Lab 1, Spring 2016

Samuel J. Dickerson

December 20, 2015

Important:

- While collaborations are allowed, the names of collaborators should be mentioned in the assignment. Each student should submit individual assignments for grading despite collaboration.
- **Paper copy** of the assignment is due **before** the beginning of the next recitation.

1 Getting started in MIPS assembly language

1. Download the Mars simulator from `http://www.cs.missouristate.edu/MARS/`
2. Launch the Mars simulator.
3. Click on the Edit tab on the top left of the window. This should show a text editor where you would write your assembly program.
4. Go to File and click on New, to start a new program.
5. Let's start with a simple program.

   Start the program with:

   .text

   This says that the following are program instructions (and not, e.g. .data). Let's do the following:

   $t1 = 7 + 3 + 9

   First, let's put 7 in $t1:

   addi $t1, $zero, 7

   This says to take what is stored in register $zero, add 7 to it, and put the result in register $t1. $zero ALWAYS contains 0, so this puts 0 + 7 into register $t1.
6. Before you can go further, MIPS needs you to save the file.
7. Now, go to the Run tab and click on assemble.

   The Text segment shows you:

   - Address – where this instruction is stored in memory
   - Code – the machine code of the instruction

- Basic – [not too helpful at this point]
- Source – the original instruction you typed in

The Data Segment in the middle of the page shows you the contents of the part of memory where data is stored. We haven't put any data in memory, so the values here are all 0 (The simulator chose all zeros in this case, but in reality the memory would be initialized to random values depending on the electrical characteristics of each cell).

The bottom window gives messages from the simulator. Any error messages will be displayed here.

8. Click on Go to execute the machine code (technically, "simulate" it).
9. Look at $t1 in the panel on the right.
10. Ok, let's continue writing the program (to get back to your program, click on Edit).

    Now we want to add 3 to our running sum:

    addi $t1, $t1, 3

    And then we want to add 9:

    addi $t1, $t1, 9

11. Assemble the program again, and run it.

    You will see that $t1 contains 0x00000013

    The 0x just means that the number following it is in hex.

1.1 Questions

1. What decimal number is 0x00000013? _____

   Assemble the program again (you need to assemble it again to be able to run it again).

   Click Step under Run (or the triangle with the 1). This will step through your instructions one by one. As each instruction executes, you will see registers and memory being updated.

2. before the first instruction is executed (the program counter is the register labeled "pc"):

   program counter = _____

3. after the first instruction is executed:

   program counter = _____

   $t1 = _____

4. after the second instruction is executed:

   program counter = _____

   $t1 = _____

5. after the third instruction is executed:

   program counter = _____

   $t1 = _____

6. Complete this sentence:

   After each instruction, the PC is incremented so that it contains

   (a) A bit which is a 0 or 1.

   (b) A byte which is 8 bits.

   (c) A word which is 4 bytes.

   In a MIPS architecture, each byte has its own address (the byte is the "addressable unit").

7. How big are instructions in MIPS?

   Number of bits = _____

   Number of bytes = _____

   Number of words = _____

   This shows a good way to learn assembly language. Enter and assemble instructions; this will show you the machine code that is produced. Then, step through execution so you can see the effects of the individual instructions.

2 Memory

Before working with memory, you need to be able to add hex digits. Here is an example of adding 2 hex digits:

3A49 + 4BA9 = 85F2

Here is how a human does this (we'll look at hardware later in the course). Note that numbers without the 0x are decimal:

- Digit 0: $0\text{x}9 + 0\text{x}9 = 9 + 9 = 18 = 0\text{x}12(1 * 16^1 + 2 * 16^0)$
  So, write down the 2 and carry the 1.
- Digit 1: $1(carry) + 0\text{x}4 + 0\text{x}A = 1 + 4 + 10 = 15 = 0\text{x}F$
  So, write down the F. There is no carry.
- Digit 2: $0\text{x}A + 0\text{x}B = 10 + 11 = 21 = 0\text{x}15(1 * 16^1 + 5 * 16^0)$
  So, write down the 5 and carry the 1.
- Digit 3: $1(carry) + 0\text{x}3 + 0\text{x}4 = 1 + 3 + 4 = 8 = 0\text{x}8$
  So, write down the 8.

You can check yourself using a calculator (make sure you understand why these are the right calculations!):
$3 * 16^3 + 10 * 16^2 + 4 * 16^1 + 9 * 16^0 = 14921$
$4 * 16^3 + 11 * 16^2 + 10 * 16^1 + 9 * 16^0 = 19369$
$8 * 16^3 + 5 * 16^2 + F * 16^1 + 2 * 16^0 = 34290$
$14921 + 19369 = 34290$ ... So, above is right!

2.1 Questions

Now, you try one:

1. 1D25 + B72A = ?????

   Now, let's look at memory. Start by entering the following into the simulator:

   .data
   .word 12, 0x91, 0x50, 20, 0x3A, 17, 0x10, 0x22, 71, 40, 0x17

   ".data" is an assembler directive (instruction to the assembler) that tells the assembler we are in the data segment of memory. ".word" is an assembler directive that tells the assembler to store the following into subsequent words in memory. The 0x values are in hex. The other values are in decimal. The data segment of memory begins at address 0x10010000 Before assembling your program, try to figure out what hex values will be stored at which memory locations by the above directives. You won't lose credit for incorrect answers here before you check the assembler — you just need to take a stab at it.

   Now assemble the above code, and look at memory.

   Let's figure out what you are looking at. The + values across the top are in hex (even though the 0x is missing — they left it off to fit more on the screen).

2. Does each box shown in the data segment window represent a byte or a word? Please explain.

   Note: in MIPS, each byte of memory does have its own address. It's just that MARS does not show an address for every byte, to fit more on the display.

3. Now that you understand what you are looking at, fill out the following table.

Words	Hex Value (show the correct number of digits! 8 hex digits = 32 bits = 1 word)
12	
0x91	
0x50	
20	
0x3A	
17	
0x10	
0x22	
71	
40	
0x17	