ENGR 3410: MP#0
CPU Building Blocks and Assembly Coding

Due: Before Class 7 October 2010

This assignment will develop some basic building blocks for your pseudo-MIPS processor, further refine your skills with Verilog and Verilog simulation, and build some assembly language programming skills. You will also be getting to know your team members in this machine problem. My time estimate is 3.5–6.5 hours per person.

1 Processor components

1.1 The Problem

Construct the following circuits, consulting any outside sources as necessary. (A good one would be one of the appendices of your textbook!) You must define, in text the operation of the following circuits in addition to actually simulating them.

- 2:1 (two-input) multiplexor
- 4:1 (four-input) multiplexor
- 2-bit decoder
- 2-bit decoder with an enable

All logic must be gate level and structural. That means no higher-level Verilog constructs, such as IF, WHILE, or FOR loops. You only have the basic gates, such as AND, OR, NOT, NAND, NOR, XOR. No assign statements (except to set a wire to a constant value), registers, case statements, et cetera. All of these basic gates must have a delay of 50 time units.

You will likely want to check out the digital logic book in reserve to get firm definitions of multiplexors and decoders. Also, as mentioned above, check out Appendix B!

2 Assembly Language Programming

Write a MIPS assembly language program that takes an array of integers and computes (and prints out) the largest number in the array. You may assume the array is not empty, and you may store the array directly in static storage (the `.data` section). Name the file `max.asm`.

To test and debug your program, you'll need a MIPS simulator. The book recommends SPIM/XSPIM, which it discusses in Appendix B:

`http://pages.cs.wisc.edu/ larus/spim.html`

I used the Mars simulator:

`http://courses.missouristate.edu/KenVollmar/MARS`

That web site includes a program that computes the first N Fibonacci numbers. It illustrates how to loop through the elements of an array, how to write a subroutine, and how to print using a system call. You'll want to initialize your array by using the `.word` assembler directive followed by a sequence of comma-separated numbers. This program can be found on the wiki.

3 Turning it in

Designate one person as the submission vehicle. In that person's SVN directory, create a directory called `mp0`. Put the items below and `max.asm` in there. Also include a file called `README.text` that contains:

- The name of your team
- The members of the team
- How long you spent on this assignment
- Anything else you think will help us (a guide to your files if there are a bunch)

I expect a semi-formal, electronic, "lab write-up" of the machine problem. It does not need to be as rigorous as lab notebooks in other, more experimental classes. It should include, at a minimum:

- A brief write-up of the experiments
- Files of all Verilog code — modules and test benches
- Simulation output (textual or waveform) for each circuit

Put your writeup (PDF) and supporting Verilog code, into the `mp0` directory mentioned above.

Other notes:

- All group members must participate in every aspect of this lab.
- Read the Verilog tutorial on the class wiki.
- We'll create a shared team SVN directory after we find out the teams.