Computer Architecture

Berk Sunar  Thomas Eisenbarth
WHY COMPUTER ARCHITECTURE?
Modern CPU Architecture

Outline

• Administrative Details
• Computer Architecture?
• Measuring Performance
Previous Knowledge

Expected background:
• Basic architecture
• Programming in C/C++/Java

Topics you should’ve seen before:
• Instruction sets, computer arithmetic, assembly programming, memory, I/O
• Pipelining, caches, virtual memory
Outline

- Administrative Details
- Computer Architecture
- Measuring Performance
What is Computer Architecture?

A hardware perspective:
• Understand modern computer components and their interaction
• Learn to evaluate and compare computers
• Enable to make own design decisions

→ Build your own computer
  – Instruction Set Design
  – Functional Organization
  – Logic Design
  – Implementation
Architecture and Other Disciplines

Architecture interacts with many other fields:

- Application Software
- Operating Systems, Compilers, Networking Software
- Computer Architecture
- Circuits, Wires, Devices, Network Hardware
Levels of Computer Architecture

Architecture:
• functional appearance to software
• opcodes, addressing modes, architected registers

Microarchitecture (= focus of this course)
• logical structure that implements the architecture
• pipelining, functional units, caches, physical registers

Realization (circuits)
• physical structure that embodies the implementation
• gates, cells, transistors, wires
Why study Computer Architecture?

Requirements are always changing:

• Are computers fast enough?
  – Computation intense applications: AI, gaming, virtual reality, gaming, complex simulation, gaming, ...

• Any goals beyond speed?
  – Power: heat dissipation, battery life, utility bill
  – Cost
  – Reliability
  – ...

Why study Computer Architecture?

**Rapid change in technology:** (approximate annual improvements)

- **IC logic Technology**
  - Transistor density +35%, die size +10-20%
  - together 40-55% doubling every 18 – 24 months (Moore’s Law)

- **Memory**
  - DRAM (memory): density +25-40%
  - FLASH (memory): density +50-60%, cost >15x cheaper than DRAM
  - Magnetic disk (mem): density +40%, cost >20x cheaper than FLASH

- Parameters change *and* change relative to one another!
- Not even including technology jumps like nanotechnologies
- Designs change even if requirements fixed
- Computer architect must plan ahead for changing tech
Computing: Yesterday vs. Today

60’s: one Computer, many users
Y2K: one Computer per user
TODAY: 1 user, countless computers

Two trends:
• Internet of Things
• Cloud Computing
Observation:

- (DRAM) transistor density doubles annually (slightly off: doubles every 18 months)

Corollaries:

- **Cost** per transistor halves annually (18 months)
- **Power** per transistor decreases with scaling
- **Speed** increases with scaling
- **Reliability** starting to decrease with scaling
Growth in Processor Performance

[Graph showing the growth in processor performance over time, with significant increases marked at 25%, 52%, and 22% per year.]
Some examples of Intel Processor Generations

<table>
<thead>
<tr>
<th>Year</th>
<th>Processor</th>
<th>transistors</th>
<th>Clock rate (MHz)</th>
<th>Performance in MIPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1982</td>
<td>80286</td>
<td>134k</td>
<td>12.5</td>
<td>2</td>
</tr>
<tr>
<td>1985</td>
<td>80386</td>
<td>275k</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>1989</td>
<td>80486</td>
<td>1.2M</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>1993</td>
<td>Pentium</td>
<td>3.1M</td>
<td>66</td>
<td>132</td>
</tr>
<tr>
<td>1997</td>
<td>Pentium Pro</td>
<td>5.5M</td>
<td>200</td>
<td>600</td>
</tr>
<tr>
<td>2001</td>
<td>Pentium 4</td>
<td>42M</td>
<td>1500</td>
<td>4500</td>
</tr>
<tr>
<td>2010</td>
<td>Core i7</td>
<td>1.17B</td>
<td>3333</td>
<td>50000</td>
</tr>
</tbody>
</table>

→ Performance improved by 25,000X
Power Consumption of CMOS

- Information stored as voltage levels –Hi =1/Lo=0
- Energy cost for a single switching event:
  \[ E_{dyn} = \frac{1}{2} \cdot C_{ld} \cdot V^2 \]
- Signal transitions dissipate power:

\[
P = \underbrace{\alpha \cdot C_{ld} \cdot V^2 \cdot f}_{\text{dynamic}} + \underbrace{V \cdot I_{\text{leak}}}_{\text{static}}
\]

Activity factor \( \alpha \) is determined by data

- Power and energy can both serve as a metric.
Reducing Power Consumption

Problem:
Heat Dissipation: IC size is 1.5 \( cm^2 \) (@80W for I7)
\( \rightarrow \) Freq. increase of 30% ended in 2003

Energy Efficiency improvements:
• Reduce Voltage (increases static power)
  – E.g. DVFS (Dynamic Voltage Frequency Scaling)
• Adjust clock frequency
• Turn off:
  – Quick power down/ race to halt
  – Stop clock (clock gating) or Power down (power gating) unused components
Better Metric: Energy

Example: race to halt
• Processor A needs 100% of power for 100% of run time
• Processor B needs 120% of power for 70% of run time

Q: Which processor is more energy efficient?

Answer:
Energy = power × run time

• $E_{\downarrow A} = 1 \times 1$
• $E_{\downarrow B} = 1.2 \times 0.7 = 0.84$

⇒ B needs only 84% of the energy of A!
Outline

• Administrative Details
• Computer Architecture
• Measuring Performance
Performance Measurement

Much of the focus of this course is on improving performance

Topics:
• performance metrics
• CPU performance equation
• benchmarks and benchmarking
• reporting averages
• Amdahl’s Law
• Little’s Law
• concepts
  – balance
  – tradeoffs
  – bursty behavior (average and peak performance)
Performance Metrics

When is a computer fast?

**Latency:** response time, execution time
- good metric for fixed amount of work (minimize time)

**Throughput:** bandwidth, work per time
- \( = \frac{1}{\text{latency}} \) when there is NO OVERLAP
- \( > \frac{1}{\text{latency}} \) when there is overlap
- in real processors, there is always overlap (e.g., pipelining)
- good metric for fixed amount of time (maximize work)

Measuring Performance:
- Execution time: \( t \) measured as:
  - wall time/response time (includes I/O) or
  - CPU time
- Performance: \( \text{Perf} = \frac{1}{t} \)
Next time:

More on:

Measuring Performance