Lecture: Performance measurement and Instruction Set Architectures

- Last Time
  - Introduction to performance
  - Computer benchmarks
  - Amdahl’s law

- Today
  - Take QUIZ 1 today over Chapter 1
  - Turn in your homework on Tuesday
  - Homework 2 is available
  - More on performance analysis
  - Introduction to ISAs

Review: latency vs. throughput

- Pizza delivery example
  - Do you want your pizza hot?
    - Low latency
  - Or do you want your pizza to be inexpensive?
    - High throughput - lots of pizzas per hour
  - Two different delivery strategies for pizza company!

In this course:
We will focus primarily on latency
(execution time for a single task)
Amdahl’s Law:
What fraction of the program are you improving?

Amdahl’s corollary

• Make the common case fast

• Examples:
  - All instructions require instruction fetch, only fraction require data
    ⇒ optimize instruction access first
  - Data locality (spatial, temporal), small memories faster
    ⇒ storage hierarchy: most frequent accesses to small, local memory
**CPU Performance Equation**

- 3 components to execution time:

\[
\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}
\]

- Factors affecting CPU execution time:

<table>
<thead>
<tr>
<th>Component</th>
<th>Inst. Count</th>
<th>CPI</th>
<th>Clock Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compiler</td>
<td>X (X)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inst. Set</td>
<td>X X (X)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization</td>
<td>X X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MicroArch</td>
<td>X X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technology</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Consider all three elements when optimizing
- Workloads change!

---

**Cycles Per Instruction (CPI)**

- Depends on the instruction
  - \( CPI_i = \text{Execution Time of Instruction } i \times \text{Clock Rate} \)

- Average cycles per instruction, \( IC = |\text{instructions}| \)

\[
CPI = \sum CPI_i \times F_i \quad \text{where } F_i = \frac{IC_i}{IC_{tot}}
\]

- Example:

<table>
<thead>
<tr>
<th>Op</th>
<th>Freq</th>
<th>Cycles</th>
<th>CPI(i)</th>
<th>%time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>50%</td>
<td>1</td>
<td>0.5</td>
<td>33%</td>
</tr>
<tr>
<td>Load</td>
<td>20%</td>
<td>2</td>
<td>0.4</td>
<td>27%</td>
</tr>
<tr>
<td>Store</td>
<td>10%</td>
<td>2</td>
<td>0.2</td>
<td>13%</td>
</tr>
<tr>
<td>Branch</td>
<td>20%</td>
<td>2</td>
<td>0.4</td>
<td>27%</td>
</tr>
<tr>
<td>CPI(total)</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Comparing and Summarizing Performance

- Fair way to summarize performance?
- Capture in a single number?
- Which of the following machines is best?

<table>
<thead>
<tr>
<th></th>
<th>Computer A</th>
<th>Computer B</th>
<th>Computer C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1</td>
<td>1</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Program 2</td>
<td>1000</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>Total Time</td>
<td>1001</td>
<td>110</td>
<td>40</td>
</tr>
</tbody>
</table>

Means

Arithmetic mean \[ \frac{1}{n} \sum_{i=1}^{n} T_i \] Can be weighted: \[ a_i T_i \]
Represents total execution time

Harmonic mean \[ \sum_{i=1}^{n} \frac{1}{R_i} \] \[ R_i = 1/T_i \]

Geometric mean \[ \left( \prod_{i=1}^{n} T_i \right)^{1/n} \] Good for mean of ratios, where the ratio is with respect to a reference.
Comparing and Summarizing Performance

- Fair way to summarize performance?
- Capture in a single number?

- Which of the following machines is best?

<table>
<thead>
<tr>
<th></th>
<th>Computer A</th>
<th>Computer B</th>
<th>Computer C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1</td>
<td>1</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Program 2</td>
<td>1000</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>Total Time</td>
<td>1001</td>
<td>110</td>
<td>40</td>
</tr>
<tr>
<td>Arithmetic Mean</td>
<td>500.5</td>
<td>55</td>
<td>20</td>
</tr>
<tr>
<td>Harmonic Mean</td>
<td>1.998</td>
<td>2.2</td>
<td>20</td>
</tr>
<tr>
<td>Geometric Mean</td>
<td>1.5</td>
<td>1.5</td>
<td>1</td>
</tr>
</tbody>
</table>

CPU Time Example

- **Computer A**: 2GHz clock, 10s CPU time
- **Designing Computer B**
  - Aim for 6s CPU time
  - Can do faster clock, but causes $1.2 \times$ clock cycles
- How fast must Computer B clock be?
CPU Time Example

- **Computer A**: 2GHz clock, 10s CPU time
- **Designing Computer B**
  - Aim for 6s CPU time
  - Can do faster clock, but causes $1.2 \times$ clock cycles
- **How fast must Computer B clock be?**

\[
\begin{align*}
\text{Clock Rate}_B &= \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6s} \\
\text{Clock Cycles}_A &= \text{CPU Time}_A \times \text{Clock Rate}_A \\
&= 10s \times 2GHz = 20 \times 10^9 \\
\text{Clock Rate}_B &= \frac{1.2 \times 20 \times 10^9}{6s} = \frac{24 \times 10^9}{6s} = 4GHz
\end{align*}
\]

Performance Summary

- **3 components to execution time:**
  \[
  \text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}
  \]
- **Depends on**
  - Algorithm: Instructions & CPI
  - Programming Language: Instructions & CPI
  - Compiler: Instructions & CPI
  - ISA: Instructions, CPI, & Cycle Time
- **Improvements**
  - Amdahl's law: what fraction of the program are you improving and by how much?
Is Speed the Last Word in Performance?

- Depends on the application!
- Cost
  - Not just processor, but other components (e.g., memory)
- Capacity
  - Many database applications are I/O bound and disk bandwidth is the precious commodity
- Power consumption
  - Trade power for performance in many applications

Power Trends

- In CMOS IC technology

\[
\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}
\]
Power

\[ P = CV^2F \]

- Capacitive load is proportional to \( |\text{transistors}| \) and fanout
- Voltage and frequency are functions of technology size & wire length (e.g., 130nm, 45nm)
- Historical Trends
  - \( \uparrow \) capacity \( \downarrow \) voltage \( \uparrow \) frequency
- Future
  - \( \uparrow \) capacity = voltage = or \( \downarrow \) frequency

Aggregate IT Energy Consumption

- Information (and communications) Technology (IT) consumes 2.5% of the world’s electricity = 1B tons of CO2 annually.
- In the US, data centers alone consume more than 60B KWH per year = energy consumed by entire transportation manufacturing sector.
- Current trends: energy usage will nearly double by 2011 for overall electricity cost of $7.4 B per year.
What does that really mean?
Environmental impact of data centers

If data centers were a country.....
Carbon emissions of world-wide DCs [Mankoff'08]

More on power at the end of the course from an expert!

Instruction set architectures
ISA is an interface (abstraction layer)

Instruction Set Architecture is a Contract

- **Contract** between programmer and the hardware
  - Defines visible state of the system
  - Defines how state changes in response to instructions

- **Programmer**:
  - ISA is model of how a program will execute

- **Hardware Designer**:
  - ISA is formal definition of the correct way to execute a program

- **ISA specification**
  - The binary encodings of the instruction set
  - How instructions modify state of the machine
ISA includes a model of the machine

A very simple model....

Memory  ────> ALU

Control and Instruction Pointer (PC)

A more typical ISA machine model

Registers  ────> ALU

Memory  ────> Control and Instruction Pointer (PC)
ISA Basics

Instruction formats
Instruction types
Addressing modes

Machine state includes
PC, memory state, register state

Machine State

- Registers
  - Size/Type
    - Program Counter (PC = IP)
    - accumulators
    - index registers
    - general registers
    - control registers
- Memory
  - Visible hierarchy (if any)
  - Addressibility
    - byte, word, bit
    - byte order (endian-ness)
    - maximum size
  - protection/relocation
Components of Instructions

- Operations (opcodes)
- Number of operands
- Operand specifiers

- Instruction encodings
- Instruction classes
  - ALU ops (add, sub, shift)
  - Branch (beq, bne, etc.)
  - Memory (ld/st)

```
add r1, r2, r3
```

Operand Number

- No Operands
  - HALT
  - NOP

- 1 operand
  - NOT R4
  - R4 ⇐ R4
  - JMP _L1

- 2 operands
  - ADD R1, R2
  - R1 ⇐ R1 + R2
  - LDI R3, #1234

- 3 operands
  - ADD R1, R2, R3
  - R1 ⇐ R2 + R3

- > 3 operands
  - MADD R4, R1, R2, R3
  - R4 ⇐ R1 + (R2*R3)
Effect of Operand Number

\[ E = (C+D) \times (C-D) \]

Assign
\[ C \Rightarrow r1 \]
\[ D \Rightarrow r2 \]
\[ E \Rightarrow r3 \]

<table>
<thead>
<tr>
<th>3 operand machine</th>
<th>2 operand machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>add r3, r1, r2</td>
<td>mov r3, r1</td>
</tr>
<tr>
<td>sub r4, r1, r2</td>
<td>add r3, r2</td>
</tr>
<tr>
<td>mult r3, r4, r3</td>
<td>sub r2, r1</td>
</tr>
<tr>
<td></td>
<td>mult r3, r2</td>
</tr>
</tbody>
</table>

Summary

- **ISA definition**
  - system state (general/special registers, memory)
  - the effect of each operation on the system state

- **Next Time**
  - Homework #1 is due - at start of class
  - ISA Design principals
  - Addressing modes
  - Data types
  - Common instruction types
  - Case studies: MIPS + others

- **Reading:** P&H 2.6-9