SPEC 2000 FAQ

• What is SPEC CPU2000?
  • A non-profit group that includes computer vendors, systems integrators, universities and consultants from around the world.

• What do CINT2000 and CFP2000 measure?
  • Being compute-intensive benchmarks, they measure performance of the
    • (1) computer's processor,
    • (2) memory architecture and
    • (3) compiler.
  • It is important to remember the contribution of the latter two components -- performance is more than just the processor.

• What is not measured?
  • The CINT2000 and CFP2000 benchmarks do not stress: I/O (disk drives), networking or graphics.

Reference: http://www.specbench.org/
### SPECint2000 (Number of processors = 1)

<table>
<thead>
<tr>
<th>Company System</th>
<th>Clock, CPU</th>
<th>SPEC</th>
<th>L2 cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dell Precision Ws 330</td>
<td>1.50 GHz P4</td>
<td>526</td>
<td>256KB(I+D)</td>
</tr>
<tr>
<td>Dell Precision Ws 330</td>
<td>1.40 GHz P4</td>
<td>505</td>
<td>256KB(I+D)</td>
</tr>
<tr>
<td>Intel VC820</td>
<td>1.13 GHz P3</td>
<td>464</td>
<td>256KB(I+D)</td>
</tr>
<tr>
<td>SGI SGI 2200 2X</td>
<td>400MHz R12k</td>
<td>347</td>
<td>8M(I+D)</td>
</tr>
<tr>
<td>Intel SE440BX-2</td>
<td>800 MHz P3</td>
<td>344</td>
<td>256KB(I+D)</td>
</tr>
<tr>
<td>Intel SE440BX-2</td>
<td>750 MHz P3</td>
<td>330</td>
<td>256KB(I+D)</td>
</tr>
<tr>
<td>SGI Origin200</td>
<td>360MHz R12k</td>
<td>298</td>
<td>4M(I+D)</td>
</tr>
</tbody>
</table>

**Pitfall: Using MIPS or Clock speed as performance metric**
"The Doom benchmark is more important than SPEC"
(paraphrased) John Hennessy in his plenary talk at FCRC '99.

<table>
<thead>
<tr>
<th>avg. fps</th>
<th>Processor</th>
<th>L1 Cache</th>
<th>Mother Board</th>
</tr>
</thead>
<tbody>
<tr>
<td>304.3</td>
<td>MIPS R4400-250</td>
<td>16+16k</td>
<td>SGI Indigo2</td>
</tr>
<tr>
<td>201.9</td>
<td>PentiumIIIE-800</td>
<td>16+16K</td>
<td>ASUS P3B-F</td>
</tr>
<tr>
<td>197.1</td>
<td>PentiumIIIE-787</td>
<td>16+16K</td>
<td>Abit BH6R1.01</td>
</tr>
<tr>
<td>196.0</td>
<td>MIPS R10000-195</td>
<td>32+32k</td>
<td>SGI Indigo2</td>
</tr>
<tr>
<td>190.5</td>
<td>PentiumIII-644</td>
<td>16+16K</td>
<td>Abit BX6 2,0</td>
</tr>
<tr>
<td>188.1</td>
<td>PentiumIII-800</td>
<td>16+16K</td>
<td>ASUS CU4VX</td>
</tr>
</tbody>
</table>

Wow! 250 Mhz MIPS beats the 800 Mhz Pentium.

avg. fps The average number of video frames per second

Reference: http://www.complang.tuwien.ac.at/misc/doombench.html
Doom,Quake games: http://www.idsoftware.com
Benchmark wars: Internet Servers

Sm@rt Reseller's January 1999 article, "Linux Is The Web Server’s Choice” said "Linux with Apache beats NT 4.0 with IIS, hands down."

In March 1999, Microsoft commissioned Mindcraft to carry out a comparison between NT and Linux.
Benchmark Wars: Linux/Solaris

PC Magazine, September 1999

Sun Microsystems SPARC architecture now jumps in!

...found that NT did a lot more disk accesses than Linux, which let Linux score about 50% better than NT.
Performance

To maximize performance, we want to minimize response time or execution time

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

To compare the relative performance, \( n \), between machine X and Y, we use

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

**Execution time** = \( \frac{\text{Total program clock cycles executed}}{\text{Clock frequency rate (MHz)}} \)

\( \text{Total program instructions exec x CPI} \) = \( \frac{\text{Total program clock cycles executed}}{\text{Clock frequency rate (MHz)}} \)

**CPI** = Average number of clock cycles per instruction

**Clock cycle time (us)** = \( \frac{1}{\text{Clock frequency rate (Mhz)}} \)
CPI Example

Given the following instruction class execution times:

\[
\text{alu} = 6\text{ns}, \text{loads} = 8\text{ns}, \text{stores} = 7\text{ns}, \text{branches} = 5\text{ns}, \text{jumps} = 2\text{ns}
\]

\[
\text{CPI} = \frac{(6\text{ns} + 8\text{ns} + 7\text{ns} + 5\text{ns} + 2\text{ns})}{5} = \frac{28}{5} = 5.6 \text{ ns}
\]

\[
= (0.2 \times 6\text{ns} + 0.2 \times 8\text{ns} + 0.2 \times 7\text{ns} + 0.2 \times 5\text{ns} + 0.2 \times 2\text{ns}) = 5.6 \text{ ns}
\]

Given the following instruction class execution times:

\[
\text{alu} = 60\%, \text{loads} = 20\%, \text{stores} = 10\%, \text{branches} = 5\%, \text{jumps} = 5\%
\]

\[
\text{alu} = 6\text{ns}, \text{loads} = 8\text{ns}, \text{stores} = 7\text{ns}, \text{branches} = 5\text{ns}, \text{jumps} = 2\text{ns}
\]

\[
\text{CPI} = (0.6 \times 6\text{ns} + 0.2 \times 8\text{ns} + 0.1 \times 7\text{ns} + 0.05 \times 5\text{ns} + 0.05 \times 2\text{ns}) = 6.25
\]
Performance example

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>A</th>
<th>B</th>
<th>L</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>=5</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>=6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instruction class</th>
<th>CPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>1</td>
</tr>
<tr>
<td>Branches</td>
<td>2</td>
</tr>
<tr>
<td>Load/Stores</td>
<td>3</td>
</tr>
</tbody>
</table>

Total CPU cycles\(_1\) = (2xA) + (1xB) + (2xL)
= (2x1) + (1x2) + (2x3) = 10 cycles

\[\text{CPI}_1 = \frac{10 \text{ cycles}}{5} = 2 \text{ average cycles per instruction}\]

Total CPU cycles\(_2\) = (4x1) + (1x2) + (1x3) = 9 cycles

\[\text{CPI}_2 = \frac{9 \text{ cycles}}{6} = 1.5 \text{ average cycles per instruction}\]

• Benchmark 2 executed more instructions, but was faster.
### MIPS Performance Example

(PH page 78)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>A</th>
<th>B</th>
<th>L</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiler 1</td>
<td>$5 \times 10^9$</td>
<td>$10^9$</td>
<td>$10^9$</td>
<td>$7 \times 10^9$</td>
</tr>
<tr>
<td>Compiler 2</td>
<td>$10^{10}$</td>
<td>$10^9$</td>
<td>$10^9$</td>
<td>$12 \times 10^9$</td>
</tr>
</tbody>
</table>

**Instruction class**

<table>
<thead>
<tr>
<th></th>
<th>CPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>1</td>
</tr>
<tr>
<td>Branches</td>
<td>2</td>
</tr>
<tr>
<td>Load/Stores</td>
<td>3</td>
</tr>
</tbody>
</table>

**Total CPU cycles**

- $\text{Total CPU cycles}_1 = (5 \times A) + (1 \times B) + (1 \times L) = 10 \times 10^9$ cycles
- $\text{Total CPU cycles}_2 = (10 \times A) + (1 \times B) + (1 \times L) = 15 \times 10^9$ cycles

**Execution time**

- $\text{Execution time}_1 = 10 \times 10^9$ cycles / 500 MHz = 20 seconds
- $\text{Execution time}_2 = 15 \times 10^9$ cycles / 500 MHz = 30 seconds

**CPI**

- $\text{CPI}_1 = 10 \times 10^9$ cycles / $7 \times 10^9 = 1.43$
- $\text{CPI}_2 = 15 \times 10^9$ cycles / $12 \times 10^9 = 1.25$

**MIPS**

- $\text{MIPS}_1 = \frac{500 \text{ Mhz}}{1.43} = 350$ MIPS
- $\text{MIPS}_2 = \frac{500 \text{ Mhz}}{1.25} = 400$ MIPS

Although $\text{MIPS}_2 > \text{MIPS}_1$ but execution time is unexpected!
Amdahl’s Law (the law of diminishing returns)

Execution Time After Improvement
= Execution Time Unaffected
+ (Execution Time Affected / Amount of Improvement)

Example:

"Suppose a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time.

How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?"

How about making it 5 times faster?

Principle: Make the common case fast
Well, let’s speed up the multiply!
Amdahl’s Law (the law of dimishing returns)

Execution Time After Improvement =
(Execution Time Affected / Amount of Improvement) + Execution Time Unaffected

Let Execution Time After Improvement be
old time / speed up =
100 seconds / 5 times faster = 20 seconds =

Execution Time needed
= 80 seconds/n + (100-80 seconds)

Equating both sides
20 = 80 seconds/n + (100-80 seconds)
0 = 80 seconds/n

No amount of multiplier speed up can make a 5 fold increase
Sources of improvement

- For a given instruction set architecture,
  - increases in CPU performance can come from three sources
    - 1. Increase the clock rate
    - 2. Improve the hardware organization that lower the CPI
    - 3. Compiler enhancements that
      - lower the instruction count or
      - generate instructions with a lower average CPI

- In addition to the above, in order to improve CPU efficiency of software benchmarks.
  - Improve the software organization (data structures, …)
Performance Summary

- **Execution time** is the only valid and unimpeachable measure of performance.

- Any measure that summarizes performance should reflect execution time.

- Designers must balance **high-performance with low-cost**.

- You should not always believe everything you read! Read carefully! (see newspaper articles, e.g., Exercise 2.37)