Lab 4: Hand Assembly

Joseph Salazar
College of Engineering
University of Miami
Miami, United States
jes409@miami.edu

***Abstract*— The ARM assembly language is widely used by professionals in the computer engineering field. However, there is very little material that introduces entry-level persons into assembly language. To help individuals better understand assembly for their futures in engineering, we propose following a quick and efficient program provided by the University of Miami's College of Engineering. This program allows individuals to practice assembly by writing a short sequence of code, followed by manipulating the hand assembly instructions.**

Keywords—ARM, assembly,

I. Introduction

The purpose of this lab is to interact and familiarize oneself with the ARM hand assembly code. By becoming familiar with hand assembly, one can "hand optimize-algorithms or maintain legacy code" [2]. These are both important reasons to learn assembly as it will benefit the programmer's future in the field of Computer Engineering. To demonstrate the importance of this, a programmer will be asked to write a simple code in ARM assembly that contains at least two arithmetic or logical operations, and two different branch conditions. It is important to note that one of the arithmetic operations must update the CPSR as well. The problem one is trying to dissect is understanding how the hand assembled code is affected by the different components that make up the ARM assembly code. For example, one should observe the hand assembled code between an "add" and "adds" operations and be able to explain how the hand assembly is affected. One will also be expected to explain this difference amongst branch operations with conditional modifiers and be able to manipulate the hand assembly in memory. This different approach will force oneself to fully understand what is happening in memory, and how one can manually make changes without changing the ARM assembly code.

II. Method/Experient

A. Overview

Before attempting this lab, one should have Altera Monitor Program installed before continuing. Then, power on the De1 board and connect it to the computer. Once connected, one should proceed to set-up their coding environment/document so that the code remains organized. After one's ".s" file is created, the programmer should include the data, global, text, start, and end directives to ensure compilation goes smoothly after code has been added. If one is not familiar with this routine, referencing the DE1 manual would be most beneficial [2].

To begin this lab, one is asked to write a short program that initializes two integers, loads them into registers, performs an arithmetic operation that updates the CPSR, and finally either branches to the beginning or branches to termination based on a condition.

Once this program is working properly, the programmer is asked to hand assemble the arithmetic operation, as well as their two branch operations. Once this is achieved, one should compare their hand-assembly with the software's to confirm that they have done it correctly. One should also note where these instructions are stored in memory at this point.

Next, now that one knows where the instructions are stored in memory and how to accurately hand assemble code, one should modify their branch operations. The modification should exchange the order of the two branch operations written in the code. For example, if one wrote "b loop" followed by "beq terminate," the memory should be changed so that it is changed to "beq loop" and "b terminate" respectively. This can be observed by comparing the differences in Figure 1 and Figure 2. Once this is done, one should now manually change the arithmetic operation so that it does not update the CPSR. This can be observed by comparing Figure 1 and Figure 3. By making these changes, the programmer will have proven to oneself that they have a deep understanding for hand assembly, and how the hexadecimal values change based on conditional modifiers and other operations in ARM assembly.

B. Results

Subs r4, r2, r3

1110	0000	0101	0010	0100	0000	0000	0011

Hand Assembly:
E0524003

B loop

1110	1010	1111	1111	1111	1111	1111	1010

Hand Assembly:
EAFFFFFA

Beq done

0000	1010	0000	0000	0000	0000	0000	0000

Hand Assembly:
0A000000

Figure 1. The following figure shows the hand-assembled code for the three main operations.

B loop becomes:

0000	1010	1111	1111	1111	1111	1111	1010

0AFFFFFA

Beq done becomes:

1110	1010	0000	0000	0000	0000	0000	0000

EA000000

Figure 2. The following figure shows the modified branch operations once they are inverted. The conditional modifiers change the hex-value.

Modifying the CPSR from Subs r4, r2, r3
Subs r4, r2, r3 becomes:

1110	0000	0100	0010	0100	0000	0000	0011

E0424003

Figure 3. The following figure shows the modified subtract operation so that it does not affect the CPSR.

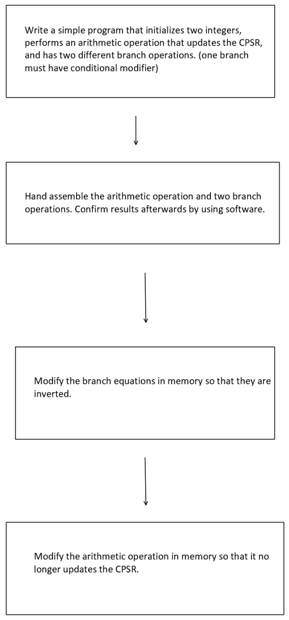
Write a simple program that initializes two integers, performs an arithmetic operation that updates the CPSR, and has two different branch operations. (one branch must have conditional modifier)
Hand assemble the arithmetic operation and two branch operations. Confirm results afterwards by using software.
Modify the branch equations in memory so that they are inverted.
Modify the arithmetic operation in memory so that it no longer updates the CPSR.

Figure 4. The following figure shows a flow chart for this exercise.

C. *Discussion*

After completing the previous tasks, one could reflect and take many lessons away. These exercises emphasize the importance of knowing how to navigate through memory, manipulate hand assembly, and be able to write hand assembly on their own. All of these are very critical in understanding for ones future career path in engineering.

Ultimately, this exercise should not provide too much difficulty because it only includes writing a very simple program. One should take their time when constructing their hand-assembly to best understand the contents of their program. If one wanted to learn even more about hand-assembly, one could add different operations in their code and hand assemble them to see if they can accurately recreate it on their own.

III. CONCLUSION

In sum, this exercise introduces instructions in ARM assembly that are essential for more difficult programs. After following the instructions, the programmer will have walked away with a deeper understanding of hand assembly and how it affects their programs. The results gathered from this exercise reinforce this idea by forcing the programmer to hand assemble on their own, as well as overwrite instructions stored in memory. Ultimately, this will help the programmer excel as a computer engineer by learning new and important content in ARM and hand assembly.

University of Miami

REFERENCES

[1] ARM Limited. ARM Architecture Reference Manual. 1996.

[2] *Documentation–ArmDeveloper*, developer.arm.com/documentation/ddi0388/latest/.

[3] "GNU Manuals Online - GNU Project - Free Software Foundation." *[A GNU Head]* , www.gnu.org/manual/manual.en.html.

[4] Pyeatt, Larry D. *Modern Assembly Language Programming with the ARM Processor*. Newnes, 2016.

Code Used

```
.data
ar1:        .word     5
ar2:        .word     4

.global _start
.text

_start:
            ldr r0,=ar1
            ldr r1,=ar2
loop:
            ldr r2,[r0]
            ldr r3,[r1]
            subs r4, r2, r3
            beq         done
            b loop
done:
            b done
.end
```