

# **ATME COLLEGE OF ENGINEERING**

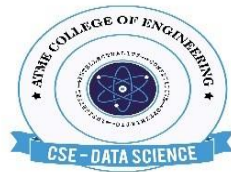
**13<sup>th</sup> KM Stone, Bannur Road, Mysore - 560 028**



# **A T M E**

**College of Engineering**

## **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING (DATA SCIENCE)**



**(ACADEMIC YEAR 2024-25)**

## **LABORATORY MANUAL**

**SUBJECT: MICROCONTROLLERS LABORATORY**

**SUB CODE: BCS402**

**SEMESTER: IV-2022 CBCS Scheme**

**Prepared By**

**Approved By**

**Dr.Vinod Kumar P  
Faculty In-charge**

**Dr. ANITHA D B  
HOD, CSE-DSE**

# **INSTITUTIONAL MISSION AND VISION**

## **Objectives**

- ☐ To provide quality education and groom top-notch professionals, entrepreneurs and leaders for different fields of engineering, technology and management.
- ☐ To open a Training-R & D-Design-Consultancy cell in each department, gradually introduce doctoral and postdoctoral programs, encourage basic & applied research in areas of social relevance, and develop the institute as a center of excellence.
- ☐ To develop academic, professional and financial alliances with the industry as well as the academia at national and transnational levels
- ☐ To develop academic, professional and financial alliances with the industry as well as the academia at national and transnational levels.
- ☐ To cultivate strong community relationships and involve the students and the staff in local community service.
- ☐ To constantly enhance the value of the educational inputs with the participation of students, faculty, parents and industry.

## **Vision**

- ☐ Development of academically excellent, culturally vibrant, socially responsible and globally competent human resources.

## **Mission**

- To keep pace with advancements in knowledge and make the students competitive and capable at the global level.
- To create an environment for the students to acquire the right physical, intellectual, emotional and moral foundations and shine as torch bearers of tomorrow's society.

- To strive to attain ever-higher benchmarks of educational excellence.

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING AND ENGINEERING**  
**(DATA SCIENCE &ENGINEERING)**

**Vision of The Department**

- To impart technical education in the field of data science of excellent quality with a high level of professional competence, social responsibility, and global awareness among the students

**Mission**

- To impart technical education that is up to date, relevant and makes students competitive and employable at global level
- To provide technical education with a high sense of discipline, social relevance in an intellectually, ethically and socially challenging environment for better tomorrow
- Educate to the global standards with a benchmark of excellence and to kindle the spirit of innovation.

**Program Outcomes(PO)**

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

- **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.
- **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.
- **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.
- **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
- **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.
- **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- **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.

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

### **Program Specific Outcomes (PSOs)**

- PSO1: Develop relevant programming skills to become a successful data scientist
- PSO2: Apply data science concepts and algorithms to solve real world problems of the society
- PSO3: Apply data science techniques in the various domains like agriculture, education healthcare for better society

### **Program Educational Objectives (PEOs):**

**PEO1:** Develop cutting-edge skills in data science and its related technologies, such as machine learning, predictive analytic, and data engineering.

**PEO2:** Design and develop data-driven solutions to real-world problems in a business, research, or social environment.

**PEO3:** Apply data engineering and data visualization techniques to discover, investigate, and interpret data.

**PEO4:** Demonstrate ethical and responsible data practices in problem solving

**PEO5:** Integrate fields within computer science, optimization, and statistics to develop better solutions

## CONTENTS

Sl.NO	Experiments	Page No
1	Using Keil software, observe the various Registers, Dump, CPSR, with a simple Assembly Language Programs (ALP)..	7
2	Develop and simulate ARM ALP for Data Transfer, Arithmetic and Logical operations (Demonstrate with the help of a suitable program).	9
3	Develop an ALP to multiply two 16-bit binary numbers	9
4	Develop an ALP to find the sum of first 10 integer numbers.	10
5	Develop an ALP to find the largest/smallest number in an array of 32 numbers.	11
6	Develop an ALP to count the number of ones and zeros in two consecutive memory locations.	14
7	Simulate a program in C for ARM microcontroller using KEIL to sort the numbers in ascending/descending order using bubble sort.	16
8	Simulate a program in C for ARM microcontroller to find factorial of a number.	17
9	Simulate a program in C for ARM microcontroller to demonstrate case conversion of characters from upper to lowercase and lower to uppercase.	18
10	Demonstrate enabling and disabling of Interrupts in ARM.	19

## INTRODUCTION

An embedded system is an electronic/electro-mechanical system designed to perform a specific function and is a combination of both hardware and firmware (software). An embedded system combines mechanical, electrical, and chemical components along with a computer, hidden inside, to perform a single dedicated purpose.

There are more computers on this planet than there are people, and most of these computers are single-chip microcontrollers that are the brains of an embedded system. Embedded systems are a ubiquitous component of our everyday lives. We interact with hundreds of tiny computers every day that are embedded into our houses, our cars, our bridges, our toys, and our work. As our world has become more complex, so have the capabilities of the microcontrollers embedded into our devices. Therefore, the world needs a trained workforce to develop and manage products based on embedded microcontrollers.

A general-purpose computing system is a combination of generic hardware and general-purpose operating system for executing a variety of application, whereas an embedded system is a combination system is a combination of special purpose hardware and embedded OS/firmware for executing a specific set of applications.

The ARM microcontroller stands for Advance Risk Machine; it is one of the extensive and most licensed processor cores in the world. The first ARM processor was developed in the year 1978 by Cambridge University, and the first ARM RISC processor was produced by the Acorn Group of Computers in the year 1985.

These processors are specifically used in portable devices like digital cameras, mobile phones, home networking modules and wireless communication technologies and other embedded systems due to the benefits, such as low power consumption, reasonable performance, etc. This article gives an overview of ARM architecture with each module's principle of working.

Keil MDK is the complete software development environment for a wide range of ARM Cortex-M based microcontroller devices. MDK includes the  $\mu$ Vision IDE and debugger, Arm C/C++ compiler, and essential middleware components. It supports all silicon vendors with more than 6,000 devices and is easy to learn and user.

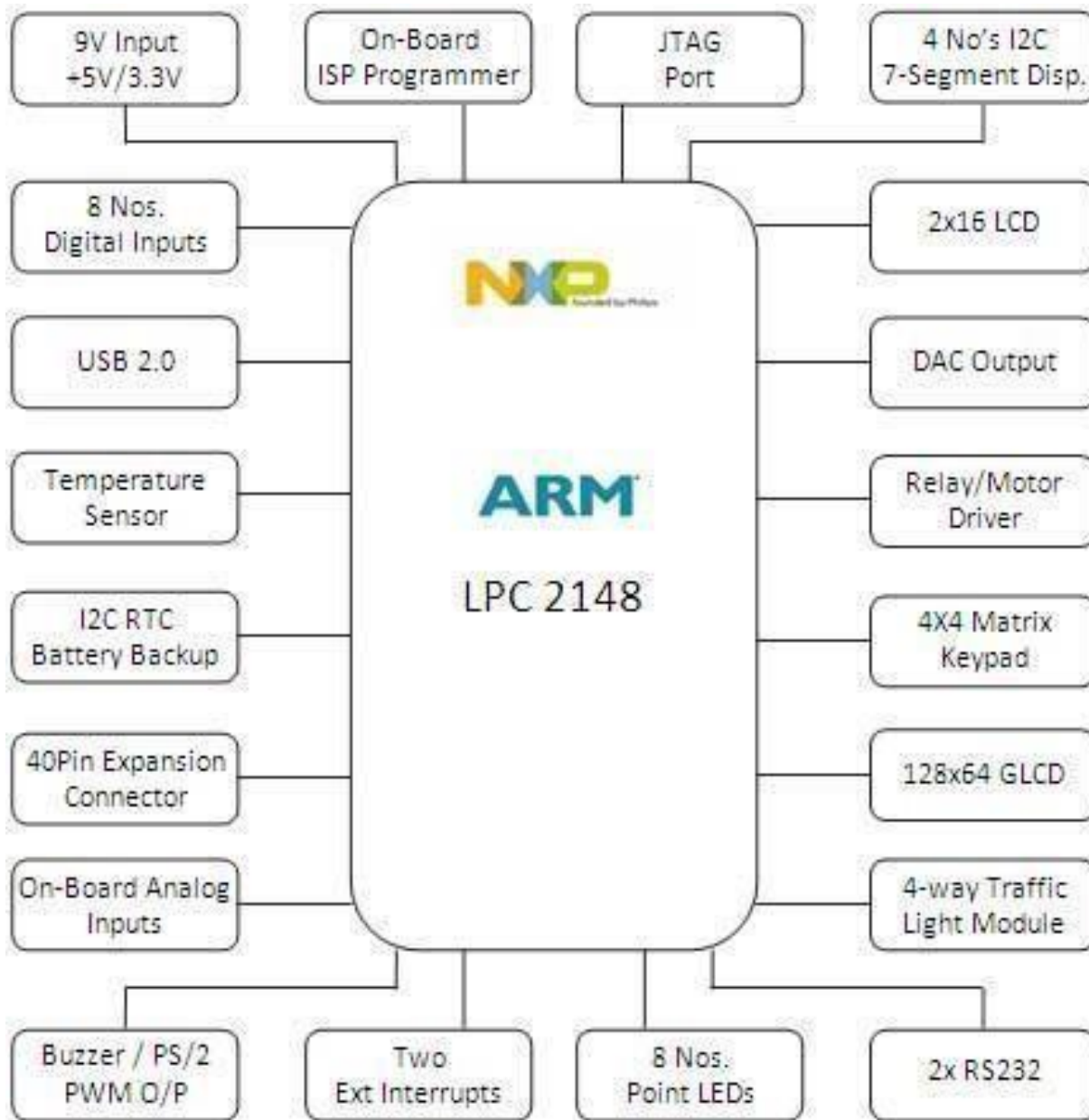
## Features of ARM Processor

AIM: To study of ARM processor system and describe the features of architecture

### Features of ARM DEVELOPMENT KIT Processor:

- 16-bit/32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package. 8 kB to 40 kB of on-chip static RAM and 32 kB to 512 kB of on-chip flash memory. 128-bit wide interface/accelerator enables high-speed 60 MHz operation. In-System/In-Application Programming (ISP/IAP) via on-chip boot loader software.
- Single flash sector/full chip erase in 400 ms and programming of 256 bytes in 1 ms. USB 2.0 Full-speed compliant device controller with 2 kB of endpoint RAM. The LPC2146/48 provides 8 kB of on-chip RAM accessible to USB by DMA.
- One or two (LPC2141/42 vs. LPC2144/46/48) 10-bit ADCs provide a total of 6/14 analog inputs, with conversion times as low as 2.44  $\mu$ s per channel. Single 10-bit DAC provides variable analog output (LPC2142/44/46/48 only). Two 32-bit timers/external event counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.
- Low power Real-Time Clock (RTC) with independent power and 32 kHz clock input. Multiple serial interfaces including two UARTs (16C550), two Fast I2Cbus (400 kbit/s), SPI and SSP with buffering and variable data length capabilities.
- Vectored Interrupt Controller (VIC) with configurable priorities and vector addresses. Up to 45 of 5 V tolerant fast general purpose I/O pins in a tiny LQFP64 package. Up to 21 external interrupt pins available.
- 60MHz maximum CPU clock available from programmable on-chip PLL with settling time of 100 $\mu$ s. On-chip integrated oscillator operates with an external crystal from 1 MHz to 25 MHz. Power saving modes include Idle and Power down.
- Individual enable/disable of peripheral functions as well as peripheral clock scaling for additional power optimization. Processor wake-up from Power-down mode via external interrupt or BOD. Single power supply chip with POR and BOD circuits: CPU operating voltage range of 3.0 V to 3.6 V (3.3 V  $\pm$  10 %) with 5 V tolerant I/O pads.

## General Block Diagram:

*Power Supply:*

- The external power can be AC or DC, with a voltage between (9V/12V, 1A output) at 230V AC input. The ARM board produces +5V using an LM7805 voltage regulator, which provides supply to the peripherals.
- LM1117 Fixed +3.3V positive regulator used for processor & processor related peripherals.

*Flash Programming Utility*

- NXP (Philips) NXP Semiconductors produce a range of Microcontrollers that feature both on-chip Flash memory and the ability to be reprogrammed using In-System Programming technology.

**PIN DIAGRAM****On-board Peripherals:**

- 8-Nos. of Point LED's (Digital Outputs)
- 8-Nos. of Digital Inputs (slide switch)
- 2 Lines X 16 Character LCD Display
- I2C Enabled 4 Digit Seven-segment display
- 128x64 Graphical LCD Display
- 4 X 4 Matrix keypad
- Stepper Motor Interface
- 2 Nos. Relay Interface
- Two UART for serial port communication through PC
- Serial EEPROM
- On-chip Real Time Clock with battery backup
- PS/2 Keyboard interface(Optional)
- Temperature Sensor
- Buzzer(Alarm Interface)
- Traffic Light Module(Optional)

**KEIL UVISION4 IDE INSTALLATION:**

*Installation of keilUvision4 as follows.*

1. Go to **EXE** folder and then **uvision4.2** in the CD and run **Keil4 Arm.exe** file.
2. **Next**
3. Click on the option “I agree to all the terms of...” and then give **Next**
4. **Next**
5. Give name and the mail id (it might be any mail id) and then **Next**
6. Click **Finish** to complete the installation.

**PROJECT CREATION IN KEILUV4 IDE:**

7. Create a project folder before creating NEW project.
8. Open **Keil uVision4 IDE** software by double clicking on “Keil Uvision4” icon.
9. Go to “**Project**” then to “**New uVision Project**” and save it with a name in the respective project folder, already you created.
10. Select the device as “**NXP**” In that “**LPC2148**” then press **OK** and then press “**YES**” button to add “**startup’s**” file.
11. In **startup** file go to **Configuration Wizard**. In **Configuration Wizard** window uncheck **PLL Setup** and check **VPBDIV Setup**.
12. Go to “**File**” In that “**New**” to open an editor window. Create your source file and use the header file “**lpc21xx.h**” in the source file and save the file. **Colour syntax highlighting** will be enabled once the file is saved with a extension such as “.C”.
13. Right click on “**Source Group 1**” and select the option “**Add Existing Files to Group Source Group1**” add the \*.C source file(s) to the group.
14. After adding the source file you can see the file in Project Window.
15. Then go to “**Project**” in that “**Translate**” to compile the File (s). Check out the **Build output** window.
16. Right click on **Target1** and select  
**options for Target Target1**. Then go to  
option “**Target**” in that
  1. Xtal 12.0MHz
  2. Select “**Use MicroLIB**”.
  3. Select **IROM1** (starting 0x0 size 0x80000).
  4. Select **IRAM1** (starting 0x40000000 size 0x8000).

**Then go to option “Output”**

1. Select “Create Hex file”.
2. Then go to option “Linker”

Select “Use Memory Layout for Target Dialog”. To come out of this window press OK.

17. Go to “**Project**” in that “**Build Target**” for building all source files such as “.C”, “.ASM”, “.h”, files, etc... This will create the \*.HEX file if no warnings & no Errors. Check out the **Build output** window.

Module-1  
Program

**Using Kiel software, observe the various registers, dump, CPSR, with a simple ALP program**

**1. Write a Program to store the data 06 and 07 in registers R6 and R7 respectively and also to move the data**

**in R6 to the registers from R0 to R5 and the data in R7 to the registers from R8 to R12.**

```
AREA PROGRAM, CODE, READONLY
MOV R6, #06 //MOV R6, 0X06
MOV R7, #07 //MOV R7, 0X07
MOV R0, R6
MOV R1, R6
MOV R2, R6
MOV R3, R6
MOV R4, R6
MOV R5, R6
MOV R8, R7
MOV R9, R7
MOV R10, R7
MOV R11, R7
MOV R12, R7
Stop B Stop
END
```

Results

**2. Write a Program to add two numbers**

```
AREA PROGRAM, CODE, READONLY
MOV R5, #05
MOV R7, #07
ADD R2, R5, R7
Stop B Stop
END
```

Result

**3. Write a Program to Set carry flag(C)**

```
AREA PROGRAM, CODE, READONLY
MOV R1, 0X80000004
MOVS R0, R1, LSL #1 //MOVS R0, R1, LSR #3
stop_here B stop_here
END
```

Result

**4. Write a Program to Set Zero flag(Z)**

```
AREA PROGRAM,CODE,READONLY
    MOV R0,0x01
    SUBS R2,R0,R0
    stop_here B stop_here
    END
```

Result

**5. Write a Program to Set Negative flag (N)**

```
AREA PROGRAM,CODE,READONLY
    MOV R1,0X77
    RSBS R0,R1,#76
    stop_here B stop_here
    END
```

Result

**6. Write a Program to add two 32 bit numbers. Read the numbers from the memory and store the sum in the memory**

```
AREA PROGRAM,CODE,READONLY
    LDR R1,=Data1
    LDR R2,=Data2
    LDR R3,=Result
    LDR R4,[R1]
    LDR R5,[R2]
    ADD R7,R4,R5
    STR R7,[R3]
    stop_here B stop_here
    Data1 DCD 0x12345678
    Data2 DCD 0x24531211
    AREA DATA3,DATA,READWRITE
    Result DCD 0x00000000
```

Result

**Module-2****Program No:01: Simple Loading and storing instructions****; simple LDR instructions**

```
AREA KA, CODE, READONLY
MOV R1, #0X40000000
LDRB R2, [R1]
LDRH R3, [R1]
LDR R4, [R1]
STOP B STOP
END
```

**Result:****Program No:2 Develop and simulate ARM ALP for Data Transfer, Arithmetic and Logical operations (Demonstrate with the help of a suitable program).**

```
AREA PROGRAM, CODE, READONLY
MOV R4, #7 ; TRANSFER DATA TO R4
MOV R5, #6 ; TRANSFER DATA TO R5
ADD R7, R5, R4; ; ARITHMETIC OPERATION
MOV R8, R7 ; REGISTER TRANSFER
AND R9, R8, R4 ; LOGICAL OPERATION
STOP B STOP
END ; MARK END OF FILE
```

**Result:****Program No:3 Develop an ALP to multiply two 16-bit binary numbers.**

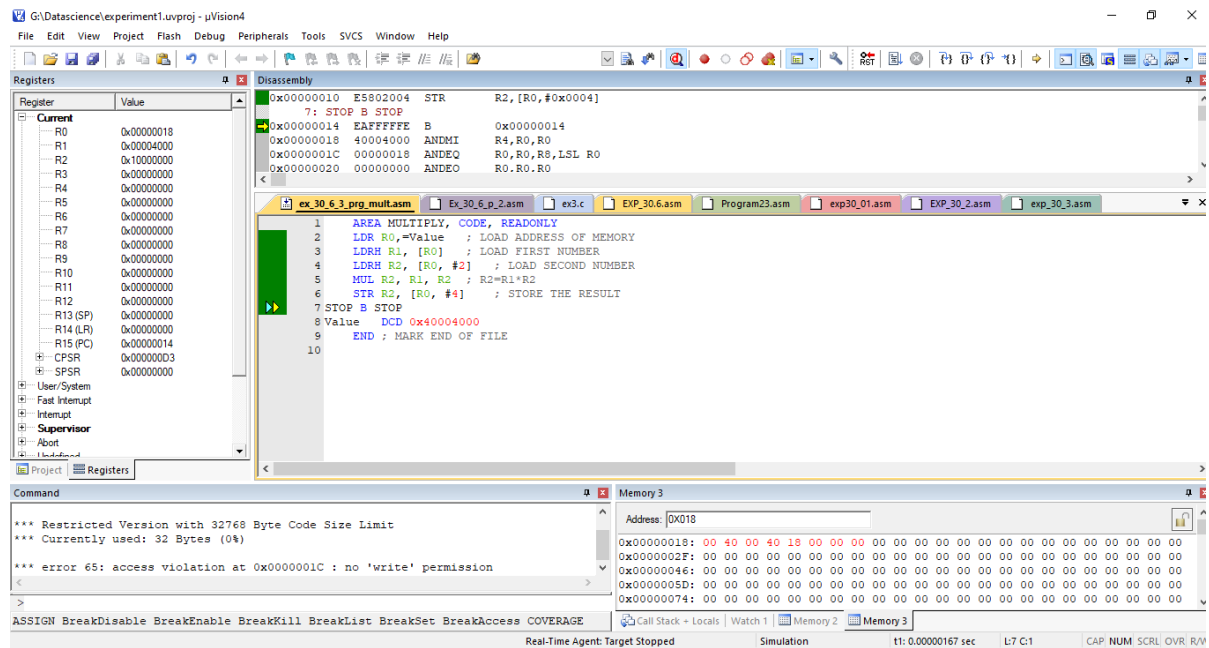
```
AREA MULTIPLY, CODE, READONLY
LDR R0, MEMORY; LOAD ADDRESS OF MEMORY
LDRH R1, [R0] ; LOAD FIRST NUMBER
LDRH R2, [R0, #2] ; LOAD SECOND NUMBER
```

```

MUL R2, R1, R2 ; R2=R1*R2
STR R2, [R0, #4] ; STORE THE RESULT
STOP B STOP
MEMORY DCD 0x40000000
END ; MARK END OF FILE

```

## Result



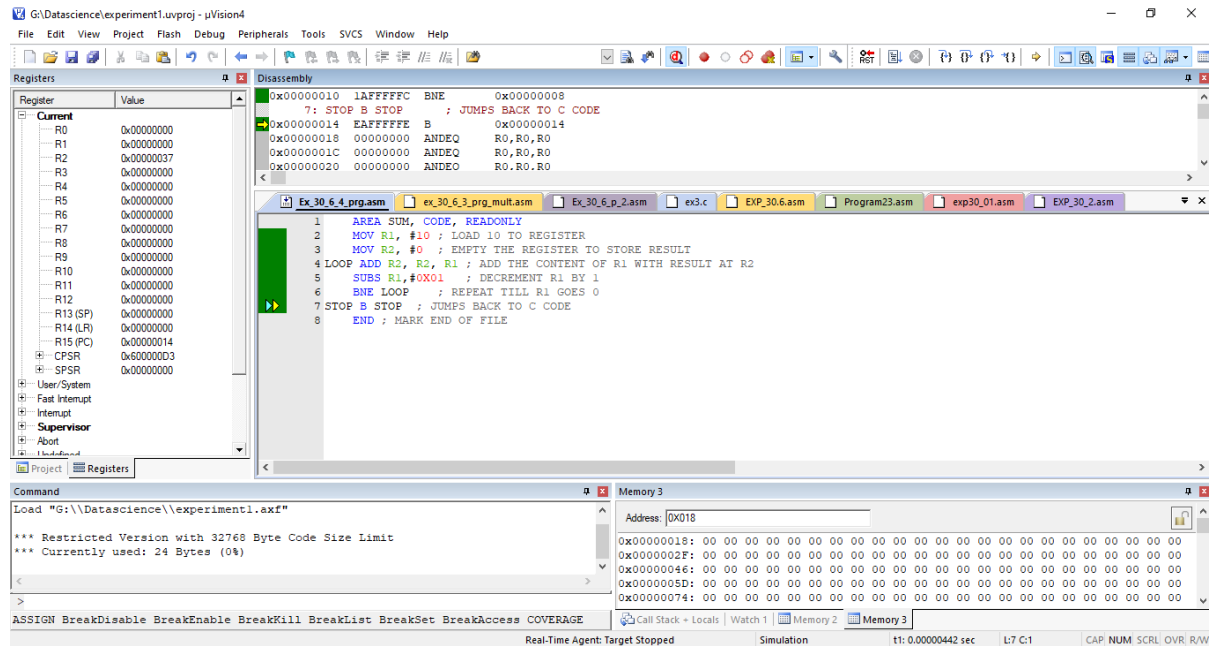
**Program No:4 Develop an ALP to find the sum of first 10 integer numbers.**

```

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
STOP B STOP ; JUMPS BACK TO C CODE
END ; MARK END OF FILE

```

## Result



**Program No:5 Develop an ALP to find the largest/smallest number in an array of 32 numbers.**

**LARGEST:****AREA LARGEST, CODE, READONLY**

**MOV R5,#6 ; INITIALISE COUNTER TO 6(I.E. N=7)**

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

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

**LOOP LDR R4,[R1],#4 ; WORD ALIGN TO 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**

**BNE LOOP ; LOOP BACK TILL ARRAY ENDS**

**LDR R4,=RESULT ; LOADS THE ADDRESS OF RESULT**

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

**XSS B XSS ; ARRAY OF 32 BIT NUMBERS(N=7)**

**VALUE1**

**DCD 0X55444444**

**DCD 0X33222222**

**DCD 0X22111111**

**DCD 0X66333333**

**DCD 0XBAAAAAAA**

**DCD 0X78888888**

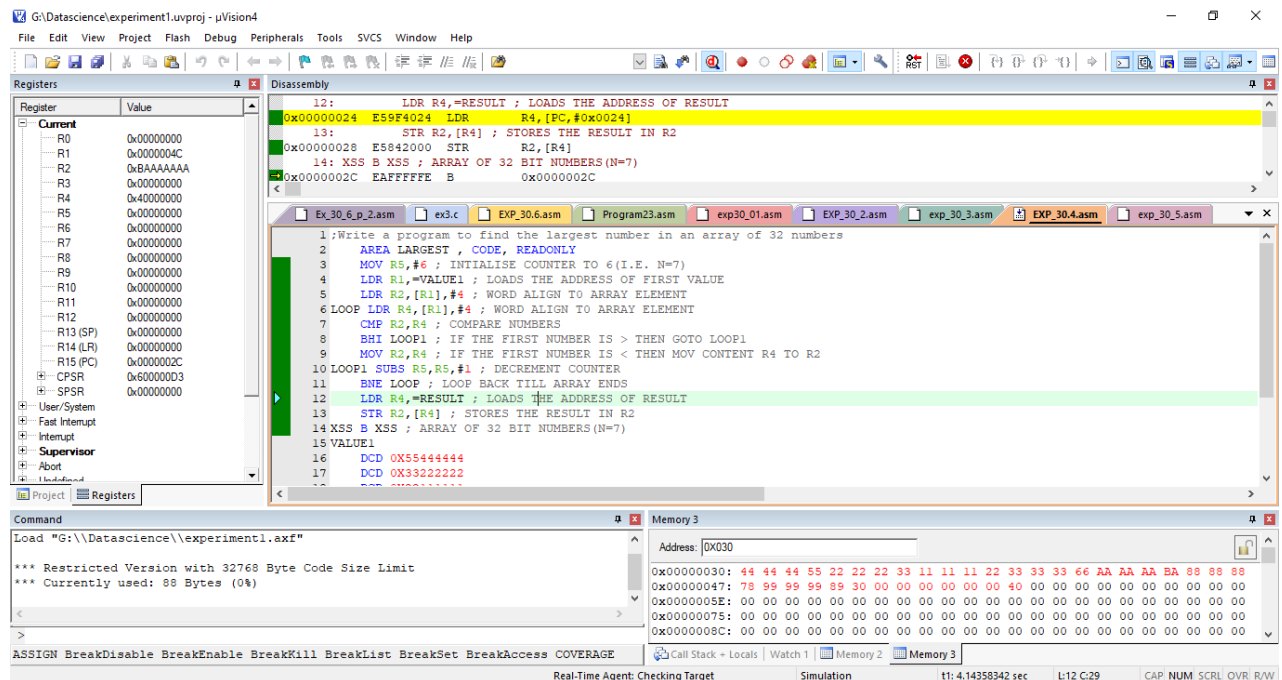
**DCD 0X89999999**

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

**RESULT DCD 0X0**

**END ; MARK END OF FILE**

Result:



**SMALLEST:**

**AREA SMALLEST , CODE, READONLY**

**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 TO ARRAY ELEMENT**

**LOOP LDR R4,[R1],#4 ; WORD ALIGN TO ARRAY ELEMENT**

**CMP R2,R4 ; COMPARE NUMBERS**

**BLS 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**

**BNE LOOP ; LOOP BACK TILL ARRAY ENDS**

**LDR R4,=RESULT ; LOADS THE ADDRESS OF RESULT**

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

**XSS B XSS ; ARRAY OF 32 BIT NUMBERS(N=7)**

**VALUE1**



**Program No:6 Develop an ALP to count the number of ones and zeros in two consecutive memory locations.**

**AREA ONEZERO , CODE, READONLY**

**MOV R2,#0 ; COUNTER FOR ONES**

**MOV R3,#0 ; COUNTER FOR ZEROS**

**MOV R7,#2 ; COUNTER TO GET TWO 32bit numbers**

**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)**

**BCS 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**

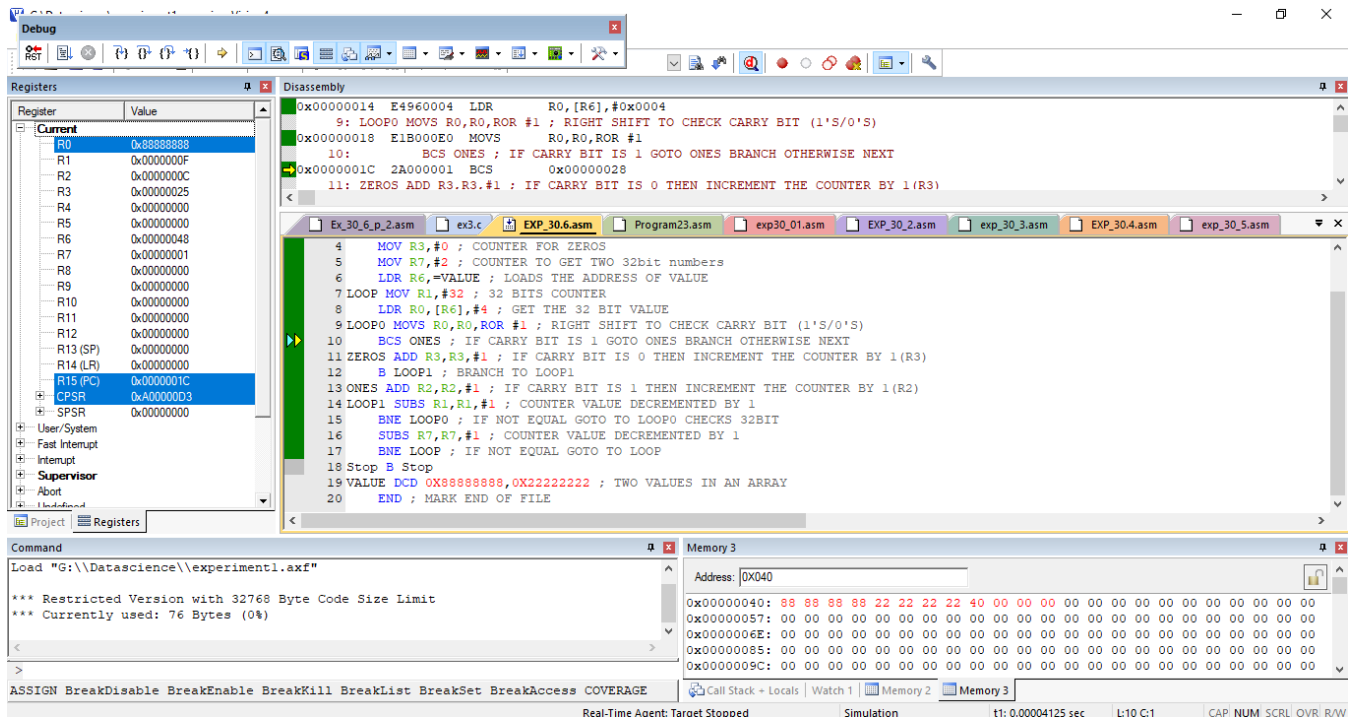
**BNE LOOP ; IF NOT EQUAL GOTO TO LOOP**

**Stop B Stop**

**VALUE DCD 0X88888888,0X22222222 ; TWO VALUES IN AN ARRAY**

**END ; MARK END OF FILE**

Result:



**Module-3:**

**Program No:7 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;
```

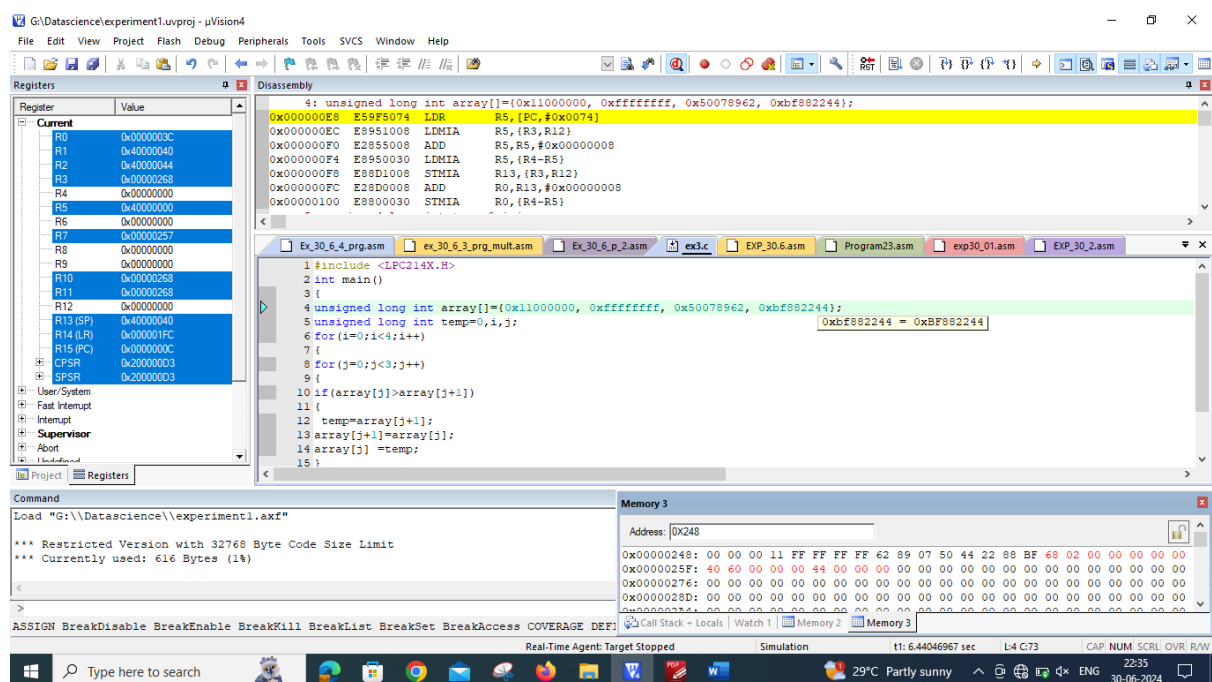
```
}
```

```
}
```

```
}
```

```
while(1);
```

```
}
```



**Program No:8 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;
}
```

Result:

**Program No:9 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++;
    }
}
```

Result:

**Module-4 & 5****Program No:10 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****Result:**