CSE 30321 – Computer Architecture I – Fall 2008
Review Session 2 Problems
September 9, 2008

Question 1:

Let's assume that we have a CPU that executes the following mix of instructions:

- 43% are ALU operations (i.e. adds, subtracts, etc.) that take 1 CC
- 21% are Load instructions (i.e. that bring data from memory to a register) that take 1 CC
- 12% are Store instructions (i.e. that write data in a register to memory) that take 2 CCs
- 24% are Jump instructions (i.e. that help to implement conditionals, etc.) that take 2 CCs

What happens if we implement 1 CC stores at the expense of a 15% slower clock?

Is this change a good idea?

- To answer, let's assume our instruction count does not change: (Call it P)
- Then, as stated in the problem, with 1 CC stores, the clock rate will increase by 15%
  - Therefore $T_{original} = T$ and $T_{1CC\ Store} = 1.15T$
- Now, we need to calculate an original CPI and a new CPI:
  - $CPI_{original} = (0.43)(1) + (0.21)(1) + (0.12)(2) + (0.24)(2) = 1.36$
  - $CPI_{1CC\ Store} = (0.43)(1) + (0.21)(1) + (0.12)(1) + (0.24)(2) = 1.24$
- Now we can calculate CPU times:
  - $Time_{original} = P \times 1.36 \times T = 1.36(PT)$
  - $Time_{1CC\ Store} = P \times 1.24 \times 1.15T = 1.43(PT)$
- New time / original time = 1.43 / 1.36 > 1
  - Therefore the new time is slower – this is **not** a good improvement.

<u>Question</u> <u>2</u>:
Assume that you have a machine that is evaluated with a suite of benchmarks that consists of 4 different types of instructions: ALU instructions, load instructions, store instructions, and branch instructions. The frequency of each instruction type – as well as the number of clock cycles per instruction – for this benchmark suite is shown in the table below:

Type	Percentage	Average CPI
ALU	47%	6.7
Load	19%	7.9
Branch	20%	5.0
Store	14%	7.1

To improve upon this base performance, the chip's design team has carefully considered a list of potential design enhancements and has narrowed the list to 3. However, because the product needs to go to fab in 9 months, there is only time to make 2 of them. The three enhancements are listed below:

(a) The team can change the adder design from a conventional ripple-carry adder to a Kogge-Stone adder. The net effect of this design change is that the clock rate will increase from 180MHz to 200MHz as all instruction types need to leverage this resource during their lifetime.

(b) The team can add extra memory ports to the on-chip cache to improve the performance of instructions that reference memory. Simulations say that this change will reduce the average CPI for loads to 7.2 and will reduce the average CPI for stores to 6.6.

(c) The team can change the design of the multiply functional unit. Multiply instructions account for 23% of all ALU instructions. Were this change to be instituted, the average CPI of a multiply instruction would be reduced from 9 to 6.

<u>Part</u> <u>A</u>:
Which of these 2 options should be chosen to ensure the greatest performance improvement?

We can answer this question by calculating execution times – a base case and then the effects of Options AB, AC, and BC.

- Recall that the formula for CPU time is:
    - CPU Time = (instructions/program) x (clock cycles/instruction) x (seconds/clock cycle)
- Also, let the number of instructions = *i*

<u>Base</u> <u>Time</u>:
CPI_{base} = 6.644
$\text{CPU Time}_{\text{base}}$ = $(i) \times [(0.47*6.7) + (0.19*7.9) + (0.2*5) + (0.14*7.1)] \times (5.56\times10^{-9}\ \text{s}) = (3.69\times10^{-8})i$

<u>Option</u> <u>A</u> <u>&</u> <u>B</u>:
- To evaluate these choices, we need to calculate a new CPI and also change the clock cycle time.

CPI_{AB} = 6.441
$\text{CPU Time}_{\text{AB}}$ = $(i) \times [(0.47*6.7) + (0.19*7.2) + (0.2*5) + (0.14*6.6)] \times (5.0\times10^{-9}\ \text{s}) = (3.22\times10^{-8})i$

Option A & C:

- To evaluate these choices, we need to calculate a new CPI and also change the clock cycle time.

For option C, we need to determine a new average CPI for the ALU type instructions. This is done as follows:

Old ALU CPI:	(0.77)(non-multiply) + (0.23)(multiply)	=	6.7
New ALU CPI:	(0.77)(non-multiply) + (0.23)(multiply)	=	x

Because we know that 'multiply' = 9 CCs, we can solve for '(0.77)(non-multiply)'. This value = 4.63.

We can then use 4.63 to solve for x. Thus, x = 4.63 + (0.23)(6) = 6.01

CPI_{AC} = 6.320

CPU Time_{AC} = (i) x [(0.47*6.01) + (0.19*7.9) + (0.2*5) + (0.14*7.1)] x $(5.0\text{x}10^{-9}\text{ s}) = (3.16\text{x}10^{-8})\text{i}$

Option B & C:

- To evaluate these choices, we need to calculate a new CPI and also change the clock cycle time back to the original clock cycle time.

CPI_{BC} = 6.117

CPU Time_{BC} = (i) x [(0.47*6.01) + (0.19*7.2) + (0.2*5) + (0.14*6.6)] x $(5.56\text{x}10^{-9}\text{ s}) = (3.4\text{x}10^{-8})\text{i}$

Thus, after evaluating all of these choices, we know the following:

Option	Execution Time
Base	$(3.69\text{x}10^{-8})\text{i}$
A,B	$(3.22\text{x}10^{-8})\text{i}$
A,C	$(3.16\text{x}10^{-8})\text{i}$
B,C	$(3.40\text{x}10^{-8})\text{i}$

Clearly, using options A and C allow for the greatest performance improvements.

Part B:

What is the new average CPI?

CPI_{AC} = [(0.47*6.01) + (0.19*7.9) + (0.2*5) + (0.14*7.1)] = 6.32

Part C:

What performance speedup do we see?

Speedup = $(3.69\text{x}10^{-8})\text{i} / (3.16\text{x}10^{-8})\text{i}$ = 1.167

= 16.7% speedup

Part D:
Why do you think that the combination you found to perform the best actually performs best?

Options A and C are best because (i) A leverages the option that impacts every instruction and (ii) while the overall percentage of multiply instructions enhanced is only 10.8% (versus option B which effects a greater percentage of instructions), the performance of option C still has a significant impact in terms of percentage and a greater impact in terms of the average number of CC per instruction saved.