cse141 Spring 2010
Discussion Section:
Midterm Review

Matt DeVuyst
Pipeline Timing

- Assume:
  - Full forwarding
  - Branch resolution in EX stage
  - Predict-taken policy
  - Branch delay slot

1. L1: lw $1, 4($2)
2.      Add $3, $1, $4
3.      Beq $3, $1, L2 // Not taken
4.      Sw $1, 8($5)
5.      Addi $2, $2, 4
6. L2: addi $2, $2, 8
7.      And $6, $2, $6
8.      Beq $1, $7, L1 // Taken
9.      sw $5, 0($6)
Performance Question

- Suppose we wanted to add the auto-increment addressing mode to the MIPS ISA -- e.g., lw R1, 1000(R2++). This saves an instruction every time we observe a load followed by an increment of the address register. Is this a good idea?

- Consider only performance, and assume that we have to increase cycle time by 5% to accommodate the new instruction,
- that 20% of our instructions are loads,
- that we can apply this change to 40% of all loads,
- and that the CPI doesn't change.
Performance Question

- Suppose we wanted to add the auto-increment addressing mode to the MIPS ISA -- e.g., lw R1, 1000(R2++). This saves an instruction every time we observe a load followed by an increment of the address register. Is this a good idea?
- Consider only performance, and assume that we have to increase cycle time by 5% to accommodate the new instruction,
- that 20% of our instructions are loads,
- that we can apply this change to 40% of all loads,
- and that the CPI doesn't change.

\[
20\% \times 40\% = 8\% \text{ fewer instructions}
\]

\[
\text{ET}_{\text{new}} = (0.92 \times \text{IC}_{\text{old}}) \times \text{CPI} \times (1.05 \times \text{CT}_{\text{old}}) = 0.966 \times \text{CPI} \times \text{IC}_{\text{old}} \times \text{CT}_{\text{old}}
\]

\[
\text{ET}_{\text{new}} = 0.966 \times \text{ET}_{\text{old}}. \text{ Good idea. Speedup} = 1/0.966 = 1.035
\]
Performance Question 2

- You improve your memory subsystem so that memory latencies are sped up by a factor of 2.4.
- After applying the optimization, you observe that you now spend half your time on waiting for memory.
- What percentage of the original execution (before the optimization) was spent waiting for memory?
Performance Question 2

- You improve your memory subsystem so that memory latencies are sped up by a factor of 2.4.
- After applying the optimization, you observe that you now spend half your time on waiting for memory.
- What percentage of the original execution (before the optimization) was spent waiting for memory?

Suppose that after the optimization, execution takes 100 seconds. 50 seconds is waiting for memory and 50 seconds is not. If we improved memory latency by 2.4x, then we used to spend $2.4 \times 50 = 120$ seconds waiting for memory. Then the original execution time was $120 + 50 = 170$ seconds. We used to spend $120/170 = 70.6\%$ of the time waiting for memory.