B.N.M. Institute of Technology

Approved by AICTE, Affiliated to VTU, Accredited as Grade A Institution by NAAC.
All UG branches – CSE, ECE, EEE, ISE & Mech.E Accredited by NBA for academic years 2018-19 to 2020-21 & valid upto 30.06.2021
Post box no. 7087, 27th cross, 12th Main, Banashankari 2nd Stage, Bengaluru- 560070, INDIA
Ph: 91-80- 26711780/81/82 Email: principal@bnmit.in, www. bnmit.org

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Vidyaya Amrutham Ashnuthe

LABORATORY MANUAL

<u>2019- 2020</u>

IV SEMESTER B.E

Course: Microcontroller & Embedded Systems Laboratory
Course Code: 18CSL48

Lab Incharge

Prashanth J
Pathanjali C

PROGRAM OUTCOMES

Engineering Graduates will be able to:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

Program Specific Outcomes

1. Analyze, design, develop and optimize solutions in C, C++, Java and .Net platforms.

2. Apply concepts in core areas of Computer Science –Data Structures, Database Management Systems, Operating Systems, Computer Architecture and Software Engineering to solve technical issues.

Course Outcomes

On the completion of this laboratory course, the students will be able to:

Cos	Statement	Bloom's Cognitive level	POs/PSOs
18CSL48.1	Demonstrate the ARM instruction set and explain how assembly language works.	Understanding	1,2 /2
18CSL48.2	Develop microcontroller software and interfacing programs using assembly language and Embedded 'C'.	Applying	1, 2, 3/1,2
18CSL48.3	Analyze the program using ARM7TDMI/LPC2148.	Analyzing	1,2, 3, 4 /1,2
18CSL48.4	Evaluate the programs on an ARM7TDMI/LPC2148 evaluation board using Embedded 'C' & Keil Uvision-4 tool/compiler.	Evaluating	1,2, 3,4, 5 /1,2

Strength of CO Mapping to PO/PSOs with Justification

Cos	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
18CSL48.1	3	3												2
18CSL48.2	3	3	3										2	2
18CSL48.3	3	3	3	2									2	2
18CSL48.4	3	3	3	2	2								2	2

SL. No.	PROGRAMS	CO Mapping
1	**Part-A:** Write a program to multiply two 16 bit binary numbers.	1,2,3
2	Write a program to find the sum of first 10 integer numbers.	1,2,3
3	Write a program to find factorial of a number**.**	1,2,3
4	Write a program to add an array of 16 bit numbers and store the 32 bit result in internal RAM.	1,2,3
5	Write a program to find the square of a number (1 to 10) using look-up table.	1,2,3
6	Write a program to find the largest/smallest number in an array of 32 Numbers.	1,2,3
7	Write a program to arrange a series of 32 bit numbers in ascending/ descending order.	1,2,3
8	Write a program to count the number of ones and zeros in two consecutive memory locations.	1,2,3
9	**Part-B:** Introduction to Interfacing Laboratory. Display "Hello World" message using Internal UART.	1,2,3,4
10	Interface and Control a DC Motor.	1,2,3,4
11	Interface a Stepper motor and rotate it in clockwise and anti-clockwise direction.	1,2,3,4
12	Determine Digital output for a given Analog input using Internal ADC of ARM controller	1,2,3,4
13	Interface a DAC and generate Triangular and Square waveforms.	1,2,3,4
14	Interface a 4x4 keyboard and display the key code on an LCD.	1,2,3,4
15	Demonstrate the use of an external interrupt to toggle an LED On/Off.	1,2,3,4
16	Display the Hex digits 0 to F on a 7-segment LED interface, with an appropriate delay in between.	1,2,3,4

SCHEME OF EVALUATION & RUBRICS FOR LABORATORY

Assessment processes:

a. Observation Write-up + execution + Viva	**20** **(7+8+5)**	**Average of all programs**
b. Lab Record	**10**	**Average of all programs**
c. Test	**10**	**Average mark considered from Test is reduced to 10 Marks for final marks calculation.**
Total	**40marks**	

Progress Tracker:

Attribute	Excellent (8)	Good (6)	Satisfactory (4)	Poor (1)
Execution (8)	Executing and rectifying the error if any in the program. The results are fully interpreted and compared manually	Executingand rectifying the errors in the program. The results are not properly interpreted and compared manually.	Able to complete the execution and rectifying the errors in the program. The results are not logically interpreted.	Not able to complete the execution and rectifying the errors in the program, also unable to interpret the results

Rubrics for Evaluation of Observation Book

Attribute	Excellent (7)		Good (5)		Satisfactory (3)		Poor (1)	
1.Write up (07)	Source code without syntax and logical errors	3	Source code with minor syntax errors	2	Source codewith syntax and logical errors	1	Incomplete Source code	0
	Code with comments.	1	Code with comments.	1	Code with comments.	1	Code with comments.	1
	Analyze and justify different cases of Input and output.	2	Analyze but not able to justify different cases of Input and output.	1	Analyze but not able to justify different cases of Input and output.	1	Analyze but not able to justify different cases of Input and output.	0
	Indentation.	1	Indentation.	1	Indentation.	0	Indentation.	0

Attribute	Excellent (8)		Good (6)		Satisfactory (4)		Poor (1)	
2.Execution (08)	Debugs the program independently	3	Debugs the program independently.	3	Debugs the program with help	2	Not able to complete the execution of program within the lab session.	1
	Executes the program for all possible inputs	3	Executes the program for all possible inputs.	3	Executes the program for one of the possible inputs.	2		
	Able to modify existing program	2						

Attribute	Excellent (5)	Good (3)	Satisfactory (2)	Poor (1)
3.Viva – voce (05)	• Answering 4-5 out of 5 questions.	• Answering 2-3 out of 5 questions.	• Answering 1-2 out of 5questions.	• Not answering any questions.

Rubrics for Evaluation of Record Book

Attribute	Excellent (10)		Good (8)		Satisfactory (6)		Poor (5)	
Record (10)	Writes the record neatly with comments and indentation	5	Writes the record neatly with comments and indentation	5	Not written clearly With comment and indentation.	3	Late submission.	5
	All possible cases of output	3	All possible cases of output.	3	All possible cases of output.	3		
	Index entry and submits on time	2						

Rubrics for Evaluation of Test

Attribute	Excellent (8)		Good (6)		Satisfactory (4)		Poor (1)	
1.Write up (08)	Source code without syntax and logical errors	4	Source code with minor syntax errors	3	Source code with syntax and logical errors	2	Incomplete Source code	0
	Code with comments and indentation	2	Code with comments and indentation	2	Code with comments and indentation	1	Code with comments and indentation	1
	Analyze and justify different cases of Input and output.	2	Analyze but not able to justify different cases of Input and output.	1	Analyze but not able to justify different cases of Input and output.	1	Analyze but not able to justify different cases of Input and output.	0

Attribute	Excellent (30-35)	Good (20-29)	Satisfactory (11-19)	Poor (0-10)
2. Execution (35)	Execution of the program for all possible inputs Able to modify the programs as per the examiners direction	Execution of the program for all possible inputs	Debugs the program. Not able to execute program for all possible inputs.	Not able to execute the program within lab session.

Attribute	Excellent (6-7)	Good (4-5)	Satisfactory (2-3)	Poor (0-1)
3.Viva – voce (07)	Answering 9-10 out of 10 questions.	Answering 6-8 out of 10 questions.	Answering 3-5 out of 10 questions.	Answering 1-2 out of 10 questions/ not answering any questions

***Two internal tests are conducted with above scheme for 50 Marks and an average mark is considered for scaling down to 10 marks for final IA calculation.**

Part-A

1. Write a program to multiply two 16 bit binary numbers.

```
        AREA MULTIPLY, CODE, READONLY

        ENTRY                           ; Mark first instruction to execute
                MOV R1,#6400                    ; STORE FIRST NUMBER IN R0

                MOV R2,#3200                    ; STORE SECOND NUMBER IN R1

                MUL R3,R1,R2                    ; MULTIPLICATION

                NOP

                NOP

                NOP

        END                             ; Mark end of file
```

OR

```
        AREA MULTIPLY, CODE, READONLY

        ENTRY                           ; Mark first instruction to execute
                LDR R1,=6400            ; STORE FIRST NUMBER IN R0

                LDR R2,=3200            ; STORE SECOND NUMBER IN R1

                MUL R3,R1,R2            ; MULTIPLICATION

STOP            B STOP

        END                             ; Mark end of file
```

OR

```
        ;Memory address are assumed in this program
        AREA MULTIPLY, CODE, READONLY
        ENTRY
                MOV R0,#0x40000000      ;Transfer the address to R0

                LDRH R2,[R0]            ;Load 16bit value from memory pointed by R0

                MOV R1,#0x40000006      ;Transfer the address to R1

                LDRH R3,[R1]            ;Load 16bit value from memory pointed by R1

                MUL R5,R2,R3            ; MULTIPLICATION

                MOV R4,#0x4000001C      ;Transfer the Result address to R4

                STR R5,[R4]             ;Store the result to the memory pointed by R4

                SWI 0x11                ; Stop execution

        END
```

2. Write a program to find the sum of first 10 integer numbers.

```
          AREA SUM, CODE, READONLY

          ENTRY
                    MOV R1,#10                ; load 10 to register

                    MOV R2,#0                 ; empty the register to store result
LOOP

                    ADD R2,R2,R1              ; add the content of R1 with result at R2

                    SUBS R1,#0x01             ; Decrement R1 by 1

                    BNE LOOP                  ; repeat till r1 goes 0


BACK                B BACK                    ; jumps back to C code

          END
```

3. Write a program to find factorial of a number.

```
          AREA  FACTORIAL , CODE, READONLY

          ENTRY                               ; Mark first instruction to execute
                    MOV R0, #3                ; STORE FACTORIAL NUMBER IN R0

                    MOV R1,R0                 ; MOVE THE SAME NUMBER IN R1


FACT                SUBS R1, R1, #1           ; SUBTRACTION

                    CMP R1, #1                ; COMPARISON

                    BEQ STOP

                    MUL R3,R0,R1;             ; MULTIPLICATION

                    MOV   R0,R3               ; Result

                    BNE FACT                  ; BRANCH TO THE LOOP IF NOT EQUAL

STOP                B STOP

          END                                 ;Mark end of file
```

4. Write a program to add an array of 16 bit numbers and store the 32 bit result in internal RAM.

```
        AREA ADDITION, CODE, READONLY

        ENTRY                           ; Mark first instruction to execute

START       MOV R5,#6                   ; INTIALISE COUNTER TO 6(i.e. N=6)

            MOV R0,#0                   ; INTIALISE SUM TO ZERO

            LDR R1,=VALUE1              ; LOADS THE ADDRESS OF FIRST VALUE

LOOP        LDRH R3,[R1],#02            ; READ 16 BIT DATA

            ADD R0,R0,R3                ; ADD R2=R2+R3

            SUBS R5,R5,#1               ; DECREMENT COUNTER

            CMP R5,#0

            BNE LOOP                    ; LOOK BACK TILL ARRAY ENDS

            LDR R4,=RESULT              ; LOADS THE ADDRESS OF RESULT

            STR R0,[R4]                 ; STORES THE RESULT IN R1

JMP         B JMP


VALUE1       DCW  0X1111, 0X2222, 0X3333, 0XAAAA, 0XBBBB, 0XCCCC
                                        ; ARRAY OF 16 BIT NUMBERS (N=6)


        AREA DATA2, DATA, READWRITE
                                        ; TO STORE RESULT IN GIVEN ADDRESS
RESULT      DCD 0X0

        END                               ; Mark end of file
```

5. Write a program to find the square of a number (1 to 10) using look-up table.

```
        AREA SQNEW, CODE, READONLY
       ENTRY
               LDR    R0, = TABLE1              ; Load start address of Lookup table
               MOV R1, #0X00                    ; Counter is initialized to 0
LOOP           LDR R3,[R0]                      ; Load first element of Lookup table
               ADD R0, R0,#04                   ; Increment the address to point to next
                                                ;  element in Lookup table
               CMP R1,#10                       ; Compare the counter (R1-10) and set the flag
               ADD R1,R1,#01                   ; Increment the counter by 1
               BNE LOOP                        ; Checks the zero flag is! =1, repeat the loop
STOP           B        STOP                   ; Unconditional flag
TABLE1         DCD 0X00000000                   ; SQUARE OF 0=0
               DCD 0X00000001                  ;  SQUARE OF 1=1
               DCD 0X00000004                   ; SQUARE OF 2=4
               DCD 0X00000009                  ; SQUARE OF 3=9
               DCD 0X00000010                  ; SQUARE OF 4=16
               DCD 0X00000019                  ; SQUARE OF 5=25
               DCD 0X00000024                  ; SQUARE OF 6=36
               DCD 0X00000031                  ; SQUARE OF 7=49
               DCD 0X00000040                  ; SQUARE OF 8=64
               DCD 0X00000051                  ; SQUARE OF 9=81
               DCD 0X00000064                  ; SQUARE OF 10=100
        END
```

6. Write a program to find the largest/smallest number in an array of 32 Numbers.

```
        AREA  LARGEST , CODE, READONLY
        ENTRY                           ; Mark first instruction to execute
START
                MOV R5,#6               ; INTIALISE COUNTER TO 6(i.e. N=7)

                LDR R1,=VALUE1          ; LOADS THE ADDRESS OF FIRST VALUE

                LDR R2,[R1],#4          ; WORD ALIGN T0 ARRAY ELEMENT

LOOP

                LDR R4,[R1],#4          ; WORD ALIGN T0 ARRAY ELEMENT

                CMP R2,R4               ; COMPARE NUMBERS

                BHI LOOP1               ; IF THE FIRST NUMBER IS > THEN GOTO

                                        ; LOOP1

                MOV R2,R4               ; IF THE FIRST NUMBER IS < THEN MOV

                                        ; CONTENT R4 TO R2

LOOP1

                SUBS R5,R5,#1           ; DECREMENT COUNTER

                CMP R5,#0               ; COMPARE COUNTER TO 0

                BNE LOOP                ; LOOP BACK TILL ARRAY ENDS

                LDR R4,=RESULT          ; LOADS THE ADDRESS OF RESULT

                STR R2,[R4]             ; STORES THE RESULT IN R2

STOP            B STOP

; ARRAY OF 32 BIT NUMBERS(N=7)

VALUE1          DCD   0X44444444

                DCD   0X22222222

                DCD   0X11111111

                DCD   0X33333333

                DCD   0XAAAAAAAA

                DCD   0X88888888

                DCD   0X99999999

        AREA DATA2, DATA, READWRITE     ; TO STORE RESULT IN GIVEN ADDRESS

                RESULT        DCD 0X0

        END                                     ; Mark end of file
```

7. Write a program to arrange a series of 32 bit numbers in ascending/ descending order.

```
        AREA  ASCENDING , CODE, READONLY
        ENTRY                                   ;Mark first instruction to execute
                MOV R8,#4                       ; INTIALISE COUNTER TO 4(i.e. N=4)

                LDR R2,=CVALUE                  ; ADDRESS OF CODE REGION

                LDR R3,=DVALUE                  ; ADDRESS OF DATA REGION

LOOP0           LDR R1,[R2],#4                  ; LOADING VALUES FROM CODE REGION

                STR R1,[R3],#4                  ; STORING VALUES TO DATA REGION

                SUBS R8,R8,#1                   ; DECREMENT COUNTER

                CMP R8,#0                       ; COMPARE COUNTER TO 0

                BNE LOOP0                       ; LOOP BACK TILL ARRAY ENDS

START1          MOV R5,#3                       ; INTIALISE COUNTER TO 3(i.e. N=4)

                MOV R7,#0                       ; FLAG TO DENOTE EXCHANGE HAS OCCURED

                LDR R1,=DVALUE                  ; LOADS THE ADDRESS OF FIRST VALUE

LOOP            LDR R2,[R1],#4                  ; WORD ALIGN T0 ARRAY ELEMENT

                LDR R3,[R1]                     ; LOAD SECOND NUMBER

                CMP R2,R3                       ; COMPARE NUMBERS

                BLT LOOP2                       ; IF THE FIRST NUMBER IS < THEN GOTO loop2

                STR R2,[R1],#-4                 ; INTERCHANGE NUMBER R2 & R3

                STR R3,[R1]                     ; INTERCHANGE NUMBER R2 & R3

                MOV R7,#1                       ; FLAG DENOTING EXCHANGE HAS TAKEN PLACE

                ADD R1,#4                       ; RESTORE THE PTR

LOOP2           SUBS R5,R5,#1                   ; DECREMENT COUNTER

                CMP R5,#0                       ; COMPARE COUNTER TO 0

                BNE LOOP                        ; LOOP BACK TILL ARRAY ENDS

                CMP R7,#0                       ; COMPARING FLAG

                BNE  START1                     ; IF FLAG IS NOT ZERO THEN GO TO START1 LOOP

JMP             B       JMP
CVALUE          DCD    0X44444444
                DCD   0X11111111
                DCD   0X33333333
                DCD   0X22222222
        AREA DATA1, DATA, READWRITE
DVALUE          DCD 0X00000000
        END
```

8. Write a program to count the number of ones and zeros in two consecutive memory locations.

```
          AREA ONEZERO, CODE, READONLY
          ENTRY                                   ;Mark first instruction to execute
                    MOV R2,#0                     ; COUNTER FOR ONES

                    MOV R3,#0                     ; COUNTER FOR ZEROS

                    MOV R7,#1                     ; COUNTER TO GET TWO WORDS

                    LDR R6,=VALUE                 ; LOADS THE ADDRESS OF VALUE


LOOP                MOV R1,#32                    ; 32 BITS COUNTER

                    LDR R0,[R6],#4                ; GET THE 32 BIT VALUE


LOOP0               MOVS R0,R0,ROR #1             ; RIGHT SHIFT TO CHECK CARRY BIT (1's/0's)

                    BHI ONES                      ; IF CARRY BIT IS 1 GOTO ONES BRANCH

                                                  ; OTHERWISE NEXT
ZEROS               ADD R3,R3,#1                  ; IF CARRY BIT IS 0 THEN INCREMENT THE

                                                  ; COUNTER BY 1(R3)

                    B LOOP1                       ; BRANCH TO LOOP1

ONES                ADD R2,R2,#1                  ; IF CARRY BIT IS 1 THEN INCREMENT THE

                                                  ; COUNTER BY 1(R2)


LOOP1               SUBS R1,R1,#1                 ; COUNTER VALUE DECREMENTED BY 1

                    BNE LOOP0                     ; IF NOT EQUAL GOTO TO LOOP0 CHECKS 32BIT

                    SUBS R7,R7,#1                 ; COUNTER VALUE DECREMENTED BY 1

                    CMP R7,#0                     ; COMPARE COUNTER R7 TO 0

                    BNE LOOP                      ; IF NOT EQUAL GOTO TO LOOP

STOP B  STOP

VALUE               DCD 0X11111111, 0XAA55AA55  ;TWO VALUES IN AN ARRAY

          END                                      ; Mark end of file
```