CSCI 2321 (Computer Design), Spring 2018

Homework 2

Credit: 40 points.

1 Reading

Be sure you have read, or at least skimmed, the assigned readings from Chapter 2 up through 2.7.

2 Honor Code Statement

Please include with each part of the assignment the Honor Code pledge or just the word "pledged", plus one or more of the following about collaboration and help (as many as apply).[1] Text *in italics* is explanatory or something for you to fill in. For written assignments, it should go right after your name and the assignment number; for programming assignments, it should go in comments at the start of your program(s).

- This assignment is entirely my own work. *(Here, "entirely my own work" means that it's your own work except for anything you got from the assignment itself — some programming assignments include "starter code", for example — or from the course Web site. In particular, for programming assignments you can copy freely from anything on the "sample programs page".)*
- I worked with *names of other students* on this assignment.
- I got help with this assignment from *source of help — ACM tutoring, another student in the course, the instructor, etc. (Here, "help" means significant help, beyond a little assistance with tools or compiler errors.)*
- I got help from *outside source — a book other than the textbook (give title and author), a Web site (give its URL), etc.. (Here too, you only need to mention significant help — you don't need to tell me that you looked up an error message on the Web, but if you found an algorithm or a code sketch, tell me about that.)*
- I provided help to *names of students* on this assignment. *(And here too, you only need to tell me about significant help.)*

3 Problems

Answer the following questions. You may write out your answers by hand or using a word processor or other program, but please submit hard copy, either in class or in one of my mailboxes (outside my office or in the ASO).

[1] Credit where credit is due: I based the wording of this list on a posting to a SIGCSE mailing list. SIGCSE is the ACM's Special Interest Group on CS Education.

Tips:

- If the assignment asks you to do one or more problems from the textbook, *be sure* you get them from the edition specified in the syllabus; these sets of problems change from edition to edition.
- If a question requires you to do calculations, please show enough work to help me understand how you got the answer you did, so if you make a mistake I can give partial credit for anything you did get right.
- For some of the problems you're asked to do something that involves converting between hexadecimal (base 16) and binary. Remember(?) that this is a straightforward process for which you don't need a calculator — each hexadecimal digit represents four binary digits.

1. (5 points) Do problem 2.9 from the textbook.
2. (5 points) For each of the following MIPS instructions, translate it into machine language, first listing all the fields (e.g., opcode) in binary and then giving the 32-bit instruction in hexadecimal.
   - `sub     $t0, $s1, $s2`
   - `addi    $s1, $s1, -1`

     (We didn't do an example of `addi` in class, but like `lw` and `sw` it's an I-format instruction.)
3. (5 points) Do problem 2.14 from the textbook.
4. (5 points) Given the following initial contents for registers `$t1` and `$t2`:

   ```
   $t1  0xFFFFFFFF
   $t2  0x12345678
   ```

   For each of the following sequences of MIPS instructions, if `$t1` and `$t2` are as above, what does `$t0` contain, in hexadecimal, after it is executed?

   -
     ```
     sll     $t0, $t1, 16
     and     $t0, $t0, $t2
     ```

   -
     ```
     srl     $t0, $t1, 16
     or      $t0, $t0, $t2
     ```

   -
     ```
     sra     $t0, $t1, 16
     and     $t0, $t0, $t2
     ```

   -
     ```
     ori     $t0, $t2, 0xFF
     ```
     (Assume that the assembler is smart enough to convert `0xFF` to an appropriate 16-bit constant.)
5. (10 points) Do problem 2.27 from the textbook.
6. (10 points) Reverse-compile the following MIPS assembly code into equivalent C (without use of `go to`), using integer variable `i` to represent the value in `$t1` and integer variable `result` to represent the value in `$s2`. (You can use `int` to represent a 32-bit integer, which is MIPS's idea of a "word". So `MemArray` is an array of 100 `int`s.)

```
        .text
        addi    $t1, $0, 0
        la      $s0, MemArray
        addi    $s2, $0, 0
LOOP:   lw      $s1, 0($s0)
        add     $s2, $s2, $s1
        addi    $s0, $s0, 4
        addi    $t1, $t1, 1
        slti    $t2, $t1, 100
        bne     $t2, $0, LOOP

        .data
MemArray: .space 400  # reserve space for 400 bytes, i.e., 100 words
```

Note that as initially posted there was a typo, namely that the address of `MemArray` was being loaded into `$s1`.