Defining Performance

Which airplane has the best performance?

- **Passenger Capacity**
  - Boeing 777
  - Boeing 747
  - BAC/Sud Concorde
  - Douglas DC-8-50

- **Cruising Range (miles)**
  - Boeing 777
  - Boeing 747
  - BAC/Sud Concorde
  - Douglas DC-8-50

- **Cruising Speed (mph)**
  - Boeing 777
  - Boeing 747
  - BAC/Sud Concorde
  - Douglas DC-8-50

- **Passengers x mph**
  - Boeing 777
  - Boeing 747
  - BAC/Sud Concorde
  - Douglas DC-8-50
Response Time (latency): how long it takes to perform a task

Throughput: total work done per unit time

For now, we will focus on response time…

\[
\text{Performance} = \frac{1}{\text{Execution Time}}
\]
Elapsed Time
- counts everything (disk and memory accesses, I/O, etc.)
- a useful number, but often not good for comparison purposes

CPU time
- doesn't count I/O or time spent running other programs
- can be broken up into system time, and user time

Our focus: user CPU time
- time spent executing the lines of code that are "in" our program
Relative Performance

Performance = \frac{1}{\text{Execution Time}}

Relative Performance = \frac{\text{Performance}_X}{\text{Performance}_Y}
= \frac{\text{Execution Time}_Y}{\text{Execution Time}_X}

Example: time taken to run a program
- 10s on A, 15s on B
- Execution Time_B / Execution Time_A
  = 15s / 10s = 1.5
- So A is 1.5 times faster than B
Operation of digital hardware are governed by a constant-rate clock.

Clock period: duration of a clock cycle
- e.g., 250ps = 0.25ns = 250×10^{-12}s

Clock frequency (rate): cycles per second
- e.g., 4.0GHz = 4000MHz = 4.0×10^9Hz
CPU Time

\[
\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}
\]

\[
= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}
\]

Performance improved by

- Reducing number of clock cycles
- Increasing clock rate
- Hardware designer must often trade off clock rate against cycle count
CPU Time Example

Computer A: 2GHz clock, 10s CPU time
Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles

How fast must Computer B clock be?

\[
\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6s}
\]

\[
\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A
= 10s \times 2GHz = 20 \times 10^9
\]

\[
\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6s} = \frac{24 \times 10^9}{6s} = 4GHz
\]
Could assume that number of cycles equals number of instructions:

This assumption is incorrect,

different instructions take different amounts of time on different machines.

Why? hint: remember that these are machine instructions, not lines of C code
A given program will require
- some number of instructions (machine instructions)
- some number of cycles
- some number of seconds

We have a vocabulary that relates these quantities:
- cycle time (seconds per cycle)
- clock rate (cycles per second)
- CPI (cycles per instruction)
  
  a floating point intensive application might have a higher CPI

- MIPS (millions of instructions per second)
  
  this would be higher for a program using simple instructions
Performance

Performance is determined by execution time

Do any of the other variables equal performance?
  - # of cycles to execute program?
  - # of instructions in program?
  - # of cycles per second?
  - average # of cycles per instruction?
  - average # of instructions per second?

Common pitfall: thinking one of the variables is indicative of performance when it really isn’t.
CPI Example

Computer A: Cycle Time = 250ps, CPI = 2.0
Computer B: Cycle Time = 500ps, CPI = 1.2
Same ISA
Which is faster, and by how much?

\[
\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A
\]
\[
= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}
\]

\[
\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B
\]
\[
= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}
\]

\[
\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2
\]

A is faster...

...by this much
CPI in More Detail

If different instruction classes take different numbers of cycles

\[ \text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i) \]

Weighted average CPI

\[ \text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left( \text{CPI}_i \times \frac{\text{Instruction Count}_i}{\text{Instruction Count}} \right) \]

Relative frequency
CPI Example

Alternative compiled code sequences using instructions in classes A, B, C

<table>
<thead>
<tr>
<th>Class</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPI for class</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>IC in sequence 1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>IC in sequence 2</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Sequence 1: IC = 5
- Clock Cycles
  = 2\times1 + 1\times2 + 2\times3
  = 10
- Avg. CPI = 10/5 = 2.0

Sequence 2: IC = 6
- Clock Cycles
  = 4\times1 + 1\times2 + 1\times3
  = 9
- Avg. CPI = 9/6 = 1.5
A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C, and they require one, two, and three cycles (respectively).

The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C
The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C.

Which sequence will be faster? How much?

What is the CPI for each sequence?
Two different compilers are being tested for a 4 GHz. machine with three different classes of instructions: Class A, Class B, and Class C, which require one, two, and three cycles (respectively). Both compilers are used to produce code for a large piece of software.

The first compiler's code uses 5 million Class A instructions, 1 million Class B instructions, and 2 million Class C instructions.

The second compiler's code uses 10 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.

Which sequence will be faster according to MIPS?

Which sequence will be faster according to execution time?
Amdahl's Law

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
Suppose we enhance a machine making all floating-point instructions run five times faster.

If the execution time of some benchmark before the floating-point enhancement is 10 seconds, what will the speedup be if half of the 10 seconds is spent executing floating-point instructions?
Example 2

We are looking for a benchmark to show off the new floating-point unit described above, and want the overall benchmark to show a speedup of 3 (i.e., take 1/3 as long to run).

One benchmark we are considering runs for 100 seconds with the old floating-point hardware.

How much of the execution time would floating-point instructions have to account for in this program in order to yield our desired speedup on this benchmark?
Performance is specific to a particular program/s
- Total execution time is a consistent summary of performance

For a given architecture performance increases come from:
- increases in clock rate (if that yields no adverse CPI affects)
- improvements in processor organization that lower CPI
- compiler enhancements that lower CPI (different distribution of instructions) and/or instruction count
- algorithm/language choices that affect instruction count and/or instruction distribution

Pitfall: expecting improvement in one aspect of a machine’s performance to affect the total performance
Performance Summary

CPU Time = \( \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}} \)

Performance depends on
- Algorithm: affects IC, possibly CPI
- Programming language: affects IC, CPI
- Compiler: affects IC, CPI
- Instruction set architecture: affects IC, CPI, \( T_c \)