CS 352 – Prof. Mark **Assignment #6** **Due April 15, 2004**

Problem #1: Out-of-order execution (15 points)

a) What is the purpose of a re-order (commit) buffer in an out-of-order processor?

b) Describe two reasons why a compiler cannot eliminate all WAW hazards, and how register renaming is able to eliminate them at runtime.

Problem #2: Branch prediction (20 points)

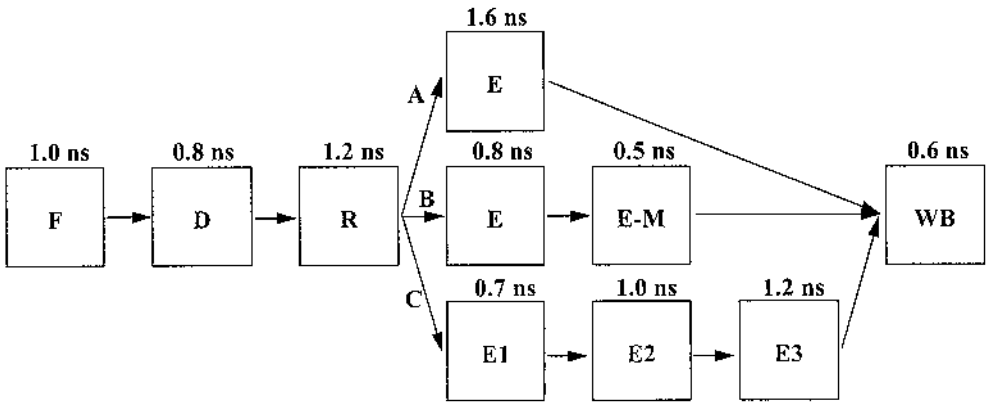
a) What is the purpose of the return address stack and why would a modern processor include it in addition to a branch target buffer?

b) When should the branch predictor be updated: speculatively as soon as a branch prediction is made, or later after the branch outcome has been determined? Consider what happens when you have back-to-back branches in a program.

Problem #3: Multiple execution units (35 points)

In the pipeline below, all instructions use execution pipeline A, except loads and stores which use pipeline B and multiples which use pipeline C. Assume that loads and stores are 28% of all instructions, and multiplies are 15% of all instructions. Furthermore, assume that only one instruction can be in *any* of the execute stages at a time (i.e. no parallel or pipelined execution). Also assume perfect bypassing. Please answer the following questions (35 points):

a) Assume that the time shown for each stage represents the combinational logic only. If each latch is 100 picoseconds, what will be the clock speed of the machine?
b) What will be the CPI of the machine?
c) What is the MIPS rating of the machine, for the instruction mix above?
d) Assume that we split the pipeline A "execute" stage into two stages, each with 0.8 nsec delay. What will be the new cycle time and MIPS rating of the machine?
e) Recompute the MIPS ragins for parts (c) and (d) assuming that each latch is 1000 picoseconds. Does your answer change? Why?



Problem #4: Dependencies and out-of-order execution (45 points)

Examine the following code sample, and produce the following four answers, using the pipeline from problem 15:

a) Draw the dataflow graph of the code, showing only true data dependences.
b) Generate a list of all dependences (RAW, WAW and WAR) using the (1-2) notation to express each dependence (that notation means that instruction 2 is dependent on instruction 1). Hint: The RAW dependences should match the graph edges from part (a).
c) Using the figure below, fill out the pipeline timing chart. Assume that there are bypass paths from both writeback and the end of each execute pipe to the register read (R) stage. Instructions may be in the different execute pipelines simultaneously but must be written back in program order. Also, please note that the pipeline stage names are labeled for pipeline C, but instructions in piepeline A will be written back in stage 5, and pipeline B in stage 6. How many cycles does it take the code fragemtne to execute?
d) Rewrite the code sample with all of the architectural registers renamed to physical registers, starting from P0 and counting upward (P1, P2, etc.) Only rename those registers whose values are created (i.e. written) in the code fragment.

1: MULT R1, R2, R3
2: ADD R4, R1, R1
3: LW R5, (4) R4
4: ADDI R5, R5, #1
5: SW R1, (4) R5
6: ADD R6, R5, R1

	F	D	R	E	E	E	WB
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							