Chapter 1

Computer Abstractions and Technology

Slides for CS 465
Fall 2016 – Section 001
1970: The IBM 7044 Computer
1970: Magnetic Core Memory

1 core = 1 bit
How Does a Computer Work?

- What is the Von Neumann computer architecture?
  - What are the elements of a Von Neumann computer?
  - How do these elements interact to execute programs?
Von Neumann Architecture

What is the Von Neumann computer architecture?

What are the elements of a Von Neumann computer?

How do these elements interact to execute programs?
Von Neumann Architecture

Introduction

ALU
Processor
Registers

Data
Memory
Program

Address bus
Data bus
Control bus

I/O subsystem

I/O device
I/O device
I/O device

Stored-program computer

system bus
Von Neumann Architecture

Additional Diagram Notes:
- General registers, FP registers, Program counter, etc.
- Sequence of instructions:
  1. Load/Store from memory to/from registers,
  2. Arithmetic/Logical Ops.,
  3. Conditional and unconditional branches.

Copyright © 2013, Daniel A. Menasce. All rights reserved.
## Instruction Formats: examples

<table>
<thead>
<tr>
<th>opcode</th>
<th>RX</th>
<th>RY</th>
<th>RZ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RZ \(\leftarrow\) RX op RY

<table>
<thead>
<tr>
<th>opcode</th>
<th>RX</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RX \(\leftarrow\) Mem(address)

Mem(address) \(\leftarrow\) RX

Unconditional or Conditional Jump to Address

<table>
<thead>
<tr>
<th>opcode</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Computer Revolution

- Progress in computer technology
  - Underpinned by Moore’s Law
- Makes novel applications feasible
  - Computers in automobiles
  - Cell phones
  - Human genome project
  - World Wide Web
  - Search Engines
- Computers are pervasive
Classes of Computers

- Personal computers
  - General purpose, variety of software
  - Subject to cost/performance tradeoff

- Server computers
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized
Classes of Computers

- **Supercomputers**
  - High-end scientific and engineering calculations
  - Highest capability but represent a small fraction of the overall computer market

- **Embedded computers**
  - Hidden as components of systems
  - Stringent power/performance/cost constraints
The PostPC Era

![Graph showing tablet, smartphone, PC, and cell phone sales over time. The graph indicates a shift in technology usage, with a peak in tablet sales around 2010 and a rise in smartphone sales from 2009 onwards. PC and cell phone sales show steady growth.]
The PostPC Era

- Personal Mobile Device (PMD)
  - Battery operated
  - Connects to the Internet
  - Hundreds of dollars
  - Smart phones, tablets, electronic glasses

- Cloud computing
  - Warehouse Scale Computers (WSC)
  - Software as a Service (SaaS)
  - Portion of software run on a PMD and a portion run in the Cloud
  - Amazon and Google
What You Will Learn

- How programs are translated into the machine language
  - And how the hardware executes them
- The hardware/software interface
- What determines program performance
  - And how it can be improved
- How hardware designers improve performance
- What is parallel processing
Understanding Performance

- **Algorithm**
  - Determines number of operations executed

- **Programming language, compiler, architecture**
  - Determine number of machine instructions executed per operation

- **Processor and memory system**
  - Determine how fast instructions are executed

- **I/O system (including OS)**
  - Determines how fast I/O operations are executed
Eight Great Ideas

- Design for *Moore’s Law*
- Use *abstraction* to simplify design
- Make the *common case fast*
- Performance *via parallelism*
- Performance *via pipelining*
- Performance *via prediction*
- *Hierarchy* of memories
- *Dependability* *via* redundancy
The number of transistors on integrated circuits doubles approximately every two years. (Gordon Moore, 1965)

Source: Wikimedia Commons
Below Your Program

- **Application software**
  - Written in high-level language

- **System software**
  - Compiler: translates HLL code to machine code
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources

- **Hardware**
  - Processor, memory, I/O controllers
Levels of Program Code

- **High-level language**
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability

- **Assembly language**
  - Textual representation of instructions

- **Hardware representation**
  - Binary digits (bits)
  - Encoded instructions and data
Components of a Computer

- Same components for all kinds of computer
  - Desktop, server, embedded

- Input/output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers
Touchscreen

- PostPC device
- Supersedes keyboard and mouse
- Resistive and Capacitive types
  - Most tablets, smart phones use capacitive
  - Capacitive allows multiple touches simultaneously
Through the Looking Glass

- LCD screen: picture elements (pixels)
- Mirrors content of frame buffer memory
Opening the Box

- Capacitive multitouch LCD screen
- 3.8 V, 25 Watt-hour battery
- Computer board
Inside the Processor (CPU)

- Datapath: performs operations on data
- Control: sequences datapath, memory, ...
- Cache memory
  - Small fast SRAM memory for immediate access to data
Inside the Processor

- Apple A5
Abstractions

The BIG Picture

- Abstraction helps us deal with complexity
  - Hide lower-level detail
- Instruction set architecture (ISA)
  - The hardware/software interface
- Application binary interface
  - The ISA plus system software interface
- Implementation
  - The details underlying and interface
A Safe Place for Data

- **Volatile main memory**
  - Loses instructions and data when power off
- **Non-volatile secondary memory**
  - Magnetic disk
  - Flash memory
  - Optical disk (CDROM, DVD)
Networks

- Communication, resource sharing, nonlocal access
- Local area network (LAN): Ethernet
- Wide area network (WAN): the Internet
- Wireless network: WiFi, Bluetooth
Technology Trends

- Electronics technology continues to evolve
  - Increased capacity and performance
  - Reduced cost

<table>
<thead>
<tr>
<th>Year</th>
<th>Technology</th>
<th>Relative performance/cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1951</td>
<td>Vacuum tube</td>
<td>1</td>
</tr>
<tr>
<td>1965</td>
<td>Transistor</td>
<td>35</td>
</tr>
<tr>
<td>1975</td>
<td>Integrated circuit (IC)</td>
<td>900</td>
</tr>
<tr>
<td>1995</td>
<td>Very large scale IC (VLSI)</td>
<td>2,400,000</td>
</tr>
<tr>
<td>2013</td>
<td>Ultra large scale IC</td>
<td>250,000,000,000</td>
</tr>
</tbody>
</table>
Semiconductor Technology

- Silicon: semiconductor
- Add materials to transform properties:
  - Conductors
  - Insulators
  - Switch
Manufacturing ICs

- Yield: proportion of working dies per wafer
Intel Core i7 Wafer

- 300mm wafer, 280 chips, 32nm technology
- Each chip is 20.7 x 10.5 mm
Integrated Circuit Cost

Cost per die = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}

\text{Dies per wafer} \approx \frac{\text{Wafer area}}{\text{Die area}}

\text{Yield} = \frac{1}{(1 + (\text{Defects per area} \times \text{Die area}/2))^2}

- Nonlinear relation to area and defect rate
  - Wafer cost and area are fixed
  - Defect rate determined by manufacturing process
  - Die area determined by architecture and circuit design
Defining Performance

Which airplane has the best performance?

- Passenger Capacity
- Cruising Range (miles)
- Cruising Speed (mph)
- Passengers x mph
Response Time and Throughput

- Response time
  - How long it takes to do a task

- Throughput
  - Total work done per unit time
    - e.g., tasks/transactions/… per hour

- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more processors?

- We’ll focus on response time for now…
Relative Performance

- Define Performance = 1/Execution Time
- “X is $n$ time faster than Y”

\[
\frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution time}_Y}{\text{Execution time}_X} = n
\]

- Example: time taken to run a program
  - 10 sec on A, 15 sec on B
  - Execution Time$_B /$ Execution Time$_A$
    \[
    = \frac{15\text{ sec}}{10\text{ sec}} = 1.5
    \]
  - So A is 1.5 times faster than B
Measuring Execution Time

- **Elapsed time**
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance

- **CPU time**
  - Time spent processing a given job
    - Discounts I/O time, other jobs’ shares
  - Comprises user CPU time and system CPU time
  - Different programs are affected differently by CPU and system performance
CPU Clocking

- Operation of digital hardware governed by a constant-rate clock

Clock period: duration of a clock cycle
- e.g., 250 ps = 0.25 ns = 250×10^{-12} s

Clock frequency (rate): cycles per second
- e.g., 4.0GHz = 4000MHz = 4.0×10^9Hz
CPU Time

CPU Time = CPU Clock Cycles \times \text{Clock Cycle Time}

\frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count
CPU Time Example

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

\[
\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6s}
\]

\[
\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A
\]

\[
= 10s \times 2GHz = 20 \times 10^9
\]

\[
\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6s} = \frac{24 \times 10^9}{6s} = 4GHz
\]
Instruction Count and CPI

Clock Cycles = Instruction Count × Cycles per Instruction

CPU Time = Instruction Count × CPI × Clock Cycle Time

= Instruction Count × CPI

Clock Rate

- Instruction Count for a program
  - Determined by program, ISA and compiler

- Average cycles per instruction
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix
CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

\[
\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A
\]
\[
= l \times 2.0 \times 250\text{ps} = l \times 500\text{ps}
\]

\[
\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B
\]
\[
= l \times 1.2 \times 500\text{ps} = l \times 600\text{ps}
\]

\[
\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{l \times 600\text{ps}}{l \times 500\text{ps}} = 1.2
\]

A is faster...

...by this much
CPI in More Detail

- If different instruction classes take different numbers of cycles

\[
\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)
\]

- Weighted average CPI

\[
\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left( \text{CPI}_i \times \frac{\text{Instruction Count}_i}{\text{Instruction Count}} \right)
\]

Relative frequency
CPI Example

- Alternative compiled code sequences using instructions in classes A, B, C

<table>
<thead>
<tr>
<th>Class</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPI for class</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>IC in sequence 1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>IC in sequence 2</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Sequence 1: IC = 5**
  - Clock Cycles
    = 2×1 + 1×2 + 2×3
    = 10
  - Avg. CPI = 10/5 = 2.0

- **Sequence 2: IC = 6**
  - Clock Cycles
    = 4×1 + 1×2 + 1×3
    = 9
  - Avg. CPI = 9/6 = 1.5
Performance Summary

Performance depends on

- Algorithm: affects IC, possibly CPI
- Programming language: affects IC, CPI
- Compiler: affects IC, CPI
- Instruction set architecture: affects IC, CPI, $T_c$
In CMOS IC technology

Power = Capacitive load × Voltage^2 × Frequency

×30

5V → 1V

×1000

Chapter 1 — Computer Abstractions and Technology — 46
Reducing Power

- Suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction

\[
\frac{P_{\text{new}}}{P_{\text{old}}} = \frac{C_{\text{old}} \times 0.85 \times (V_{\text{old}} \times 0.85)^2 \times F_{\text{old}} \times 0.85}{C_{\text{old}} \times V_{\text{old}}^2 \times F_{\text{old}}} = 0.85^4 = 0.52
\]

- The power wall
  - We can’t reduce voltage further
  - We can’t remove more heat

- How else can we improve performance?
Constrained by power, instruction-level parallelism, memory latency
Multiprocessors

- Multicore microprocessors
  - More than one processor per chip
- Requires explicitly parallel programming
  - Compare with instruction level parallelism
    - Hardware executes multiple instructions at once
    - Hidden from the programmer
- Hard to do
  - Programming for performance
  - Load balancing
  - Optimizing communication and synchronization
SPEC CPU Benchmark

- Programs used to measure performance
  - Supposedly typical of actual workload
- Standard Performance Evaluation Corp (SPEC)
  - Develops benchmarks for CPU, I/O, Web, …
- SPEC CPU2006
  - Elapsed time to execute a selection of programs
    - Negligible I/O, so focuses on CPU performance
  - Normalize relative to reference machine
  - Summarize as geometric mean of performance ratios
    - CINT2006 (integer) and CFP2006 (floating-point)
<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Instruction Count x 10^8</th>
<th>CPI</th>
<th>Clock cycle time (seconds x 10^-9)</th>
<th>Execution Time (seconds)</th>
<th>Reference Time (seconds)</th>
<th>SPECratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpreted string processing</td>
<td>perl</td>
<td>2252</td>
<td>0.60</td>
<td>0.376</td>
<td>508</td>
<td>9770</td>
<td>19.2</td>
</tr>
<tr>
<td>Block-sorting compression</td>
<td>bzip2</td>
<td>2390</td>
<td>0.70</td>
<td>0.376</td>
<td>629</td>
<td>9650</td>
<td>15.4</td>
</tr>
<tr>
<td>GNU C compiler</td>
<td>gcc</td>
<td>794</td>
<td>1.20</td>
<td>0.376</td>
<td>358</td>
<td>8050</td>
<td>22.5</td>
</tr>
<tr>
<td>Combinatorial optimization</td>
<td>mcf</td>
<td>221</td>
<td>2.66</td>
<td>0.376</td>
<td>221</td>
<td>9120</td>
<td>41.2</td>
</tr>
<tr>
<td>Go game (AI)</td>
<td>go</td>
<td>1274</td>
<td>1.10</td>
<td>0.376</td>
<td>527</td>
<td>10490</td>
<td>19.9</td>
</tr>
<tr>
<td>Search gene sequence</td>
<td>hmmor</td>
<td>2616</td>
<td>0.60</td>
<td>0.376</td>
<td>500</td>
<td>9330</td>
<td>15.8</td>
</tr>
<tr>
<td>Chess game (AI)</td>
<td>sjeng</td>
<td>1948</td>
<td>0.80</td>
<td>0.376</td>
<td>586</td>
<td>12100</td>
<td>20.7</td>
</tr>
<tr>
<td>Quantum computer simulation</td>
<td>libquantum</td>
<td>659</td>
<td>0.44</td>
<td>0.376</td>
<td>109</td>
<td>20720</td>
<td>190.0</td>
</tr>
<tr>
<td>Video compression</td>
<td>h264avc</td>
<td>3793</td>
<td>0.50</td>
<td>0.376</td>
<td>713</td>
<td>22130</td>
<td>31.0</td>
</tr>
<tr>
<td>Discrete event simulation library</td>
<td>omnetpp</td>
<td>367</td>
<td>2.10</td>
<td>0.376</td>
<td>290</td>
<td>6250</td>
<td>21.5</td>
</tr>
<tr>
<td>Games/path finding</td>
<td>astar</td>
<td>1250</td>
<td>1.00</td>
<td>0.376</td>
<td>470</td>
<td>7020</td>
<td>14.9</td>
</tr>
<tr>
<td>XML parsing</td>
<td>xalanckmk</td>
<td>1045</td>
<td>0.70</td>
<td>0.376</td>
<td>275</td>
<td>6900</td>
<td>25.1</td>
</tr>
<tr>
<td>Geometric mean</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25.7</td>
</tr>
</tbody>
</table>
SPEC Power Benchmark

- Power consumption of server at different workload levels
  - Performance: ssj_ops/sec
  - Power: Watts (Joules/sec)

Overall ssj_ops per Watt = \left( \sum_{i=0}^{10} ssj\_ops_i \right) / \left( \sum_{i=0}^{10} power_i \right)
## SPECpower_ssj2008 for Xeon X5650

<table>
<thead>
<tr>
<th>Target Load %</th>
<th>Performance (ssj_ops)</th>
<th>Average Power (Watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>865,618</td>
<td>258</td>
</tr>
<tr>
<td>90%</td>
<td>786,688</td>
<td>242</td>
</tr>
<tr>
<td>80%</td>
<td>698,051</td>
<td>224</td>
</tr>
<tr>
<td>70%</td>
<td>607,826</td>
<td>204</td>
</tr>
<tr>
<td>60%</td>
<td>521,391</td>
<td>185</td>
</tr>
<tr>
<td>50%</td>
<td>436,757</td>
<td>170</td>
</tr>
<tr>
<td>40%</td>
<td>345,919</td>
<td>157</td>
</tr>
<tr>
<td>30%</td>
<td>262,071</td>
<td>146</td>
</tr>
<tr>
<td>20%</td>
<td>176,061</td>
<td>135</td>
</tr>
<tr>
<td>10%</td>
<td>86,784</td>
<td>121</td>
</tr>
<tr>
<td>0%</td>
<td>0</td>
<td>80</td>
</tr>
<tr>
<td>Overall Sum</td>
<td>4,787,166</td>
<td>1,922</td>
</tr>
<tr>
<td>Σssj_ops/Σpower =</td>
<td></td>
<td>2,490</td>
</tr>
</tbody>
</table>
Pitfall: Amdahl’s Law

- Improving an aspect of a computer and expecting a proportional improvement in overall performance

\[ T_{\text{improved}} = \frac{T_{\text{affected}}}{\text{improvement factor}} + T_{\text{unaffected}} \]

- Example: multiply accounts for 80s/100s
  - How much improvement in multiply performance to get 5x overall?

  \[ 20 = \frac{80}{n} + 20 \]

  - Can’t be done!

- Corollary: make the common case fast
Fallacy: Low Power at Idle

- Look back at i7 power benchmark
  - At 100% load: 258W
  - At 50% load: 170W (66%)
  - At 10% load: 121W (47%)

- Google data center
  - Mostly operates at 10% – 50% load
  - At 100% load less than 1% of the time

- Consider designing processors to make power proportional to load
Pitfall: MIPS as a Performance Metric

- MIPS: Millions of Instructions Per Second
  - Doesn’t account for
    - Differences in ISAs between computers
    - Differences in complexity between instructions

\[
\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6}
\]

\[
= \frac{\text{Instruction count}}{\text{Instruction count} \times \text{CPI} \times 10^6}
= \frac{\text{Clock rate}}{\text{CPI} \times 10^6}
\]

- CPI varies between programs on a given CPU
Concluding Remarks

- Cost/performance is improving
  - Due to underlying technology development
- Hierarchical layers of abstraction
  - In both hardware and software
- Instruction set architecture
  - The hardware/software interface
- Execution time: the best performance measure
- Power is a limiting factor
  - Use parallelism to improve performance