COURSE LABORATORY MANUAL

A. LABORATORY OVERVIEW

Degree:	BE	Programme:	EC
Semester:	IV	Academic Year:	2024-25
Laboratory Title:	Microcontroller Laboratory	Laboratory Code:	BECL456A
L-T-P-S:	0-0-2-0	Duration of SEE:	2 Hrs
Total Contact Hours:	24	SEE Marks:	50
Credits:	1	CIE Marks:	50
Lab Manual Author:	Mahabaleshwara Bhat P	Sign	Dt : 25/01/25
Checked By:	Rajani Rai B	Sign	Dt : 25/01/25

*The SEE will be conducted for 100 marks and proportionally reduced to 60 marks.

B. DESCRIPTION

1. PREREQUISITES:

- Basic Electronics(BBEE103)
- Digital System Design using Verilog(BEC302)

2. BASE COURSE:

- Microcontrollers-BEC405A

3. COURSE OUTCOMES:

At the end of the course, the student will be able to;

1. Write Assembly language programs to perform data transfer, arithmetic and logical operations and use simulation tools for programming and building a microcontroller system
2. Apply the concepts of timers and counter for building counters.
3. Develop microcontroller based system using C programming.
4. Interface hardware devices to 8051 and verify/control the functionality

4. RESOURSES REQUIRED:

- Software resources: Keil uVision3
- Hardware resources: Personal Computer,CRO, DC power supply, Interfacing kit, Stepper motor, connecting cables (RS 232), communication bus(JTAG)

Prepared by: Mahabaleshwara Bhat P Checked by : Rajani Rai B HOD

COURSE LABORATORY MANUAL

5. RELEVANCE OF THE COURSE:

- Embedded Systems
- Data structuire using C

6. GENERAL INSTRUCTIONS:

- Students should come prepared by learning the theory, instuctions to be used and circuit diagram (if any)
- Students should sign in the LOG REGISTER while entering and leaving the laboratory
- Students should come with observation and record note book to the laboratory
- While performing the experiments, handle the equipments with care. Any breakage should immediately be reported
- After completing the laboratory exercise, make sure to shutdown the system properly and return the lab equipments to concerned staff

7. CONTENTS:

Expt No.	Title of the Experiments	RBT	CO
	I. Assembly Language Programming		
IA. Data Transfer Programs:			
1	Write an ALP to move a block of n bytes of data from source (20h) to destination (40h) using Internal RAM	L3	CO1
2	Write an ALP to move a block of n bytes of data from source (2000h) to destination (2050h) using External RAM	L3	CO1
3	Write an ALP To exchange the source block starting with address 20h, (Internal RAM) containing N (05) bytes of data with destination block starting with address 40h (Internal RAM).	L3	CO1
4	Write an ALP to exchange the source block starting with address 10h (Internal memory), containing n (06) bytes of data with destination block starting at location 00h (External memory).	L3	CO1
IB. Arithmetic & Logical Operation Programs:			
5	Write an ALP to add the byte in the RAM at 34h and 35h, store the result in the register R5 (LSB) and R6 (MSB), using Indirect Addressing Mode.	L3	CO1
6	Write an ALP to subtract the bytes in Internal RAM 34h & 35h store the result in register R5 (LSB) & R6 (MSB)	L3	CO1
7	Write an ALP to multiply two 8-bit numbers stored at 30h and 31h and store16- bit result in 32h and 33h of Internal RAM.	L3	CO1
8	Write an ALP to perform division operation on 8-bit number by 8-bit number	L3	CO1
9	Write an ALP to separate positive and negative in a given array.	L3	CO1
10	Write an ALP to separate even or odd elements in a given array.	L3	CO1
11	Write an ALP to arrange the numbers in Ascending & Descending order.	L3	CO1
12	Write an ALP to find Largest & Smallest number from a given array starting from 20h & store it in Internal Memory location 40h	L3	CO1

IC. Counter Operation Programs:			
13	Write an ALP for Decimal UP-Counter.	L3	CO2
14	Write an ALP for Decimal DOWN-Counter	L3	C02
15	Write an ALP for Hexadecimal UP-Counter	L3	C02
16	Write an ALP for Hexadecimal DOWN-Counter.	L3	CO2
	II. C Programming		
17	Write an 8051 C program to find the sum of first 10 Integer Numbers.	L3	CO3
18	Write an 8051 C program to find Factorial of a given number.	L3	CO3
19	Write an 8051 C program to find the Square of a number (1 to 10) using Look-Up Table.	L3	CO3
20	Write an 8051 C program to count the number of Ones and Zeros in two consecutive memory locations.	L3	CO3
	III. Hardware Interfacing Programs		
21	Write an 8051 C program to Generate Sine & Square waveforms using DAC interface.	L3	CO4
22	Write an 8051 C Program to rotate stepper motor in Clock & Anti-Clockwise direction.	L3	CO4
23	Open ended experiment - 1	L3	CO4
24	Open ended experiment - 2	L3	CO3

8. REFERENCE:

Suggested Learning Resources:

1. "The 8051Microcontroller: Hardware, Software and Applications", V Udayashankara and M S Mallikarjuna Swamy, McGraw Hill Education,1st edition, 2017.

-

C. EVALUATION SCHEME

For CBCS 2022 scheme:

1. Laboratory Components: 30 Marks (Laboratory Components will be evaluated for 50 and converted to 30 Marks. Observation Writeup – 10 Marks + Lab Conduction – 10 Marks + Viva-Voce – 10 Marks + Record Writing – 20 Marks).
2. Laboratory IA tests: 100 Marks which will be converted to 20Marks
3. Continuous Internal Evaluation (CIE) = 30 + 20 = 50 Marks.
4. *SEE:* 50* Marks
   (*The SEE will be conducted for 100 marks and proportionally reduced to 50 marks)

-

D1. ARTICULATION MATRIX

Mapping of CO to PO

	POs											
COs	1	2	3	4	5	6	7	8	9	10	11	12
1.Write Assembly language programs to perform data transfer, arithmetic and logical operations and use simulation tools for programming and building a microcontroller system	2	2	2	2	2	-	-	2	-	-	2	2
2.Apply the concepts of timers and counter for building counters	2	2	2	2	-	-	-	-	-	-	2	2

3.Develop microcontroller based system using C programming.	3	3	3	3	-	-	-	-	-	-	3	3
4.Interface hardware devices to 8051 and verify/control the functionality	3	3	3	3	2	-	-	-	2	-	3	3

Note: Mappings in the Tables D1 (above) and D2 (below) are done by entering in the corresponding cell the Correllation Levels in terms of numbers. For Slight (Low): 1, Moderate (Medium): 2, Substantial (High): 3 and for no correllation: " - ".

D2. ARTICULATION MATRIX CO v/s PSO

Mapping of CO to PSO		
	PSOs	
COs	1	2
1. Write Assembly language programs to perform data transfer, arithmetic and logical operations and use simulation tools for programming and building a microcontroller system	2	-
2. Apply the concepts of timers and counter for building counters.	2	-
3. Develop microcontroller based system using C programming.	2	-
4. Interface hardware devices to 8051 and verify/control the functionality	2	-

-

E. EXPERIMENTS

1. EXPERIMENT NO: IA

2. TITLE: ASSEMBLY LANGUAGE PROGRAMMING

3. LEARNING OBJECTIVES:
   - To manipulate data using the registers and MOV instruction
   - Assemble and run data transfer program
   - Apply different methods for data transfer and data exchange

4. AIM: Write an 8051 assembly level program to

1.Move a block of n bytes of data from source (20h) to destination (40h) using Internal RAM

2.Move a block of n bytes of data from source (2000h) to destination (2050h) using External RAM

3.Exchange the source block starting with address 20h, (Internal RAM) containing N (05) bytes of data with destination block starting with address 40h (Internal RAM)

4.Exchange the source block starting with address 10h (Internal memory), containing n (06) bytes of data with destination block starting at location 00h (External memory).

5. MATERIAL / EQUIPMENT REQUIRED:
   - Software resources: -Keil uVision3

6. THEORY / HYPOTHESIS:
   - DATA TRANSFER INSTRUCTIONS: Data transfer instructions moves the content from a register/memory location to another register/memory location. It has the following format : **8051-MOV destination, source** ; copy source to destination
   - This instruction tells CPU to move the content from source operand to the destination operand
   - After execution, the content of destination operand is replaced by source operand whereas, source operand content will remain the same

Examples:

1. MOV A,Rn; Moves the content from a register to accumulator
2. MOV A,direct; Moves the content from a memory location in RAM to accumulator
3. MOV A,@Ri; Moves the content from RAM location (Ri holds address) to accumulator
4. MOV A,#data; Moves the immediate data to accumulator
5. MOV Rn,A; Moves the content from accumulator to a register
6. MOV Rn,direct; Moves the content from a location in RAM to a register
7. MOV Rn,#data; Moves the immediate data to a register
8. MOV direct,A; Moves the content fom accumulator to a location in RAM
9. MOV direct,Rn; Moves the content from a register to a location in RAM
10. MOV direct,direct; Moves the content from a location to another in RAM
11. MOV direct,@Ri; Moves the content fom a RAM location to another location
12. MOV direct,#data; Moves the immediate data to a RAM location
13. MOV @Ri,A; Moves the content from accumulator to a location specified by Ri
14. MOV @Ri,direct; Moves a byte from a RAM location to a location specified by Ri
15. MOV @Ri,#data; Moves the immediate data to a location specified by Ri
16. MOV DPTR,#data; Moves a 16-bit data to data pointer register
17. MOVC A,@A+DPTR; Moves a code byte from ROM location (address=A+DPTR) to accumulator
18. MOVC A,@A+PC; Moves a code byte from ROM location (address=A+PC) to accumulator
19. MOVX A,@Ri; Moves the content from external RAM to accumulator
20. MOVX A,@DPTR; Moves the content from external RAM to accumulator
21. MOVX @Ri,A; Moves the content from accumulator to external RAM
22. MOVX @DPTR,A; Moves the content from accumulator to external RAM

23. PUSH direct; Stores a byte in the stack
24. POP direct; Retrieves a byte frrom the stack
25. XCH A,Rn; Exchanges the content of a register with accumulator
26. XCH A,direct; Exchanges the content of RAM location with the accumulator
27. XCH A,@Ri; Exchanges the content of RAM with the accumulator
28. XCHD A,@Ri; Exchanges the lower nibble of content of register with the accumulator

7. PROGRAM:

1. TO MOVE A BLOCK OF N DATA BYTES FROM 20H TO 40H LOCATION.

```
        Org 0000h           ;Program initialization.
        MOV R2, #0AH        ;Initialize the counter.
        MOV R0, #20H        ;Initialize the source memory location.
        MOV R1, #40H        ;Initialize the destination memory location
BACK:   MOV A, @R0          ;copy the content from source location R0 to accumulator
        MOV @R1, A          ;copy the content from accumulator to destination location R1
        INC R0              ;Increment the source memory location address by one
        INC R1              ;Increment the destination memory location address by one
        DJNZ R2, BACK       ;Decrement the counter register R2 by one , if count ≠0 repeat
        NOP                 No operation
        END                 End of the pogram
```

2. WRITE AN ALP TO MOVE A BLOCK OF N BYTES OF DATA FROM SOURCE (2000H) TO DESTINATION (20H) USING EXTERNAL RAM .

```
        Org 0000h           ;Program initialization.
        MOV DPTR, #2000H    ;INITIALIZE THE SOURCE MEMORY LOCATION.
        MOV R0, #20H        ; Initialize the destination memory location
        MOV R1, #0AH        ;Initialize the counter.
BACK:   MOVX A, @DPTR       ;copy the content from external source location dptr to
                            accumulator
        MOV @R0, A          ;copy the content from accumulator to destination location R0
        INC DPTR            ;Increment the source memory location address by one
        INC R0              ;Increment the destination memory location address by one
        DJNZ R1, BACK       ;Decrement the counter register R1 by one , if count ≠0 repeat
        NOP                 No operation
        END                 End of the pogram
```

3.WRITE AN ALP TO EXCHANGE THE SOURCE BLOCK STARTING WITH ADDRESS 20H, (INTERNAL RAM) CONTAINING N (05) BYTES OF DATA WITH DESTINATION BLOCK STARTING WITH ADDRESS 40H (INTERNAL RAM).

```
        Org 0000H           ;Program initialization
        MOV R2, #05H        ;Initialize the counter.
        MOV R0, #20H        ;Initialize the source memory location.
        MOV R1, #40H        ;Initialize destination memory location.
AGAIN:  MOV A, @R0
        XCH A, @R1          ;Exchange the data at source accumulator.
        MOV @R0, A          ;Destination memory location.
        INC R0              ;Increment the source memory pointer.
        INC R1              ;Increment the destination memory pointer
        DJNZ R2, AGAIN      ;Decrement the count if count ≠0, repeat.
STOP:   SJMP STOP
        END
```

4.WRITE AN ALP TO EXCHANGE THE SOURCE BLOCK STARTING WITH ADDRESS 10H (INTERNAL MEMORY), CONTAINING N (06) BYTES OF DATA WITH DESTINATION BLOCK STARTING AT LOCATION 00H (EXTERNAL MEMORY).

```
        Org 0000H           ;Program initialization
        MOV R2, #06H        ;Initialize the counter.
        MOV R0, #10H        ;Initialize the source memory location.
        MOV DPTR, #00H      ;Initialize destination memory location.
AGAIN:  MOVX A, @DPTR       ;Move the data from external memory location to accumlator
        MOV   B , @R0       ;Move the data from internal memory location to B register
        XCH A,B             ;Exchange the contents of A and B register
        MOV X @DPTR,A       ;Move the contents of accumulator to external memory
        MOV @R0,B           ;Move the contents of B register to internal memory
        INC R0              ;Increment the source memory pointer.
        INC DPTR            ;Increment the destination memory pointer
        DJNZ R2, AGAIN      ;Decrement the count if count ≠0, repeat.
STOP:   SJMP STOP
        END
```

8. OUTPUTS:

1.Before execution:

Address	20H	21H	22H	23H	24H	25H	30H
Data	56	33	21	99	66	21	99

Address	40H	41H	42H	43H	44H	45H	46H
Data	0	0	0	0	0	0	0

After execution

Address	20H	21H	22H	23H	24H	25H	30H
Data	56	33	21	99	66	21	99

Address	40H	41H	42H	43H	44H	45H	46H
Data	56	33	21	99	66	21	99

2.Before execution

Address	2000H	2001H	2002H	2003H	2004H	2005H	2006H
Data	56	33	21	99	66	21	99

Address	20H	21H	22H	23H	24H	25H	30H
Data	0	0	0	0	0	0	0

After Execution

Address	2000H	2001H	2002H	2003H	2004H	2005H	2006H
Data	56	33	21	99	66	21	99

Address	20H	21H	22H	23H	24H	25H	30H
Data	56	33	21	99	66	21	99

3.Before Execution

Address	20H	21H	22H	23H	24H	25H	30H
Data	56	33	21	99	66	21	99

Address	40H	41H	42H	43H	44H	45H	46H
Data	60	70	80	90	A0	B0	C0

After Execution

Address	20H	21H	22H	23H	24H	25H	30H
Data	60	70	80	90	A0	B0	C0
Address	40H	41H	42H	43H	44H	45H	46H
Data	56	33	21	99	66	21	99

4.Before Execution

Address	10H	11H	12H	13H	14H	15H	16H
Data	56	33	21	99	66	21	99

Address	00H	01H	02H	03H	04H	05H	06H
Data	60	70	80	90	A0	B0	C0

After Execution

Address	10H	11H	12H	13H	14H	15H	16H
Data	60	70	80	90	A0	B0	C0

Address	00H	01H	02H	03H	04H	05H	06H
Data	56	33	21	99	66	21	99

8. RESULTS & CONCLUSIONS:

- Programs on data transfer, data exchange are executed and outputs are verified

9. LEARNING OUTCOMES :

- The data transfer instructions, exchange instructions ,conditional/unconditional jump instructions and its application for the various programs involving data transfer between memories(internal and external) and verified

9. APPLICATION AREAS:

- Search Engines(computer algorithms)

10. REMARKS:

-

COURSE LABORATORY MANUAL

1. EXPERIMENT NO: IB

2. TITLE: ARITHMETIC & LOGICAL OPERATION PROGRAMS

3. LEARNING OBJECTIVES:

- To understanad the various arithmetic and logical operations in 8051Microcontroller using assembly language programs.

4. AIM: Write a program to:

a. Add two numbers. b. Subtract two numbers.
c. Multiply two numbers. d. Divide two numbers.
e. sort even and odd numbers f.sort positive and negative numbers
g.sort in ascending and descending order g.find largest and smallest number

5. MATERIAL / EQUIPMENT REQUIRED:

- Software tool: Keil uVision3

6. THEORY / HYPOTHESIS:

ARITHMETIC INSTRUCTIONS: Arithmetic instructions perform several basic operations such as addition, subtraction, division, multiplication etc.

LOGICAL OPERATIONS:The 8051 supports a number of logical operations. There is separate Boolean processor integrated within the 8051 microcontroller. It has its own instruction set, accumulator and bit addressable RAM. Carry flag serves as the accumulator. The instructions that allow bit manipulation perform operation like complement bit, set bit, clear bit. Logical bitwise AND ,OR operations are also supported. The results of these bit-wise logical operations are stored into the carry bit, which works as an accumulator.

It has the following format :

ADD A,Rn	Adds the content of register and accumulator
ADD A,direct	Adds the content of RAM and accumulator
ADD A,@Ri	Adds the content of RAM and accumulator
ADD A,#data	Adds the immediate data to the accumulator
ADDC A,Rn	Adds the content of register and accumulator with a carry flag
ADDC A,direct	Adds the content of RAM and accumulator with a carry flag
ADDC A,@Ri	Adds the indirect RAM to the accumulator with a carry flag
ADDC A,#data	Adds the immediate data to the accumulator with a carry flag
SUBB A,Rn	Subtracts the register contents and carry flag(CF) from the accumulator
SUBB A,direct	Subtracts RAM contents and CF from accumulator
INC A	Increments the content of accumulator by 1
INC Rn	Increments the register content by 1
INC Rx	Increments the direct byte by 1
DEC A	Decrements the accumulator by 1
DEC Rn	Decrements the contents of register by 1
INC DPTR	Increments the Data Pointer by 1
MUL AB	Multiplies A and B
DIV AB	Divides A by B

DA A	Decimal adjustment of the accumulator according to BCD code

7. PROGRAM:
5. WRITE AN ALP TO ADD THE BYTE IN THE RAM AT 34H AND 35H, STORE THE RESULT IN THE REGISTER R5 (LSB) AND R6 (MSB), USING INDIRECT ADDRESSING MODE.

Org 0000h	
MOV A,34H	;Data in address 34H is moved to Accumulator
ADD A,35H	;perform addition
MOV R5H,A	;store the result in R5H
JNC LAST	;Check if carry is generated
INC R6	
LAST: NOP	
END	

6. WRITE AN ALP TO SUBTRACT THE BYTES IN INTERNAL RAM 34H & 35H STORE THE RESULT IN REGISTER R5 (LSB) & R6 (MSB)

Org 0000h	
MOV A,34H	;Data in address 34H is moved to Accumulator
SUBB A,35H	;perform subtraction
MOV R5H,A	;store the result in R5H
JNC LAST	
DEC R6	
LAST: NOP	
END	

7.WRITE AN ALP TO MULTIPLY TWO 8-BIT NUMBERS STORED AT 30H AND 31H AND STORE16- BIT RESULT IN 32H AND 33H OF INTERNAL RAM.

Org 0000h	
MOV A,30H	;Store the number in Reg A.
MOV B,31H	;Save the number in Reg B.
MUL AB	;Perform multiplication
MOV 32H,A	;Store the lower byte of result in 32h
MOV 33H,B	;Store the higher byte of result in 33h
NOP	
END	

8.WRITE AN ALP TO PERFORM DIVISION OPERATION ON 8-BIT NUMBER BY 8-BIT NUMBER

Org 0000h	;Program initialization
MOV A,30H	; Store the dividend in Reg A.

DIV AB	; Perform division
MOV 32H,A	; Store the quotient in 32h
MOV 33H,B	; Store the remainder in 33.;
NOP	
END	
9.WRITE AN ALP TO SEPARATE POSITIVE AND NEGATIVE IN A GIVEN ARRAY	
ORG 0000H	;Program initialisation
MOV DPTR,#1000H	;load DPTR with 1000H in external memory.This holds the input array
MOV R0,#20H	;R0 points to address 20H where positive numbers are stored
MOV R1,#30H	;R1 points to address 30H where negative numbers are stored
MOV R2,#08H	;Load counter value
UP: MOVX A,@DPTR	;Fetch the data from external memory to accumulator
MOV B,A	;Store A value in B for future use
RLC A	;Rotate left the bits by one bit position to the left
JC NEG	;If carry is generated number is negative
MOV @R0,B	;Store the positive number
INC R0	;Increment R0 by one to store the next positive byte
JMP LAST	
NEG:MOV @R1,B	;Store the negative byte
INC R1	;Increment R1 by one to store the next negative byte
LAST: INC DPTR	;Fetch the next byte
DJNZ R2,UP	;Decrement the counter by ONE
NOP	
END	
10.WRITE AN ALP TO SEPARATE EVEN AND ODD NUMBERS IN A GIVEN ARRAY	
ORG 0000H	;Program initialisation
MOV DPTR,#1000H	load DPTR with 1000H in external memory.This holds the input array
MOV R0,#20H	;R0 points to address 20H where EVEN numbers are stored
MOV R1,#30H	;R1 points to address 30H where ODD numbers are stored

MOV R2,#08H	;Load counter value
UP: MOVX A,@DPTR	;Fetch the data from external memory to accumulator
MOV B,A	;Store A value in B for future use
RRC A	;Rotate left the bits by one bit position to the right
JC ODD	;If carry is generated number is odd
MOV @R0,B	;Store the even number
INC R0	;Increment R0 by one to store the next even byte
JMP LAST	
ODD:MOV @R1,B	;Store the odd byte
INC R1	;Increment R1 by one to store the next odd byte
LAST: INC DPTR	;Fetch the next byte
DJNZ R2,UP	;Decrement the counter by one
NOP	
END	
11A.WRITE AN ALP TO ARRANGE THE NUMBERS IN ASCENDING ORDER.	
ORG 0000H	;Program initialization
MOV R3,#06H	;Initialize the COUNT for external loop.
EXTERNALLOOP: MOV R0,#20H	;initialize the array.
MOV R1,#21H	;initialize the array.
MOV R2,#06H	;nitialize the COUNT for Internal loop.
INNERLOOP: MOV A,@R0	;Fetch the first byte
MOV B,@R1	;Fetch the second byte
CJNE A,B,CONTINUE	;Check if the two bytes are same
CONTINUE: JC ORDER	;If there is carry go to the specified label. This indicates that bytes are in required order
MOV @R1,A	;If there is no carry,it indicates that bytes are not in proper order hence exchange the contents
MOV @R0,B	
ORDER: INC R0	;increment R0 register
INC R1	;increment R1 register
DJNZ R2,INNERLOOP	;decrement the count in the internal loop by 1 and check if it is 0.If 0 then execute the next instruction
DJNZ R3,EXTERNALLOOP	;decrement the count in the external loop by 1

	and check if it is 0.If it is not 0 then goto the label external loop
NOP	;No operation .used for delay
END	;END of the program
11B WRITE AN ALP TO ARRANGE THE NUMBERS IN DESCENDING ORDER.	
ORG 0000H	;Program initialization
MOV R3,#06H	;Initialize the COUNT for external loop.
EXTERNALLOOP: MOV R0,#20H	;initialize the array.
MOV R1,#21H	;initialize the array.
MOV R2,#06H	;nitialize the COUNT for Internal loop.
INNERLOOP: MOV A,@R0	;Fetch the first byte
MOV B,@R1	;Fetch the second byte
CJNE A,B,CONTINUE	;Check if the two bytes are same
CONTINUE: JNC ORDER	;If there is no carry go to the specified label. This indicates that bytes are in required order
MOV @R1,A	;If there is no carry,it indicates that bytes are not in proper order hence exchange the contents
MOV @R0,B	
ORDER: INC R0	;increment R0 register
INC R1	;increment R1 register
DJNZ R2,INNERLOOP	;decrement the count in the internal loop by 1 and check if it is 0.If 0 then execute the next instruction
DJNZ R3,EXTERNALLOOP	;decrement the count in the external loop by 1 and check if it is 0.If it is not 0 then goto the label external loop
NOP	;No operation .used for delay
END	;END of the program
12A WRITE AN ALP TO FIND THE LARGEST NUMBER FROM A GIVEN ARRAY STARTING FROM 20H & STORE IT IN INTERNAL MEMORY LOCATION 40H	
ORG 0000H	;Program initialization
MOV R2,#05H	;Initialize the counter.
MOV R0,#20H	;Initialize the array.
MOV A,@R0	;Get a byte of data from the array.
UP:INC R0	;Increment R0
MOV B,@R0	;Get the next number.

CJNE A,B,CONT	;Compare A with B
CONT: JNC LARGEST	;If carry=1;A holds the smallest number.Goto the label
MOV A,B	;Store the smallest number in A register
LARGEST:DJNZ R2,UP	;If count ≠ 0 repeat.
MOV 40H,A	;Store the result in 40H memory location
NOP	
END	END of the program
12B Write an ALP to find the Smallest number from a given array starting from 20h & store it in Internal Memory location 40h	
ORG 0000H	;Program initialization
MOV R2,#05H	;Initialize the counter.
MOV R0,#20H	;Initialize the array.
MOV A,@R0	;Get a byte of data from the array.
UP:INC R0	;Increment R0
MOV B,@R0	;Get the next number.
CJNE A,B,CONT	;Compare A with B
CONT: JC SMALLEST	;If carry=0;A holds the smallest number.Goto the label
MOV A,B	;Store the smallest number in A register
SMALLEST:DJNZ R2,UP	;If count ≠ 0 repeat.
MOV 40H,A	;Store the result in 40H memory location
NOP	
END	

8. OUTPUTS:

5.Addition:

Before Execution		After Execution	
Memory	**Data**	**Memory**	**Data**
34H	AA	20H	AA
35H	75	21H	75
R5H	0	22H	1F
R6	0	R6	01

6.Subtraction:

Before Execution		After Execution	
Memory	**Data**	**Memory**	**Data**
34H	AA	20H	AA
35H	75	21H	75
R5H	0	22H	35
R6	0	R6	0

7.Multiplication:

Before Execution		After Execution	
Memory	**Data**	**Register**	**Data**
34H	AA	34H	AA
35H	75	35H	75
R5H	0	R5H	B2
R6	0	R6	4D

8.Division:

Before Execution		After Execution	
Memory	**Data**	**Register**	**Data**
34H	25	34H	25
35H	5	35H	5
R5H	0	R5H	7
R6	0	R6	2

9.Seperate Positive and negative numbers

Before Execution

Address	1000H	1001h	1002H	1003H	1004H	1005H	1006H	1007H	1008H	1009H
Data	0xD3	0xC8	0x22	0xF7	0x2D	0x17	0x4C	0x5C	0xA6	0xA7

Address	20 H	21 H	22 H	23H	24H			30 H	31 H	32 H	33 H	34H	
Data	00	00	00	00	00			00	00	00	00	00	

After Execution:

Address	1000H	1001h	1002H	1003H	1004H	1005H	1006H	1007H	1008H	1009H
Data	0xD3	0xC8	0x22	0xF7	0x2D	0x17	0x4C	0x5C	0xA6	0xA7

Address	20 H	21 H	22 H	23H	24H			30 H	31 H	32 H	33 H	34H	
Data	0x22	0x2D	0x17	0x4C	0x5C			0xD3	0xC8	0xF7	0xA6	0xA7	

10.Seperate even and odd numbers

Before Execution

Address	1000H	1001h	1002H	1003H	1004H	1005H	1006H	1007H	1008H	1009H
Data	0X97	0X43	0XAA	0X82	0X23	0X3F	0X53	0X42	0X58	0X16

Address	20 H	21 H	22 H	23H	24H	25H		30 H	31 H	32 H	33 H	34H	35H
Data	00	00	00	00	00	00		00	00	00	00	00	00

After Execution

Address	1000H	1001h	1002H	1003H	1004H	1005H	1006H	1007H	1008H	1009H
Data	0X17	0X43	0X5A	0X62	0X23	0X3F	0X53	0X42	0X58	0X16

Address	20 H	21 H	22 H	23H	24H	25H		30 H	31 H	32 H	33 H	34H	35H
Data	0X5A	0X62	0X42	0X58	0X16	0		0X17	0X43	0X23	0X3F	0X53	00

11A.Sort in ascending order

Before Execution

Address	20H	21H	22H	23H	24H	25H	26H
Data	56	33	21	99	66	22	5

After Execution

Address	20H	21H	22H	23H	24H	25H	26H
Data	5	21	22	33	56	66	99

11B.Sort in Descending order
Before Execution

Address	20H	21H	22H	23H	24H	25H	26H
Data	56	33	21	99	66	22	5

After Execution

Address	20H	21H	22H	23H	24H	25H	26H
Data	99	66	56	33	22	21	5

12A:Find the largest number in a array:
Before Execution:

Address	20H	21H	22H	23H	24H	25H	40H
Data	56	33	21	99	66	21	00

After Execution:

Address	20H	21H	22H	23H	24H	25H	40H
Data	56	33	21	99	66	21	99

12B:Find the Smallest number in a array:
Before Execution:

Address	20H	21H	22H	23H	24H	25H	40H
Data	56	33	21	99	66	21	00

After Execution:

Address	20H	21H	22H	23H	24H	25H	40H
Data	56	33	21	99	66	21	21

9. LEARNING OUTCOMES :

- Output for various arithmetic instruction like addition, subtraction, division and multiplication and finding smallest and largest in a array. Sorting: even and odd, positive and negative,ascending and descending order are observed.

10. APPLICATION AREAS:

- To find the sum of array
- To find GCD and LCM
- To find the average of given N numbers

11. REMARKS:

- -
- -

1. EXPERIMENT NO: IC

2. TITLE: COUNTERS

3. LEARNING OBJECTIVES:

- To study the fundamentals of basic sequential logic concepts.
- To program the 8051 counters in assembly level language

4. AIM:Write a program to:

a) Design hexadecimal up counter
b) Design hexadecimal down counter
c) Design decimal up counter
d) Design decimal down counter

5. MATERIAL / EQUIPMENT REQUIRED:

- Software Required-Keil uVision3

6. THEORY / HYPOTHESIS:

- Counting is frequently required in digital computers and other digital systems to record the number of events occurring in a specified interval of time. Normally an electronic counter is used for counting the number of pulses coming at the input line in a specified time period.
- The counter must possess memory since it has to remember its past states.

7. PROGRAM

13) HEADECIMAL UP COUNTER

```
        ORG 0000H
        MOV A,#00H
BACK:MOV P0,A
        INC A
        CALL DELAY
        JMP BACK
        DELAY:NOP
        MOV R0,#0AAH
  UP3:MOV R1,#0FFH
  UP2:MOV R2,#0FFH
  UP1:DJNZ R2,UP1
        DJNZ R1,UP2
        DJNZ R0,UP3
        RET
        END
```

14)HEXADECIMAL DOWN COUNTER

```
      ORG 0000H
      MOV A,#0FFH
BACK:MOV P0,A
      DEC A
      CALL DELAY
      JMP BACK
      DELAY:NOP
      MOV R0,#0AAH
```

```
        UP3:MOV R1,#0FFH
        UP2:MOV R2,#0FFH
   UP1:DJNZ R2,UP1
        DJNZ R1,UP2
        DJNZ R0,UP3
        RET
        END
```

15) DECIMAL UP COUTER

```
          ORG 0000H
           BEGIN:MOV A,#00H
           BACK:MOV P0,A
                    INC A
                   CALL DELAY
                   CJNE A,#0AH ,BACK
                   JMP BEGIN
                   DELAY:NOP
                   MOV R0,#0AAH
                   UP3:MOV R1,#0FFH
                   UP2:MOV R2,#0FFH
           UP1:DJNZ R2,UP1
                    DJNZ R1,UP2
                    DJNZ R0,UP3
                    RET
                    END
```

16) DECIMAL UP COUTER

```
           ORG 0000H
   BEGIN:MOV A,#09H
   BACK:MOV P0,A
           CALL DELAY
           DEC A
          CJNE A,#0FFH,BACK
          JMP BEGIN
          DELAY:NOP
          MOV R0,#0AAH
    UP3 :MOV R1,#0FFH
   UP2:MOV R2,#0FFH
   UP1:DJNZ R2,UP1
           DJNZ R1,UP2
           DJNZ R0,UP3
           RET
          END
```

8. GRAPHS / OUTPUTS:

7a)After Execution

Port0	
Data	00,01,02.....AA.....FF

7b)After Execution

Port0	
Data	FF,FE,02.....AA.....00

7c)After Execution

Port0	
Data	00,01,02.....09

7d)After Execution

Port0	
Data	09,08,07.....00

8. RESULTS & CONCLUSIONS:

- The programs for hexadecimal/decimal UP and DOWN counter is executed and output is verified

9. LEARNING OUTCOMES :

- The hexadecimal and decimal up-down counter program is analyzed and implemented.

10. APPLICATION AREAS:

- BCD counters are used in consumer appliances
- Digital clocks design

11. REMARKS:

- -

1. EXPERIMENT NO: II

2. TITLE: WRITE A 8051 C PROGRAM TO FIND THE SUM OF FIRST 10 INTEGER NUMBERS.

3. LEARNING OBJECTIVES:

- Understand the software tool required for programming in 8051 C.

4.AIM: To write a 8051 c program to find the sum of first 10 integer numbers.

To write a C program to find Factorial of a given number.

To write a C program to find the Square of a number (1 to 10) using Look-Up Table

To write a C program to count the number of Ones and Zeros in two consecutive memory locations.

5. MATERIAL / EQUIPMENT REQUIRED:

- Software Resources:Keil uVision-3

6. THEORY / HYPOTHESIS:

Unlike assembly, C has advantage of processor- independence and is not specific to any particular microprocessor/ microcontroller or any system. This makes it convenient for a user to develop programs that can run on most of the systems.

7. PROGRAM :

17.SUM OF FIRST 10 INTEGER NUMBERS.

```
#INCLUDE <REG51.H> VOID MAIN()
{
 int i, sum;
 sum = 0;
 for (i = 1; i <= 10; i++)
 {
  sum += i;
 }
 P0=SUM;// the result is stored in the sum variable
}
```

18. C PROGRAM TO FIND FACTORIAL OF A GIVEN NUMBER

```
#include <reg51.h>
UNSIGNED INT FACTORIAL(UNSIGNED INT N)
{
UNSIGNED INT I, RESULT = 1;   FOR (I = 1; I <= N; I++) {
RESULT = RESULT * I;
 }
RETURN RESULT;
 }
VOID MAIN()
{
UNSIGNED INT N = 5; // PUT THE NUMBER WHOSE FACTORIAL YOU WANT TO
                                    FIND HERE
UNSIGNED INT RESULT = FACTORIAL(N);// result now contains the factorial of n
p0=result;
}
```

19.WRITE AN 8051 C PROGRAM TO FIND THE SQUARE OF A NUMBER (1 TO 10) USING LOOK-UP TABLE.

```
#include<reg51.h>

unsigned int square_lookup_table[] = {0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100};

void main()

{
```

```
unsigned char n = 5; // put the number whose square you want to find here   unsigned int square =
square_lookup_table[n];// square now contains the square of n
 P0=square;
}
```

20.WRITE AN 8051 C PROGRAM TO COUNT THE NUMBER OF ONES AND ZEROS IN TWO CONSECUTIVE MEMORY LOCATIONS.

```
#INCLUDE <REG51.H> // INCLUDE 8051 REGISTER DEFINITIONS VOID ()
{
UNSIGNED CHAR NUM1, NUM2, I, COUNT_ONES = 0, COUNT_ZEROS = 0;
// initialize the two numbers with some value
 num1 = 0x23;
 num2 = 0x23;
// loop through each bit of num1 and num2     for(i = 0; i < 8; i++)
{
      // count the number of ones and zeros in the current bit position        if((num1 & (1 << i)) != 0)
{
count_ones++;
 }
else
{
count_zeros++;
 }
 if((num2 & (1 << i)) != 0)
{
count_ones++;
} else
{
count_zeros++;
}
 }
  // infinite loop
 while(1)
{
}
}
```

8. OUTPUTS:
Sum of Natutal numbers numbers:
0,1,1,2,3,5,8,D,15,22
Factorial of a number
Input-5!,Output-78
Square of a number:
Input-3,Output=9
Count of 1's and 0'S
Input-15H;Output:

ones=3,Zeros=5

9. RESULTS & CONCLUSIONS:
- The result of Sum of natural numbers, factorial, Square of a number, Count of 1's and Zeros are verified

10. LEARNING OUTCOMES :
- Understand the instruction set of Keil uVision3 , and the software tool required for programming in C 8051.

11. APPLICATION AREAS:
- Design of Calculator,Error detection and correction , data compression

12. REMARKS:

-

1. EXPERIMENT NO: 21

2. TITLE: WRITE AN 8051 C PROGRAM TO GENERATE SINE & SQUARE WAVEFORMS USING DAC INTERFACE.

3. LEARNING OBJECTIVES:
- To study and analyze DAC interface
- To generate sine, triangular and square waveform using DAC interface

4. AIM:
a) Assembly program to generate sine wave
b) Assembly program to generate square wave

5. MATERIAL / EQUIPMENT REQUIRED:
- Software resource: Keil Micro Vision-3
- DAC Interfacing kit
- CRO

6. THEORY / HYPOTHESIS:

DAC(Digital to Analog Convertor) is used widely to convert digital pulses into analog pulses. DAC contains binary weighted and R/2R ladder. The vast majority of integrated circuit DACs, including the MC1408 (DAC0808) used in this section, use the R/2R method since it can achieve a much higher degree of precision. The first criterion for judging a DAC is its resolution, which is a function of the number of binary inputs. The common ones are 8, 10, and 12 bits. The number of data bit inputs decides the resolution of the DAC since the number of analog output levels is equal to 2^n, where n is the number of data bit inputs. Therefore, an 8-input DAC such as the DAC0808 provides 256 discrete voltage (or current) levels of output. Similarly, the 12-bit DAC provides 4096 discrete voltage levels. There are also 16-bit DACs, but they are more expensive.

To generate a sine wave, we first need a table whose values represent the magnitude of the sine of angles between 0 and 360 degrees. The values for the sine function vary from -1.0 to +1.0 for 0- to 360-degree angles. Therefore, the table values are integer numbers representing the voltage magnitude for the sine of theta. This method ensures that only integer numbers are output to the DAC by the 8051 micro-controller.

7. PROGRAM:

a) Program to generate sine waveform using DAC

```
        org 0000h
        ljmp main
  main:mov count, #00h
        mov dptr, #table          ;dptr points to  location of the table
        mov a, count
        mov dptr, #table
     bk:movc a,@a+dptr          ;((a)+dptr)=(a)
        inc count
        mov p2,a
        mov a,count
        cjne a,#34h,bk           ;(a)!=34h then jump bk
        sjmp main


table:
db
80h,90h,0a1h,0b1h,0c0h,0cdh,0dah,0e5h,0eeh,0f6h,0fbh,0feh,0ffh,0ffh,0feh,0fbh,0f6h,0eeh,0e5h,0
dah,0cdh,0c0h,0b1h,0a1h,90h,80h,80h,70h,5fh,4fh,40h,33h,26h,1bh,12h,0ah,05h,02h,00h,00h,02h,
05h,0ah,12h,1bh,26h,33h,40h,4fh,5fh,70h,80h
```

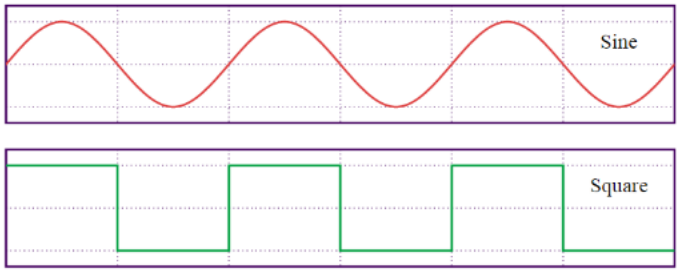
b) Program to generate square waveform using DAC

```
            org 0000h
            ljmp main
      main: mov a,#00h
        up: mov p2,a
            acall delay
            cpl a                ;complimenting (a) to generate alt high and low
            sjmp up
     delay: mov r0,#0ah
      loop: mov r1,#0ffh
       up1: djnz r1,up1
            djnz r0,loop
```

ret

sjmp main

8. GRAPHS / OUTPUTS:



9. RESULTS & CONCLUSIONS:

a) Sine Wave
Amplitude =.....Sec
Frequency =.....Hz

b) Square Wave
Amplitude =.....Sec
Frequency =.....Hz

10.LEARNING OUTCOMES :

- Different waveforms are observed using DAC interface

11. APPLICATION AREAS:

- Audio amplifier
- Video Encoder
- Calibration
- Motor Control
- Digital Potentiometer

15. REMARKS:

- -
- -
- -

-

1. EXPERIMENT NO: 22

2. TITLE: WRITE AN 8051 C PROGRAM TO ROTATE STEPPER MOTOR IN CLOCK & ANTI-CLOCKWISE DIRECTION.

3. LEARNING OBJECTIVES:

- To know the basic operation of stepper motor
- Code 8051C program to control and operate a stepper motor

4. AIM:

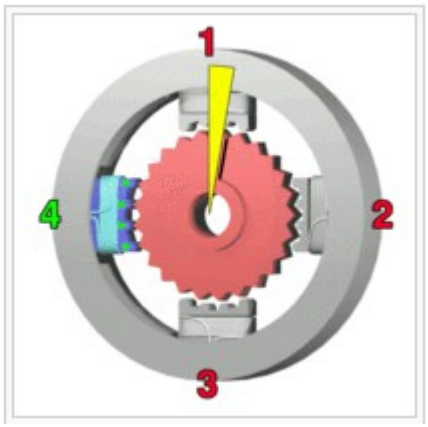
- Drive a stepper motor interface to rotate the motor in specified direction

5. MATERIAL / EQUIPMENT REQUIRED:

- Stepper motor interfacing kit
- Stepper motor
- DC power supply
- Keil uVision 3

6. THEORY / HYPOTHESIS:

- A stepper motor is widely used device that translates electrical pulses into mechanical movements.Each pulse moves the shaft through a fixed angle. Stepper motors effectively have multiple "toothed" electromagnets arranged around a central gear-shaped piece of iron. The electromagnets are energized by an external control circuit, such as a microcontroller. To make the motor shaft turn, the first electromagnet is given power, which magnetically attracts the gear's teeth. When the gear's teeth are aligned to the first electromagnet, they are slightly offset from the next electromagnet. This means that when the next electromagnet is turned on and the first is turned off, the gear rotates slightly to align with the next one. From there the process is repeated. Each of those rotations is called a "step", with an integer number of steps making a full rotation. In that way, the motor can be turned by a precise angle.



7. PROCEDURE / PROGRAMME / ACTIVITY:

```
#include "c:\ride\inc\51\reg51.h"
#define phasea 0x07
#define phaseb  0x0b
```

```
#define phasec 0x0d
#define phased 0x0e

void clockwise(void);
void delay(void);
int i;

void main ()
{
while(1)
{
clockwise();
}
}

void clockwise(void)
{
 P2 = phase;                //different speeds are loaded in P2
delay();
delay();

 P2 = phaseb;
 delay();
 delay();

P2 = phasec;
 delay();
delay();

P2 = phased;
 delay();
 delay();
}

void delay(void)
```

```
{
unsigned int i;
//delay_ms(250);
for(i=0;i<=40000;i++);
}
```

8. RESULTS & CONCLUSIONS:
   - Stepper motor rotates continuously in clockwise direction.

9. LEARNING OUTCOMES:
   - The program can be implemented to drive a stepper motor interface that rotate the motor in specified direction (clockwise or counter-clockwise) by N steps.

10. APPLICATION AREAS:
Programing the stepper motor which is present in:
   - Disk drives
   - Printers
   - Robotics

11. REMARKS:

-

1. EXPERIMENT NO: Open Ended Experiment 1

2. TITLE: INTERFACE A SIMPLE TOGGLE SWITCH TO 8051 AND WRITE AN ALP TO GENERATE AN INTERRUPT WHICH SWITCHES ON AN LED

3. LEARNING OBJECTIVES:
   - To Interface 8051 with LED and control the ON and OFF using Interrupts

4. AIM:
   - To interface a toggle switch with an LED and control its switching

5. MATERIAL / EQUIPMENT REQUIRED:
   - Microcontroller Kit
   - Keil microvision3 software

6. THEORY / HYPOTHESIS:
   - Switches and LED's are the basic example of input and output devices. The switch is a basic input device, use to control the operation of any output device. It basically breaks the electrical circuit and interrupts the flow of current.Light-emitting diode(LED) is a semiconductor ligh source that emits light when current flows through it. LED's have two leads one is the cathode and another one is the anode.

7. PROGRAM:

```
ORG 0000H
SETB P0.0
UP: MOV A,P0
RRC A
JC ON
CLR P1.0
```

```
JMP UP
ON: SETB P1.0
JMP UP
END
```

8. RESULTS & CONCLUSIONS:

- Whenver the switch is in ON state,the LED turn ON else it will be in OFF state

9. LEARNING OUTCOMES :

- LED interface with 8051 is analysed and implemented

10. APPLICATION AREAS:

- Display system,Automotive Lighting; Dimming of lights.

11. REMARKS:

- -
- -
- -

1. EXPERIMENT NO: open ended experiment 2

2. TITLE: Write a C program to (i) transmit and (ii) to receive a set of characters serially by interfacing 8051 to a terminal.

3. LEARNING OBJECTIVES:

- To explain the basic operation of transmitting and receiving set of characters serially.
- To interface transmitting and receiving set of characters serially with 8051 using C programing

4. AIM:

- Transmit and receive set of characters serially by interfacing 8051 to a terminal.

5. MATERIAL / EQUIPMENT REQUIRED:

- Microcontroller Kit, Keil microvision3 software

6. THEORY / HYPOTHESIS:

- To allow data transfer between the PC and an 8051 system without any error, we must make sure that the baud rate of the 8051 system matches the baud rate of the PC's COM port. The 8051 transfers and receives data serially at many different baud rates. The baud rate in the 8051 is programmable.

7. PROGRAM :

C program to transmit the message "YES" serially at 9600 baud, 8-bit data, 1 stop bit. Do this continuously.

```
#include<reg51.h>

void SerTx(unsigned char);

void main(void)

{

TMOD=0x20;

TH1=0xFD;

SCON=0x50;

TR1=1;while(1)
```

```
{
SerTx('Y');
SerTx('E');
SerTx('S');
}
}
void SerTx(unsigned char x)
{
SBUF=x;
while(TI==0);
TI=0;
}
```

C program to receive bytes of data serially and put them in P1. Set baud rate at 4800, 8-bit data and 1 stop bit.

```
#include<reg51.h>
void main(void)
{
unsigned char mybyte;
TMOD=0x20;
TH1=0xFA;
SCON=0x50;
TR1=1;
WHILE(RI==0);
mybyte=SBUF;
P1=mybyte;
RI=0;
}
}
```

8. RESULTS & CONCLUSIONS:

- Interfacing of transmitting and receiving set of characters serially with 8051 using C programing studied.

9. LEARNING OUTCOMES :

- Interfacing of transmitting and receiving set of characters serially with 8051 using C programing studied

10. APPLICATION AREAS:

- Serial transmission is often used in 10G connectivity or data transfer with great distances.

11. REMARKS:

- -
- -
- -