# CSCI 2321 (Computer Design), Spring 2020

# Homework 3

**Credit:** 25 points.

## 1 Reading

Be sure you have read, or at least skimmed, all assigned sections of Chapter 2 and Appendix A.

## 2 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. (5 points) Consider this fragment of MIPS code, intended to be a typical MIPS version of if/else:

```
    slt     $t0, $s1, $s2
    beq     $t0, $zero, Else
    addi    $s3, $s3, 1
    addi    $s4, $s4, 1
    j       After
Else:
    addi    $s3, $s3, -1
    addi    $s4, $s4, -1
After:
```

   Translate the `beq` and `j` instructions into machine language, assuming that the first instruction is at memory location `0x00400040`. (You don't have to translate the other instructions, just those two. As in Homework 2, first list all the fields (e.g., opcode) in binary and then give the 32-bit instruction in hexadecimal.)

2. (5 points) The MIPS assembler supports a number of *pseudoinstructions*, which look like regular instructions but which assemble into one or more other machine instructions. We've seen how SPIM assembles the `la` pseudoinstruction into a combination of `lui` and `ori`. As another example, pseudoinstruction `ble` generates two instructions, a `slt` and then a `bne`, using "assembly temporary" register `$at`, with

```
ble $t0, $t1, There
```

   being translated to

```
slt $at, $t1, $t0
beq $at, zero, There
```

If you wanted the assembler to support the following pseudoinstructions, say what code (using real instructions) the assembler should generate for the given examples. As with `ble`, you should use `$at` if you need an additional temporary register.

- `bnz` with the two operands (register number and target label/address), that branches to the target if the register contents are nonzero. Example:
  ```
  bnz $s0, There
  ```
- `swap` with two register-number operands, which exchanges the values in the two registers. Example:
  ```
  swap $s0, $s1
  ```

## 3 Programming Problems

Do the following programming problems. You will end up with at least one code file per problem. Submit your program source (and any other needed files) by sending mail to my TMail address (or you can use `bmassing@cs.trinity.edu`) with each file as an attachment. Please use a subject line that mentions the course and the assignment (e.g., "csci 2321 hw 3" or "computer design hw 3"). You can develop your programs on any system that provides the needed functionality, but I will test them on one of the department's Linux machines, so you should probably make sure they work in that environment before turning them in.

1. (15 points) For this problem your mission is to make a small change to the code you wrote for problem 5 from Homework 2 (the one with the nested loops) and then add code to make it a complete program that prompts for values for `a` and `b` and prints values in `D`.

   The change is to change the index into `D` from `4*i + j` to `b*i + j`. So in other words, the code should be this:

   ```
   for (i = 0; i < a; i++)
       for (j = 0; j < b; j++)
           D[b*i + j] = i + j;
   ```

   This will assign values to consecutive locations in `D`, from index 0 through index `a*b-1`, which was my intent all along, but, well, I made a mistake. You can either change your own code, or you can start with my solution to the slightly revised problem, shared with you via Google Drive.

   Then add code to make the result a complete program that, run from SPIM, prompts for values for `a` and `b` and prints the ending values of the elements of `D` that were changed. I.e., for output the program should do the equivalent of the C code

   ```
       for (k=0; k<a*b; k++)
           printf("%d\n", D[k]);
   ```

   Programs `echo.s` and `echoint.s` on the sample programs page show how to input and output text and integer values.

   Note that for this program you will need do full multiplication to compute both `b*i` and `a*b`, and we haven't talked about the instructions for that, but for now the simplest way is to use pseudo-instruction `mul`, which takes three registers as operands, e.g.,

   ```
   mul $t0, $s1, $s2
   ```

   to multiply the contents of $s1 and $s2 and put the result in `$t0`.

## 4 Honor Code Statement

Include the Honor Code pledge or just the word "pledged", plus *at least one 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 programming assignments, this should go in the body of the e-mail or in a plain-text file `honor-code.txt` (no word-processor files please).

- 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.)*

## 5 Essay

Include a brief essay (a sentence or two is fine, though you can write as much as you like) telling me what about the assignment you found interesting, difficult, or otherwise noteworthy. For programming assignments, it should go in the body of the e-mail or in a plain-text file `essay.txt` (no word-processor files please).

[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.