



SAI VIDYA INSTITUTE OF TECHNOLOGY

(Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi and Govt. of Karnataka)
Accredited by NBA, New Delhi (CSE, ISE, ECE, MECH, CIVIL), NAAC-'A' Grade
Rajanukunte, Bengaluru – 560 064

Tel: 080-2846 8196, email: hodcse@saividya.ac.in

web: www.saividya.ac.in



MOTTO

"Learn to lead"

VISION

Contribute dedicated, skilled, intelligent engineers and business administrators to architect strong India and the world.

MISSION

- *To provide quality education and skilled training to produce dedicated engineers and managers.*
- *To promote research, innovation, and ethical practices by creating supportive environment.*
- *To undertake collaborative projects with academia and industry that transform young minds into Socially Responsible Citizen and Globally Competent Professionals.*
- *To enhance personality traits which lead to entrepreneurship qualities among the students.*

MICROCONTROLLERS [BCS402]

(As per Visvesvaraya Technological University Syllabus)

Compiled by:

Pro. H.S.Poornima Gowda
Assistant Professor
Dept. of CSE

Prof. Abhijith D A
Assistant Professor
Dept. of CSE

Name: _____

USN : _____

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
2023-24

Disclaimer

The information contained in this document is the proprietary and exclusive property of Sai Vidya Institute of Technology, Bengaluru except as otherwise indicated. No part of this document, in whole or in part, may be reproduced, stored, transmitted, or used for course material development purposes without the prior written permission of Sai Vidya Institute of Technology, Bengaluru. The information contained in this document is subject to change without notice. The information in this document is provided for informational purposes only.

Trademark



Edition: 2023-24

Document Owners

The primary contacts for questions regarding this document are:

Authors: **1. Pro. H.S.Poornima Gowda**
 2. Prof. Abhijith .D. A.

Department: **Computer Science & Engineering**

Contact email id: **1. poornima.hs@saividya.ac.in**
 2. abhijith.da@saividya.ac.in

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DEPARTMENT VISION

Contribute dedicated, skilled, intelligent Computer Engineers to architect strong India and the world.

DEPARTMENT MISSION

- *To promote activities that imparts high quality technical education in core domains of computer engineering and also interdisciplinary areas.*
- *To imbibe innovative projects and research based skills using software tools to solve real time problems for building modern India and the world.*
- *To create a conducive environment for lifelong learning with ethical responsibilities to produce globally competent Computer professionals and entrepreneurs.*

PROGRAM EDUCATIONAL OBJECTIVES

- *PEO 1: Graduates will have the expertise in analyzing real time problems and providing appropriate solutions related to Computer Science & Engineering.*
- *PEO 2: Graduates will have the knowledge of fundamental principles and innovative technologies to succeed in higher studies, and research.*
- *PEO 3: Graduates will continue to learn and to adapt technology developments combined with deep awareness of ethical responsibilities in profession.*

Program Outcomes

- 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, 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 Specific Outcomes

PSO-1: Demonstrate the knowledge and understanding of working principles, design, implement, test and evaluate the hardware and software components of a computer system.

PSO-2: Apply standard Software Engineering practices and strategies project development.

PSO-3: Demonstrate the knowledge of Discrete Mathematics, Data management and Data Engineering.

MICROCONTROLLER (Effective from the academic year 2023 -2024) SEMESTER – IV			
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		
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 ARM Architectural features and Instructions.● Develop programs using ARM instruction set for an ARM Microcontroller.● Explain C-Compiler Optimizations and portability issues in ARM Microcontroller.● Apply the concepts of Exceptions and Interrupt handling mechanisms in developing applications.● Demonstrate the role of Cache management and Firmware in Microcontrollers.			

LIST OF PROGRAMS

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

Assessment Details (both CIE and SEE)

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

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**.

CIE for the practical component of the IPCC

1. **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.
2. 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.
3. 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.
4. 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.
5. Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IPCC for **25 marks**.
6. The student has to secure 40% of 25 marks to qualify in the CIE of the practical component of the IPCC.

Introduction to ARM7 LPC2148 Microcontroller

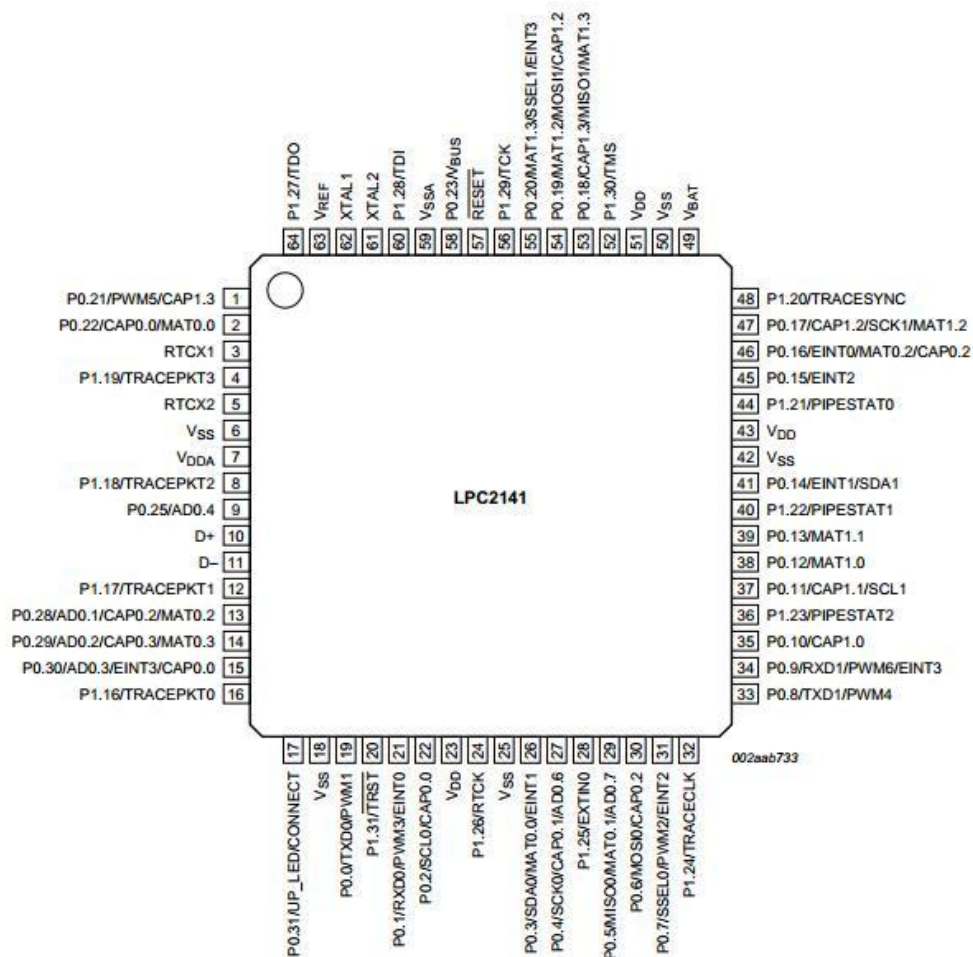
Introduction to ARM7 LPC2148 Microcontroller

ARM-Advanced RISC Machine is a 32-bit RISC (Reduced Instruction Set Computer) processor architecture developed by ARM Holdings.

ARM7 is most successful and widely used processor family in embedded system applications.

LPC2148 is manufactured by NXP Semiconductor (Phillips) and it is preloaded with many in-built features and peripherals. This makes it more efficient and reliable choice for an high-end application developer.

Pin diagram of LPC2148:



It's more important to have basic knowledge of pin configuration, memory, IO ports and basic registers.

MEMORY:

LPC2148 has 32kB on chip SRAM and 512kB on chip FLASH memory. This chip has built in support up to 2kB end point USB RAM. This memory is more than enough for almost all applications.

INPUT/OUTPUT PORTS (GPIO of LPC2148)

LPC2148 has two IO ports each of 32-bit wide, provided by 64 IO pins. Ports are named as P0 and P1. Pins of each port labeled as **Px,y** where “x” stands for port number, 0 or 1. Where “y” stands for pin number usually between 0 to 31. Each pin can perform multiple functions.

REGISTERS FOR C PROGRAMMING

IOSEL0

Port 0 has 32 pins (P0.0 to P0.31). Each pin can have multiple functions. On RESET, all pins are configured as GPIO pins. However we can re-configure using the registers IOSEL0 and IOSEL1.

IOSEL0 is used to select function of **P0.0 to P0.15**. Each pin has up to 4 functions so 2 bits/pin in IOSEL0 is provided for selecting function.

00	Function 0 (Default Function= GPIO)
01	Function 1
10	Function 2
11	Function 3

IOSEL1

IOSEL1 is used to select function of Pins P0.16 to P0.31

IOSEL2

IOSEL2 is used to select function of Pins P1.16 to P1.31

IO0DIR

IO0DIR is used to configure pins of Port 0-P0 as input or output pins.

1= Output Pin

0= Input Pin

Example: IO0DIR=0x0000FFFF means P0.0 to P0.15 are configured as output pins and P0.16 to P0.31 are configured as input pins.

Program ARM7 LPC2148 Microcontroller

To program microcontroller we will be using MDK-ARM Keil μ Vision4 IDE and Flash Magic Tool. Keil has Lite or Evaluation edition which limits the code size of 32kB. This could be more than enough for our projects. Flash Magic is utility, we'll use this to load hex file into flash memory of LPC2148 microcontroller. We also need hardware's i.e. microcontroller evaluation board to run and test Hardware interface code.

How to Install MDK-ARM Version 4.74 Evaluation Software

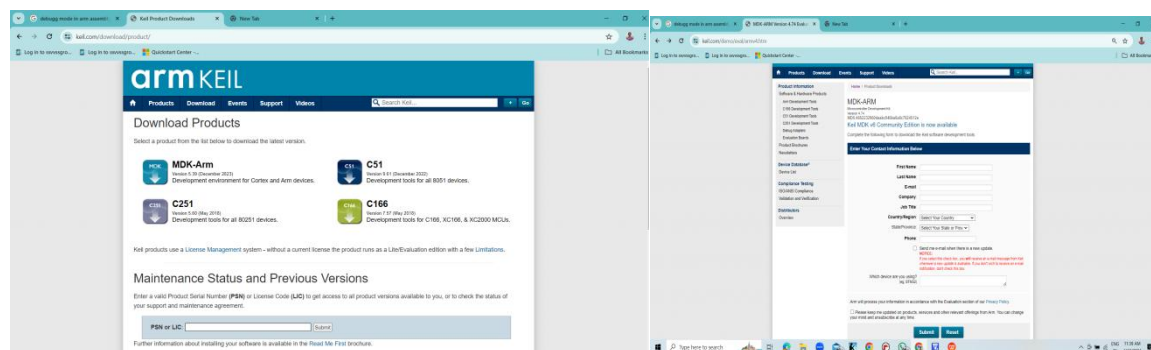
Installation of MDK-ARM Keil μ Vision4 is pretty straight forward. Evaluation version of Keil for ARM is available to download here: <https://www.keil.com/demo/eval/armv4.htm>

This is free software (evaluation version). This software is an Integrated Development Environment(IDE), which integrated text editor to write program, a compiler and it will convert our source code into HEX file.

Here is the step by step guide to install this software.

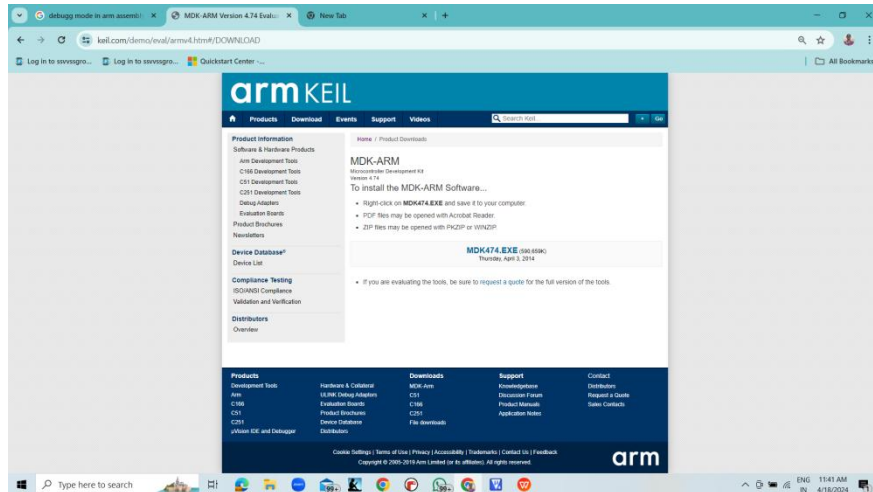
STEP 1

Go to <https://www.keil.com/demo/eval/armv4.htm> **→Download → Product Downloads →Hit on MDK-Arm. Enter your contact information with valid address, phone number and email. Fill in all fields of form. Download is free for evaluation version.**



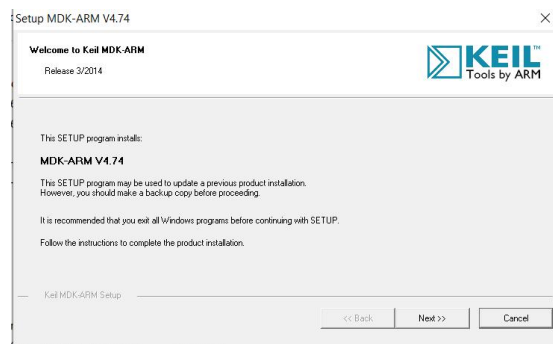
STEP 2

Then click on **MDK474.EXE** and **Download** it on your computer.



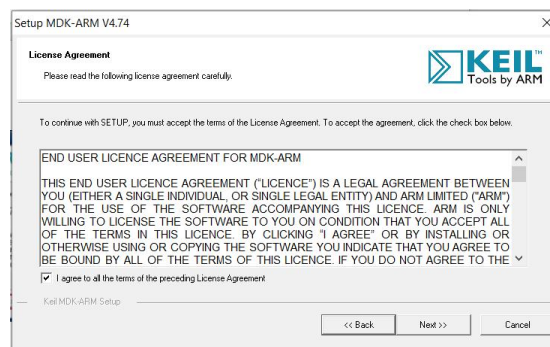
STEP 3

Next step is to run setup file **MDK474.EXE** and then we'll get pop-up box, hit on **Next** and Proceed Installation.



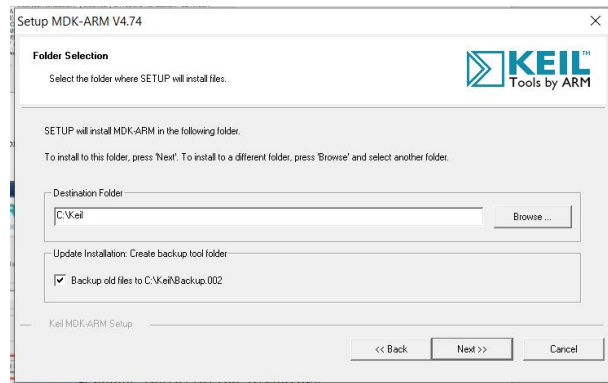
STEP 4

Read license agreement, check **I agree to all the terms....**, and click **Next**.



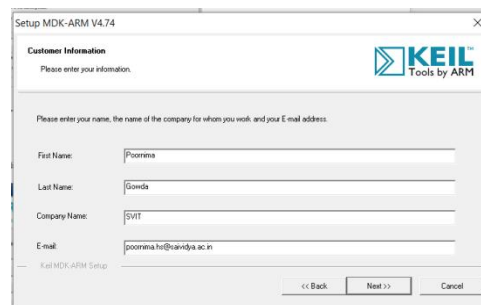
STEP 5

Select Destination folder where you want to install Keil or default destination is already there. And hit on **Next**.



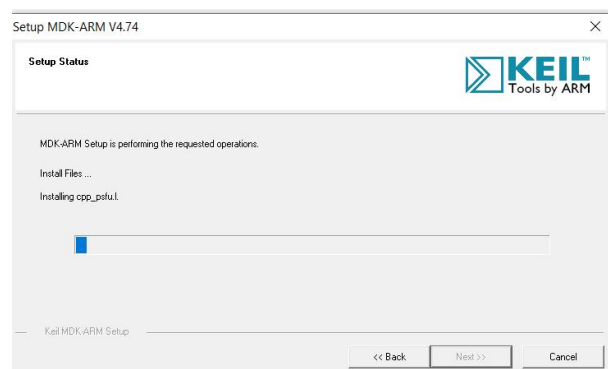
STEP 6

Fill up required fields with all relevant information and click on **Next**.



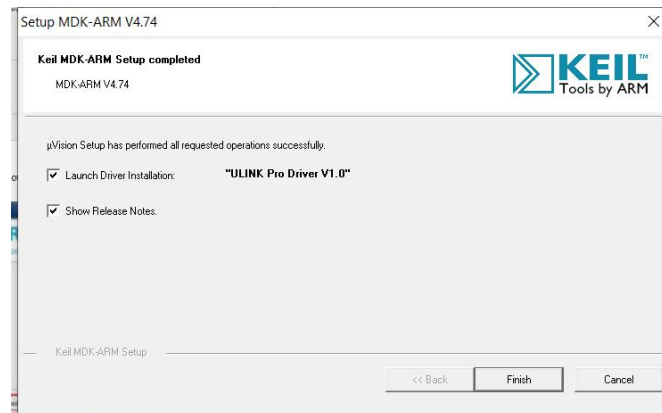
STEP 7

Wait for installation completes and hit on **Next**.



STEP 8

Tick on show release notes, deselect remaining (as per your choice) and click on **Finish**.



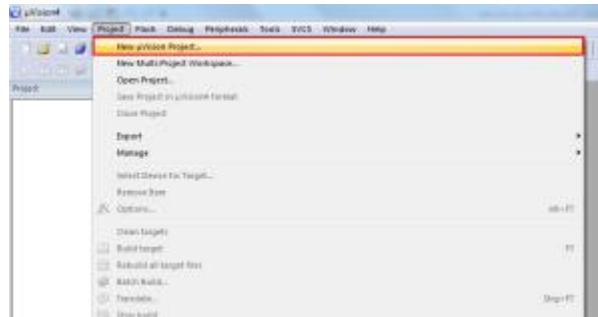
Create New Keil Project for ARM7 LPC2148:

Here is simple guide to start working with Keil μ Vision which can be used for:

- a) Writing programs in C/C++ or Assembly
- b) Compiling and assembling programs
- c) Debugging programs
- d) Creating HEX, AXF and BIN file
- e) Test program without real hardware

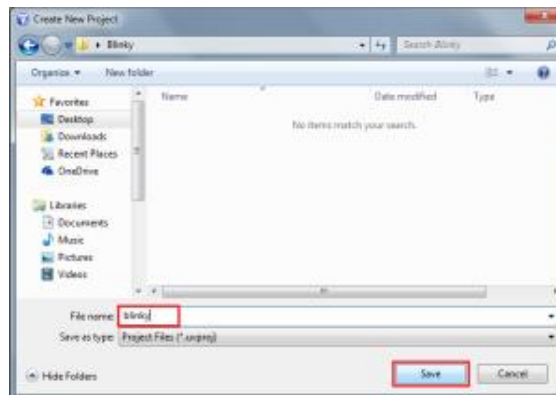
Here is step by step guide to create fresh new project for ARM7 LPC2148 Microcontroller using MDK-ARM μ Vision4.

1. Once we've installed Keil, click on Keil μ Vision4 icon. This will appear on desktop after installation Keil.
2. Click on **Project** menu, and then hit on **New uVision Project**



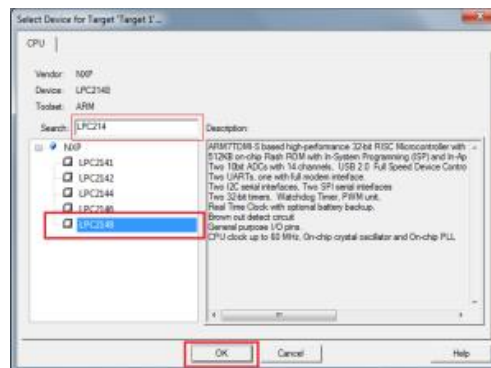
Create New Project Keil LPC2148

3. Create new project folder by typing the project name.



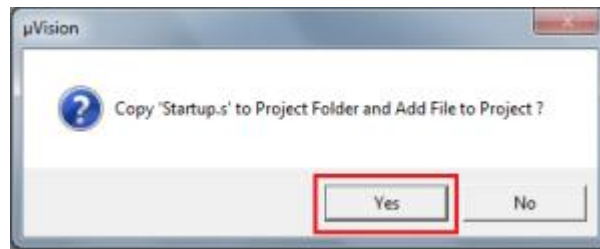
Give Name to New Keil Project

4. Select target Device vendor. In this case NXP (founded by Phillips)



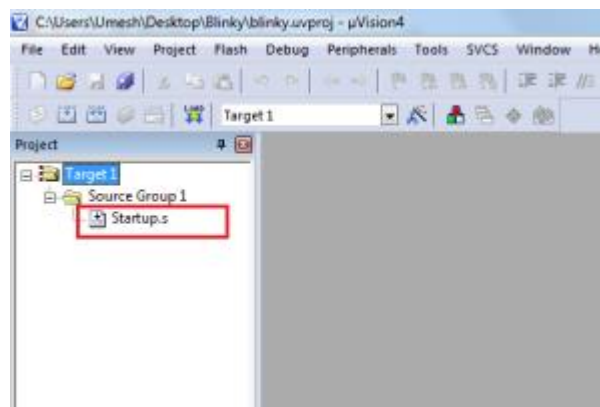
Select Device in Keil

5. Expand NXP icon and select specific chip i.e. LPC2148 After this, a dialog box will pop-up on screen. This will ask you whether to copy startup code for LPC2148. Click on **Yes**.



Add Startup File For LPC2148

6. Then we'll get basic workplace to get start with writing code in. When we expand **Target1** in left project pane. We see **Startup.s** is already added which is necessary for running code in Keil.

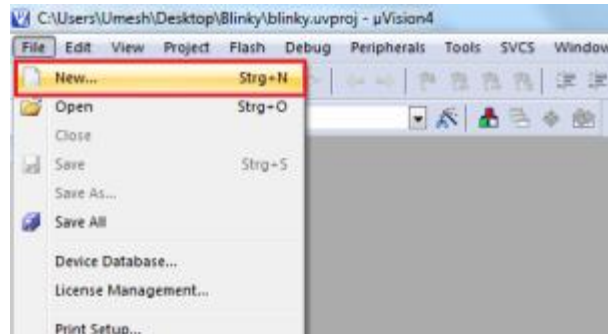


Ready Workplace For LPC2148

NOTE:

Code will not run without startup.s file. Because startup code executes immediately upon reset of the target system and performs the following listed operations.

- a. Defines interrupt and exception vectors
 - b. Configures the CPU clock source (on same device)
 - c. Initializes the external bus controller
 - d. Copies the exception vectors from ROM and RAM for systems with memory remapping
 - e. Initializes other low level peripherals, if necessary
 - f. Reserves and initializes the stack for all modes
 - g. Reserves the heap
 - h. Transfers control to the main C function
7. Now click on **File** menu and hit on **New**.



Create New File Keil

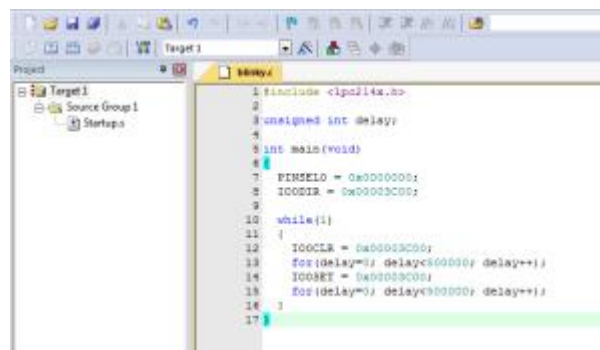
8. Write code in assembly or C language and save file with FileName.s or Filename.c and **Save**.

NOTE: Don't forget to save as with .s or .c extension



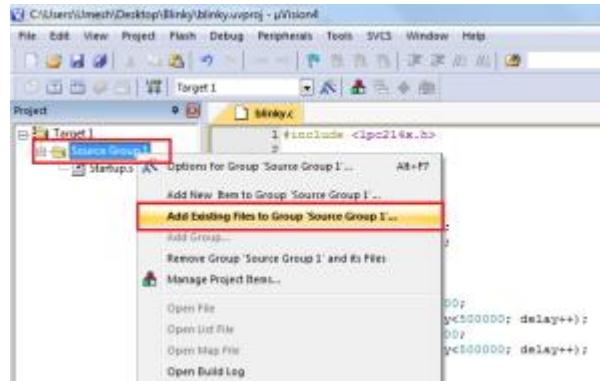
Add File To Keil Project

9. Once we save file We'll quickly get complete code with syntax highlighted.

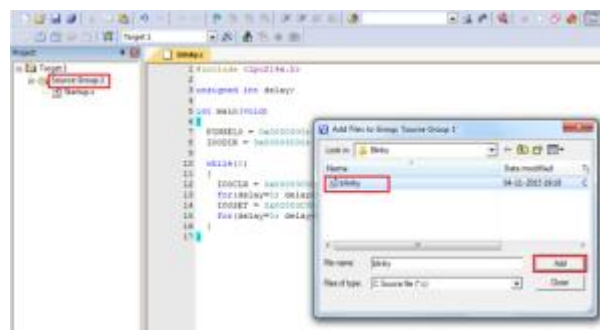


Code Highlighted In Keil LPC2148

10. Now we can add file by adding it to **Group Source 1**.

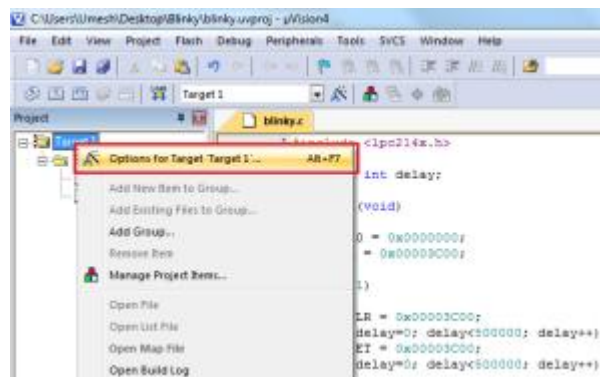


Add Source File To Project LPC2148



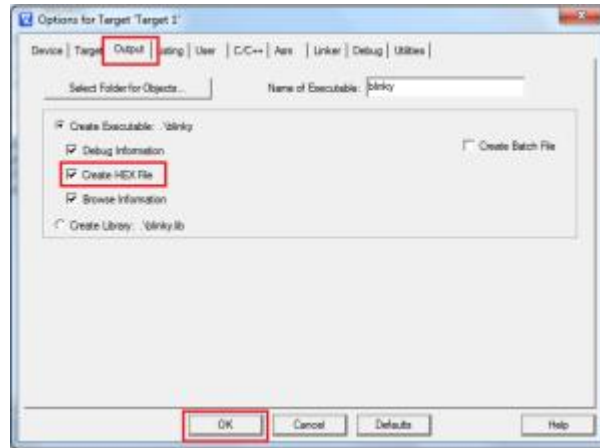
Add File To Keil Project

11. Now right click on **Target 1** and hit on **Options for target “Target 1”**



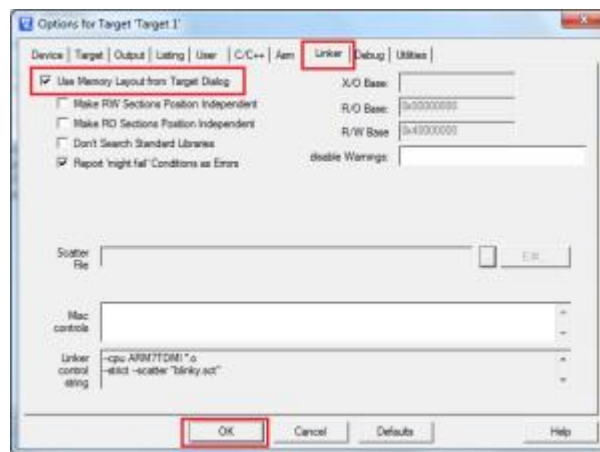
Set Options For Target LPC2148

12. In **Options for target “Target 1”**. In **output** tab, click on check box **Create HEX File**.



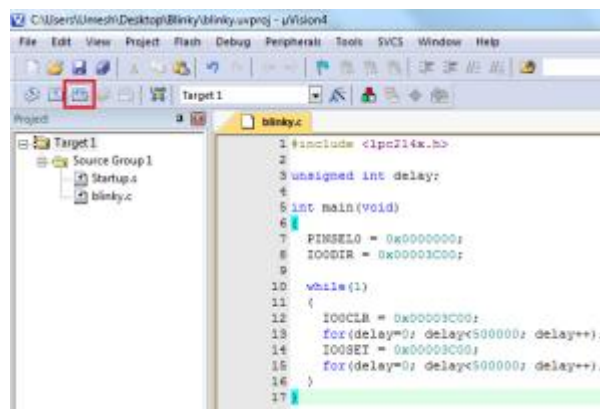
Set Output Keil LPC2148

13. Then go to Linker tab. Click on **Use Memory Layout for Target Dialog**



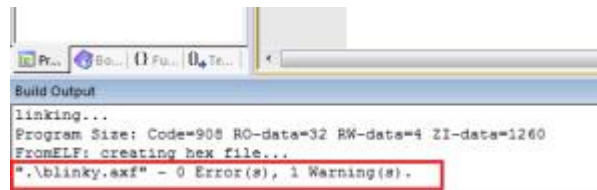
Setup Linker Option Keil LPC2148

14. Then hit on **Rebuild** All target files

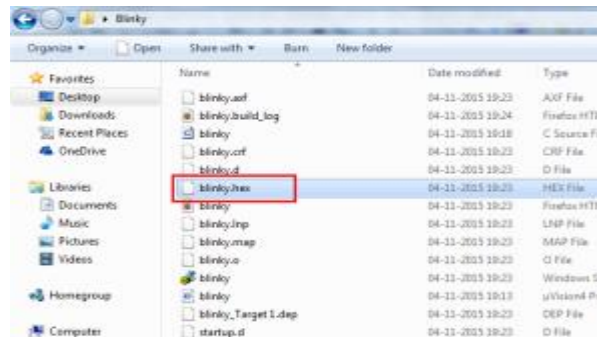


Rebuild The Keil Project LPC2148

15. Now we can see Build output. If we check project directory at this point. We'll find generated HEX file. We'll use it to load or burn into LPC2148 MCU using Flash Magic to execute our program using hardware boards.

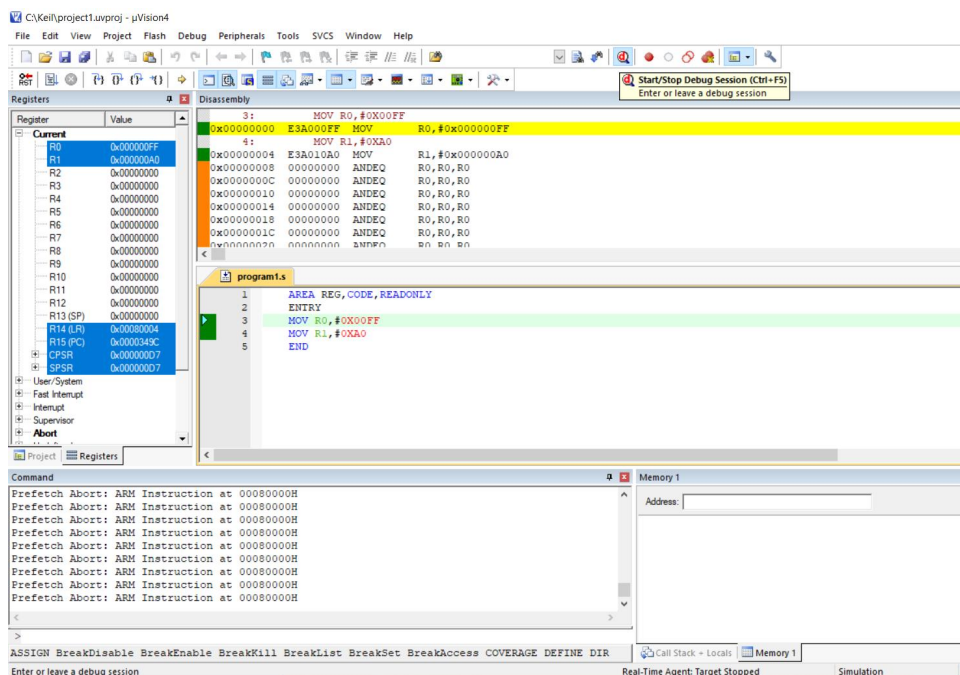


Build Output Keil LPC2148



HEX File Generated Using Keil LPC2148

16. To run the program in keil software,click on debug button to enter into Debug mode.



The debugger provides a number of features that help us to follow the execution of a program. It allows us to examine what effect each instruction has on the processor's registers.

The debugger helps us do the following:

- Single-step through a program, stopping after every instruction.
- Inspect the values in the processor's registers and memory to see how they have changed after

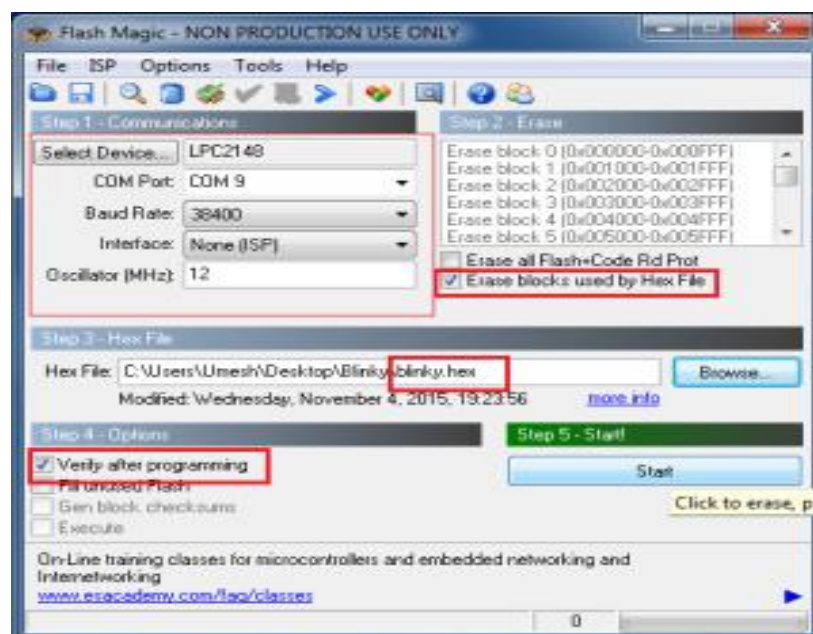
each instruction.

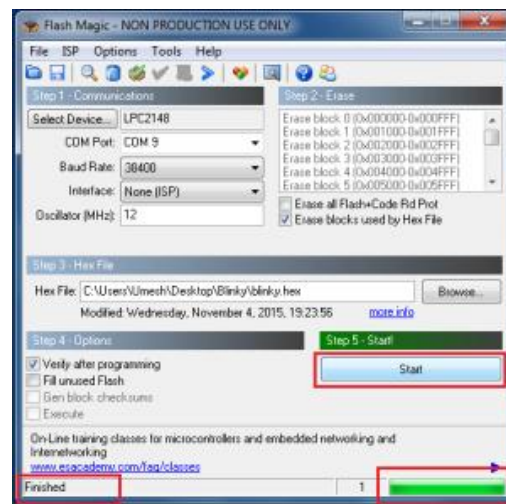
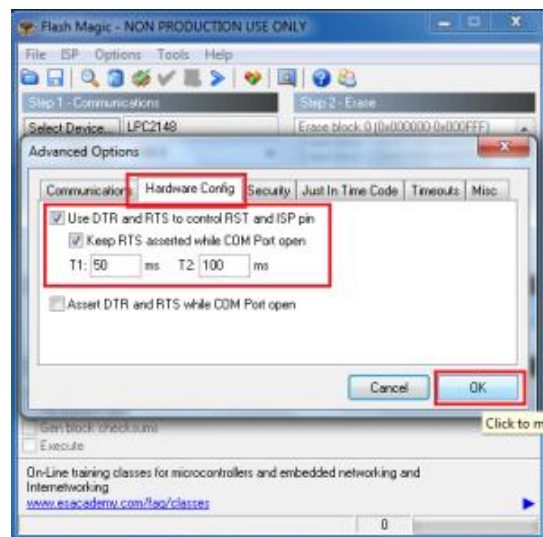
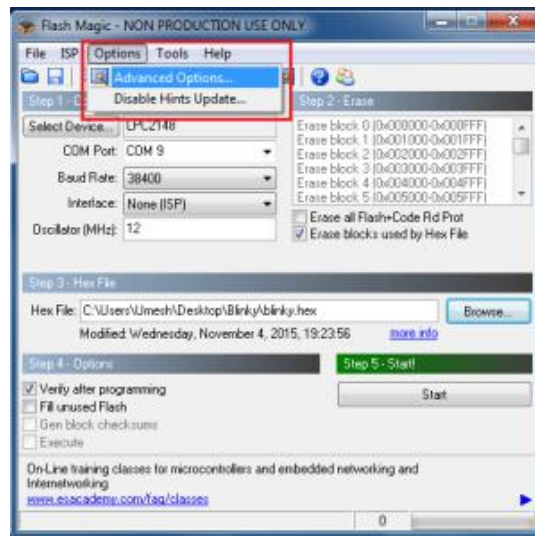
- c) Set breakpoints to run to a certain point and then stop.

17. For Hardware implementation we need to Setup Flash Magic and load HEX file into LPC2148 Microcontroller.

Basic setting in Flash Magic:

- a) Open the flash magic software and follow the below steps.
- b) Select the IC from Select Menu.
- c) Select the COM Port. Check the device manager for detected Com port.
- d) Select Baud rate from 9600-115200
- e) Select None ISP Option.
- f) Oscillator Freq 12.000000(12Mhz).
- g) Check the Erase blocks used by Hex file option
- h) Browse and Select the hex file.
- i) Check the Verify After Programming Option.
- j) If DTR and RTS are used then go to Options->Advanced Options-> Hardware Config and select the Use DTR and RTS Option.
- k) Hit the Start Button to flash the hex file.
- l) Once the hex file is flashed, Reset the board. Now the controller should run your application code.





1.

LAB PROGRAMS

- 1) Using Keil software, observe the various Registers, Dump, CPSR, with a simple Assembly Language Programs (ALP).

AREA register, CODE, READONLY

ENTRY

MOV R0,#0x000056 ; Move immediate number to R0

MOV R1,#0x000089 ; Move immediate number to R1

```
LDR R5,=DATA ; load address of number into R5
```

LDRH R3, [R5] ; load First word from DATA address to R3

```
LDRH R4, [R5,#2] ; load second word from DATA address to R4
```

```
MRS R6, cpsr ; move the data from status register to R6
```

STOP B STOP

DATA DCW 0X1234,0x6789 ; Declaration of no's in the memory

END

OUTPUT

Register	Value
R0	0x00000056
R1	0x00000089
R2	0x00000000
R3	0x00001234
R4	0x00006789
R5	0x0000001C
R6	0x000000D3
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000018
CPSR	0x000000D3
N	0
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13

[illegible]

2. Develop and simulate ARM ALP for Data Transfer, Arithmetic and Logical operations (Demonstrate with the help of a suitable program).

(a) ALP for Data Transfer operations

AREA register, CODE, READONLY

ENTRY

MOV R0,#0X0034 ;Move immediate number to R0

MOV R1,R0 ;Move R1<--R0

MOV R2,R1,LSL#2 ;R2 = 4*R1

MOVS R3, R1, LSR #1 ;R2 = 2*R1

LDR R4,=DATA ; load address of number into R4

LDR R5,[R4] ;load First word from DATA address to R5

LDRH R6, [R4,#2] ;load second word from DATA address to R6

LDRH R7, [R4,#4] ;load third word from DATA address to R7

MRS R8,cpsr ;Move status register content to R8

STOP B STOP

DATA DCW 0X1234,0x6789,0xF7A4 ; Declaration of no's in the memory

END

Output:

The screenshot displays the ARM simulator interface. On the left, the 'Memory 1' window shows a memory dump starting at address 0x00000000. The data at address 0x0000001A is highlighted in red: 34 12 89 67 A4 F7 00 00 28 00 00 00. On the right, the 'Registers' window shows the current state of the processor registers. R0 is 0x00000034, R1 is 0x00000034, R2 is 0x000000D0, R3 is 0x0000001A, R4 is 0x00000028, R5 is 0x67891234, R6 is 0x00006789, R7 is 0x0000F7A4, R8 is 0x000000D3, R9 is 0x00000000, R10 is 0x00000000, R11 is 0x00000000, R12 is 0x00000000, R13 (SP) is 0x00000000, R14 (LR) is 0x00000000, R15 (PC) is 0x00000024, CPSR is 0x000000D3, and SPSR is 0x00000000. The CPSR bits are also shown: N=0, Z=0, C=0, V=0, I=1, F=1, T=0, M=0x13.

Register	Value
R0	0x00000034
R1	0x00000034
R2	0x000000D0
R3	0x0000001A
R4	0x00000028
R5	0x67891234
R6	0x00006789
R7	0x0000F7A4
R8	0x000000D3
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000024
CPSR	0x000000D3
SPSR	0x00000000

Bit	Value
N	0
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13

(b) **ALP for Arithmetic operations**

```
AREA register, CODE, READONLY
ENTRY
MOV R1, #0x00F0
ADD R0 , R1 , #0xFFFFFFFF4F ; R0 = R1 + 0xFFFFFFFF4F
ADDS R2, R1 , #0xFFFFFFFF4F ; R2 = R1 + 0xFFFFFFFF4F and flags updated
ADCS R3 , R1 , R2 ;R3=R1+R2+CARRY and flags updated
ADD R4 , R1 , R1 , LSL #1 ; R4 = 3*R1
MOV R5 , #0x00F0
SUB R6 , R5 , #0x0F0 ; R6 = R5 - 0x0F0
SUBS R7, R5 , #0x0F0 ; R7 = R5 + 0x0F0 and flags updated
SBCS R8 ,R5 , R6 ;R8=R5-R6-CARRY and flags updated
SUBS R9, R1, R1, LSL #1 ; R9 = R1-2*R1, RESULT ISNEGATIVE ,set Nflag
STOP B STOP
END
```

Output:

Register	Value
Current	
R0	0x0000003F
R1	0x000000F0
R2	0x0000003F
R3	0x00000130
R4	0x000002D0
R5	0x000000F0
R6	0x00000000
R7	0x00000000
R8	0x000000F0
R9	0xFFFFFFFF10
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000028
CPSR	0x800000D3
N	1
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13
SPSR	0x00000000
User/System	

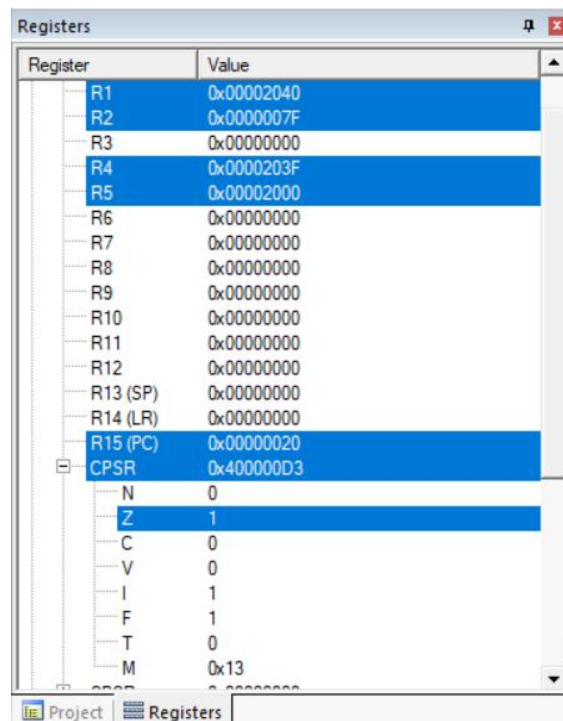
(C) ALP for Logical operations

```
AREA LOGICAL, CODE, READONLY
ENTRY

MOV R0, #0X0000      ;R0=0X0000
MOV R1, #0x2040       ;R1=0X2040
MOV R2, #0x007F       ;R2=0X007F
ORR R0, R1, R2         ;R0 = R1 OR R2
AND R3, R1, #0X0F      ;R3 = R1 AND 0F
    ANDS R3, R1, #0X0F ;R3 = R1 AND 0F , UPDATES Z-FLAG
EOR R4, R1, R2         ;R4 = R1 XOR R2
BIC R5, R1, R2         ;R5 = R1 AND NOT(R2)

STOP B STOP
END
```

OUTPUT:



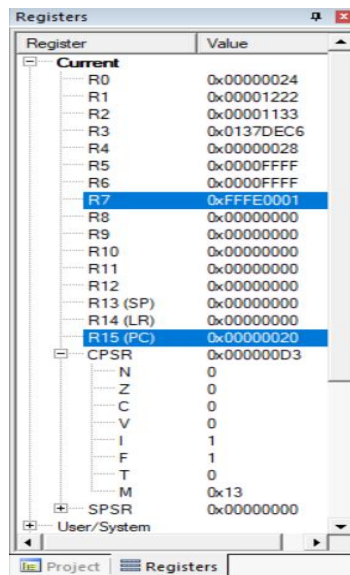
Register	Value
R1	0x00002040
R2	0x0000007F
R3	0x00000000
R4	0x0000203F
R5	0x00002000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000020
CPSR	0x400000D3
N	0
Z	1
C	0
V	0
I	1
F	1
T	0
M	0x13

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

```
AREA Multiply, CODE, READONLY
ENTRY
LDR R0, =NUM1      ; load address of multiplicand
LDRH R1, [R0]       ; load First number
LDRH R2, [R0,#2]    ; load Second number
MUL R3, R1, R2      ; R3 = R1 x R2
LDR R4, =NUM2      ; load address of multiplicand
LDRH R5, [R4]       ; load First number
LDRH R6, [R4,#4]    ; load Second number
SMULLS R7, R8, R5, R6 ; R8,R7 = R5 x R6

STOP B STOP
NUM1 DCW 0X1222,0X1133 ; Declaration of no's to be multiply
NUM2 DCW 0XFFFF,0XFFFF
END
```

Output:



Register	Value
R0	0x00000024
R1	0x00001222
R2	0x00001133
R3	0x0137DEC6
R4	0x00000028
R5	0x0000FFFF
R6	0x0000FFFF
R7	0xFFFE0001
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000020
CPSR	0x000000D3
N	0
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13
SPSR	0x00000000

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

AREA ADD1TO10, CODE, READONLY

ENTRY

```
MOV R1,#10           ;length of array
```

```
LDR R2,=ARRAY           ;Load the starting address of the array
```

MOV R4,#0 ;Initial sum

NEXT LDR R3,[R2],#4 ;Load first integer of the array in R3

ADD R4,R4,R3 ;R4=sum of integers

```
SUBS R1,R1,#1           ;repeat until R1=0
```

```
BNE NEXT ;If z-flag is not set repeat
```

```
MOV R5,#0X40000000    ; initialize memory address to store the result in memory
```

STR R4,[R5] ; store the result in the address stored in R5

STOP B STOP

ARRAY DCD 1,2,3,4,5,6,7,8,9,10

END

Output:

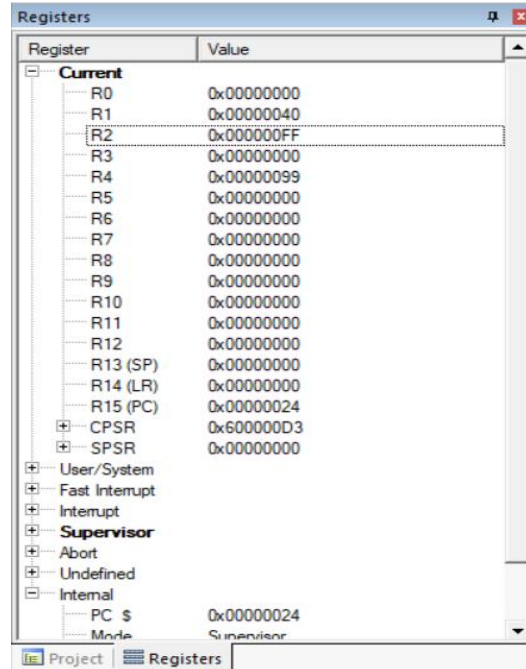
Register	Value
Current	
R0	0x00000000
R1	0x00000000
R2	0x00000050
R3	0x0000000A
R4	0x00000037
R5	0x40000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x60000000
R14 (LR)	0x00000000
R15 (PC)	0x00000024
+ CPSR	0x600000D3
+ SPSR	0x00000000
+ User/System	
+ Fast Interrupt	

[illegible]

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

```
AREA LARGE, CODE, READONLY
ENTRY
MOV R5, #5                ;R5 = length of array - 1
LDR R1, =ARRAY            ;load starting addressing of array
LDR R2, [R1], #4          ;load 1st element of array
LOOP LDR R4, [R1], #4      ;load next element of array
    CMP R2, R4             ;compare 1st and 2nd element
    BHI NEXT              ;R2=largest value
    MOV R2, R4
NEXT SUBS R5, R5, #1       ;decrement the counter after every comparison
    BNE LOOP              ; repeat until R5=0
STOP B STOP
ARRAY DCD 0X23, 0X45, 0XFF, 0X76, 0X12, 0X99
END
```

OUTPUT:



Register	Value
Current	
R0	0x00000000
R1	0x00000040
R2	0x000000FF
R3	0x00000000
R4	0x00000099
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000024
CPSR	0x600000D3
SPSR	0x00000000
Supervisor	
Abort	
Undefined	
Internal	
PC \$	0x00000024
Mode	Supervisor

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

MOV R7, #2	;Counter of 2 numbers
LDR R6 , =LOOKUP	;Load starting address of numbers
LOOP MOV R1 , #32	;Number of bits in each number
LDR R0 , [R6]	;Load 1st number to r0
NEXTBIT MOVS R0 , R0 , ROR #1	;Check the bit is one or zero
BHI ONES	; IF CF=1 increment r2 else increment r3
ZEROS ADD R3 , R3 , #1	;R3 stores count of 0s
B REPEAT	
ONES ADD R2 , R2 , #1	;R2 stores count of 1s
REPEAT SUBS R1, R1 , #1	; Decrement the bit counter by 1 till 0
BNE NEXTBIT	; Repeat until r1=0
ADD R6, R6, #4	; Load r6=address of next number
SUBS R7, R7 , #1	; Decrement the number counter by 1 till 0
BNE LOOP	
STOP B STOP	
LOOKUP DCD 0X5,0X7	; Memory address of lookup table
END	

OUTPUT:

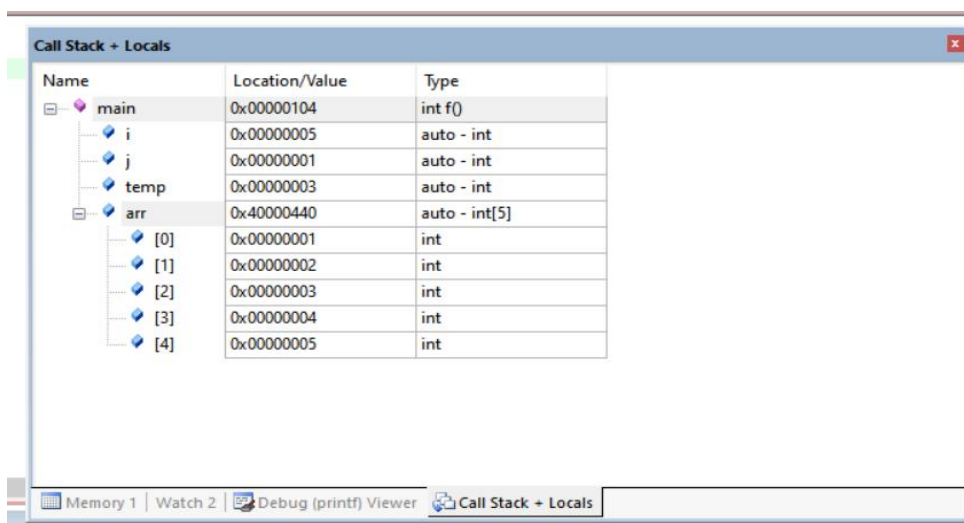
Register	Value
Current	
R0	0x00000007
R1	0x00000000
R2	0x00000005
R3	0x00000038
R4	0x00000000
R5	0x00000000
R6	0x0000004C
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000040
CPSR	0x600000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	

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(void)
{
    volatile int i, j, temp;
    int arr[ ] = {4,1,3,5,2};           //the array to be sort
    //bubble sort
    for (i = 1; i < 5; i++)
    {
        for (j = 0; j < 5 - i; j++)
        {
            if (arr[j] > arr[j + 1])    //Compares first and second element of the array
            {
                temp = arr[j];          // If first element is greater, swap the elements of the array
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    while(1);
}
```

Output:



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

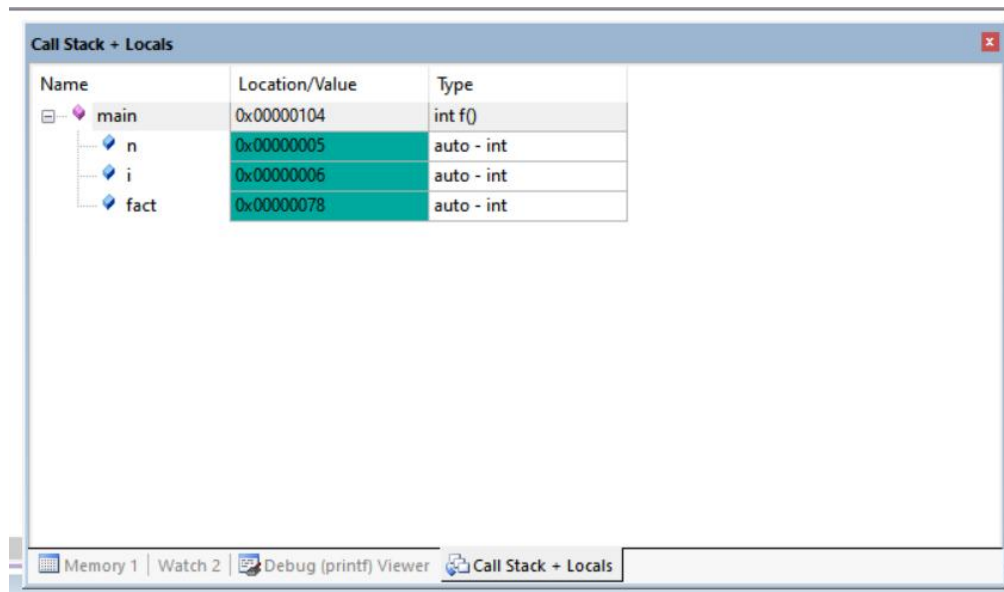
```
#include <lpc214x.h>

int main()
{
    volatile int n, i, fact = 1;
    n = 5;                                // initialize the number

    for (i = 1; i <= n; ++i)
        fact *= i;                        // factorial of the number

    while(1);
}
```

OUTPUT:



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>

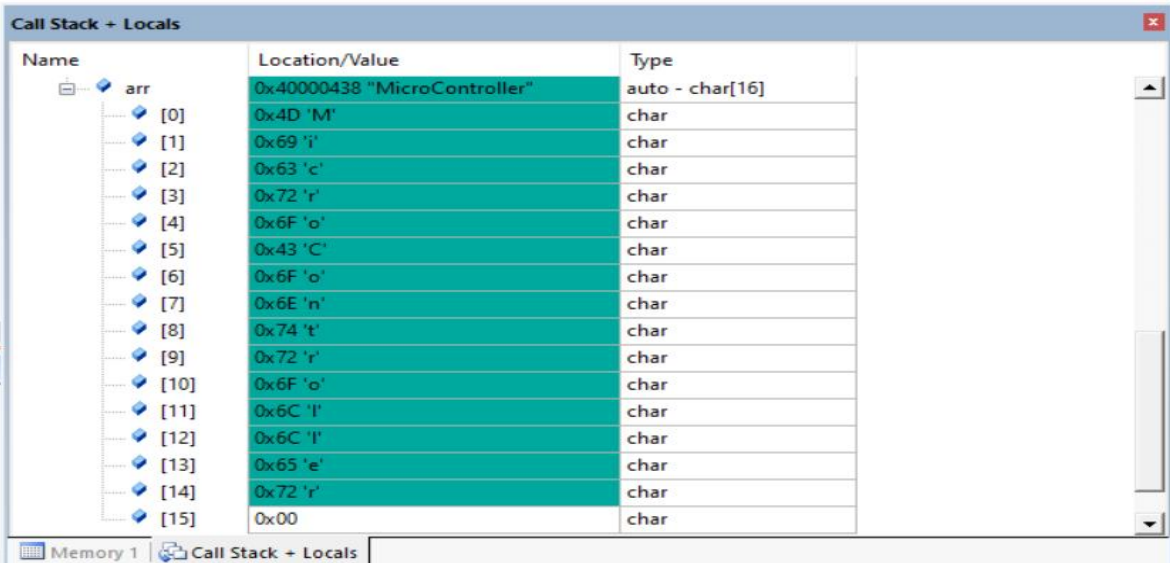
int main(void)
{
    volatile int i;

    char tran_arr[20];                // declare the variable to store translated string

    char arr[ ]="MicroController" ;    // Initialize Input String
    for(i=0; arr[i]!=0;i++)
    {
        if (arr[i] >= 'a' && arr[i] <= 'z')
        {
            tran_arr[i] = arr[i] - 32 ;           // transalte lower to Upper case
        }
        else if(arr[i] >= 'A' && arr[i] <= 'Z')
        {
            tran_arr[i] = arr[i]+ 32;           // translate Upper to Lower case
        }
    }
    while(1);
}
```

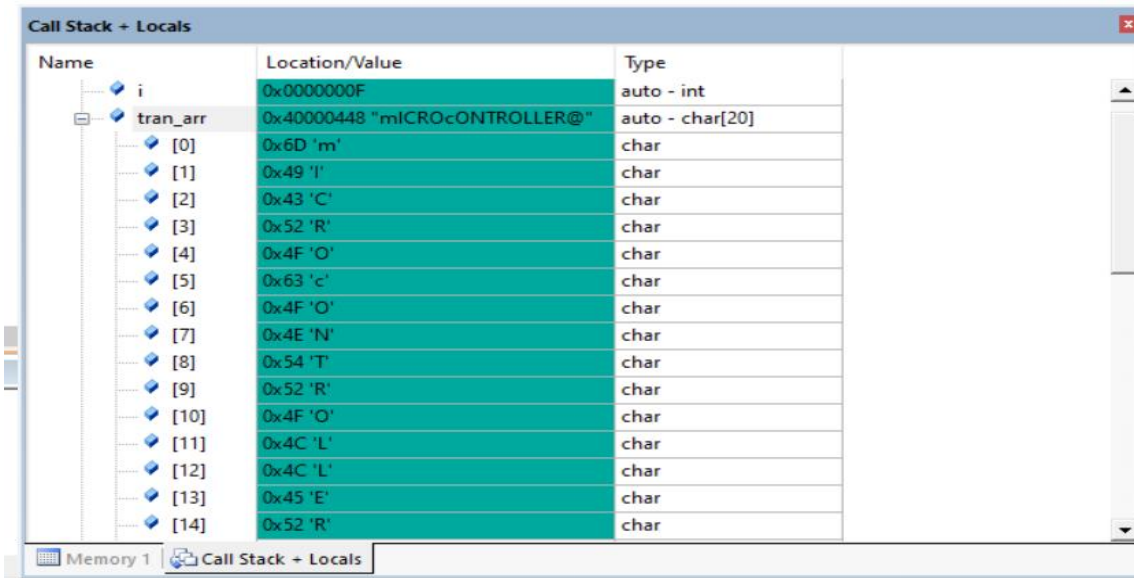
OUTPUT:

Input sting:



Name	Location/Value	Type
arr	0x40000438 "MicroController"	auto - char[16]
[0]	0x4D 'M'	char
[1]	0x69 'i'	char
[2]	0x63 'c'	char
[3]	0x72 'r'	char
[4]	0x6F 'o'	char
[5]	0x43 'C'	char
[6]	0x6F 'o'	char
[7]	0x6E 'n'	char
[8]	0x74 't'	char
[9]	0x72 'r'	char
[10]	0x6F 'o'	char
[11]	0x6C 'l'	char
[12]	0x6C 'l'	char
[13]	0x65 'e'	char
[14]	0x72 'r'	char
[15]	0x00	char

Translated String:



Name	Location/Value	Type
i	0x0000000F	auto - int
tran_arr	0x40000448 "mICROcONTROLLER@"	auto - char[20]
[0]	0x6D 'm'	char
[1]	0x49 'I'	char
[2]	0x43 'C'	char
[3]	0x52 'R'	char
[4]	0x4F 'O'	char
[5]	0x63 'c'	char
[6]	0x4F 'O'	char
[7]	0x4E 'N'	char
[8]	0x54 'T'	char
[9]	0x52 'R'	char
[10]	0x4F 'O'	char
[11]	0x4C 'L'	char
[12]	0x4C 'L'	char
[13]	0x45 'E'	char
[14]	0x52 'R'	char

10 Demonstrate enabling and disabling of Interrupts in ARM.

```
#include <lpc214x.h>

void initClocks(void);
void initTimer0(void);
__irq void timer0ISR(void);
int main(void)
{
    initClocks();                // Initialize PLL to setup clocks
    initTimer0();                // Initialize Timer0
    IO0DIR = (1<<10);           // Configure pin P0.10 as Output
    IO0PIN = (1<<10);

    T0TCR = (1<<0);              // Enable timer

    while(1);                    // Infinite Idle Loop
}

void initTimer0(void)
{
    T0CTCR = 0x0;                //Set Timer Mode
    T0PR = 60000-1;              //Increment T0TC at every 60000 clock cycles
                                //60000 clock cycles @60Mhz = 1 mS

    T0MR0 = 5000-1;              //Zero Indexed Count-hence subtracting 1
    T0MCR = (1<<0) | (1<<1);     //Set bit0 & bit1 to Interrupt & Reset TC on MR0
    VICVectAddr4 = (unsigned )timer0ISR; //Pointer Interrupt Function (ISR)
    VICVectCntl4 = (1<<5) | 4;    //(bit 5 = 1)->to enable Vectored IRQ slot
                                //bit[4:0]) -> this the source number

    VICIntEnable = (1<<4);        // Enable timer0 interrupt
    T0TCR = (1<<1);              // Reset Timer
}

__irq void timer0ISR(void)
{

```

```

long int readVal;
readVal = T0IR;           // Read current IR value
IO0PIN ^= (1<<10);        // Toggle LED at Pin P0.10
T0IR = readVal;           // Write back to IR to clear Interrupt Flag
VICVectAddr = 0x0;        // End of interrupt execution
}

void initClocks(void)
{
    PLL0CON = 0x01;        //Enable PLL
    PLL0CFG = 0x24;        //Multiplier and divider setup
    PLL0FEED = 0xAA;       //Feed sequence
    PLL0FEED = 0x55;

    while(!(PLL0STAT & 0x00000400)); //is locked?

    PLL0CON = 0x03;        //Connect PLL after PLL is locked
    PLL0FEED = 0xAA;       //Feed sequence
    PLL0FEED = 0x55;
    VPBDIV = 0x01;        //PCLK is same as CCLK i.e.60 MHz
}

```

Output:

General Purpose Input/Output 0 (GPIO 0) - Slow Interface

GPIO0

IO0DIR: 0x00000000 31 Bits 24 23 Bits 16 15 Bits 8 7 Bits 0

IO0SET: 0x00000000

IO0CLR: 0x00000000

IO0PIN: 0x82FFFFFF

Pins: 0xF2FFFFFF

Timer 0

Prescaler

PR: 0x00000000

PC: 0x00000000

Timer

TCR: 0x00000000

TC: 0x00000000

☐ Enable

☐ Reset

Interrupt Register

IR: 0x00000000

Match Channels

MCR: 0x00000000

MR0: 0x00000000

MR1: 0x00000000

MR2: 0x00000000

MR3: 0x00000000

☐ Interrupt on MR0

☐ Interrupt on MR1

☐ Interrupt on MR2

☐ Interrupt on MR3

☐ Reset on MR0

☐ Reset on MR1

☐ Reset on MR2

☐ Reset on MR3

☐ Stop on MR0

☐ Stop on MR1

☐ Stop on MR2

☐ Stop on MR3

EMC0: Nothing

EMC1: Nothing

EMC2: Nothing

EMC3: Nothing

☐ External Match 0

☐ External Match 1

☐ External Match 2

☐ External Match 3

☐ MR0 Interrupt

☐ MR1 Interrupt

☐ MR2 Interrupt

☐ MR3 Interrupt

Capture Channels

CCR: 0x00000000

CR0: 0x00000000

CR1: 0x00000000

CR2: 0x00000000

CR3: 0x00000000

☐ Rising Edge 0

☐ Rising Edge 1

☐ Rising Edge 2

☐ Rising Edge 3

☐ Falling Edge 0

☐ Falling Edge 1

☐ Falling Edge 2

☐ Falling Edge 3

☐ Interrupt on Event 0

☐ Interrupt on Event 1

☐ Interrupt on Event 2

☐ Interrupt on Event 3

☐ CAP0.0

☐ CAP0.1

☐ CAP0.2

☐ CAP0.3

☐ CR0 Interrupt

☐ CR1 Interrupt

☐ CR2 Interrupt

☐ CR3 Interrupt

Count Control

CTCR: 0x00000000

Mode: Timer

Counter Input: CAP0.0

General Purpose Input/Output 0 (GPIO 0) - Slow Interface

GPIO0

IO0DIR: 0x00000400 31 Bits 24 23 Bits 16 15 Bits 8 7 Bits 0

IO0SET: 0x00000400

IO0CLR: 0x00000000

IO0PIN: 0x82FFFFFF

Pins: 0xF2FFFFFF

Timer 0

Prescaler

PR: 0x0000EASF

PC: 0x00007A8C

Timer

TCR: 0x00000001

TC: 0x00000FC8

☒ Enable

☐ Reset

Interrupt Register

IR: 0x00000000

Match Channels

MCR: 0x00000003

MR0: 0x00001387

MR1: 0x00000000

MR2: 0x00000000

MR3: 0x00000000

☒ Interrupt on MR0

☐ Interrupt on MR1

☐ Interrupt on MR2

☐ Interrupt on MR3

☒ Reset on MR0

☐ Reset on MR1

☐ Reset on MR2

☐ Reset on MR3

☐ Stop on MR0

☐ Stop on MR1

☐ Stop on MR2

☐ Stop on MR3

EMC0: Nothing

EMC1: Nothing

EMC2: Nothing

EMC3: Nothing

☐ External Match 0

☐ External Match 1

☐ External Match 2

☐ External Match 3

☐ MR0 Interrupt

☐ MR1 Interrupt

☐ MR2 Interrupt

☐ MR3 Interrupt

Capture Channels

CCR: 0x00000000

CR0: 0x00000000

CR1: 0x00000000

CR2: 0x00000000

CR3: 0x00000000

☐ Rising Edge 0

☐ Rising Edge 1

☐ Rising Edge 2

☐ Rising Edge 3

☐ Falling Edge 0

☐ Falling Edge 1

☐ Falling Edge 2

☐ Falling Edge 3

☐ Interrupt on Event 0

☐ Interrupt on Event 1

☐ Interrupt on Event 2

☐ Interrupt on Event 3

☐ CAP0.0

☐ CAP0.1

☐ CAP0.2

☐ CAP0.3

☐ CR0 Interrupt

☐ CR1 Interrupt

☐ CR2 Interrupt

☐ CR3 Interrupt

Count Control

CTCR: 0x00000000

Mode: Timer

Counter Input: CAP0.0

VIVA QUESTIONS

1. LPC 2148 pro development board has _____on chip memory.

- a) 500k
- b) 625k
- c) 512k
- d) 425k

ANS:-C

2. In LPC 2148 we require separate programmer?

- a) True
- b) False

ANS:- B

3. It provides real time debugging with the on chip real monitor software.

- a) True
- b) False

ANS:- A

4. Who is the founder of LPC2148 board?Intel

- a) Atmel
- b) Motorola
- c) Philips

ANS:- D

5. What is the program counter value when the board turns on?

- a) 0x00000
- b) 0xFFFFF
- c) Where the previous program ends
- d) At the location where we write the code

ANS:- C

6. Which IDE is supported by LPC2148 board?

- a) Code Blocks
- b) AVR Studio 4
- c) Keil uVersion 4
- d) Walldorf

ANS:- D

7. _____bit ARM7TDMI controller is present?

- a) 128 bit
- b) 8 bit
- d) 64 bit
- d) 32 bit

ANS:- C

8. USB 2.0 full speed compliant device controller with _____of end point RAM.

- a) 6 kB
- b) 4 kB
- c) 2 kB
- d) 8 kB

ANS:- B

9. What is the processor used by ARM7?

- a) 8-bit CISC
- b) 8-bit RISC
- c) 32-bit CISC
- d) 32-bit RISC

ANS:-D

10. What is the instruction set used by ARM7?

- a) 16-bit instruction set
- b) 32-bit instruction set
- c) 64-bit instruction set
- d) 8-bit instruction set

ANS:-A

11. How many registers are there in ARM7?

- a) 35 register(28 GPR and 7 SPR)
- b) 37 registers(28 GPR and 9 SPR)
- c) 37 registers(31 GPR and 6 SPR)
- d) 35 register(30 GPR and 5 SPR)

ANS:-C

12. ARM7 has a in-built debugging device?

- a) True
- b) False

ANS:-A

13. What is the capability of ARM7 f instruction for second?

- a) 110 MIPS
- b) 150 MIPS
- c) 125 MIPS
- d) 130 MIPS

ANS:- D

14. Which of the following has the same instruction set as ARM7?

- a) ARM6
- b) ARMv3
- c) ARM71a0
- d) ARMv4T

ANS:- B

15. What are t, d, m, I stands for in ARM7TDMI?

- a) Timer, Debug, Multiplex, ICE
- b) Thumb, Debug, Multiplier, ICE
- c) Timer, Debug, Modulation, IS
- d) Thumb, Debug, Multiplier, ICE

ANS:- A

16. ARM stands for _____

- a) Advanced RISC Machine
- b) Advanced RISC Methodology
- c) Advanced Reduced Machine
- d) Advanced Reduced Methodology

ANS:- C

17. What are the profiles for ARM architecture?

- a) A,R
- b) A,M
- c) A,R,M
- d) R,M

ANS:- C

18. ARM7DI operates in which mode?

- a) Big Endian
- b) Little Endian
- c) Both big and little Endian
- d) Neither big nor little Endian

ANS:- C

19. In which of the following ARM processors virtual memory is present?

- a) ARM7DI
- b) ARM7TDMI-S
- c) ARM7TDMI
- d) ARM7EJ-S

ANS:- A

20. How many instruction pipelining is used in ARM7EJ-S?

- a) 3-Stage
- b) 4-Stage
- c) 5-Stage
- d) 2-stage

ANS:- C

21. How many bit data bus is used in ARM7EJ-s?

- a) 32-bit
- b) 16-bit
- c) 8-bit
- d) Both 16 and 32 bit

ANS:-A

22. What is the cache memory for ARM710T?

- a) 12Kb
- b) 16Kb
- c) 32Kb:-
- d) 8Kb

ANS:- D

23. _____ is a complete line of home IoT devices that includes smart switches.

- a) Awair
- b) Canary
- c) Belkin's WeMo
- d) Cinder

ANS:- C

24. UART is similar to _____

- a) SPI protocol
- b) I2C protocol
- c) HTTP protocol
- d) MQTT protocol

ANS:-B

25. What does UART contains?

- a) Parallel register
- b) Shift register
- c) Clock
- d) Parallel shift register

ANS:- B

26. Communication in UART is _____

- a) Only simple
- b) Only duplex
- c) Only full duplex
- d) Simplex, half duplex, full duplex

ANS:-D

27. Start bit of UART is logic high.

- a) True
- b) False

ANS:-B

28. Which error occurs when the receiver can't process the character?

- a) Overrun error
- b) Underrun error
- c) Framing error
- d) Break condition

ANS:- A

29. What is the speed of the 8250 UART?

- a) 4800bits/sec
- b) 1200bits/sec
- c) 12000bit/sec
- d) 9600bits/sec

ANS:- D

30. Which error occurs when UART transmitter has completed sending a character and the transmit buffer is empty?

- a) Overrun error
- b) Underrun error
- c) Framing error
- d) Break condition

ANS:- B

31. Which error occurs when the designated start and stop bits are not found?

- a) Overrun error

- b) Underrun error
- c) Framing error
- d) Break condition

ANS:-C

32. Secure digital card application uses which protocol?

- a) UART
- b) SPI
- c) I2C
- d) USART

ANS:- B

33. SPI device communicates in _____

- a) Simplex
- b) Half duplex
- c) Full duplex
- d) Both half and full duplex

ANS:- C

34. Do SPI have/has a single master?

- a) True
- b) False

ANS:- A

35. SPI is described as Asynchronous serial interface.

- a) True
- b) False

ANS:- B

36. How many logic signals are there in SPI?

- a) 5 signals
- b) 6 signals
- c) 4 signals
- d) 7 signals

ANS:-A

37. SPI uses how many lines?

- a) 4 lines
- b) 1 line
- c) 3 lines
- d) 2 lines

ANS:- D

38. The main importance of ARM micro-processors is providing operation with _____

- (a) Low cost and low power consumption
- (b) Higher degree of multi-tasking
- (c) Lower error or glitches
- (d) Efficient memory management

ANS:-A

39. ARM processors where basically designed for _____

- (a) Main frame systems
- (b) Distributed systems
- (c) Mobile systems
- (d) Super computers

ANS:-C

40. The ARM processors doesn't support Byte address ability ?

- (a) True
- (b) False

ANS:-B

41. The address space in ARM is _____

- (a) 2^{24}
- (b) 2^{64}
- (c) 2^{16}
- (d) 2^{32}

ANS:-D

42. The address system supported by ARM systems is/are _____

- (a) Little Endian
- (b) Big Endian
- (c) X-Little Endian
- (d) Both Little & Big Endian

ANS:-D

43. Memory can be accessed in ARM systems by _____instructions.

- i) Store
- ii) MOVE
- iii) Load
- iv) arithmetic
- v) logical

- (a) i,ii,iii
- (b) i,ii
- (c) i,iv,v
- (d) iii,iv,v

ANS:- B

44. RISC stands for _____

- (a) Restricted Instruction Sequencing Computer
- (b) Restricted Instruction Sequential Compiler
- (c) Reduced Instruction Set Control.
- (d) Reduced Induction Set Computer.

ANS:- C

45. In ARM, PC is implemented using _____

- (a) Caches
- (b) Special function register
- (c) General purpose register
- (d) Stack

ANS:- C

46. The additional duplicate register used in ARM machines are called as

- (a) Copied-registers
- (b) Banked registers
- (c) EXtra registers
- (d) Extential registers

ANS:-B

47. The banked registers are used for

- (a) Switching between supervisor and interrupt mode
- (b) Extended storing
- (c) Same as other general purpose registers
- (d) None of the mentioned

ANS:- A

48. Each instruction in ARM machines is encoded into ____Word.

- (a) 2 byte
- (b) 3 byte
- (c) 4 byte
- (d) 8 byte

ANS:-C

49. All instructions in ARM are conditionally executed.

- (a) True
- (b) False

ANS:-A

50. Thumb-2 technology is implemented in which of the following?

- (a) All ARM processors
- (b) All ARMv7 processors
- (c) ARMv7-A processors only
- (d) ARMv7-A and ARMv7-R but not ARMv7-M

ANS:-B

51. The ARM processor registers R13, R14, and R15 are architecturally used for special purposes.

Which is the correct respective sequence of special purpose registers?

- (a) PC, LR, SP
- (b) LR, PC, SP
- (c) SP, LR, PC
- (d) LR, SP, PC

ANS:- C

52. CISC stands for

- (a) Complex Instruction Sequencing Computer
- (b) Complex Instruction Sequential Compiler
- (c) Complex Instruction Set Control
- (d) Complex Induction Set Computer

ANS:- C

53. Microcontrollers are called _____

- (a) application-specific integrated circuit
- (b) applied system integration control
- (c) application-specified integration circuit

ANS:- A

54. The addressing mode where the EA of the operand is the contents of Rn is _____

- a) Pre-indexed mode
- b) Pre-indexed with write back mode
- c) Post-indexed mode
- d) None of the mentioned

ANS:-C

55. The effective address of the instruction written in Post-indexed mode, MOVE[Rn]+Rm is _____

- a) $EA = [Rn]$.
- b) $EA = [Rn + Rm]$.
- c) $EA = [Rn] + Rm$
- d) $EA = [Rm] + Rn$

ANS:- A

56. LPC 1768 pro development board has _____ on chip memory.

- a) 500k
- b) 625k
- c) 512k
- d) 425k

ANS:- C

e)

ANS:- A

57. In LPC 1768 we require separate programmer?

- a) True
- b) False

ANS:- B

58. Which LCD display is present in LPC 2148 Development Board?

- a) 8*8 LED
- b) 2*32 LCD
- c) 2*16 LCD connected peripherally
- d) 2*16 LCD on-chip

ANS:- D

59. It have in system programming or in application programming?

- a) True
- b) False

ANS:-A

60. It provides real time debugging with the on chip real monitor software.

- a) True
- b) False

ANS:- A

61. Who is the founder of LPC1768 board?

- a) Intel
- b) Atmel
- c) Motorola
- d) Philips

ANS:-D

62. What is the program counter value when the board turns on?

- a) 0x00000
- b) 0xFFFFF
- c) Where the previous program ends
- d) At the location where we write the code

ANS:- A

63. Which IDE is supported by LPC2148 board?

- a) Code Blocks
- b) AVR Studio 4
- c) Keil uVersion 4
- d) Walldorf

ANS:- C

64. _____bit ARM CortexM3 controller is present?

- a) 128 bit
- b) 8 bit
- d) 64 bit
- d) 32 bit

ANS:- D

65. What is the processor used by ARM Cortex M3?

- a) 8-bit CISC
- b) 8-bit RISC
- c) 32-bit CISC
- d) 32-bit RISC

ANS:-D

66. What is the instruction set used by ARM Cortex M3?

- a) 16-bit instruction set
- b) 32-bit instruction set
- c) 64-bit instruction set
- d) 8-bit instruction set

ANS:- B

67. How many registers are there in ARM Cortex M3?

- a) 35 register(28 GPR and 7 SPR)
- b) 37 registers(28 GPR and 9 SPR)
- c) 37 registers(31 GPR and 6 SPR)
- d) 35 register(30 GPR and 5 SPR)

ANS:- C

68. ARM Cortex M3 has a in-built debugging device?

- a) True
- b) False

ANS:- A

69. What is the capability of ARM Cortex M3 instruction for second?

- a) 110 MIPS
- b) 150 MIPS
- c) 125 MIPS
- d) 130 MIPS

ANS:- B

70. In Cortex-R processor series, which among the following represent/s dual core configuration along with the space saving the floating point unit?

- a. Cortex-R 4
- b. Cortex-R 5
- c. Cortex-R 7
- d. All of the above

ANS:-B

71. Which types of an embedded systems involve the coding at a simple level in an embedded 'C', without any necessity of RTOS?

- a. Small Scale Embedded Systems
- b. Medium Scale Embedded Systems
- c. Sophisticated Embedded Systems
- d. All of the above

ANS:-

A

72. What is/are the configuration status of control unit in RISC Processors?

- a. Hardwired
- b. Microprogrammed
- c. Both a and b
- d. None of the above

ANS:-

A

73. How is the nature of instruction size in CISC processors?

- a. Fixed
- b. Variable
- c. Both a and b
- d. None of the above

ANS:-

B

74. Performance of Cortex-M3 processor with CoreMark 1.0 benchmark is

- a. 1.25 MHz
- b. 3.34 MHz
- c. 12.56 MHz
- d. 1.98 MHz

ANS:-

B