CSCI 2321 (Computer Design), Spring 2020

Homework 7

Credit: 30 points.

1 Reading

Be sure you have read, or at least skimmed, Chapter 4 up through section 4.4.

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).

- In this assignment your mission is to trace through what happens during execution of different instructions using the single-cycle implementation of the MIPS architecture as represented in Figure 4.17 in the textbook. You are to assume that at the beginning of a clock cycle the following is true for each of the instructions. (As usual, a value of the form `0xN` denotes a base-16 value of `N`, e.g., `0x10` denotes 16 in base 10.)

  - The program counter (PC) has a value of `0x4`.
  - Location `0x4` in the instruction memory contains the binary representation of the MIPS assembler instruction in question.
  - Register and data-memory contents are as described for individual instructions.

  At the point at which values are written into state elements, what values will the following have?

  - Input and output of the block labeled `PC`.
  - Input and output of instruction memory.
  - Inputs and output of the two adders at the top of the diagram (they don't have names, so let's call them the top-left and top-right adders).
  - All control signals output from the logic block labeled `Control` (`RegDst`, `ALUSrc`, etc.); get these from Figure 4.18.
  - Inputs and outputs of register file (`Read register 1`, `Read data 1`, etc.).
  - Inputs and output of the main ALU. Inputs include control signal `ALU control` (output of the block labeled `ALU control`, from Figure 4.13). Outputs include control signal `Zero`. (Meanings of `ALU control` are given in Figure 4.12.)
  - Inputs and output of data memory.
  - Any values changed in state elements:
    - For the PC, give the new value.
    - If any registers are changed, give their number and new contents.

  * If anything in data memory is changed, give its address and new contents, the address as a hexadecimal number and the new contents as base-10 or hexadecimal.

For 32-bit values it's okay to just give the value in either base 10 or hexadecimal (e.g. for 10 you can either write 10 or `0xa`); for all other values show the binary form with the correct number of bits. Or if a value doesn't make any difference to what is saved into the state elements, just write "not used" (e.g., if `RegWrite` is zero, the values of `Write register` and `Write data` are not used). *RESOURCES* below includes a template file that lists everything you need to fill in, plus the examples from the April 13 lecture. *Note* that in order to determine output of the ALU you have to know what the various values of `ALU control` mean. This information is shown in Figure 4.12, and also in Appendix B.

RESOURCES:

- Template for answers `hw07-template.txt`.
- `add` example `hw07-add.txt`.
- `lw` example `hw07-lw.txt`.
- `beq` example `hw07-beq.txt`.

HINTS:

- My suggestion for how to proceed is to use the approach I use in the examples from the April 13 lecture: First write down what you know (current PC). Then start filling in fields: Output of the PC element is its current value; this feeds into the instruction memory and lets you determine output from the instruction memory (the questions give you its machine-language form). You can now write down inputs and outputs of the top-left adder, some inputs to the register file (the two read registers), and the main control-logic block. From those you can get more values, continuing until everything is filled in.
- I think you will learn more if you try to go strictly by what the circuit (in the figure) does rather than trying to apply what you know about what should happen, though when you're done the results should be consistent with what you think should happen! (E.g., if it's supposed to store n in register r `RegWrite` should be 1, the write register number should be r, and the data input to the register file should be n.

(Yes, this is tedious, but I think it's the only way to really understand how this all works.)

Instructions to trace:

1. (7 points)
   `sub $t2, $t0, $t1`, if `$t0` contains the value `0x6` and `$t1` contains the value `0x2`. (In machine language this is `0x01095022`.)
2. (7 points)
   `lw $t0, 4($s0)`, if `$s0` contains the value `0x10000000`, and the data memory starting at `0x10000000` contains the values `0x1`, `0x2`, `0x3`, and `0x4` (with each value occupying 4 bytes). (In machine language this is `0x8e080004`.)

3. (7 points)

   `sw $t0, 4($s0)`, if `$s0` contains the value `0x10000000`, `$t0` contains the value `0x10`, and the data memory starting at `0x10000000` contains the values `0x1`, `0x2`, `0x3`, and `0x4` (with each value occupying 4 bytes). (In machine language this is `0xae080004`.)

4. (9 points)

   `beq $t0, $t1, LBL`, if `$t0` contains the value `0x2`, `$t1` contains the value `0x1`, and `LBL` corresponds to the instruction at location `0x18` in instruction memory (offset of `0x10` from updated PC). Also say what if anything would change if `$t1` contained the value `0x2` instead of `0x1`. (In machine language this is `0x11090004`.)

3 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.)*

4 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.