Goals for Performance Lecture

- Understand performance, speedup, throughput, latency
- Relationship between cycle time, cycles/instruction (CPI), number of instructions (the *performance equation*)
- Amdahl's Law
- Know how to do problems in the homework!

Definitions

Performance is in units of things-per-time

- Miles per hour, bits per second, widgets per day …
- Bigger is better

If we are primarily concerned with response time:

- Performance(x) = 1 / ExecutionTime(x)

“X is n times faster than Y” means

- n = Performance(X) / Performance(Y) = Speedup
- If X is 1.yz times faster than Y, we can informally say that X is yz% faster than Y. Speedup is better.

Latency vs. Throughput

Latency (Response Time)

- How long does it take for my job to run?
- How long does it take to execute a job?
- How long must I wait for the database query?

Throughput

- How many jobs can the machine run at once?
- What is the average execution rate?
- How much work is getting done?

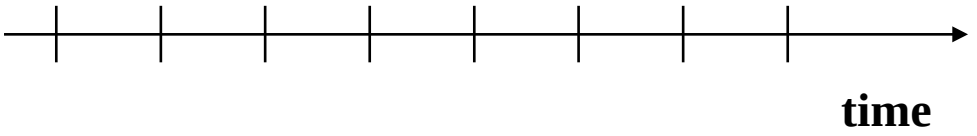
If we upgrade a machine with a new processor what do we increase?

If we add a new machine to the lab what do we increase?

Clock Cycles

Instead of reporting execution time in seconds, we often use cycles:

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$



Clock “ticks” indicate when to start activities

Cycle time = time between ticks = seconds per cycle

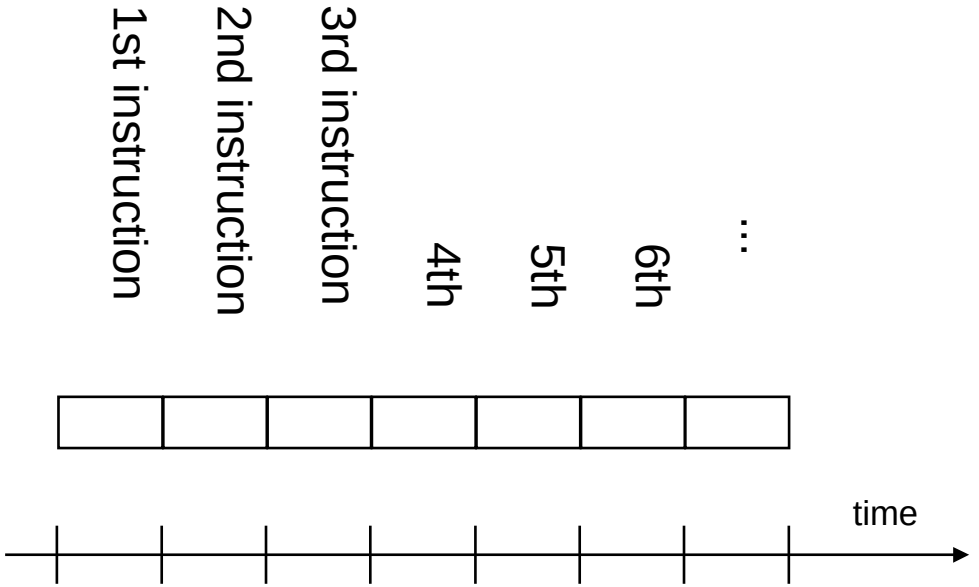
Clock rate (frequency) = cycles per second (1 Hz = 1 cycle/sec)

- A 200 MHz clock has a cycle time of …

$$\frac{1}{200 \times 10^6} \times 10^9 = 5 \text{ nanoseconds}$$

How many cycles in a program?

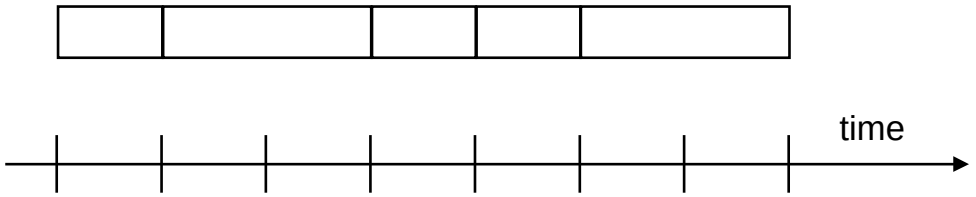
Could assume that # of cycles = # of instructions



This assumption is incorrect:

- different instructions take different amounts of time on different machines (even with the same instruction set).
- Why?

Different #s of cycles for diff’nt instrs



Multiplication takes more time than addition

Floating point operations take longer than integer ones

Accessing memory takes more time than accessing registers

Important point: changing the cycle time often changes the number of cycles required for various instructions (more later)

Example instruction latencies

Imagine Stream Processor:

On ALU:

- Integer adds: 2 cycles
- FP adds: 4 cycles
- Logic ops (and, or, xor): 1
- Equality: 1
- < or >: 2
- Shifts: 1
- Float->int: 3
- Int->float: 4
- Select (a?b:c): 1

Other functional units:

- Integer multiply: 4
- Integer divide: 22
- Integer remainder: 23
- FP multiply: 4
- FP divide: 17
- FP sqrt: 16

CPI

How many clock cycles, *on average*, does it take for every instruction executed?

We call this CPI (“Cycles Per Instruction”).

Its inverse (1/CPI) is IPC (“Instructions Per Cycle”).

CISC machines: this number is

- high(er)

RISC machines: this number is

- low(er)

CPI: Average Cycles per Instruction

CPI = (CPU Time * Clock Rate) / Instruction Count
= Clock Cycles / Instruction Count

$$\text{CPI} = \sum_{i=1}^{n} \text{CPI}_i \times \text{F}_i \qquad \text{where} \quad \text{F}_i = \frac{\text{I}_i}{\text{Instruction Count}}$$

On Imagine, integer adds are 2 cycles, FP adds are 4.
Consider an application that has 1/3 integer adds and 2/3 FP adds.
What is its CPI?

Given a 3 GHz machine, how many instrs/sec?

The Performance Equation

Time = Cycle Time * CPI * Instruction Count

- = seconds/cycle * cycles/instr * instrs/program
- => seconds/program

Performance = Clock Rate * IPC * 1/I

- = cycles/second * instr/cycle * program/instr
- => programs/second
  - Clock rate * IPC = instr/second = IPS (MIPS)

“The only reliable measure of computer performance is time.”

Remember

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 (without adverse CPI affects)
- improvements in processor organization that lower CPI
- compiler enhancements that lower CPI and/or instruction count

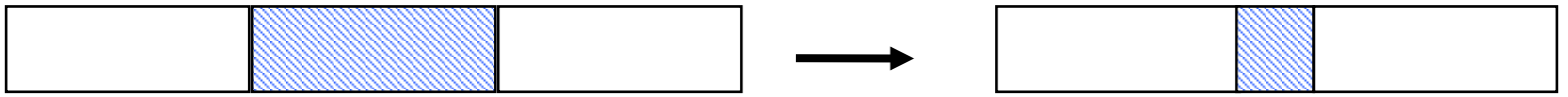
Pitfall: expecting improvement in one aspect of a machine's performance to affect the total performance

You should not always believe everything you read! Read carefully! (see newspaper articles)

Amdahl's Law

Speedup due to enhancement E:

$$\text{Speedup(E)} = \frac{\text{ExTime w/o E}}{\text{ExTime w/ E}} = \frac{\text{Performance w/ E}}{\text{Performance w/o E}}$$



Suppose that enhancement E accelerates a fraction F of the task by a factor S and the remainder of the task is unaffected:

$$\text{ExTime (with E)} = ((1-F) + F/S) * \text{ExTime(without E)}$$

$$\text{Speedup (with E)} = \frac{1}{(1-F) + F/S}$$

Design Principle: Make the common case fast!

There are many ways to express Amdahl's Law!

Undergrad Productivity

Average CPE student spends:

- 4 hours sleeping
- 2 hours eating
- 18 hours studying

Magic pill (purchased from dude at the UU) gives you all sleeping, eating in 1 minute!

What's the speedup on sleeping/eating?

How much more productive can you get?

Undergrad Productivity

$$\text{Speedup (with E)} = \frac{1}{(1-F) + F/S}$$

F = accelerated fraction = 0.25 (6 hrs/24 hrs)

S = speedup = 6 hrs / 1 minute = 360

Overall speedup:

- 1 / [(1-0.25) + (0.25/360)]
- ~= 1 / (1-0.25)
- ~= 1.33
- 33% more productive!