What Affects Cost?

1. Learning curve:
   - The more experience in manufacturing a component, the better the yield
   - In general, a chip, board or system with twice the yield will have half the cost.
   - The learning curve is different for different components, complicating design decisions

2. Volume
   - Larger volume increases rate of learning curve
   - Doubling the volume typically reduce cost by 10%

3. Commodities
   - Are essentially identical products sold by multiple vendors in large volumes
   - Foil the competition and drive the efficiency higher and thus the cost down
A dollar in 1977 = $2.95 in 2001
Cost/MB = $500 in 1997
= $0.35 in 2000
= $0.08 in 2001

Demand exceeded supply ➔ price slow drop

Each generation drops in price by a factor of 10 to 30 over its lifetime
Cost Trends for Processors

Price drop due yield enhancements
### Cost vs. Price

<table>
<thead>
<tr>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>List Price</td>
<td></td>
</tr>
<tr>
<td>Avg. Selling Price</td>
<td></td>
</tr>
<tr>
<td>Average Discount</td>
<td>25% to 40%</td>
</tr>
<tr>
<td>Gross Margin</td>
<td>34% to 39%</td>
</tr>
<tr>
<td>Direct Cost</td>
<td>6% to 8%</td>
</tr>
<tr>
<td>Component Cost</td>
<td>15% to 33%</td>
</tr>
</tbody>
</table>

- **Component Costs**: raw material cost for the system’s building blocks
- **Direct Costs** (add 25% to 40%) recurring costs: labor, purchasing, scrap, warranty
- **Gross Margin** (add 82% to 186%) nonrecurring costs: R&D, marketing, sales, equipment maintenance, rental, financing cost, pretax profits, taxes
- **Average Discount** to get List Price (add 33% to 66%): volume discounts and/or retailer markup
Chip Prices (August 1993) for a volume of 10,000 units

<table>
<thead>
<tr>
<th>Chip</th>
<th>Area (mm²)</th>
<th>Total Cost</th>
<th>Price</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>386DX</td>
<td>43</td>
<td>$9</td>
<td>$31</td>
<td></td>
</tr>
<tr>
<td>486DX2</td>
<td>81</td>
<td>$35</td>
<td>$245</td>
<td>No Competition</td>
</tr>
<tr>
<td>PowerPC 601</td>
<td>121</td>
<td>$77</td>
<td>$280</td>
<td></td>
</tr>
<tr>
<td>DEC Alpha</td>
<td>234</td>
<td>$202</td>
<td>$1231</td>
<td>Recoup R&amp;D?</td>
</tr>
<tr>
<td>Pentium</td>
<td>296</td>
<td>$473</td>
<td>$965</td>
<td></td>
</tr>
</tbody>
</table>
The Role of Performance

- Hardware performance is a key to the effectiveness of the entire system
- Performance has to be measured and compared to evaluate various design and technological approaches
- To optimize the performance, major affecting factors have to be known
- For different types of applications, different performance metrics may be appropriate and different aspects of a computer system may be most significant
- Instructions use and implementation, memory hierarchy and I/O handling are among the factors that affect the performance
Defining Performance

• Performance means different things to different people, therefore its assessment is subtle.

Analogy from the airlines industry:

• How to measure performance for an airplane?
  – Cruising speed (How fast it gets to the destination)
  – Flight range (How far it can reach)
  – Passenger capacity (How many passengers it can carry)

<table>
<thead>
<tr>
<th>Airplane</th>
<th>Passenger capacity</th>
<th>Cruising range (miles)</th>
<th>Cruising speed (m.p.h)</th>
<th>Passenger throughput (Passenger × m.p.h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boeing 777</td>
<td>375</td>
<td>4630</td>
<td>610</td>
<td>228,750</td>
</tr>
<tr>
<td>Boeing 747</td>
<td>470</td>
<td>4150</td>
<td>610</td>
<td>286,700</td>
</tr>
<tr>
<td>BAC/Sud Concorde</td>
<td>132</td>
<td>4000</td>
<td>1350</td>
<td>178,200</td>
</tr>
<tr>
<td>Douglas DC-8-50</td>
<td>146</td>
<td>8720</td>
<td>544</td>
<td>79,424</td>
</tr>
</tbody>
</table>

Criteria of performance evaluation differs among users and designers.
Performance Metrics

• Response (execution) time:
  – The time between the start and the completion of a task
  – Measures user perception of the system speed
  – Common in reactive and time critical systems, single-user computer, etc.

• Throughput:
  – The total number of tasks done in a given time
  – Most relevant to batch processing (billing, credit card processing)
  – Mainly used for input/output systems (disk access, printer, etc.)

• Example
  – Faster processor
    ⇒ Enhances both response time and throughput
  – Adding additional processors to parallelize separate tasks
    ⇒ Enhances only throughput

Decreasing response time always improves throughput
Response-time Metric

- Maximizing performance means minimizing response (execution) time

\[
\text{Performance} = \frac{1}{\text{Execution time}}
\]
Response-time Metric

\[
\text{Performance} = \frac{1}{\text{Execution time}}
\]

- Performance of Processor P1 is better than P2 if
  - For a given work load L
  - P1 takes less time to execute L than P2

\[
\text{Performance (P}_1) > \text{Performance (P}_2) \text{ w.r.t L} \\
\Rightarrow \text{Execution time (P}_1,\text{L}) < \text{Execution time (P}_2,\text{L})
\]
Response-time Metric

\[
\text{Performance} = \frac{1}{\text{Execution time}}
\]

• Relative performance captures the performance ratio
  – For the same work load

\[
\frac{\text{CPU Performance (P}_2\text{)}}{\text{CPU Performance (P}_1\text{)}} = \frac{\text{Total execution time (P}_1\text{)}}{\text{Total execution time (P}_2\text{)}}
\]
**Designer’s Performance Metrics**

- Users and designers measure performance using different metrics
  - Users: quotable metrics (GHz)
  - Designers: program execution

\[
\text{CPU execution time for a program} = \frac{\text{CPU clock cycles for a program} \times \text{Clock cycle time}}{\text{Clock rate}}
\]

- Designer focuses on reducing the clock cycle time and the number of cycles per program
- Many techniques to decrease the number of clock cycles also increase the clock cycle time or the average number of cycles per instruction (CPI)
Example

A program runs in 10 seconds on a computer “A” with a 400 MHz clock. We desire a faster computer “B” that could run the program in 6 seconds. The designer has determined that a substantial increase in the clock speed is possible, however it would cause computer “B” to require 1.2 times as many clock cycles as computer “A”. What should be the clock rate of computer “B”?

CPU time (A) = \( \frac{\text{CPU clock cycles}}{\text{Clock rate (A)}} \)

10 seconds = \( \frac{\text{CPU clock cycles of program}}{400 \times 10^6 \text{ cycles/second}} \)

CPU clock cycles of program = 10 seconds \( \times \) 400 \( \times \) 10\(^6\) cycles/second

= 4000 \( \times \) 10\(^6\) cycles

To get the clock rate of the faster computer, we use the same formula

6 seconds = \( \frac{1.2 \times \text{CPU clock cycles of program}}{\text{clock rate (B)}} \) = \( \frac{1.2 \times 4000 \times 10^6 \text{ cycles}}{\text{clock rate (B)}} \)

clock rate (B) = \( \frac{1.2 \times 4000 \times 10^6 \text{ cycles}}{6 \text{ second}} \) = 800 \( \times \) 10\(^6\) cycles/second
Calculation of CPU Time

CPU time = Instruction count × CPI × Clock cycle time

Or

\[
\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}
\]

\[
\text{CPU time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}
\]

<table>
<thead>
<tr>
<th>Component of performance</th>
<th>Units of measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU execution time for a program</td>
<td>Seconds for the program</td>
</tr>
<tr>
<td>Instruction count</td>
<td>Instructions executed for the program</td>
</tr>
<tr>
<td>Clock cycles per instructions (CPI)</td>
<td>Average number of clock cycles/instruction</td>
</tr>
<tr>
<td>Clock cycle time</td>
<td>Seconds per clock cycle</td>
</tr>
</tbody>
</table>
CPU Time (Cont.)

- CPU execution time can be measured by running the program
- The clock cycle is usually published by the manufacture
- Measuring the CPI and instruction count is not trivial
  - Instruction counts can be measured by: a software profiling, using an architecture simulator, using hardware counters on some architecture
  - The CPI depends on many factors including: processor structure, memory system, the mix of instruction types and the implementation of these instructions
CPU Time (Cont.)

- Designers sometimes uses the following formula:

\[
\text{CPU clock cycles} = \sum_{i=1}^{n} CPI_i \times C_i
\]

Where:
- \( C_i \) is the count of number of instructions of class \( i \) executed
- \( CPI_i \) is the average number of cycles per instruction for that instruction class
- \( n \) is the number of different instruction classes
Example

Suppose we have two implementation of the same instruction set architecture. Machine “A” has a clock cycle time of 1 ns and a CPI of 2.0 for some program, and machine “B” has a clock cycle time of 2 ns and a CPI of 1.2 for the same program. Which machine is faster for this program and by how much?

Both machines execute the same instructions for the program. Assume the number of instructions is “I”,

CPU clock cycles (A) = I × 2.0
CPU clock cycles (B) = I × 1.2

The CPU time required for each machine is as follows:

CPU time (A) = CPU clock cycles (A) × Clock cycle time (A)
= I × 2.0 × 1 ns = 2 × I ns

CPU time (B) = CPU clock cycles (B) × Clock cycle time (B)
= I × 1.2 × 2 ns = 2.4 × I ns

Therefore machine A will be faster by the following ratio:

\[
\frac{CPU \text{ Performance (A)}}{CPU \text{ Performance (B)}} = \frac{CPU \text{ time (B)}}{CPU \text{ time (A)}} = \frac{2.4 \times I \text{ ns}}{2 \times I \text{ ns}} = 1.2
\]
Comparing Code Segments

A compiler designer is trying to decide between two code sequences for a particular machine. The hardware designers have supplied the following facts:

<table>
<thead>
<tr>
<th>Instruction class</th>
<th>CPI for this instruction class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
</tr>
</tbody>
</table>

For a particular high-level language statement, the compiler writer is considering two code sequences that require the following instruction counts:

<table>
<thead>
<tr>
<th>Code sequence</th>
<th>Instruction count for instruction class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Which code sequence executes the most instructions? Which will be faster? What is the CPI for each sequence?

Answer:

Sequence 1: executes $2 + 1 + 2 = 5$ instructions
Sequence 2: executes $4 + 1 + 1 = 6$ instructions
Comparing Code Segments

Using the formula: \( \text{CPU clock cycles} = \sum_{i=1}^{n} CPI_i \times C_i \)

Sequence 1: CPU clock cycles = \((2 \times 1) + (1 \times 2) + (2 \times 3)\) = 10 cycles
Sequence 2: CPU clock cycles = \((4 \times 1) + (1 \times 2) + (1 \times 3)\) = 9 cycles

Therefore Sequence 2 is faster although it executes more instructions

Using the formula: \( \text{CPI} = \frac{\text{CPU clock cycles}}{\text{Instruction count}} \)

Sequence 1: CPI = \(\frac{10}{5}\) = 2
Sequence 2: CPI = \(\frac{9}{6}\) = 1.5

Since Sequence 2 takes fewer overall clock cycles but has more instructions it must have a lower CPI
The Role of Performance

- Hardware performance is a key to the effectiveness of the entire system
- Performance has to be measured and compared to evaluate designs
- To optimize the performance, major affecting factors have to be known
- For different types of applications
  - different performance metrics may be appropriate
  - different aspects of a computer system may be most significant
- Instructions use and implementation, memory hierarchy and I/O handling are among the factors that affect the performance
Maximizing performance means minimizing response (execution) time.

**Performance** = \( \frac{1}{\text{Execution time}} \)
## Calculation of CPU Time

The CPU time can be calculated using the formula:

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

### Table: Instr. Count, CPI, Clock Rate

<table>
<thead>
<tr>
<th></th>
<th>Instr. 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>Instruction Set</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Organization</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Technology</td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

### CPU clock cycles

The CPU clock cycles can be calculated as:

$$\text{CPU clock cycles} = \sum_{i=1}^{n} CPI_i \times C_i$$

Where:
- $C_i$ is the count of number of instructions of class $i$ executed
- $CPI_i$ is the average number of cycles per instruction for that instruction class
- $n$ is the number of different instruction classes