

# CSCI 564 Advanced Computer Architecture

## Lecture 02: Performance Metrics

---

Dr. Bo Wu (with modifications by Sumner Evans)

March 3, 2021

Colorado School of Mines

# What features do you care about in a computer?

- Quietness (dB)
- Speed
- Perceived speed
- Responsiveness
- Battery life
- **Good lookin'**
- Volume
- Dimensions
- Portability
- **Weight**
- **Size**
- Flexibility
- Reliability
- Expandability
- Upgradability
- Workmanship
- Memory bandwidth
- **RGB**
- Power consumption
- Good support
- Popularity
- **MKBHD recommended**
- Thermal performance
- Display quality
- It's a Mac
- Ergonomics
- **FPS for Fortnite**
- Sound quality
- Network speed
- **Good webcam for Zoom**
- Connectivity / ports
  - **USB-C**
  - Thunderbolt
  - HDMI
  - Ethernet
  - Bluetooth
  - Floppy disk drive
- Warranty
- Storage capacity
- Storage speed
- Peripherals
- Quality
- Bells and whistles
- Price

We don't care about most of these for this class.

## Metrics

---

## Metric: Latency

Latency is the most common metric in architecture

- **Latency = runtime**
- Speed =  $\frac{1}{\text{Latency}}$
- “Performance” usually, but not always, means latency

A measured latency is for **some particular task**

- A CPU doesn't have a latency
- An *application* has a latency on a *particular CPU*.

# Metric: Latency: Why Does It Matter?

- Application responsiveness
  - Any time a person is waiting
  - GUIs
  - Games
  - Internet services (from the user's perspective)
- “Real-time” applications
  - Tight constraints enforced by the real-world
  - Anti-lock braking systems — “hard” real-time
  - Multi-media applications — “soft” real-time



## Metric: Speedup

Speedup is the **ratio of two latencies**

$$\text{Speedup} = \frac{\text{Latency}_{\text{old}}}{\text{Latency}_{\text{new}}}$$

$\text{Speedup} > 1 \Rightarrow$  performance increased

$\text{Speedup} < 1 \Rightarrow$  performance decreased

If machine  $A$  is two times faster than machine  $B$ , then

$\text{Latency}_A = \text{Latency}_B/2$ . Thus, the speedup of  $B$  relative to  $A$  is

$$\text{Speedup} = \frac{\text{Latency}_A}{\text{Latency}_B} = \frac{\text{Latency}_B/2}{\text{Latency}_B} = \frac{1}{2} = 0.5.$$

## Metric: Speedup: Why is it Useful?

Speedup (and other ratio metrics) allow for the comparison of two systems without reference to any absolute units.

- We can say “doubling the clock speed will give a 2x speedup” without knowing anything about a concrete latency.
- It’s much easier than saying “If the program’s latency was 1,254 seconds, doubling the clock rate would reduce the latency to 627 seconds.”

## Metric: Throughput (Bandwidth)

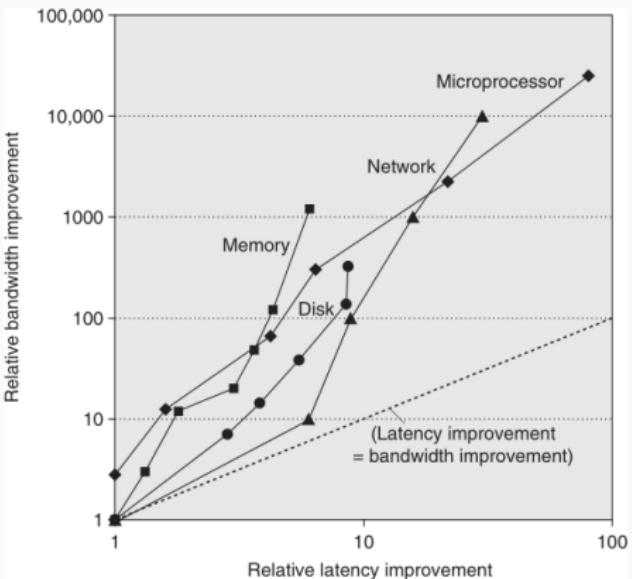
Throughput is the **number of tasks completed per unit of time**

- Throughput is independent from the exact total number of tasks.
- In what scenarios is throughput important?
  - Data center servers (YouTube or Netflix, for example)
  - High-performance computing

# Latency Lags Bandwidth

Bandwidth improvements have outpaced latency across the main computer technologies.

Page 19 of the book



## Latency Lags Bandwidth: Why?

*There is an old network saying: bandwidth problems can be cured with money. Latency problems are harder because the speed of light is fixed — you can't bribe God.*

*[Anonymous]*

*Some jobs are just hard to parallelize!*

*Dr. Bo Wu*

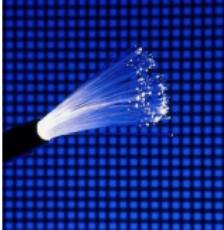
## Application Question

---

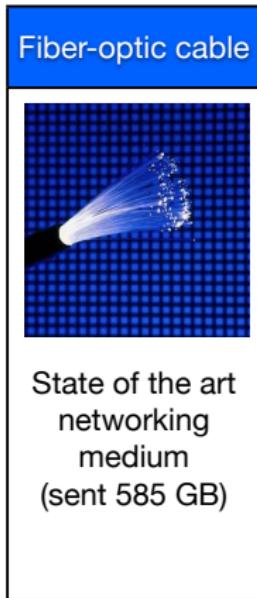
**Question:** I have two processors:  $A$  and  $B$ . I'm only interested in application  $C$ . The latency of finishing  $C$  on  $A$  is much smaller than that on  $B$ . What can I say about the *bandwidth difference* between  $A$  and  $B$ .

**Answer:** absolutely nothing!

# The Internet “Land”-Speed Record

Fiber-optic cable	
	
State of the art networking medium (sent 585 GB)	
Latency (s)	1800
BW (GB/s)	1.13

# The Internet “Land”-Speed Record

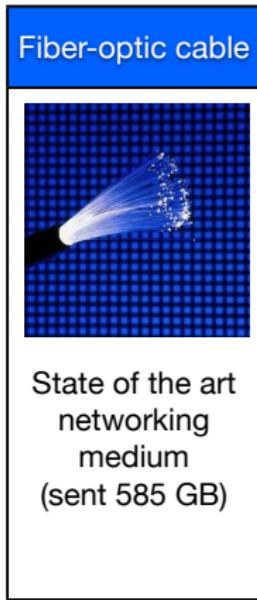


"Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway."

-- Andrew S. Tanenbaum

Latency (s)	1800
BW (GB/s)	1.13

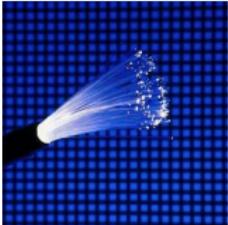
# The Internet “Land”-Speed Record



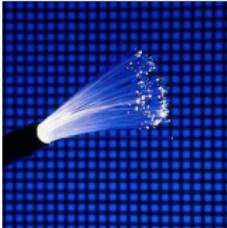
3.5 in Hard drive  
3 TB  
0.68 kg

Latency (s)	1800
BW (GB/s)	1.13

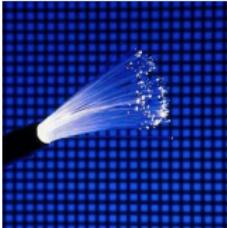
# The Internet “Land”-Speed Record

Fiber-optic cable	Subaru Outback
	
State of the art networking medium (sent 585 GB)	Sensible station wagon
Cargo	183 kg
Speed	119 MPH
Latency (s)	1800
BW (GB/s)	1.13

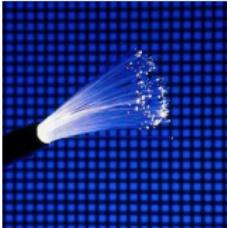
# The Internet “Land”-Speed Record

Fiber-optic cable	Subaru Outback
 A bundle of optical fibers against a dark blue background with a grid pattern.	 A silver Subaru Outback station wagon viewed from the front quarter angle.
State of the art networking medium (sent 585 GB)	Sensible station wagon
Cargo	183 kg
Speed	119 MPH
Latency (s)	1800
BW (GB/s)	563,984
	0.0014

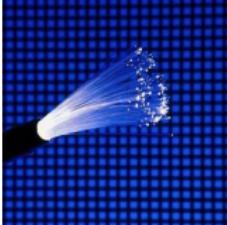
# The Internet “Land”-Speed Record

Fiber-optic cable	 A bundle of optical fibers against a blue grid background.	Subaru Outback	 A silver Subaru Outback station wagon.	B1-B	 A white B1-B supersonic bomber aircraft in flight.
Cargo	State of the art networking medium (sent 585 GB)	183 kg	25,515 kg	Supersonic bomber	
Speed		119 MPH	950 MPH		
Latency (s)	1800	563,984			
BW (GB/s)	1.13	0.0014			

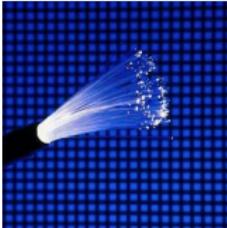
# The Internet “Land”-Speed Record

Fiber-optic cable	Subaru Outback	B1-B
		
State of the art networking medium (sent 585 GB)	Sensible station wagon	Supersonic bomber
Cargo	183 kg	25,515 kg
Speed	119 MPH	950 MPH
Latency (s)	1800	563,984
BW (GB/s)	1.13	0.0014
		70,646
		1.6

# The Internet “Land”-Speed Record

Fiber-optic cable	Subaru Outback	B1-B	Hellespont Alhambra
			
State of the art networking medium (sent 585 GB)	Sensible station wagon	Supersonic bomber	World's largest supertanker
Cargo	183 kg	25,515 kg	400,975,655 kg
Speed	119 MPH	950 MPH	18.9 MPH
Latency (s)	1800	563,984	70,646
BW (GB/s)	1.13	0.0014	1.6

# The Internet “Land”-Speed Record

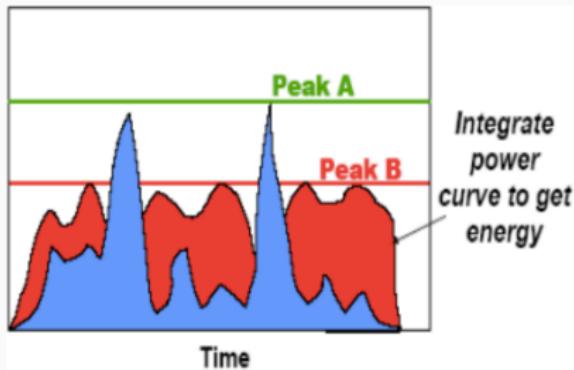
Fiber-optic cable	Subaru Outback	B1-B	Hellespont Alhambra
			
State of the art networking medium (sent 585 GB)	Sensible station wagon	Supersonic bomber	World's largest supertanker
Cargo	183 kg	25,515 kg	400,975,655 kg
Speed	119 MPH	950 MPH	18.9 MPH
Latency (s)	1800	563,984	70,646
BW (GB/s)	1.13	0.0014	1.6
			1,587,301
			1114.5

## Metric: Power

- Energy: measured in Joules
- Power: *rate* of energy consumption

### Example:

- System A: higher peak power
- System B: lower peak power



## Metric: Price

---

The importance is self-explanatory.

## Derived Metrics

---

# Derived Metrics

---

Often we care about multiple metrics at once.

- Examples (bigger is better):
  - Bandwidth per dollar (e.g., in networking  $(\text{GB/s})/\$$ )
  - Bandwidth/Watt (e.g., in memory systems  $(\text{GB/s})/\text{W}$ )
  - Work/Joule (i.e., instructions/Joule)
  - In general: multiply by bigger-is-better metrics, divide by smaller-is-better.
- Examples (smaller is better):
  - Cycles/instruction (i.e., time per work)
  - Latency  $\times$  Energy — “Energy Delay Product”
  - In general: multiply by smaller-is-better metrics, divide by bigger-is-better.

## Derived Metric: Energy-Delay

---

- Mobile systems must balance latency (delay) and battery (energy) usage for computation.
- The energy-delay product (EDP) is a smaller-is-better metric.
  - Base units: delay in seconds; energy in Joules
  - EDP units: (Joules)(s)
- Some use  $\text{Energy} \times \text{Delay}^2$  instead.

## What's the Right Metric?

There is no universally correct metric! In fact, you can make up any metric you want to evaluate computer systems.

- Latency for compiling the Linux kernel using GCC.
- Latency for compiling “hello world” in Rust.
- Frames per second at max settings in Cyberpunk 2077.
- $(\text{DB transactions}/\text{second})/\$/$ .
- The right metric depends on the situation.
  - What does the computer need to accomplish?
  - What constraints is it under?
- Usually some (relatively simple) combination of the metrics we've already discussed.
- **We will mostly focus on performance (latency and/or bandwidth)**

**Now we have metrics, how do we compare different architectures?**

## Benchmarks

---

## Benchmarks: Make Comparable Measurements

---

- A **benchmark suite** is a set of programs that are representative of a *set of problems*.
  - Server computing (SPECINT)
  - Scientific computing (SPECFP)
  - Geekbench (Mobile phones)
- There is no “best” benchmark suite because *there are so many different problems!*
- Real software performance may not match the benchmark, because the benchmark is only a subset of problems!

# Classes of Benchmarks

- **Microbenchmarks** measure one feature of a system.
  - For example, memory accesses or communication speed
- **Kernels** measure the most compute-intensive part of applications
- **Full application benchmarks** measure the performance of a full application
  - SpecINT and SpecFP (for servers)
  - Other suites for databases, web servers, graphics, ...

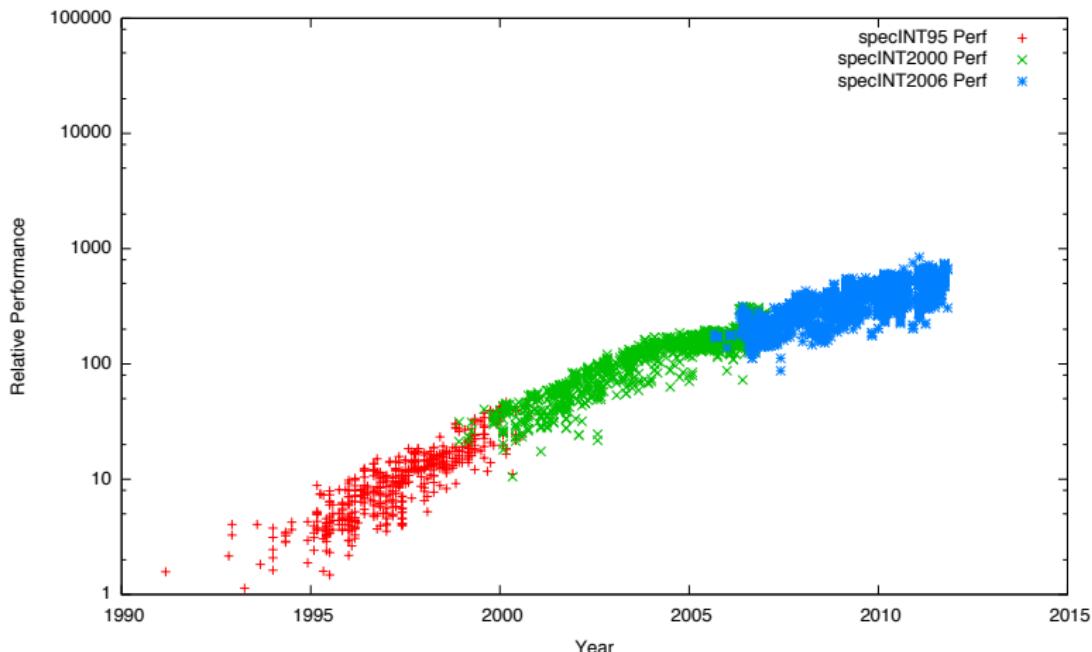
# SPECINT 2006

- In what ways are these not representative?

Application	Language	Description
400.perlbench	C	PERL Programming Language
401.bzip2	C	Compression
403.gcc	C	C Compiler
429.mcf	C	Combinatorial Optimization
445.gobmk	C	AI: go
456.hmmmer	C	Search Gene Sequence
458.sjeng	C	AI: chess
462.libquantum	C	Quantum Computing
464.h264ref	C	Video Compression
471.omnetpp	C++	Discrete Event Simulation
473.astar	C++	Path-finding Algorithms
483.xalancbmk	C++	XML Processing

# SPECINT 2006

- Despite all that, benchmarks are quite useful.
  - e.g., they allow long-term performance comparisons



## The CPU Performance Equation

---

# The Performance Equation (PE)

**Problem:** We would like to model how architecture impacts performance (latency).

**Solution:** We need a way to quantify performance in terms of architectural parameters.

- **Instruction Count** — The number of instructions the CPU executes.
- **Cycles per instruction** — The ratio of cycles required to execute the program to the number of instructions executed.
- **Cycle time** — The length of a clock cycle in seconds

**The first fundamental theorem of computer architecture:**

$$\text{Latency} = \text{Instruction Count} \times \text{Cycles/Instruction} \times \text{Seconds/Cycle}$$

$$L = IC \times CPI \times CT$$

# The PE as a Mathematical Model

$$\text{Latency} = \text{Instruction Count} \times \text{Cycles/Instruction} \times \text{Seconds/Cycle}$$

- The PE models the real computer system!
  - Latency changes linearly with IC, CPI, and CT
- We can make some inferences from this fact. Namely, there must be several ways to improve performance
  - Reduce IC, reduce CPI, or reduce CT
- We can also use this to evaluate potential trade-offs
  - Reducing CT by 50% and increasing CPI by 2 gives us a net win (assuming that CPI was > 2 in the original system).

# Reducing Cycle Time

---

- Cycle time is a function of *the processor's design*.
  - If the design does less work during a clock cycle, it's cycle time will be shorter.
  - *We will revisit this when we discuss pipelining.*
- Cycle time is a function of *process technology*.
  - If we scale a fixed design to a more advanced process technology, it's clock speed will go up.
  - *However, clock rates aren't increasing much, due to power problems.*
- Cycle time is a function of *manufacturing variation*.
  - Manufacturers "bin" individual CPUs by how fast they can run.
  - The more you pay, the faster your chip will run.

## The Clock Speed Corollary

$$\text{Latency} = \text{Instruction Count} \times \text{Cycles/Instruction} \times \text{Seconds/Cycle}$$

- We use *clock speed* more than second/cycle.
- Why? **Consumers like bigger is better metrics!**
- Clock speed is measured in Hz (cycles, just like physics). Normally, there's a metric prefix (MHz, GHz, etc.).
  - $x \text{ Hz} \Rightarrow \frac{1}{x} \text{ seconds per cycle}$
  - $2.5 \text{ GHz} \Rightarrow \frac{1}{2.5 \times 10^9 \text{ s}} \text{ seconds (0.4ns) per cycle}$

$$\text{Latency} = (\text{Instr. Count} \times \text{Cycles/Instr.}) / (\text{Clock speed in Hz})$$

# A Note About Instruction Count

---

- The instruction count we care about is the **dynamic instruction count**.
- “Dynamic”
  - Measures the number of instructions that the program executes at *run-time*.
  - Example: When I ran that program, it executed 1 million *dynamic* instructions.
- “Static”
  - Measures the number of instructions in the compiled code (proportional to binary size).
  - Example: When I compiled that function, it produced 10 *static* instructions.

## Reducing Instruction Count (IC)

---

- There are many ways to implement a particular computation
  - Algorithmic improvements (use quicksort instead of bubble sort)
  - Compiler optimizations (use -O4 with GCC)
- If one version requires executing fewer dynamic instructions, the PE predicts it will be faster.
  - Assuming that the CPI and clock speed remain the same...
  - A  $x\%$  reduction in IC should give a speedup of  $\frac{1}{1-0.01x}$  times.
  - For example, a 20% reduction in IC  $\Rightarrow \frac{1}{1-0.20} = 1.25x$  speedup.

## Factors which Impact Instruction Count

- *Different programs* do different amounts of work
  - Playing a DVD vs writing a word document
- The same program may do different amounts of work depending on the *input*.
  - Compiling a 1000-line program vs. a 100-line program
- The same program may require different numbers of instructions on *different ISAs*.
  - x86 has FSQRT, but in MIPS you would have to implement that as a set of instructions
- To make a meaningful comparison between two computer systems, they must be doing the *same work*.
  - They may execute a different number of instructions (different ISA or compiler), but the *task* they accomplish must be exactly the same.

## Cycles Per Instruction

CPI is the most complex term of the PE because many factors impact it

- The compiler
- The program's inputs
- The processor's design (we will discuss this extensively)
- The memory system (we will discuss this extensively)

CPI **is not** the number of cycles required to execute one instruction!

CPI **is** the *ratio* of the number of cycles required to execute a program and that program's IC. **It is an average.**

You may find  $1/\text{CPI}$  (Instructions Per Cycle; IPC) to be more intuitive because it emphasizes that it's an average.

## Practice: Calculating CPI: Worksheet Problem 1

Compute the CPI for the following program and machine:

- Program: 10% floating-point operations, 20% memory access instructions, 40% branch/jump instructions, and the rest are ALU operations.
- Machine: cycles required for each of the following operations:
  - floating-point operations: 11 cycles
  - memory access instructions: 50 cycles\*
  - branch/jump instructions: 5 cycles\*
  - ALU operations: 2 cycles

\* We are making a few simplifying assumptions for this problem, both of these will become more complicated later.

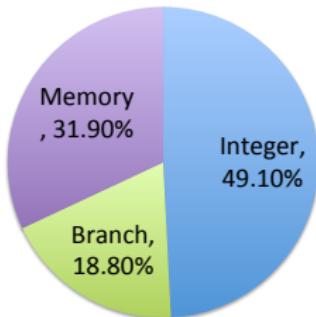
# Instruction Mix and CPI

- Different programs need different kinds of instructions
  - e.g., “Integer apps” don’t do much floating point math.
- The compiler also has some flexibility in which instructions it uses.
- As a result the combination and ratio of instruction types that programs execute (their *instruction mix*) varies.

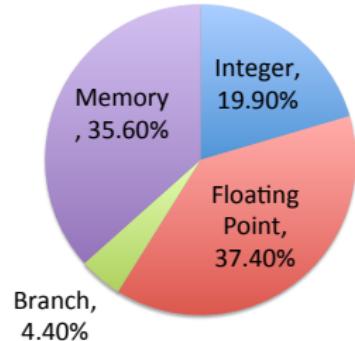
# Instruction Mix and CPI

- Different programs need different kinds of instructions
  - e.g., “Integer apps” don’t do much floating point math.
- The compiler also has some flexibility in which instructions it uses.
- As a result the combination and ratio of instruction types that programs execute (their *instruction mix*) varies.

Spec INT 2006



Spec FP 2006



Spec INT and Spec FP are popular benchmark suites

# Instruction Mix and CPI

Instruction Type	Cycles
Integer +, -,  , &, branches	1
Integer multiply	3-5
integer divide	11-100
Floating point +, -, *, etc.	3-5
Floating point /, sqrt	7-27
Loads and Stores	1-100s

These values are for Intel's Nehalem processor

# Instruction Mix and CPI

- Instruction selections (and, therefore, instruction selection) impacts CPI because some instructions require extra cycles to execute
- All these values depend on the particular implementation, not the ISA.

Instruction Type	Cycles
Integer +, -,  , &, branches	1
Integer multiply	3-5
integer divide	11-100
Floating point +, -, *, etc.	3-5
Floating point /, sqrt	7-27
Loads and Stores	1-100s

These values are for Intel's Nehalem processor

## Program Inputs and CPI

---

- Different inputs make programs behave differently
  - They execute different functions.
  - The branches will go in different directions.
  - These all affect the instruction mix (and instruction count) of the program.

## Practice: Comparing Similar Systems: Worksheet Problem 2

$$\text{Latency} = \text{Instruction Count} \times \text{Cycles/Instruction} \times \text{Seconds/Cycle}$$

Assume that we are running the *same program* on two different systems. **What is the speedup of System B relative to System A?**

- System A:
  - Instruction count: 3 million
  - Cycles per instruction: 4.7
  - Seconds per cycle: 1ns
- System B:
  - Instruction count: 4 million
  - Cycles per instruction: 2.3
  - Seconds per cycle: 1ns

## Practice: Comparing Similar Systems: Worksheet Problem 3

$$\text{Latency} = \text{Instruction Count} \times \text{Cycles/Instruction} \times \text{Seconds/Cycle}$$

Assume that we are running the *same program* on two different systems. **What is the speedup of System B relative to System A?**

- System A:
  - Instructions per cycle: 0.4
  - Clock speed: 3.7 GHz
- System B:
  - Instructions per cycle: 0.3
  - Clock speed: 4.0 GHz

## Comparing Similar Systems

- Often, we will compare systems that are partially the same.
  - Example: different CPUs, but the same program.
  - Example: same CPU, different programs.
- In these cases, many terms of the PE are not relevant.
  - If the CPU doesn't change, neither does CT, so latency can be measured in cycles:

$$\frac{\text{Instructions}}{\text{Instruction}} \times \frac{\text{Cycles}}{\text{Instruction}} = \text{Cycles}$$

- If the workload is fixed, IC doesn't change, so performance can be measured in Instructions/Second:

$$1 / \left( \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}} \right)$$

- If the workload *and* clock rate are fixed, the latency is equivalent to CPI (smaller-is-better).

## Comparing Similar Systems

Another way of thinking about this is that you are using the *speedup equation*, but not comparing anything yet.

$$\text{Speedup} = \frac{\text{Latency}_{\text{old}}}{\text{Latency}_{\text{new}}}$$

Example: if we have the same CPU, the CT is constant ( $\text{CT}_{\text{old}} = \text{CT}_{\text{new}}$ ).

$$\text{Speedup} = \frac{\text{Latency}_{\text{old}}}{\text{Latency}_{\text{new}}} = \frac{\text{IC}_{\text{old}} \times \text{CPI}_{\text{old}} \times \cancel{\text{CT}_{\text{old}}}}{\text{IC}_{\text{new}} \times \text{CPI}_{\text{new}} \times \cancel{\text{CT}_{\text{new}}}}$$

### Warning

You can only ignore terms in the PE if they are *identical* across two systems.

## Practice: Comparing Similar Systems: Worksheet Problem 4

You have a processor that runs at 4.9 GHz with a CPI of 1.4.

You can either spend \$10,000 to hire a CS@Mines graduate for two weeks to optimize your algorithm so that it requires 37% less instructions to execute as before (assume same CPI).

Or, you can spend \$1,500 on a new CPU that runs at 5.3 GHz (with the same CPI).

**Which option gives you the biggest performance gain per dollar spent?**

## Dropping Terms from the PE

- The PE is built to make it easy to focus on aspects of latency by dropping terms.
- Example:  $\text{CPI} \times \text{CT}$ 
  - $\text{Seconds}/\text{Instruction} = \text{IS}$  (instruction latency)
  - $1/\text{IS} = \text{Inst}/\text{Sec}$  or M(ega)IPS, FLOPS
  - Could also be called “raw speed”
  - CPI is still in terms of some particular application or instruction mix.
- Example:  $\text{IC} \times \text{CPI}$ 
  - Clock-speed independent latency (cycle count)

## Treating PE Terms Differently

The PE also allows us to apply “rules of thumb” and/or make projections.

**Example:** “CPI in modern processors is between 1 and 2”

- $L = IC \times CPI_{guess} \times CT$
- In this case, IC corresponds to a particular application, but  $CPI_{guess}$  is an estimate.

**Example:** The new processor will reduce CPI by 50% and reduce CT by 50%.

- $L = IC \times 0.5CPI \times 0.5CT$
- Now CPI and CT are both estimates, and the resulting L is also an estimate. IC may not be an estimate.

## Abusing the PE

- Beware of Guaranteed Not To Exceed (GTNE) metrics
- Example: “Processor X has a speed of 10 GOPS (giga insts/sec)”
  - This is equivalent to saying that the average instruction latency is 0.1ns.
  - **But no workload is given!**
  - Does this means that  $L = IC * 0.1\text{ns}$ ? **Probably not!**
- The above claim (probably) means that the processor is capable of 10 GOPS under perfect conditions
  - The vendor promises it will never go faster.
  - That's very different than saying how fast it will go in practice.
- It may also mean they get 10 GOPS on an industry standard benchmark

## The top500 List

---

You may have wondered, “what is the fastest computer in the world?”

[top500.org](http://top500.org) publishes a list of the fastest 500 computers in the world.

They report *floating point operations per second* (FLOPS). They use the linpack benchmark suite (dense matrix computation)

The top machine now is Supercomputer Fugaku in Japan.

Is it meaningful or fair? There is a new list [graph500.org](http://graph500.org) which “complement[s] the Top 500 with data intensive applications”.