DAYANANDA SAGAR ACADEMY OF TECHNOLOGY & MANAGEMENT

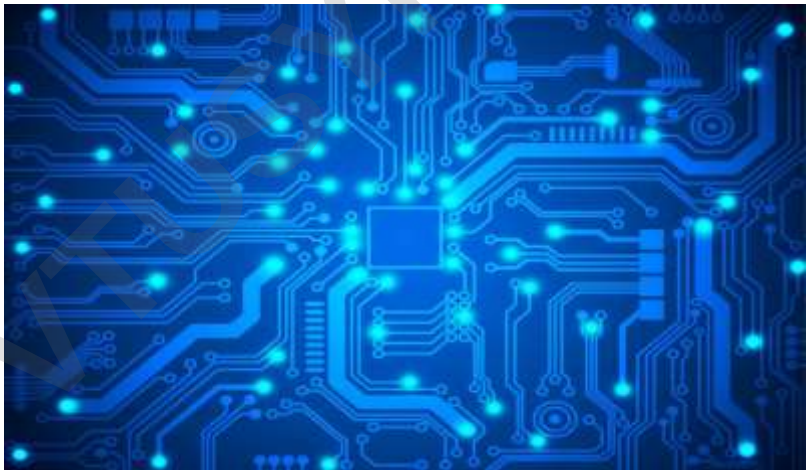
Affiliated to VTU
Approved by AICTE
Accredited by NAAC with A+ Grade
6 Programs Accredited by NBA
(CSE, ISE, ECE, EEE, MECH, CIVIL)

2023-2024

MICROCONTROLLERS

LABORATORY MANUAL- Integrated Course

(BCS402)



Compiled by:

Prof. Bhavya V

Asst. Prof, Dept. of CSE

Dr. Kavitha C
Prof. & Head of the Department,
Computer Science and Engineering,
DSATM

Dr. M Ravishankar
Principal, DSATM

Vision

Epitomize CSE graduate to carve a niche globally in the field of computer science to excel in the world of information technology and automation by imparting knowledge to sustain skills for the changing trends in the society and industry.

Mission

- M1: To educate students to become excellent engineers in a confident and creative environment through world-class pedagogy.
- M2: Enhancing the knowledge in the changing technology trends by giving hands-on experience through continuous education and by making them to organize & participate in various events.
- M3: Impart skills in the field of IT and its related areas with a focus on developing the required competencies and virtues to meet the industry expectations.
- M4: Ensure quality research and innovations to fulfill industry, government & social needs.
- M5: Impart entrepreneurship and consultancy skills to students to develop self-sustaining life skills in multi-disciplinary areas.

Program Educational Objectives (PEO's):

- PEO1: Engage in professional practice to promote the development of innovative systems and optimized solutions for Computer Science and Engineering.
- PEO2: Adapt to different roles and responsibilities in interdisciplinary working environment by respecting professionalism and ethical practices within organization and society at national and international level.
- PEO3: Graduates will engage in life-long learning and professional development to acclimate the rapidly changing work environment and develop entrepreneurship skills.

Programme Specific Outcomes (PSO's):

- PSO1: Foundation of Mathematical Concepts: Ability to use mathematical methodologies to crack problem using suitable mathematical analysis, data structure and suitable algorithm.
- PSO2: Foundation of Computer System: Ability to interpret the fundamental concepts and methodology of computer systems. Students can understand the functionality of hardware and software aspects of computer systems.
- PSO3: Foundations of Software Development: Ability to grasp the software development lifecycle and methodologies of software systems. Possess competent skills and knowledge of software design process. Familiarity and practical proficiency with a broad area of programming concepts and provide new ideas and innovations towards research.
- PSO4: Foundations of Multi-Disciplinary Work: Ability to acquire leadership skills to perform professional activities with social responsibilities, through excellent flexibility to function in multi-disciplinary work environment with self-learning skills.

MICROCONTROLLERS		Semester	4
Course Code	BCS402	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	3:0:2:0	SEE Marks	50
Total Hours of Pedagogy	40 hours Theory + 8-10 Lab Slots	Total Marks	100
Credits	04	Exam Hours	3
Examination nature (SEE)	Theory		

Exp. No.	Experiment Title	Date of execution
	Introduction to Keil IDE- Sample Programs	5
1.	Using Keil software, observe the various Registers, Dump, CPSR, with a simple Assembly Language Programs (ALP).	6
2.	Develop and simulate ARM ALP for Data Transfer, Arithmetic and Logical operations (Demonstrate with the help of a suitable program).	6
3.	Develop an ALP to multiply two 16-bit binary numbers.	7
4.	Develop an ALP to find the sum of first 10 integer numbers.	8
5.	Develop an ALP to find the largest/smallest number in an array of 32 numbers.	9
6.	Develop an ALP to count the number of ones and zeros in two consecutive memory locations.	10
7.	Simulate a program in C for ARM microcontroller using KEIL to sort the numbers in ascending/descending order using bubble sort.	11
8.	Simulate a program in C for ARM microcontroller to find factorial of a number.	12
9.	Simulate a program in C for ARM microcontroller to demonstrate case conversion of characters from upper to lowercase and lower to uppercase.	13
10.	Demonstrate enabling and disabling of Interrupts in ARM.	14

Course outcome (Course Skill Set)

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

CO1: **Interpret** the ARM Architectural features and Instructions.

CO2: **Analyze** C-Compiler Optimizations and portability issues in ARM Microcontroller.

CO3: **Apply** the concepts of Exceptions and Interrupt handling mechanisms in developing applications.

CO4: **Demonstrate** the role of Cache management and Firmware in Microcontrollers.

CO5: **Develop** programs using ARM/C instruction set for an ARM Microcontroller.

CO6: **Design** a microcontroller based system solve real time problems.

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is

50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

CIE for the theory component of the IPCC (maximum marks 50)

- IPCC means practical portion integrated with the theory of the course.
- CIE marks for the theory component are 25 marks and that for the practical component is 25 marks.
- 25 marks for the theory component are split into 15 marks for two Internal Assessment Tests (Two Tests, each of 15 Marks with 01-hour duration, are to be conducted) and 10 marks for other assessment methods mentioned in 22OB4.2. The first test at the end of 40-50% coverage of the syllabus and the second test after covering 85-90% of the syllabus.
- Scaled-down marks of the sum of two tests and other assessment methods will be CIE marks for the theory component of IPCC (that is for 25 marks).
- The student has to secure 40% of 25 marks to qualify in the CIE of the theory component of IPCC.

CIE for the practical component of the IPCC

- 15 marks for the conduction of the experiment and preparation of laboratory record, and 10 marks for the test to be conducted after the completion of all the laboratory sessions.
- On completion of every experiment/program in the laboratory, the students shall be evaluated including viva-voce and marks shall be awarded on the same day.
- The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and scaled down to 15 marks.
- The laboratory test (duration 02/03 hours) after completion of all the experiments shall be conducted for 50 marks and scaled down to 10 marks.
- Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IPCC for 25 marks.
- The student has to secure 40% of 25 marks to qualify in the CIE of the practical component of the IPCC.

ARM Microntrollers

Keil MDK is the complete software development environment for a range of Arm Cortex-M based microcontroller devices. MDK includes the µVision IDE and debugger, Arm C/C++ compiler, and essential middleware components.

MDK Introduction

MDK helps you to create embedded applications for ARM Cortex-M processor based devices. MDK is a powerful, yet easy to learn and use development system. It consists of MDK-Core and software packs, which can be downloaded and installed based on the requirements of your application.

MDK Tools

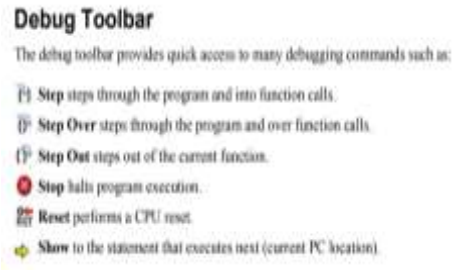
The MDK Tools include all the components that you need to create, build, and debug an embedded application for ARM based microcontroller devices. MDK-Core consists of the genuine Keil µVision IDE and debugger with leading support for Cortex-M processor-based microcontroller devices including the new ARMv8-M architecture. DS-MDK contains the Eclipse-based DS-5 IDE and debugger and offers multi-processor support for devices based on 32-bit Cortex-A processors or hybrid systems with 32-bit Cortex-A and Cortex-M processors.

MDK includes two ARM C/C++ Compilers with assembler, linker, and highly optimize run-time libraries tailored for optimum code size and performance:

- ARM Compiler version 5 is the reference C/C++ compiler available with a TÜV certified qualification kit for safety applications, as well as long-term support and maintenance.
- ARM Compiler version 6 is based on the innovative LLVM technology and supports the latest C language standards including C++11 and C++14. It offers the smallest size and highest performance for Cortex-M targets.

Create Application: The required steps for creating application programs are listed below:

1. Create Project File explains how to create a µVision4 project file.
2. Select Device describes how tool settings, peripherals, and dialogs are customized.
3. Set Options for Target explains how to specify relevant parameters for your target.
4. Create Source File describes how to add new files to the project.
5. Create File Group explains how to structure the code files for a better overview.
6. Specify Memory Layout describes how to change file and group attributes.
7. Build Project describes the build-options.



Create Project File

µVision4 maintains the files that belong to a project in one project file. It takes only a few steps to create a new project file with µVision4:

1. Select Project - New Project from the µVision4 menu. This opens a standard Windows dialog, which prompts you forthe new project file name.
2. Create a new folder Test.
3. Switch to the new folder and type the project name Test1.µVision4 automatically adds the extension uvproj.
4. Click Save & Select Device for Target 'Target 1' is opened;select NXP (founded by Philips). Select the microcontroller you use. For this example, choose the NXP(founded by Philips). Select the device LPC 2148 and click OK.
5. LPC 2100 represents the series of the controller. Click Yes to add the startup-code, which is provided for most devices. The startup-code provides configuration settingsfor the selected device.
6. Set Options for Target: Now Enter Clock frequency as 12MHZ and Select the Use Micro LIB option.Now click on output tab and select the Create Hex File option.
7. Click on new document for writing the program & save the code with .c extension.
8. Expand the Target 1 in the Project folder, Right click on the source Group1 & select Add Files to Group 'Source Group 1'. Selectthe file from the folder where project is saved & click on add.
9. Build Project: Build the project once the coding is done and check for errors. For building the project click F7.
10. Translate and link the source files of the application with a clickon one of the build- buttons located in the Build Toolbar.
11. Errors or warnings are displayed in the Build Output Window. Double-click on a message to jump to the line where the incident occurred.
12. Once you have generated your application successfully, youcan start debugging.

Experiment 1 & 2

Aim: Using Keil software, observe the various Registers, Dump, CPSR, with a simple Assembly Language Programs (ALP). Develop and simulate ARM ALP for Data Transfer, Arithmetic and Logical operations

```
        AREA Program, CODE, READONLY
        ENTRY
        MOV R5,#8              ; Transfer the data to register
        MOV R6,#4
        ADD R7,R6,R5           ; Perform Arithmetic operation
        MOV R8,R7              ; Register transfer
        AND R8,R5              ; Perform Logical operation
        ORR R7,R6
        EOR R7,R7
here    B here                 ; all done
MEMORY  DCD 0x40000000
        END
```

Results

Observe the results in General Purpose registers and CPSR register

Observation

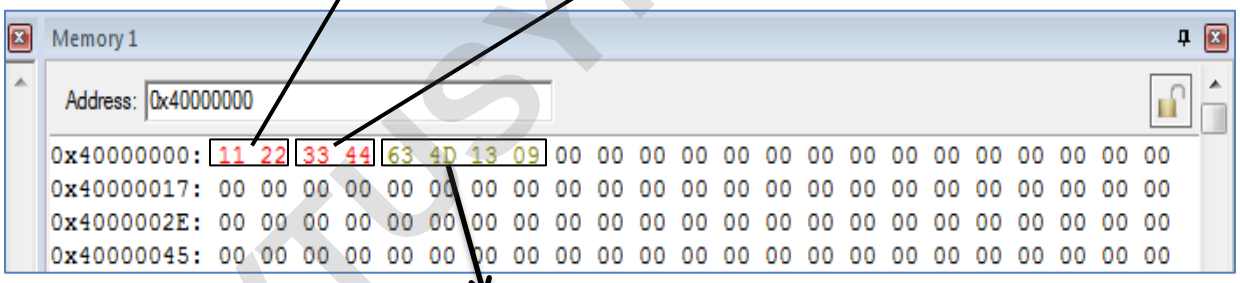
Experiment 3

Aim: Develop an ALP to multiply two 16-bit binary numbers.

```
        AREA Multiply, CODE, READONLY
        ENTRY
        LDR   R0, MEMORY          ; load Address of memory
        LDRH  R1, [R0]            ; load First number
        LDRH  R2, [R0, #2]        ; load Second number
        MUL   R2, R1, R2          ; R2=R1*R2
        STR   R2, [R0, #4]        ; Store the result
here    B here                    ; all done
MEMORY  DCD   0x40000000
        END
```

Results

Memory location: Input 1:0x40000000 or [R0] **Input 2:** 0x40000002 or [R0+02]



MUL Result: 0x40000004 **or** Contents of R2

Observation

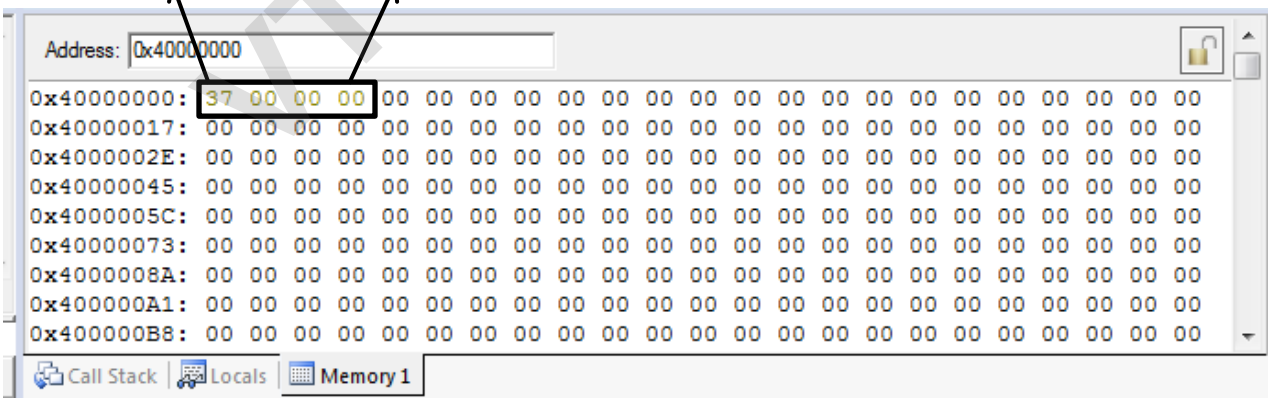
Experiment 4

Aim: Develop an ALP to find the sum of first 10 integer numbers.

```
        AREA SUM, CODE, READONLY
        ENTRY
        MOV R0,#10          ; Counter ( 10 integer number)
        MOV R1,#0           ; Partial sum
        MOV R2, #1          ; First number
NEXT    ADD R1,R1,R2        ; add partial sum with first number
        ADD R2,#1           ; update next inter number
        SUBS R0,#1          ; Decrement the counter
        BNE NEXT            ;if counter ! =0 the repeat
        LDR R3, RES         ; Get the address of the result
        STR R1,[R3]         ; store the result (Final sum)
B1      B B1                ; Stop
RES     DCD 0X400000000
        END
```

Results

Least Significant Byte Most Significant Byte



Observation

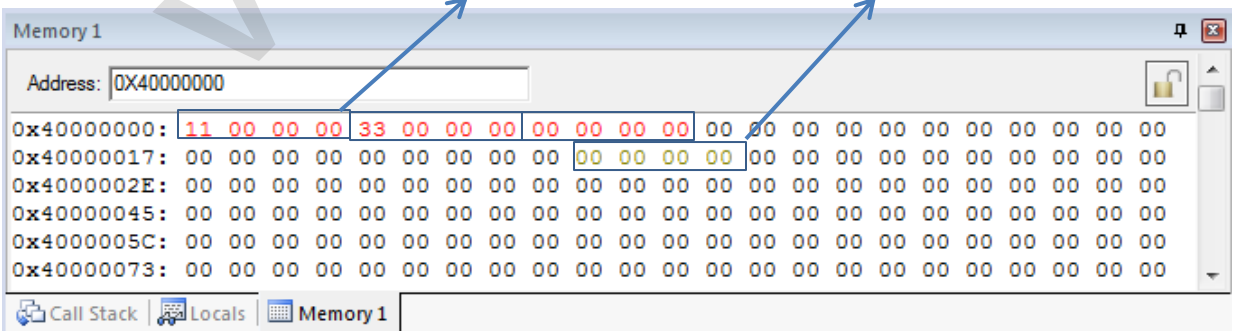
Experiment 5

Aim: Develop an ALP to find the largest/smallest number in an array of 32 numbers.

```
AREA LARGEST, CODE, READONLY
        ENTRY
        MOV R5,#3              ; Count of elements in array (each 32-bit)
        LDR R0,INPUT
        LDR R2,[R0]
        SUBS R5,#1             ; Decrement the count of array elements
NEXT    ADD R0,#4              ; Increment the array pointer to next number
        LDR R3,[R0]
        CMP R2,R3
        BLS LARGE              ; Use BHS to find the largest number
        MOV R2,R3
LARGE   SUBS R5,#1
        BNE NEXT
        LDR R1,RESULT
        STR R2,[R1]
HERE    B HERE
INPUT   DCD   0X40000000
RESULT  DCD   0X40000020
        END
```

Results

32 bit array [0] to array [2] Smallest or Largest number



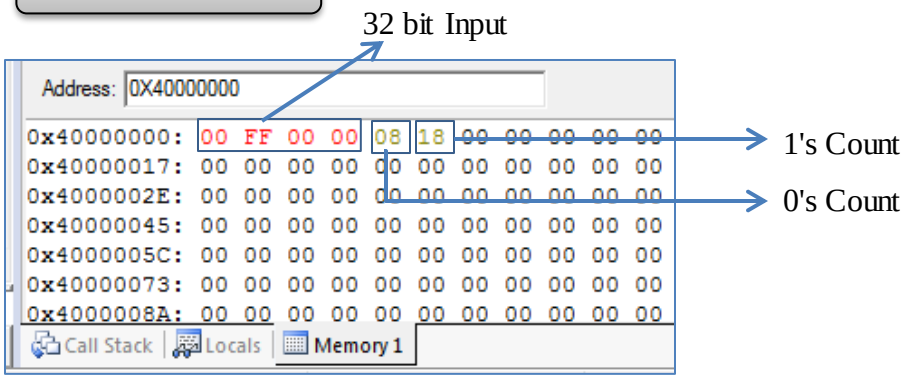
Observation

Experiment 6

Aim: Develop an ALP to count the number of ones and zeros in two consecutive memory locations.

```
        AREA PROGRAM, CODE, READONLY
        ENTRY
        LDR R0,MEMORY
        LDR R1,[R0]
        MOV R4, #32
ROTATE  RORS R1,#1
        BCS ONES
        ADD R3,R3,#1
        B NEXT
ONES    ADD R2,R2,#1
NEXT    ADD R4,R4,#-1
        CMP R4,#0
        BNE ROTATE
        ADD R0,R0,#04
        STRB R2,[R0]
        ADD R0,R0,#1
        STRB R3,[R0]
HERE    B HERE
MEMORY  DCD 0X40000000
        END
```

Results



Observation

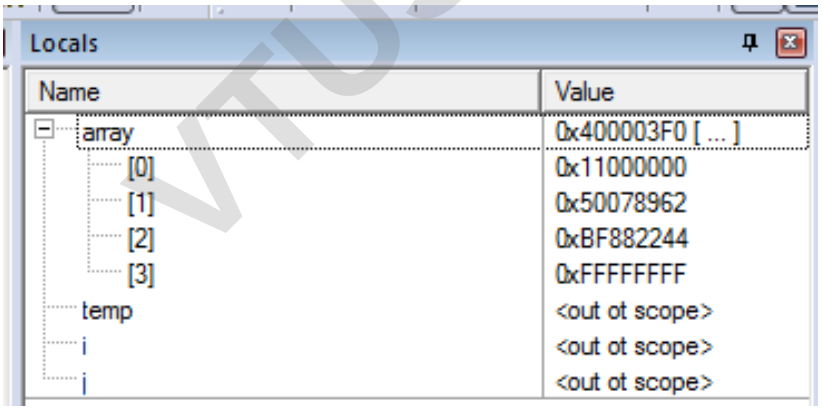
Experiment 7

Aim: Simulate a program in C for ARM microcontroller using KEIL to sort the numbers in ascending/descending order using bubble sort.

```
#include <LPC214X.H>

int main()
{
        unsigned long int array[]={0x11000000, 0xffffffff, 0x50078962, 0xbf882244};
        unsigned long int temp=0,i,j;
for(i=0;i<4;i++)
{
                for(j=0;j<3;j++)
                {
                        if(array[j]>array[j+1])
                        {
                                temp=array[j+1];
                                array[j+1]=array[j];
                                array[j] =temp;
                        }
                }
}while(1);
}
```

Results



Observation

Experiment 8

Aim: Simulate a program in C for ARM microcontroller to find factorial of a number.

```
#include <LPC214X.H>
#include <stdio.h>

int main()
{
        unsigned int i, num=5;
        unsigned long fact=1, factorial=1 ;

        for(i=1; i<=num; i++)
        {       factorial *=i;
        }
if(i==num)
 factorial=fact;

}
```

Results

Locals

Name	Value
i	0x00000006
num	0x00000005
fact	0x00000001
factorial	0x00000078

Observation

Experiment 9

Aim: Simulate a program in C for ARM microcontroller to demonstrate case conversion of characters from upper to lowercase and lower to uppercase.

```
#include<LPC214X.H>
char *msg = "hello world";
int main()
{

PINSEL0 = 0X5; U0LCR = 0X83;

U0DLM= 0X00; U0DLL = 0X61; U0LCR = 0X03;

while(*msg != 0x00)
{
while (!(U0LSR &0X20));
if(*msg>=97)
        {
                *msg=*msg-32;
        }
 else
   {
                *msg=*msg+32;
        }
U0THR =*msg;
 msg++;
}
}
```

Experiment 10

Aim: Demonstrate enabling and disabling of Interrupts in ARM.

Demonstrate the use of an external interrupt to toggle an LED On/Off.

```
#include <LPC214x.H>
void init_ext_interrupt(void);
irq void Ext_ISR(void);
int main (void)
{ init_ext_interrupt();                    // initialize the external interrupt while (1)
{}
}
void init_ext_interrupt()                  //Initialize Interrupt
{       EXTMODE = 0x4;                     //Edge sensitive mode on
        EINT2 EXTPOLAR &= ~(0x4);          //Falling Edge Sensitive
        PINSEL0 = 0x0000C000;              //Select Pin function P0.15 as EINT2VIC
        IntSelect&= ~ (1<<16);             // EINT2 selected as IRQ 16 VICVectAddr5
        = (unsigned int) Ext_ISR;          // address of the
        ISR VICVectCntl5 = (1<<5) | 16;
        VICVectAddr5=( unsigned int) Ext_ISR;
        VICVectCntl5=(1<<5) | 16;
        VICIntEnable = (1<<16);            // EINT2 interrupt enabled
        EXTINT &= ~(0x4);
}
irq void Ext_ISR(void)                     // Interrupt Service Routine-ISR
{
IO1DIR |=0xFF000000;                       //make Port P1.31 to P1.24 as output
IO1PIN ^= 0xFF000000;                      // Turn ON Buzzer
//delay(10);
//IO1PIN |= 0x00000000;//(1<<25);          // Turn OFF Buzzer EXTINT |= 0x4; //clear
interrupt
VICVectAddr = 0;                           // End of interrupt execution
```