California State University, Northridge

ECE 422L - Computer Architecture

Fall 2022

Lab 2

Assembly Warm up -Part 2

February 23, 2023

Instructor: Dr. Shahnam Mirzaei

Mark Attia

Dima Basbous

**Table of Contents**

**Introduction 3**

**Procedure 3**

**Testing Strategy 3**

**Results 3**

**Conclusion 4**

**Appendix 4**

**Introduction**

The purpose of this experiment is to familiarize ourselves with the keil uvision software.

We will be able to do our entire setup with the tiva boards and continue to learn the basics of ARM Assembly programming. The idea here is to keep familiarizing ourselves with the process of ARM programming so that we can build on us foundation later with more instructions and complex manipulation. This will allow us to get even more practice with manipulating different types of data and practicing being efficient with our code.

**Procedure**

1. Create Project Lab2

2. Ensure all details are correct for compiling code

3. Simulation debugger

4. Drivers for board installed -Using Tiva Core (for setup but not needed this lab)

5. Use sample program provided in manual to complete task 1 (average)

6. Use debug tool to monitor step by step execution and logic (make sure it's right)

7. Complete task 2- Calculate factorial of given number and store result in memory

8. Use simulation debug to check logic and step by step results

9. Complete task 3 – Calculate Fibonacci sequence and store result in memory

10. Use simulation debug to check logic and step by step result

It is also worth noting that within the debug tool we are allowed access to our memory as

well, meaning we can check any address we send data to ensure everything is being.

executed as expected.

**Testing Strategy**

The testing strategy for this experiment comes from being able to start our own project.

based on the demos provided to us prior to this experiment. Initializing our project correctly and

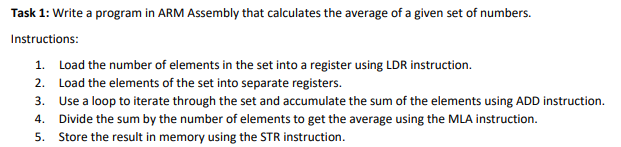
making sure our files are all in the right place is half the battle to reduce errors when we write.

our program. After this it is mostly a matter of understanding the basic instructions and them

functionalities and how we can manipulate them to achieve each task. So long as we can create a

program that can create a target, we will be able to use the debug tool to troubleshoot.

**Results**



**A picture containing graphical user interface

Description automatically generated**

**Fig. 1: Average**

R2=N where N=10. Here we calculate the average by computing the sum of 10+9+8+7+6+5+4+3+2+1=55. We then divide 55/10 to get 5.5 which the assembler will store into register R0 as 5.

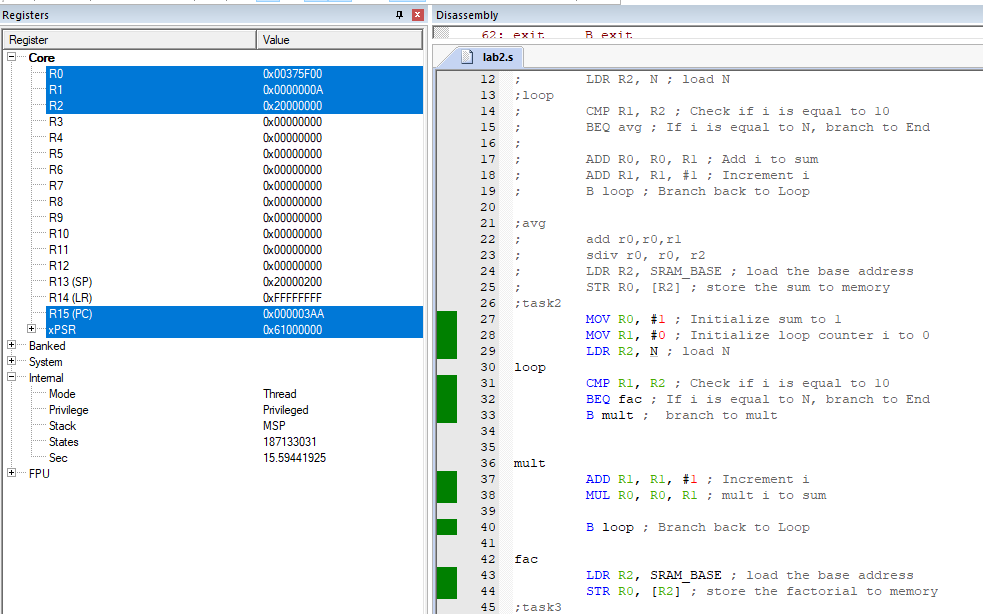
Graphical user interface, text, application

Description automatically generated

**Fig. 2: R0 stored in Memory.**

Graphical user interface, text, application

Description automatically generated



**Fig. 2: Calculate Factorial**

Here R1=N where N=10. We calculate 10! To be 0x00375f00 which is hex for 3628800 stored in R0.

Graphical user interface, text, application

Description automatically generated

**Fig. 3: Store Factorial in Memory**

**Text

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Fig. 3: Fibonacci Sequence of N**

Here N is not shown as a register value as we used it to decrement our loop. N was initialized to be 10 into R2 and used as a counter to perform the sequence. We see the sequence of N=10 yields 55 or 0x37 in R0.

Graphical user interface, text, application

Description automatically generated

**Fig. 4: Store Fibonacci result in Memory.**

**Conclusion**

This experiment allowed us to continue getting familiar with using keil uvision and ARM assembly programming. It is helpful to have some programming background for understanding certain logic but essential to understand the differences between assembly and high-level programming. Having strong troubleshooting skills is essential for assembly programming as that is what allows us to truly understand the logic we are implementing. Here we got a little more practice with loops and different branch conditions and also implemented manipulations of DCD values. Together this forms the backbone for assembly programming and will allow us to do many more things with microcontrollers for future experiments.

**Appendix**

https://github.com/EgyptiansFTW/Computer-Architecure-Lab.git