Date: 25/07/2025

EXPERIMENT-1(B)

AIM: Write a following assembly language programs (ALP) using logical instructions in 8051 Micro Controller.

a) Basic logical Operations
b) Packed BCD to ASCII Conversion
c) ASCII to Packed BCD Conversion
d) Counting the number of ones in Hexadecimal Number

TOOLS REQUIRED: PC, Keil µvision5

THEORY:

In 8051 microcontroller programming, logical instructions are used to manipulate data at the bit level. These instructions include AND, OR, XOR, and CPL. They allow the programmer to set, clear, toggle, or mask bits in registers or memory without affecting the other bits. Logical operations are important in controlling hardware, checking or setting flags, performing bitwise calculations, and making decisions in the program. They form the foundation for most embedded system operations, especially when handling input/output ports and controlling devices.

The packed BCD to ASCII conversion is used when numeric data stored in packed BCD format (two digits per byte) needs to be displayed on a screen or sent over a serial interface. The process separates the high nibble and low nibble of the byte and adds 30h to convert them into readable ASCII characters. Conversely, ASCII to packed BCD conversion is used to take numeric input in ASCII form and convert it back to packed BCD so it can be used in calculations or stored efficiently. Another common logical program is counting the number of ones in a hexadecimal number, which is often used in parity checking, error detection, or analyzing bit patterns. This is done by checking each bit of the number, usually by masking and shifting, and increasing a counter whenever a 1 is found.

These programs demonstrate how 8051 microcontroller logical instructions can be applied for data manipulation, conversion, arithmetic preparation, and bit-level control. Understanding these basic operations is crucial for writing efficient and effective embedded system programs.

Logical instructions in 8051 are also widely used in looping, decision-making, and controlling program flow. For example, instructions like DJNZ (Decrement and Jump if Not Zero) are combined with logical operations to create loops that process data repeatedly until a condition is met. Using these instructions with registers and memory allows the microcontroller to efficiently handle arrays, perform arithmetic on multiple values, or process input from sensors. By mastering logical operations and bit manipulation, a programmer can write programs that are both compact and fast, which is essential in embedded systems where memory and processing power are limited.

PROCEDURE:

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

Creating Project:

3. Go to project ->click on new uVision project -> create a new folder saved with experiment number within the already existed register number folder in D drive mentioned in step 1, -> enter the project name -> click on save.
4. Select the device for target -> In devices ->Enter P89C51RD2XX in Search toolbar -> click on ok -> 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 -> click on new->go to 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 .asm code file ->click on add-> click on close.
7. Write the assembly language program in .asm code file and save it.

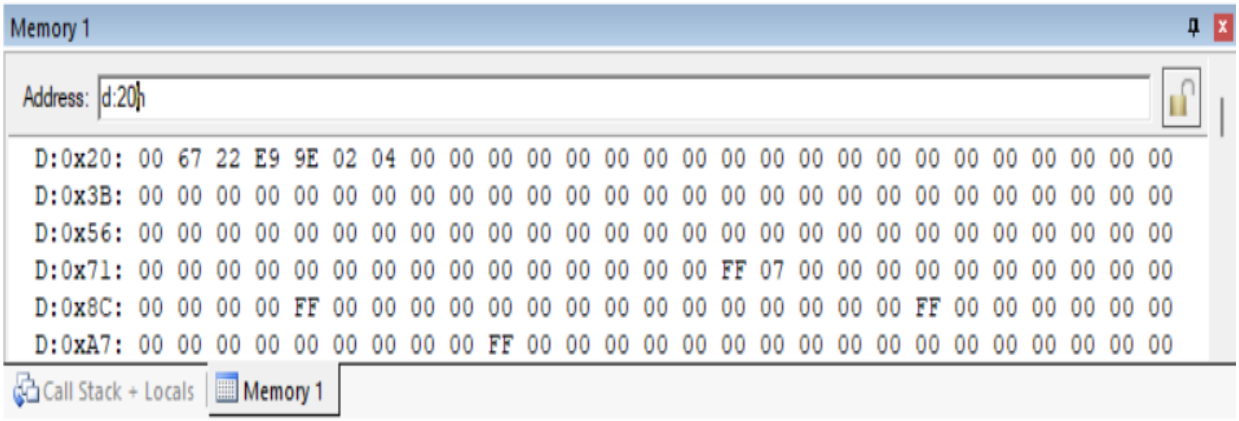
Executing the Code File:

8. Right-click on .asm code file->Click on 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 Register windows, Memory windows, Serial window.

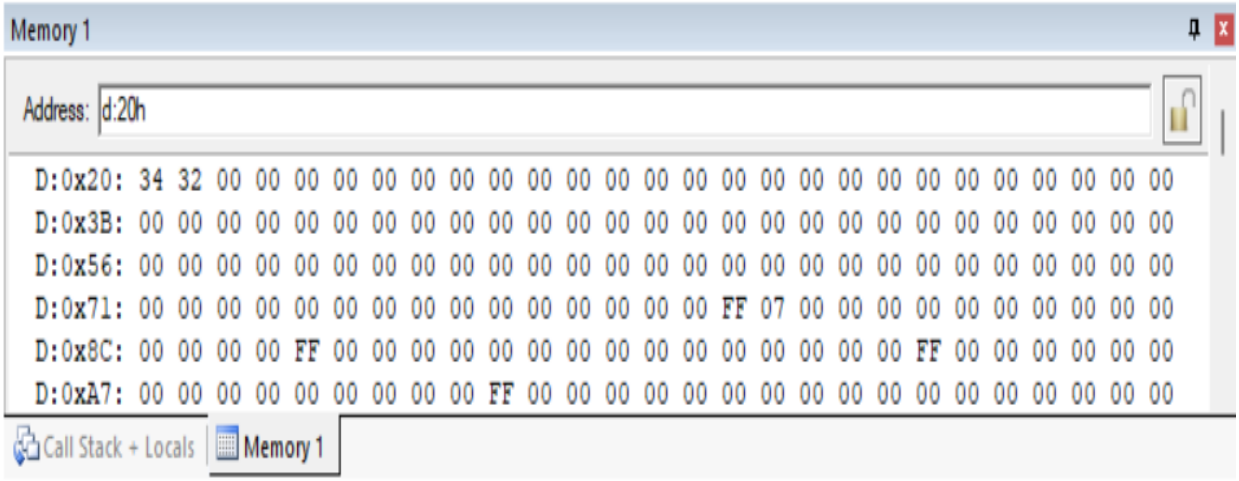
OUTPUT:

View -> Memory Windows -> memory 1

a) Basic logical Operations:



b) Packed BCD to ASCII Conversion:



PROGRAMS:

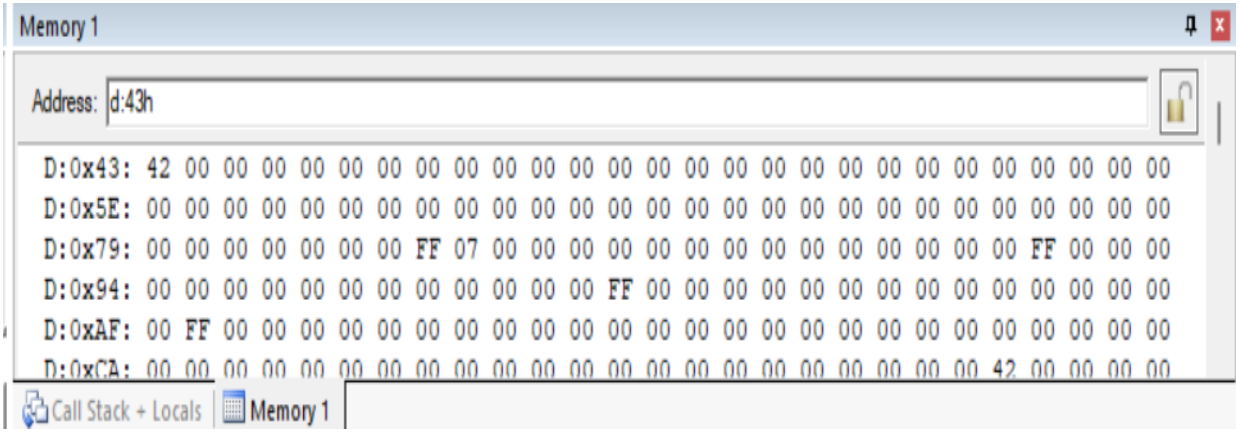
a) Basic logical Operations:

ADDRESS	OPCODES	LABELS	MNEMONICS	OPERANDS
0000	7867		MOV	R0,#67H
0002	58		ANL	A,R0
0003	F520		MOV	20H,A
0005	7445		MOV	A,#45H
0007	48		ORL	A,R0
0008	F521		MOV	21H,A
000A	7404		MOV	A,#04H
000C	7445		MOV	A,#45H
000E	68		XRL	A,R0
000F	F522		MOV	22H,A
0011	7416		MOV	A,#16H
0013	F4		CPL	A
0014	F523		MOV	23H,A
0016	C4		SWAP	A
0017	F524		MOV	24H,A
0019	7404		MOV	A,#04H
001B	03		RR	A
001C	F525		MOV	25H,A
001E	23		RL	A
001F	F526		MOV	26H,A
			END	

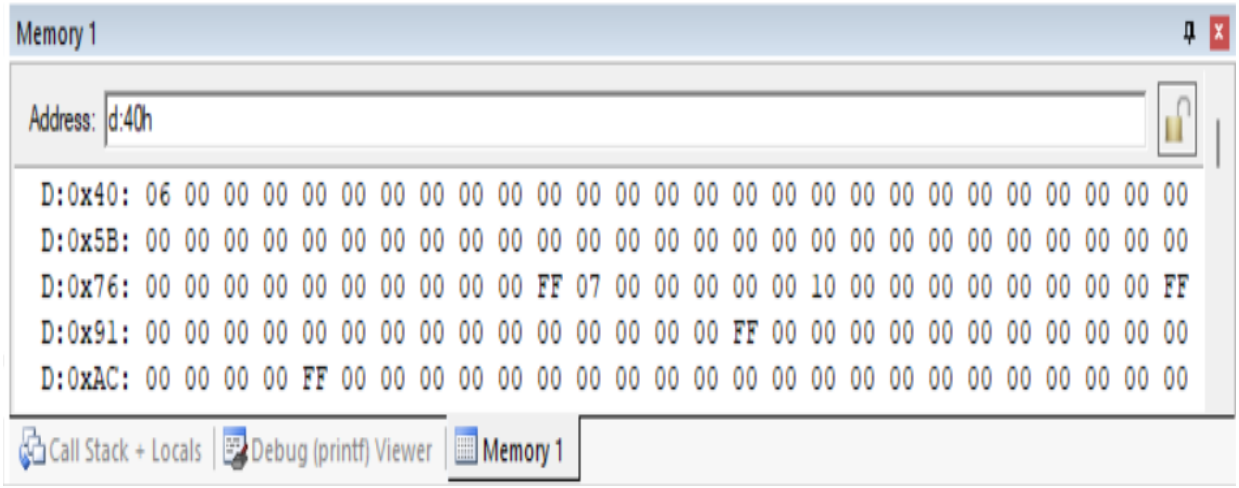
b) Packed BCD to ASCII Conversion:

ADDRESS	OPCODES	LABELS	MNEMONICS	OPERANDS
0000	74 42		MOV	A, #42H
0002	C4		SWAP	A
0003	54 0F		ANL	A, #0FH
0005	24 30		ADD	A, #30H
0007	F8		MOV	20H, A
0008	74 42		MOV	A, #42H
000A	54 0F		ANL	A, #0FH
000C	24 30		ADD	A, #30H
000E	F9		MOV	21H, A
000F			END	

c) ASCII to Packed BCD Conversion:



d)Counting the number of ones in Hexadecimal Number:



c)ASCII to Packed BCD Conversion :

ADDRESS	OPCODES	LABELS	MNEMONICS	OPERANDS
0000			ORG	00H
0000	75 F3 34		MOV	R0, #34H
0003	75 F2 32		MOV	R1, #32H
0006	92		MOV	A, R0
0007	75 F0 30		SUBB	A, #30H
000A	SWAP		SWAP	A
000B	75 F0 34		MOV	R2,A
000D	93		MOV	A,R1
000E	F5 40		SUBB	A,#30H
0010	75 F0 30		ORL	A,R2
0013	75 F0 02		MOV	43H,A
0015	F5 41		END	

d)Counting the number of ones in Hexadecimal Number:

ADDRESS	OPCODES	LABELS	MNEMONICS	OPERANDS
0000			ORG	00H
0000	7808		MOV	R0,#08H
0002	7900		MOV	R1,#00H
0004	749F		MOV	A,#9FH
0006	C3	L1	CLR	C
0007	33		RLC	A
0008	5001		JNC	L2
000A	09		INC	R1
000B	D8F9	L2	DJNZ	R0,L1
000D	8940		MOV	40H,R1
			END	

RESULT:

a) Assembly language programs (ALP) using logical instructions in 8051 Micro Controller for Basic Logical Operations (AND, OR, XOR, NOT) successfully executed.

b) Assembly language programs (ALP) using logical instructions in 8051 Micro Controller to convert Packed BCD to ASCII successfully executed.

c) Assembly language programs (ALP) using logical instructions in 8051 Micro Controller to convert ASCII to Packed BCD successfully executed.

d)Assembly language programs (ALP) using logical instructions in 8051 Micro Controller to count the number of ones in a hexadecimal number successfully executed.