

CS352 – Assignment #6

Weight: 50 points

Due date: Monday, Oct 20, 2008 (beginning of class)

1. We have a program executing on a pipelined processor as described in Chapter 6 of the textbook. We have measured the relative frequencies for the instruction groups to be as shown in the following table:

Instr. grp	Rel. freq.
Load	0.26
Store	0.12
Arith	0.51
Branch	0.07
Jump	0.02
Other	0.02

The base CPI, with no stalls, for this pipeline is 1.0

- a. This program executes on a processor with a clock rate of 1.6GHz, and the number of instructions executed is 1.75×10^9 . If we assume that the CPI is 1.0, what is the estimated execution time for this program?
- b. Now assume that 30% of the load instructions are immediately followed by an arithmetic instruction that uses the loaded data item. Each of these causes a 1 cycle load-use stall cycle. Give the average CPI for the program on this processor if these load-use stalls are counted.
- c. The predict-not-taken strategy for handling conditional branches is being used. With this strategy, every taken branch and every jump causes a 1-cycle stall. If 75% of the branches are taken, give the average CPI for the pipelined processor if only the brch/jmp stalls are counted.
- d. What is the average CPI if both the load-use and the brch/jmp stalls are counted?
- e. How long will the program and processor from part (a) take if the avg. CPI from part (d) is used?
- f. Someone suggests using the delayed-branch strategy for dealing with brch/jmp instructions (instead of the predict-not-taken strategy). They estimate that 65% of the delay slots will be filled with useful instructions (instead of with NOP instructions). What is the average CPI for the processor if only the wasted cycles for the delayed-branch strategy?
- g. If we count the load-use stalls and the wasted delayed-branch cycles, what is the average CPI? How long will the program from part (a) take with this average CPI?

We have the following instructions, registers and memory contents, give the values in the pipeline registers as each instruction is processed. Show any forwarding and/or stall cycles. Give the register values after each instruction completes.

lw \$9, 0(\$16) \$8 = 0x00000000 \$16 = 0x00001000 mem[0x00001000] = 0x00000008
 add \$10, \$17, \$9 \$9 = 0x00001111 \$17 = 0x00000004 mem[0x00001004] = 0x00000020
 add \$18, \$18, \$9 \$10 = 0x00002222 \$18 = 0x00000008 mem[0x00001008] = 0x00000030

Fill in the following diagram, giving the contents of the inter-stage registers for each cycle; use data forwarding if necessary; if there is are stalls, insert NOPs in the correct cycles:

Cycle	1	2	3	4	5	6	7	8
IF/ID.op	_____	_____	_____	_____	_____	_____	_____	_____
IF/ID.rs	_____	_____	_____	_____	_____	_____	_____	_____
IF/ID.rt	_____	_____	_____	_____	_____	_____	_____	_____
IF/ID.rd	_____	_____	_____	_____	_____	_____	_____	_____
IF/ID.imm	_____	_____	_____	_____	_____	_____	_____	_____
ID/EX.op	_____	_____	_____	_____	_____	_____	_____	_____
ID/EX.rs	_____	_____	_____	_____	_____	_____	_____	_____
ID/EX.rt	_____	_____	_____	_____	_____	_____	_____	_____
ID/EX.rd	_____	_____	_____	_____	_____	_____	_____	_____
ID/EX.se(im)	_____	_____	_____	_____	_____	_____	_____	_____
ID/EX.A	_____	_____	_____	_____	_____	_____	_____	_____
ID/EX.B	_____	_____	_____	_____	_____	_____	_____	_____
EX/ME.op	_____	_____	_____	_____	_____	_____	_____	_____
EX/ME.dst	_____	_____	_____	_____	_____	_____	_____	_____
EX/ME.B	_____	_____	_____	_____	_____	_____	_____	_____
EX/ME.alu	_____	_____	_____	_____	_____	_____	_____	_____
ME/WB.op	_____	_____	_____	_____	_____	_____	_____	_____
ME/WB.dst	_____	_____	_____	_____	_____	_____	_____	_____
ME/WB.alu	_____	_____	_____	_____	_____	_____	_____	_____
ME/WB.mdr	_____	_____	_____	_____	_____	_____	_____	_____

\$9 =

\$10 =

\$18 =