Experiment NO. 7(a)

Aim: Wite an assembly language program using 8086-microprocessor to convert Packed BCD number to Unpacked BCD number.

Software Used: emu8086 Simulator

Program:

```
org 100h
MOV AL,[2000H]
MOV AH,AL
AND AL,0FH
MOV CL,04H
SHR AH,CL
MOV [2001H],AX
Ret
```

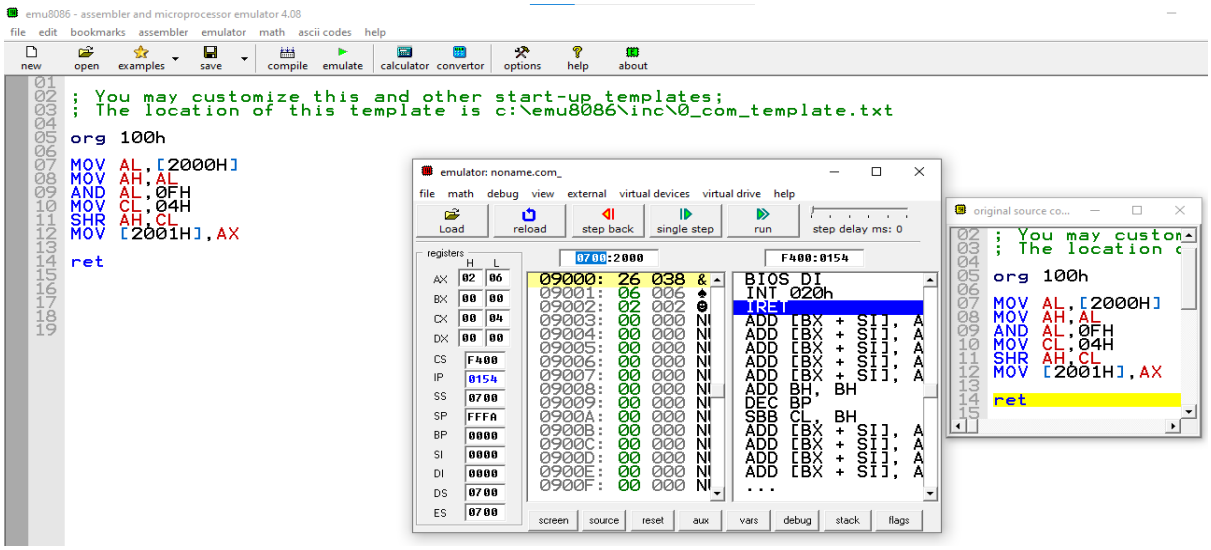
Observation:

Input Value: 2000H = 26

Output Value: 2001H = 06

2002H = 02

Output:



Result: Thus, we have converted Packed BCD number to Unpacked BCD number using 8086-microprocessor.

Experiment NO. 7(b)

Aim: Wite an assembly language program using 8086-microprocessor to convert Packed BCD number to ASCII code.

Software Used: emu8086 Simulator

Program:

```
org 100h
MOV AL,[2000H]
MOV AH,AL
AND AL,0FH
MOV CL,04H
SHR AH,CL
OR AX, 3030H
MOV [2001H],AX
Ret
```

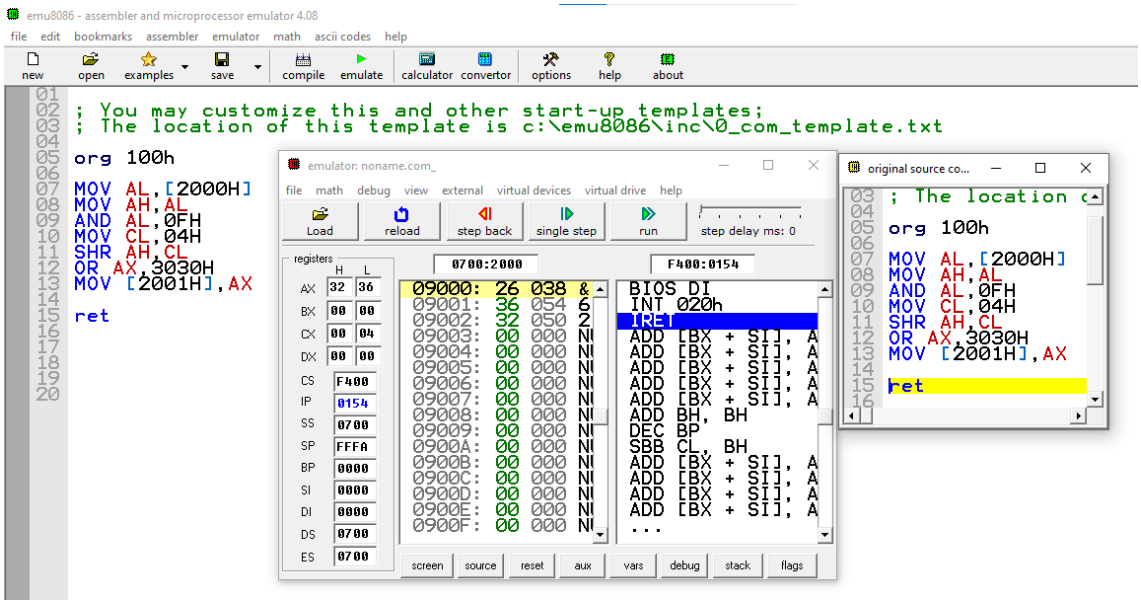
Observation:

Input Value: 2000H = 26

Output Value: 2001H = 36

2002H = 32

Output:



Result: Thus, we have converted Packed BCD number to ASCII code using 8086-microprocessor.

Experiment NO. 8

Aim: Write an Assembly Language Program to interface LCD with 8051 and display Subject Name and Class of Student on it.

Software Used: Keil IDE uvision4

Program:

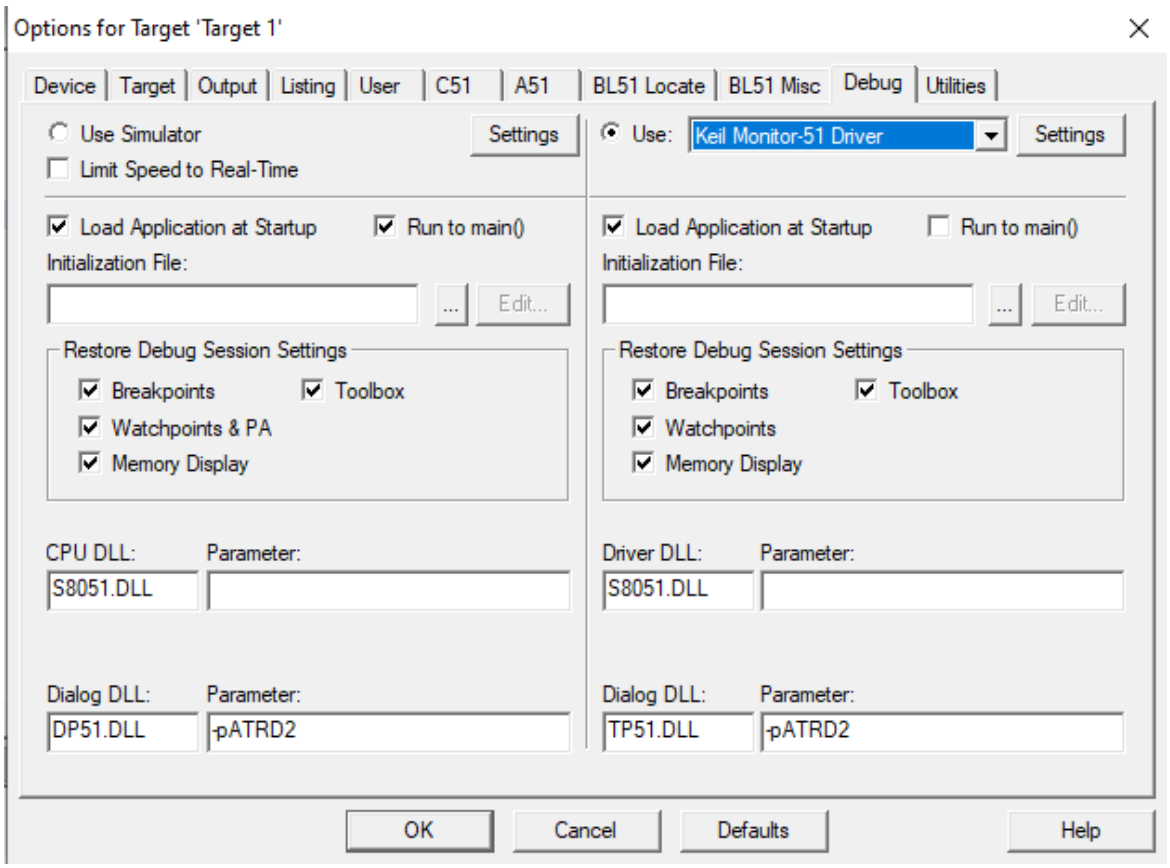
```
PORT_A EQU 0E000H
PORT_B EQU 0E001H
PORT_C EQU 0E002H
CWR EQU 0E003H
ORG 00H
SJMP 30H
ORG 30H
W2:ACALL INIT_LCD
ACALL CLR_LCD
START:MOV DPTR,#MSG
LCALL STRING
MOV A,#0C0H
CALL CMD
MOV DPTR,#MSG1
LCALL STRING
SJMP $
INIT_LCD:MOV DPTR,#CWR
MOV A,#80H
MOVX @DPTR,A
MOV R1,#1FH
ACALL DELAY
MOV R3,#03H
INIT:MOV A,#38H
```

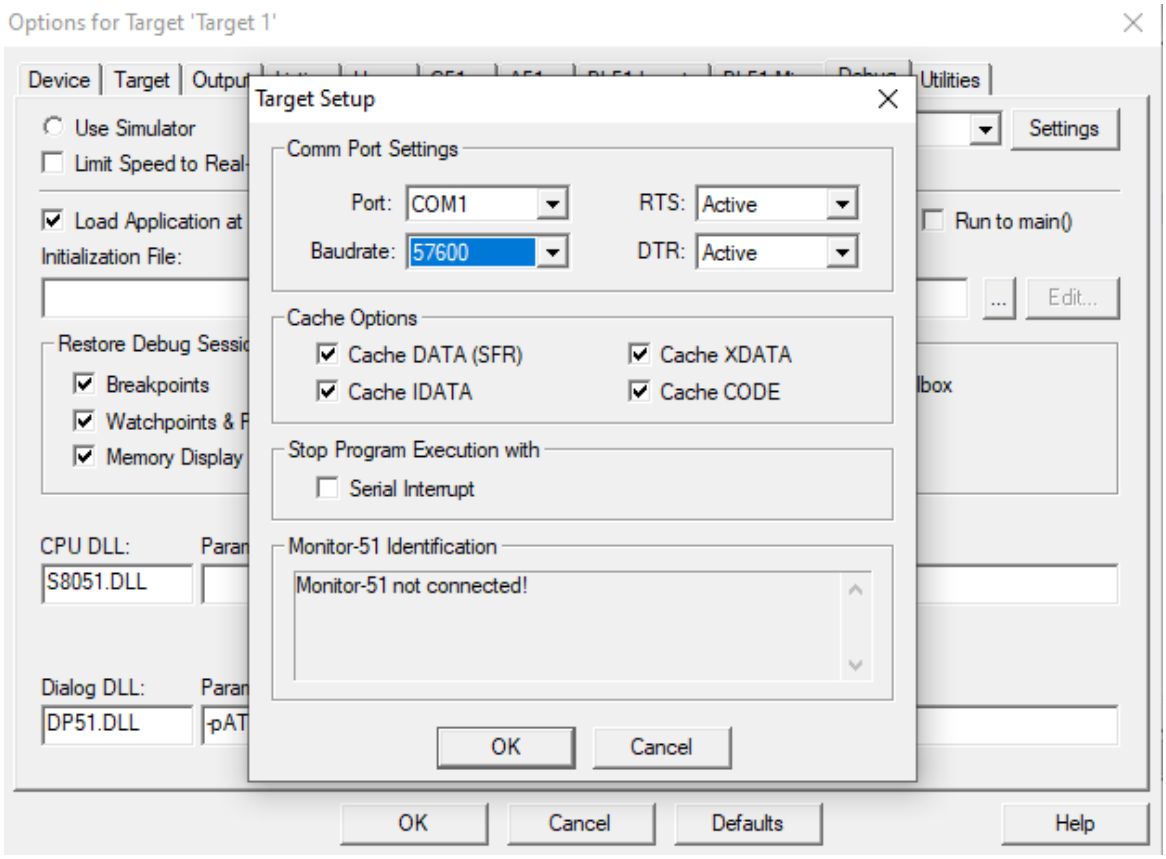
```
LCALL CMD
MOV R1,#3FH
LCALL DELAY
DJNZ R3,INIT
MOV A,#38H
LCALL CMD
MOV A,#0CH
LCALL CMD
RET
CMD:MOV DPTR,#PORT_A
MOVX @DPTR,A
MOV A,#0FBH
MOV DPTR,#PORT_B
MOVX @DPTR,A
MOV A,#0F8H
MOV DPTR,#PORT_B
MOVX @DPTR,A
MOV A,#0FCH
MOV DPTR,#PORT_B
MOVX @DPTR,A
MOV A,#0F8H
MOV DPTR,#PORT_B
MOVX @DPTR,A
MOV R1,#10H
LCALL DELAY
RET
DWRITE:PUSH DPL
```

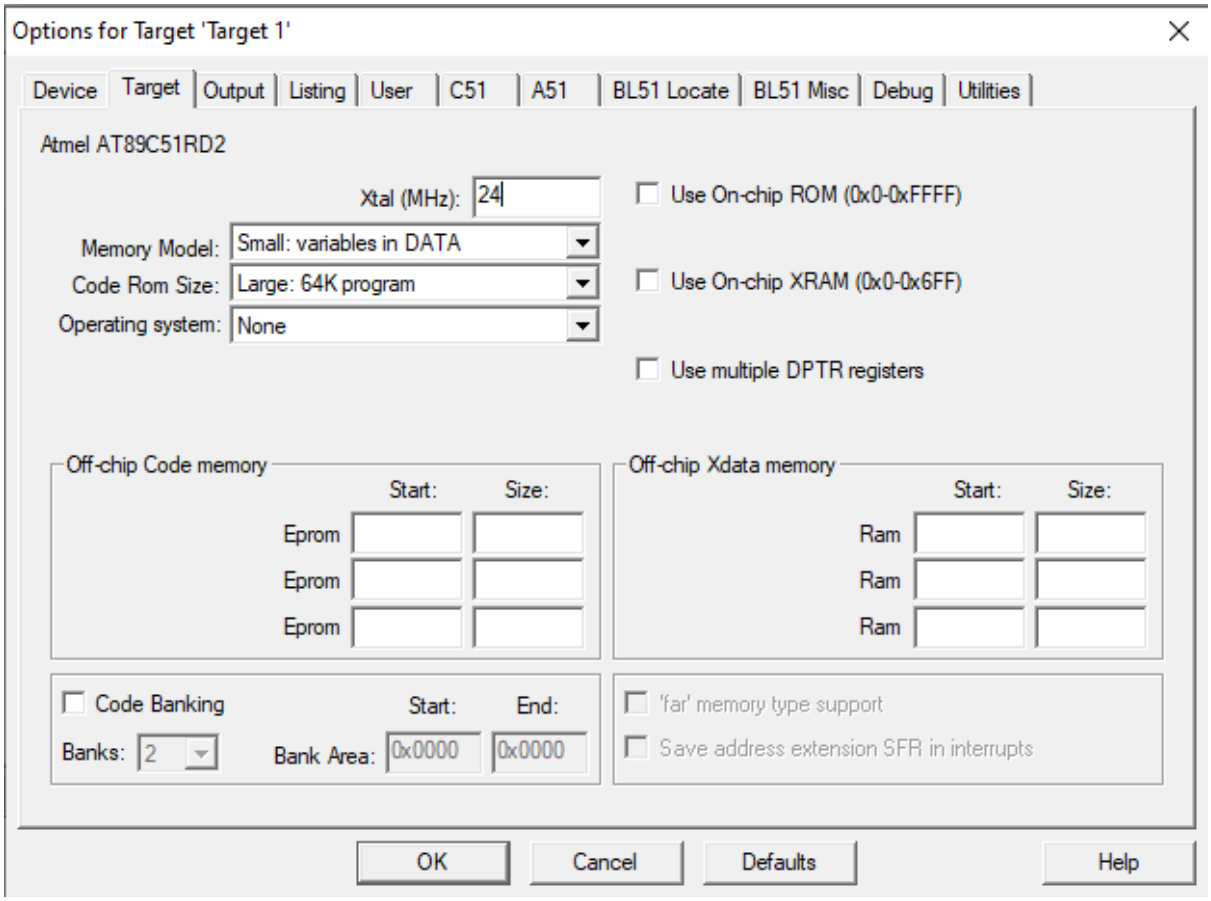
```
PUSH DPH
MOV DPTR,#PORT_A
MOVX @DPTR,A
MOV A,#0FAH
MOV DPTR,#PORT_B
MOVX @DPTR,A
MOV A,#0F9H
MOV DPTR,#PORT_B
MOVX @DPTR,A
MOV A,#0FDH
MOV DPTR,#PORT_B
MOVX @DPTR,A
MOV A,#0F9H
MOV DPTR,#PORT_B
MOVX @DPTR,A
MOV R1,#10H
LCALL DELAY
POP DPH
POP DPL
RET
CLR_LCD:MOV A,#01H
CALL CMD
MOV A,#80H
CALL CMD
RET
DELAY:MOV R2,#02H
DLY:DJNZ R2,DLY
```

```
DJNZ R1,DLY
RET
STRING: CLR A
MOVC A,@A+DPTR
CJNE A,#00,CONTINUE
RET
CONTINUE:ACALL DWRITE
INC DPTR
JMP STRING
ORG 100H
MSG:DB'ECE',00H
MSG1:DB'MPMC',00H
END
```

Output:


Options for Target 'Target 1'
Device
Target
Output
Listing
User
C51
A51
BL51 Locate
BL51 Misc
Debug
Utilities
Use Simulator
Settings
Limit Speed to Real-Time
Use:
Keil Monitor-51 Driver
Settings
Load Application at Startup
Run to main()
Initialization File:
Edit...
Restore Debug Session Settings
Breakpoints
Toolbox
Watchpoints & PA
Memory Display
Watchpoints
CPU DLL:
Parameter:
S8051.DLL
Driver DLL:
Dialog DLL:
DP51.DLL
-pATRD2
TP51.DLL
OK
Cancel
Defaults
Help


Options for Target 'Target 1'
Target Setup
Comm Port Settings
Port:
COM1
RTS:
Active
Baudrate:
57600
DTR:
Cache Options
Cache DATA (SFR)
Cache XDATA
Cache IDATA
Cache CODE
Stop Program Execution with
Serial Interrupt
Monitor-51 Identification
Monitor-51 not connected!
OK
Cancel
Defaults
Help



Result: Thus, Subject Name and Class is displayed on LCD using 8051.

Experiment NO. 9

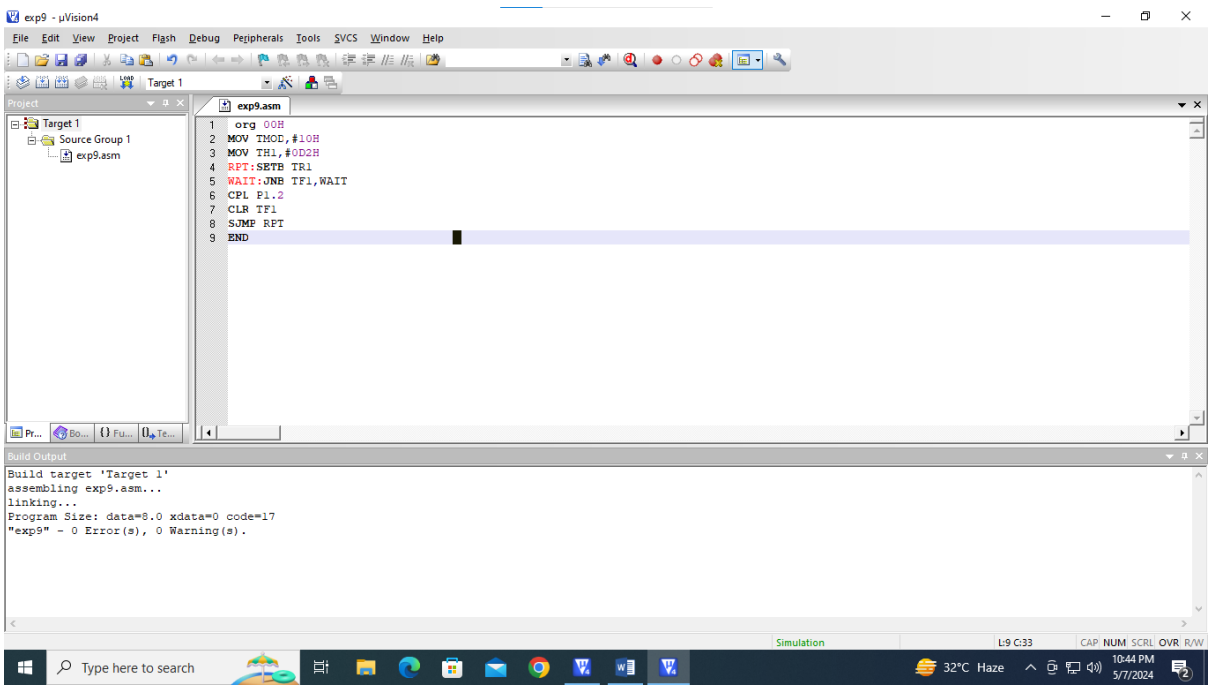
Aim: Write an Assembly Language Program to generate a square wave of 10 KHz using Timer 1 in mode 1 using 8051.

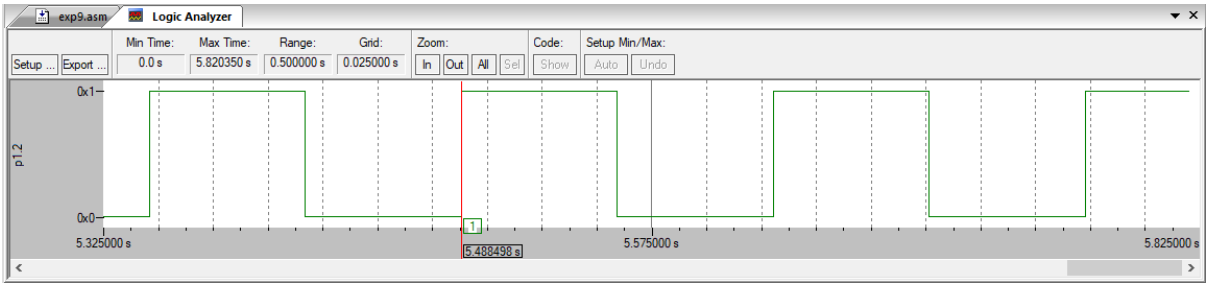
Software Used: Keil IDE uvision4

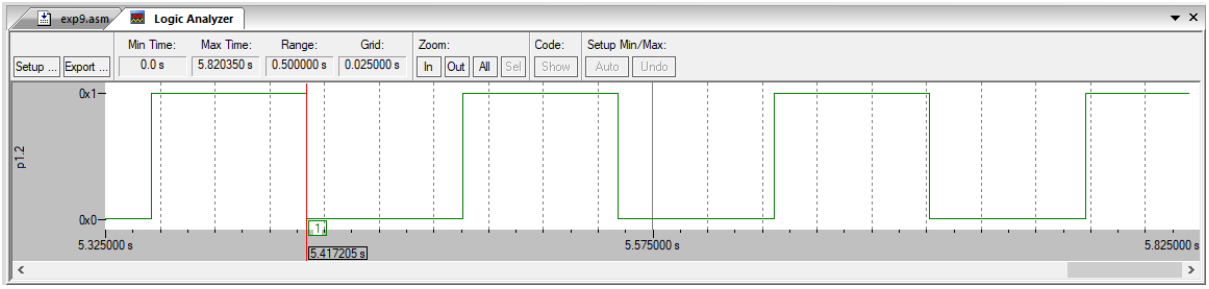
Program:

```
ORG 00H
MOV TMOD,#10H
MOV TH1,#0D2H
RPT:SETB TR1
WAIT:JNB TF1,WAIT
CPL P1.2
CLR TF1
SJMP RPT
END
```

Output:







Result: Thus, a square waveform is generated using Timer 1 of 8051.

Experiment NO. 10

Aim: Design a Minor project using 8051 Microcontroller (Diwali lights with 8 different combinations).

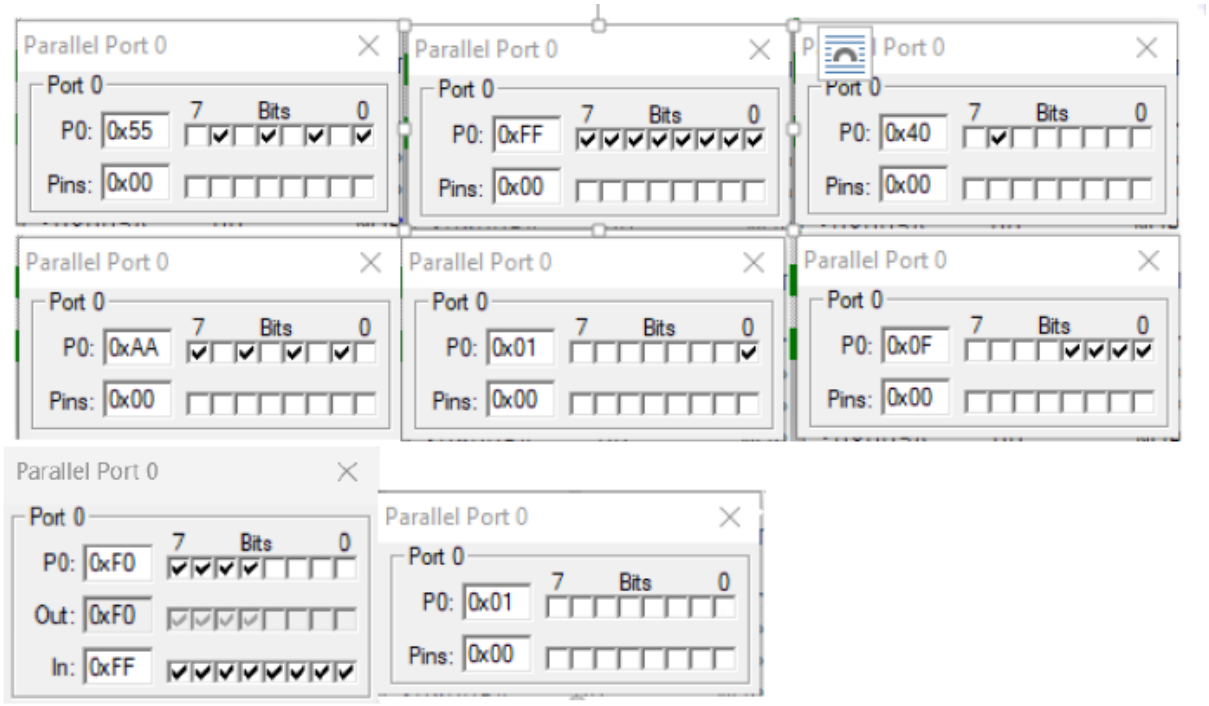
Software Used: Keil IDE uvision4

Program:

```
ORG 0000H
MAIN: MOV P0, #0F0H
ACALL DELAY
MOV P0, #0FH
ACALL DELAY
MOV P0, #AAH
ACALL DELAY
MOV P0, #55H
ACALL DELAY
ACALL DELAY
MOV P0, #00H
ACALL DELAY
MOV P0, #0FFH
ACALL DELAY
ACALL ROT_RGT
ACALL DELAY
ACALL ROT_LFT
ACALL DELAY
LJMP MAIN
ROT_RGT: MOV A, #00H
MOV P0, A
SETB C
MOV R3,#08
```

```
RR1: RRC A MOV P0,A
ACALL DELAY
DJNZ R3, RR1
RET
ROT_LFT: MOV A,#00H
MOV P0, A
SETB C
MOV R3, #08H
RL1: RLC A
MOV P0, A
ACALL DELAY
DJNZ R3, RL1
RET
DELAY: MOV R2, #150H
L3: MOV R1, #255
L2: MOV R0, #255
L1: DJNZ R0, L1
DJNZ R1,L2
DJNZ R2, L3
RET
END
```

Output:



Result: Thus, Diwali lights with 8 different combinations have been shown on port 0.

Experiment NO. 11

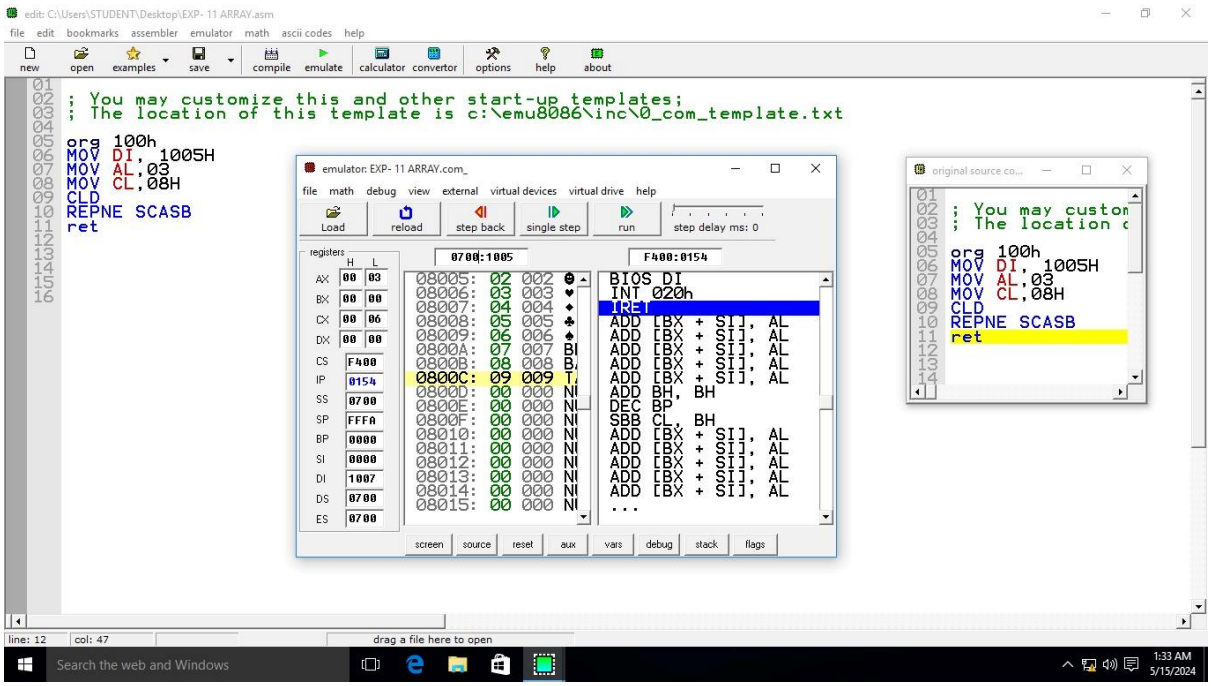
Aim: Write a program in Assembly Language to find a number from an array using string instruction.

Software Used: emu8086 Simulator

Program:

```
ORG 100H

MOV DI,1005H

MOV AL,03

MOV CL,08H

CLD

REPNE SCASB

RET
```

Output:



Result: Thus, the number is found in the array.

Experiment NO. 12

Aim: Design a minor project using 8086 microprocessors (Stepper Motor Execution)

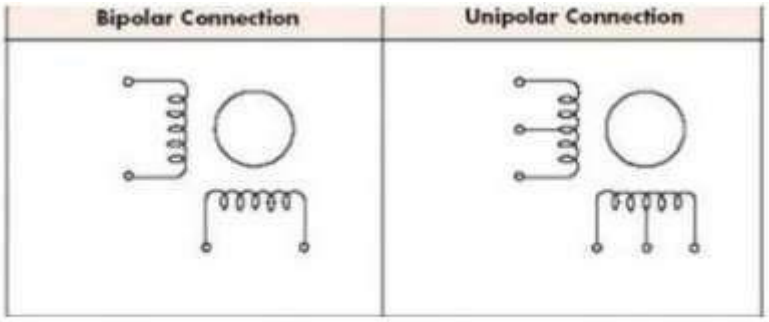
Software Used: emu8086 Simulator

Theory:

Stepper motor is interfaced to 8086 with the help of 8255 IC (Programmable Peripheral Interface). 8255 has two modes of operation BSR (Bit Set Reset) mode & IO (Input Output)mode. The mode is determined by the Control Word (CW) register; (here the CW address is 26). A stepper motor is an electric motor that rotates in discrete step increments. The movement of each step is precise and repeatable; therefore, the motor's position can be controlled precisely without any feedback mechanism, as long as the motor is carefully sized to the application. This type of control eliminates the need for expensive sensing and feedback devices such as optical encoders. The position is known simply by keeping track of the input step pulses. It is one of the most versatile forms of positioning systems. They are typically digitally controlled as part of an open loop system, and are simpler and more rugged than closed loop servo systems. Industrial applications include high speed pick and place equipment and multi-axis CNC machines, often directly driving lead screws or ball screws. In the field of optics, they are frequently used in precision positioning equipment such as linear actuators, linear stages, rotation stages, goniometers, and mirror mounts. Other uses are in packaging machinery, and positioning of valve pilot stages for fluid control systems. Commercially, stepper motors are used in floppy disk drives, flatbed scanners, computer printers, plotters, slot machines, image scanners, compact disc drives and many more devices.

Coil Windings and Stepping Mechanism:

There are two common winding arrangements for the electromagnetic coils: bipolar and unipolar (Fig 4). The described stepping sequence utilizes the bipolar winding. Each phase consists of a single winding. By reversing the current in the windings, electromagnetic polarity is reversed. A unipolar stepper motor has one winding with centre tap per phase. Each section of windings is switched on for each direction of magnetic field. Since in this arrangement a magnetic pole can be reversed without switching the direction of current, the commutation circuit can be made very simple for each winding.



Two common winding arrangements

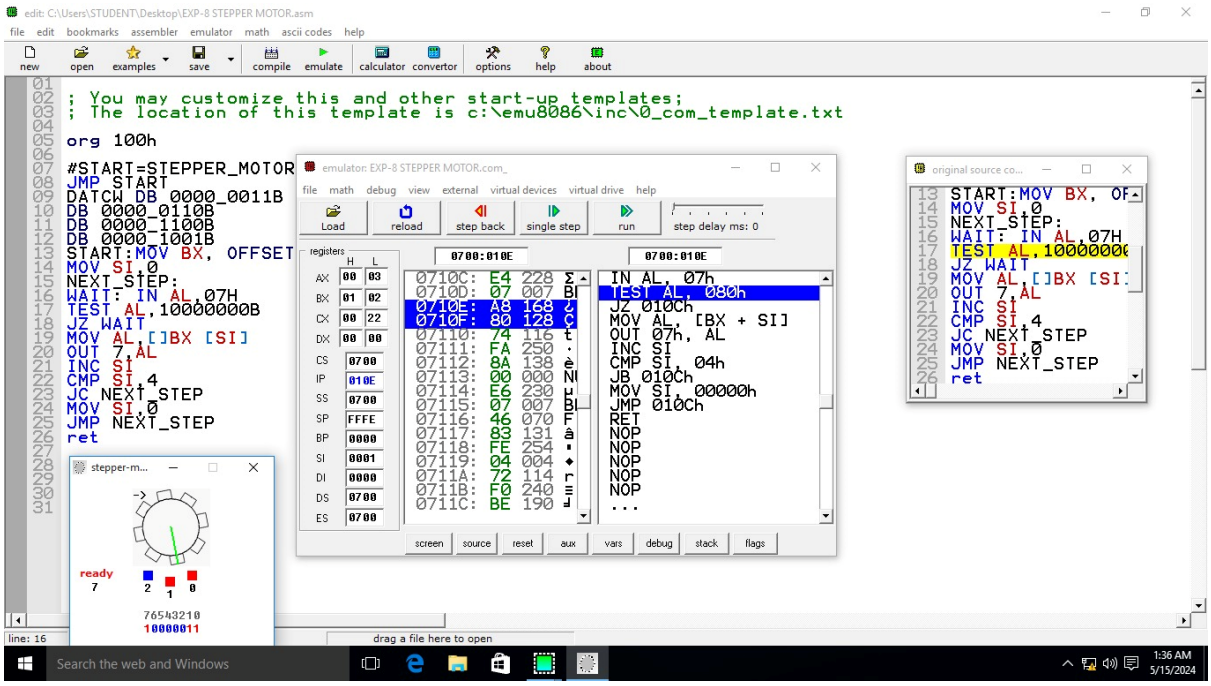
Program:

```
ORG 100H
#START=STEPPER_MOTOR.EXE#
JMP START
DATCW DB 0000_0011B
DB 0000_0110B
DB 0000_1100B
DB 0000_1001B
START: MOV BX,OFFSET DATCW
MOV SI,0
NEXT_STEP:
WAIT: IN AL,07h
TEST AL,10000000B
JZ WAIT
MOV AL,[BX][SI]
OUT 7, AL
INC SI
CMP SI,4
JC NEXT_STEP
MOV SI,0
```

JMP NEXT_STEP

RET

Output:



Result: Thus, the Stepper Motor runs clockwise in direction.