CSCE 212: FINAL EXAM **Fall 2009**

Name (please print):_____ **Total points: ___/100**

Instructions
This is a **CLOSED BOOK** and **CLOSED NOTES** exam. However, you may use calculators, scratch paper, and the green MIPS reference card from your textbook. Ask the proctor if you have any questions. Good luck, and have a good winter break!

1. Consider the following code segment:

```
        li $s0,0
        li $s1,0
        li $t0, 16
loop:   lw $s2,vals($s0)
        add $s1,$s1,$s2
        addi $s0, $s0, 4
        blt $s0,$t0,loop
```

Assume this code is to be executed on a pipelined CPU that supports data forwarding and has a fixed three cycle latency for all branches.

a. (10 points) Even with forwarding, this code contains a "load hazard." Describe what this means and how the code could be scheduled (instructions reordered) to compensate for this so no bubbles need to be inserted into the pipeline.

load hazards are data hazards where a register that is written by a load instruction is needed by the next instruction. in this case, forwarding is impossible since the data in question is produced at (the end of) the MEM stage but needed at the beginning of the EX stage.

this occurs between instructions 5 and 4. to compensate, swap instructions 5 and 6.

b. (10 points) Determine how many cycles are required to execute the scheduled code (from part a). Assume each instruction requires one clock cycle except for branches (which require three).

there are three initial instructions, each executed once. there are four instructions in the loop (one is a branch), which it executed four times. this equals 3*1 + (3*1 + 1*3) * 4 = 3 + 24 = 27 cycles. note that the load didn't require a delay since we scheduled the code.

c. (10 points) This code contains a loop with a fixed iteration count of four. Re-write the code and "unroll" the loop. In other words, remove the branch instruction and replicate the body of the loop four times. To maximize performance, you may, if you choose, adjust the way the array is indexed to eliminate the need for counter in $s0.

(next page)

```
li $s1,0

lw $s2,vals
add $s1,$s1,$s2

lw $s2,vals+4
add $s1,$s1,$s2

lw $s2,vals+8
add $s1,$s1,$s2

lw $s2,vals+12
add $s1,$s1,$s2
```

d. (10 points) After unrolling the loop, determine the new cycle count and calculate the speedup over the original loop from part b.

9 instructions, 9 cycles, speedup = 27/9 = 3

2. (10 points) Describe the advantage of floating-point representation over fixed-point representation. Describe why it is more expensive (in terms of hardware resources and time) to perform floating-point arithmetic as opposed to fixed-point arithmetic.

floating-point has better range.

it's more expensive because the arithmetic hardware must treat the various fields within the floating-point representation with special care, as opposed to simply sending the input values directly into a basic adder, multiplier, or divider circuit.

3. (10 points) Assume I can afford to build a cache that contains 4096 bytes to hold only the data contents (i.e. not including the tags or valid bits). Further assume that the cache is direct-mapped and has a block size of 32 words. How many address bits will be used for the index field?

there are:
4096 bytes / (32 words/line * 4 bytes/word) = 4096/128 = 32 lines in the cache
$\log_2 32 = 5$ address bits for the index field

4. (10 points) Assume the MIPS instruction set didn't contain any shift instructions (but is otherwise the same), and the assembler implements SLL, SRL, and SRA as pseudo-instructions. Show how the following pseudo-instruction could be translated:

```
sra $s0, $s0, 8
```

Hint: think about the relationship between shifts and arithmetic.

```
li $1,256              # 2^8
div $s0,$s0,$1
```

5. (10 points) Describe control hazards. What causes them? What is their effect relative to performance? Describe at least one method for reducing their effect and why it is effective.

they are caused by the fact that branch instructions cannot be resolved until they reach later stages of the pipeline, making it uncertain which instruction to fetch immediately after the branch instruction. they usually cause stalls (bubbles, nops) to be inserted immediately after each branch, causing the CPI for branch instructions to be > 1. one way to deal with them is branch prediction, where the next instruction is predicted and cleared if the prediction turns out to be bad.

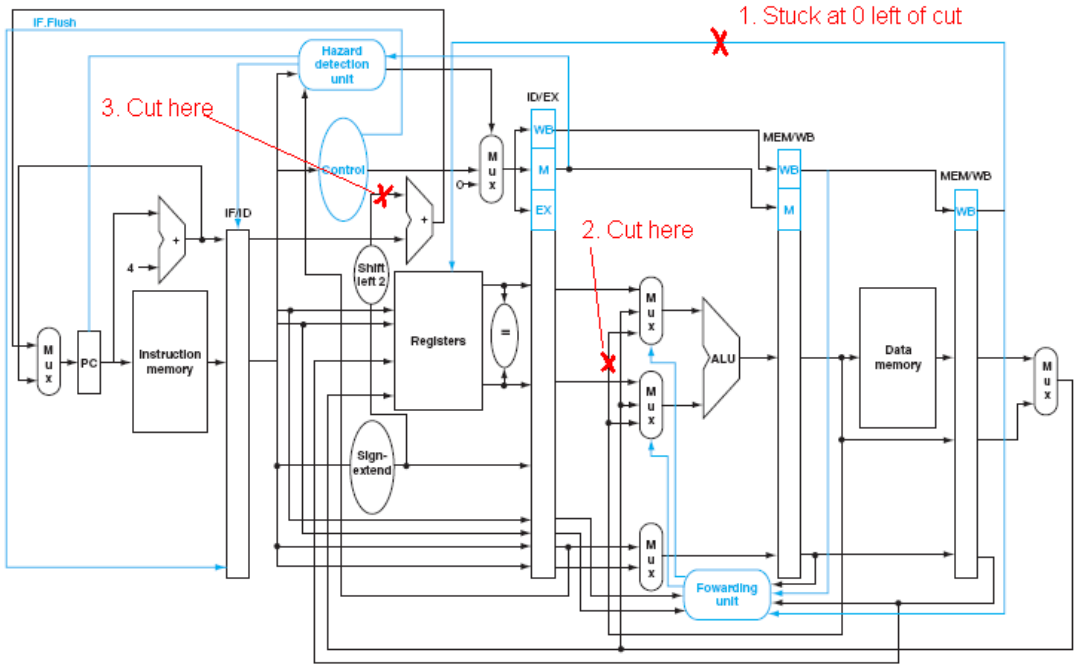
6. (10 points) Describe the disadvantage of a processor design that executes every instruction in a single clock cycle.

the cycle time must be adjusted to be long enough to accommodate the slowest instruction, which penalizes faster instructions.

7. (10 points) Write a MIPS subroutine called **`three_to`** that takes an argument *n* in register $a0, calculates the value of three raised to the power of the argument (i.e. 3^n), and returns the result in register $v0. $a0 must be >= 0. The subroutine should save and restore all the caller's registers on the stack. Hint: the subroutine requires a loop.

```
three_to:   addi $sp,$sp,-8
            sw $a0,0($sp)          # push $a0 on the stack
            sw $t0,4($sp)          # push $t0 on the stack
            li $v0,1
loop:       bgtz $a0,skip
            j exit
skip:       move $t0,$v0           # save original $v0
            add $v0,$v0,$v0        # double $v0
            add $v0,$t0,$v0        # add in original value (mult. by 3)
            addi $a0,$a0,-1        # decrement counter
            j loop
exit:       lw $t0,4($sp)          # pop $t0
            lw $a0,0($sp)          # pop $a0
            addi $sp,$sp,4
            jr $ra                 # return
```

8. (10 points EXTRA CREDIT) **BONUS:** For the MIPS datapath shown below, several lines are marked with "X". For each one:
   - Describe in words the negative consequence of cutting this line relative to the working,unmodified processor.
   - Provide a snippet of code that will fail
   - Provide a snippet of code that will still work



1) no registers can ever be written back to the register file
2) no values can ever be forwarded from MEM to EX for rs
3) branch targets will always be wrong (taken branches will go to garbage addresses)