Jordan Hyman
CMPE 12
Section 3
11/18/2013

# Lab 4 Write-Up

- Introduction
  - The purpose of this lab was to give us a better understanding of ARM assembly, as well as how to not only light up, but control an LED via the Raspberry PI.
- Procedure
  - Part 1
    - Re-type the provided code into vim/nano
    - Save as a .s file
    - Compile the code using "as" and "ld"
    - Run the compiled code in GDB and find the register values
    - Try out the TUI interface in GDB
  - Part 2
    - Re-type the provided code into vim/nano
    - Replace the "???" lines with code that will allow the file to properly compile and execute, making sure to only use the allowed instructions.
  - Part 3
    - Light an LED using the Raspberry PI by following the detailed explanation
  - Part 4
    - Connect the LED to header pin 7
    - Do not get confused
    - Use the commands provided to control the GPIO 7
- Algorithm/Other Data
  - Part 2
    - Line 17 needs to branch to workdone on the chance that no argument is given
    - Lines 20 and 21 help to initialize r10 as the string length storage as well as
    - Onemore essentially checks the string for more letters, incrementing r10 and then ending once the word is written
    - Reverse then takes the string and places it into reverse order
      - In order for reverse to end, r11 was initialized at 0 and told to increment for every cycle of reverse. When r11 == r10, reverse is finally exited.
        - NOTE: This was thought up while in the lab with Anthony Dupar and Beau Meimban, the three of us helping one another with Part 2.
- Other Information
  - What are the values for all the registers after executing 5 assembly instructions after the _break?
    - R0: 13, R1:65688, R2: 13, R7: 4, R13: (void *) 0xbefffd30, R15: (void(*)()) 0x8088 <_start+20>
      - Any unmentioned registers were 0.
  - What do you see in the top window after executing 3 ARM instructions?
    - The registers and their values
      - R0: 1, R1: 65688, R2: 13, CPSR: 16, SP: 0xbefffd30, PC: 0x8080

        - Again, any unmentioned registers were 0.
- If you were to run the code from task 1 on your lab3 LC-3, how many clock cycles would be required for it to complete?
    - Ten cycles
- What do each of the commands do?
    - ld: used in the second part of the compilation process
    - as: used in the first part of the compilation process
    - and: ANDs two things together and saves it to a register
    - add: Adds two things together and saves it to a register
    - beq: Branch if Equal To
    - bge: Branch if Greater Than Or Equal To
    - bl: Branch and Link
    - bne: Branch if Not Equal To
    - blt: Branch if Less Than
    - cmp: set CC comparing two items
    - ldr: loads 32bits
    - ldrb: loads a byte
    - mov: Sets a register equal to something, LSs and Ass can also be done with this
    - strb: mem[x] = y, lower 8 bits from y
    - sub: Subtracts one value from another and saves it to a register
    - swi: OS call, like TRAP
- What is GDB used for? What does it do?
    - GDB is used for debugging. With it, you are able to execute a program step by step.
- What is "-o" and what does this command do?
    - It states where the target is.
- What is the hex code for where the program starts for task 1?
    - 0x0
- What is the voltage drop across a single LED?
    - 3.3V
- If you increased the resistance of the resistor in task 4, what would happen to the LED?
    - The light would be dimmer
- Instead of pin 7, what other pins could have been used to achieve the same goal in task 4? What line(s) of code would change?
    - 0 → 6: gpio mode/write # out/1/0