CprE 381 – Computer Organization and Assembly Level Programming

Homework #7
Assigned: March 7
Due: March 28 (9:00am)

[Note from Joe: This is a somewhat challenging two-week assignment on the design and analysis of pipelined processors. Please use this as a guide to help you gain confidence for some of the problems you will see on Midterm #2 (two of these questions are former Midterm #2 questions). As always, please make use of the Blackboard discussion space to ask questions and to help others as well.]

Reading: Patterson & Hennessy, Sections 4.5-4.8

1) Tracing Pipelined MIPS Code

(a) The following piece of code is executed using the pipeline shown in Figure 4.56 on page 370:

```
lw  $5, 40($2)
add $6, $3, $2
and $8, $4, $3
or  $7, $2, $1
sub $9, $2, $1
```

At cycle 1, right before the instructions are executed, the processor state is as follows:

i. The PC has the value 100_{ten}, the address of the `lw` instruction.
ii. Every register has the initial value 20_{ten} plus the register number (e,g, register `$8` has the initial value of 28_{ten}).
iii. Every memory word accessed as data has the initial value 1000_{ten} plus the byte address of the word (e.g. Memory[8] has the initial value 1008_{ten}).

Determine the value of every field in the four pipeline registers in cycle 5.

(b) Consider executing the following code on the pipelined datapath of Figure 4.60 on page 375:

```
add $2, $3, $1
sub $4, $3, $5
add $5, $3, $7
add $7, $6, $1
add $8, $2, $6
```

At the end of the fifth cycle of execution, which registers are being read and which register will be written?

2) Data Hazards and Forwarding

(a) Identify all of the data dependencies in the following code. Which dependencies are data hazards that will be resolved by forwarding? Which dependencies are data hazards that will cause a stall?

```
add $1, $2, $3
sw $2, 0($1)
lw  $1, 4($2)
add $2, $2, $1
```

(b) With regard to the program in Exercise 1b), explain what the forwarding unit is doing during the fifth cycle of execution. If any comparisons are being made, mention them.

(c) With regard to the program in Exercise 1b), explain what the hazard detection unit is doing during the fifth cycle of execution. If any comparisons are being made, mention them.

3) Optimizing Pipelined MIPS Code

(a) How could we modify the following code to make use of a delayed branch slot?

```
Loop: lw   $2, 100($3)
      addi $3, $3, 4
      beq  $3, $4, Loop
```

(b) The example on page 338 shows how to *maximize* performance on our pipelined datapath with forwards and stalls on a use following a load. Rewrite the following code to *minimize* performance on this datapath – that is, reorder the instructions so that this sequence takes the *most* clock cycles to execute while still obtaining the same result.

```
lw  $2, 100($6)
lw  $3, 200($7)
add $4, $2, $3
add $6, $3, $5
sub $8, $4, $6
lw  $7, 300($8)
beq $7, $8, Loop
```

4) Advanced Data Hazards

(a) Consider an instruction sequence used for a memory-to memory copy:

```
lw $2, 100($5)
sw $2, 200($6)
```

The elaboration starting on page 371 discusses this situation and states that additional forwarding hardware can improve its performance. Show the necessary additions to the datapath of Figure 4.57 to allow code like this to run without stalling. Include forwarding equations (such as the ones appearing on pages 366–369) for all of the control signals for any new or modified multiplexors in your datapath. Finally, rewrite the stall formula on page 372 so that this code sequence won't stall.

(b) Consider the following instructions as they is being executed on a five stage pipelined datapath:

```
lw $1, 40($6)
add $2, $3, $1
add $1, $6, $4
sw $2, 20($4)
and $1, $1,$4
```

i. If there is no forwarding or hazard detection, what `NOP`s would need to be inserted to ensure correct execution?

ii. If the processor has forwarding, but no hazard detection unit, what would happen when this code executes?