All depends on the program.

Floating point operations take 20 cycles, which runs at 1.2 GHz by default, the unit, meaning all floating point operations take 1 cycle. Groupe Orange runs at 1 GHz, and provides a unit making machine is better?

Review Problem 15
CPU execution time for a program = Instructions for a program * CPI * \( \frac{1}{\text{Clock rate}} \)

CPU clock cycles for a program = Instructions for a program * Cycles per Instruction (CPI)

How do the # of instructions in a program relate to the execution time?
What machine is faster for this program, and by how much?

Machine B has a clock cycle time of 20 ns and a CPI of 1.2.

Machine A has a clock cycle time of 10 ns and a CPI of 2.0.

For some program

Suppose we have two implementations of the same instruction set (ISA).
<table>
<thead>
<tr>
<th>CPI</th>
<th>ν̄h = 0.4</th>
<th>ν̄s = 0.3</th>
<th>ν̄l = 0.1</th>
<th>ν̄a = 0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2</td>
<td>2%</td>
<td>10%</td>
<td>20%</td>
<td>50%</td>
</tr>
<tr>
<td></td>
<td>Branch</td>
<td>Store</td>
<td>Load</td>
<td>ALU</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Type Cycles</th>
<th>Type Frequency</th>
<th>Cycles * Frequency</th>
<th>Type Cycles * Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ CPI = \sum_{\text{Types}} \left( \text{Cycles} \times \text{Frequency} \right) \]

Memory access, integer math, floating point, control flow

Different types of instructions can take very different amounts of cycles.
1. A data cache reduced the average load time to 2 cycles. How much faster would the machine be if:

<table>
<thead>
<tr>
<th>Type</th>
<th>Frequency</th>
<th>Type Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch</td>
<td>20%</td>
<td>2</td>
</tr>
<tr>
<td>Store</td>
<td>10%</td>
<td>3</td>
</tr>
<tr>
<td>Load</td>
<td>20%</td>
<td>5</td>
</tr>
<tr>
<td>ALU</td>
<td>60%</td>
<td>1</td>
</tr>
</tbody>
</table>

2. Branch prediction shaved a cycle off the branch time.

3. Two ALU instructions could be executed at once.

\[
\frac{0.5 \times 5 + 1.0 + 0.3 + 0.4}{2.2} \times 1.13 = \frac{1.95}{2.2} = 0.9
\]

\[\frac{2.0}{2.2} = 0.9\]

\[\frac{1.6}{2.2} = 0.75\]
\[ \frac{\text{improve}}{80} = 0 \]

\[ \text{improve} = 0 \]

\[ \frac{\text{improve}}{1} = \frac{0}{80} \]

\[ 20 = 20 + 80 \times \frac{\text{improve}}{1} \]

5 times faster?

Example: Assume a program runs in 100 seconds on a machine, with multiplicity responsible for 80 seconds of this time. How much do we have to speed up?

\[ \frac{\text{improve}}{80} = 5 \]

\[ 25 = 20 + 80 \times \frac{\text{improve}}{1} \]

\[ \frac{1}{100} = \frac{n}{4} \]

\[ n = 4 \]

Multiply to make the program run 4 times faster?

The impact of a performance improvement is limited by what is NOT improved:

\[ \text{Execution time} = \frac{\text{Amount of improvement}}{1} \]

* affects affected

\[ \frac{\text{Execution time} \times \text{time}}{\text{Execution time} + \text{Execution time}} = \frac{\text{improve}}{1} \]

\[ \text{Warning: Amdahl's Law} \]
<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Type Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch</td>
<td>2</td>
</tr>
<tr>
<td>Store</td>
<td>3</td>
</tr>
<tr>
<td>Load</td>
<td>5</td>
</tr>
<tr>
<td>ALU</td>
<td>1</td>
</tr>
</tbody>
</table>

Compiler B on Program X: 5 Billion ALU, 1 Billion Load
Compiler A on Program X: 10 Billion ALU, 1 Billion Load

1 GHz machine, with two different compilers

Higher MIPS (millions instructions per second) doesn’t always mean better CPU

Execution Time: A: 15 B: 10

Compiler B on Program X: 5 Billion ALU, 1 Billion Load

Compiler A on Program X: 10 Billion ALU, 1 Billion Load

1 GHz machine, with two different compilers

Higher MIPS (millions instructions per second) doesn’t always mean better CPU

Execution Time: A: 15 B: 10

Example: 1.25 x CPI x Instruction Program = 1 x 10^9 x 1.1 x 0.5 x 10^9 = 2.2 seconds

Example: 1.25 x CPI x Instruction Program = 1 x 10^9 x 2.5 x 10^9 = 2.5 seconds

Example: 1.25 x CPI x Instruction Program = 1 x 10^9 x 2.5 x 10^9 = 2.5 seconds

Warning: 2: BIPS, GHz ≠ Performance
Example: RISC vs. CISC

Improving performance must balance each constraint

Clock Rate $\downarrow$  

$\Rightarrow$ CPI $\downarrow$

number of instructions to implement computations $\uparrow$

Better performance:

$\frac{Clock Rate}{CPI} = \frac{Instructions}{CPU execution time}$

Machine Performance:

Processor Performance Summary