

Chapter 3 Homework

Due October 22, 1998

Question 1 (40 points): Pipeline Hazards

Exercise 3.3 p. 216

Question 2 (30 points): Pipeline Branch Performance

You are analyzing the pipeline for a CISC processor, which uses the following pipeline structure:

IF1 IF2 ID AG MEM1 MEM2 EX WB

where

IF1: Instruction fetch, part 1; increment the PC
IF2: Instruction fetch, part 2
ID: Instruction decode, register fetch
AG: Address generation for loads, stores, and branch targets
MEM1: Operand access, part 1 (send address to memory)
MEM2: Operand access, part 2 (do the read or write)
EX ALU operation and condition code setting
WB register file writeback

Conditional branch instructions test the global condition code, which is set in the EX stage of some previous instruction.

To simplify the estimation of pipeline branch performance, we assume that there are no structural or data hazards.

a) Fill in the following table showing the number of stall cycles for three different pipeline branch strategies. Assume that the instruction which sets the condition code immediately precedes the branch instruction.

<u>Conditional branch strategy</u>	Conditional branch taken	Conditional branch not taken	Unconditional branch (ie, jump)
Stall until everything is known			
assume not taken			
assume taken			

b) Assume that 20% of all instructions are conditional branches, of which 60% actually go to the target. Assume 10% of all instructions of unconditional branches. What is the CPI for each of the three branch strategies?

c) Performance can be improved by having the compiler schedule instructions so that the branch and the preceding instruction which sets the condition code are separated by one or more instructions that do not set the condition code. How many such instructions would need to be inserted for maximum performance when using the best method from part (b)? Why?

d) But the compiler cannot do optimal instruction rescheduling. It can separate conditional branches from the instruction which sets the condition code only by the following amounts:

Number of separating instructions	Percentage of conditional branches
0	20%
1	40%
2	30%
3	10%
<hr/>	
Total	100%

With this instruction-rescheduling compiler, what is the new CPI for the branch strategy which was best in part (b)? What percentage performance improvement did this new software gain us?

Question 3 (30 points): Pipeline Performance

In this problem, we will be looking at a machine that we will call the "Long Word DLX". This machine executes "bundles" of two data-independent DLX instructions in every cycle. The bundles have fixed lengths of 8 bytes (64 bits) and there are a few restrictions that the compiler must follow when compiling for this machine:

- 1) Only one control flow instruction may be executed in a bundle.
- 2) Instructions in the same bundle must not have any data dependencies.

The proposed pipeline for this machine is as follows:

b	IF	ID/RF ID/RF	EX EX	Mem Mem	WB WB				
b+1		IF	ID/RF ID/RF	EX EX	Mem Mem	WB WB			
b+2			IF	ID/RF ID/RF	EX EX	Mem Mem	WB WB		
b+3				IF	ID/RF ID/RF	EX EX	Mem Mem	WB WB	
b+4					IF	ID/RF ID/RF	EX EX	Mem Mem	WB WB

- a) Can this machine execute code that has been compiled for DLX? Explain.
- b) How many and what types of Memory and register ports will be needed to avoid structural hazards?
- c) After compiling the benchmarks on the Long word DLX and executing them, we find that 20% of the dynamically executed bundles contain one NOP. What is the percentage increase in executed code size?
- d) On the Long Word DLX, a register that is written in the current cycle will not be able to read the updated value until the next cycle.

List all the physical forwarding paths needed to minimize the stalls. For each forwarding path, state the number of register index comparators that will be needed.

e) The following is a table that gives the dynamic breakdown of the instruction types in a bundle. The totals on the right hand side represent the percentage of the second instruction in the bundle. The totals on the bottom represent the percentage of the first instruction in the bundle.

		First Instruction in bundle				<i>Total</i>
		Branch	Load/store	ALU	NOP	
Second Instruction in Bundle	Branch	-	-	25%	5%	<i>30%</i>
	Load/Store	-	5%	10%	-	<i>15%</i>
	ALU	-	15%	25%	-	<i>40%</i>
	NOP	-	10%	5%	-	<i>15%</i>
	<i>Total</i>	-	<i>30%</i>	<i>65%</i>	<i>5%</i>	<i>100%</i>

What percentage of the dynamically executed instructions are Branch, Load/Store, ALU, NOPS?

f) The load and branch delay for this machine are the same as the DLX in terms of cycles with a predict not-taken scheme.

Assuming that 10% of the bundles that have memory operations are followed by a bundle that uses a loaded value from that operation, and that 40% of branching instructions are taken, what is the average CPB (cycles per bundle) for this machine?

g) Using your calculation for CPB, what is the CPI of this machine assuming that NOPs are not useful instructions?

