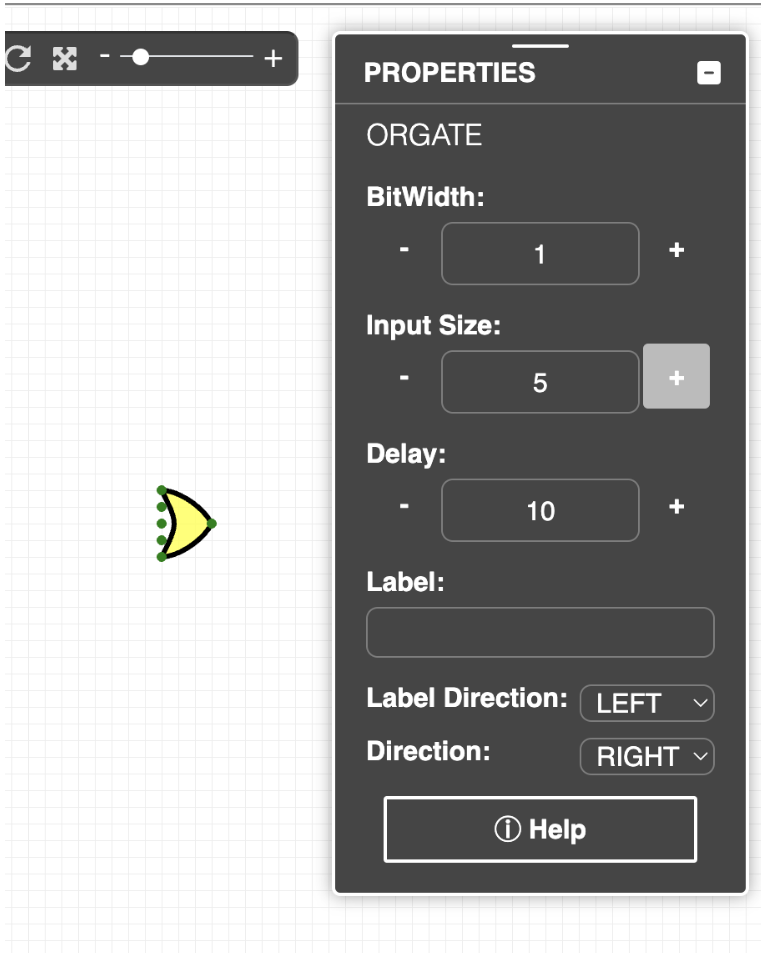
CSCI 210: Computer Architecture
Lecture 20: Performance

Stephen Checkoway
Slides from Cynthia Taylor

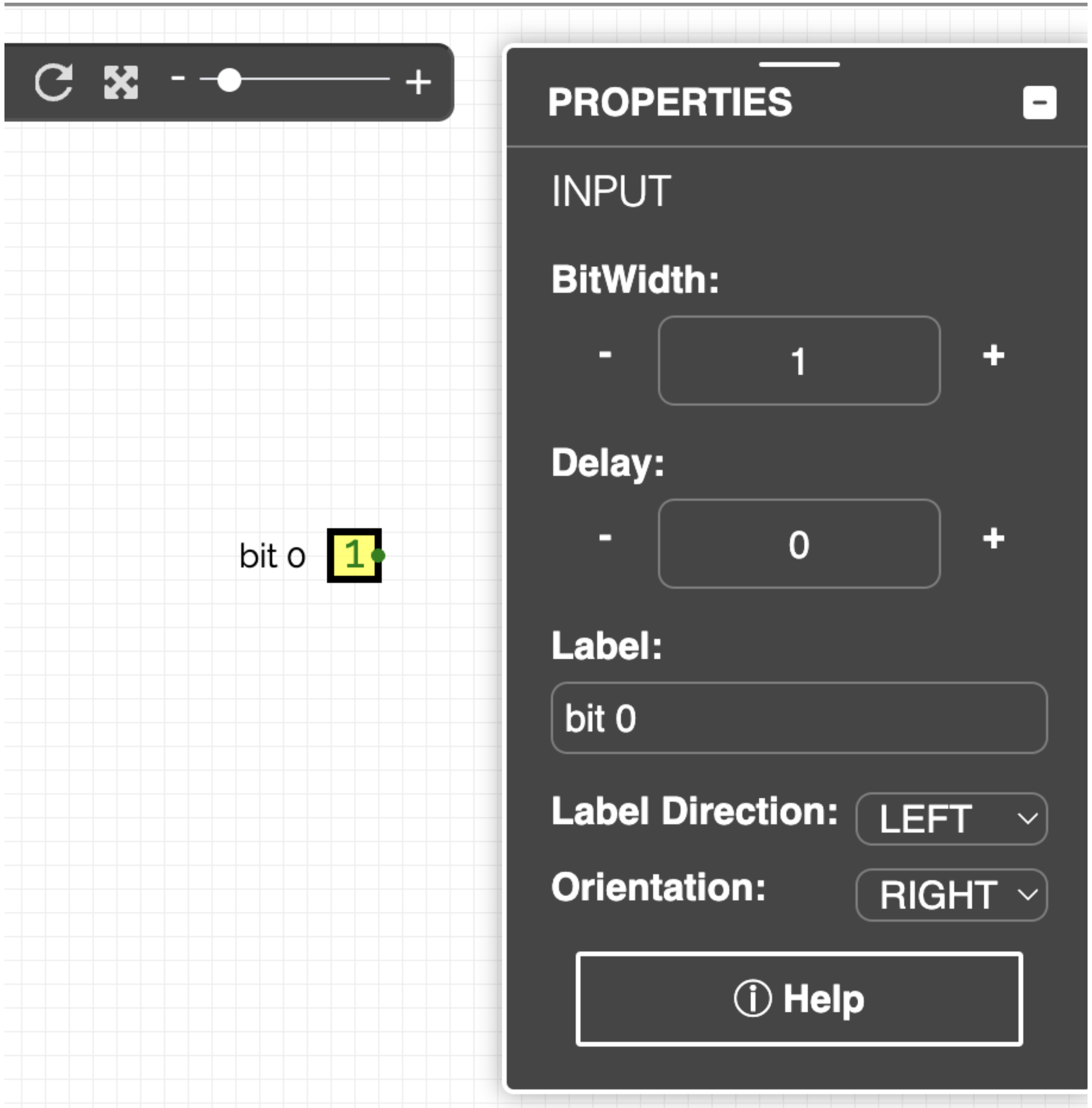
Circuitverse Tips

- Add more inputs to the gates if you’re combining lots of inputs



Circuitverse Tips

- Circuitverse will reorder inputs and outputs for subcircuits
- Label your inputs and outputs so you can tell what they are



Circuitverse/Lab Questions?

CS History: Performance

	Millions of Instructions per Second	Instructions per cycle	Instructions per cycle per core	Year
UNIVAC I	0.002	0.0008	0.0008	1951
IBM 7030 ("Stretch")	1.200	0.364	0.364	1961
VAX-11/780	1.000 MIPS	0.2	0.2	1977
Intel i860	25 MIPS	1	1	1989
Intel Core i7 3770K (4-core)	106,924 MIPS	27.4	6.9	2012
Raspberry Pi 2 (quad-core ARM Cortex A7)	4,744 MIPS	4.744	1.186	2014
Intel Core i5-11600K (6-core)	346,350 MIPS	57.72	11.73	2021

Note: Millions of instructions per second is a misleading metric as it does not tell us what the instructions do

Measures of “Performance”

- Execution Time
- Frame Rate
- Throughput (operations/time)
- Responsiveness
- Performance / Cost
- Performance / Power

Match (**Best**) Performance Metric to Domain

<u>Performance Metrics</u>

1. Network Bandwidth (data/sec)
2. Network Latency (ms per roundtrip)
3. Frame Rate (frames/sec)
4. Throughput (operations/sec)

	Online Games	High-def video	Torrent Download	Server Cluster
A	4	3	1	2
B	4	1	3	2
C	2	1	3	4
D	2	3	1	4
E	None of the above			

Metrics for running a program

- Execution Time – how long does it take to run?
- CPI – (clock) cycles per instruction
- Instruction Count – how many instructions does it have?
- Clock cycle time

A note on cycles per instruction

- Different instructions can take different lengths of time.
  – Multiplication and division take longer than addition and logical operations
  – Floating point takes longer than integer operations
  – Memory instructions take longer than everything else

All Together Now

$$\text{CPU Execution Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

All Together Now

seconds

CPU Execution Time = Instruction Count X CPI X Clock Cycle Time

instructions

cycles/instruction

seconds/cycle

- You have a 1 billion (10^9) instruction program, a 500 MHz processor, and an execution time of 3 seconds. What is the CPI for this program?

- Note that 1 MHz = 1 million (10^6) cycles per second

Selection	CPI
A	3
B	15
C	1.5
D	15*10^9
E	None of the above

seconds

CPU Execution Time = Instruction Count X CPI X Clock Cycle Time

instructions

cycles/instruction

seconds/cycle

Who Affects Performance?

$$\underset{\text{Time}}{\text{CPU Execution}} = \overset{IC}{\underset{\text{Count}}{\text{Instruction}}} \times \text{CPI} \times \overset{CT}{\underset{\text{Time}}{\text{Clock Cycle}}}$$

- There are a number of people involved in processor / programming design
- Each of these elements of the performance equation can be impacted by different designer(s)
- Next slides will be about who can impact what

Who Affects Performance?

$$\text{CPU Execution Time} = \overset{\color{blue}{IC}}{\text{Instruction Count}} \times \text{CPI} \times \overset{\color{blue}{CT}}{\text{Clock Cycle Time}}$$

- What can a programmer influence?

Selection	Impacts
A	IC
B	IC, CPI
C	IC, CPI, and CT
D	IC and CT
E	None of the above

Who Affects Performance?

		IC				CT
CPU Execution Time	=	Instruction Count	X	CPI	X	Clock Cycle Time

- What can a compiler influence?

Selection	Impacts
A	IC
B	IC, CPI
C	IC, CPI, and CT
D	CPI and CT
E	None of the above

Who Affects Performance?

$$\text{CPU Execution Time} = \overset{\text{IC}}{\text{Instruction Count}} \times \text{CPI} \times \overset{\text{CT}}{\text{Clock Cycle Time}}$$

- What can an instruction set architect influence?

Selection	Impacts
A	IC
B	IC, CPI
C	IC, CPI, and CT
D	CPI and CT
E	None of the above

Who Affects Performance?

		IC				CT
CPU Execution Time	=	Instruction Count	X	CPI	X	Clock Cycle Time

- What can a hardware designer influence? Assume they are designing a chip for a fixed ISA.

Selection	Impacts
A	IC
B	IC, CPI
C	IC, CPI, and CT
D	CPI and CT
E	None of the above

If we run two different programs on the same machine, how do the number of instructions, CPI, and clock cycle time compare?

	Number of instructions	CPI	Clock cycle time
A	Same	Same	Same
B	Different	Same	Same
C	Different	Different	Same
D	Different	Different	Different
E	Different	Same	Different

If we run the same program on two different machines with different ISAs, how do the number of instructions, CPI, and clock cycle time compare?

	Number of instructions	CPI	Clock cycle time
A	Same	Same	Same
B	Same	Same	Different
C	Same	Different	Different
D	Different	Different	Different
E	Different	Same	Same

If we run the same program on two different machines with the same ISA, how do the number of instructions, CPI, and clock cycle time compare?

	Number of instructions	CPI	Clock cycle time
A	Same	Same	Same
B	Same	Same	Different
C	Same	Different	Different
D	Different	Different	Different
E	Different	Same	Same

How we can measure CPU performance

- Millions of instructions per second

- Performance on benchmarks—programs designed to measure performance

- Performance on real programs

MIPS (not the name of the architecture)

$$\text{MIPS} = \text{Millions of Instructions Per Second}$$

$$= \frac{\text{Instruction Count}}{\text{Execution Time} * 10^6}$$

$$= \frac{\text{Clock rate}}{\text{CPI} * 10^6}$$

- program-dependent
- deceptive

Speedup

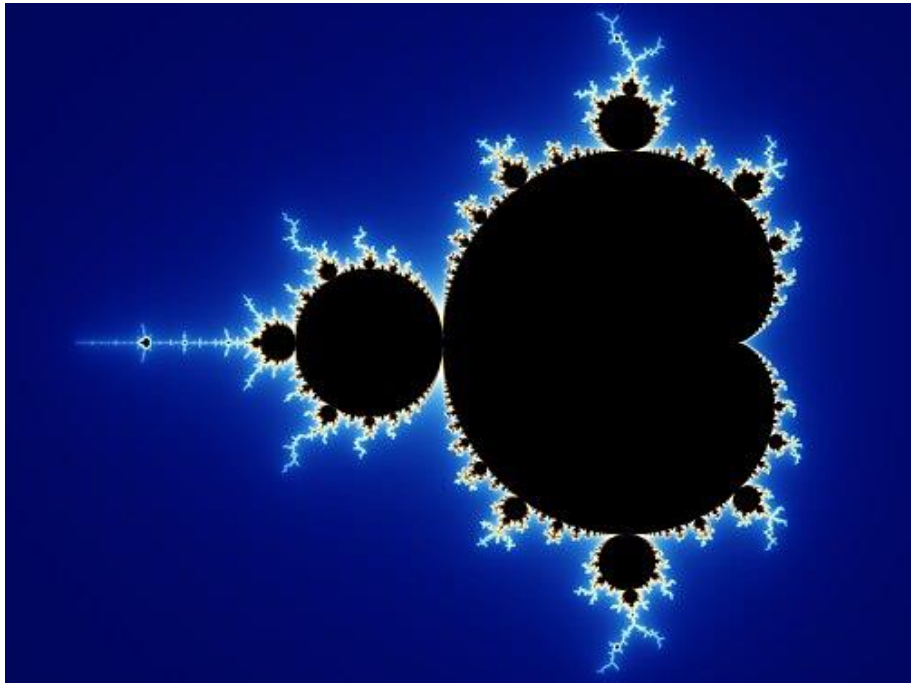
- Often want to compare performance of one machine/program/system against another

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

$$\text{Speedup (A over B)} = \frac{\text{Performance}_A}{\text{Performance}_B}$$

$$\text{Speedup (A over B)} = \frac{ET_B}{ET_A}$$

Two programs are written to compute the Mandelbrot set fractal, one in Rust and one in Java. The Rust version runs in 246 ms. The Java version runs in 1153 ms. What is the speedup of the Rust version over the Java version?



A. (246 ms) / (1153 ms) = 0.21
B. (1153 ms) / (246 ms) = 4.69
C. (1153 ms) – (246 ms) = 907 ms
D. (0.246 s) * (1.153 s) = 0.29 s^2

Data from https://programming-language-benchmarks.vercel.app/problem/mandelbrot on 2025-10-15

Jane has written a program P in Rust that solves mazes

When P is compiled for debug (cargo build) and run on a particular input maze, it takes 2.3×10^9 instructions with an average CPI of 2.0 cycles/instruction

When P is compiled with optimizations (cargo build --release) and run on the same input, it takes 2.0×10^9 instructions with an average CPI of 1.9 cycles/instruction

In your groups, compute (1) the execution time for the debug build; (2) the execution time for the optimized build; and (3) the speedup of the optimized build over the debug build. [Hint, 1 and 2 should be expressed as multiples of the cycle time and 3 should be a number.]

Select A on your clicker when you've finished

Amdahl’s Law

$$\text{Execution time after improvement} = \frac{\text{Execution Time Affected}}{\text{Amount of Improvement}} + \text{Execution Time Unaffected}$$

Amdahl’s Law and Parallelism

- Our program is 90% parallelizable (segment of code executable in parallel on multiple cores) and runs in 100 seconds with a single core. What is the execution time if you use 4 cores (assume no overhead for parallelization)?

$$\text{Execution time after improvement} = \frac{\text{Execution Time Affected}}{\text{Amount of Improvement}} + \text{Execution Time Unaffected}$$

Selection	Execution Time
A	25 seconds
B	32.5 seconds
C	50 seconds
D	92.5 seconds
E	None of the above

Amdahl's Law

- So what does Amdalh's Law ***mean*** at a high level?

Selection	"BEST" message from Amdahl's Law
A	Parallel programming is critical for improving performance
B	Improving serial code execution is ultimately the most important goal.
C	Performance is strictly tied to the ability to determine which percentage of code is parallelizable.
D	The impact of a performance improvement is limited by the percent of execution time affected by the improvement
E	None of the above

When an image classification program C is run on a corpus of images, C takes 80 seconds and spends 40% of its time computing matrix multiplications. After the linear algebra library is replaced with one that performs matrix multiplications 1.5 times faster, how long does C take to run on the corpus?

A. (80 s)*0.4/1.5 + (80 s)*0.6 = 69.3 s
B. (80 s)*0.6/1.5 + (80 s)*0.4 = 64 s
C. (80 s)*0.4*1.5 + (80 s)*0.6 = 96 s
D. (80 s)*0.6*1.5 + (80 s)*0.4 = 104 s
E. None of the above

C takes 80 seconds to run and spends 40% of its time computing matrix multiplications. What is the maximum possibly speedup for C that can be achieved by replacing the matrix multiplication functions with faster ones? [Hint: Apply Amdahl's law with an infinite improvement.]

A. (80 s)*0.4 / (80 s) = 0.4
B. (80 s)*0.6 / (80 s) = 0.6
C. (80 s) / [(80 s)*0.4] = 2.5
D. (80 s) / [(80 s)*0.6] = 1.7
E. Not enough information given

The table gives the percentage of instructions in a program P and the number of cycles per instruction for each instruction type. If P contains 4×10^9 instructions and the processor runs at 2 GHz (2×10^9 cycles per second), what is the execution time of P?

Instruction type	% of total instructions	CPI
Integer	55%	2
Load/Store	30%	3
Branch	15%	4

A. 4 s

B. 5.2 s

C. 6 s

D. 7.4 s

The next version of the processor executes integer instructions in 1 cycle instead of 2. What does Amdahl's law tell us about the execution time of P on this version of the processor (assuming the clock rate remains the same) and the original execution time is 5.2 s? (Be careful about what fraction of the old execution time is affected by the improvement!)

Instruction type	% of total instructions	CPI
Integer	55%	2→1
Load/Store	30%	3
Branch	15%	4

A. 8.06 s
B. 4.10 s
C. 3.77 s
D. 2.60 s
E. I know the answer but don't have a calculator to do the arithmetic

$$\text{Execution time after improvement} = \frac{\text{Execution Time Affected}}{\text{Amount of Improvement}} + \text{Execution Time Unaffected}$$

Things are not so simple! The version of the processor with the faster integer instructions cannot run as fast as before. The new version runs at 1.5 GHz instead of 2 GHz. What is the new execution time?

Instruction type	% of total instructions	CPI
Integer	55%	2→1
Load/Store	30%	3
Branch	15%	4

A. 8.20 s
B. 6.18 s
C. 5.47 s
D. I know the answer but don't have a calculator to do the arithmetic

The new version of the processor runs program P slower than the old version even though the majority of the instructions run faster than before! How is that possible?

A. It’s not possible!
B. Amdahl’s law tells us only about 40% of the execution time was affected by the improvement but 100% of the instructions were slowed down at the same time and the limited speedup wasn’t sufficient to overcome the universal slow down
C. Due to the reduction in clock speed, every program would run slower, not just P

What is the maximum possible speed up of the new processor over the old processor on some program? (Hint: Consider a program with a different mix of instruction types.) Recall that speedup of *new* over *old* is the execution time of *old* divided by the execution time of *new*.

Instruction type	% of total instructions	CPI
Integer	55%	$2 \rightarrow 1$
Load/Store	30%	3
Branch	15%	4

A. 2.0
B. 1.5
C. 0.8
D. I know the answer but don't have a calculator to do the arithmetic

Key Points

- Be careful how you specify “performance”
- Execution time = IC * CPI * CT
- Use real applications, if possible
- Make the common case fast