CSE775: Computer Architecture

Chapter 1: Fundamentals of Computer Design
Computer Architecture Topics

Input/Output and Storage

- Disks, WORM, Tape
- RAID

Emerging Technologies
- Interleaving Memories

Memory Hierarchy

- DRAM

L2 Cache

- Coherence, Bandwidth, Latency

L1 Cache

- Addressing, Protection, Exception Handling

VLSI

Instruction Set Architecture

- Pipelining, Hazard Resolution, Superscalar, Reordering, Prediction, Speculation, Vector, DSP

Pipelining and Instruction Level Parallelism
Computer Architecture Topics

- Processor-Memory-Switch
- Interconnection Network
- Multiprocessors
- Networks and Interconnections
- Shared Memory, Message Passing, Data Parallelism
- Network Interfaces
- Topologies, Routing, Bandwidth, Latency, Reliability

PM PM PM ... PM PM PM
Measurement and Evaluation

Architecture is an iterative process:
- Searching the space of possible designs
- At all levels of computer systems

Creativity

Cost / Performance Analysis

Good Ideas

Bad Ideas

Mediocre Ideas
Issues for a Computer Designer

• Functional Requirements Analysis (Target)
  – Business – transactional support/decimal arithmetic
  – General Purpose – balanced performance for a range of tasks

• Level of software compatibility
  – PL level
    • Flexible, Need new compiler, portability an issue
  – Binary level (x86 architecture)
    • Little flexibility, Portability requirements minimal

• OS requirements
  – Address space issues, memory management, protection

• Conformance to Standards
Computer Systems: Technology Trends

• 1988
  – Supercomputers
  – **Massively Parallel** Processors
  – Mini-supercomputers
  – Minicomputers
  – Workstations
  – PC’s

• 2008
  – Powerful PC’s and laptops
  – Clusters delivering **Petaflop performance**
  – Embedded Computers
  – PDAs, I-Phones, ..
Technology Trends

• *Integrated circuit logic technology* – a growth in transistor count on chip of about 40% to 55% per year.

• *Semiconductor RAM* – capacity increases by 40% per year, while cycle time has improved very slowly, decreasing by about one-third in 10 years. Cost has decreased at rate about the rate at which capacity increases.

• *Magnetic disc technology* – in 1990’s disk density had been improving 60% to 100% per year, while prior to 1990 about 30% per year. Since 2004, it dropped back to 30% per year.

• *Network technology* – Latency and bandwidth are important. Internet infrastructure in the U.S. has been doubling in bandwidth every year. High performance Systems Area Network (such as InfiniBand) delivering continuous reduced latency.
Why Such Change in 20 years?

• Performance
  – Technology Advances
    • CMOS (complementary metal oxide semiconductor) VLSI dominates older technologies like TTL (Transistor Transistor Logic) in cost AND performance
  – Computer architecture advances improves low-end
    • RISC, pipelining, superscalar, RAID, …

• Price: Lower costs due to …
  – Simpler development
    • CMOS VLSI: smaller systems, fewer components
  – Higher volumes
  – Lower margins by class of computer, due to fewer services
In 90’s, the main source of innovations in computer design has come from **RISC-style pipelined** processors. In the last several years, the annual growth rate is (only) 10-20%.
Growth in Performance of RAM & CPU

- Mismatch between CPU performance growth and memory performance growth!!
- And, almost unchanged memory latency
- Little instruction-level parallelism left to exploit efficiently
- Maximum power dissipation of air-cooled chips reached
Cost of Six Generations of DRAMs
Cost of Microprocessors
Components of Price for a $1000 PC
Integrated Circuits Costs

IC cost = Die cost + Testing cost + Packaging cost

Final test yield

Die cost = Wafer cost

Dies per Wafer * Die yield

Dies per wafer = ŝ * (Wafer_diam / 2)^2 - ŝ * Wafer_diam / 2 * Die Area - Test dies

Die Yield = Wafer yield * \(1 + \frac{\text{Defects_per_unit_area} \times \text{Die Area}}{\alpha}\)^

Die Cost goes roughly with die area^4
Failures and Dependability

• Failures at any level costs money
  – Integrated circuits (processor, memory)
  – Disks
  – Networks

• Costs Millions of Dollars for 1 hour downtime (Amazon, Google, ..)

• No concept of downtime at the middle of night

• Systems need to be designed with fault-tolerance
  – Hardware
  – Software
Performance and Cost

<table>
<thead>
<tr>
<th>Plane</th>
<th>DC to Paris</th>
<th>Speed</th>
<th>Passengers</th>
<th>Throughput (pmph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boeing 747</td>
<td>6.5 hours</td>
<td>610 mph</td>
<td>470</td>
<td>286,700</td>
</tr>
<tr>
<td>BAD/Sud Concodre</td>
<td>3 hours</td>
<td>1350 mph</td>
<td>132</td>
<td>178,200</td>
</tr>
</tbody>
</table>

- **Time to run the task** (ExTime)
  - Execution time, response time, latency
- **Tasks per day, hour, week, sec, ns …** (Performance)
  - Throughput, bandwidth
The Bottom Line:
Performance (and Cost)

"X is n times faster than Y" means

\[
\frac{\text{ExTime}(Y)}{\text{ExTime}(X)} = \frac{\text{Performance}(X)}{\text{Performance}(Y)}
\]

• Speed of Concorde vs. Boeing 747

• Throughput of Boeing 747 vs. Concorde
Metrics of Performance

Application
  Programming Language
    Compiler
      ISA

(millions) of Instructions per second: MIPS
(millions) of (FP) operations per second: MFLOP/s

Datapath Control
  Function Units
    Transistors Wires Pins

Answers per month
Operations per second

Megabytes per second
Cycles per second (clock rate)
Computer Engineering Methodology

Evaluate Existing Systems for Bottlenecks

Implementation Complexity

Technology Trends

Implement Next Generation System

Simulate New Designs and Organizations

Benchmarks

Workloads
Measurement Tools

- Benchmarks, Traces, Mixes
- Hardware: Cost, delay, area, power estimation
- Simulation (many levels)
  - ISA, RT, Gate, Circuit
- Queuing Theory
- Rules of Thumb
- Fundamental “Laws”/Principles
- Understanding the limitations of any measurement tool is crucial.
Issues with Benchmark Engineering

- Motivated by the bottom dollar, good performance on classic suites → more customers, better sales.
- Benchmark Engineering → Limits the longevity of benchmark suites
- Technology and Applications → Limits the longevity of benchmark suites.
SPEC: System Performance Evaluation Cooperative

• First Round 1989
  – 10 programs yielding a single number ("SPECmarks")

• Second Round 1992
  – SPECInt92 (6 integer programs) and SPECfp92 (14 floating point programs)
  – “benchmarks useful for 3 years”


• SPEC 2006 (CINT2006, CFP2006)

• Server Benchmarks
  – SPECWeb
  – SPECFS

• TPC (TPA-A, TPC-C, TPC-H, TPC-W, …)
SPEC 2000 (CINT 2000) Results
SPEC 2000 (CFP 2000) Results
Reporting Performance Results

• Reproducibility
• Apply them on publicly available benchmarks. Pecking/Picking order
  – Real Programs
  – Real Kernels
  – Toy Benchmarks
  – Synthetic Benchmarks
How to Summarize Performance

• Arithmetic mean (weighted arithmetic mean) tracks execution time: \( \text{sum}(T_i)/n \) or \( \text{sum}(W_i*T_i) \)

• Harmonic mean (weighted harmonic mean) of rates (e.g., MFLOPS) tracks execution time: \( n/\text{sum}(1/R_i) \) or \( 1/\text{sum}(W_i/R_i) \)
How to Summarize Performance (Cont’d)

• Normalized execution time is handy for scaling performance (e.g., X times faster than SPARCstation 10)

• But do not take the arithmetic mean of normalized execution time, use the Geometric Mean = \((\text{Product}(R_i)^{1/n})\)
Performance Evaluation

• “For better or worse, benchmarks shape a field”
• Good products created when have:
  – Good benchmarks
  – Good ways to summarize performance
• Given sales is a function in part of performance relative to competition, investment in improving product as reported by performance summary
• If benchmarks/summary inadequate, then choose between improving product for real programs vs. improving product to get more sales; Sales almost always wins!

• *Execution time is the measure of computer performance!*
Simulations

• When are simulations useful?

• What are its limitations, I.e. what real world phenomenon does it not account for?

• The larger the simulation trace, the less tractable the post-processing analysis.
Queuing Theory

• What are the distributions of arrival rates and values for other parameters?

• Are they realistic?

• What happens when the parameters or distributions are changed?
Quantitative Principles of Computer Design

• Make the Common Case Fast
  – Amdahl’s Law

• CPU Performance Equation
  – Clock cycle time
  – CPI
  – Instruction Count

• Principles of Locality

• Take advantage of Parallelism
Amdahl's Law

Speedup due to enhancement E:

\[
\text{Speedup}(E) = \frac{\text{ExTime w/o E}}{\text{ExTime w/ E}} = \frac{\text{Performance w/ E}}{\text{Performance w/o E}}
\]

Suppose that enhancement E accelerates a fraction F of the task by a factor S, and the remainder of the task is unaffected.
Amdahl’s Law

$$\text{ExTime}_{\text{new}} = \text{ExTime}_{\text{old}} \times \left(1 - \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}\right) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{ExTime}_{\text{old}}}{\text{ExTime}_{\text{new}}} = \frac{1}{\left(1 - \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}\right) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$
Amdahl’s Law (Cont’d)

• Floating point instructions improved to run 2X; but only 10% of actual instructions are FP

\[
\text{ExTime}_{\text{new}} = \\
\text{Speedup}_{\text{overall}} =
\]
**CPU Performance Equation**

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

<table>
<thead>
<tr>
<th></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</td>
<td>(X)</td>
<td></td>
</tr>
<tr>
<td>Inst. Set.</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Organization</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Technology</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Cycles Per Instruction

“Average Cycles per Instruction”

\[ CPI = \frac{(CPU \text{ Time} \times \text{Clock Rate})}{\text{Instruction Count}} \]

\[ = \frac{Cycles}{\text{Instruction Count}} \]

\[ CPU \text{ time} = \text{CycleTime} \times \prod_{i=1}^{n} CPI_i \]

“Instruction Frequency”

\[ CPI = \prod_{i=1}^{n} CPI_i \times F_i \]

where \( F_i \frac{1}{\text{Instruction Count}} \)

Invest Resources where time is Spent!
Example: Calculating CPI

<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>.5</td>
<td>(33%)</td>
</tr>
<tr>
<td>Load</td>
<td>20%</td>
<td>2</td>
<td>.4</td>
<td>(27%)</td>
</tr>
<tr>
<td>Store</td>
<td>10%</td>
<td>2</td>
<td>.2</td>
<td>(13%)</td>
</tr>
<tr>
<td>Branch</td>
<td>20%</td>
<td>2</td>
<td>.4</td>
<td>(27%)</td>
</tr>
</tbody>
</table>

Typical Mix: 1.5