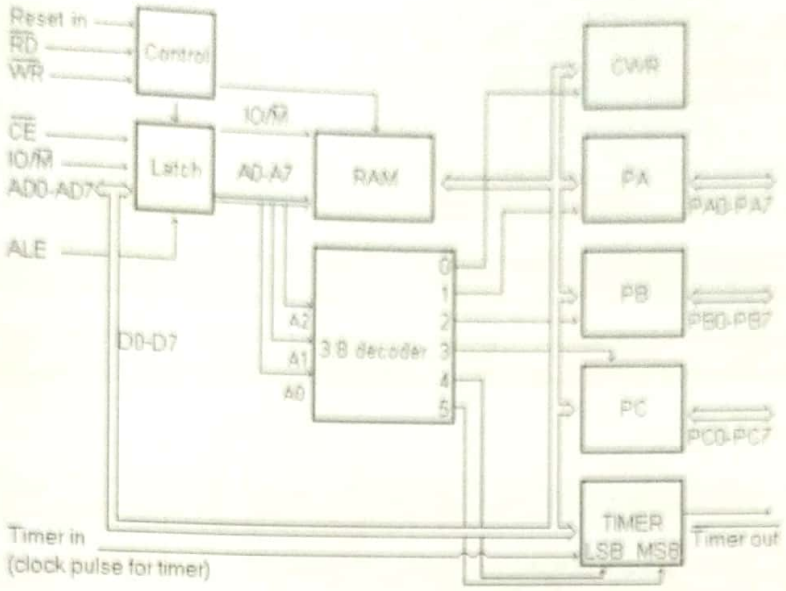
The control word format of 8155 is shown in figure

D7	D6	D5	D4	D3	D2	D1	D0
Timer command		IEB	IEA	PC		PB	PA

- D0, D1: mode for PA and PB, 0=i/p, 1=o/p
- D2, D3: mode for PC
- D4, D5: interrupt enable for PA and PB, 0=disable 1=enable
- D6, D7: Timer command:
  - 00: No effect
  - 01: Stop if running else no effect
  - 10: Stop after terminal count (TC) if running, else no effect

11: Start if not running, reload at TC if running

The block diagram of 8155 is shown in figure:



A2	A1	A0	Port
0	0	0	Command/status reg.
0	0	1	PA
0	1	0	PB
0	1	1	PC
1	0	0	Timer LSB
1	0	1	Timer MSB

Status register format:

D7	D6	D5	D4	D3	D2	D1	D0
X	Timer	INTE-B	BF-B	INTR-B	INTE-A	BF-A	INTR-A

D6: Timer. Latched high when TC is reached, low when status reg is read or reset is done.

EXPERIMENT NO.16

Objective: Inter facing of IC 8279.
Apparatus required: 8085 Simulator, PC, 8085 Microprocessor toolkit, 8279 Interface board.VXT parallel bus,Regulated D.C power supply

Program:

```
START:  LXI H, 4130H
        MVI D,0FH; Initialize counter.
        MVI A,10H
        OUT CCH  ;Set Mode and Display.
        MVI  A,CCH;Clear display.
        OUT C2H
        MVI A,90H  ;Write Display
        OUT C2H
LOOP:   MOV A,M
        OUT C0H
        CALL  DELAY
        INX  H
        DCR  D
        JNZ  LOOP
        JMP  START

DELAY:  MVI B,A0H
LOOP2:  MVI  CFFH
LOOP1:  DCR  C
        JNZ   LOOP1
        DCR  B
        JNZ LOOP2
        RET
```

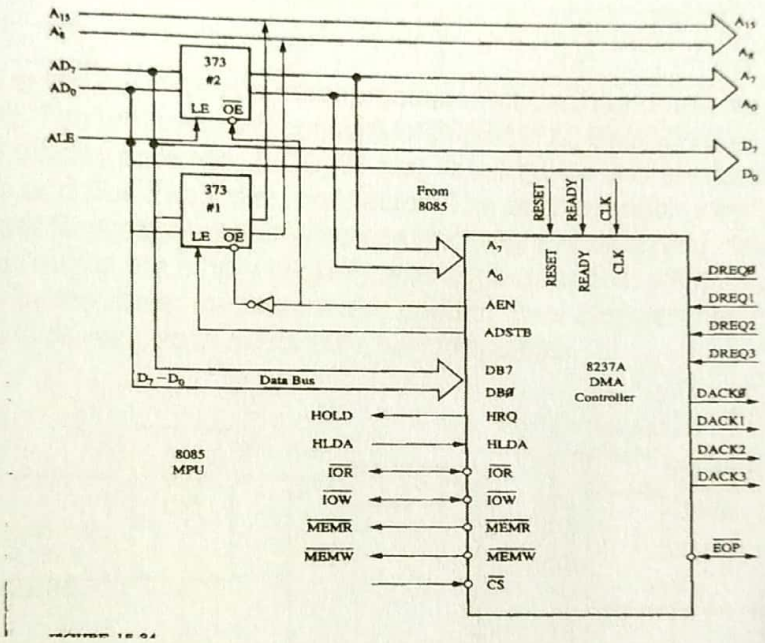
Result: Thus 8279 controller was interfaced with 8085 and program for rolling display was executed successfully.

EXPERIMENT NO.17

Objective: Inter facing of IC 8257.

Apparatus required: 8085 Simulator, PC

Theory: The DMA (8257/8237) is used to transfer data bytes between I/O and system memory at high speed. it includes eight data lines,four control signals and eight address lines,however it needs 16 address lines to acess 64K bytes therefore an additional eight lines must be generated When a transfer begins,DMA places the low order bytes on the address bus and the high order bytes on the data bus and asserts AEN and ADSTB (address strobe). These two signals are used to latch the high order bytes from the data bus thus it places the 16 bit address on the system bus.After the transfer of first byte,the latch is updated when the lower bytes generates a carry. To implement the DMA transfer,the 8257 should be initialized by writing into various control registers in the DMA channels and interfacing section. To initialize the 8257,the following steps are necessary:



1. write a control word in the mode register than selects the channel and specifices the type of transfer and the DMA mode.
2. write a control word in the command register that specifies parameters such as priority among four channel DREQ and DACK active levels,and timing and enables the 8257.
3. write the starting address of the data to be transferred in the channel memory address register.
4. write the count in the channel count register.

Program:

```
MVI A 00000100B
OUT 08H
MVI A 00000111B
OUT 0B H
MVI A 75H
OUT 06H
```

EXPERIMENT NO 18

Objective: Microprocessor based stepper Motor control.

Apparatus required: 8085 Simulator, PC. Stepper motor interface Board.

THEORY:- A motor in which the rotor is able to assume only discrete stationary angular position is a stepper motor. The rotor motion occurs in a stepwise manner from one equilibrium position to next.

The motor under our consideration uses 2- phase scheme of operation. In this scheme, any two adjacent stator winding are energized. The switching condition for the above said scheme is shown in table.

		Clockwise			Anti-clockwise		
A1	B1	A2	B2	A1	B1	A2	b2
1	0	0	1	1	0	1	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	0
1	0	1	0	1	0	0	1

The hardware setup consists of a microprocessor motherboard and stepper motor interface board. The motherboard consists of 8085 MPU, 8KB EPROM, 8KB RAM, Keyboard and display controller 8279, 21-key Hex-keypad and six numbers of seven segment LEDs and Bus Expansion connector. The stepper motor interface consists of driver transistors for stepper motor windings and address decoding circuit. The microprocessor output the binary sequence through data bus, which are converted to current pulses by the driver transistors and used to drive stepper motor. The software for the system is developed in 8085 assembly language.

Result: is executed.

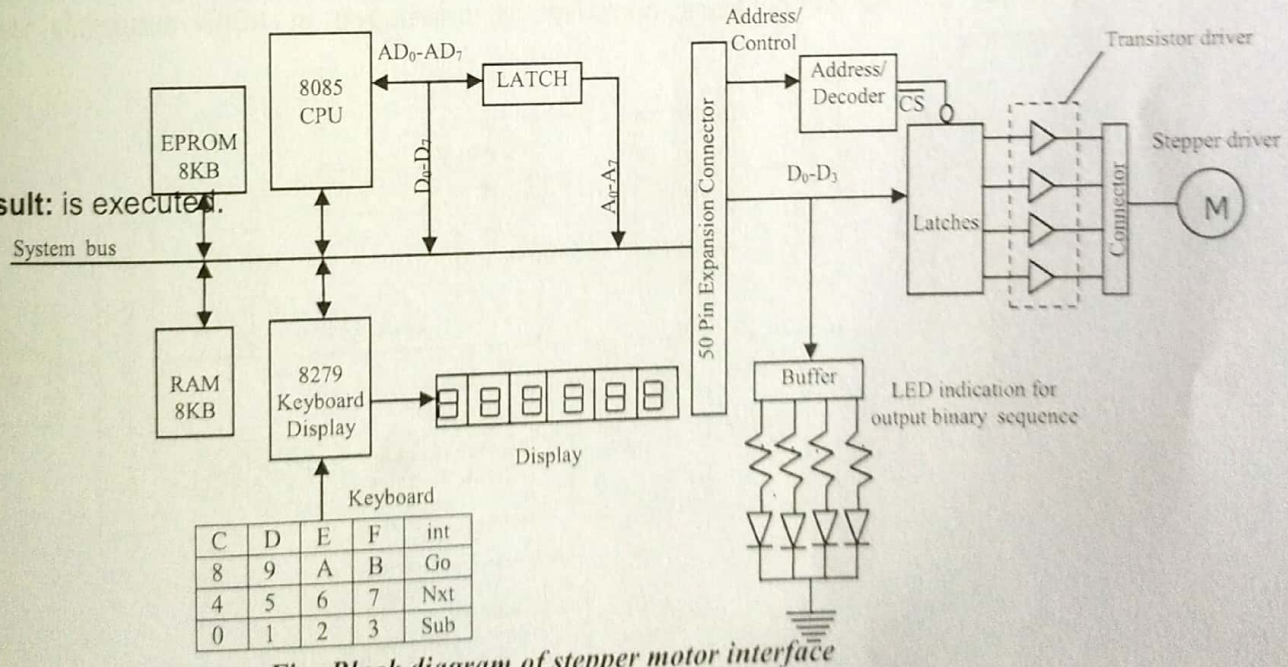


Fig: Block diagram of stepper motor interface