Performance

Same handout as Friday
Not all computers are created equal!
How do we evaluate computer architectures?

- Think of 5 characteristics that differentiate computers?

1. price
2. I/O devices available
3. performance & special-purpose
4. programming model
5. memory addressability, size
6. power
7. physical size
8. reliability
Can some computer processors compute things that others can’t?

a) Yes

b) No
Two notions of performance

<table>
<thead>
<tr>
<th>Aircraft</th>
<th>DC to Paris</th>
<th>Passengers</th>
</tr>
</thead>
<tbody>
<tr>
<td>747</td>
<td>6 hours</td>
<td>500</td>
</tr>
<tr>
<td>Concorde</td>
<td>3 hours</td>
<td>125</td>
</tr>
</tbody>
</table>

- Which has higher performance?

- From a passenger’s viewpoint: latency (time to do the task)
  - hours per flight, execution time, response time

- From an airline’s viewpoint: throughput (tasks per unit time)
  - passengers per hour, bandwidth

- Latency and throughput are often in opposition
Some definitions for performance

- Relative performance: “x is N times faster than y”

\[
\text{Speedup} = \frac{\text{performance}(x)}{\text{performance}(y)}
\]

- If we are primarily concerned with latency,

\[
\frac{\text{perf}(x)}{\text{perf}(y)} = \frac{\frac{1}{\text{latency}(x)}}{\frac{1}{\text{latency}(y)}} = \frac{\text{latency}(y)}{\text{latency}(x)} \quad \leftarrow \text{lower is better}
\]

- If we are primarily concerned with throughput,

\[
\frac{\text{throughput}(x)}{\text{throughput}(y)} = \frac{\text{throughput}(x)}{\text{throughput}(y)} \quad \leftarrow \text{higher is better}
\]
The Iron Law of Computing: The time it takes to run a program depends on three factors

1. The number of dynamic instructions $N$ in the program
   - Executing more instructions tends to take longer.

2. The kind of instructions in the program
   - Some instructions take more CPU cycles than others
   - Let $c$ be the average number of cycles per instruction (CPI)

3. The time $t$ per CPU clock cycle (clock-cycle time)
   \[
   \text{CPU time} = \text{Instructions executed} \times \text{CPI} \times \text{Clock cycle time}
   \]

   \[
   \text{Seconds/Program} = \frac{\text{Instructions/Program} \times \text{Clock cycles/Instruction} \times \text{Seconds/Clock cycle}}{}
   \]
Iron Law Component (1) **Dynamic instruction Count** is determined during runtime

Not the number of lines of code!!!!

```c
for (int i = 0; i < N; i++) {
    // 10 lines of code;
}
```

**Static lines of code:** 11

**Dynamic instructions:** \(~12*N + 1\)
Iron Law Component (2) The **Average Cycles per instruction** depends on both the machine and the program

- Example: \( \text{CPI(floating-point operations)} > \text{CPI(integer operations)} \)

- Example: Improved processor may execute same instructions in fewer cycles

- Single-cycle machine: each instruction takes at least 1 cycle (CPI = 1)
  - CPI can be > 1 due to memory stalls and slow instructions
  - CPI can be < 1 on **superscalar** machines
Iron Law Component (3) **Clock Cycle Time** is determined by the worst-case path delays between clocked state elements

- 1 cycle = minimum time it takes the CPU to do any work

- clock cycle time = \( \frac{1}{\text{clock frequency}} \)
  - Example: 500MHz processor has a cycle time of 2ns (nanoseconds)
  - Example: 2GHz (2000MHz) CPU has a cycle time of just 0.5ns

- Higher frequency is usually better for performance
How do we improve performance for each component of the Iron Law of Computing?

CPU time = Instructions executed × CPI × Clock cycle time

- We can improve performance by making any component smaller

<table>
<thead>
<tr>
<th>Program</th>
<th>Compiler</th>
<th>ISA</th>
<th>Organization</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction Executed</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CPI</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Clock Cycle Time</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a)</th>
<th>b) X</th>
<th>c) X</th>
<th>d) X</th>
<th>e) _</th>
<th>i&gt;clicker</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>_</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
Example 1: ISA-compatible processors

- Let’s compare the performances two x86-based processors.
  - An 800MHz AMD Duron, with a CPI of 1.2 for an MP3 compressor.
  - A 1GHz Pentium III with a CPI of 1.5 for the same program.

- Compatible processors implement identical instruction sets and will use the same executable files, with the same number of instructions.
- But they implement the ISA differently, which leads to different CPIs.

\[
\text{CPU time}_{\text{AMD},P} = \text{Instructions}_P \times \text{CPI}_{\text{AMD},P} \times \text{Cycle time}_{\text{AMD}} = \frac{3}{200}
\]

\[
\text{CPU time}_{\text{P3},P} = \text{Instructions}_P \times \text{CPI}_{\text{P3},P} \times \text{Cycle time}_{\text{P3}} = \frac{3}{200}
\]
Example 2: Comparing across ISAs

- Intel’s Itanium (IA-64) ISA is designed facilitate executing multiple instructions per cycle. If an Itanium processor achieves an average CPI of .3 (3 instructions per cycle), how much faster is it than a Pentium4 (which uses the x86 ISA) with an average CPI of 1?

a) Itanium is three times faster  
b) Itanium is one third as fast  
c) Not enough information

*Same frequency*