इ रि से ट
माइक्रोप्रोसेसर प्रयोगशाला
प्रयोग नं: एम पी 1

IRISET

Microprocessor Laboratory

EXPERIMENT NO.: MP- 1

नाम
Name : ___________ ____

अनुक्रमांक
Roll No : ________________

पाठ्यक्रम
Course : ________________

दिनांक
Date : ________________

प्राप्त अंक
Marks Awarded :

अनुदेशक का अधाक्षर
Instructor Initial :

Experiment No. MP -1

Introduction To 8085 Microprocessor Kit

Description of 8085 Basic Functions

The following basic functions are required to be known by every trainee before being able to practice programming on 8085 kit.

1. <u>Storing Program / Data In Memory</u>

a) Press the following keys in the given sequence

RESET → EXMEM → Enter required Memory Address → NEXT → Enter Opcode → NEXT → Enter Opcode or data → NEXT

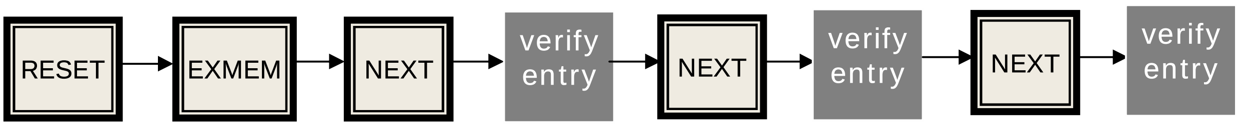
b) When Reset key is pressed the display should be **‘µP 85’**
c) Even after the entry of last opcode or data byte, the NEXT key must be pressed
d) That means this process should end with the pressing of NEXT key.

Exercise:

1) Enter the following data given in table-1, starting from 3500 memory register
2) Now verify the correctness of data entered into memory, by the below given key sequence.
3) After every NEXT key press the entries can be verified.
4) And record the same in the table-1, under data available column.

S.No	Memory Address	Data stored	Data available
1	3500	25	
2	3501	A2	
3	3502	56	
4	3503	7D	
5	3504	88	

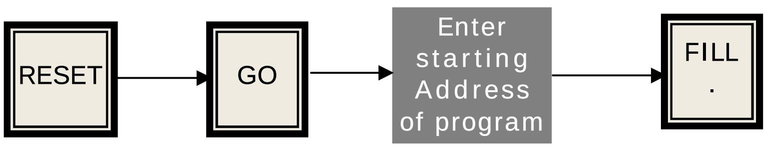
Table-1



2. Execution of Program

a) Entire Program Execution At One Go

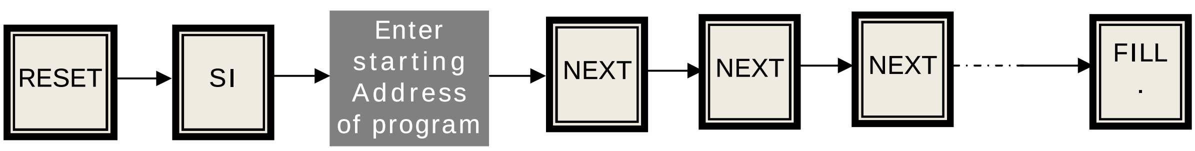
1. In this mode of **program execution** the entire program is executed at one stretch
2. For the execution of a program that is stored in memory by the above sequence of steps, follow the sequence shown below



3. After the above sequence of operations, the display shows **'E'** which means the program is **executed** or execution is going on.

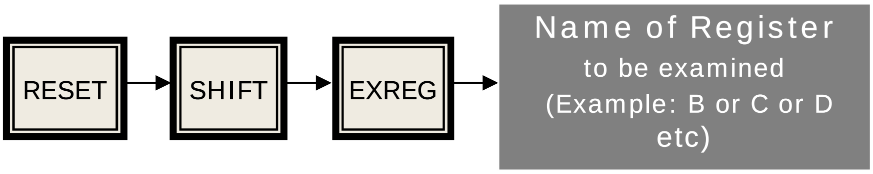
b) Single Instruction/Step (SI) Mode of Operation

1) In this mode, the program is executed in steps of single instruction at a time. The below given key sequence is used for executing programs in this mode. This procedure is used mainly for debugging (error detection/correction in) a program.
2) When a program is executed in this mode, the microprocessor halts after executing only the present instruction. For the execution of the next following instruction in the program, the NEXT key must be pressed again, and so on. For terminating the program at any point, the FILL key has to be pressed, instead of NEXT key.



3. <u>To Examine the Contents of Internal Registers of Microprocessor</u>

1) Follow the key sequence given below to examine or check the contents of any internal register of µP.



2) Contents of a register appear on the display

3) By pressing NEXT key, the contents of a next register of microprocessor can be examined, and so on.

Exercise:

1) Do the following

   a) Enter the below given program into memory from location 2000H onwards. Execute it as per the sequence given in **2**.

   b) Then verify the contents of each register of the microprocessor using the function sequence in **3**.

   c) And note down the same in the space given against each instruction.

MVI A, 25 ;
MVI B, 37 ;
MVI C, 41 ;
MVI D, 55 ;
MVI E, 62 ;
MVI H, 20 ;
MVI L, 50 ;
MVI M, 77 ;
HLT

2) In the table-1 given below, note down the contents of **any other registers** that you came across while examining the above registers contents.

S.No	Name of Register	Contents	Remarks
1			
2			
3			
4			
5			
6			

Table-1

3) The following is a program for ADDITION of **two** 8-bit values which are assumed to be available at 2050 and 2051 memory locations. Load this program and execute it. Verify and record the results by using different 8-bit values, as shown below in table-2.

```
        LXI   H, 2050      ; Point to the 1st value in memory
        MVI   B, 00        ; Reserve B - Register for storing Carry Flag
        MOV   A, M         ; Copy value at 2050 into A – register
        INX   H            ; increment address in HL to point to 2nd value.
        ADD   M            ; add both the values
        JNC   Skip*        ; if no carry go to skip
        INC   B            ; increment B if there is a carry
Skip:   LXI   H, 2060      ; load new address in HL
        MOV   M, A         ; store result at 2060 memory register
        INX   H            ; point to memory register 2061
        MOV   M, B         ; store result at 2060
        HLT                ; stop
```

***** Enter the line address of the instruction LXI H, 2060 in place of this **Skip**

S.No	1st value	2nd value	Sum @ 2060	Carry @ 2061
1	54	33		
2	87	92		
3				
4				

Table-2

Review Questions:

1) What is the content that you could find in A register after executing the program in exercise 1 above? And what do you understand by this?

2) Where is the data, that was stored with instruction ' MVI M, 77' in the above program, available and why?

इ रि से ट
माइक्रोप्रोसेसर प्रयोगशाला
प्रयोग नं: एम पी 2

IRISET

Microprocessor Laboratory

EXPERIMENT NO.: MP- 2

नाम Name :	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _		
अनुक्रमांक Roll No :	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _	प्राप्त अंक Marks Awarded	:
पाठ्यक्रम Course :	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _		
दिनांक Date :	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _	अनुदेशक का अधाक्षर Instructor Initial	:

Experiment No. MP-2

Exchange of Data Between Two Arrays Using 8085

Description

The program given below is meant for exchanging data in two equal sized arrays. For the sake of executing this program, assume that the two data arrays are of TEN bytes each and their starting addresses are 2050 and 2060.

Procedure:

1. Load the following program into memory at any location other than 2050 to 206F.
2. Load data into the two arrays as given below in table-1.
3. Execute the program and check whether data in the arrays got exchanged.

Program:

```
       LXI   H, 2050    ; load data array address into HL
       LXI   D, 2060    ; load data array address into DE
       MVI   C, 0A      ; counter to store no. of bytes in arrays
Rpt:   MOV   A, M       ; get a byte from 1st array
       XCHG             ;
       MOV   B, M       ; get a byte from 2nd array
       MOV   M, A       ;
       XCHG             ; exchange them
       MOV   M, B       ;
       INX   H          ; go to next bytes
       INX   D          ;
```

```
DCR  C          ;
JNZ  Rpt        ; continue exchange till all bytes are exchanged
HLT             ; end  the program
```

S.No	1^{st} Array Data Starting at **2050**	2^{nd} Array Data Starting at **2060**
1	25	A6
2	33	B1
3	44	C2
4	52	D6
5	11	9B
6	20	4F
7	78	F2
8	84	3D
9	17	6C
10	01	5A

Table-1

Review Questions:

1) What can be the maximum size of each array in the above program?
2) Write a program for locating a given number in an array.
3) Explain the logic of the above program through a flowchart .

नाम
Name : ____________ ____
अनुक्रमांक
Roll No : ________________
पाठ्यक्रम
Course : ________________
दिनांक
Date : ________________

प्राप्त अंक
Marks Awarded :

अनुदेशक का अधाक्षर
Instructor Initial :

Experiment No. MP-3

Tone Generation & Flashing of LEDs using 8085 Program

Description

In the following circuit, two LEDs and one loud speaker are interfaced to 8085 microprocessor through 8255, PPI. Let us write a program which performs the following operations:

1) When the switch is OFF, keeps the LEDs flashing.
2) And when the switch is ON, lights LEDs steadily and generates an audible tone through the speaker.

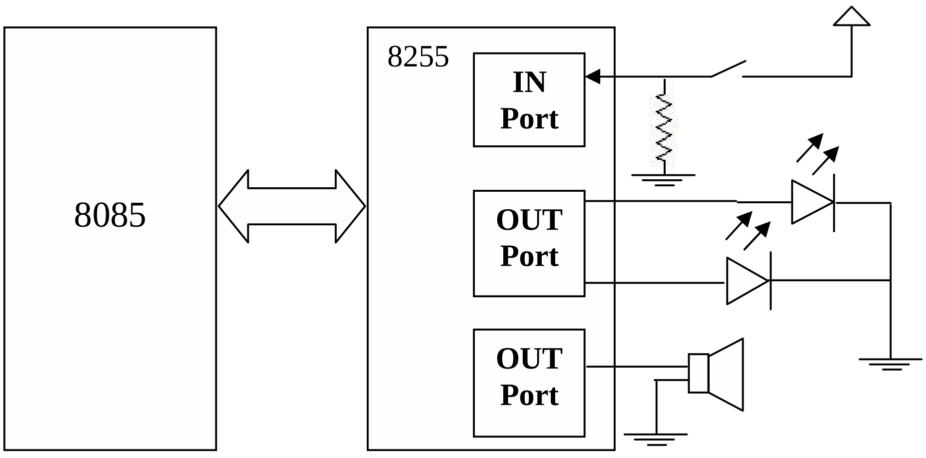


Fig. Interfacing LEDs and a Loud speaker with 8085

Procedure:

1) Make connections as shown in the above figure.
2) Prepare CW for 8255 to have one IN port and two OUT ports
3) Load the below given program using the CW and execute it.
4) Operate the switch from OFF to ON and vice versa
5) Check for flashing of LEDs when switch is OFF and
6) Observe for steady light on LEDs and a tone output from the speaker when switch is ON

Program:

Note: *Read the required **I/O addresses** from the Kit –Manual*
*Chose suitable values for **xx** and **kk** which give proper time delay.*
xx should be greater kk

```
          MVI   A, CW              ; Store Control Word .
          OUT   CWR                ; Send Control Word to CWR

Read:     IN    In-port            ; Wait till switch is ON
          ANI   01                 ;
          CPI   01                 ;
          JZ    Tone               ;

          MVI   A, 01              ; Flash the lights
          OUT   Out-port-1         ;
          CALL  Delay-1            ;
          MVI   A, 10              ;
          OUT   Out-port-1         ;
          JMP   Read               ;

Tone:     MVI   A, 03              ; Steady lights
          OUT   Out-port-1         ;
          MVI   A, 01              ; Tone output
          OUT   Out-port-2         ;
          CALL  Delay-2            ;
          MVI   A, 00
          OUT   Out-port-2         ;
          JMP   Read               ;
```

<u>Subroutines</u>

```
Delay-1:  MVI   C, FF              ; Interval for Flashing
          MVI   B, xx              ;
Rpt :     DCR   C                  ;
          JNZ   Rpt                ;
          DCR   B                  ;
```

```
            JNZ   Rpt
            RET                        ;

Delay-2:    MVI   C, FF                ; Interval  for  Tone frequency
            MVI   B, kk                ;
 Agn:       DCR  C                     ;
            JNZ   Agn                  ;
            DCR  B                     ;
            JNZ   Agn
            RET                        ;
```

Exercise:

1) Select different values for *xx* and *kk* to obtain different time intervals for flashing and different tone frequencies.
2) Write a program to generate 1000Hz tone using programmable timer 8253.

Review Questions:

1) Which values of xx and kk had produced proper flashing and good audible tone?
2) What is the logic used in this program for tone generation or for LED flashing?
3) Represent the above program logic through a flow chart.

इ रि से ट
माइक्रोप्रोसेसर प्रयोगशाला
प्रयोग नं: एम पी 4

IRISET

Microprocessor Laboratory

EXPERIMENT NO.: MP 4

नाम Name	:	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _		
अनुक्रमांक Roll No	:	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _	प्राप्त अंक Marks Awarded	:
पाठ्यक्रम Course	:	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _		
दिनांक Date	:	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _	अनुदेशक का अधाक्षर Instructor Initial	:

Experiment No. MP- 4

8085 Program For Implementing A 3-Aspect Signal

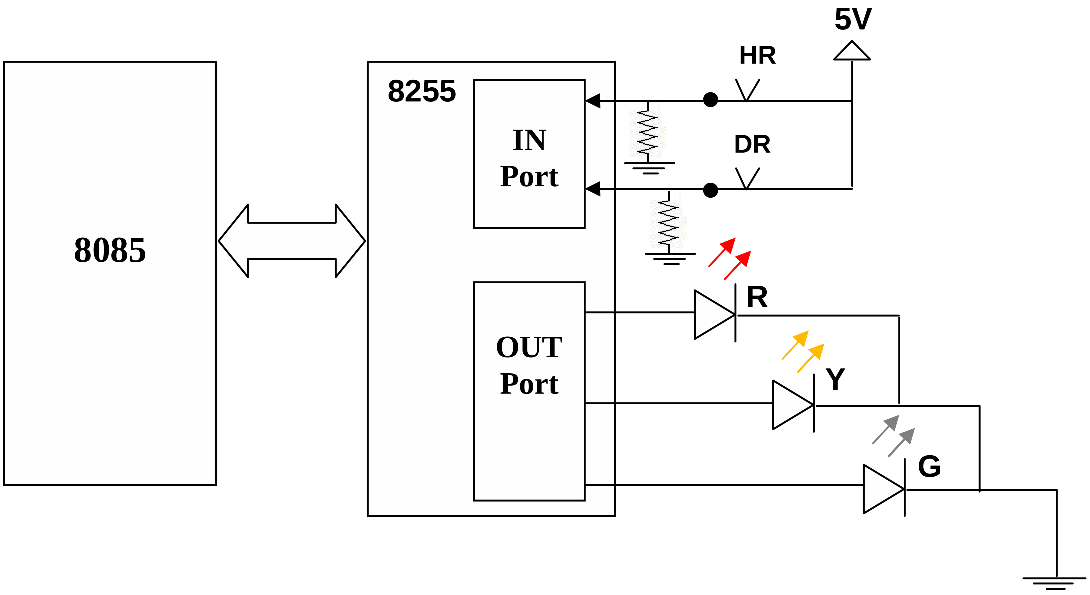


FIG. Simulation of a 3-Aspect Signal

Description

The above figure gives a simple example of how 3-aspect signaling can be implemented using a microprocessor based system, for example an **SSI**. In this schematic circuit only two input parameters HR and DR are used just to explain the concept. But the actual circuit in SSI uses more number of input parameters for controlling the signal aspect.

In this example the aspect of the signal depends on the status of HR and DR relays, which is shown in the below table.

S.No	**HR** relay	**DR** relay	**Signal Aspect**
1	Drop ↓	Drop ↓	**Red**
2	Pickup ↑	Drop ↓	**Yellow**
3	Pickup ↑	Pickup ↑	**Green**

Table-1

Procedure:

7) Make connections as shown in above figure.
8) Load the below given program and execute it.
9) Vary HR & DR status and check whether the aspects of signal are compliant with the data in table-1.

Program:

```
          LXI  SP, 3F00          ;Initialize stack pointer
          MOV  AL, CW            ; Initialize 8255
          OUT  CWR               ;

Red :     MVI  A, 01             ;
          OUT  port B            ; Show Red Aspect

Read:     IN  port A             ; Read HR & DR status
          ANI  03                ;
          MOV  B, A              ;

Confirm:  CALL Delay             ; Wait for a while
          IN  port A             ; Read Relay Status once again
          ANI  03                ;
          CMP  A, B              ; Conform the Status
          JNE  Read              ;

          CPI  A, 01             ; check whether pro-yellow status
          JNZ  Green             ; if not go to pro- green check

          MVI  A, O2             ;
          OUT  port- B           ; Show Yellow Aspect
          JMP  Read              ;

Green:    CPI  03                ; check whether pro- green
          JNZ  Red               ; if not go to Red aspect
          MVI  A, O3             ;
          OUT  port- B           ; Show Green Aspect
          JMP  Read              ;
```

Delay Subroutine

```
Delay :    LXI  D, 20FF        ;
Linger:    DCR  E              ;
           JNZ  Linger         ;
           DCR  D              ;
           JNZ  Linger         ;
           RET                 ;
```

(Note: Read the required I/O addresses from the kit manual)

Exercise:

1) Keep HR down and DR up and observe what aspect you will get.
2) Check for fail-safe feature in the program.

Review Questions:

1) In exercise 1 above, did the program implement the fail-safe feature for the signal ? Explain how?

2) What are the input values available at IN port for different status of HR & DR ? And also what output data the microprocessor needs to send for showing the different signal aspects? Show with a table.

इ रि से ट

माइक्रोप्रोसेसर प्रयोगशाला

प्रयोग नं: एम पी 5

IRISET

Microprocessor Laboratory

EXPERIMENT NO.: MP- 5

नाम
Name : ___________ ____

अनुक्रमांक
Roll No : ________________

प्राप्त अंक
Marks Awarded :

पाठ्यक्रम
Course : ________________

दिनांक
Date : ________________

अनुदेशक का अधाक्षर
Instructor Initial :

Experiment No. MP- 5

Stepper Motor Operation

Objective: To control the operation and rotation of a stepper motor using **8085** program

Description

A **stepper motor** is a brushless, synchronous electric motor that can divide a full rotation into a large number of **steps**.

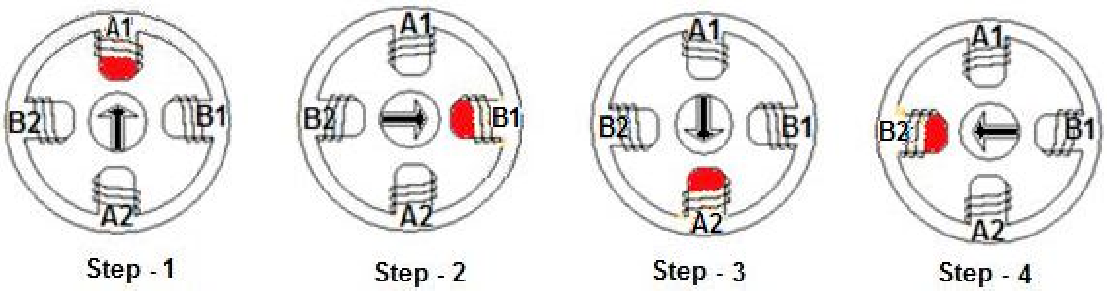


Fig. 1 - Clockwise rotation of stepper motor by sequential energization of Single Winding at a time

Stepper motors have multiple windings which act like electromagnets when energized by dc supply. These windings are arranged in circular shape and there is a rotor in the center of the circle. Generally, the rotor is a permanent magnet which rotates in the direction of an energized (winding) electromagnet. This rotor is a small toothed wheel which engages with the teeth of a bigger diameter wheel, which has a shaft in the center of it. The electromagnets are energized one after the other in a proper order, by an external control circuit, to make the motor shaft rotate in small **steps.** A large number of such steps make a full rotation of the motor shaft. By controlling the number of steps the motor can be rotated by almost to any precise angle.

S.No.	A1	A2	B1	B2	Hex Equivalent
1	1	0	0	0	8
2	0	0	1	0	2
3	0	1	0	0	4
4	0	0	0	1	1

Table-1 - Single winding Energization Data for a stepper motor of 4-windings for clockwise rotation in steps

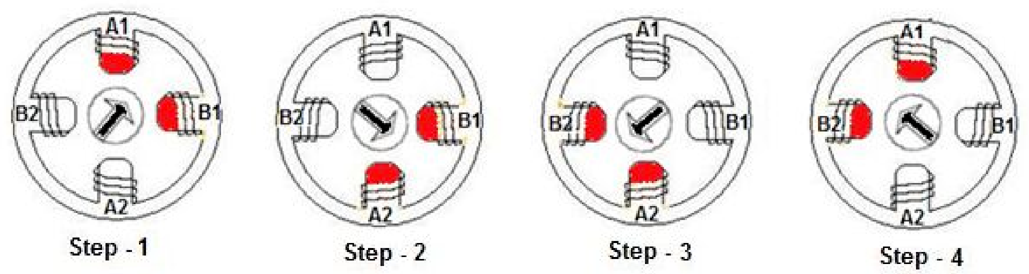


Fig. 2- Clockwise rotation of stepper motor by sequential energization of two adjacent windings at a time

S.No.	A1	A2	B1	B2	Hex Equivalent
1	1	0	1	0	A
2	0	1	1	0	6
3	0	1	0	1	5
4	1	0	0	1	9

Table-2 - Two winding Energization Data f or a stepper motor of 4-windings for clockwise rotation in steps

Applications of Stepper Motor:

In floppy disk drives, flatbed scanners, computer printers, plotters etc.

Procedure:

1) Connect given stepper motor to any one port of 8255, on the kit
2) Load the below given program
3) Load step data from Table-1 in memory
4) Execute and observe the motor rotation
5) Load the step data in reverse order and observe for reverse rotation
6) Now, load step data from Table-2 above and repeat steps 3 and 4
7) By changing the value of **D-register** in delay sub-routine observe the variations in motor speed.

8085 Program:

```
        LXI  SP, 3500         ; Initialize Stack
        MVI   A, CW           ; Initialize 8255
        OUT   CWR             ;
        MVI   C, 04           ;

Rpt:    LXI   H, 2050         ; Load data table pointer
Step:   CALL Delay            ;
        MOV   A, M            ; get step data from the table
        OUT   port addr       ; send it to the motor
        INX   H               ; point to next step data in the table
        DCR   C               ;
        JNZ   Step
        JMP  Rpt
```

Delay Subroutine

```
Delay:  MVI   D, FF     ; delay values
        MVI   E, FF     ;
Lp1:    DCR   E         ;
        JNZ   Lp1       ;
        DCR   D         ;
        JNZ   Lp1       ;
        RET             ;
```

Exercise:

1) By changing the value of **D-register** in the delay sub-routine of the above program observe the variations in the motor speed. Try to get visibly step-less (smooth) rotation.
2) Count the **number of steps** needed for completing **one full rotation** of the shaft.

Review Questions:

1) What value in **D** did give a smooth rotation which was visibly step-less?
2) How many steps were needed for one full rotation of the shaft?

नाम
Name : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
अनुक्रमांक
Roll No : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
पाठ्यक्रम
Course : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
दिनांक
Date : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

प्राप्त अंक
Marks Awarded :

अनुदेशक का अधाक्षर
Instructor Initial :

Experiment No. MP-6

Introduction To 8086 Microprocessor Kit

Objective

Familiarization with the basic functions of the kit which are needed while practicing programming on 8086 microprocessor kits.

Description of 8086 Kit Basic Functions

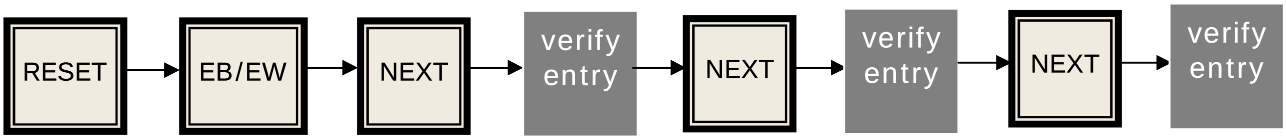
The following basic functions are required to be known by every trainee before being able to practice programming on the 8086 kit.

4. Storing Program In Memory

a) Follow the below given key-sequence to store a program(or data) in memory

RESET → EB/EW → Enter Memory Address → NEXT → Enter Opcode → NEXT → Enter Opcode or data → NEXT

b) When Reset key is pressed the display should be **'µP 86'**
c) Even after entry of last opcode or data byte, the NEXT key must be pressed
d) That means this process should end with the pressing of NEXT key.
e) As an exercise enter the following data starting from 250 memory register
f) 25, 34, 56, 75, 88, 99, 01 and 4F
g) Now verify the correctness of data entered into memory, by the below given key sequence.

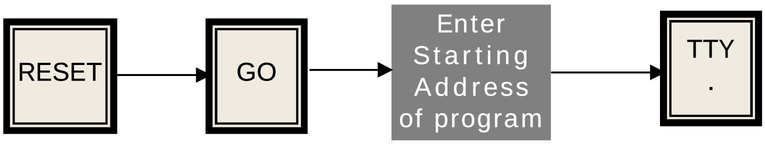


5) After every NEXT key press the entries can be verified on the display.

5. Execution of Program

c) Entire Program Execution At One Go

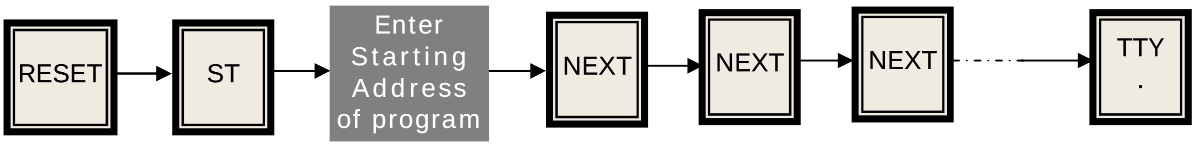
1) In this mode of **program execution** the entire program is executed at one stretch
2) For the execution of a program that is stored in memory by the above sequence of steps follow the sequence shown below



3) After the above sequence of operations, the display shows **'E'** which means the program is **executed** or execution is going on.

d) Single Instruction/Step (ST) Mode of Operation

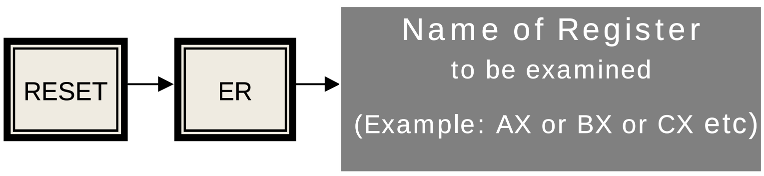
1) In this mode a program is executed in steps of single instruction at a time. The key sequence given below is used for this purpose.



2) After the execution of every instruction the NEXT key must be pressed to go to the execution of the very next instruction in the program or it can terminated at that point itself with the FILL key. This mode is used for debugging programs.

6. To Examine the Contents of Internal Registers of Microprocessor

a) Use key sequence given below to check the contents of any internal registers of µP.



b) Contents of the register appear on the display
c) By pressing NEXT key other next register contents can be examined and so on

Exercise:

4) Do the following

   d) Enter the below given program into memory from location 2000H onwards. Execute it as per the sequence given in procedure **2 (b)**. Terminate the program before HLT.

   e) Then without pressing RESET key, using the function sequence **3** verify the contents of each register of the microprocessor.

   f) And note down the same in the space given against each instruction.

```
MOV   AX,   2540     ;
MOV   BX,   3755     ;
MOV   CX,   4102     ;
MOV   DX,   5566     ;
MOV   SI,   6200     ;
MOV   DI,   6300     ;
MOV   BP,   0250     ;
MOV   SP,   0750     ;
MOV   [SI], AX       ;
MOV   [DI], BX       ;
HLT
```

5) Also note down the contents of ***any other registers,*** that you came across while examining the above given registers contents, in table-1 given below.

S.No	Name of Register	Contents	Remarks
1			
2			
3			
4			
5			
6			

Table-1

6) Store ten bytes of data, as given below in table-2 , in memory locations from 350 onwards. Then, after pressing RESET key, check whether the same data is available in the memory or not.

S.No	Memory Address	Data stored	Data available
1	350	45	
2	351	3A	
3	352	55	
4	353	4F	
5	354	78	
6	355	F2	
7	356	67	
8	357	90	
9	358	34	
10	359	20	

Table-2

Review Questions:

3) Were the same contents found in the registers after executing the program in exercise 1 above? And what do you understand by this?

4) Where are the data that were stored with instructions ' MOV [SI], AX and MOV [DI], BX, in the above program, available and why?

इ रि से ट
माइक्रोप्रोसेसर प्रयोगशाला
प्रयोग नं: एम पी 7

IRISET

Microprocessor Laboratory

EXPERIMENT NO.: MP 7

नाम Name	:	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _		
अनुक्रमांक Roll No	:	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _	प्राप्त अंक Marks Awarded	:
पाठ्यक्रम Course	:	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _		
दिनांक Date	:	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _	अनुदेशक का अधाक्षर Instructor Initial	:

Experiment No. MP-7

Verification of The Arithmetic and Logic Functions of 8086

Objective: Verification of the arithmetic functions Addition, Subtraction, Multiplication & Division, and the logic functions OR, AND, XOR & CMP.

Given Data: Use the data given below, for performing these functions.

9367 & **8734**

Addition

1) Load and execute the following program for **addition** of the given values.
2) Then verify the result at 260 onwards and store it in table-1

Program:

```
        MOV  AX, 9367       ; Load  9367  in  to  AX
        MOV  BX, 8734       ; Load  8734  in  to  BX
        MOV  CL,  00        ; Clear  CL Register  for  storing  Carry Flag

        ADD  AX, BX         ; Add  BX  value  to AX

        JNC  Skip           ; If  no carry go to skip
        INC  CL             ; Increment  B  if  there is a CARRY

Skip:   MOV  [0260],  AX    ; Store SUM at  260 & 261 memory  registers
        MOV  [0262],  CL    ; Store  CARRY value  at  262
        HLT                 ; Stop
```

Exercise:

Write and execute programs for performing the below mentioned functions, except addition.

1) Using the same data given above, perform the remaining **ARITHMETIC** operations and store the results and flags in table-1 below

Functions → To be performed	**ADD**	**SUB**	**MUL**	**DIV**
Result				
Flags SET				

Table-1

2) Perform the **LOGIC** operations given below on the data given above and store the results and flags in table-2

Functions → To be performed	**OR**	**AND**	**XOR**	**CMP**
Result				
Flags SET				

Table-2

Review Questions:

1) How did you perform **multiplication** and **division** functions? Briefly explain.
2) Did you get a result value after the compare function? Explain why.

इ रि से ट
माइक्रोप्रोसेसर प्रयोगशाला
प्रयोग नं: एम पी 8

IRISET

Microprocessor Laboratory

EXPERIMENT NO.: MP 8

नाम
Name : ____________ ____

अनुक्रमांक
Roll No : ________________

पाठ्यक्रम
Course : ________________

दिनांक
Date : ________________

प्राप्त अंक
Marks Awarded :

अनुदेशक का अधाक्षर
Instructor Initial :

Experiment No. MP- 8

Addition of An Array of 16 bit Values

Assume that the array is located at memory location, 250, and onwards

Procedure:

1) Enter the program given below into the memory of 8086 kit
2) Store in memory any **five (n) 16 bit values** starting from **250 location** onwards
3) These 16 bit values need to be stored **low byte first** and then in next location **high byte**
4) Execute the program and check the **result** (sum and flag) at 260 onwards

8086 Program:

```
        MOV   SI, 0250        ; Point to the 1st value in memory
        MOV   AX, 0000        ; Clear Accumulator
        MOV   BL, 00          ; Clear BX Register for storing Carry Flag
        MOV   CX, 0005        ; counter ( n)
Again:  ADD   AX, [ SI ]      ; Add one 16 bit value to AX
        JNC   Skip            ; if no carry go to skip
        INC   BL              ; increment B if there is a CARRY
Skip:   INC   SI              ;
        INC   SI              ; increment address in SI to point to next value
        LOOP  Again           ; Repeat addition CX times
        MOV   [0260], AX      ; store SUM at 260 &261 memory registers
        MOV   [0262], BL      ; store CARRY value at 262
        HLT                   ; stop
```

Exercise:

1) In a table show the values of the array you have used and record in it the SUM and CARRY values, you obtained.
2) Perform addition of TEN 16 bit values using the same program.
3) Write a program, (or at least a flowchart) in the space below, to add SIX 8 bit values in the similar manner and verify its working.

Review Questions:

3) In the above program what is the function of LOOP instruction?
4) Why the address in SI register is incremented twice when pointing to next value ?
5) What changes you did in the program for adding ten 16 bit values?

नाम Name	:	____________ ____		
अनुक्रमांक Roll No	:	________________	प्राप्त अंक Marks Awarded	:
पाठ्यक्रम Course	:	________________		
दिनांक Date	:	________________	अनुदेशक का अधाक्षर Instructor Initial	:

Experiment No. MP- 9

Rearranging Bytes in an Array in ASCENDING order

Procedure:

1) Assume that the array is available in memory from location 300 onwards
2) Store the below given program in memory from location 200 onwards
3) Also store ten data bytes of your choice in an array, starting at 300 in memory. Record the same in table-1 given below.
4) Execute the program and then check whether the data bytes are rearranged in ascending order, at location 400 onwards. Record this too in table-1 given below.

8086 Program:

```
        MOV   SI, 0300     ; load data array address into SI
        MOV   DI, 0400     ; initialize DI with destination address of data array
        MOV   BX, 0009     ; counter ( n-1)
        MOV   CX, 0009     ; counter ( n-1)
Again:  MOV   AL, [SI]     ; Copy value at 2050 into A – register
        INC   SI           ; increment address in SI to point to next byte
        MOV   DX, SI ;
Rpt:    CMP   AL, [SI]     ; compare two bytes
        JB    skip         ; jump if AL value is smaller
        MOV   AH, [SI]     ;
        MOV   [SI ], AL    ; bring smaller byte into AL
        MOV   AL, AH;
Skip:   INC   SI           ; increment SI to point to next byte
        LOOP  Rpt          ; repeat same process for remaining bytes
        MOV   [DI], AL     ; store data in ascending order from 400 location
```

```
        INC  DI            ;
        DEC   BX           ; reduce  the  array size by one
        JZ   End           ;
        MOV CX, BX         ; Copy  this reduced array count  into counter
        MOV  SI,  DX       ;
        LOOP  Again        ; repeat the same for remaining bytes of data
End:    HLT                ; end  the program
```

S.No	Memory Location	Stored Data bytes	Memory Location	Rearranged Data Bytes
1	300		400	
2	301		401	
3	302		402	
4	303		403	
5	304		404	
6	305		405	
7	306		406	
8	307		407	
9	308		408	
10	309		409	

Table-1

Exercise:

4) Try to achieve the same thing through a different program logic.
5) Modify the program for rearranging the same data in DESCENDING order and verify its functioning by execution. Mention the modifications you did.

Review Questions:

1) What is the maximum no. of data bytes this program can handle?

	इ रि से ट माइक्रोप्रोसेसर प्रयोगशाला प्रयोग नं: एम पी 10	**IRISET** **Microprocessor Laboratory** **EXPERIMENT NO.: MP 10**

नाम
Name : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

अनुक्रमांक
Roll No : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

प्राप्त अंक
Marks Awarded :

पाठ्यक्रम
Course : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

दिनांक
Date : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

अनुदेशक का अधाक्षर
Instructor Initial :

Experiment No. MP- 10

Interfacing a Relay and a Switch using 8255.

Procedure:

10) Make connections as shown in figure below.
11) Load the program given below and execute it.
12) Now, check whether the operation of relay is being controlled by the switch.
13) When the relay picks up LED glows.

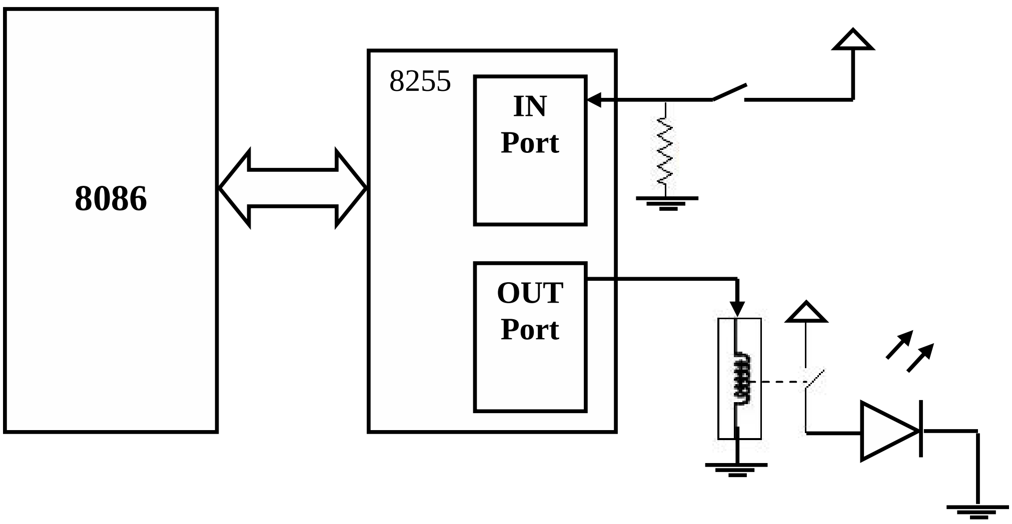


Fig. Interfacing a Switch and a Relay to 8086

Note: Read the required **I/O addresses** from the kit manual

8086 Program:

```
          MOV   DX, CWR address        ; Store Control Word Reg. Addr in DX
          MOV   AL, CW                 ; Load Control Word in to AL
          OUT   DX, AL                 ; Send Control Word to CWR
          MOV   DX, IN-Port address    ; Load Input-port address in DX

Read:     CALL Delay                   ; Wait till switch is ON
          IN  AL, DX                   ;
          CMP  AL, 00                  ;
          JZ  Read                     ;
          MOV  BH, AL                  ;

Confirm:  CALL Delay                   ; If switch is ON
          IN  AL, DX                   ; confirm it once again
          CMP  AL, BH                  ;
          JNE  Read                    ;
        ( NOT AL      )                ; Skip or include this
          MOV  DX, OUT-Port address    ; Send output to relay
          OUT  DX, AL                  ;
          JMP  Confirm                 ;
```

Subroutine For Delay

```
Delay:    MOV  CX, 20FF                ; Delay value
Linger:   MOV  BL, 10                  ;
          LOOP  Linger                 ;
          RET                          ; go to main program
```

Exercise:

1) Execute the program with and without NOT AL instruction and observe the difference in the relay operation.
2) In subroutine reduce CX value (20FF) to 000F and observe its effect on the relay operation.

Review Questions:

1) Mention the difference with and without NOT AL instruction in the program?
2) What was the effect of reducing CX value to 000F, in subroutine?

-

इ रि से ट

माइक्रोप्रोसेसर प्रयोगशाला

प्रयोग नं: एम पी 11

IRISET

Microprocessor Laboratory

EXPERIMENT NO.: MP 11

नाम
Name : _____________ ____

अनुक्रमांक
Roll No : ________________

प्राप्त अंक
Marks Awarded :

पाठ्यक्रम
Course : ________________

दिनांक
Date : ________________

अनुदेशक का अधाक्षर
Instructor Initial :

Experiment No. MP- 11

Software Implemented 3-Aspect Signal

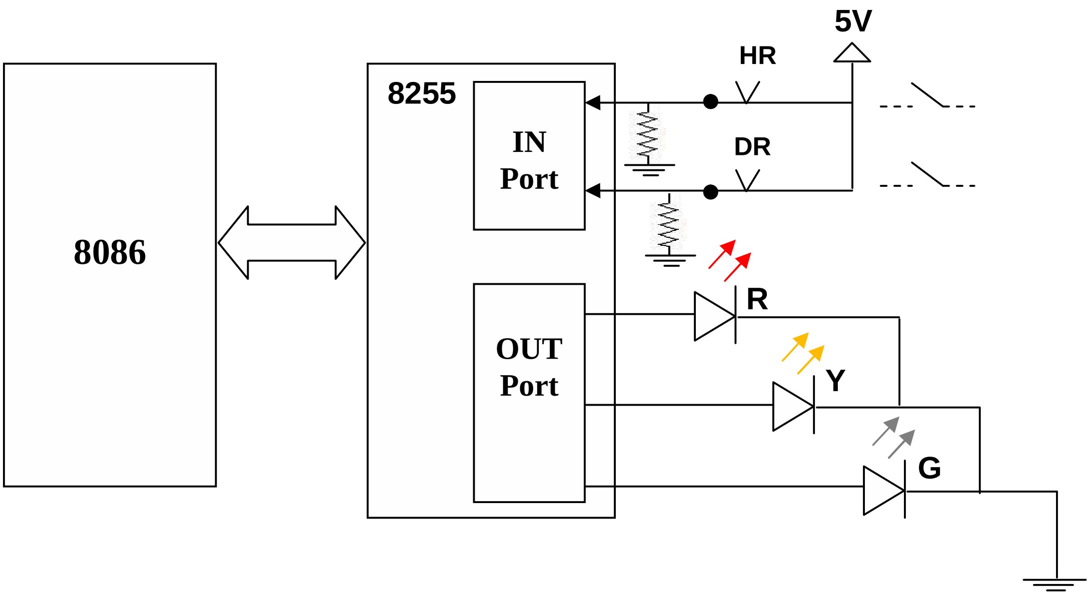


FIG. Simulation of a 3-Aspect Signal

Description

The figure shown above gives a simple example of how 3-aspect signaling can be implemented through a microprocessor based system, for example an **SSI**. In this schematic circuit, only two input parameters HR and DR are used just to explain the concept. But the actual circuit in SSI uses more number of input parameters for controlling the signal aspect.

In this example the aspect of the signal depends on the status of HR and DR relays, which is shown in the table-1 below.

S.No	**HR** relay	**DR** relay	**Signal Aspect**
1	Drop ↓	Drop ↓	**RED**
2	Pickup ↑	Drop ↓	**YELLOW**
3	Pickup ↑	Pickup ↑	**GREEN**

Table-1

Procedure:

14) Make connections as shown in figure given above.
15) Load the program given below and execute it.
16) Vary HR & DR status and check whether the aspects of signal are compliant with the data in table-1.

8086 Program :

```
        MOV   DX, CWR                  ;
        MOV   AL, CW                   ; Initialize 8255
        OUT   DX, AL                   ;
        MOV   DX, OUT Port- address    ;
Red:    MOV  AL, 01                    ;
        OUT  DX, AL                    ;Show Red Aspect first
        MOV   DX, IN Port- address     ;
Read:   IN  AL, DX                     ; Read HR & DR status
        AND  AL, 03                    ;
        MOV   BL, AL                   ;
        CALL Delay                     ; wait for a while
        IN  AL, DX                     ; again read input
        AND  AL, 03                    ;
        CMP   AL, BL                   ; Ensure the Status of HR &DR
        JNE   Read                     ;
        CMP   AL, 01                   ; check whether pro yellow
        JNZ   Green                    ; if not go to check- for- green
        MOV  AL, O2                    ;
        MOV   DX, OUT Port- address    ;
        OUT  DX, AL                    ; Show Yellow Aspect
```

```
          JMP  Read               ;

Green:    CMP   AL,  11           ; check whether pro green
          JNZ   Red               ; if not go to Red aspect
          MOV  AL, O3             ;
          OUT  DX,  AL            ; Show Green Aspect
          JMP  Read               ;
```

Delay Subroutine

```
Delay :   MOV  CX,  20FF          ; Delay value
Linger :  MOV  BL,  10            ;
          LOOP  Linger            ;
          RET                     ;
```

(Note: Read the required **I/O addresses** from the kit manual)

Exercise:

1) Keep HR down and DR up and observe what aspect will you get.
2) Check for fail-safe feature in the program.

Review Questions:

1) In exercise 1 above, did the program implement the fail-safe feature for the signal ? Explain how?
2) What are the input values available at IN port for different status of HR & DR ? And also what output data the microprocessor needs to send for showing the different signal aspects? Show with a table.

इ रि से ट

माइक्रोप्रोसेसर प्रयोगशाला

प्रयोग नं: एम पी 12

IRISET

Microprocessor Laboratory

EXPERIMENT NO.: MP 12

नाम
Name : _____________ ____

अनुक्रमांक
Roll No : ________________

प्राप्त अंक
Marks Awarded :

पाठ्यक्रम
Course : ________________

दिनांक
Date : ________________

अनुदेशक का अधाक्षर
Instructor Initial :

Experiment No. MP- 12 / (MC-5)

Stepper Motor Operation Using 8086 / 8051

Objective : Controlling the Rotation and Operation of a **Stepper Motor through 8086** & **8051** programs

Description:

A **stepper motor** is a brushless, synchronous electric motor that can divide a full rotation into a large number of **steps**.

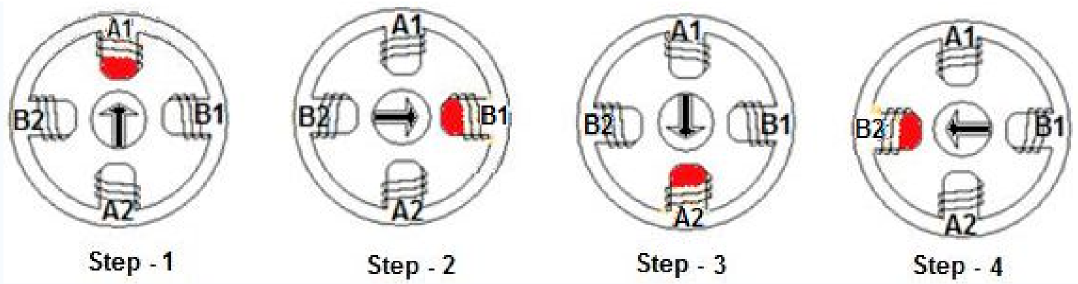


Fig. 1 - Clockwise rotation of the rotor by sequential energization of a single winding at a time in a 4-winding stepper motor.

Stepper motors have multiple windings as shown in the above figure-1. These windings act like electromagnets when energized by dc supply. All these windings are arranged in circular shape and there is a rotor in the center of the circle. Generally, this rotor is a permanent magnet which rotates in the direction of an energized (winding) electromagnet. This rotor is a small toothed wheel which engages with the teeth of a bigger diameter wheel, which has a shaft in the center of it. The electromagnets are energized one after the other in a proper order, by an external control circuit, to make the motor shaft rotate in small **steps.** A large number of such steps make a full rotation of the motor shaft. By controlling the number of steps the motor can be rotated by almost to any precise angle.

S.No.	A1	A2	B1	B2	Hex Equivalent
1	1	0	0	0	8
2	0	0	1	0	2
3	0	1	0	0	4
4	0	0	0	1	1

Table-1 - Step Data for Energization of single- winding at a time in a 4-winding Stepper motor for clockwise rotation in steps

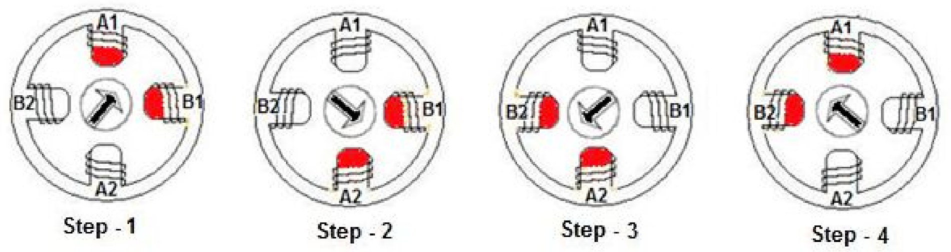


Fig. 2 - Clockwise rotation of the rotor by sequential energization of two adjacent windings at a time in a 4-winding stepper motor.

S.No.	A1	A2	B1	B2	Hex Equivalent
1	1	0	1	0	A
2	0	1	1	0	6
3	0	1	0	1	5
4	1	0	0	1	9

Table-2 - Step Data for Energization of 2- windings at a time in a 4-winding Stepper motor for clockwise rotation

Applications of Stepper Motor:

In computer disk drives, flatbed scanners, computer printers, plotters etc.

Procedure:

8) Connect given stepper motor to any one port of 8255, on the kit
9) Load the program given below
10) Load step data from Table-1 in memory
11) Execute and observe the motor rotation
12) Load the step data in reverse order and observe for reverse rotation
13) Now, load step data from Table-2 above and repeat steps 3 and 4

1) 8086 PROGRAM :

```
        MOV   DX,  CWR                ; Point to CWR in 8255
        MOV   AL,  CW                 ; Initialize 8255 with CW 80
        OUT   DX, AL                  ;
        MOV   AH, 04                  ;
        MOV   DX,  Port-C address     ;

Rpt:    MOV   SI,  Table Address      ; point to step data in table
Step:   CALL  Delay                   ;
        MOV   AL,  [SI]               ; get step data from the table
        OUT   DX, AL                  ; send it to the motor
        INC   SI                      ; point to next step data in the table
        DEC  AH                       ;
        JNZ   Step
        JMP  Rpt
```

Delay Subroutine

```
Delay :  MOV  CX,  20FF               ; Delay value
Linger:  MOV  BL,  25                 ;
         LOOP  Linger                 ;
         RET                          ;
```

2) 8051 PROGRAM:

```
        MOV   81,  #65                ; Initialize stack
        MOV   DPTR, #CWR Address      ; Point to CWR in 8255
        MOV   A,  #CW                 ; Initialize 8255 with CW 80
        MOVX  @DPTR,  A               ;

        MOV   30,  #08                ; Load step data in internal RAM
        MOV   31,  #02                ; Load step data in internal RAM
        MOV   32,  #04                ; Load step data in internal RAM
        MOV   33,  #01                ; Load step data in internal RAM
```

```
         MOV   DPTR,  #Port-A address      ;

Rpt:     MOV   R1, #30                     ;  Point  to step data-table
         MOV   R2, 04                      ;

Send:    MOV   A,  @R1                     ;
         MOVX  @DPTR,  A                   ;  Send step data to motor
         CALL  Delay                       ;  wait till motor takes a step
         INC   R1                          ;  get next step data
         DJNZ  R2, Send                    ;  send all the four step data bytes
         SJMP  Rpt                         ; if four bytes are sent, repeat again
```

Delay Subroutine

```
Delay:   MOV    R3, #FF          ;  delay  value
  Lp2:   MOV    R4, #FF          ; delay  value
  Lp1:   DJNZ   R4,  Lp1         ;
         DJNZ   R3,  Lp2         ;
         RET                     ;Return to main program
```

Exercise:

1) By changing the value of CX / R3 in the delay sub-routine of the above program observe the variations in the motor speed. Try to get visibly step-less smooth rotation.
2) Count the number of steps needed for completing one full rotation of the shaft.

Review Questions:

1) What value in CX / R3 gave a smooth rotation which was visibly step-less?
2) How many steps were needed for one full rotation of the shaft?

इ रि से ट

माइक्रोकन्ट्रोलर प्रयोगशाला

प्रयोग नं: एम सी 1

IRISET

Microcontroller Laboratory

EXPERIMENT NO.: MC-1

नाम
Name : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

अनुक्रमांक
Roll No : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

प्राप्त अंक
Marks Awarded :

पाठ्यक्रम
Course : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

दिनांक
Date : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

अनुदेशक का अधाक्षर
Instructor Initial :

Experiment No. MC-1

Familiarization with 8051 Microcontroller Kit

Objective:

The main objective of this practical session is to introduce to the basic functions needed for practicing programming on microcontroller kits

Description of 8085 Basic Functions:

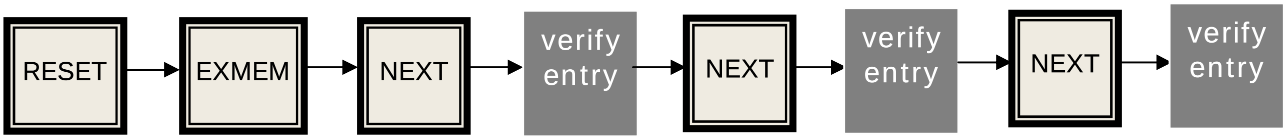
The following basic functions are required to be known by every trainee before being able to practice programming on 8051 kit.

1. **<u>Storing Program In Memory</u>**

a) Press the following keys in the given sequence

RESET → EXMEM → Enter required Memory Address → NEXT → Enter Opcode → NEXT → Enter Opcode or data → NEXT

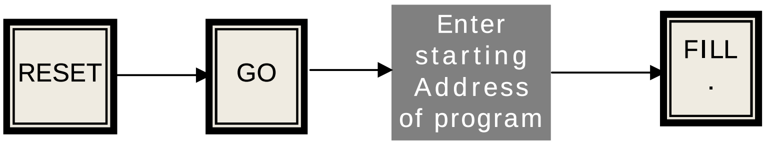
b) When Reset key is pressed the display should be **'UP 51'**
c) Even after entry of last opcode or data byte, the NEXT key must be pressed
d) That means this process should end with the pressing of NEXT key.
e) As an exercise enter the following data starting from 2000 memory register
f) 25, F6, 56, 75, 57, 99, 01
g) Then, verify the correctness of data entered into memory, by the below given key-sequence.
h) After every NEXT key press the entries can be verified.



2. **<u>Execution of Program</u>**

e) <u>Entire Program Execution At One Go</u>

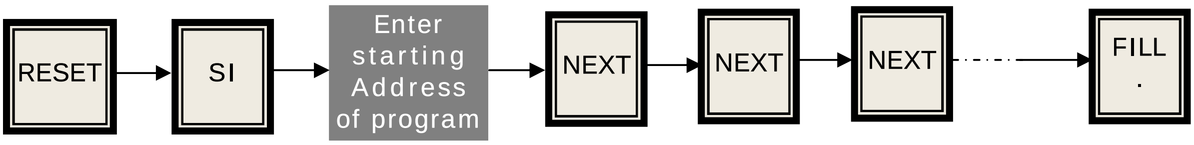
1. In this mode of **program execution** the entire program is executed at one stretch
2. For the execution of a program that is stored in memory by the above sequence of steps follow the sequence shown below



3. After the above sequence of operations the display shows **'E'** which means the program is **executed** or execution is going on.

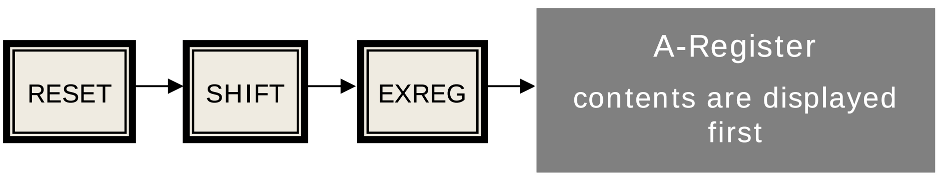
f) <u>Single Instruction/Step (SI) Mode of Operation</u>

1. In this mode the program is executed in steps of single instruction at a time. The below given key sequence is used for execution of program
2. After the execution of every instruction the NEXT key has to be pressed for going to the execution of next instruction or it can be terminated at that point itself with FILL key. Purpose of this process is for debugging a program in case of any errors.



3. **<u>To Examine the Contents of Internal Registers of 8051</u>**

a) Use the fallowing sequence to check the contents of any internal register of µC.



b) Contents of A - register appear on the display

c) By pressing NEXT key other next register contents can be examined one by one.

d) After executing a program in SI mode the contents of internal registers are affected.

Exercise:

1) Perform the following functions

2) Enter the below given program-1 into memory from location 2000H onwards. Execute it as per the sequence given in function procedure **2 (b)**.

3) Then verify the correctness of contents of each internal register of the microcontroller using EXREG function, the function sequence **3**.

Program-1:

```
        MOV   A,    #25           ;
        MOV   DPTR, #2250         ;
        MOV   R0,   #41           ;
        MOV   R1,   #55           ;
        MOV   R2,   #62           ;
        MOV   R3,   #22           ;
        MOV   R4,   #33           ;
        MOV   R5,   #44           ;
        MOV   R6,   #55           ;
        MOV   R7,   #66           ;
        MOV   F0,   #77           ; F0   means  B
        MOV   81,   #88           ; 81   means  SP
        MOV   88,   #99           ; 88   means  TCON
        MOV   89,   #AA           ; 89   means  TMOD
        MOV   D0,   #45           ; D0   means  PSW
        MOV   98,   #23           ; 98   means  SCON
        MOV   A8,   #CC           ; A8   means  IE
        MOV   B8,   #EE           ; B8   means  IP
        MOVX  @DPTR, A            ;
end:    SJMP  end                 ; End of the program
```

2. The below given is a program for the ADDITION of two 8 bit values which are assumed to be available at 2050 and 2051 memory locations. Load this program (and also any two 8 bit numbers at 2050 and 2051) and execute it. Verify and record the results for different 8bit values, as shown in table-1 below.

Program-2:

```
        MOV  DPTR,  # 2050  ; Point  to the 1st value  in memory
        MOV    R1, #00      ; Reserve Register  for  storing  Carry Flag
        MOVX   A,  @DPTR    ; Copy  value  at  2050  into  A – register
        MOV   R2,  A        ; Copy  this value into R1
        INC    DPTR         ; increment address in DPTR  to point to 2nd value.
        MOVX   A,  @DPTR    ; Copy  value  at  2050  into  A – register
        ADD   A, R2         ; add  both the values
        JNC  Skip           ; if no carry go to skip
        INC   R1            ; increment  B  if   there  is  a carry
Skip:   DPTR   #2060        ; load new address in DPTR
        MOVX   @DPTR, A     ; store result at 2060 memory register
        INC    DPTR         ; point  to memory register  2061
        MOV   A,  R1        ; get carry into A
        MOVX   @DPTR, A     ; store carry at 2061 memory register
end:    SJMP   end          ; stop
```

S.No	1st value	2nd value	Sum @2060	Carry@2061
1	22	72		
2	98	83		
3				
4				

Table-1

Review Questions:

1) What is IP register in 8051? Could you find out the contents of it with EXREG function?
2) In the 1st program above, what is the data that is transferred with the instruction ' MOVX @DPTR, A' and where can you find it ?

इ रि से ट
माइक्रोकन्ट्रोलर प्रयोगशाला
प्रयोग नं: एम सी 2

IRISET

Microcontroller Laboratory

EXPERIMENT NO.: MC-2

नाम
Name : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
अनुक्रमांक
Roll No : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
पाठ्यक्रम
Course : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
दिनांक
Date : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

प्राप्त अंक
Marks Awarded :

अनुदेशक का अधाक्षर
Instructor Initial :

Experiment No. MC- 2

Addition of Numbers of an Array

Objective: Summing up of all the numbers of an array using 8051

Program: Assume that the array starting address is **2050** and size is **Ten** bytes.

```
      MOV  DPTR, # 2050  ; Load array starting address into DPTR
      MOV  R1, #00       ; Reserve a Register for storing Carry Flag
      MOV  R2, #09       ; Load array count (n-1) into R2
      MOVX A, @DPTR      ; Move 1st byte of array into A - register
      MOV  R3, A         ; shift this value into R3
Rpt:  INC  DPTR          ; increment address in DPTR to point to next value.
      MOVX A, @DPTR      ; Bring next value of array into A
      ADD  A, R3         ; add both the values
      JNC  Skip          ; if no carry go to 'Skip'
      INC  R1            ; increment R1 if there is a carry
Skip: MOV  R3, A         ;copy the sum into R3
      DJNZ R2, Rpt       ;decrement R2 and jump to 'Rpt' if count is not zero
      MOV  DPTR, #2060   ; load new address in DPTR
      MOVX @DPTR, A      ; store result at 2060 memory register
      INC  DPTR          ; point to memory register 2061
      MOV  A, R1         ; get carry into A
```

```
        MOVX   @DPTR, A    ; store carry at 2061 memory register
end:    SJMP   end         ; stop
```

Procedure:

1) Load the above program into memory in the 8051 kit, starting at 2000.
2) Also load an array of ten byte into memory, starting at 2050.
3) Execute the program in SI (single instruction) mode.
4) And verify the result and carry values at 2060 and 2061 respectively.

Exercise:

1) Increase the array size to 20. Execute the program and verify the results.

Review Questions:

1) Why the count value 09 is used (loaded into R2), in the above program?
2) What can be the maximum size of the array that can be handled by the above program? For arrays above this maximum size, what changes are needed in the program?

इ रि से ट
माइक्रोकन्ट्रोलर प्रयोगशाला
प्रयोग नं: एम सी 3

IRISET

Microcontroller Laboratory

EXPERIMENT NO.: MC- 3

नाम
Name : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
अनुक्रमांक
Roll No : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
पाठ्यक्रम
Course : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
दिनांक
Date : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

प्राप्त अंक
Marks Awarded :

अनुदेशक का अधाक्षर
Instructor Initial :

Experiment No. MC- 3

Flashing 'IRISET' on the Display

Objective: To show the letters IRISET flashing on the display of the kit

Program: Assume that the data for displaying characters IRISET is stored at 2050 and data for a blank display at 2060

```
Start:  MOV   DPTR, #2050   ; Character data address
        LCALL 06F7          ; Call display sub routine at 06F7
        MOV   R1, #FF       ; Flashing delay values
Lp2:    MOV   R2, #FF       ;
Lp1:    DJNZ  R2, Lp1       ;
        DJNZ  R1, Lp2       ;
        MOV   DPTR, #2060   ; Blank data address
        LCALL 06F7          ; Call display sub routine
Lp4:    MOV   R2, #FF       ; Load array count into R2
Lp3:    DJNZ  R2, Lp3       ;
        DJNZ  R1, Lp4       ;
SJMP    Start               ; continue the same
```

Data for Character Display:

2050: **9F F5 9F 49 61 E1**

2060: **FF FF FF FF FF FF**

Procedure:

1) Load the above program into memory in the 8051 kit starting at 2000.
2) Also load the given data for display of IRISET at 2050 and 2060.
3) Execute the program.
4) Now , look for a flashing IRISET on the display

Exercise:

1) Display the name of your Railway zone, (for example: SC RLY) using the same program.
2) Try to decrease the flashing rate of the display by suitably changing the program.

Review Questions:

1) Did you succeed in displaying the name of your railway? If yes, what is the data that you have entered at 2050?
2) What changes did you make in the program for decreasing flashing rate?

इ रि से ट
माइक्रोकन्ट्रोलर प्रयोगशाला
प्रयोग नं: एम सी 4

IRISET

Microcontroller Laboratory

EXPERIMENT NO.: MC- 4

नाम
Name : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
अनुक्रमांक
Roll No : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
पाठ्यक्रम
Course : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
दिनांक
Date : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

प्राप्त अंक
Marks Awarded :

अनुदेशक का अधाक्षर
Instructor Initial :

Experiment No. MC- 4

Finding Out Even and Odd numbers

Objective: To find out whether an 8 bit number is odd or even and also to indicate the same through the display of the kit

Program:

```
       MOV   81, #65           ; Initialize stack
       LCALL 073E              ; Call key- board sub-routine at 073E
       INC   DPTR              ;
       MOVX  A, @DPTR          ;
       ORL   A, #FE            ;
       CJNE A, #FE, Odd        ; check LS bit, if set output Odd
       MOV   DPTR, #201A       ; Point to display Even
SJMP   Out                     ;
Odd:   MOV  DPTR, #2020        ;  Point to display Odd
Out:   LCALL 06F7              ; Call display sub- routine
end:   SJMP  end               ; Stop
```

Data for displaying Even or Odd:

201A:	**61**	**83**	**61**	**D5**	**FF**	**FF**
2020:	**03**	**85**	**85**	**FF**	**FF**	**FF**

Procedure:

1) Load the above program into memory in the 8051 kit starting at 2000.
2) Also load the display data for Even/Odd at 201A and 2020.
3) Execute the program.
4) Then enter a 2-digit number of your choice and press the FILL key.
5) On the display appears either 'Even' or 'Odd' depending on the value entered by you.

Exercise:

1) Make changes in the above program/data to get display of Even and Odd in reverse order. That is NEVE for Even and DDO for odd.
2) Similarly get the display of your **Batch No**. for Even and **Roll No**. for Odd values, by suitably changing the program/data.

Review Questions:

1) Have you got the display of EVEN and ODD in reverse order? If yes how?
2) Write down the data that you have used for the display of RESET and SET in the above exercise.
3) Examine and write down the codes of both key board subroutine, available at 073E, and display subroutine, at 06F7 memory locations, in the kit.