Measuring Performance

Performance Tradeoff
Announcements

• HW 7 due tomorrow

• HW 8 will be out tonight

• Lab 4C due next Monday

• Lab 5 will be released tomorrow
Review: Spectrum of Associativity

- **A $K$-way set associative cache with $N$ blocks**
  - Number of cache sets $S = N / K$
    - Number of index bits $= \log_2(S)$
  - When $K = N$, fully associative cache
    - ONE cache set $\rightarrow$ zero index bits
  - When $K = 1$ (one-way), direct mapped cache
    - $N$ cache sets

- **Increasing the associativity**
  - Typically improves the hit rate (fewer conflicts)
  - But increases the hit time (takes longer to search)
Example: Cache Address Breakdown

• Assuming 16-bit memory addresses, how many bits are associated with the tag, index, and offset of the following configurations?

• 16 blocks, 16B per block, 2-way set associative
Review: Pipelined Processor w/ Cache Hierarchy
How Do We Measure Performance?

• **Execution time**: The time between the start and completion of a program (or task)

• **Throughput**: Total amount of work done in a given time

• **Improving performance means**
  – Reducing execution time, or
  – Increasing throughput
CPU Execution Time

• **Amount of time the CPU takes to run a program**

• **Derivation**

\[ \text{CPU execution time} = I \times CPI \times CT \]

- number of instructions in the program
- average number of cycles per instruction
- clock cycle time \((1/\text{frequency})\)
Instruction Count (I)

• **Total number of instructions executed by the processor for a given program**

• **Factors**
  – Instruction set
  – Mix of instructions chosen by the compiler
Cycle Time (CT)

- Clock period \((1/\text{frequency})\)

Factors
- Instruction set
- Structure of the processor and memory hierarchy
Cycles Per Instruction (CPI)

• **Average number of cycles required to execute each instruction**

• **Factors**
  – Instruction set
  – Mix of instructions chosen by the compiler
  – Ordering of the instructions by the compiler
  – Structure of the processor and memory hierarchy
Processor Organization
Impact on CPI (Example 1)

With forwarding: Reduced stall cycles
Lower CPI, potentially reduced execution time
Processor Organization
Impact on CPI (Example 2)

Only one delay slot needed with branch resolved in ID
Lower CPI
Filling the branch delay slot with a useful instruction
Hazard detection adds stall cycles (higher CPI) to ensure correctness
Discussion: Performance Tradeoff

Example 1

Shallow vs. Deep Pipelining
A Rough Breakdown of CPI

- $\text{CPI}_{\text{base}}$ is the base CPI in an ideal scenario where instruction fetches and data memory accesses incur no extra delay.

- $\text{CPI}_{\text{memhier}}$ is the (additional) CPI spent for accessing the memory hierarchy when a miss occurs in caches.

- $\text{CPI}_{\text{total}}$ is the overall CPI
  - $\text{CPI}_{\text{total}} = \text{CPI}_{\text{base}} + \text{CPI}_{\text{memhier}}$
Impact of the Memory Hierarchy

- **With no caches (only main memory)**
  - Every instruction is read from main memory
  - Every load and store instruction accesses main memory
  - Assume
    - 100 cycles to access main memory
    - 25% of all instructions are loads, 10% are stores

\[
\text{CPI}_{\text{memhier}} = 100 + (0.25 + 0.1) \times 100 = 135
\]
Impact of L1 Caches

- **With L1 caches**
  - L1 instruction cache miss rate = 2%
  - L1 data cache miss rate = 5%
  - Miss penalty = 100 cycles (access main memory)
  - 20% of all instructions are loads, 10% are stores

\[ CPI_{memhier} = \]
Impact of L1 Caches

• **With L1 caches**
  – L1 instruction cache miss rate = 2%
  – L1 data cache miss rate = 5%
  – Miss penalty = 100 cycles (access main memory)
  – 20% of all instructions are loads, 10% are stores

\[
\text{CPI}_{\text{memhier}} = 0.02 \times 100 + (0.2+0.1) \times 0.05 \times 100 = 3.5
\]
Exercise: CPI Estimation

• **With L1 caches**
  – L1 instruction cache miss rate = 2%
  – L1 data cache miss rate = 5%
  – Miss penalty = 50 cycles
  – 40% of all instructions are loads/stores

• \( \text{CPI}_{\text{memhier}} = ? \)
Impact of L1+L2 Caches

- With L1 and L2 caches
  - L1 instruction cache miss rate = 2%
  - L1 data cache miss rate = 5%
  - L2 access time = 15 cycles
  - L2 miss rate = 25%
  - L2 miss penalty = 100 cycles (access main memory)
  - 20% of all instructions are loads, 10% are stores

\[ \text{CPI}_{\text{memhier}} = \]
Impact of L1+L2 Caches

• **With L1 and L2 caches**
  – L1 instruction cache miss rate = 2%
  – L1 data cache miss rate = 5%
  – L2 access time = 15 cycles
  – L2 miss rate = 25%
  – L2 miss penalty = 100 cycles (access main memory)
  – 20% of all instructions are loads, 10% are stores

• \[ \text{CPI}_{\text{memhier}} = 0.02 \times (15 + 0.25 \times 100) + 0.30 \times 0.05 \times (15 + 0.25 \times 100) = 1.4 \]
Relative Performance

- Used to compare the performance of machines

- Used to report the performance benefit [loss] of adding [subtracting] a feature

\[
\frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution Time}_Y}{\text{Execution Time}_X}
\]
Relative Performance Example

- **Relative** $\text{CPI}_{\text{memhier}}$ of memory hierarchy alternatives
- **Assuming** $I$ and $CT$ stay the same

\[
\frac{\text{Performance}_{L1+L2}}{\text{Performance}_{L1}} = \frac{\text{CPI}_{\text{memhier}_{L1}}}{\text{CPI}_{\text{memhier}_{L1+L2}}} = \frac{3.4}{1.36} = 2.5
\]
Relative Performance Example

- Relative $\text{CPI}_\text{total}$ of memory hierarchy alternatives
  - Assume $\text{CPI}_\text{base} = 1.5$

\[
\frac{\text{Performance}_{L1+L2}}{\text{Performance}_{L1}} = \frac{\text{CPI}_{\text{total}_{L1}}}{\text{CPI}_{\text{total}_{L1+L2}}} = \frac{1.5 + 3.4}{1.5 + 1.36} = 1.7
\]
Amdahl’s Law

• Performance improvement possible with a given enhancement is limited by the amount that the enhancement is used

\[
\text{Execution Time}_{\text{enhanced}} = \frac{\text{Execution Time affected}}{\text{Amount of improvement}} + \text{Execution Time unaffected}
\]

• Example
  – Feature improves multiply operations by factor of 10
  – Total execution time of a program is 100 sec
  – Multiply operations consume 5 sec of the total

\[
\text{Execution Time}_{\text{enhanced}} = \frac{5}{10} + 95 = 95.5 \text{ sec}
\]
Discussion: Performance Tradeoff
Example 2

ADD 8(R1),0(R1),4(R1) vs.

LW R2,0(R1)
LW R3,4(R1)
ADD R2,R2,R3
SW R2,8(R1)

Complex Instruction vs. Simple Instruction
Before Next Class

• H&H 8.4

Next Time

Virtual Memory