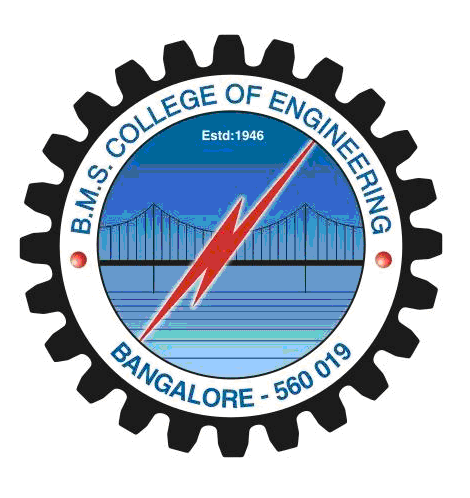
****

**B.M.S. College of Engineering, Bangalore – 19**

**(Autonomous & Affiliated College under VTU)**

**Department of Electronics & Communication Engineering**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Course Title** | **ARM Processor & Programming LAB** | | | | |
| **Course Code** | **21ES4CCAPP** | **Credits** | **4** | **L-T-P** | **3:0:1** |

**Course Objectives:**

**The student should be made to:**

* **Learn the working of ARM processor**
* **Understand the Building Blocks of Embedded Systems**
* **Learn the concept of memory map and memory interface**
* **Write programs to interface memory, I / Os with processor**
* **Study the interrupt performance.**

**Lab requisites:**

**Software and hardware tools used for ARM PROGRAMMING:**

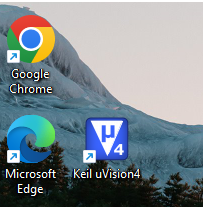
* **Keil µVision4 IDE** **(ASSEMBLY PROGRAMS – SIMULATION - PART - A)**
* **ARM7TDMI - LPC 2148 EVALUATION BOARD FOR INTERFACING EXPERIMENTS (PART - B)**

**PART-A EXPERIMENTS USING Keil µVision4 IDE (SIMULATION)**

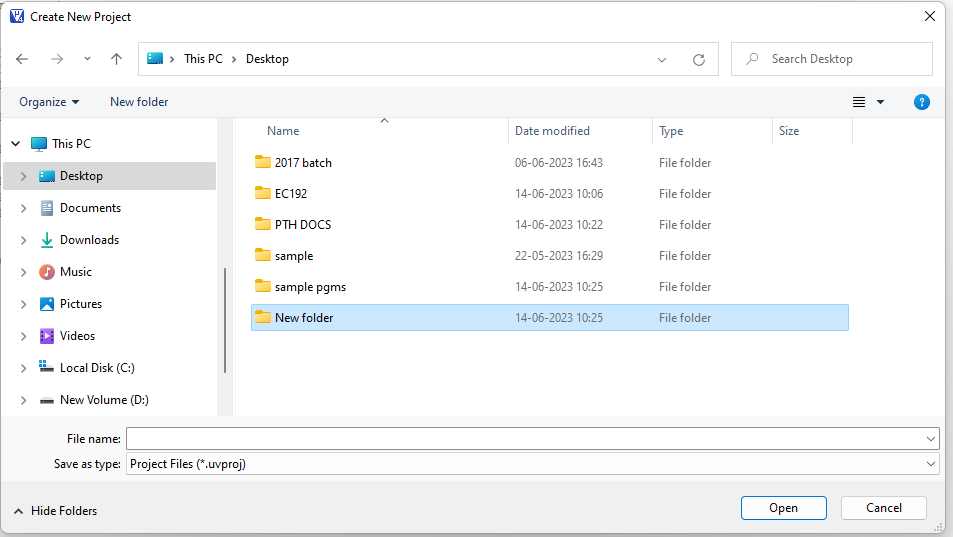
1. Add a series of 16-bit numbers stored in sequential location in the memory (called Table) and store the result in memory.
2. Add two 64-bit numbers and store the result in a memory location.
3. Sum of first 10 integer numbers.
4. Multiply two 16-bit binary numbers.
5. Find the factorial of a given number.
6. Divide an 8-bit variable into two 4-bit nibbles and store one nibble in each byte of a 16-bit variable. Store the disassembled byte in memory location (pointed by result).
7. Compare 2 values stored in memory location and store the higher value in a memory location (pointed by result).
8. Find the largest in a series of numbers stored in memory.

**PART A - ASSEMBLY PROGRAMS EXECUTION STEPS - Keil µVision4 IDE**

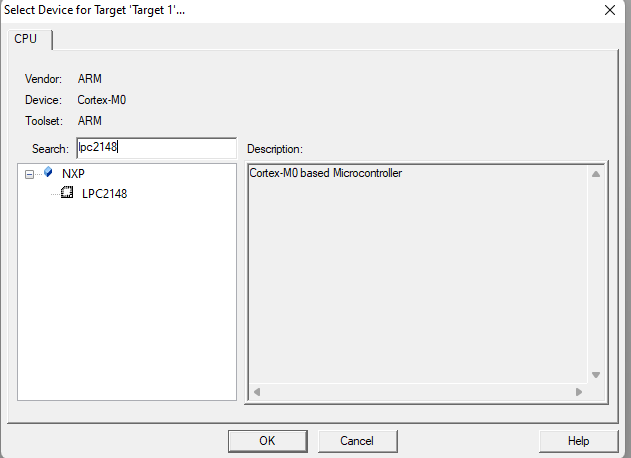
* Open **Keil µVision4 IDE** software by double clicking on **Keil µVision4** icon:



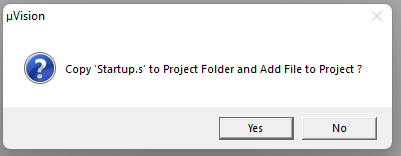
* Go to “**Project**”, select “**New µvision Project**” and save the project:



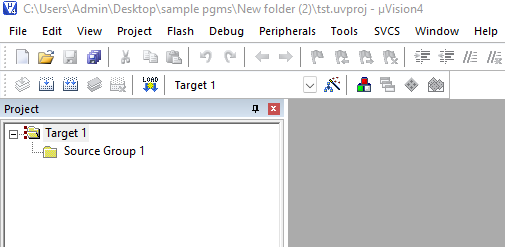
* Once the project is created, select the **target device** - select **LPC2148**, click on OK



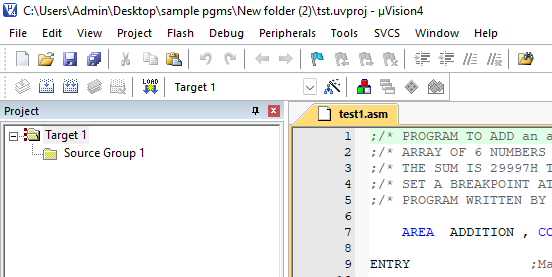
* A dialog box with **“copy startup. s to project folder and add file to project?”** will pop up, select **NO**.



* Thus, the target is created along with source group1 as shown below:

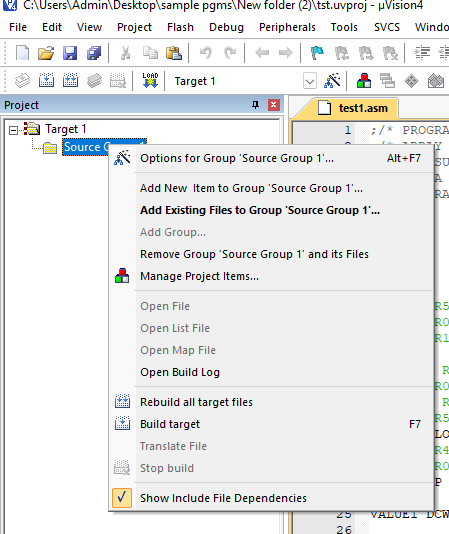


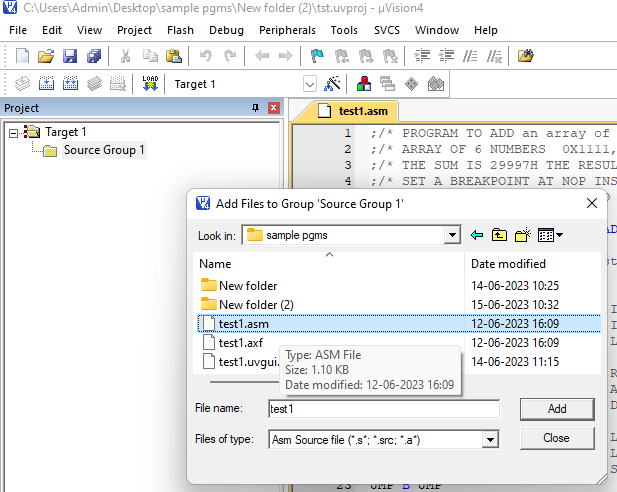
* Open the text editor. Write the program and save the program with the extension **.asm**



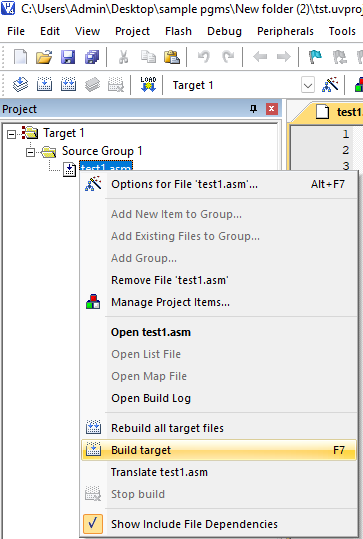
* Once the program is saved, the program file must be added to the **target.**

Right click on the “Source Group 1”, click on **Add existing files to group “source group 1”** and then select the assembly program and click on Add, Close.

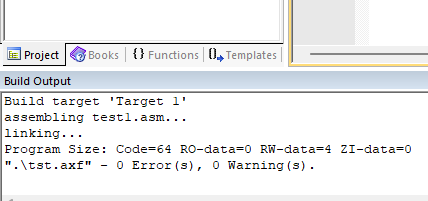




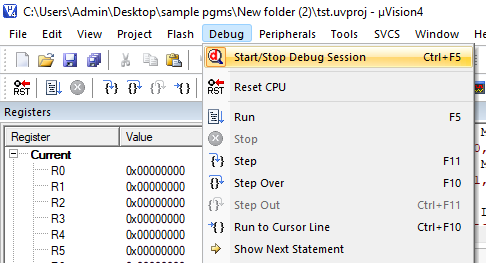
* Once the program is added to the source group, right click, select Build target or press F7



* Once the target is built, it displays the information with any errors or zero errors.



* If no errors, we can proceed for debug, select start /stop debugging option.

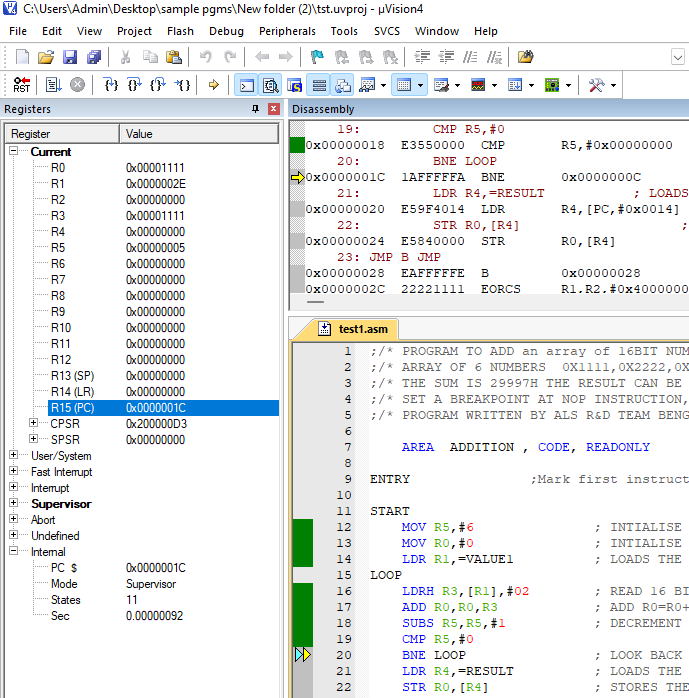


* Then select Run option from the debug menu or can use the key F5.

If F5 key - continuous run

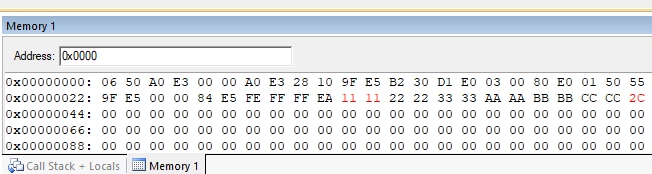
If F11key - step one line

If F10key – step over line

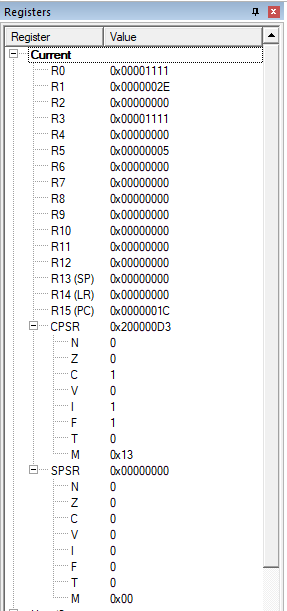


* The results obtained will be verified in the respective registers and in memory window as shown below:

Memory window:

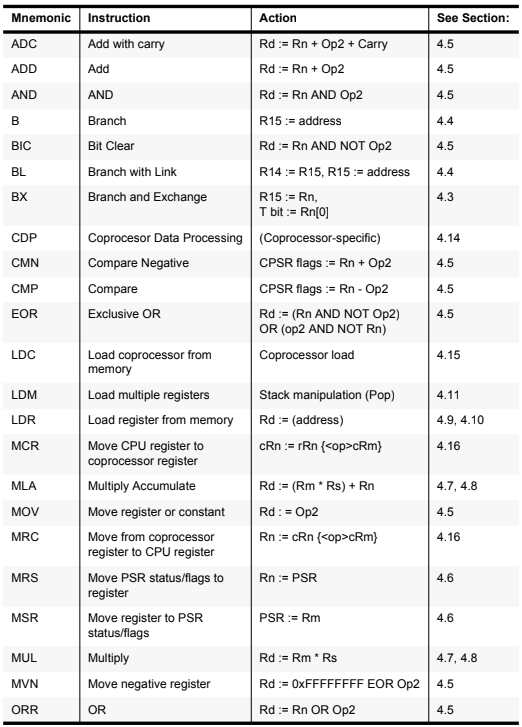


Registers window:

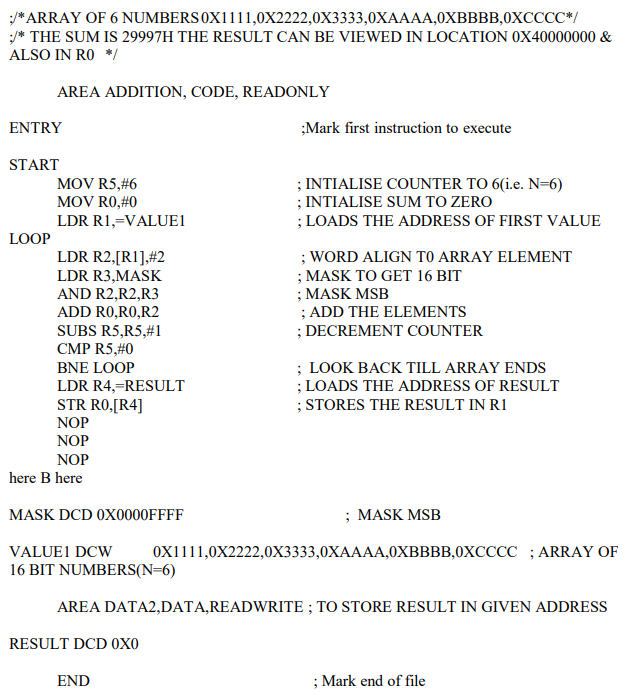


**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

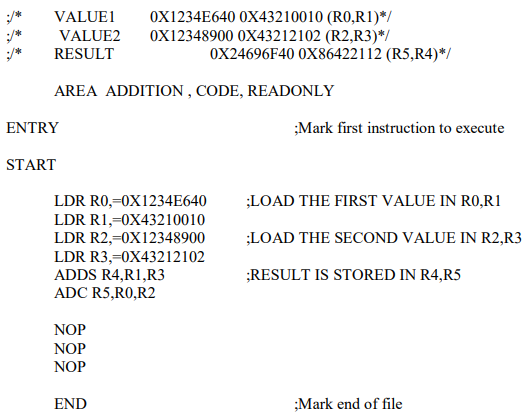
**Instruction set (for reference)**

****

* + 1. **Write an ALP to Add a series of 16-bit numbers stored in sequential location in the memory (called Table) and store the result in memory.**



* + 1. **Write an ALP to Add two 64-bit numbers and store the result in a memory location.**



* + 1. **Write an ALP to find sum of first 10 integer numbers.**

AREA SUM1, CODE, READONLY

START

MOV R1, #10;

MOV R2, #00;

LOOP

ADDS R2, R2, R1;

SUBS R1, R1, #01;

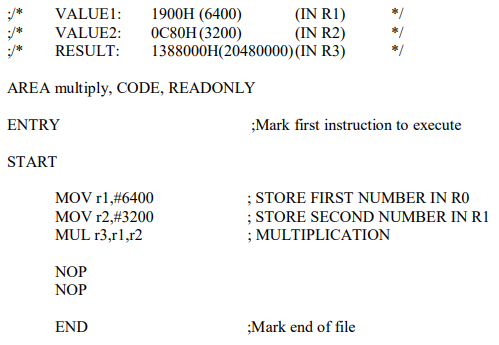
BNE LOOP;

L B L

NOP

END

* + 1. **Write an ALP to Multiply two 16-bit binary numbers.**



* + 1. **Find the factorial of a given number**

AREA FACTORIAL, CODE, READONLY

ENTRY ;Mark first instruction to execute

START

MOV r0, #7 ; STORE FACTORIAL NUMBER IN R0

MOV r1, r0 ; MOVE THE SAME NUMBER IN R1 FACT

FACT SUBS r1, r1, #1 ; SUBTRACTION

CMP r1, #1 ; COMPARISON

STOP BEQ STOP

MUL r3, r0, r1 ; MULTIPLICATION

MOV r0, r3 ; Result

BNE FACT ; BRANCH TO THE LOOP IF NOT EQUAL STOP

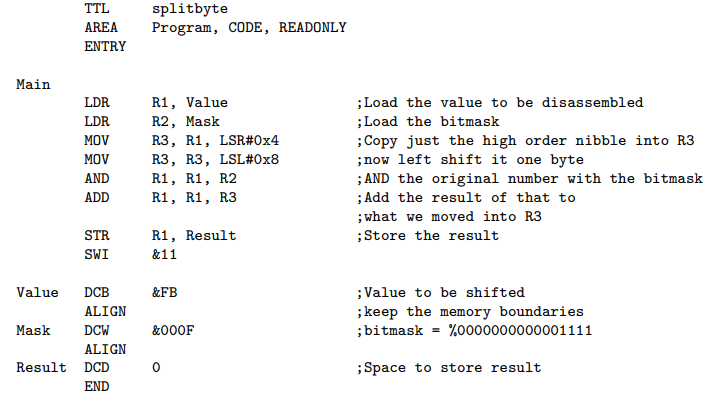
NOP

NOP

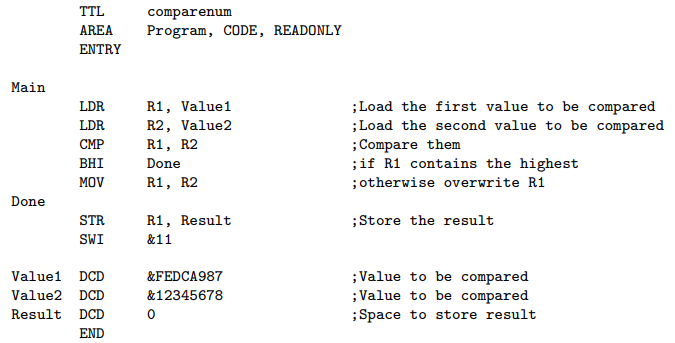
NOP

END ;Mark end of file

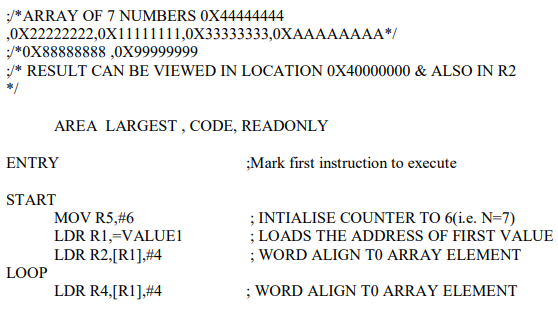
* + 1. **Divide an 8-bit variable into two 4-bit nibbles and store one nibble in each byte of a 16-bit variable. Store the disassembled byte in memory location (pointed by result).**

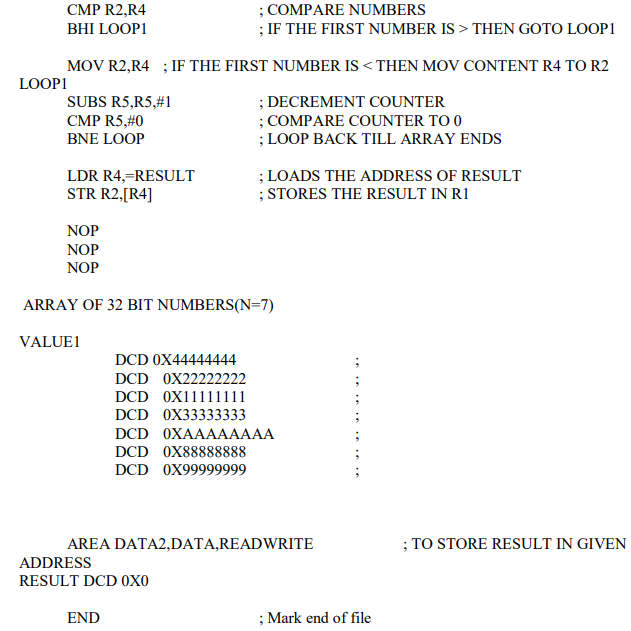
****

* + 1. Compare 2 values stored in memory location and store the higher value in a memory location (pointed by result).



* + 1. Find the largest in a series of numbers stored in memory.





\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*