



A T M E

College of Engineering

13th KM Stone, Bannur Road, Mysore - 560 028

**Department of CSE–Artificial Intelligence & Machine Learning
(Academic Year 2024-25)**



LABORATORY MANUAL

SUBJECT: MICROCONTROLLERS & EMBEDDED SYSTEMS

SUB CODE: BC0601

SEMESTER: VI

SCHEME: 2022

Prepared By

**Ms. GEETHA.B
Instructor**

Verified By

**Dr.HUSSANA JOHAR R B
Assoc. Professor
Dept. of CSE-AI &ML**

Approved by

**Dr. ANIL KUMAR C.J
Assoc. Professor & Head
Dept. of CS-AI &ML**

Institute Vision

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

Institute 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 Vision

To impart technical education in the field of Artificial intelligence and machine learning of topnotch quality with a high level of professional competence, social obligation, and global cognizance among the students.

Department Mission

- To impart technical education that is up to date, relevant and makes students to compete at global level
- Fostering an ambiance where students can adopt the suitable moral, intellectual, emotional, and physical attributes to shine as the leaders of tomorrow's society.
- To strive to meet ever higher educational standard.

Program Outcomes (PO's)

- 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 life-long learning in the broadest context of technological change

Program Educational Objectives (PEO's):

PEO1: Graduates will be able to hone their problem-solving abilities and capacity to offer solutions to challenges that arise in the actual world.

PEO2: Able to design and develop AI based solutions to real-world problems in a business, research, or social environment.

PEO3: Graduates shall acquire and inculcate corporate culture, core attributes, and leadership qualities as well as professional etiquette's and lifelong learning.

Program Specific Outcomes (PSO's)

PSO1: Ability to design and develop artificial intelligent based solutions by applying optimal algorithms to solve real world issues.

PSO2: Ability to apply suitable AI tools and techniques to offer solutions in the various domains of engineering.

Microcontrollers & Embedded Systems		Semester	6
Course Code	BCO601	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/practical		
Sl.NO	Experiments		
1	Develop a program to multiply two 16 bit binary numbers.		
2	Write a program to find the sum of first 10 integer numbers.		
3	Write a program to find factorial of a number.		
4	Write a program to add an array of 16 bit numbers and store the 32 bit result in internal RAM		
5	Write a program to find the square of a number (1 to 10) using look-up table.		
6	Write a program to find the largest/smallest number in an array of 32 numbers .		
7	Display “Hello World” message using Internal UART.		
8	Interface a Stepper motor and rotate it in clockwise and anti-clockwise direction		
9	Display the Hex digits 0 to F on a 7-segment LED interface, with an appropriate delay in between		
10	Interface a 4x4 keyboard and display the key code on an LCD.		
Course outcomes (Course Skill Set): At the end of the course, the student will be able to:			
<ul style="list-style-type: none">● Explain the architectural features and instructions of ARM microcontroller● Apply the knowledge gained for Programming ARM for different applications.● Demonstrate Interfacing of external devices and I/O with ARM microcontroller.● Interpret the basic hardware components and their selection method based on the characteristics and attributes of an embedded system.● Develop the hardware /software co-design and firmware design approaches.			

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.

SEE for IPCC

Theory SEE will be conducted by University as per the scheduled timetable, with common question papers for the course (**duration 03 hours**)

1. The question paper will have ten questions. Each question is set for 20 marks.
2. There will be 2 questions from each module. Each of the two questions under a module (with a maximum of 3 sub-questions), **should have a mix of topics** under that module.
3. The students have to answer 5 full questions, selecting one full question from each module.
4. Marks scored by the student shall be proportionally scaled down to 50 Marks

The theory portion of the IPCC shall be for both CIE and SEE, whereas the practical portion will have a CIE component only. Questions mentioned in the SEE paper may include questions from the practical component.

CONTENTS

1.	INTRODUCTION	1
2.	LABORATORY SESSION 1 –KEIL INTRODUCTION	5
SI No.	Experiments	Page No
PART A		
1.	Write a program to multiply two 16 bit binary numbers.	7
2.	Write a program to find the sum of first 10 integer numbers.	8
3.	Write a program to find factorial of a number.	9
4.	Write a program to add an array of 16 bit numbers and store the 32 bit result in internal RAM	10
5.	Write a program to find the square of a number (1 to 10) using look-up table.	12
6.	Write a program to find the largest/smallest number in an array of 32 numbers	13
PART B		
7.	Display “Hello World” message using Internal UART.	19
8.	Interface a Stepper motor and rotate it in clockwise and anti-clockwise direction.	21
9.	Interface a 4x4 keyboard and display the key code on an LCD.	23
10.	Display the Hex digits 0 to F on a 7-segment LED interface, with an appropriate delay in between.	29

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.

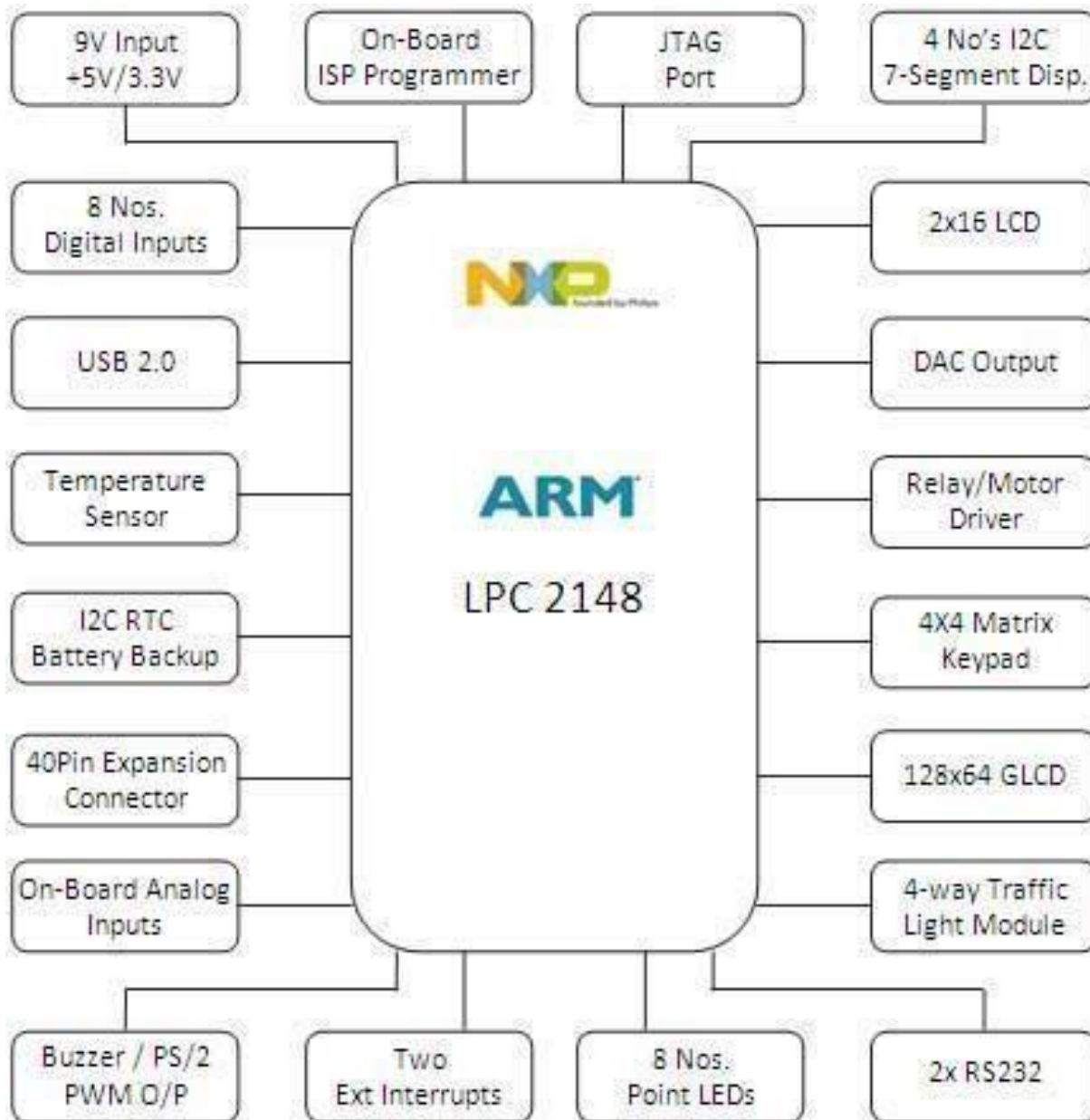
Keil MDK is the complete software development environment for a wide range of ARM Cortex-M based microcontroller devices. MDK includes the μ Vision IDE and debugger, Arm C/C++ compiler, and essential middleware components. It supports all silicon vendors with more than 6,000 devices and is easy to learn 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 μ 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 μ 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 ($3.3 \text{ V} \pm 10 \%$) 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 keil uVision4 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 Group 1**” “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.

PART -A

Program No.1:-Write a program to multiply two 16 bit binary numbers.

AIM: To write a program to multiply two 16 bit binary numbers using ARM7TDMI/LPC2148 using an evaluation board/simulator and the required software tool.

```
AREA MULTIPLY , CODE, READONLY
```

```
ENTRY                                ;MARK FIRST INSTRUCTION TO EXECUTE
```

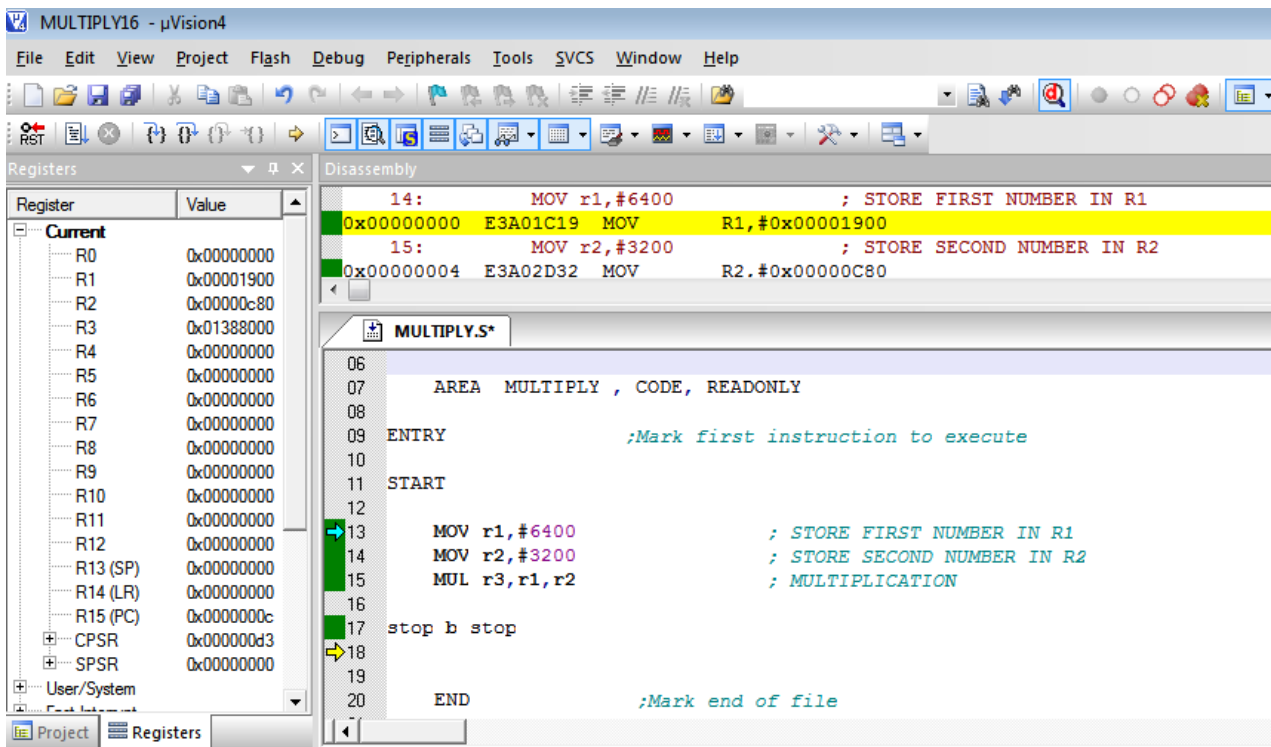
```
START
```

```
MOV R1,#6400                        ; STORE FIRST NUMBER IN R1
MOV R2,#3200                        ; STORE SECOND NUMBER IN R2
MUL R3,R1,R2                        ; MULTIPLICATION
```

```
STOP B STOP
```

```
END                                ;MARK END OF FILE
```

RESULT:



Program No.2:-Write a program to find the sum of first 10 integer numbers.

AIM: To write a program to find the sum of first 10 integer numbers using ARM7TDMI/LPC2148 using an evaluation board/simulator and the required software tool.

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

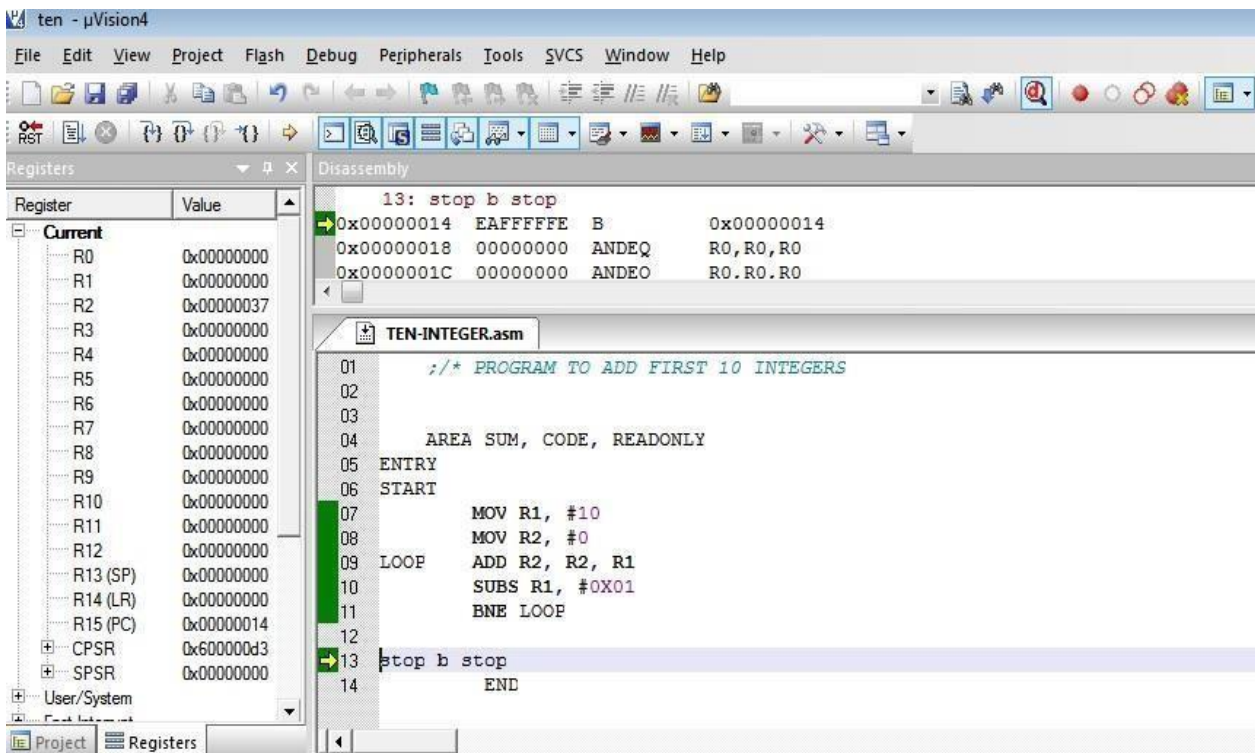
SUBS R1, #0X01 ; DECREMENT R1 BY 1

BNE LOOP ; REPEAT TILL R1 GOES 0

STOP B STOP ; JUMPS BACK TO C CODE

END

RESULT:



Program No.3 :-Write a program to find factorial of a number.

AIM: To write a program to find factorial of a number using ARM7TDMI/LPC2148 using an evaluation board/simulator and the required software tool.

AREA FACTORIAL , CODE, READONLY

ENTRY ;MARK FIRST INSTRUCTION TO EXECUTE

START

MOV R0, #4 ; STORE FACTORIAL NUMBER IN R0
MOV R1,R0 ; MOVE THE SAME NUMBER IN R1

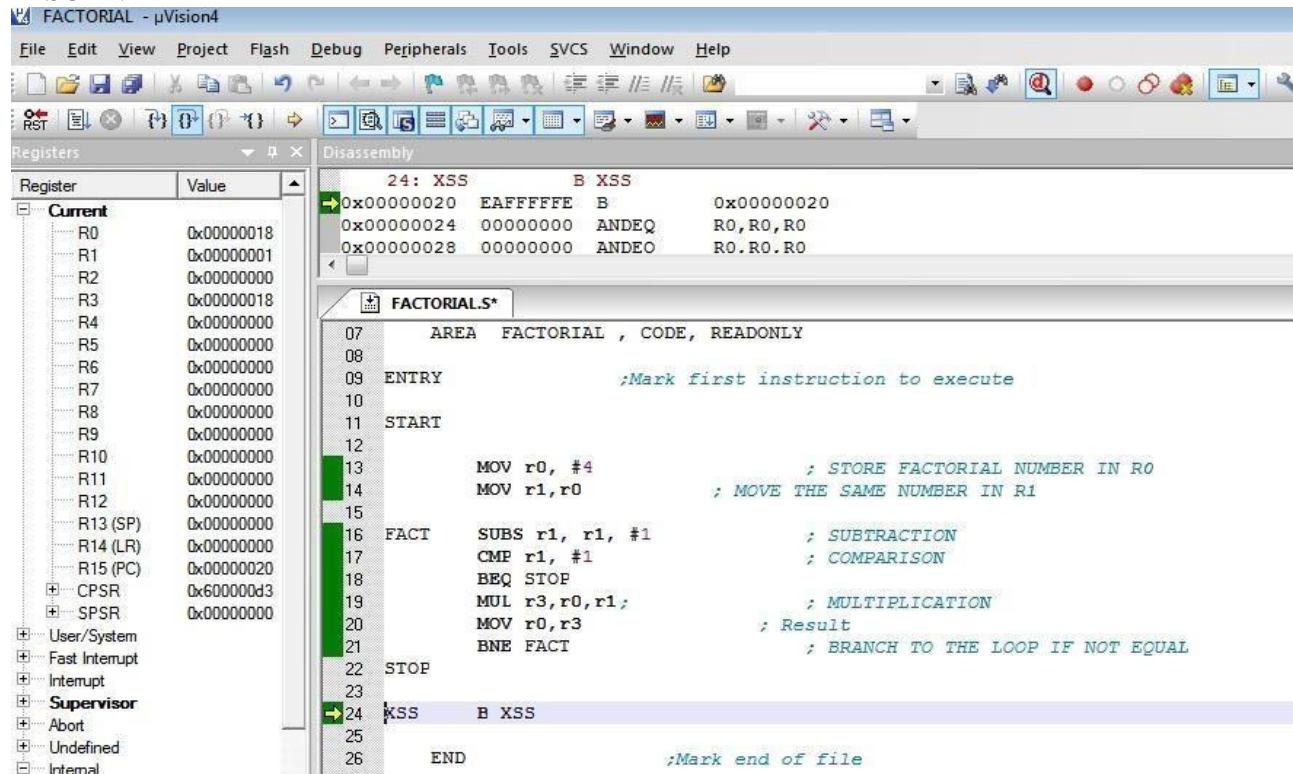
FACT SUBS R1, R1, #1 ; SUBTRACTION
CMP R1, #1 ; COMPARISON
BEQ STOP
MUL R3,R0,R1; ; MULTIPLICATION
MOV R0,R3 ; RESULT
BNE FACT ; BRANCH TO THE LOOP IF NOT EQUAL

STOP

STOP B STOP

END ;MARK END OF FILE

RESULT:



Program No.4:-Write a program to add an array of 16 bit numbers and store the 32 bit result in internal RAM

AIM: To write a program to add an array of 16 bit numbers and store the 32 bit result in internal RAM using an evaluation board/simulator and the required software tool.

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 FIST VALUE

LOOP

LDR R2,[R1],#2 ; WORD ALIGN TO ARRAY ELEMENT
 LDR R3,MASK ; MASK TO GET 16 BIT
 AND R2,R2,R3 ; MASK MSB
 ADD R0,R0,R2 ; ADD THE ELEMENTS
 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

XSS B XSS

MASK DCD 0X0000FFFF ; MASK MSB

VALUE1 DCW 0X1111,0X2222,0X3333,0XAAAA,0XB BBB,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

RESULT:

The screenshot displays the ADD - μVision4 IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. Below the menu is a toolbar with various icons for file operations, debugging, and viewing. The main workspace is divided into two panes: the Register window on the left and the Disassembly window on the right.

Register Window:

Register	Value
Current	
R0	0x00029997
R1	0x00000044
R2	0x0000cccc
R3	0x0000ffff
R4	0x40000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000030
CPSR	0x600000d3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	

Disassembly Window:

```

27: XSS B XSS
0x00000030 EAffffff B 0x00000030
0x00000034 0000ffff DD 0x0000ffff
0x00000038 22221111 DD 0x22221111

ADD.S
04  ;/* SET A BREAKPOINT AT NOP INSTRUCTION,RUN THE PROGRAM & CHECK THE RESULT
05
06
07      AREA  ADDITION , CODE, READONLY
08
09  ENTRY                      ;Mark first instruction to execute
10
11  START
12      MOV  R5,#6              ; INITIALISE COUNTER TO 6(i.e. N=6)
13      MOV  R0,#0              ; INITIALISE SUM TO ZERO
14      LDR  R1,=VALUE1         ; LOADS THE ADDRESS OF FIRST VALUE
15  LOOP
16      LDR  R2,[R1],#2         ; WORD ALIGN TO ARRAY ELEMENT
17      LDR  R3,MASK             ; MASK TO GET 16 BIT
18      AND  R2,R2,R3           ; MASK MSB
19      ADD  R0,R0,R2           ; ADD THE ELEMENTS
20      SUBS R5,R5,#1           ; DECREMENT COUNTER
21      CMP  R5,#0              ;
22      BNE  LOOP               ; LOOK BACK TILL ARRAY ENDS
  
```

Program No.5:- Write a program to find the square of a number (1 to 10) using look-up table.

AIM: To write a program to find the square of a number (1 to 10) using look-up table. using ARM7TDMI/LPC2148 using an evaluation board/simulator and the required software tool.

```

        AREA SQUARE , CODE, READONLY
ENTRY                                     ;MARK FIRST INSTRUCTION TO EXECUTE

START
    LDR    R0, = TABLE1                ; LOAD START ADDRESS OF LOOKUP TABLE
    LDR R1, = 3                          ; LOAD NO WHOSE SQUARE IS TO BE FIND
    MOV R1, R1, LSL#0X2                 ; GENERATE ADDRESS CORRESPONDING TO SQUARE OF GIVEN NO
    ADD R0, R0, R1                       ; LOAD ADDRESS OF ELEMENT IN LOOKUP TABLE
    LDR R3, [R0]                        ; GET SQUARE OF GIVEN NO IN R3

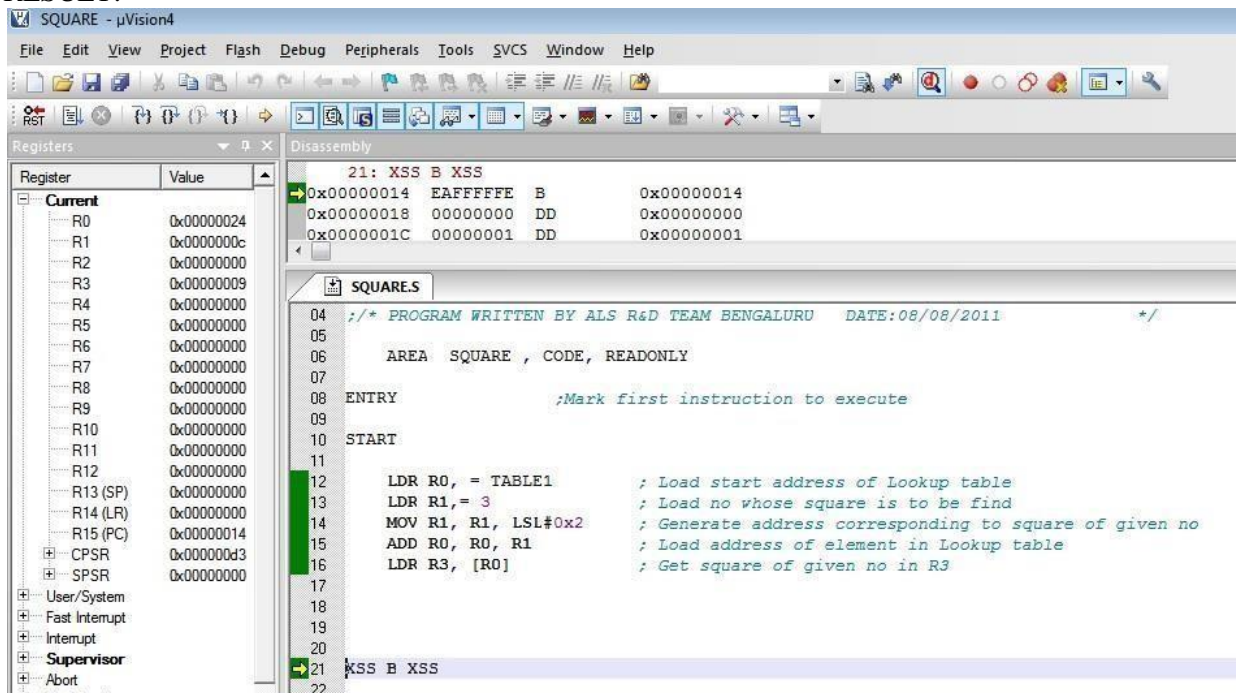
STOP B STOP                             ;LOOKUP TABLE CONTAINS SQUARES OF NOS FROM 0
                                        ;TO 10 (IN HEX)

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                                       ; MARK END OF FILE

```

RESULT:



Program No.6:- Write a program to find the largest/smallest number in an array of 32 numbers.

AIM: To write a program to find the largest/smallest number in an array of 32 numbers using ARM7TDMI/LPC2148 using an evaluation board/simulator and the required software tool.

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 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
    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

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

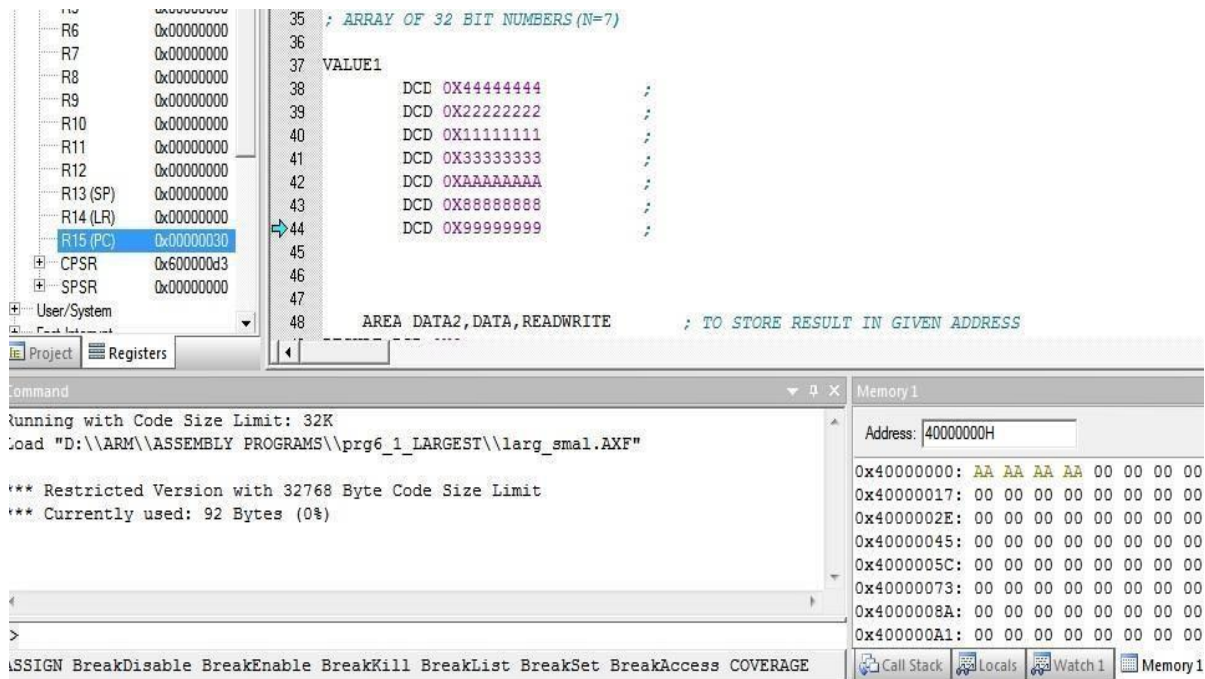
VALUE1
    DCD 0X44444444
    DCD 0X22222222
    DCD 0X11111111
    DCD 0X33333333
    DCD 0XAAAAAAAA
    DCD 0X88888888
    DCD 0X99999999

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

END                                  ; MARK END OF FILE

```

RESULT:



AREA SMALLEST , 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 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
    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

XSS B XSS                          ; 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

RESULT:

The screenshot displays an ARM assembler interface with the following components:

- Registers Panel:** Shows registers R5 through R15 (PC), CPSR, and SPSR, all with values of 0x00000000.
- Assembly Code:**

```

33
34
35 ; ARRAY OF 32 BIT NUMBERS (N=7)
36
37 VALUE1
38 DCD 0X44444444 ;
39 DCD 0X22222222 ;
40 DCD 0X11111111 ;
41 DCD 0X22222222 ;
42 DCD 0XAAAAAAAA ;
43 DCD 0X88888888 ;
44 DCD 0X99999999 ;
45
46

```
- Command Window:**

```

Running with Code Size Limit: 32K
Load "D:\\ARM\\ASSEMBLY PROGRAMS\\prg6_2_SMALLEST\\SMAL_LARG.AXF"

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 92 Bytes (0%)

```
- Memory Window:** Shows a memory dump starting at address 40000000H. The first few lines show hexadecimal values and their byte representations:

```

0x40000000: 11 11 11 11 00 00 00 00
0x40000017: 00 00 00 00 00 00 00 00
0x4000002E: 00 00 00 00 00 00 00 00
0x40000045: 00 00 00 00 00 00 00 00
0x4000005C: 00 00 00 00 00 00 00 00
0x40000073: 00 00 00 00 00 00 00 00
0x4000008A: 00 00 00 00 00 00 00 00
0x400000A1: 00 00 00 00 00 00 00 00

```
- Bottom Panel:** Includes a status bar with "ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE" and a tabbed interface with "Call Stack", "Locals", "Watch1", and "Memory1".

PART- B

PROCEDURE: PROJECT CREATION IN KEILUV4 IDE:

1. Create a project folder before creating NEW project.
 2. Open **Keil uVision4 IDE** software by double clicking on “Keil Uvision4” icon.
 3. Go to “**Project**” then to “**New uVision Project**” and save it with a name in the respective project folder, already you created.
 4. Select the device as “**NXP**” In that “**LPC2148**” then press **OK** and then press “**YES**” button to add “**startup.s**” file.
 5. In **startup** file go to **Configuration Wizard**. In **Configuration Wizard** window uncheck **PLL Setup** and check **VPBDIV Setup**.
 6. 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. **Color syntax highlighting** will be enabled once the file is saved with a extension such as “.C”.
 7. Right click on “**Source Group 1**” and select the option “**Add Existing Files to Group Source Group 1**” “add the *.C source file(s) to the group.
 8. After adding the source file you can see the file in Project Window.
 9. Then go to “**Project**” in that “**Translate**” to compile the File (s). Check out the **Build output** window.
 10. 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).
 1. 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**.
11. 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.

FLASHMAGIC:**Options -> Advanced options -> Hardware Config**

Enable these options only

Use DTR and RTS to control RST and ISP pin Keep RTS asserted while COM port open

Press OK then do the below settings

Step1. Communications:

1. Device : LPC2148
2. Com Port : COM1
3. Baud Rate: 9600
4. Interface : None(ISP)
5. Oscillator : 12MHz

Step2. ERASE:

1. Select "Erase Blocks Used By Hex File".

Step3. Hex file:

1. Browse and select the Hex file which you want to download.

Step4. Options

1. Select "Verify after programming".

Step5. Start:

2. Click Start to download the hex file to the controller.

After downloading the code the program starts executing in the hardware, then remove the ISP jumper JP7.

Program No.7:-Display “Hello World” message using Internal UART.

AIM: To write a program to Display “Hello World” message using Internal UART using an ARM7TDMI/LPC2148 evaluation board using evaluation version of Embedded 'C' & Keil Uvision-4 tool/compiler.

```
#include <lpc214x.h>
void uart_interrupt(void)__irq;

unsigned char temp;
unsigned char rx_flag=0,tx_flag=0;

int main(void)
{
    PINSEL0=0X00000005;           //select TXD0 and RXD0 lines
    U0LCR = 0X00000083;           //enable baud rate divisor loading and
    U0DLM = 0X00;                 //select the data format
    U0DLL = 0x13;                 //select baud rate 9600 bps
    U0LCR = 0X00000003;
    U0IER = 0X03;                 //select Transmit and Receive interrupt

    VICVectAddr0 = (unsigned long)uart_interrupt; //UART 0 INTERRUPT
    VICVectCntl0 = 0x20|6;        // Assign the VIC channel uart-0 to interrupt priority 0
    VICIntEnable = 0x00000040;    // Enable the uart-0 interrupt

    rx_flag = 0x00;
    tx_flag = 0x00;

    while(1)
    {
        while(rx_flag == 0x00);   //wait for receive flag to set
        rx_flag = 0x00;           //clear the flag
        while(tx_flag == 0x00);   //wait for transmit flag to set
        tx_flag = 0x00;           //clear the flag
    }
}

// Do this forever
void uart_interrupt(void)__irq
{
    temp = U0IIR;
    temp = temp & 0x06;

    if(temp == 0x02)
    {
```

```
        tx_flag = 0xff;
        VICVectAddr=0;
    }
    else if(temp == 0x04)
    {
        U0THR = U0RBR;
        rx_flag = 0xff;
        VICVectAddr=0;
    }
}
```

Program No.8:-Interface a Stepper motor and rotate it in clockwise and anti-clockwise direction

AIM: To write a program to Interface a Stepper motor and rotate it in clockwise and anti-clockwise direction using an ARM7TDMI/LPC2148 evaluation board using evaluation version of Embedded 'C' & Keil Uvision-4 tool/compiler.

```
#include <LPC21xx.H>

void clock_wise(void);
void anti_clock_wise(void);

unsigned long int var1,var2;
unsigned int i=0,j=0,k=0;

int main(void)
{
    PINSEL0 = 0x00FFFFFF;           //P0.12 to P0.15 GPIO
    IOODIR |= 0x0000F000;           //P0.12 to P0.15 output

    while(1)
    {
        for(j=0;j<50;j++)           // 20 times in Clock wise Rotation
            clock_wise();

        for(k=0;k<65000;k++);        // Delay to show anti_clock Rotation

        for(j=0;j<50;j++)           // 20 times in Anti Clock wise Rotation
            anti_clock_wise();

        for(k=0;k<65000;k++);        // Delay to show clock Rotation

    }                                // End of while(1)
}                                    // End of main

void clock_wise(void)
{
    var1 = 0x00000800;               //For Clockwise
    for(i=0;i<=3;i++)                // for A B C D Stepping
    {
        var1 = var1<<1;              //For Clockwise
        var2 = ~var1;
        var2 = var2 & 0x0000F000;

        IO0PIN = ~var2;

        for(k=0;k<3000;k++);         //for step speed variation
    }
}

void anti_clock_wise(void)
{
    var1 = 0x00010000;               //For Anticlockwise
```

```
for(i=0;i<=3;i++)           // for A B C D Stepping
{
    var1 = var1>>1;          //For Anticlockwise
    var2 = ~var1;
    var2 = var2 & 0x0000F000;

    IO0PIN = ~var2;
    for(k=0;k<3000;k++);      //for step speed variation
}
}
```

Program No.9:-Interface a 4x4 keyboard and display the key code on an LCD

AIM: To write a program to a 4x4 keyboard and display the key code on an LCD

```
#include<lpc21xx.h>
#include<stdio.h>

void lcd_init(void);
void clr_disp(void);
void lcd_com(void);
void lcd_data(void);
void wr_cn(void);
void wr_dn(void);
void scan(void);
void get_key(void);
void display(void);
void delay(unsigned int);
void init_port(void);

unsigned long int scan_code[16]= {0x00EE0000,0x00ED0000,0x00EB0000,0x00E70000,
                                0x00DE0000,0x00DD0000,0x00DB0000,0x00D70000,
                                0x00BE0000,0x00BD0000,0x00BB0000,0x00B70000,
                                0x007E0000,0x007D0000,0x007B0000,0x00770000};

unsigned char ASCII_CODE[16]= {'0','1','2','3',
                              '4','5','6','7',
                              '8','9','A','B',
                              'C','D','E','F'};

unsigned char row,col;

unsigned char temp,flag,i,result,temp1;
unsigned int r,r1;
unsigned long int var,var1,var2,res1,temp2,temp3,temp4;
unsigned char *ptr,disp[] = "4X4 KEYPAD";
unsigned char disp0[] = "KEYPAD TESTING";
unsigned char disp1[] = "KEY = ";
int main()
{
    init_port();           // __ARMLIB_enableIRQ();
    delay(3200);           //port intialisation
    lcd_init();            //delay
    delay(3200);           //lcd intialisation
    clr_disp();            //delay
    delay(500);            //clear display
                           //delay

                           //.....LCD DISPLAY TEST.....//
```

```
ptr = disp;
temp1 = 0x81; // Display starting address
lcd_com();
delay(800);

while(*ptr!="0')
{
    temp1 = *ptr;
    lcd_data();
    ptr ++;
}

//.....KEYPAD Working .....//

while(1)
{
    get_key();
    display();
}

//end of main()

void get_key(void) //get the key from the keyboard
{
    unsigned int i;
    flag = 0x00;
    IO1PIN=0x000f0000;
    while(1)
    {
        for(row=0X00;row<0X04;row++) //Writing one for col's
        {
            if( row == 0X00)
            {
                temp3=0x00700000;
            }
            else if(row == 0X01)
            {
                temp3=0x00B00000;
            }
            else if(row == 0X02)
            {
                temp3=0x00D00000;
            }
            else if(row == 0X03)
            {
                temp3=0x00E00000;
            }
            var1 = temp3;
            IO1PIN = var1; // each time var1 value is put to port1
            IO1CLR =~var1; // Once again Conforming (clearing all other bits)
```

```
        scan();
        delay(100);                //delay
        if(flag == 0xff)
            break;
    }                               // end of for

        if(flag == 0xff)
            break;

    }                               // end of while

for(i=0;i<16;i++)
{
    if(scan_code[i] == res1)        //equate the scan_code with res1
    {
        result = ASCII_CODE[i];    //same position value of ascii code
        break;                     //is assigned to result
    }
}
}                                   // end of get_key();

void scan(void)
{
    unsigned long int t;
    temp2 = IO1PIN;                 // status of port1
    temp2 = temp2 & 0x000F0000;      // Verifying column key
    if(temp2 != 0x000F0000)          // Check for Key Press or Not
    {
        delay(1000);               //delay(100)//give debounce delay check again
        temp2 = IO1PIN;
        temp2 = temp2 & 0x000F0000; //changed condition is same

        if(temp2 != 0x000F0000)     // store the value in res1
        {
            flag = 0xff;
            res1 = temp2;
            t = (temp2 & 0x00F00000); //Verfying Row Write
            res1 = res1 | t;          //final scan value is stored in res1
        }
        else
        {
            flag = 0x00;
        }
    }
} // end of scan()

void display(void)
{
    ptr = disp0;
    temp1 = 0x80;                   // Display starting address of first line
    lcd_com();
```



```
        while(*ptr!='\0')
        {
            temp1 = *ptr;
            lcd_data();
            ptr ++;
        }

ptr = disp1;
temp1 = 0xC0;                                // Display starting address of second line
lcd_com();

        while(*ptr!='\0')
        {
            temp1 = *ptr;
            lcd_data();
            ptr ++;
        }
temp1 = 0xC6;                                //display address for key value
        lcd_com();
temp1 = result;
        lcd_data();
    }

void lcd_init (void)
{
    temp = 0x30;
    wr_cn();
    delay(3200);

    temp = 0x30;
    wr_cn();
    delay(3200);

    temp = 0x30;
    wr_cn();
    delay(3200);

    temp = 0x20;
    wr_cn();
    delay(3200);

    // load command for lcd function setting with lcd in 4 bit mode,
    // 2 line and 5x7 matrix display

    temp = 0x28;
    lcd_com();
    delay(3200);
```

```
// load a command for display on, cursor on and blinking off
```

```
temp1 = 0x0C;  
lcd_com();  
delay(800);
```

```
// command for cursor increment after data dump
```

```
temp1 = 0x06;  
lcd_com();  
delay(800);
```

```
temp1 = 0x80;  
lcd_com();  
delay(800);
```

```
}
```

```
void lcd_data(void)
```

```
{  
    temp = temp1 & 0xf0;  
    wr_dn();  
    temp = temp1 & 0x0f;  
    temp = temp << 4;  
    wr_dn();  
    delay(100);  
}
```

```
void wr_dn(void)
```

```
////write data reg
```

```
{  
    IOOCLR = 0x000000FC;           // clear the port lines.  
    IOOSET = temp;                 // Assign the value to the PORT lines  
    IOOSET = 0x00000004;           // set bit RS = 1  
    IOOSET = 0x00000008;           // E=1  
    delay(10);  
    IOOCLR = 0x00000008;  
}
```

```
void lcd_com(void)
```

```
{  
    temp = temp1 & 0xf0;  
    wr_cn();  
    temp = temp1 & 0x0f;  
    temp = temp << 4;  
    wr_cn();  
    delay(500);  
}
```

```
void wr_cn(void)
```

```
//write command reg
```

```
{  
    IOOCLR = 0x000000FC;           // clear the port lines.
```

```
        IO0SET      = temp;                // Assign the value to the PORT lines
        IO0CLR = 0x00000004;              // clear bit RS = 0
        IO0SET      = 0x00000008;          // E=1
        delay(10);
        IO0CLR = 0x00000008;
    }

void clr_disp(void)
{
    // command to clear lcd display
    temp1 = 0x01;
    lcd_com();
    delay(500);
}

void delay(unsigned int r1)
{
    for(r=0;r<r1;r++);
}

void init_port()
{
    IO0DIR = 0x000000FC;                  //configure o/p lines for lcd
    IO1DIR = 0xFFFF0FFF;
}
```

Program No.10:-Display the Hex digits 0 to F on a 7-segment LED interface, with an appropriate delay in between

AIM: To write a program to Demonstrate the use of an external interrupt to toggle an LED On/Off using an ARM7TDMI/LPC2148 evaluation board using evaluation version of Embedded 'C' & Keil Uvision-4 tool/compiler.

// ***** changes made by JK On 090115

/////////////////////////////////DISPLAY ARE CONNECTED IN COMMON CATHODE MODE/////////////////////////////////

Port0 Connected to data lines of all 7 segment displays

```

a
----
f| g|b
|.....|
e| |c
---- . dot
d
a = P0.16
b = P0.17
c = P0.18
d = P0.19
e = P0.20
f = P0.21
g = P0.22
dot = P0.23

```

Select lines for four 7 Segments

```

DIS1 P0.28
DIS2 P0.29
DIS3 P0.30
DIS4 P0.31

```

Values Corresponding to Alphabets 1, 2, 3 and 4

*/

```
#include <LPC21XX.h>
```

```

unsigned int delay;
unsigned int Switchcount=0;
unsigned int Disp[16]={0x003F0000, 0x00060000, 0x005B0000, 0x004F0000,
0x00660000,0x006D0000, 0x007D0000, 0x00070000, 0x007F0000, 0x006F0000,
0x00770000,0x007C0000,0x00390000, 0x005E0000, 0x00790000, 0x00710000 };

```

```

#define SELDISP1 0x10000000 //P0.28
#define SELDISP2 0x20000000 //P0.29
#define SELDISP3 0x40000000 //P0.30
#define SELDISP4 0x80000000 //P0.31
#define ALLDISP 0xF0000000 //Select all display
#define DATAPORT 0x00FF0000 //P0.16 to P0.23 Data lines connected to drive Seven
Segments

```

```
int main (void)
{
    PINSEL0 = 0x00000000;
    PINSEL1 = 0x00000000;
    IO0DIR = 0xF0FF0000;
    IO1DIR = 0x00000000;

    while(1)
    {
        IO0SET |= ALLDISP;           // select all digits
        IO0CLR = 0x00FF0000;        // clear the data lines to 7-segment displays
        IO0SET = Disp[Switchcount]; // get the 7-segment display value from the
array
        if(!IO1PIN & 0x00800000)    // if the key is pressed
        {
            for(delay=0;delay<100000;delay++) // delay
            {}

            if((IO1PIN & 0x00800000)) // check to see if key has been released
            {
                Switchcount++;
                if(Switchcount == 0x10) // 0 to F has been displayed ? go back to 0
                {
                    Switchcount = 0;
                    IO0CLR = 0xF0FF0000;
                }
            }
        }
    }
}
```