CPE 315

Spring 2019
Seng
Laboratory #4
Complete by 11:59pm Monday night (*5/20/19*)

Updates:

- none

Lab Description:

Objectives:

- To build a working MIPS pipelined processor simulator

Lab Description:

In the previous lab, you wrote an emulator which executes MIPS instructions. For this lab, you will add a CPU simulator which will model the flow of instructions through a pipelined processor. Your program will simulate a 5-stage pipeline similar to the pipeline datapath studied in class.

Your processor should accurately simulate the following pipeline delays:

- 3 cycle delay for taken conditional branches
- 1 cycle delay for a use-after-load condition
- 1 cycle delay for any unconditional jump (j, jal, and jr)

Implementation hints:

- In an actual CPU datapath, all of the inter-stage pipeline register are written at the same time. Because this is a program which simulates simultaneous register updates, it is sometimes easier to simulate the pipeline in reverse order:

```
simulate_cpu_cycle() {
        write_back();
        memory();
        execute();
        decode();
        fetch();
}
```

- The emulation of the program can be separate from the simulation of the processor. That is, you can do the emulation in a stage besides the EX stage. For example, you may choose to do program emulation in the IF or ID stage. Because the emulation will not necessarily occur in the EX stage, the register state will not exactly be synchronized with the state of the pipeline registers. That behavior will be fine for the assignment.

- Focus on getting instructions to flow smoothly through the pipeline before working on any of the timing delays.

The simulator will run into either of 2 modes: interactive or script

Interactive mode

In interactive mode, the program should display a prompt and wait for input:

```
mips>
```

The program should accept the following commands (note the differences from Lab 3):

- h = show help
- d = dump register state
- p = show pipeline registers
- s = step through a single clock cycle step (i.e. simulate 1 cycle and stop)
- s *num* = step through *num* clock cycles
- r = run until the program ends and display timing summary
- m *num1 num2* = display data memory from location *num1* to *num2*
- c = clear all registers, memory, and the program counter to 0
- q = exit the program

Example interactive session:

```
mips> p

pc      if/id   id/exe  exe/mem mem/wb
0       empty   empty   empty   empty

mips> s

pc      if/id   id/exe  exe/mem mem/wb
1       addi    empty   empty   empty

mips> s

pc      if/id   id/exe  exe/mem mem/wb
2       add     addi    empty   empty

mips> r

Program complete
CPI = 2.53      Cycles = 253    Instructions = 100
```

Script mode

Your program should take a second optional command line argument which is a script file. This script file will contain a list of commands and will automate testing of your simulator.

Your program should run from the command line with 1 optional argument:

lab4 assembly_file.asm *script_file*

The first argument is an assembly file in a format similar to the previous lab. The second argument is an optional script file.

Sample input/output

lab4_test1.asm lab4_test1.output lab4_test1.script
lab4_test2.asm lab4_test2.output lab4_test2.script
lab4_fib10.asm lab4_fib10.output lab4_fib10.script
lab4_fib20.asm lab4_fib20.output lab4_fib20.script