

CS 6290 Homework 2

Fall 2007

Due: 9/25 at start of class

There are two problems in this homework. Each problem is worth 10 points. You should turn in your solutions one problem per page, in-order (problem 1 first, then problem 2, etc.). Show your work. No points will be given for providing only a numeric answer without showing or explaining (briefly, please) how you got the answer.

Problem 1:

A 2-wide processor has 4 unified reservation stations (numbered 0 through 3), an 8 ROB entries (numbered 0 through 7), and a load-store queue with 3 entries (numbered 0 through 2). The pipeline of the processor allows two instructions per cycle to be fetched in one cycle, decoded in the next cycle, and (if resources permit) issued in the cycle after that. The processor has three fully pipelined execution units: one ALU, one load/store, and one multiply/divide unit. The ALU can execute add, sub, not, or, and and, but not mul/div instructions. Each unit can begin execution of one instruction in each cycle, so the processor can begin to execute 3 instructions in a cycle if instructions of the right kinds of are ready to execute. Instruction execution takes 1, 2, and 4 cycles, for integer, load/store, and floating point, respectively. There are two CDBs, and each instruction writes the result on the CDB in the cycle that follows the one in which it completes execution. An instruction becomes eligible for execution in the cycle that follows the one in which its last operand was broadcast on the CDB. An instruction becomes eligible for commit (if other constraints permit it) in the cycle after it has broadcast its results to the CDB. The processor can commit up to two instructions per cycle.

Assuming that no instruction has yet been fetched at the beginning, for each instruction determine the cycle in which it is fetched, decoded, issued, begins executing, uses the CDB, and commits. For instruction issue, also specify which reservation station, load/store queue entry, and ROB entry the instruction is using. The program executed by the processor is as follows:

Instruction	T_{Fet}	T_{Dec}	T_{Iss}	RS	LSQ	ROB	T_{Exe}	T_{CDB}	T_{Com}
addi v0, a2, 0x3									
mulv v0,v0,0x03									
sub sp, sp, v0									
addi a2, sp, 24									
sub sp, sp, v0									
addi a3, sp, 24									
sub sp, sp, v0									
addi t0, sp, 24									
sub sp, sp, v0									
lw v1, -32740(sp)									
addi a0, sp, 24									
sub sp, sp, v0									
lw v0, -32740(sp)									
sw a3, 18732(v1)									
lw v1, -32740(sp)									
sw a2, 18728(v0)									
lw v0, -32740(sp)									
sw a0, 18740(v1)									
lw a0, -32740(sp)									
sw t0, 18724(v0)									
lw v0, -32740(sp)									
lw t9, -28608(sp)									
addi a1, sp, 24									
addi a0, a0, 18804									
addi a2, zero, 124									
sw a1, 18736(v0)									
move a1, zero									

Problem 2:

A processor encounters the sequence of conditional branches given below. For each branch, we list its address (the address of the branch instruction) and the direction (taken or not taken).

Part A: We are using an 4-entry predictor with 2-bit counters as entries, and all entries start off with a value of zero (strong not taken). For each branch show: which entry is used to predict it, the prediction, whether the prediction is correct, and the state of the entire 4-entry predictor (value of each entry) after the branch in question updates the predictor. The predictor entry is selected using the lowermost 2 bits of the branch address.

Address	T/NT	Entry	Prediction	Correct?	Predictor State			
0x4652a8	NT	0	NT	Yes	0	0	0	0
0x4652b1	NT							
0x4652b3	T							
0x4652c1	NT							
0x4652c2	NT							
0x465300	T							
0x4652a8	NT							
0x4652b1	NT							
0x4652b3	NT							
0x4652c1	T							
0x4652c2	NT							
0x465300	T							
0x4652a8	NT							
0x4652b1	T							
0x4652b3	T							
0x4652c1	T							
0x4652c2	NT							
0x465300	T							
0x4652a8	T							
0x4652b3	NT							
0x4652a8	NT							
0x4652b1	T							
0x4652c1	T							
0x4652c2	NT							
0x465300	T							

Part B: We are using an 8-entry predictor, with a 1-bit history and two 1-bit counters in each entry (the first is for when the history bit is zero, the second is for when the history bit is one). All entries start off as all-zeroes: history is zero (not taken), and both 1-bit counters are zero (not taken). For each branch, show: the prediction, whether the prediction is correct, and the state of the entire 8-entry predictor (content of each entry: history, first counter, second counter) after the branch in question updates the predictor. Also, in the predictor state circle the 1-bit counter that was considered. The predictor entry is selected using the lowermost 3 bits of the branch address.

Address	T/NT	Pred?	Corr?	Predictor State								
0x4652a8	NT	NT	Yes	000	000	000	000	000	000	000	000	
0x4652b1	NT											
0x4652b3	T											
0x4652c1	NT											
0x4652c2	NT											
0x465300	T											
0x4652a8	NT											
0x4652b1	NT											
0x4652b3	NT											
0x4652c1	T											
0x4652c2	NT											
0x465300	T											
0x4652a8	NT											
0x4652b1	T											
0x4652b3	T											
0x4652c1	T											
0x4652c2	NT											
0x465300	T											
0x4652a8	T											
0x4652b3	NT											
0x4652a8	NT											
0x4652b1	T											
0x4652c1	T											
0x4652c2	NT											
0x465300	T											