Date: 10/10/2025

EXPERIMENT -8

PROGRAM ON 7 SEGMENT DISPLAY MECHANISM

AIM: To Write an Assembly Language Program to display the digits in port 1 and their positions in port 2.

TOOLS REQUIRED: PC with Keil µvision5

THEORY:

A seven-segment display is an electronic display device used to represent decimal numbers. It consists of seven light-emitting diodes (LEDs) arranged in a figure “8” pattern and labeled as segments a, b, c, d, e, f, g, with an optional decimal point (DP). In this experiment, Port 2 of the 8051 microcontroller is connected to the seven-segment display such that P2.0 → a, P2.1 → b, …, P2.6 → g, and P2.7 → DP.

To display a digit on the seven-segment display, specific segments must be turned ON while others remain OFF. For example, to display the digit ‘0’, segments a, b, c, d, e, f are turned ON, and segment g is OFF. The display can be configured as common-cathode (CC) or common-anode (CA) depending on whether a segment turns ON with a logic ‘1’ or logic ‘0’.

In the common-cathode configuration, a logic ‘1’ sent to a segment pin turns it ON, whereas in common-anode, a logic ‘0’ turns it ON. A lookup table is created in the program to hold the corresponding binary or hexadecimal values for each digit (0–9). When the program runs, it reads the required pattern from the table and outputs it to Port 2, lighting the corresponding segments to form the digit.

This experiment helps students understand I/O port interfacing and pattern generation using lookup tables. The same concept can be extended to multiple digits by multiplexing and refreshing each display position rapidly.

SEVEN SEGMENT DISPLAY – SINGLE MODULE (COMMON CATHODE LOOKUP TABLE):

	P2.7	**P2.6**	**P2.5**	**P2.4**	**P2.3**	**P2.2**	**P2.1**	**P2.0**	
CHAR	**CC**	**g**	**f**	**e**	**d**	**c**	**b**	**a**	**HEX**
0	0	0	1	1	1	1	1	1	0x3F
1	0	0	0	0	0	1	1	0	0x06
2	0	1	0	1	1	0	1	1	0x5B
3	0	1	0	0	1	1	1	1	0x4F
4	0	1	1	0	0	1	1	0	0x66
5	0	1	1	0	1	1	0	1	0x6D
6	0	1	1	1	1	1	0	1	0x7D
7	0	0	0	0	0	1	1	1	0x07
8	0	1	1	1	1	1	1	1	0x7F
9	0	1	1	0	1	1	1	1	0x6F

PROCEDURE:

1. Turn on the computer, create a folder on D drive saved with Register Number.
2. Open Keil µVision5 in desktop or windows, start menu > all programs > open Keil uVision5.

Creating Project:

3. Go to project > click on new µVision project > create a new folder saved with experiment number within the already existed register number folder in D drive mentioned in step 1.Then enter the project name > click on save.
4. A Dialogue Box will appears with name, Select the device for target. In devices, Enter P89C51RD2XX and click on Ok.Then select No for dialog box message "Copy STARTUP.A51 to project folder and add files to project". (or) Choose NXP > to select the device P89C51RD2XX > click on ok > select No for Copy STARTUP.A51 to project folder.

Creating Coding File:

5. Go to file > new > save(choose the path to save the file, It is saved within the name of experiment number folder mentioned in step 3) > enter a file name with extension.asm > save the file.

Linking the Coding File to Project :

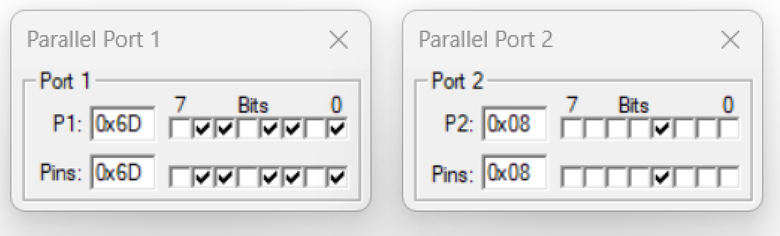
6. Right-click on Source group1 in project bar > Add existing files to source group1 > choose the experiment number folder path and select all files in the folder > select the reqired .asm code file > add > close.
7. Write the assembly language program in .asm code file and save it.

Executing the Code File:

8. Right-click on .asm code file > Build target (to check the errors i.e 0-Errors,0-Warning).
9. Go to debug > Click on Start/Stop Debug Session > click on ok for dialog box message "running code size limit 2K" > and Click on RUN in debug label
10. Observe the output in ports by click on peripherals > I/O port > port 1 and port 2 .

OUTPUTS OBTAINED:

- On peripherals > I/O port > port 1 and port 2.
- Observed change on port when program is running

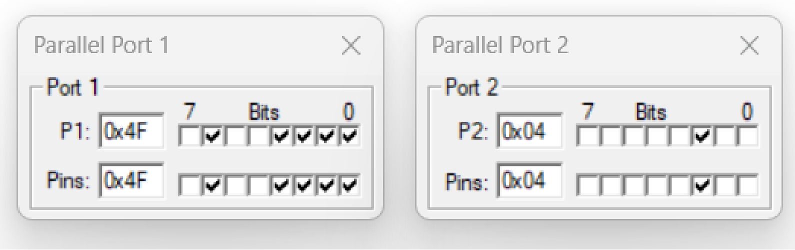


P1: 0x6D □☑☑□☑☑□☑

Pins: 0x6D □☑☑□☑☑□☑

P2: 0x08 □□□□☑□□□

Pins: 0x08 □□□□☑□□□

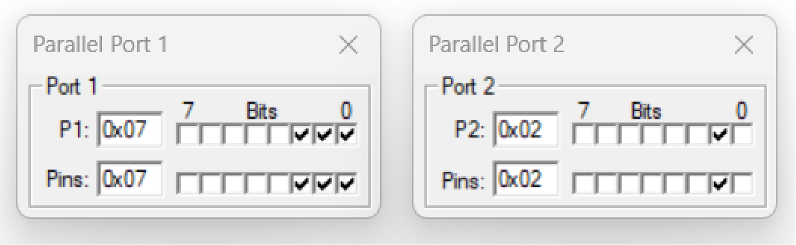


P1: 0x4F □☑□□☑☑☑☑

Pins: 0x4F □☑□□☑☑☑☑

P2: 0x04 □□□□□☑□□

Pins: 0x04 □□□□□☑□□

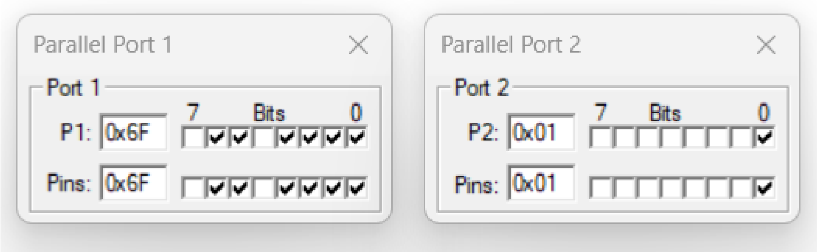


P1: 0x07 □□□□□☑☑☑

Pins: 0x07 □□□□□☑☑☑

P2: 0x02 □□□□□□☑□

Pins: 0x02 □□□□□□☑□



P1: 0x6F □☑☑□☑☑☑☑

Pins: 0x6F □☑☑□☑☑☑☑

P2: 0x01 □□□□□□□☑

Pins: 0x01 □□□□□□□☑

Input Digit	7-Segment Code (Hex)	Explanation
5	6D	Segments a, f, g, c, d glow → displays 5
3	4F	Segments a, b, g, c, d glow → displays 3
7	07	Segments a, b, c glow → displays 7
9	6F	Segments a, b, f, g, c, d glow → displays 9

PROGRAMS:

ADDRESS	OPCODES	LABELS	MNEMONICS	OPERANDS
0400			ORG	0400H
0400	3F065B4F666D7D0 77F6F00		DB	3FH,06H,5BH,4FH,66H, 6DH,7DH,07H,7FH,6FH,00H
0000			ORG	0000H
0000	900400		MOV	DPTR,#0400H
0003		AGAIN		
0003	7405		MOV	A,#05H
0005	93		MOVC	A,@A+DPTR
0006	75A008		MOV	P2,#08H
0009	F590		MOV	P1,A
000B	112D		ACALL	DELAY
000D	7403		MOV	A,#03H
000F	93		MOVC	A,@A+DPTR
0010	75A004		MOV	P2,#04H
0013	F590		MOV	P1,A
0015	112D		ACALL	DELAY
0017	7407		MOV	A,#07H
0019	93		MOVC	A,@A+DPTR
001A	75A002		MOV	P2,#02H
001D	F590		MOV	P1,A
001F	112D		ACALL	DELAY
0021	7409		MOV	A,#09H
0023	93		MOVC	A,@A+DPTR
0024	75A001		MOV	P2,#01H
0027	F590		MOV	P1,A
0029	112D		ACALL	DELAY
002B	80D6		SJMP	AGAIN
002D		DELAY		
002D	7890		MOV	R0,#90H
002F	79FF	WAIT1	MOV	R1,#0FFH
0031	7AFF	WAIT2	MOV	R2,#0FFH
0033	DAFE	WAIT3	DJNZ	R2,WAIT3
0035	D9FA		DJNZ	R1,WAIT2
0037	D8F6		DJNZ	R0,WAIT1
0039	22		RET	
			END	

RESULT: The Assembly Language Program to display the digits in port 1 and their positions in port 2 is successfully executed.