### Part I

#### Background and Motivation

<table>
<thead>
<tr>
<th>Parts</th>
<th>Chapters</th>
</tr>
</thead>
</table>
| I. Background and Motivation  | 1. Combinational Digital Circuits  
2. Digital Circuits with Memory  
3. Computer System Technology  
4. Computer Performance        |
| II. Instruction-Set Architecture | 5. Instructions and Addressing  
6. Procedures and Data  
7. Assembly Language Programs  
8. Instruction-Set Variations  |
| III. The Arithmetic/Logic Unit | 9. Number Representation  
10. Adders and Simple ALUs    
11. Multipliers and Dividers  
12. Floating-Point Arithmetic |
| IV. Data Path and Control     | 13. Instruction Execution Steps  
14. Control Unit Synthesis  
15. Pipelined Data Paths      
16. Pipeline Performance Limits |
| V. Memory System Design       | 17. Main Memory Concepts  
18. Cache Memory Organization  
19. Mass Memory Concepts      
20. Virtual Memory and Paging |
| VI. Input/Output and Interfacing | 21. Input/Output Devices  
22. Input/Output Programming  
23. Buses, Links, and Interfacing  
24. Context Switching and Interrupts |
| VII. Advanced Architectures   | 25. Road to Higher Performance  
26. Vector and Array Processing  
27. Shared-Memory Multiprocessing  
28. Distributed Multicomputing |
About This Presentation

This presentation is intended to support the use of the textbook *Computer Architecture: From Microprocessors to Supercomputers*, Oxford University Press, 2005, ISBN 0-19-515455-X. It is updated regularly by the author as part of his teaching of the upper-division course ECE 154, Introduction to Computer Architecture, at the University of California, Santa Barbara. Instructors can use these slides freely in classroom teaching and for other educational purposes. Any other use is strictly prohibited. © Behrooz Parhami

<table>
<thead>
<tr>
<th>Edition</th>
<th>Released</th>
<th>Revised</th>
<th>Revised</th>
<th>Revised</th>
<th>Revised</th>
</tr>
</thead>
</table>
I Background and Motivation

Provide motivation, paint the big picture, introduce tools:
- Review components used in building digital circuits
- Present an overview of computer technology
- Understand the meaning of computer performance
  (or why a 2 GHz processor isn’t 2 \times as fast as a 1 GHz model)

<table>
<thead>
<tr>
<th>Topics in This Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1</td>
</tr>
<tr>
<td>Chapter 2</td>
</tr>
<tr>
<td>Chapter 3</td>
</tr>
<tr>
<td>Chapter 4</td>
</tr>
</tbody>
</table>
1 Combinational Digital Circuits

First of two chapters containing a review of digital design:
• Combinational, or memoryless, circuits in Chapter 1
• Sequential circuits, with memory, in Chapter 2

<table>
<thead>
<tr>
<th>Topics in This Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Signals, Logic Operators, and Gates</td>
</tr>
<tr>
<td>1.2 Boolean Functions and Expressions</td>
</tr>
<tr>
<td>1.3 Designing Gate Networks</td>
</tr>
<tr>
<td>1.4 Useful Combinational Parts</td>
</tr>
<tr>
<td>1.5 Programmable Combinational Parts</td>
</tr>
<tr>
<td>1.6 Timing and Circuit Considerations</td>
</tr>
</tbody>
</table>
1.1 Signals, Logic Operators, and Gates

<table>
<thead>
<tr>
<th>Name</th>
<th>NOT</th>
<th>AND</th>
<th>OR</th>
<th>XOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphical symbol</td>
<td><img src="image" alt="NOT" /></td>
<td><img src="image" alt="AND" /></td>
<td><img src="image" alt="OR" /></td>
<td><img src="image" alt="XOR" /></td>
</tr>
<tr>
<td>Operator sign and alternate(s)</td>
<td>$x'$</td>
<td>$xy$</td>
<td>$x \lor y$</td>
<td>$x \oplus y$</td>
</tr>
<tr>
<td>Output is 1 iff:</td>
<td>Input is 0</td>
<td>Both inputs are 1s</td>
<td>At least one input is 1</td>
<td>Inputs are not equal</td>
</tr>
<tr>
<td>Arithmetic expression</td>
<td>$1 - x$</td>
<td>$xy$ or $xy$</td>
<td>$x + y - xy$</td>
<td>$x + y - 2xy$</td>
</tr>
</tbody>
</table>

Figure 1.1 Some basic elements of digital logic circuits, with operator signs used in this book highlighted.
The Arithmetic Substitution Method

<table>
<thead>
<tr>
<th>Expression</th>
<th>Converted to Arithmetic Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>$z' = 1 - z$</td>
<td>NOT</td>
</tr>
<tr>
<td>$xy$</td>
<td>AND same as multiplication</td>
</tr>
<tr>
<td>($\text{when doing the algebra, set } z^k = z$)</td>
<td></td>
</tr>
<tr>
<td>$x \lor y = x + y - xy$</td>
<td>OR</td>
</tr>
<tr>
<td>$x \oplus y = x + y - 2xy$</td>
<td>XOR</td>
</tr>
</tbody>
</table>

Example: Prove the identity $xyz \lor x' \lor y' \lor z' \equiv ? 1$

LHS = $[xyz \lor x'] \lor [y' \lor z']$

= $[xyz + 1 - x - (1 - x)xyz] \lor [1 - y + 1 - z - (1 - y)(1 - z)]$

= $[xyz + 1 - x] \lor [1 - yz]$

= $(xyz + 1 - x) + (1 - yz) - (xyz + 1 - x)(1 - yz)$

= $1 + xy^2z^2 - xyz$

= $1$ = RHS

This is addition, not logical OR
Variations in Gate Symbols

AND

OR

NAND

NOR

XNOR

Figure 1.2 Gates with more than two inputs and/or with inverted signals at input or output.
Gates as Control Elements

Figure 1.3  An AND gate and a tristate buffer act as controlled switches or valves. An inverting buffer is logically the same as a NOT gate.
Wired OR and Bus Connections

Figure 1.4 Wired OR allows tying together of several controlled signals.
Control/Data Signals and Signal Bundles

Figure 1.5 Arrays of logic gates represented by a single gate symbol.

(a) 8 NOR gates
(b) 32 AND gates
(c) k XOR gates
1.2 Boolean Functions and Expressions

Ways of specifying a logic function

- Truth table: $2^n$ row, “don’t-care” in input or output

- Logic expression: $w' (x \lor y \lor z)$, product-of-sums, sum-of-products, equivalent expressions

- Word statement: Alarm will sound if the door is opened while the security system is engaged, or when the smoke detector is triggered

- Logic circuit diagram: Synthesis vs analysis
### Table 1.2  Laws (basic identities) of Boolean algebra.

<table>
<thead>
<tr>
<th>Name of law</th>
<th>OR version</th>
<th>AND version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>$x \lor 0 = x$</td>
<td>$x \land 1 = x$</td>
</tr>
<tr>
<td>One/Zero</td>
<td>$x \lor 1 = 1$</td>
<td>$x \land 0 = 0$</td>
</tr>
<tr>
<td>Idempotent</td>
<td>$x \lor x = x$</td>
<td>$x \land x = x$</td>
</tr>
<tr>
<td>Inverse</td>
<td>$x \lor x' = 1$</td>
<td>$x \land x' = 0$</td>
</tr>
<tr>
<td>Commutative</td>
<td>$x \lor y = y \lor x$</td>
<td>$x \land y = y \land x$</td>
</tr>
<tr>
<td>Associative</td>
<td>$(x \lor y) \lor z = x \lor (y \lor z)$</td>
<td>$(x \land y) \land z = x \land (y \land z)$</td>
</tr>
<tr>
<td>Distributive</td>
<td>$x \lor (y \land z) = (x \lor y) \lor (x \lor z)$</td>
<td>$x \land (y \lor z) = (x \land y) \lor (x \land z)$</td>
</tr>
<tr>
<td>DeMorgan’s</td>
<td>$(x \lor y)' = x' \land y'$</td>
<td>$(x \land y)' = x' \lor y'$</td>
</tr>
</tbody>
</table>
Proving the Equivalence of Logic Expressions

Example 1.1

- Truth-table method: Exhaustive verification

- Arithmetic substitution
  \[ x \lor y = x + y - xy \]
  \[ x \oplus y = x + y - 2xy \]
  Example: \[ x \oplus y \equiv? x' y \lor x y' \]
  \[ x + y - 2xy \equiv? (1-x)y + x(1-y) - (1-x)yx(1-y) \]

- Case analysis: two cases, \( x = 0 \) or \( x = 1 \)

- Logic expression manipulation
1.3 Designing Gate Networks

- AND-OR, NAND-NAND, OR-AND, NOR-NOR

- Logic optimization: cost, speed, power dissipation

\[(x \lor y)' = x'y'\]

(a) AND-OR circuit  
(b) Intermediate circuit  
(c) NAND-NAND equivalent

Figure 1.6 A two-level AND-OR circuit and two equivalent circuits.
Seven-Segment Display of Decimal Digits

Figure 1.7 Seven-segment display of decimal digits. The three open segments may be optionally used. The digit 1 can be displayed in two ways, with the more common right-side version shown.
BCD-to-Seven-Segment Decoder

Example 1.2

Figure 1.8 The logic circuit that generates the enable signal for the lowermost segment (number 3) in a seven-segment display unit.
1.4 Useful Combinational Parts

• High-level building blocks

• Much like prefab parts used in building a house

• Arithmetic components (adders, multipliers, ALUs) will be covered in Part III

• Here we cover three useful parts: multiplexers, decoders/demultiplexers, encoders
Multiplexers

Figure 1.9 Multiplexer (mux), or selector, allows one of several inputs to be selected and routed to output depending on the binary value of a set of selection or address signals provided to it.
Decoders/Demultiplexers

Figure 1.10  A decoder allows the selection of one of $2^a$ options using an $a$-bit address as input. A demultiplexer (demux) is a decoder that only selects an output if its enable signal is asserted.
Encoders

Figure 1.11  A $2^a$-to-a encoder outputs an $a$-bit binary number equal to the index of the single 1 among its $2^a$ inputs.
1.5 Programmable Combinational Parts

A programmable combinational part can do the job of many gates or gate networks

Programmed by cutting existing connections (fuses) or establishing new connections (antifuses)

- Programmable ROM (PROM)
- Programmable array logic (PAL)
- Programmable logic array (PLA)
Figure 1.12  Programmable connections and their use in a PROM.
PALs and PLAs

(a) General programmable combinational logic

(b) PAL: programmable AND array, fixed OR array

(c) PLA: programmable AND and OR arrays

Figure 1.13 Programmable combinational logic: general structure and two classes known as PAL and PLA devices. Not shown is PROM with fixed AND array (a decoder) and programmable OR array.
1.6 Timing and Circuit Considerations

Changes in gate/circuit output, triggered by changes in its inputs, are not instantaneous

- Gate delay $\delta$: a fraction of, to a few, nanoseconds
- Wire delay, previously negligible, is now important (electronic signals travel about 15 cm per ns)
- Circuit simulation to verify function and timing
Glitching

Using the PAL in Fig. 1.13b to implement \( f = x \lor y \lor z \)

\[
x = 0
\]

\[
y
\]

\[
z
\]

\[
a = x \lor y
\]

\[
f = a \lor z
\]

Figure 1.14  Timing diagram for a circuit that exhibits glitching.
CMOS Transmission Gates

(a) CMOS transmission gate: circuit and symbol

(b) Two-input mux built of two transmission gates

Figure 1.15  A CMOS transmission gate and its use in building a 2-to-1 mux.
2 Digital Circuits with Memory

Second of two chapters containing a review of digital design:
- Combinational (memoryless) circuits in Chapter 1
- Sequential circuits (with memory) in Chapter 2

<table>
<thead>
<tr>
<th>Topics in This Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Latches, Flip-Flops, and Registers</td>
</tr>
<tr>
<td>2.2 Finite-State Machines</td>
</tr>
<tr>
<td>2.3 Designing Sequential Circuits</td>
</tr>
<tr>
<td>2.4 Useful Sequential Parts</td>
</tr>
<tr>
<td>2.5 Programmable Sequential Parts</td>
</tr>
<tr>
<td>2.6 Clocks and Timing of Events</td>
</tr>
</tbody>
</table>
2.1 Latches, Flip-Flops, and Registers

(a) SR latch

(b) D latch

(c) Master-slave D flip-flop

(d) D flip-flop symbol

(e) k-bit register

Figure 2.1 Latches, flip-flops, and registers.
Latches vs Flip-Flops

Figure 2.2  Operations of D latch and negative-edge-triggered D flip-flop.
Reading and Modifying FFs in the Same Cycle

Figure 2.3     Register-to-register operation with edge-triggered flip-flops.
2.2 Finite-State Machines

Example 2.1

Figure 2.4 State table and state diagram for a vending machine coin reception unit.
Sequential Machine Implementation

Figure 2.5  Hardware realization of Moore and Mealy sequential machines.
2.3 Designing Sequential Circuits

Example 2.3

Figure 2.7 Hardware realization of a coin reception unit (Example 2.3).

Inputs
- Quarter in $q$
- Dime in $d$

Outputs
- Final state is 1xx
- Output $e$

FF2
- D FF2
- Q
- C

FF1
- D FF1
- Q
- C

FF0
- D FF0
- Q
- C
2.4 Useful Sequential Parts

- High-level building blocks
- Much like prefab closets used in building a house
- Other memory components will be covered in Chapter 17 (SRAM details, DRAM, Flash)
- Here we cover three useful parts: shift register, register file (SRAM basics), counter
Figure 2.8  Register with single-bit left shift and parallel load capabilities. For logical left shift, serial data in line is connected to 0.
Register File and FIFO

Figure 2.9 Register file with random access and FIFO.
Figure 2.10  SRAM memory is simply a large, single-port register file.
Figure 2.11  Synchronous binary counter with initialization capability.
2.5 Programmable Sequential Parts

A programmable sequential part contains gates and memory elements.

Programmed by cutting existing connections (fuses) or establishing new connections (antifuses).

- Programmable array logic (PAL)
- Field-programmable gate array (FPGA)
- Both types contain macrocells and interconnects.
Figure 2.12  Examples of programmable sequential logic.

(a) Portion of PAL with storable output

(b) Generic structure of an FPGA

PAL and FPGA
2.6 Clocks and Timing of Events

Clock is a periodic signal: clock rate = clock frequency
The inverse of clock rate is the clock period: 1 GHz ↔ 1 ns
Constraint: Clock period ≥ \( t_{\text{prop}} + t_{\text{comb}} + t_{\text{setup}} + t_{\text{skew}} \)

Figure 2.13 Determining the required length of the clock period.
Synchronization

Figure 2.14    Synchronizers are used to prevent timing problems arising from untimely changes in asynchronous signals.
Level-Sensitive Operation

Figure 2.15  Two-phase clocking with nonoverlapping clock signals.
3 Computer System Technology

Interplay between architecture, hardware, and software
- Architectural innovations influence technology
- Technological advances drive changes in architecture

### Topics in This Chapter

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>From Components to Applications</td>
</tr>
<tr>
<td>3.2</td>
<td>Computer Systems and Their Parts</td>
</tr>
<tr>
<td>3.3</td>
<td>Generations of Progress</td>
</tr>
<tr>
<td>3.4</td>
<td>Processor and Memory Technologies</td>
</tr>
<tr>
<td>3.5</td>
<td>Peripherals, I/O, and Communications</td>
</tr>
<tr>
<td>3.6</td>
<td>Software Systems and Applications</td>
</tr>
</tbody>
</table>
3.1 From Components to Applications

Figure 3.1  Subfields or views in computer system engineering.
What Is (Computer) Architecture?

Like a building architect, whose place at the engineering/arts and goals/means interfaces is seen in this diagram, a computer architect reconciles many conflicting or competing demands.

Figure 3.2
3.2 Computer Systems and Their Parts

Figure 3.3  The space of computer systems, with what we normally mean by the word “computer” highlighted.
Figure 3.4  Classifying computers by computational power and price range.

Price/Performance Pyramid

- **Embedded**
  - $10s
- **Personal**
  - $100s
- **Workstation**
  - $1000s
- **Server**
  - $10s Ks
- **Mainframe**
  - $100s Ks
- **Super**
  - $Millions

Differences in scale, not in substance.
Automotive Embedded Computers

Figure 3.5 Embedded computers are ubiquitous, yet invisible. They are found in our automobiles, appliances, and many other places.
Figure 3.6    Notebooks, a common class of portable computers, are much smaller than desktops but offer substantially the same capabilities. What are the main reasons for the size difference?
Figure 3.7  The (three, four, five, or) six main units of a digital computer. Usually, the link unit (a simple bus or a more elaborate network) is not explicitly included in such diagrams.
### 3.3 Generations of Progress

#### Table 3.2 The 5 generations of digital computers, and their ancestors.

<table>
<thead>
<tr>
<th>Generation (begun)</th>
<th>Processor technology</th>
<th>Memory innovations</th>
<th>I/O devices introduced</th>
<th>Dominant look &amp; fell</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (1600s)</td>
<td>(Electro-) mechanical</td>
<td>Wheel, card</td>
<td>Lever, dial, punched card</td>
<td>Factory equipment</td>
</tr>
<tr>
<td>1 (1950s)</td>
<td>Vacuum tube</td>
<td>Magnetic drum</td>
<td>Paper tape, magnetic tape</td>
<td>Hall-size cabinet</td>
</tr>
<tr>
<td>2 (1960s)</td>
<td>Transistor</td>
<td>Magnetic core</td>
<td>Drum, printer, text terminal</td>
<td>Room-size mainframe</td>
</tr>
<tr>
<td>3 (1970s)</td>
<td>SSI/MSI</td>
<td>RAM/ROM chip</td>
<td>Disk, keyboard, video monitor</td>
<td>Desk-size mini</td>
</tr>
<tr>
<td>4 (1980s)</td>
<td>LSI/VLSI</td>
<td>SRAM/DRAM</td>
<td>Network, CD, mouse, sound</td>
<td>Desktop/laptop micro</td>
</tr>
<tr>
<td>5 (1990s)</td>
<td>ULSI/GSI/WSI, SOC</td>
<td>SDRAM, flash</td>
<td>Sensor/actuator, point/click</td>
<td>Invisible, embedded</td>
</tr>
</tbody>
</table>
Figure 3.8 The manufacturing process for an IC part.
Effect of Die Size on Yield

Figure 3.9  Visualizing the dramatic decrease in yield with larger dies.

Die yield = \(_{\text{def}} (\text{number of good dies}) / (\text{total number of dies})

\[
\text{Die yield} = \text{Wafer yield} \times [1 + (\text{Defect density} \times \text{Die area}) / a]^a
\]

Die cost = (cost of wafer) / (total number of dies \times \text{die yield})
= (cost of wafer) \times (\text{die area} / \text{wafer area}) / (\text{die yield})
3.4 Processor and Memory Technologies

Figure 3.11 Packaging of processor, memory, and other components.

(a) 2D or 2.5D packaging now common
(b) 3D packaging of the future
Figure 3.10  Trends in processor performance and DRAM memory chip capacity (Moore’s law).
Pitfalls of Computer Technology Forecasting

“DOS addresses only 1 MB of RAM because we cannot imagine any applications needing more.” Microsoft, 1980

“640K ought to be enough for anybody.” Bill Gates, 1981

“Computers in the future may weigh no more than 1.5 tons.” Popular Mechanics

“I think there is a world market for maybe five computers.” Thomas Watson, IBM Chairman, 1943

“There is no reason anyone would want a computer in their home.” Ken Olsen, DEC founder, 1977

“The 32-bit machine would be an overkill for a personal computer.” Sol Libes, ByteLines
3.5 Input/Output and Communications

Figure 3.12 Magnetic and optical disk memory units.
Figure 3.13 Latency and bandwidth characteristics of different classes of communication links.
3.6 Software Systems and Applications

Figure 3.15  Categorization of software, with examples in each class.
Figure 3.14 Models and abstractions in programming.
4 Computer Performance

Performance is key in design decisions; also cost and power
- It has been a driving force for innovation
- Isn’t quite the same as speed (higher clock rate)

<table>
<thead>
<tr>
<th>Topics in This Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Cost, Performance, and Cost/Performance</td>
</tr>
<tr>
<td>4.2 Defining Computer Performance</td>
</tr>
<tr>
<td>4.3 Performance Enhancement and Amdahl’s Law</td>
</tr>
<tr>
<td>4.4 Performance Measurement vs Modeling</td>
</tr>
<tr>
<td>4.5 Reporting Computer Performance</td>
</tr>
<tr>
<td>4.6 The Quest for Higher Performance</td>
</tr>
</tbody>
</table>
4.1 Cost, Performance, and Cost/Performance

![Graph showing the decrease in computer cost over time from 1960 to 2020. The y-axis represents computer cost in thousands ($1 K), millions ($1 M), and billions ($1 G). The x-axis represents calendar years from 1960 to 2020. The graph shows a downward trend indicating a decrease in cost over time.]

Jan. 2007 Computer Architecture, Background and Motivation
Figure 4.1 Performance improvement as a function of cost.
4.2 Defining Computer Performance

Figure 4.2  Pipeline analogy shows that imbalance between processing power and I/O capabilities leads to a performance bottleneck.
Performance of Aircraft: An Analogy

Table 4.1  Key characteristics of six passenger aircraft: all figures are approximate; some relate to a specific model/configuration of the aircraft or are averages of cited range of values.

<table>
<thead>
<tr>
<th>Aircraft</th>
<th>Passengers</th>
<th>Range (km)</th>
<th>Speed (km/h)</th>
<th>Price ($M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airbus A310</td>
<td>250</td>
<td>8 300</td>
<td>895</td>
<td>120</td>
</tr>
<tr>
<td>Boeing 747</td>
<td>470</td>
<td>6 700</td>
<td>980</td>
<td>200</td>
</tr>
<tr>
<td>Boeing 767</td>
<td>250</td>
<td>12 300</td>
<td>885</td>
<td>120</td>
</tr>
<tr>
<td>Boeing 777</td>
<td>375</td>
<td>7 450</td>
<td>980</td>
<td>180</td>
</tr>
<tr>
<td>Concorde</td>
<td>130</td>
<td>6 400</td>
<td>2 200</td>
<td>350</td>
</tr>
<tr>
<td>DC-8-50</td>
<td>145</td>
<td>14 000</td>
<td>875</td>
<td>80</td>
</tr>
</tbody>
</table>
Different Views of Performance

Performance from the viewpoint of a passenger: **Speed**

Note, however, that flight time is but one part of total travel time. Also, if the travel distance exceeds the range of a faster plane, a slower plane may be better due to not needing a refueling stop.

Performance from the viewpoint of an airline: **Throughput**

Measured in passenger-km per hour (relevant if ticket price were proportional to distance traveled, which in reality it is not)

<table>
<thead>
<tr>
<th>Aircraft</th>
<th>Speed (passenger-km/hr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airbus A310</td>
<td>$250 \times 895 = 0.224$ M</td>
</tr>
<tr>
<td>Boeing 747</td>
<td>$470 \times 980 = 0.461$ M</td>
</tr>
<tr>
<td>Boeing 767</td>
<td>$250 \times 885 = 0.221$ M</td>
</tr>
<tr>
<td>Boeing 777</td>
<td>$375 \times 980 = 0.368$ M</td>
</tr>
<tr>
<td>Concorde</td>
<td>$130 \times 2200 = 0.286$ M</td>
</tr>
<tr>
<td>DC-8-50</td>
<td>$145 \times 875 = 0.127$ M</td>
</tr>
</tbody>
</table>

Performance from the viewpoint of FAA: **Safety**
Cost Effectiveness: Cost/Performance

Table 4.1  Key characteristics of six passenger aircraft: all figures are approximate; some relate to a specific model/configuration of the aircraft or are averages of cited range of values.

<table>
<thead>
<tr>
<th>Aircraft</th>
<th>Passengers</th>
<th>Range (km)</th>
<th>Speed (km/h)</th>
<th>Price ($M)</th>
<th>Larger values better</th>
<th>Smaller values better</th>
</tr>
</thead>
<tbody>
<tr>
<td>A310</td>
<td>250</td>
<td>8 300</td>
<td>895</td>
<td>120</td>
<td>0.224</td>
<td>536</td>
</tr>
<tr>
<td>B 747</td>
<td>470</td>
<td>6 700</td>
<td>980</td>
<td>200</td>
<td>0.461</td>
<td>434</td>
</tr>
<tr>
<td>B 767</td>
<td>250</td>
<td>12 300</td>
<td>885</td>
<td>120</td>
<td>0.221</td>
<td>543</td>
</tr>
<tr>
<td>B 777</td>
<td>375</td>
<td>7 450</td>
<td>980</td>
<td>180</td>
<td>0.368</td>
<td>489</td>
</tr>
<tr>
<td>Concorde</td>
<td>130</td>
<td>6 400</td>
<td>2 200</td>
<td>350</td>
<td>0.286</td>
<td>1224</td>
</tr>
<tr>
<td>DC-8-50</td>
<td>145</td>
<td>14 000</td>
<td>875</td>
<td>80</td>
<td>0.127</td>
<td>630</td>
</tr>
</tbody>
</table>
Concepts of Performance and Speedup

Performance = 1 / Execution time  
Performance = 1 / CPU execution time

\[(\text{Performance of } M_1) / (\text{Performance of } M_2) = \text{Speedup of } M_1 \text{ over } M_2 \]
\[= (\text{Execution time of } M_2) / (\text{Execution time of } M_1)\]

terminology: \(M_1\) is \(x\) times as fast as \(M_2\) (e.g., 1.5 times as fast)
\(M_1\) is \(100(x - 1)\%\) faster than \(M_2\) (e.g., 50% faster)

CPU time = Instructions \(\times\) (Cycles per instruction) \(\times\) (Secs per cycle)
\[= \text{Instructions} \times \text{CPI} / \text{Clock rate}\]

Instruction count, CPI, and clock rate are not completely independent, so improving one by a given factor may not lead to overall execution time improvement by the same factor.
Elaboration on the CPU Time Formula

CPU time = Instructions × (Cycles per instruction) × (Secs per cycle)
= Instructions × Average CPI / (Clock rate)

Instructions: Number of instructions executed, not number of instructions in our program (dynamic count)

Average CPI: Is calculated based on the dynamic instruction mix and knowledge of how many clock cycles are needed to execute various instructions (or instruction classes)

Clock rate: 1 GHz = 10^9 cycles / s (cycle time 10^{-9} s = 1 ns)
200 MHz = 200 × 10^6 cycles / s (cycle time = 5 ns)
Faster Clock ≠ Shorter Running Time

Figure 4.3  Faster steps do not necessarily mean shorter travel time.
4.3 Performance Enhancement: Amdahl’s Law

Figure 4.4 Amdahl’s law: speedup achieved if a fraction $f$ of a task is unaffected and the remaining $1 - f$ part runs $p$ times as fast.

$s = \frac{1}{f + (1 - f)/p} \leq \min(p, 1/f)$
Amdahl’s Law Used in Design

Example 4.1

A processor spends 30% of its time on flp addition, 25% on flp mult, and 10% on flp division. Evaluate the following enhancements, each costing the same to implement:

a. Redesign of the flp adder to make it twice as fast.
b. Redesign of the flp multiplier to make it three times as fast.
c. Redesign the flp divider to make it 10 times as fast.

Solution

a. Adder redesign speedup = \( \frac{1}{0.7 + \frac{0.3}{2}} = 1.18 \)
b. Multiplier redesign speedup = \( \frac{1}{0.75 + \frac{0.25}{3}} = 1.20 \)
c. Divider redesign speedup = \( \frac{1}{0.9 + \frac{0.1}{10}} = 1.10 \)

What if both the adder and the multiplier are redesigned?
Amdahl’s Law Used in Management

Example 4.2

Members of a university research group frequently visit the library. Each library trip takes 20 minutes. The group decides to subscribe to a handful of publications that account for 90% of the library trips; access time to these publications is reduced to 2 minutes.

a. What is the average speedup in access to publications?
b. If the group has 20 members, each making two weekly trips to the library, what is the justifiable expense for the subscriptions? Assume 50 working weeks/yr and $25/h for a researcher’s time.

Solution

a. Speedup in publication access time $= 1 / [0.1 + 0.9 / 10] = 5.26$
b. Time saved $= 20 \times 2 \times 50 \times 0.9 \times (20 - 2) = 32,400 \text{ min} = 540 \text{ h}$
Cost recovery $= 540 \times $25 = $13,500 = \text{Max justifiable expense}$
4.4 Performance Measurement vs Modeling

Execution time

Program

Figure 4.5 Running times of six programs on three machines.
Generalized Amdahl’s Law

Original running time of a program = $1 = f_1 + f_2 + \ldots + f_k$

New running time after the fraction $f_i$ is speeded up by a factor $p_i$

$$\frac{f_1}{p_1} + \frac{f_2}{p_2} + \ldots + \frac{f_k}{p_k}$$

Speedup formula

$$S = \frac{1}{\frac{f_1}{p_1} + \frac{f_2}{p_2} + \ldots + \frac{f_k}{p_k}}$$

If a particular fraction is slowed down rather than speeded up, use $s_j f_j$ instead of $f_j/p_j$, where $s_j > 1$ is the slowdown factor.
Performance Benchmarks

Example 4.3

You are an engineer at Outtel, a start-up aspiring to compete with Intel via its new processor design that outperforms the latest Intel processor by a factor of 2.5 on floating-point instructions. This level of performance was achieved by design compromises that led to a 20% increase in the execution time of all other instructions. You are in charge of choosing benchmarks that would showcase Outtel’s performance edge.

a. What is the minimum required fraction $f$ of time spent on floating-point instructions in a program on the Intel processor to show a speedup of 2 or better for Outtel?

Solution

a. We use a generalized form of Amdahl’s formula in which a fraction $f$ is speeded up by a given factor (2.5) and the rest is slowed down by another factor (1.2): 

$$\frac{1}{1.2(1 - f) + f/2.5} \geq 2 \Rightarrow f \geq 0.875$$
Performance Estimation

Average CPI = \( \sum_{\text{All instruction classes}} (\text{Class-}i \text{ fraction}) \times (\text{Class-}i \text{ CPI}) \)

Machine cycle time = \( \frac{1}{\text{Clock rate}} \)

CPU execution time = \( \text{Instructions} \times \frac{\text{Average CPI}}{\text{Clock rate}} \)

Table 4.3  Usage frequency, in percentage, for various instruction classes in four representative applications.

<table>
<thead>
<tr>
<th>Application → Instr’n class</th>
<th>Data compression</th>
<th>C language compiler</th>
<th>Reactor simulation</th>
<th>Atomic motion modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: Load/Store</td>
<td>25</td>
<td>37</td>
<td>32</td>
<td>37</td>
</tr>
<tr>
<td>B: Integer</td>
<td>32</td>
<td>28</td>
<td>17</td>
<td>5</td>
</tr>
<tr>
<td>C: Shift/Logic</td>
<td>16</td>
<td>13</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>D: Float</td>
<td>0</td>
<td>0</td>
<td>34</td>
<td>42</td>
</tr>
<tr>
<td>E: Branch</td>
<td>19</td>
<td>13</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>F: All others</td>
<td>8</td>
<td>9</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>
CPI and IPS Calculations

Example 4.4 (2 of 5 parts)

Consider two implementations $M_1$ (600 MHz) and $M_2$ (500 MHz) of an instruction set containing three classes of instructions:

<table>
<thead>
<tr>
<th>Class</th>
<th>CPI for $M_1$</th>
<th>CPI for $M_2$</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>5.0</td>
<td>4.0</td>
<td>Floating-point</td>
</tr>
<tr>
<td>I</td>
<td>2.0</td>
<td>3.8</td>
<td>Integer arithmetic</td>
</tr>
<tr>
<td>N</td>
<td>2.4</td>
<td>2.0</td>
<td>Nonarithmetic</td>
</tr>
</tbody>
</table>

a. What are the peak performances of $M_1$ and $M_2$ in MIPS?
b. If 50% of instructions executed are class-N, with the rest divided equally among F and I, which machine is faster? By what factor?

Solution

a. Peak MIPS for $M_1 = 600 / 2.0 = 300$; for $M_2 = 500 / 2.0 = 250$
b. Average CPI for $M_1 = 5.0 / 4 + 2.0 / 4 + 2.4 / 2 = 2.95$;
   for $M_2 = 4.0/4 + 3.8/4 + 2.0/2 = 2.95 \rightarrow M_1$ is faster; factor 1.2
MIPS Rating Can Be Misleading

Example 4.5

Two compilers produce machine code for a program on a machine with two classes of instructions. Here are the number of instructions:

<table>
<thead>
<tr>
<th>Class</th>
<th>CPI</th>
<th>Compiler 1</th>
<th>Compiler 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>600M</td>
<td>400M</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>400M</td>
<td>400M</td>
</tr>
</tbody>
</table>

a. What are run times of the two programs with a 1 GHz clock?
b. Which compiler produces faster code and by what factor?
c. Which compiler’s output runs at a higher MIPS rate?

Solution

a. Running time 1 (2) = (600M × 1 + 400M × 2) / 10⁹ = 1.4 s (1.2 s)
b. Compiler 2’s output runs 1.4 / 1.2 = 1.17 times as fast

c. MIPS rating 1, CPI = 1.4 (2, CPI = 1.5) = 1000 / 1.4 = 714 (667)
4.5 Reporting Computer Performance

Table 4.4  Measured or estimated execution times for three programs.

<table>
<thead>
<tr>
<th></th>
<th>Time on machine X</th>
<th>Time on machine Y</th>
<th>Speedup of Y over X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program A</td>
<td>20</td>
<td>200</td>
<td>0.1</td>
</tr>
<tr>
<td>Program B</td>
<td>1000</td>
<td>100</td>
<td>10.0</td>
</tr>
<tr>
<td>Program C</td>
<td>1500</td>
<td>150</td>
<td>10.0</td>
</tr>
<tr>
<td>All 3 prog’s</td>
<td>2520</td>
<td>450</td>
<td>5.6</td>
</tr>
</tbody>
</table>

Analogy: If a car is driven to a city 100 km away at 100 km/hr and returns at 50 km/hr, the average speed is not \((100 + 50) / 2\) but is obtained from the fact that it travels 200 km in 3 hours.
### Comparing the Overall Performance

**Table 4.4** Measured or estimated execution times for three programs.

<table>
<thead>
<tr>
<th></th>
<th>Time on machine X</th>
<th>Time on machine Y</th>
<th>Speedup of Y over X</th>
<th>Speedup of X over Y</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Program A</strong></td>
<td>20</td>
<td>200</td>
<td>0.1</td>
<td>10</td>
</tr>
<tr>
<td><strong>Program B</strong></td>
<td>1000</td>
<td>100</td>
<td>10.0</td>
<td>0.1</td>
</tr>
<tr>
<td><strong>Program C</strong></td>
<td>1500</td>
<td>150</td>
<td>10.0</td>
<td>0.1</td>
</tr>
</tbody>
</table>

- **Arithmetic mean**
  - 6.7
- **Geometric mean**
  - 2.15

Geometric mean does not yield a measure of overall speedup, but provides an indicator that at least moves in the right direction.
Effect of Instruction Mix on Performance

Example 4.6 (1 of 3 parts)

Consider two applications DC and RS and two machines M₁ and M₂:

<table>
<thead>
<tr>
<th>Class</th>
<th>Data Comp.</th>
<th>Reactor Sim.</th>
<th>M₁’s CPI</th>
<th>M₂’s CPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: Ld/Str</td>
<td>25%</td>
<td>32%</td>
<td>4.0</td>
<td>3.8</td>
</tr>
<tr>
<td>B: Integer</td>
<td>32%</td>
<td>17%</td>
<td>1.5</td>
<td>2.5</td>
</tr>
<tr>
<td>C: Sh/Logic</td>
<td>16%</td>
<td>2%</td>
<td>1.2</td>
<td>1.2</td>
</tr>
<tr>
<td>D: Float</td>
<td>0%</td>
<td>34%</td>
<td>6.0</td>
<td>2.6</td>
</tr>
<tr>
<td>E: Branch</td>
<td>19%</td>
<td>9%</td>
<td>2.5</td>
<td>2.2</td>
</tr>
<tr>
<td>F: Other</td>
<td>8%</td>
<td>6%</td>
<td>2.0</td>
<td>2.3</td>
</tr>
</tbody>
</table>

a. Find the effective CPI for the two applications on both machines.

Solution

a. CPI of DC on M₁: \(0.25 \times 4.0 + 0.32 \times 1.5 + 0.16 \times 1.2 + 0 \times 6.0 + 0.19 \times 2.5 + 0.08 \times 2.0 = 2.31\)

DC on M₂: 2.54
RS on M₁: 3.94
RS on M₂: 2.89
4.6 The Quest for Higher Performance

State of available computing power ca. the early 2000s:

Gigaflops on the desktop
Teraflops in the supercomputer center
Petaflops on the drawing board

Note on terminology (see Table 3.1)

Prefixes for large units:
Kilo = $10^3$, Mega = $10^6$, Giga = $10^9$, Tera = $10^{12}$, Peta = $10^{15}$

For memory:
$K = 2^{10} = 1024, \ M = 2^{20}, \ G = 2^{30}, \ T = 2^{40}, \ P = 2^{50}$

Prefixes for small units:
Micro = $10^{-6}$, Nano = $10^{-9}$, Pico = $10^{-12}$, Femto = $10^{-15}$
Performance Trends and Obsolescence

Figure 3.10  Trends in processor performance and DRAM memory chip capacity (Moore’s law).

“Can I call you back? We just bought a new computer and we’re trying to set it up before it’s obsolete.”
Figure 4.7  Exponential growth of supercomputer performance.
Figure 4.8 Milestones in the DOE’s Accelerated Strategic Computing Initiative (ASCI) program with extrapolation up to the PFLOPS level.
Performance is Important, But It Isn’t Everything

Figure 25.1 Trend in computational performance per watt of power used in general-purpose processors and DSPs.