

CS4760 Homework 1: Basic Pipelining, Tomasulo's Algorithm, Scoreboarding and Dynamic Branch Prediction

Due Wednesday, October 23, 1998 in class (12:15pm) (No late homework accepted)

For this assignment and for all other homework assignments, you may collaborate with other students, but the work you turn in must be your own. If you work with other students, list their names along with your own.

Show all work to receive full credit on answers.

1. Look at the instruction sequence in Problem 3.1 in the book. Identify all WAR, WAW and RAW dependencies in the instruction stream. (Some of these types do not occur in the DLX pipeline, but identify all the dependencies anyway.)
2. Problem 3.1 in book.
3. Problem 3.14 in book.
4. Dynamic Instruction Scheduling

(a) Identify all WAR, WAW and RAW dependencies in the following instruction sequence. (Pay careful attention to the load LD and store SD instructions.):

```
LD F2, 16(R6)
ADDD F2, F2, F4
DIVD F6, F2, F0
SUBD F0, F2, F10
SD F6, 32(R3)
```

(b) Fill in the blank templates for the Scoreboard AND for Tomasulo's Algorithm for this instruction sequence.

(The templates are in the Handouts section of the class web page.)

Number the cycles at which each phase occurs.

For scoreboarding, assume one integer unit, two floating point multiply units, one F.P. add unit, one F.P. divide unit. (Same units as in book example)

For Tomasulo's, assume three FP ADD units, 2 FP MULT units, 6 load buffers and three store buffers. (Same units as in book example)

Assume there is a cache miss causing a stall of 8 cycles on the execution of the LD.

Assume FP adds/subs take 2 cycles, Mults take 10 cycles and Divides take 20 cycles.

Assume the store is a cache hit and executes in one cycle.

Assume many instructions can read from the register file simultaneously.

For the Tomasulo example, recall that only one instruction can drive the CDB at a time.

*** For the RAW, WAR and WAW hazards identified in part (a), explain how each is resolved in the Scoreboarding and Tomasulo algorithms.

5. Consider the following code. (The marks indicate instructions that are ignored in this example)

```
LOOP1:  ADDI R4, R0, #4
        .....
LOOP 2:  SUBI R4, R4, #1
        .....
        BNEZ R4, LOOP2
        .....
        BEQZ R8, LOOP1
        .....
```

- (a) Focusing on the inner loop (LOOP2) only, analyze the branch behavior. Assume no other instruction changes the value of register R4. What percentage of the time is the BNEZ branch instruction taken and not taken?
- (b) Choose the best static branch prediction scheme for the BNEZ instruction. What percentage of the time will this static branch prediction be correct for LOOP2?
- (c) Now consider dynamic branch prediction. Draw the state machine for a one-bit branch predictor. Be sure to clearly identify or define the meaning of each state. For the inner loop (LOOP2), what will be the misprediction rate of the one-bit branch predictor?
- (d) Now draw the state diagram for a 2-bit dynamic branch predictor. Again, clearly label all states. What will be the misprediction rate of the 2-bit branch predictor for LOOP2?
6. (a) Draw and label the fields of a branch history buffer with 32 entries that uses a 2-bit branch prediction scheme.
- (b) In which pipeline stage do we consult the branch history buffer?
- (c) Show an entry in the buffer for a branch with PC=60, targetPC=84, and the following history: Taken Taken Not Taken Taken Not Taken
7. (a) Draw and label the fields of a branch target buffer with 32 entries that uses a 2-bit branch prediction scheme.
- (b) In what pipeline stage do we consult the branch target buffer?
- (c) Show an entry in the buffer for a branch with PC=36, targetPC=64, and the following history: Not Taken Not Taken Taken Taken Not Taken Taken