Comprehensive Examination
Hardware and Architecture
Spring 2003

Problem 1 (20 points). Combinational Components

Please answer the questions below. No explanation is required!

1. In general, if a decoder has n input hits and one enable bit; how many output bits does it have?

2. In general; if a selector or multiplexor has n input bits and m output bits, how many selector bits does it have?

3. How many levels of 2-to-1 selectors would be required to build a 32 bit barrel shifter?

4. What is the smallest number of 2-to-t selectors required to build this 32 bit barrel shifter?

5. An 8 bit adder/subtractor can be built with ONLY the following components: 8 full adders and 8 (circle one below)
   - bit magnitude comparator
   - XOR gates
   - AND gates
   - 1-to-2 decoders

6. We would like to build a 3-to-8 decoder using smaller components. If we can only use 1-to-2 decoders, what is the smallest number of 1-to-2 decoders required?

7. We would like to build a 3-to-8 decoder using smaller components. If we can also use 2-to-4 decoders, how many of each type are required if we are trying to minimize the number of decoders we need to use?

Problem 2 (20 points)

Consider the following code:

```
br1     for (i=O;i<N;i++) {
br2     if (A[i] > 1) {
           …
        }
br3     if (B[i] <= 1) {
           …
        }
br4     if (A[i] > 1 && B[i] <= 1) {
           …
        }
      }
```

Assume that A[] and B[] contain floating point numbers randomly distributed between 0 and 2. Also. assume that N is very large. Consider all four branches (the for loop conditional branch, and the three "if" branches). What is the branch misprediction rate **for each branch** in this loop using the branch prediction architectures below? Assume the history based predictors index into a table of 2-hit saturating up and down counters. Assume all branches update the history registers with correct branch directions before the next branch is predicted. Assume no aliasing in the 2-bit table between the 4 branches.

(a) Draw the 2-bit saturating up and clown predictor, labeling the state: transitions, and the prediction provided for each state.

(b) Assume a completely local predictor that uses a large local history found using the PC to index into the table of 2-bit conters.

(c) Assume a global (correlating) predicator that, uses a global history register as the index to the PHT.

Problem 3 (20 points)

This problem examines performance differences between physically awl virtually indexed acid tagged Data Caches and their interaction with the Translation Lookaside Buffer (TLB). Assume that the data TLB miss rate is 10% and the data cache miss rate is 20%. The cache miss penalty is 40 cycles. A TLB miss takes 50 cycles to process a miss. Assume that it takes 1 full cycle to access the cache acid also 1 full cycle to access the TLB.

For each problem below, draw the relationship /design of the Data Cache, TLB, and Main Memory, specifying the order in which they are accessed.

(a) What is the Average Memory Access Time in cycles for a, physical indexed and physical tagged cache?

(b) What is the Average Memory Access Time in cycles for a virtual indexed and physical tagged cache?

(c) What is the Average Memory Access Time in cycles for a virtual indexed awl virtual tagged cache'?

if	id	ex1	ex2	ml	m2	wb

Problem 4 (20 points)

Assume a second-generation MIPS processor is superpipelined (meaning the longer stages are cut in half, reducing the cycle time), as shown. Registers are read in stage *id* and written in *wb*. There are two execution stages, *ex1* and *ex2*. All ALU operations start at *ex1*, but arithmetic operations (add, sub, etc.) complete the computation at the end of *ex2*, while logical operations (e.g., and, or, shift) produce the result by the end of *ex1*. Memory (cache) accesses take two cycles (stages *ml* and *m2*), with the result (on a load) being available at the end of *m2*. Address calculation begins in *ex1*. Assume forwarding sufficient to minimize execution time whenever forwarding is necessary. (If you need to make other assumptions, make sure they are reasonable ones, and clearly stated). Show the pipelined execution of the following instructions, with forwarding and bubbles all shown.

add $5, $4, $1	if	id	ex 1	ex2	ml	m2	wb	
lw $12, 1000($5)		if	id					
and $6, $6, $4								
or $10, $6, $5'								
add $8, $6, $12								
sw $8, 2000($10)								

5. (20 points) Dynamic Scheduling
For each of the following codes, give me the *approximate* speedup expected from a reservation-station (Tomasulo) style of architecture, over a scoreboard style (which has no support for hardware register renaming). A scoreboard architecture resolves WAW hazards before issue, and WAR hazards before commit. Assume integer add/sub instruction latency of 1 cycle, load latency of two cycles, and integer multiply latency of 12 cycles (by latency of *L*, I mean that a dependent instruction, barring other hazards, can begin execution *L* cycles after the producing instruction begins). Assume perfect branch prediction, and that the application is simply one long sequence of this loop. Please examine the code *carefully*. Round speedups to the nearest power of 2. Assume no branch delay slot.

```
loop:   addi $6, $4, #8
        lw $4,1000($6)
        add $7,$6,$8
        mul $5, $2, $6
        sub $12,$3,$4
        sw $5, 3000($12)
        beq $6, $0, loop
```

speedup (tomasulo over scoreboard) assuming a scalar architecture

max speedup (tomasulo/scoreboard) on wide superscalar architecture

```
loop:   addi $6, $6, #8
        lw $4,1000($6)
        add $7,$4,$8
        mul $5, $5, $6
        sub $12,$3,$4
        sw $5, 3000($12)
        beq $6, $0, loop
```

speedup on scalar architecture

max speedup on wide superscalar architecture

```
loop:   addi $6, $6, #8
        lw $4, 1000($7)
        add $7,$4,$8
        sub $12,$3,$7
        mul $7,$2,$6
        sw $7,3000($12)
        beq $6, $0, loop
```

speedup on scalar architecture

max speedup on wide superscalar architecture