Oregon State University

CS 472 - Computer Architecture

Spring 2014

Lab 2 - The ARM ISA

Author:
Drake Bridgewater
Ryan Phillips

Professor:
Kevin McGrath

April 17, 2014

Getting started

Begin with reading the introduction the KEIL simulator on pages 16-29 of the Student Workbook which accompanies the textbook

Part 1

Please provide answers to the problems in Problem Set 5 in the Student Workbook. Here we provide an introduction to the Keil ARM processor development system.

1)

Write a simple program to perform: $Z = A + B + B - (D \times E)$

The instructions you may use are ADD, SUB, and MUL. Assume that the data is in registers r0 to r4 (representing A to E) and the result is put in r5.

Enter your program into the Keil simulator and run it. You can use move instruction to load data into registers. Do you get the expected answer?

While watching the register reflect each line of code I was able to view each commands effects on the bits. This allowed me to realize that the add operation and subtraction operations are performed exactly as expected, but once the multiplication command came into play I noticed that the numbers di

```
AREA lab2_1, CODE, READONLY
ENTRY

MOV r0, #2                          ;a
MOV r1, #3                          ;b
MOV r2, #4                          ;c
MOV r3, #5                          ;d
MOV r4, #6                          ;e


ADD r1,r1,r1                        ;r1=r1+r1 || b=b+b
ADD r1,r1,r0                        ;r1=r1+r0 || b=a+(b+b)

;for muls you have to have diff destination register
MULS r5,r3,r4                       ;r5=r3+r4 || z=d*e
SUB r5,r1,r5                        ;r5=r1-r3 || z=b-(d*e)

END
```

2)

Now assume A, B, C, D and E are 16-bit values in memory. You can load them by using a DCD directive. Remember that you use a label to define the first memory location and you can put successive values on the same line by separating them by commas. However, since each data item needs its own name, you are going to have to use one directive per element; that is:,

A DCD 4 B DCD 12 C DCD -2

Enter the program, compile (build) it and test it.

3)

Write a program that includes deliberate syntax errors. Enter it in the development system, assemble (build) it and then debug it.

Part 2: Examination of compiler output

In the homework, you were asked to write an assembly routine that checks for palindromes in odd length strings. For this portion of the lab, please write a simple C program that does the same thing, use the web based compiler and compile your C code to ARM assembly (use arm-linux-gnueabi-g++-4.6 with option -O0). Compare this assembly with what you wrote. Modify the optimization level from 0 to one of the values in the set 1,2,3,s. What changes? Why do you think these changes were made?

For this lab, you will need to create a write-up discussing the differences you see in Part 2. Do you see any interesting features being used? Is the hand assembly you wrote significantly different than what the compiler produced? Which uses fewer instructions? In other words, provide an analysis of Part 2.