EECS 470 Winter 2024

Homework 2

Due Friday February 2nd at 10pm. Half credit if late and turned in by noon on 2/3

This is an individual assignment; all of the work should be your own. **Assignments that difficult to read will lose *at least* 50% of the possible points and we may not grade them at all. This assignment is worth a bit less than 2% of your grade in the class and is graded out of 100 points. You may drop one homework assignment or quiz.**

---

1) In standard written English, explain what conditions must be true in order for Tomasulo's algorithm (the algorithm found in sections 3.4 & 3.5 of our book) to do the following. (i.e. what conditions would have to be true for these actions to be done?) **[12]**
   a) A given instruction has its registers renamed (both source and destination).
   b) A given instruction is sent to a reservation station.
   c) A given instruction is sent to its execution unit.
   d) A given instruction is allowed to leave its execution unit.

2) You are given a standard 5-stage pipeline which is also 2-way superscalar. Branches are resolved in the memory stage and predicted not-taken. 20% of the instructions are branches and 45% of those are not taken. Assume the *only* source of stalling (that is the only thing that prevents an IPC of 2) is branches. What would be the IPC of the processor? (hint, think about what it means for 2 instructions to be in the same stage at the same time with respect to program order!) Show and explain your work. **[10]**

3) Consider the following assembly instructions:

```
start:  R1=MEM[R2+0]
        R2=MEM[R2+4]
        branch done if(R2==0)
        R3=R1+R3
        branch start
done:   halt
```

   a) If all entries in the rename table currently point to the appropriate architected register, and if the next available physical register is P1 (then P2, P3, etc.) show how each assembly command executed will be translated if the above program branches to done after the 3rd time R2 is changed. Assume branches are predicted correctly. You will need to "translate" many of the commands more than once. You may assume there are an unlimited number of physical registers. Your answer should be similar to what is shown on page 48 of lecture 3. **[6]**
   b) Describe, using "C/C++" terms, what the above program seems to be doing (think structures and pointers…) **[3]**.

4) Explain the following sentence from McFarling's "Combining Branch Predictors" paper **[7]**

   As we would expect, gselect-best performs better than either bimodal or global prediction since both are essentially degenerate cases.

5) Consider a local pattern history predictor where The BHT has 32 entries, each with 3 bits of history. The predictors in the PHT are each 1 bit. What steady-state prediction rate would this predictor get on a branch that had the following pattern? You can assume that there are no other branches. **[6]**
   a) 3 taken, 1 not taken repeating forever (T,T,T,NT,T,T,T,NT, etc.)
   b) 4 taken, 1 not taken repeating forever (T,T,T,T,NT,T,T,T,T,NT, etc.)
   c) 5 taken, 1 not taken repeating forever (T,T,T,T,T,NT,T,T,T,T,T,NT, etc.)

Code for problems 6 and 7.
Consider the following code sequence (destination register specified last):

```
A:    LD 0(R1),F2
B:    MULTD F0,F2,F4
C:    ADDD F2,F4,F6
D:    LD 8(R1),F4
E:    MULTD F0,F4,F10
F:    ADDD F2,F2,F2
G:    ADDD F0,F2,F2
H:    SD 8(R1),F2
I:    ADDDI F0, #16, F0
J:    DIVDI F10, #3, F12
K:    BNEQ F6,F12,A
```

Functional units & latencies:	
1 Load	2
1 Store	1
1 Branch	1
1 FP Adder	2
1 FP Multiplier	5
1 FP Divider	10

6) Determine how long the loop takes to complete on an in-order machine where an instruction may not issue (start execution) until the preceding instruction is writing back its result, whether they are dependent or not. Create a table showing when each instruction *finishes* fetch/decode, execute, and writeback. Report the total number of cycles per iteration. We have filled in the first two rows of the table below. The number in parenthesis is just the number of cycles the instruction takes. **[15]**

Instruction	Fetch/Decode (FD)	Execute (X)	Writeback (W)
`A:    LD 0(R1),F2 (2)`	1	3	4
`B:    MULTD F0,F2,F4 (5)`	4	9	10

7)

a) Now, draw a dependence graph, where the vertices correspond to instructions and the edges correspond to true control and data dependences among the instructions. (Because this graph shows the data flow of an instruction sequence, it is sometimes called a dataflow graph). Annotate each instruction with its latency **[10]**

b) Imagine an out-of-order processor with infinite issue width, an unbounded number of functional units and CDBs, and with ideal renaming implemented (that is all false dependencies are eliminated). Instructions may not start execution until any result it is dependent (truly dependent) on has completed writeback. Your answer should include a table that looks something like this.

Instruction	Has all data	Start EX	Finish EX	Writeback
`A:    LD 0(R1),F2 (2)`	1	2	3	4
`B:    MULTD F0,F2,F4 (5)`	4	5	9	10

List the instructions in the order they have all their data, breaking ties by placing those earlier in program order above those later in program order. **[16]**

8) We claim that for the original version of Tomasulo's algorithm that we need to wait to issue a branch to a functional unit until all older instructions have completed execution.

a) In your own words, explain what problem we are solving by adding this restriction. Be clear and concise. **[3]**

b) Explain how this restriction limits the ILP available to the processor. **[3]**

c) In class we've claimed that loads can cause an exception to occur. In your own words, explain what an exception is and give an example of when a load would cause one. **[3]**

d) In your own words, explain why the loads would need that same restriction as branches. **[2]**

e) Explain how adding a reorder buffer eliminates this restriction. Be clear and detailed. **[4]**