Laboratory Exercise 7

Using the ARM A9 Processor

Revision of November 4, 2021

The purpose of this lab is to:

1. learn how to write basic programs in assembly language

This exercise provides an introduction to the ARM A9 processor, its assembly language, and development tools. A portion of the mark for this lab will be tied to questions posed by the TA in your lab session.

1 Preparation

To prepare for this exercise you need to be familiar with the ARM A9 processor architecture and its assembly language. This processor is described in the tutorial called Introduction to the ARM Processor, which is available on the Quercus page. In Section 5.1, it is highly recommended you ignore the pre-indexed and post-indexed addressing modes for now. You should also ignore Section 6.1.1 on loading multiple registers. Likewise, for now, ignore Section 6.3.1 and Section 6.10 on conditional instructions. The ARM processor is also described in Appendix D of the Computer Organization textbook for ECE253. You are strongly encouraged to watch the ARM tutorial video provided on Quercus prior to starting the lab.

For this exercise, you will use an ARM A9 processor that is part of a computer system called the DE1-SoC Computer. This computer system is implemented inside the Intel Cyclone V SoC chip on the Intel DE1-SoC lab board. You can develop your code at home using a simulator and then during your practical session, you can run the code directly on the ARM A9 processor. For Labs 8 and 9, you will be required to demo your code on the lab boards, so it would be good practice to try this in Lab 7 as well.

The simulator for the ARM processor on the DE1-SoC chip has been created by one of our recent Ph.D. graduates, Dr. Henry Wong. The simulator is available here:

`https://cpulator.01xz.net/`

On the landing page, as the system, select "ARMv7 DE1-SoC". You will then enter an environment where you can type assembly code and execute it. Use the simulator to develop and test the solutions for Labs 7-9 of ECE253. You should employ testing and debugging strategies similar to those you have used in other programming classes. The simulator allows you to single-step through code, set breakpoints, etc.

2 Part I

In this part, you will write your own ARM assembly language program that adds together a list of positive numbers (numbers>0) and deposits their sum into register R7, as well as counting the number of positive numbers and depositing the count into register R8. The list of numbers is terminated by a -1; the size of the list is not known in advance. Skeleton code is given below in Figure 1. The list of numbers is at a memory location with label TESTNUM. In this case, the value 21 (0x15) should appear in R7 when the program complete; the value 5 (0x5) should appear in R8 when the program completes. The automarker will include other test cases beyond this example.

1. In the cpulator website, write your assembly language code.
2. Compile and load the program. Fix any errors that you encounter. Once the program is loaded into memory in the DE1-SoC Computer, single step through it to see how the program works.
3. Save your program as part1.s
4. Prior to submitting your code for marking, please remove the line starting with “TEST-NUM: ”.

```
            .text
            .global     _start
_start:
...
... Write your code here
...

END:        B           END

TEST_NUM:   .word       1,2,3,5,0xA,-1

            .end
```

Figure 1: Skeleton Code for Part 1

3 Part II

Perform the following:

1. Create a new assembly language program. Start by typing the assembly language code shown in Figure 2 into the cpulator simulator. This code uses an algorithm to count the longest string of 1s in a word of data. The algorithm uses shift and AND operations to find the required result—make sure that you understand how this works.
2. Compile and load the program. Fix any errors that you encounter (if you mistyped some of the code). Once the program is loaded into memory in the DE1-SoC Computer, single step through it to see how the program works.
3. Save your program as part2.s
4. You will not be asked to submit this part.
5. You will be asked to modify this code as part of Lab 8, so its useful to understand it now.

```
/* Program that counts consecutive 1's */
          .text
          .global   _start
_start:
          LDR       R2, =TEST_NUM // load the data word into R2
          LDR       R1, [R2]
          MOV       R0, #0         // R0 will hold the result
LOOP:     CMP       R1, #0         // loop until the data contains no more 1's
          BEQ       END
          LSR       R2, R1, #1     // perform SHIFT, followed by AND
          AND       R1, R1, R2
          ADD       R0, #1         // count the string lengths so far
          B         LOOP

END:      B         END

TEST_NUM: .word     0x103fe00f

          .end
```

Figure 2: Code for Part 2

4 Submission

Please submit part1.s. Be sure to account for any submission guidelines mentioned above.