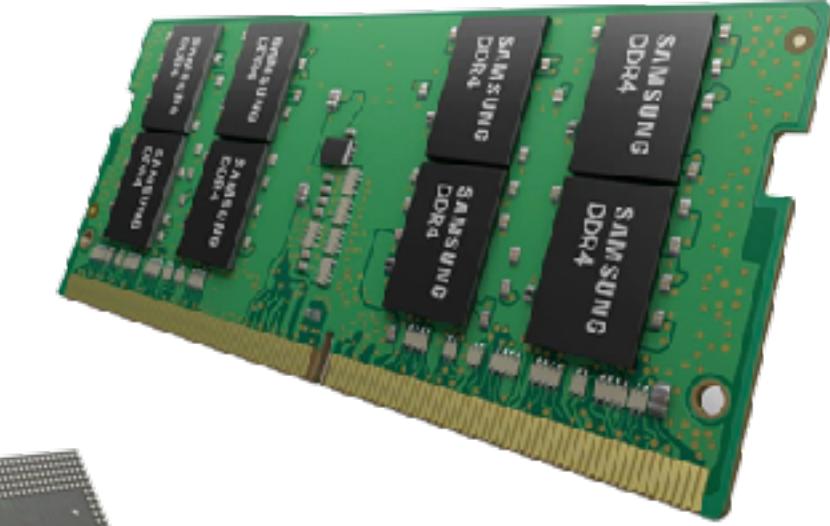


Performance (1):

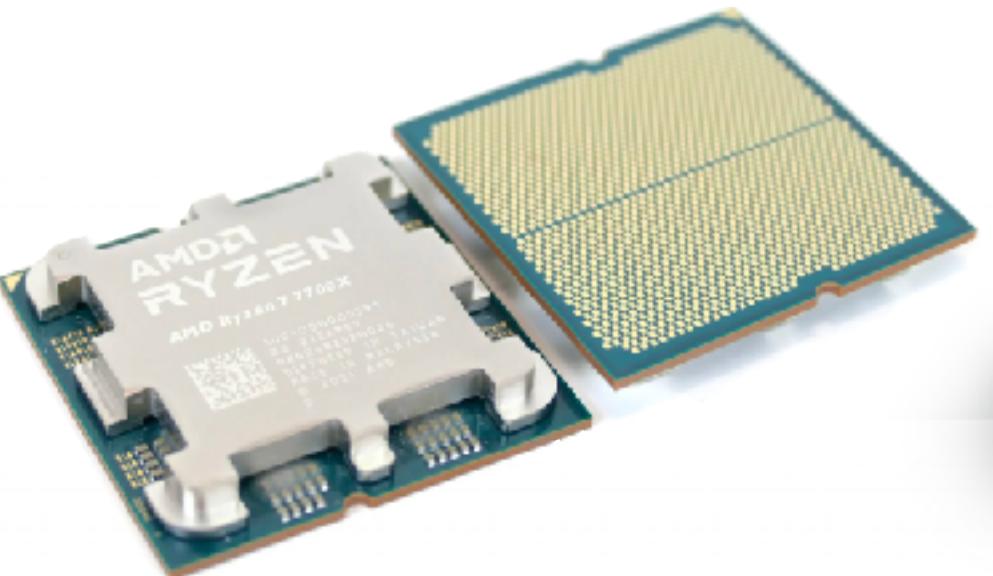
What does “perfect” mean?

Hung-Wei Tseng

Recap: Processors and memory modules are everywhere!

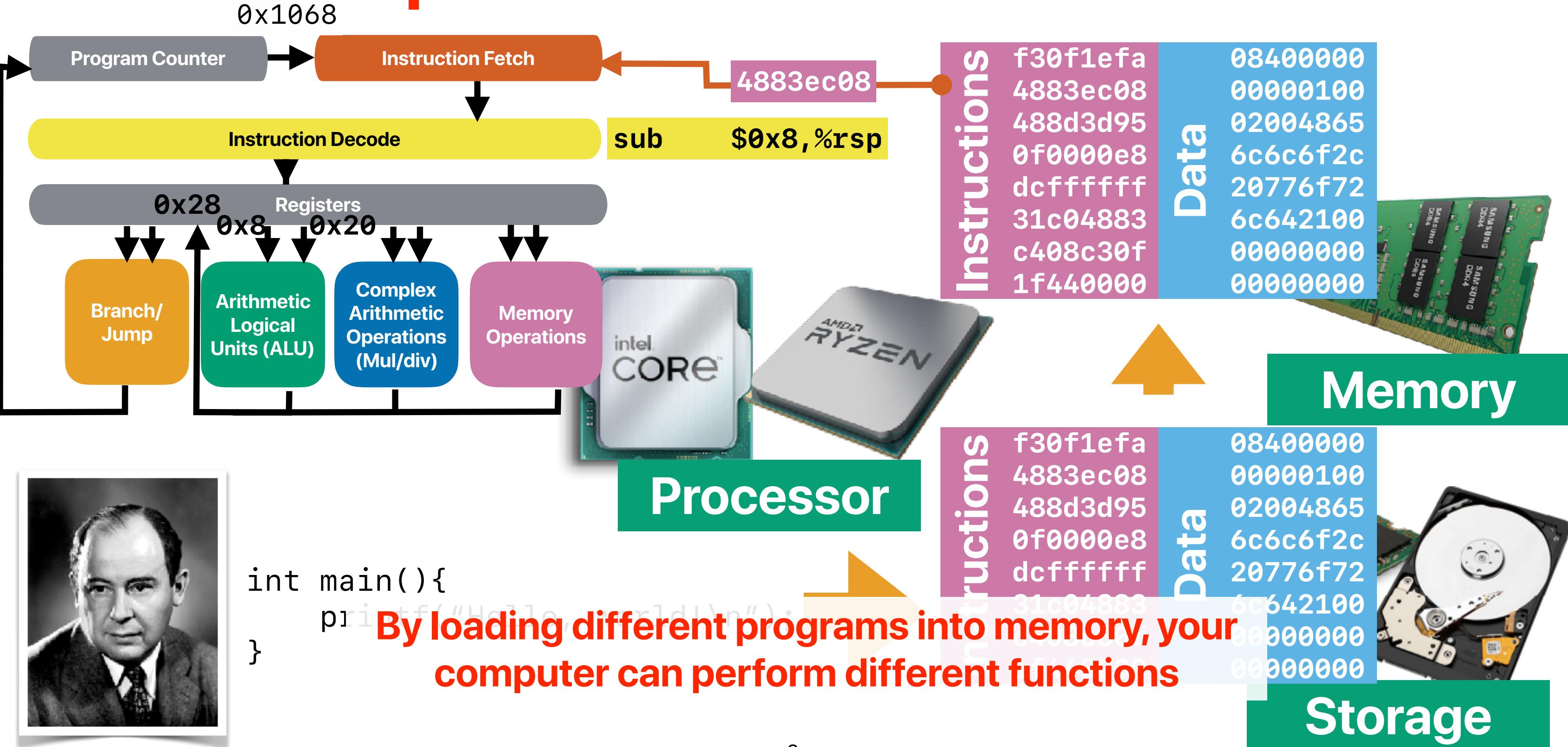


Processors



Memory

Recap: von Neumann architecture



Recap: Demo

```
if(option)
    std::sort(data, data + arraySize);      O(nlog2n)
for (unsigned c = 0; c < arraySize*1000; ++c) {
    int t = std::rand();
    if (data[c%arraySize] >= t)            O(n)
        sum++;
}
if option is set to 1: O(nlog2n) — but faster!!!
```

otherwise, O(n): *O(n*)

Recap: Demo (2)

A

```
for(i = 0; i < ARRAY_SIZE; i++)
{
    for(j = 0; j < ARRAY_SIZE; j++)
    {
        c[i][j] = a[i][j]+b[i][j];
    }
}
```

$O(n^2)$

A Lot Better!

B

```
for(j = 0; j < ARRAY_SIZE; j++)
{
    for(i = 0; i < ARRAY_SIZE; i++)
    {
        c[i][j] = a[i][j]+b[i][j];
    }
}
```

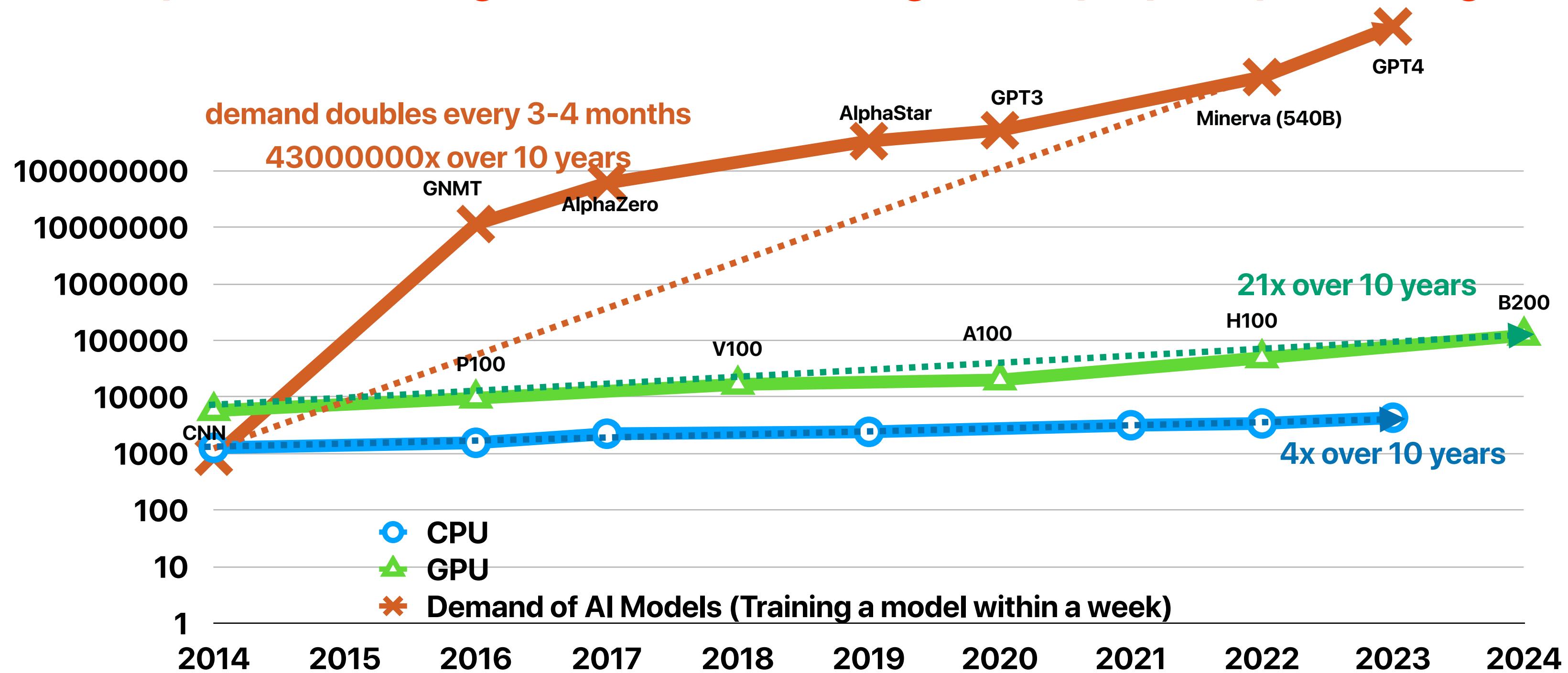
$O(n^2)$

Complexity

Performance?

Worse

Recap: Mis-matching AI/ML demand and general-purpose processing



<https://ourworldindata.org/grapher/artificial-intelligence-training-computation>

ChatGPT 3.5

threats, knowledge of security principles and cryptography are crucial for protecting data and systems from unauthorized attacks.

9. **Artificial Intelligence and Machine Learning:** AI and ML are transforming various industries and applications. Understanding concepts like neural networks, deep learning, reinforcement learning, and natural language processing is becoming increasingly important.

10. **Human-Computer Interaction (HCI):** HCI focuses on designing interactive systems that are user-friendly and efficient. Understanding user behavior, usability principles, and interaction design is essential for developing effective user interfaces.

These topics serve as foundational knowledge for computer scientists and are applicable across various domains and specialties within the field. Additionally, interdisciplinary areas such as bioinformatics, computational biology, and computational neuroscience are gaining importance, highlighting the interdisciplinary nature of computer science.

HT You
What is computer architecture?

ChatGPT

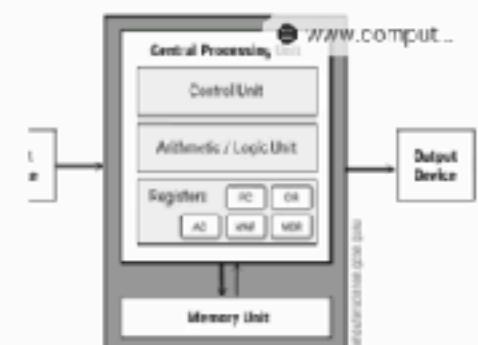
Message ChatGPT...

puter a

Gemini

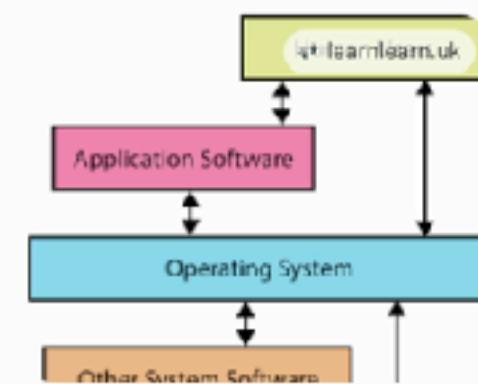
See the latest updates to the Gemini AI Beta Privacy Policy

well as the software that controls them. An understanding of computer architecture is essential for writing efficient programs and designing new computing systems.



Computer Architecture computer science

- Operating Systems: An operating system (OS) is the software that manages the hardware and software resources of a computer. It provides a platform for running other programs and applications. Operating systems are essential for the smooth operation of computers.



Operating Systems computer science

These are just a few of the most important topics in computer science. These foundational concepts are essential for understanding how computers work and for creating new technologies.

What is computer architecture?

Gemini may display inaccurate info, including about people, so double-check its responses. [Your privacy](#)

Which side do you feel the performance is better?

Best National

Schools in the National University are a full range of undergraduate research producing groundbreaking results.

To unlock full rankings, SAT/ACT scores

SUMMARY ▾



443

Scho

Sc

Loca

Cit

All

Rank

Nat

Best Computer Science Schools

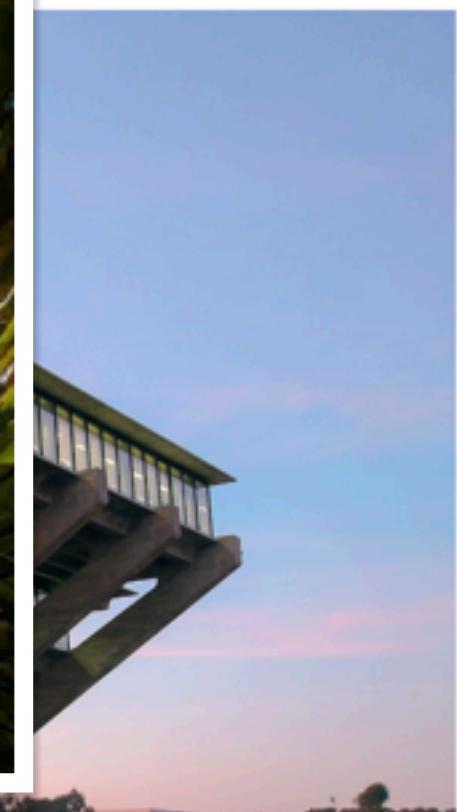
Ranked in 2022, part of Best Science Schools

Earning a graduate degree in computer technology companies and colleges are reflects its average rating on a scale from institutions. [Read the methodology](#) ▾



UC San Diego Ranked No. 1 Public University by Washington Monthly

Campus celebrated as a leader in social mobility, research and public

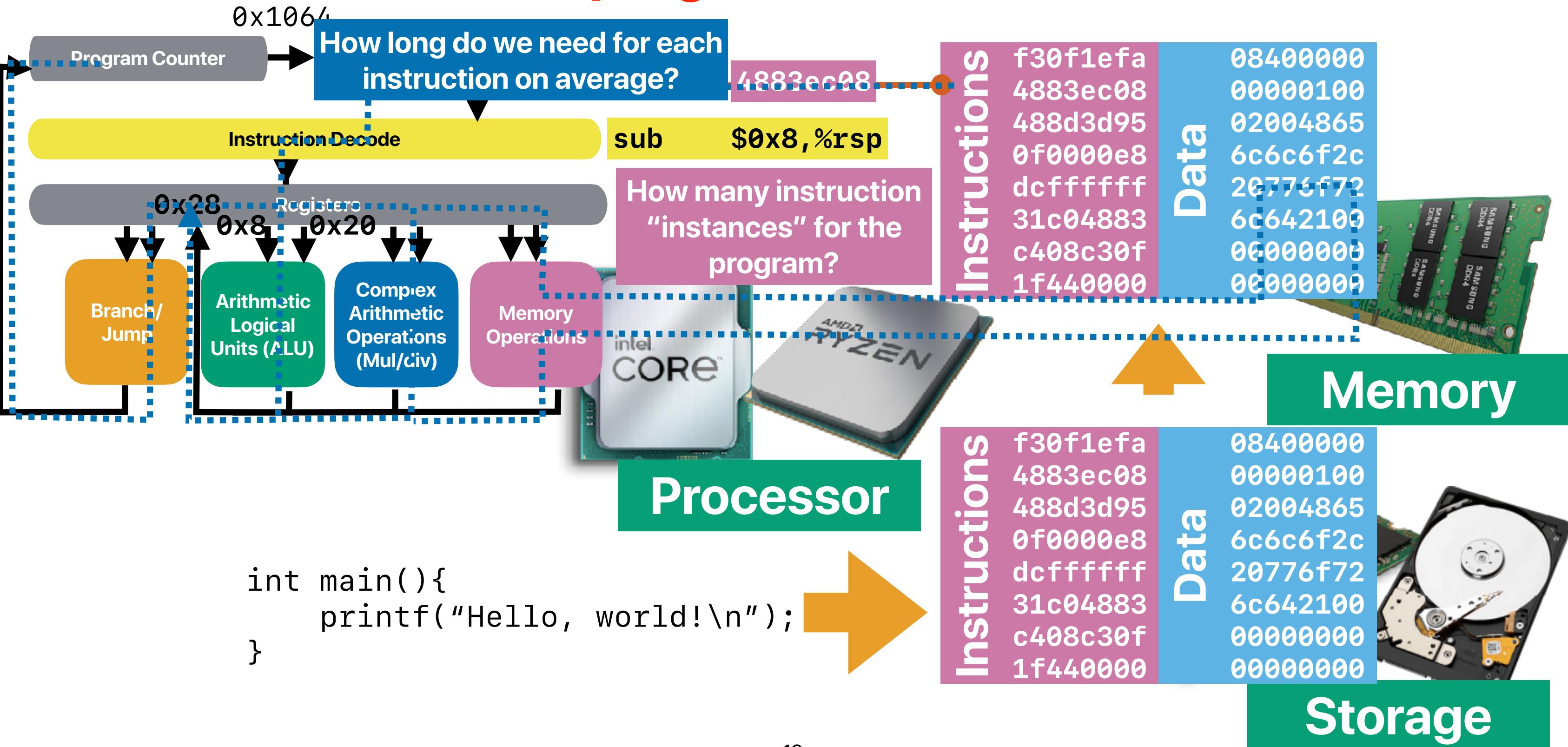


“End-to-end latency” — how long it takes to execute a program is the default/most intuitive performance metric

Outline

- Definition of “Performance”
- The classical CPU performance equation
- What affects performance
 - Programming languages
 - Programmers

Execution time of a program in the von Neumann model



CPU Performance Equation

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

$$Execution\ Time = \frac{Instructions}{Program} \times \frac{Cycles}{Instruction} \times \frac{Seconds}{Cycle}$$

$$ET = IC \times CPI \times CT$$

$$1GHz = 10^9Hz = \frac{1}{10^9}sec\ per\ cycle = 1\ ns\ per\ cycle$$

$\frac{1}{Frequency(i.e.,\ clock\ rate)}$

Classic CPU Performance Equation (ET of a program)

$$\text{Execution Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

How many instruction
 "instances" for the
 program?

How long do we need for each
 instruction on average?

C Code	x86 instructions
<pre>int init_data(int64_t *data, int data_size) { register unsigned int i = 0; for(i = 0; i < data_size; i++) { s+=data[i]; } return s; }</pre> <p style="color: orange; font-size: 2em; vertical-align: middle;">1000000000x</p>	<pre>init_data: .LFB16: endbr64 testl %esi, %esi jle .L2 leal -1(%rsi), %ecx xorq %rax, %rax .L3: movslq (%rdi), %rdx addq \$4, %rdi addq %rdx, %rax cmpq %rcx, %rdi jne .L3 .L2: xorlq %rax, %rax ret</pre>
<pre>int main(int argc, char **argv) { int *data = malloc(8000000000); init_data(data, 100000000) return 0; }</pre>	

If data memory access instructions takes 5 cycles,
branch 2 cycles, others take only 1 cycle, CPU freq. = 4 GHz

$$CPI_{average} = 20\% \times 5 + 20\% \times 2 + 60\% \times 1 = 2$$

←memory inst.

←branch inst.

$$ET = (5 \times 10^9) \times 2 \times \frac{1}{4 \times 10^9} \text{ sec} = 2.5 \text{ sec}$$

Takeaways: What is and what affects “performance”?

- Latency is the most fundamental performance metric —
Instruction count, cycles per instruction, cycle time define the latency of execution on CPUs

What Affects Each Factor in Performance Equation

Programming languages & performance

Use “performance counters” to figure out!

- Modern processors provides performance counters
 - instruction counts
 - cache accesses/misses
 - branch instructions/mis-predictions
- How to get their values?
 - You may use “perf stat” in linux
 - You may use Instruments —> Time Profiler on a Mac
 - Intel’s vtune — only works on Windows w/ intel processors
 - You can also create your own functions to obtain counter values



What's the Opportunity?

Matrix Multiply: relative speedup to a Python version (18 core Intel)

Version	Speed-up	Optimization
Python	1	
C	47	Translate to static, compiled language
C with parallel loops	366	Extract parallelism
C with loops & memory optimization	6,727	Organize parallelism and memory access
Intel AVX instructions	62,806	Use domain-specific HW

from: "There's Plenty of Room at the Top," Leiserson, et. al., *to appear*.

Programming languages

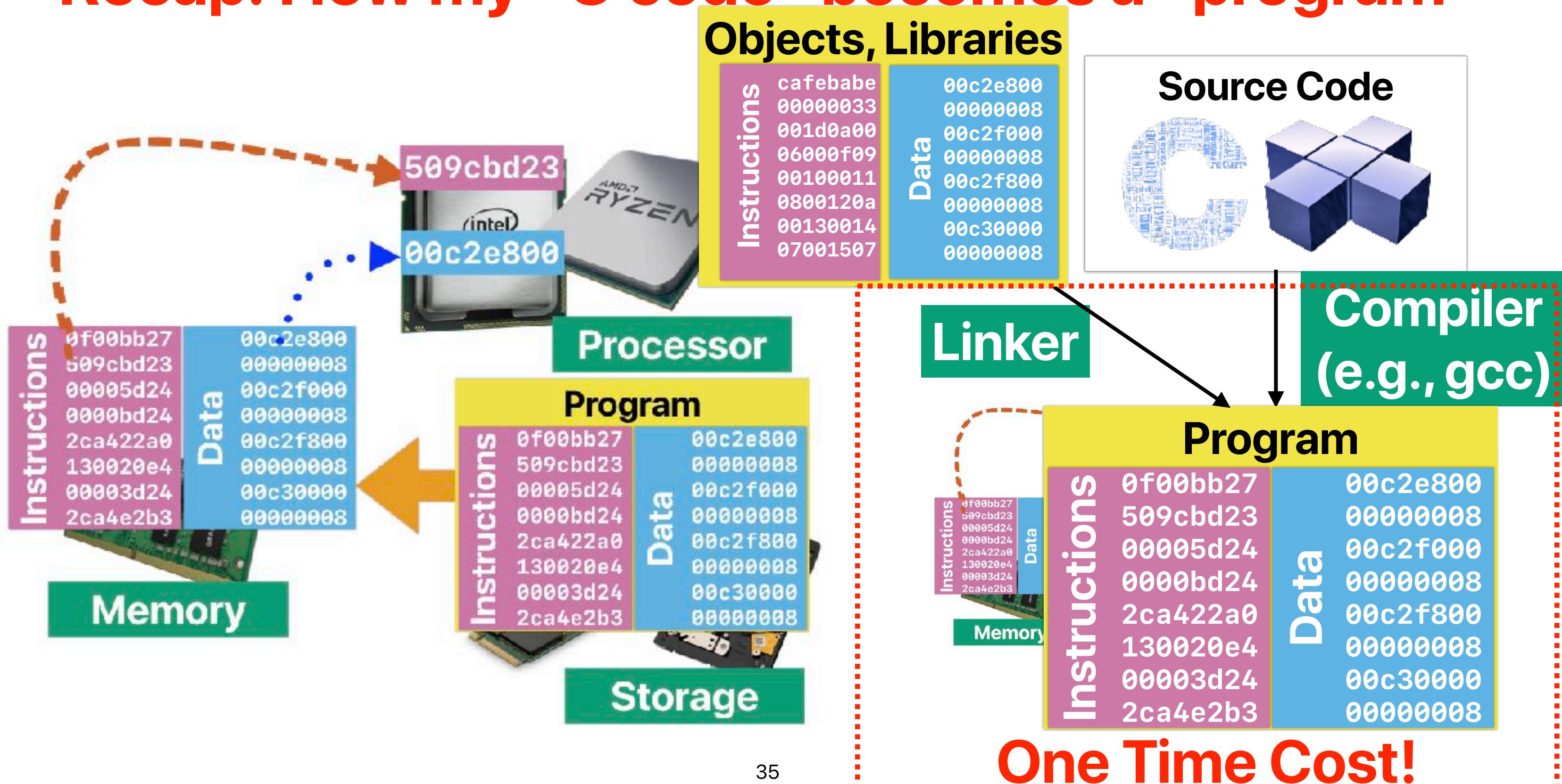
- How many instructions are there in “Hello, world!”

	Instruction count	LOC	Ranking
C	600k	6	1
C++	3M	6	2
Java	~145M	8	5
Perl	~12M	4	3
Python	~33M	1	4
GO (Interpreter)	~1200M	1	6
GO (Compiled)	~1.7M	1	
Rust	~1.4M	1	

Programming languages

- Which of the following programming language needs to highest instruction count to print “Hello, world!” on screen?
 - A. C
 - B. C++
 - C. Java
 - D. Perl
 - E. Python

Recap: How my “C code” becomes a “program”



Start with this simple program in C

```
int A[] =  
{1,2,3,4,5,6,7,8,9,10,1,2,3,4  
,5,6,7,8,9,10};
```

Compiler

Contents of section .data:
0000 01000000 02000000 03000000 04000000
0010 05000000 06000000 07000000 08000000
0020 09000000 0a000000 0b000000 0c000000
0030 03000000 04000000 05000000 06000000
0040 07000000 08000000 09000000 0a000000

control flow
operations
logical operations

```
int main()  
{  
    int i=0, sum=0;  
    for(i = 0; i < 20; i++)  
    {  
        sum += A[i];  
    }  
    return 0;  
}
```

memory access
arithmetic operations

main:
.LFB0:
endbr64
pushq %rbp
movq %rsp, %rbp
movl \$0, -8(%rbp)
movl \$0, -4(%rbp)
movl \$0, -8(%rbp)
jmp .L2

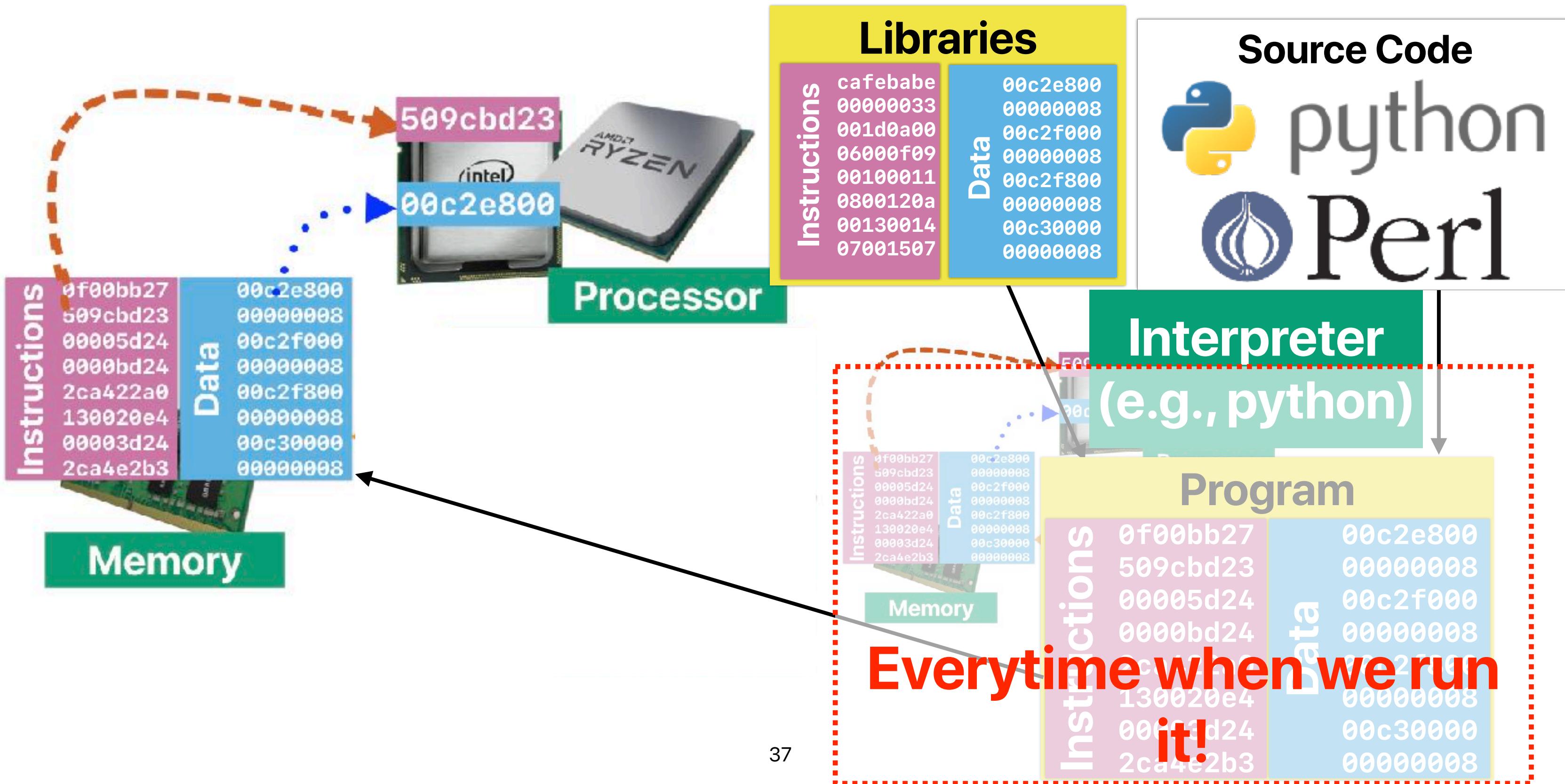
.L3:
movl -8(%rbp), %eax
cltq
leaq 0(%rax,4), %rdx
leaq A(%rip), %rax

Compiler

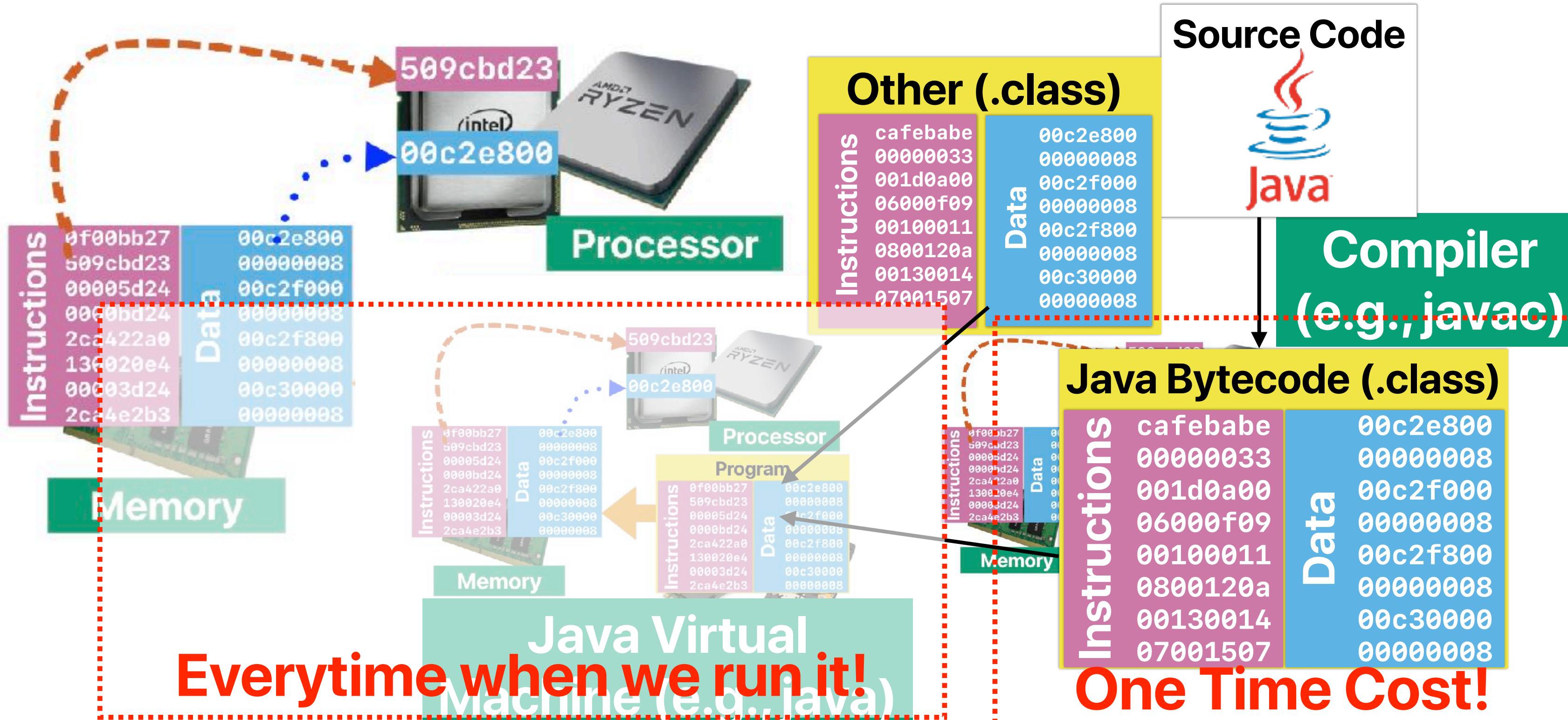
Contents of section .text:
0000 f30f1efa 554889e5 c745f800 000000c7
0010 45fc0000 0000c745 f8000000 00eb1e8b
0020 45f84898 488d1405 00000000 488d0500
0030 0000008b 04020145 fc8345f8 01837df8
0040 137edcb8 00000000 5dc3

Instructions

Recap: How my “Python code” becomes a “program”

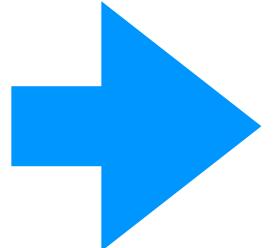


Recap: How my “Java code” becomes a “program”



What's in your java classes?

```
public static int fibonacci(int n) {  
    if(n == 0)  
        return 0;  
    else if(n == 1)  
        return 1;  
    else  
        return fibonacci(n - 1) + fibonacci(n - 2);  
}
```

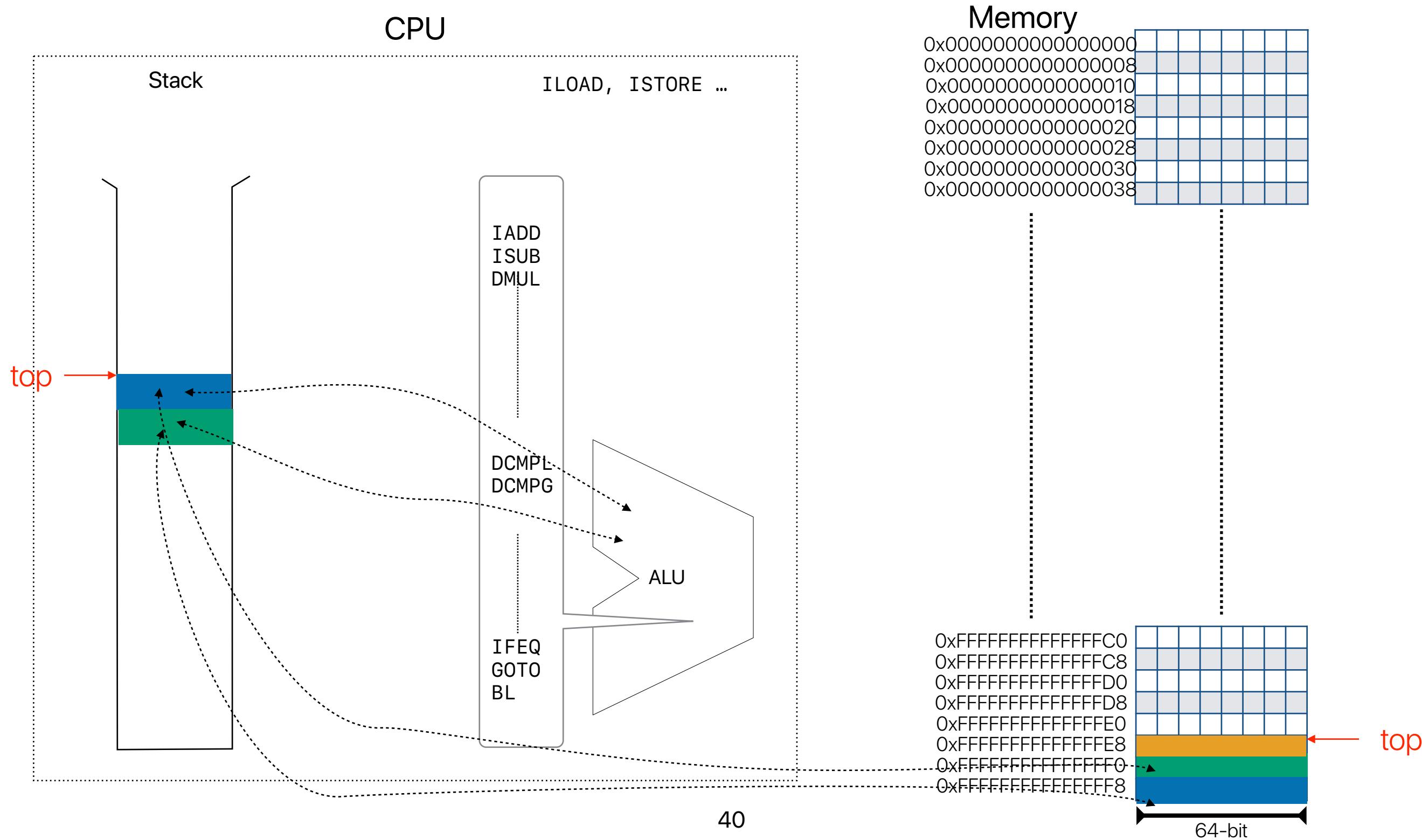


**Most instructions doesn't
have an argument!**

0: iload_0	6
1: ifne	
4: iconst_0	
5: ireturn	
6: iload_0	
7: iconst_1	
8: if_icmpne	13
11: iconst_1	
12: ireturn	
13: iload_0	
14: iconst_1	
15: isub	
16: invokestatic #2	#2
19: iload_0	
20: iconst_2	
21: isub	
22: invokestatic #2	#2
25: iadd	
26: ireturn	

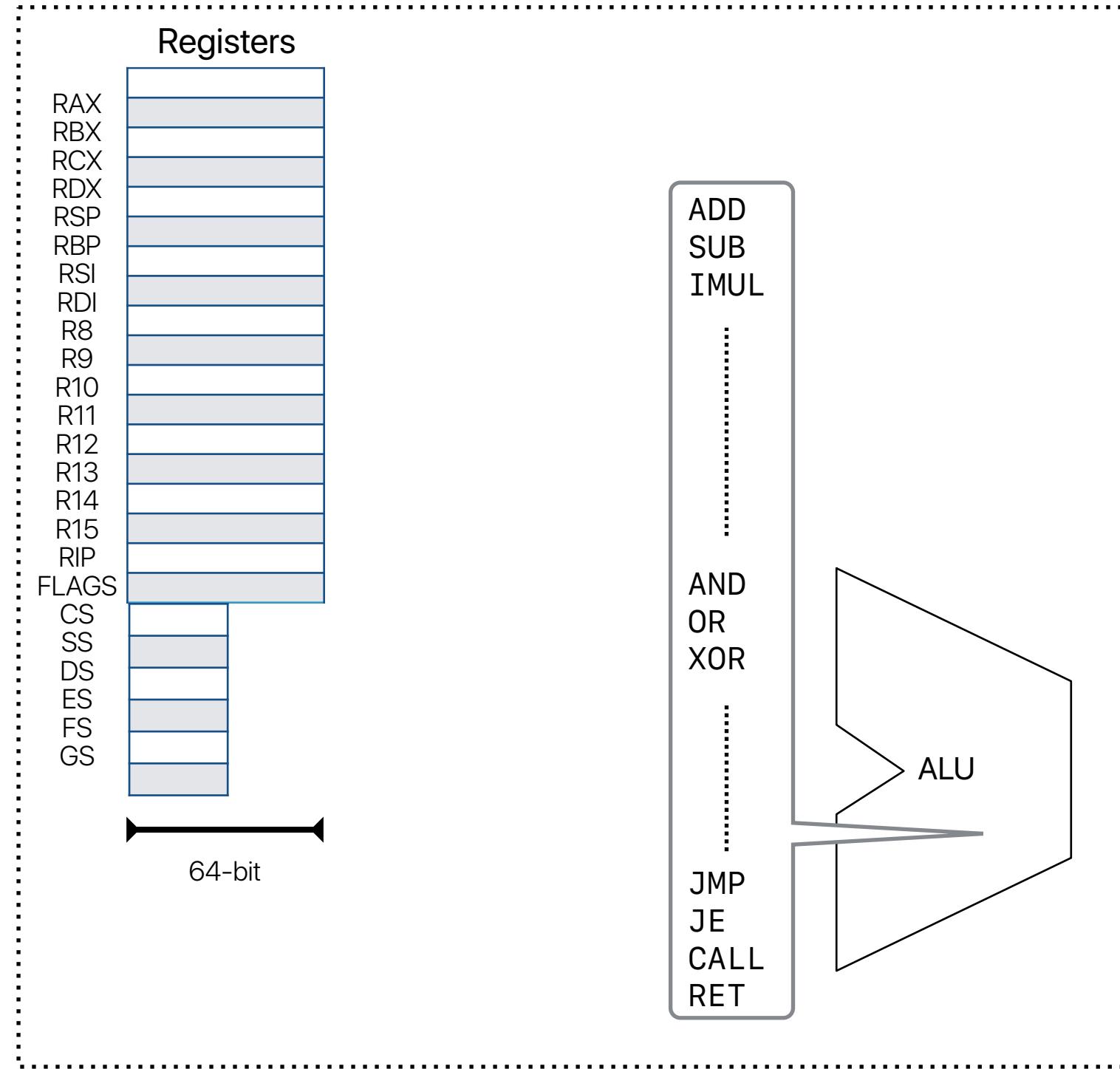
labels

“Abstracted” Java Architecture

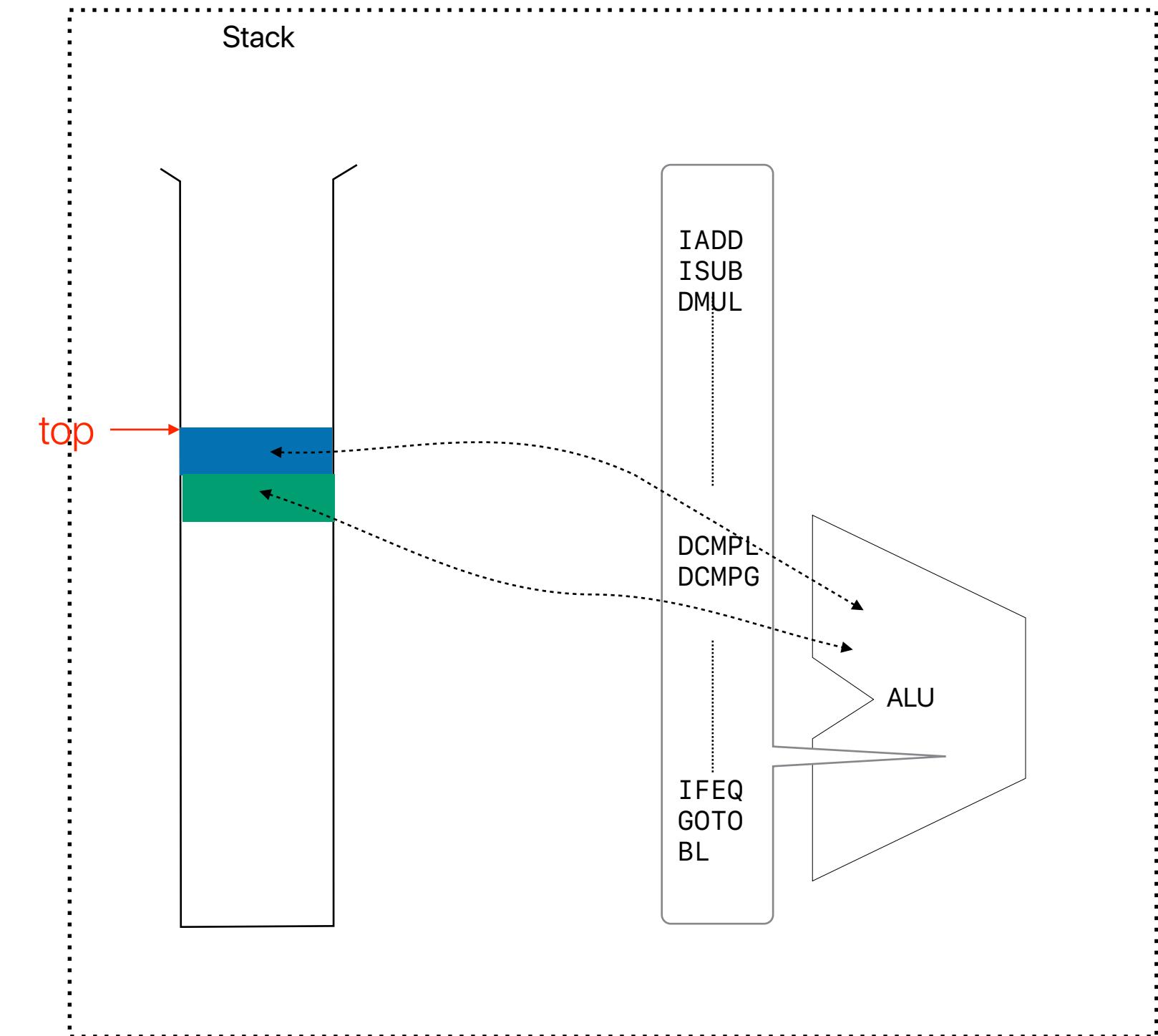


Mismatching real architectures vs JVM abstraction

x86 CPU



JVM's CPU abstraction



Takeaways: What is and what affects “performance”?

- Latency is the most fundamental performance metric — Instruction count, cycles per instruction, cycle time define the latency of execution on CPUs
- Different programming languages can generate machine operations with different orders of magnitude performance — programmers need to make wise choice of that!

**What about programmer &
performance?**

Demo — programmer & performance

A

```
for(i = 0; i < ARRAY_SIZE; i++)  
{  
    for(j = 0; j < ARRAY_SIZE; j++)  
    {  
        c[i][j] = a[i][j]+b[i][j];  
    }  
}
```

B

```
for(j = 0; j < ARRAY_SIZE; j++)  
{  
    for(i = 0; i < ARRAY_SIZE; i++)  
    {  
        c[i][j] = a[i][j]+b[i][j];  
    }  
}
```

$O(n^2)$

Complexity

$O(n^2)$

Instruction Count?

Clock Rate

CPI

Programmers can also set the cycle time

<https://software.intel.com/sites/default/files/comment/1716807/how-to-change-frequency-on-linux-pub.txt>

```
=====
Subject: setting CPU speed on running linux system
```

If the OS is Linux, you can manually control the CPU speed by reading and writing some virtual files in the "/proc"

1.) Is the system capable of software CPU speed control?

If the "directory" /sys/devices/system/cpu/cpu0/cpufreq exists, speed is controllable.

-- If it does not exist, you may need to go to the BIOS and turn on EIST and any other C and P state control and vi:

2.) What speed is the box set to now?

Do the following:

```
$ cd /sys/devices/system/cpu
$ cat ./cpu0/cpufreq/cpuinfo_max_freq
3193000
$ cat ./cpu0/cpufreq/cpuinfo_min_freq
1596000
```

3.) What speeds can I set to?

Do

```
$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
```

It will list highest settable to lowest; example from my NHM "Smackover" DX58SO HEDT board, I see:

```
3193000 3192000 3059000 2926000 2793000 2660000 2527000 2394000 2261000 2128000 1995000 1862000 1729000 1596000
```

You can choose from among those numbers to set the "high water" mark and "low water" mark for speed. If you set "h

4.) Show me how to set all to highest settable speed!

Use the following little sh/ksh/bash script:

```
$ cd /sys/devices/system/cpu # a virtual directory made visible by device drivers
$ newSpeedTop=`awk '{print $1}' ./cpu0/cpufreq/scaling_available_frequencies`
$ newSpeedLcw=$newSpeedTop # make them the same in this example
$ for c in ./cpu[0-9]* ; do
>   echo $newSpeedTop >${c}/cpufreq/scaling_max_freq
>   echo $newSpeedLow >${c}/cpufreq/scaling_min_freq
> done
$
```

5.) How do I return to the default - i.e. allow machine to vary from highest to lowest?

Edit line # 3 of the script above, and re-run it. Change the line:

```
$ newSpeedLcw=$newSpeedTop # make them the same in this example
```

To read

Check your assignment 1 regarding your power in cycle time!

Toggle Code Cells o.s. instruction Count

6.3.1. Running The Experiment Mutiple Times

6.3.2. Increasing the Size of array

6.3.3. Question 3 (Correctness)

6.4. Cycle Time

6.4.1. Question 4 (Completeness)

6.5. Cycles Per Instruction

6.5.1. Floating Point vs Integer Operations

6.5.2. Question 5 (Completeness)

6.5.3. Question 6 (Completeness)

6.5.4. The Compiler's Effect

6.5.5. Question 7 (Correctness)

6.5.6. Question 8 (Correctness)

6.5.7. Code Structure

6.5.8. Question 9 (Correctness)

6.5.9. Question 10 (Correctness)

7. Amdahl's Law

7.0.1. Question 11 (Correctness)

7.0.2. Question 12 (Challenging)

7.0.3. Question 13 (Challenging)

Python 3 (ipyker)

6.4. Cycle Time

Next, we'll take a look at how clock rate affects performance. Before we do, though, let's see what our options are for clock rate on our machine:

```
[26]: !lcs203 run 'cpupower frequency-info -n'
```

```
analyzing CPU 0:  
driver: intel_pstate  
CPUs which run at the same hardware frequency: 0  
CPUs which need to have their frequency coordinated by software: 0  
maximum transition latency: Cannot determine or is not supported.  
hardware limits: 800.000 MHz - 4.300000 GHz  
available cpufreq governors: performance powersave  
current policy: frequency should be within 800.000 MHz and 4.300000 GHz.  
The governor "powersave" may decide which speed to use  
within this range.  
current CPU frequency: Unable to call hardware  
current CPU frequency: 800.000 MHz (asserted by call to kernel)  
boost state support:  
Supported: yes  
Active: yes
```

As you can see, the processors in our target systems can run between 800MHz and 4300MHz.

Let's see how that affects things by plotting execution time as a function of clock speed (we are skipping 800MHz for the moment. We'll come back to it.). The readings for the current clock speed may vary from run to run. It just ends up at whatever the last experiment left it at.

Kick off the cell below to collect the data. While it's running answer this question:

```
[27]: !lcs203 run './microbench.exe -o cycle_time.csv -M 800 1600 2000 3200 4000 4300 -f baseline_int -r 50'
```

```
Execution complete
```

Takeaways: What is and what affects “performance”?

- Latency is the most fundamental performance metric —
Instruction count, cycles per instruction, cycle time define the latency of execution on CPUs
- Different programming languages can generate machine operations with different orders of magnitude performance —
programmers need to make wise choice of that!
- Programmers can control all three factors in the classic performance equation when composing the program

Compilers & performance

Takeaways: What is and what affects “performance”?

- Latency is the most fundamental performance metric — Instruction count, cycles per instruction, cycle time define the latency of execution on CPUs
- Different programming languages can generate machine operations with different orders of magnitude performance — programmers need to make wise choice of that!
- Programmers can control all three factors in the classic performance equation when composing the program
- Compiler optimization does not always help
 - Compiler optimizes code based on some assumptions that may not be true on all computers
 - Programmers' can write code in a way facilitating optimizations!

What matters to the classical CPU performance equation?

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

$$\text{Execution Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

$$ET = IC \times CPI \times CT$$

- IC (Instruction Count)
 - ISA, Compiler, algorithm, programming language, **programmer**
- CPI (Cycles Per Instruction)
 - Machine Implementation, microarchitecture, compiler, application, algorithm, programming language, **programmer**
- Cycle Time (Seconds Per Cycle)
 - Process Technology, microarchitecture, **programmer**

How about complexity?

How about “computational complexity”

- Algorithm complexity provides a good estimate on the performance if —
 - Every instruction takes exactly the same amount of time
 - Every operation takes exactly the same amount of instructions

These are unlikely to be true

Takeaways: What is and what affects “performance”?

- Latency is the most fundamental performance metric — Instruction count, cycles per instruction, cycle time define the latency of execution on CPUs
- Different programming languages can generate machine operations with different orders of magnitude performance — programmers need to make wise choice of that!
- Programmers can control all three factors in the classic performance equation when composing the program
- Compiler optimization does not always help
 - Compiler optimizes code based on some assumptions that may not be true on all computers
 - Programmers' can write code in a way facilitating optimizations!
- Complexity does not provide good assessment on real machines due to the idealized assumptions

If there was one right thing that
you've done, what & why is that?

Outline

- What does better mean?
- Amdahl's Law and its implications

Quantitative Analysis of “Better”

Speedup

- The relative performance between two machines, X and Y. Y is n times faster than X

$$n = \frac{\text{Execution Time}_X}{\text{Execution Time}_Y}$$

- The speedup of Y over X

$$\text{Speedup} = \frac{\text{Execution Time}_X}{\text{Execution Time}_Y}$$

Takeaways: What is and what affects “performance”?

- The only definition of Y is Speedup times faster than X —

$$\text{Speedup} = \frac{\text{Execution Time}_X}{\text{Execution Time}_Y}$$

Amdahl's Law in the

Amdahl's Law — and It's Implication in the Multicore Era

Mark D. Hill, University of Wisconsin-Madison
Michael R. Marty, Google

Augmenting Amdahl's law with a corollary for multicore hardware makes it relevant to future generations of chips with multiple processor cores. Obtaining optimal multicore performance will require further research in both extracting more parallelism and making sequential cores faster.

Mark D. Hill, University of Wisconsin-Madison

Michael R. Marty, Google

In IEEE Computer, vol. 41, no. 7

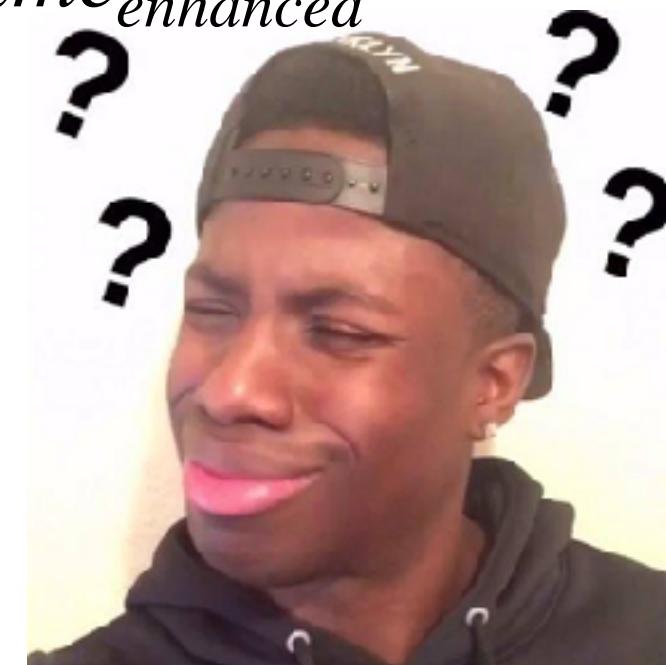
Amdahl's Law



$$\text{Speedup}_{\text{enhanced}}(f, s) = \frac{1}{(1-f) + \frac{f}{s}}$$

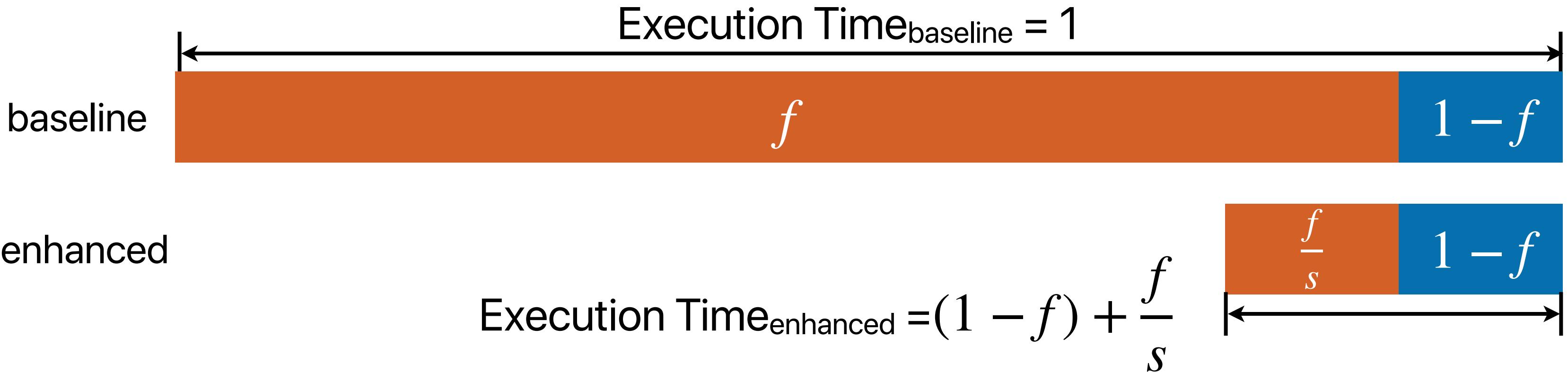
- f — The fraction of time in the original program
 s — The speedup we can achieve on f

$$\text{Speedup}_{\text{enhanced}} = \frac{\text{Execution Time}_{\text{baseline}}}{\text{Execution Time}_{\text{enhanced}}}$$



Amdahl's Law

$$Speedup_{enhanced}(f, s) = \frac{1}{(1 - f) + \frac{f}{s}}$$



$$Speedup_{enhanced} = \frac{Execution\ Time_{baseline}}{Execution\ Time_{enhanced}} = \frac{1}{(1 - f) + \frac{f}{s}}$$

FINAL FANTASY XV

POCKET EDITION HD



Amdahl's Law on Multiple Optimizations

- We can apply Amdahl's law for multiple optimizations
- These optimizations must be dis-joint!
 - If optimization #1 and optimization #2 are dis-joint:



$$Speedup_{enhanced}(f_{Opt1}, f_{Opt2}, s_{Opt1}, s_{Opt2}) = \frac{1}{(1 - f_{Opt1} - f_{Opt2}) + \frac{f_{Opt1}}{s_{Opt1}} + \frac{f_{Opt2}}{s_{Opt2}}}$$

- If optimization #1 and optimization #2 are not dis-joint:



$$Speedup_{enhanced}(f_{OnlyOpt1}, f_{OnlyOpt2}, f_{BothOpt1Opt2}, s_{OnlyOpt1}, s_{OnlyOpt2}, s_{BothOpt1Opt2}) = \frac{1}{(1 - f_{OnlyOpt1} - f_{OnlyOpt2} - f_{BothOpt1Opt2}) + \frac{f_{BothOpt1Opt2}}{s_{BothOpt1Opt2}} + \frac{f_{OnlyOpt1}}{s_{OnlyOpt1}} + \frac{f_{OnlyOpt2}}{s_{OnlyOpt2}}}$$

Amdahl's Law Corollary #1

- The maximum speedup is bounded by

$$\text{Speedup}_{max}(f, \infty) = \frac{1}{(1-f) + \frac{f}{\infty}}$$

$$\text{Speedup}_{max}(f, \infty) = \frac{1}{(1-f)}$$

Intel kills the remnants of Optane memory

The speed-boosting storage tech was already on the ropes.



By [Michael Crider](#)

Staff Writer, PCWorld | JUL 29, 2022 6:59 AM PDT

Optane hy

1TB WD Bla

1TB Samsung EVO 9

You spent a lot of money but can only get 20% performance gain!



Image: Intel

MILLIPORE
SIGMA



MISSION® es

targeting mol
2010204k13r

Corollary #1 on Multiple Optimizations

- If we can pick just one thing to work on/optimize



$$Speedup_{max}(f_1, \infty) = \frac{1}{(1 - f_1)}$$

$$Speedup_{max}(f_2, \infty) = \frac{1}{(1 - f_2)}$$

$$Speedup_{max}(f_3, \infty) = \frac{1}{(1 - f_3)}$$

$$Speedup_{max}(f_4, \infty) = \frac{1}{(1 - f_4)}$$

The biggest f_x would lead to the largest $Speedup_{max}$!

Corollary #2 — make the common case fast!

- When f is small, optimizations will have little effect.
- Common == **most time consuming** not necessarily the most frequent
- The uncommon case doesn't make much difference
- The common case can change based on inputs, compiler options, optimizations you've applied, etc.

Takeaways: find the right thing to do

- Definition of “Speedup of Y over X” or say Y is n times faster than X:

$$speedup_{Y_over_X} = n = \frac{Execution\ Time_X}{Execution\ Time_Y}$$

- Amdahl's Law — $Speedup_{enhanced}(f, s) = \frac{1}{(1-f) + \frac{f}{s}}$

$$Speedup_{max}(f, \infty) = \frac{1}{(1-f)}$$

- Corollary 1 — each optimization has an upper bound

- Corollary 2 — make the common case (the most time consuming case) fast!

$$Speedup_{max}(f_1, \infty) = \frac{1}{(1-f_1)}$$

$$Speedup_{max}(f_2, \infty) = \frac{1}{(1-f_2)}$$

$$Speedup_{max}(f_3, \infty) = \frac{1}{(1-f_3)}$$

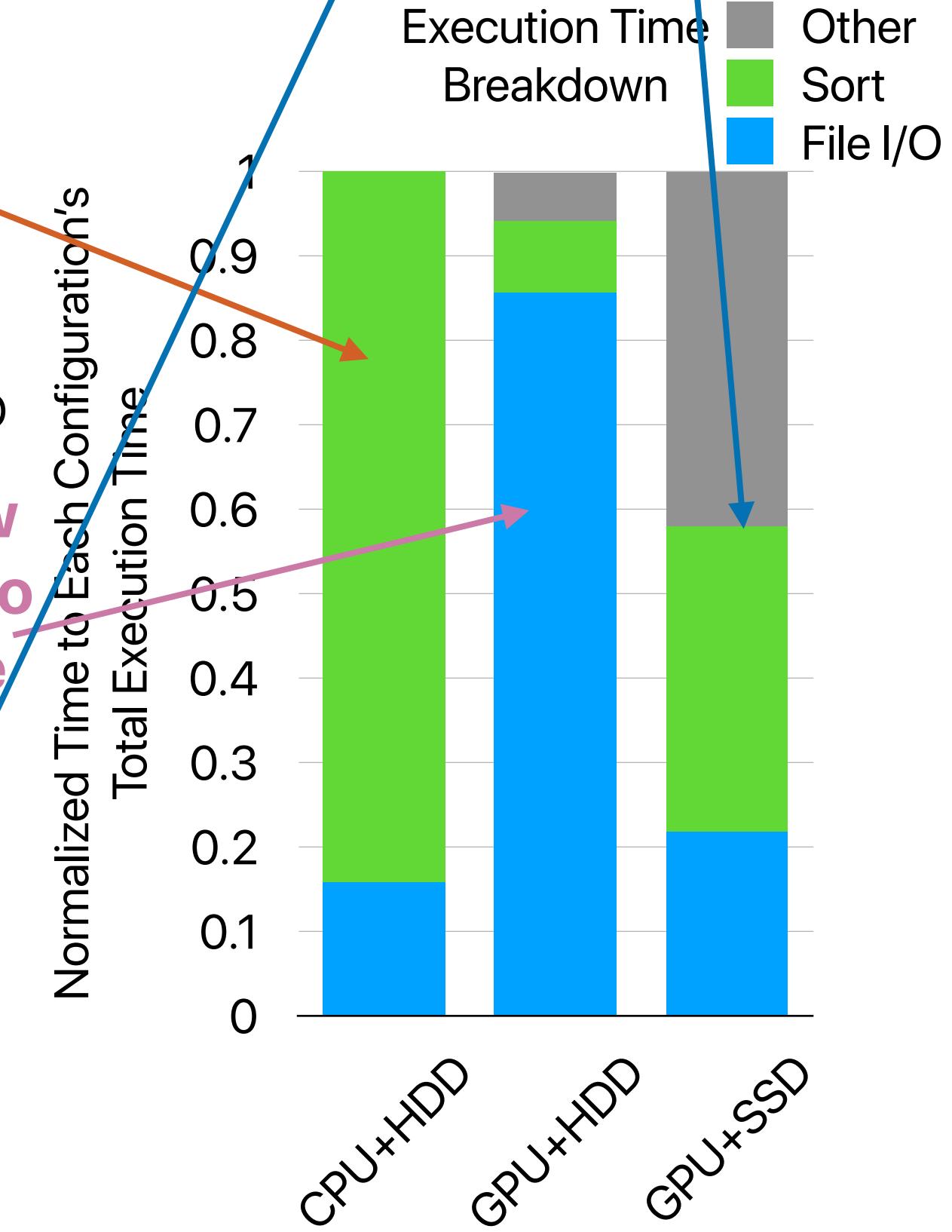
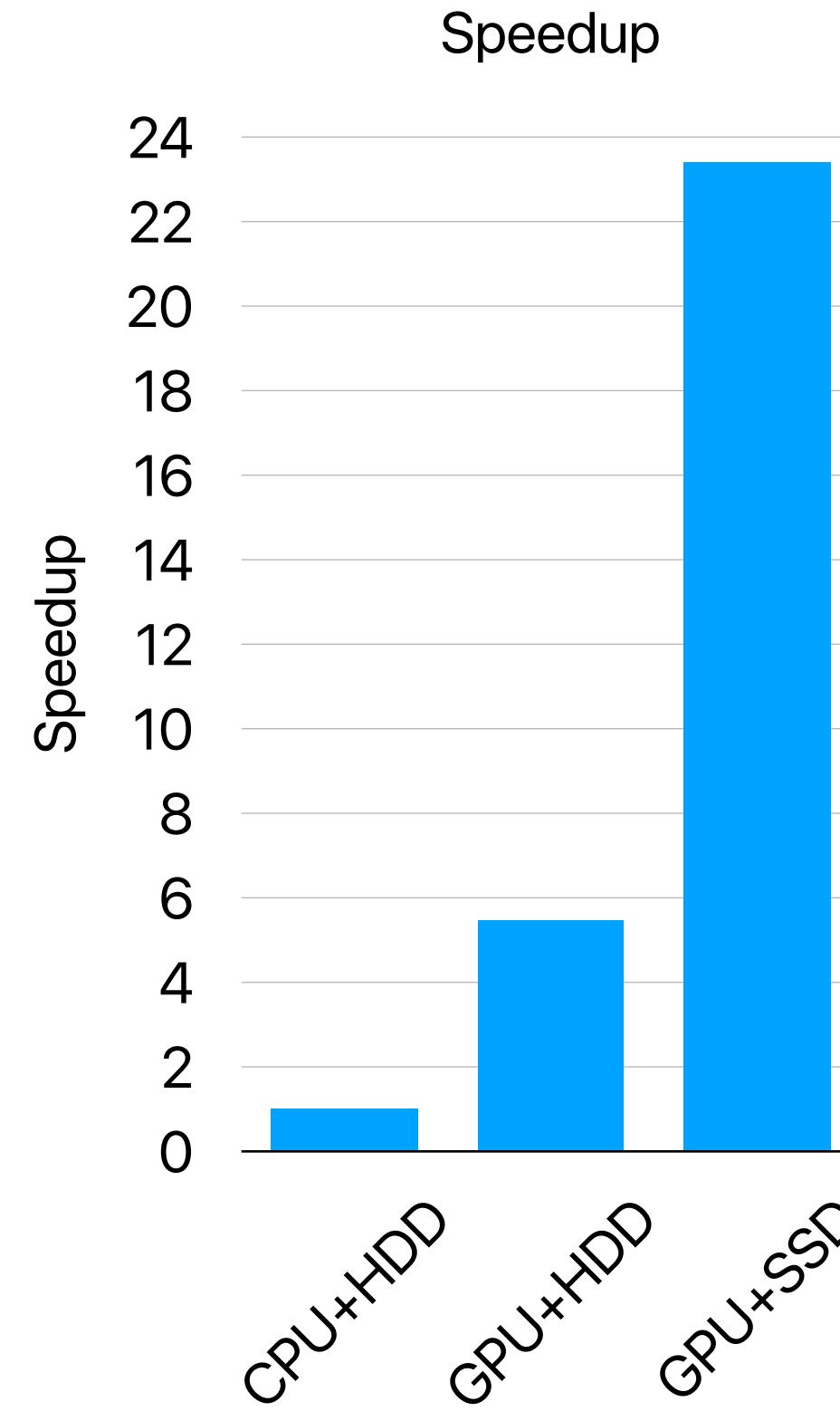
$$Speedup_{max}(f_4, \infty) = \frac{1}{(1-f_4)}$$

Identify the most time consuming part

- Use perf
 - perf record command
 - perf report —stdio
- Use prof
 - Compile your program with -pg flag
 - Run the program
 - It will generate a gmon.out
 - gprof your_program gmon.out > your_program.prof
 - It will give you the profiled result in your_program.prof
 - Or insert timestamps

Demo — sort

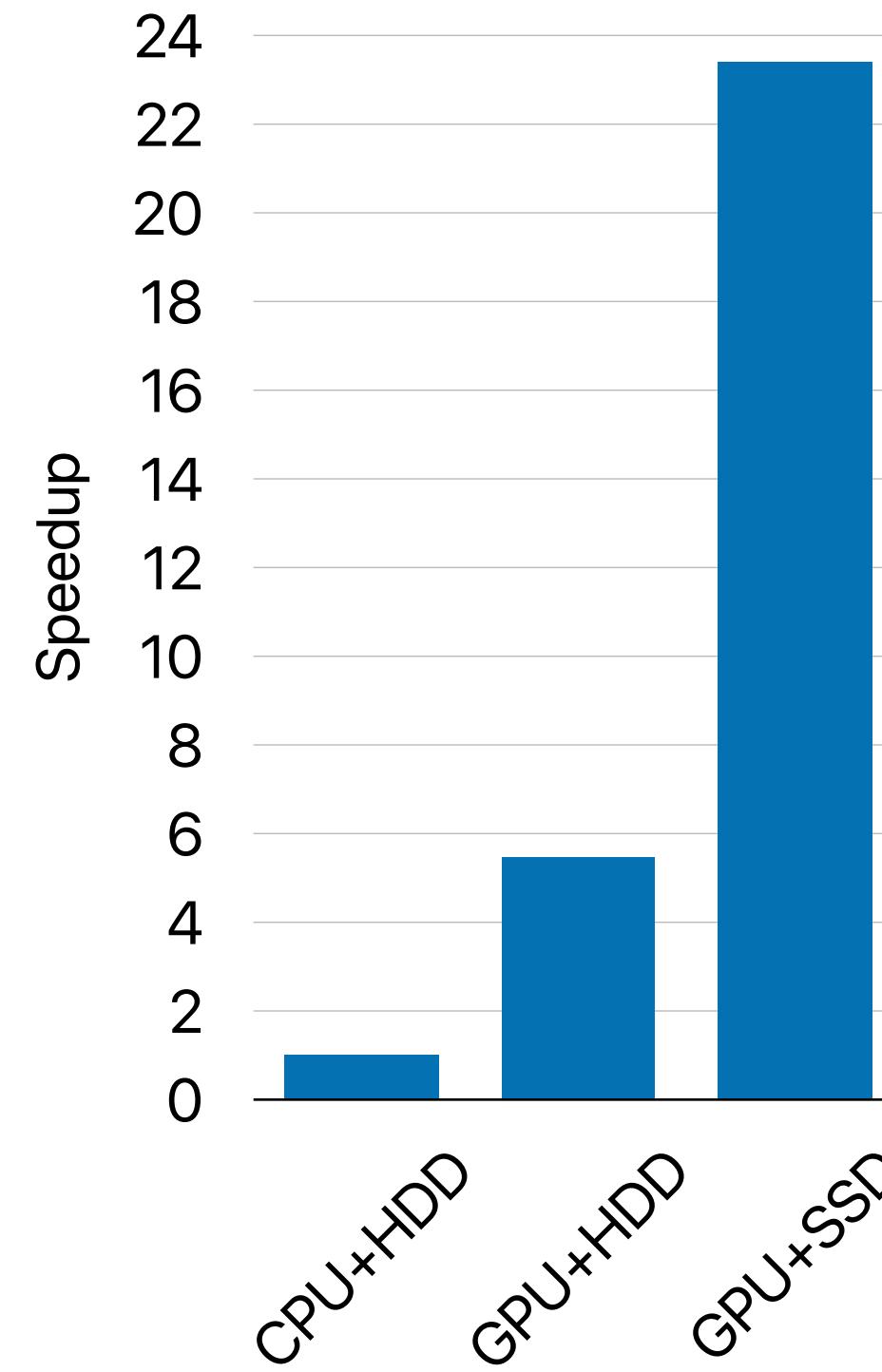
Something else (e.g., data movement) matters more



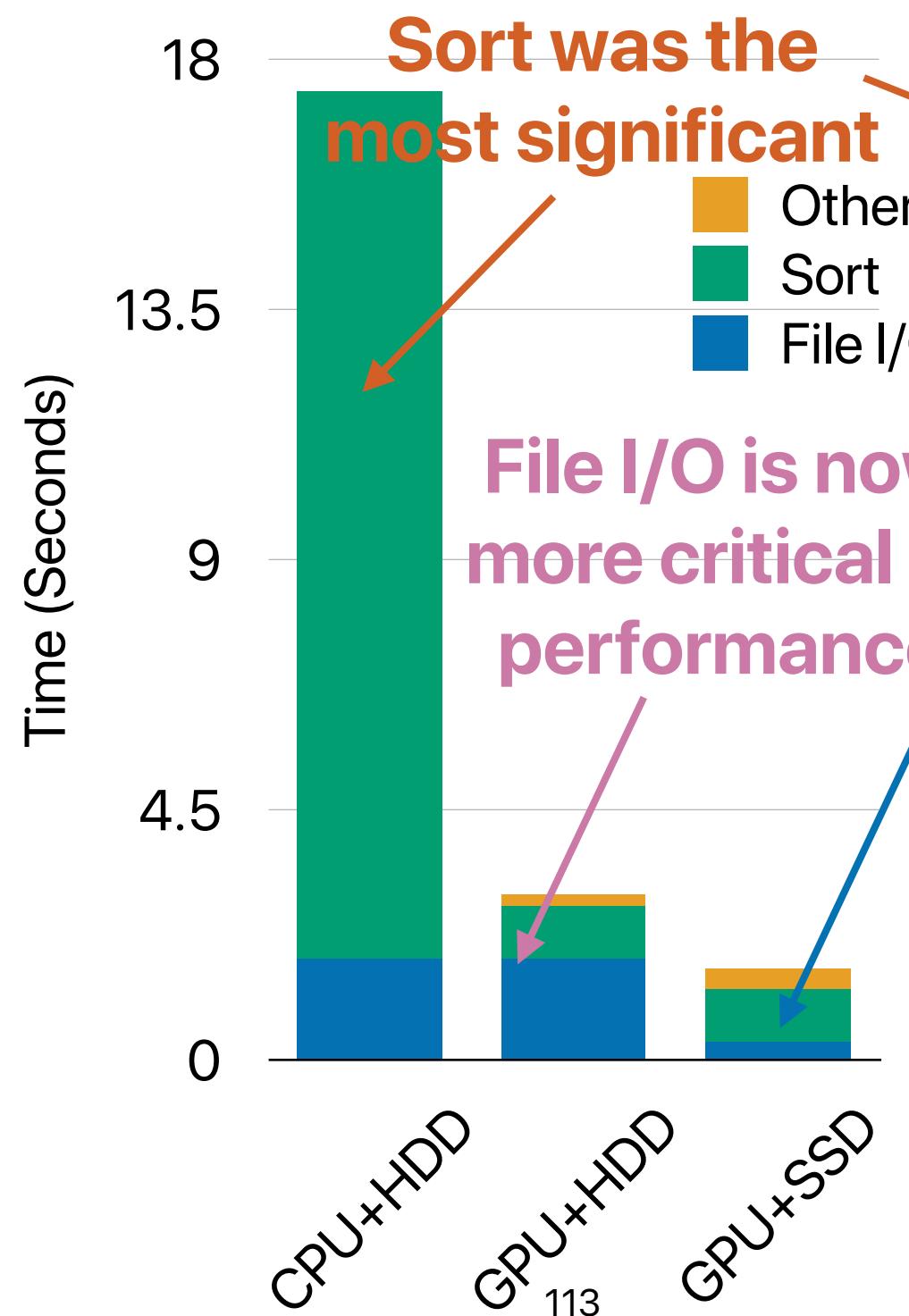
Demo — sort

Something else (e.g., data movement) matters more

Speedup



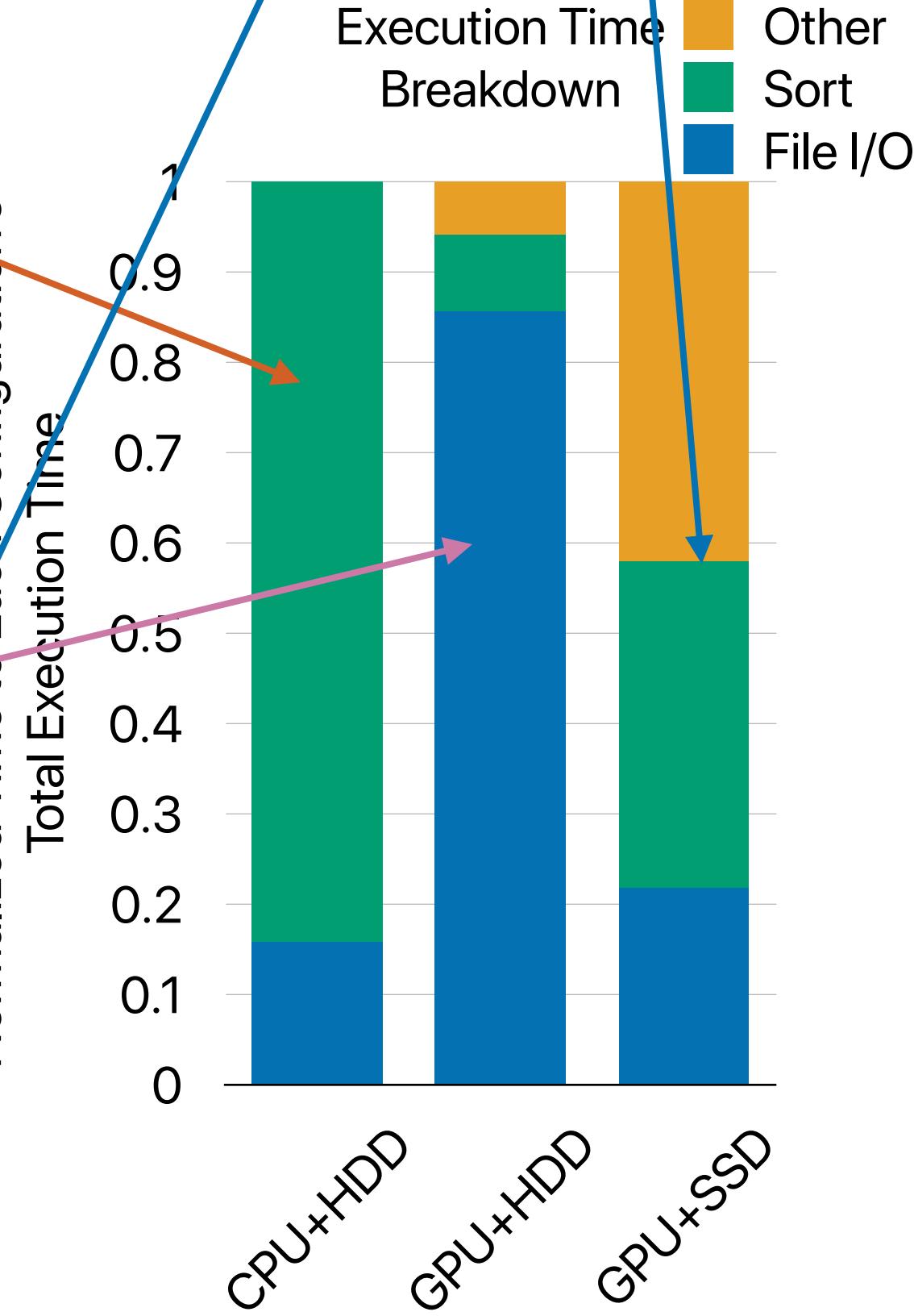
Cumulative Execution Time



Sort was the
most significant

File I/O is now
more critical to
performance

Normalized Time to Each Configuration's Total Execution Time



Execution Time Breakdown

Corollary #3: optimization has a moving target



- With optimization, the common becomes uncommon.
- An uncommon case will (hopefully) become the new common case.
- Now you have a new target for optimization — You have to revisit “Amdahl’s Law” every time you applied some optimization

Something else (e.g., data movement) matters more now

Takeaways: find the right thing to do

- Definition of “Speedup of Y over X” or say Y is n times faster than X:

$$speedup_{Y_over_X} = n = \frac{Execution\ Time_X}{Execution\ Time_Y}$$

- Amdahl's Law — $Speedup_{enhanced}(f, s) = \frac{1}{(1-f) + \frac{f}{s}}$
- Corollary 1 — each optimization has an upper bound
- Corollary 2 — make the common case (the most time consuming case) fast!
- Corollary 3 — Optimization has a moving target

$$Speedup_{max}(f, \infty) = \frac{1}{(1-f)}$$

$$Speedup_{max}(f_1, \infty) = \frac{1}{(1-f_1)}$$

$$Speedup_{max}(f_2, \infty) = \frac{1}{(1-f_2)}$$

$$Speedup_{max}(f_3, \infty) = \frac{1}{(1-f_3)}$$

$$Speedup_{max}(f_4, \infty) = \frac{1}{(1-f_4)}$$

Amdahl's Law on Multicore Architectures

- Symmetric multicore processor with n cores (if we assume the processor performance scales perfectly)

$$\text{Speedup}_{\text{parallel}}(f_{\text{parallelizable}}, n) = \frac{1}{(1 - f_{\text{parallelizable}}) + \frac{f_{\text{parallelizable}}}{n}}$$

Demo — merge sort v.s. bitonic sort on GPUs

Merge Sort

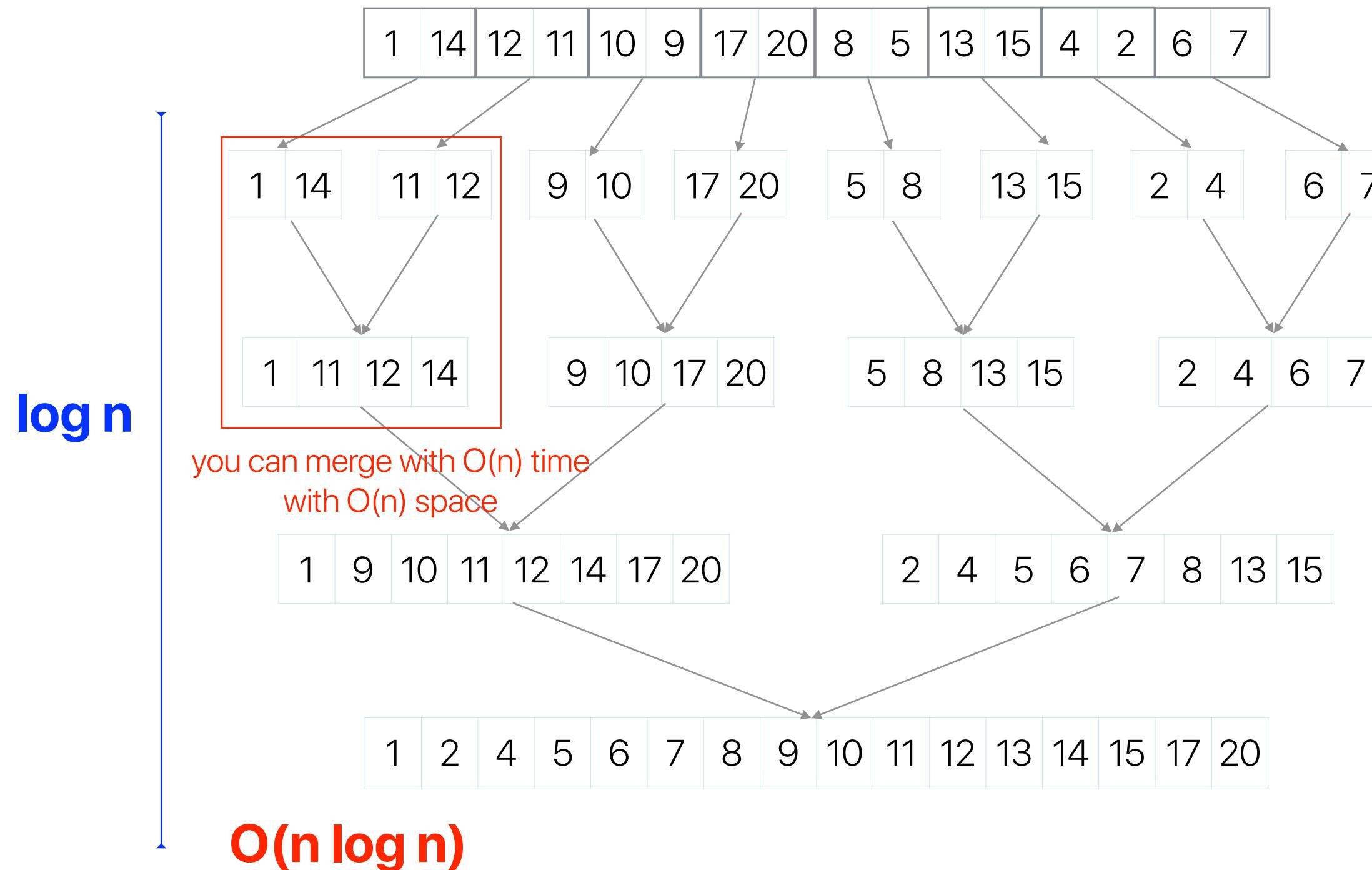
$$O(n \log_2 n)$$

Bitonic Sort

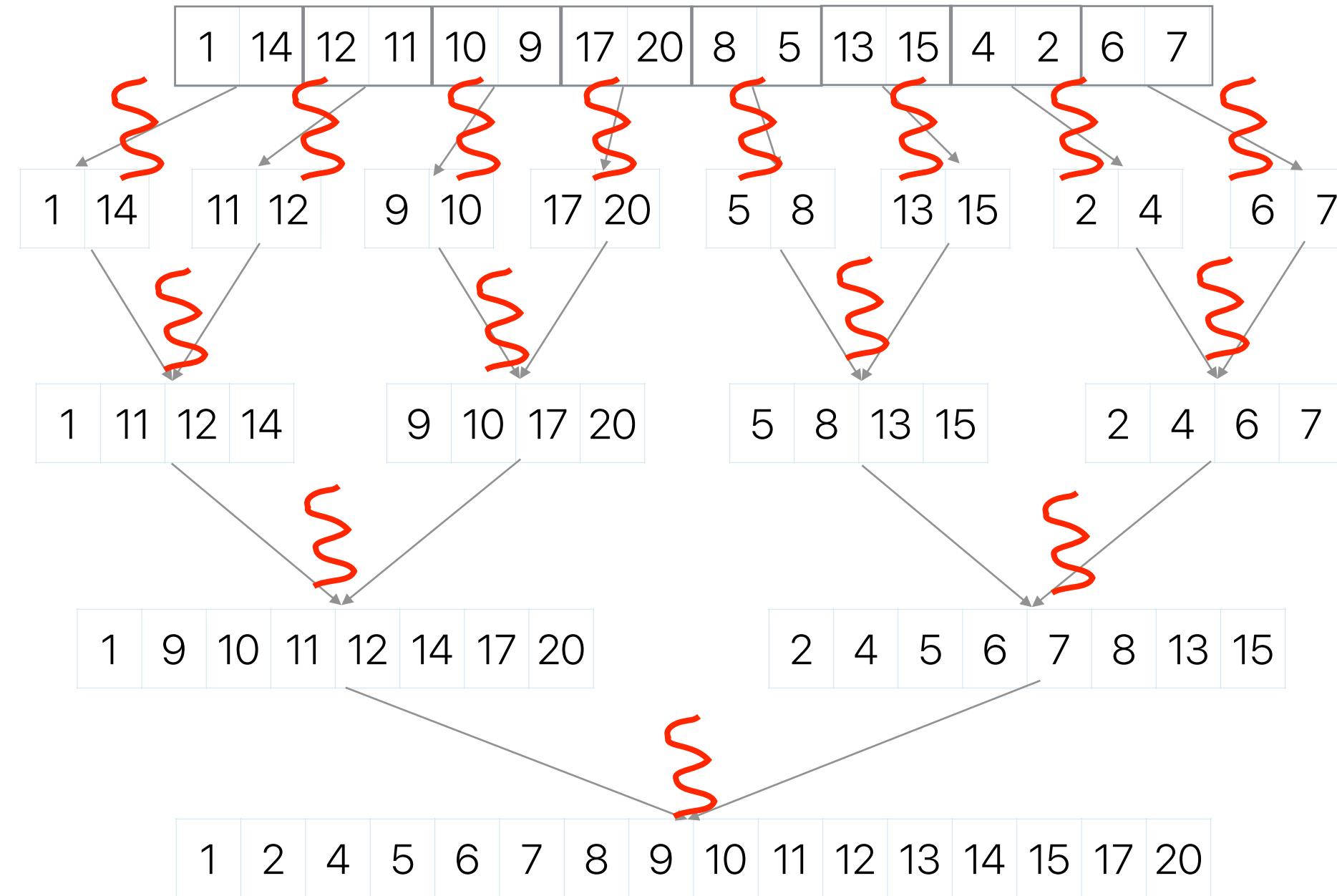
$$O(n \log_2^2 n)$$

```
void BitonicSort() {  
    int i,j,k;  
  
    for (k=2; k<=N; k=2*k) {  
        for (j=k>>1; j>0; j=j>>1) {  
            for (i=0; i<N; i++) {  
                int ij=i^j;  
                if ((ij)>i) {  
                    if ((i&k)==0 && a[i] > a[ij])  
                        exchange(i,ij);  
                    if ((i&k)!=0 && a[i] < a[ij])  
                        exchange(i,ij);  
                }  
            }  
        }  
    }  
}
```

Merge sort



Parallel merge sort



What's the speedup of merge sort using Amdahl's Law

The degree of parallelism is $1, 2, 4, \dots, \frac{n}{2}$

at step $1, 2, 3, \dots, \log_2(n)$

The ideal speedup of each step is $1, 2, 4, \dots, \frac{n}{2}$ or say $1, 2, 4, \dots, 2^{\log_2(n)-1}$ if we have **unlimited** parallelism

if we assume equal amount of time in each step in the baseline,

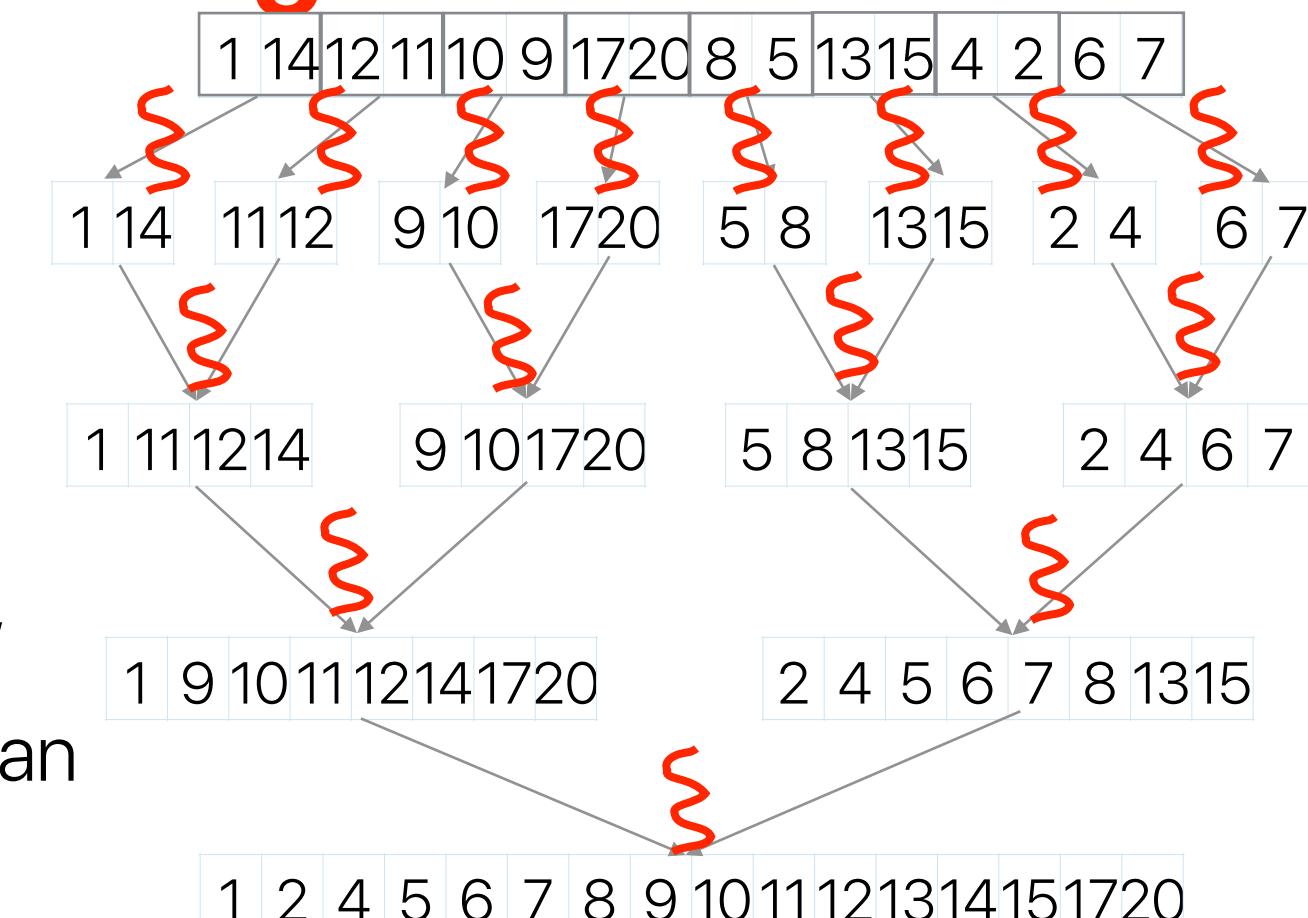
each step is going to take $\frac{1}{\lg(n)}$ portion of time in the baseline, an

the first $\frac{1}{\lg(n)}$ is not parallelizable (i.e., $(1 - x) = \frac{1}{\lg(n)}$)

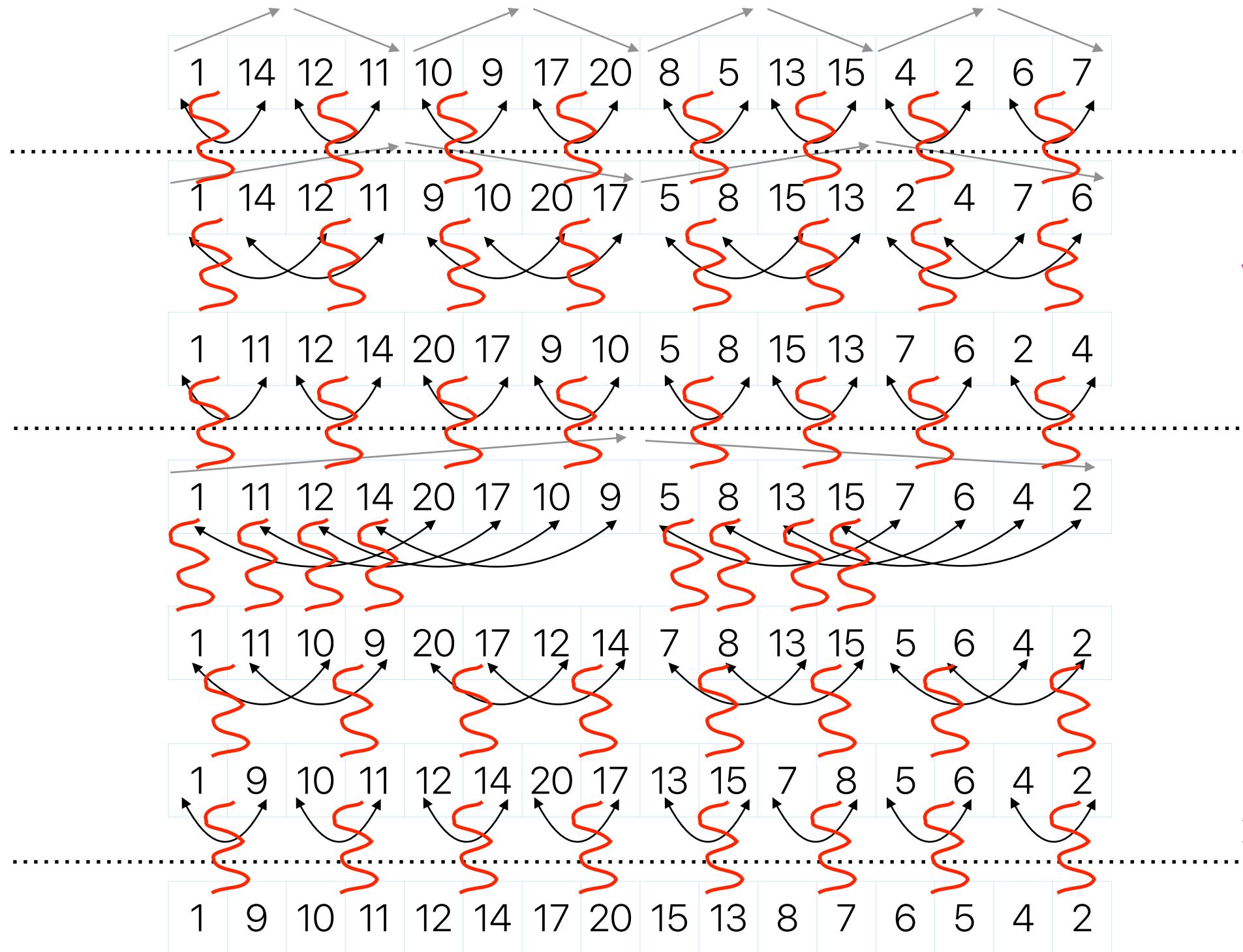
So the Amdahl's Law's evaluation will become

$$\frac{1}{\frac{1}{\lg(n)} + \frac{1}{\lg(n) \times 2} + \frac{1}{\lg(n) \times 4} + \dots + \frac{1}{\lg(n) \times 2^{\lg(n)-1}}} = \frac{1}{\frac{1}{\lg(n)}(1 + \frac{1}{2} + \frac{1}{4} + \dots + \frac{1}{2^{\lg(n)-1}})}$$

$$= \frac{1}{\frac{1}{\lg(n)}(1 + 1 - \frac{1}{2^{\lg(n)-1}})} = \frac{\frac{2}{\lg(n)}}{2} = \frac{2}{\lg(n)}$$



Bitonic sort



```
void BitonicSort() {  
    int i,j,k;  
  
    for (k=2; k<=N; k=2*k) {  
        for (j=k>>1; j>0; j=j>>1) {  
            for (i=0; i<N; i++) {  
                int ij=i^j;  
                if ((ij)>i) {  
                    if ((i&k)==0 && a[i] > a[ij])  
                        exchange(i,ij);  
                    if ((i&k)!=0 && a[i] < a[ij])  
                        exchange(i,ij);  
                }  
            }  
        }  
    }  
}
```

What's the speedup of bitonic sort using Amdahl's Law

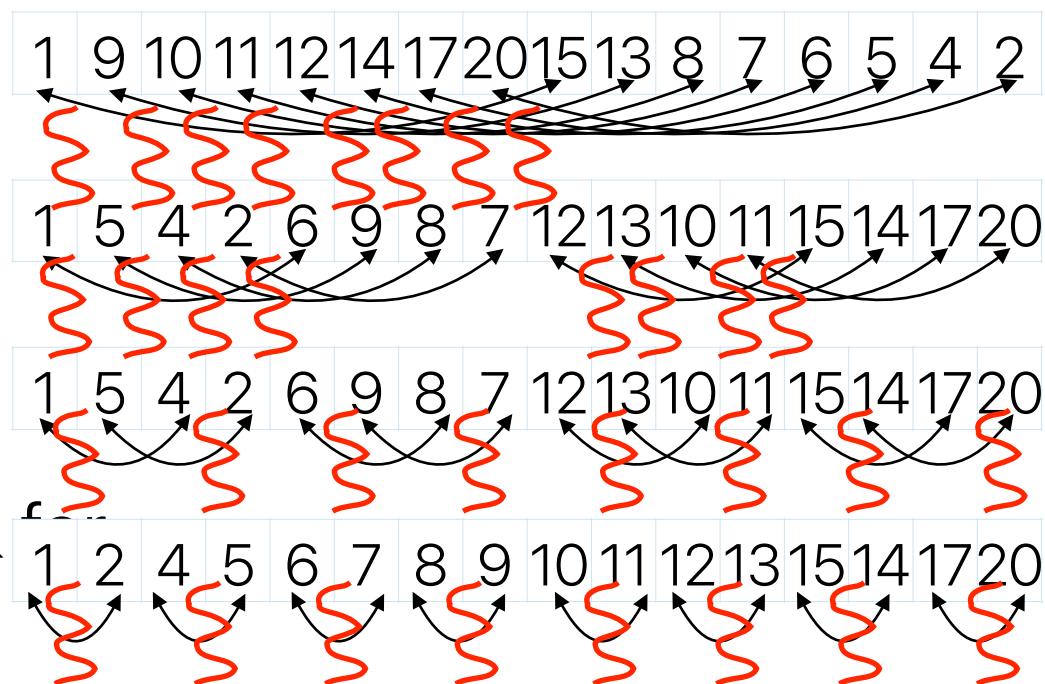
The degree of parallelism is always $\frac{n}{2}$

at step 1, 2, 3, ..., $\log_2(n)$

The ideal speedup of each step is $\frac{n}{2}$ if we have **unlimited** parallelism & each step of bitonic sort. However, bitonic sort will have $\lg(n) \times$ more steps than merge sort.

If the baseline is merge sort — the speedup of using bitonic sort is

$$\frac{\frac{1}{\frac{1 \times \lg(n)}{\frac{n}{2}}}}{\frac{1 \times \lg(n)}{2}} = \frac{n}{2 \lg(n)} > \frac{\lg(n)}{2}$$



What if we have only p processors?

For **bitonic sort**, The degree of parallelism is always $\frac{n}{2}$

at step 1, 2, 3, ..., $\log_2(n)$, but we have only p processors...

The theoretical speedup of each step is $\max(\frac{n}{2}, p) = p$ since n is very likely to be larger than p & assume equal amount of time in each step in the baseline

So the Amdahl's Law's evaluation will become $\frac{1}{\frac{p}{lg(n)}} = \frac{p}{lg(n)} = \frac{1024}{30} = 34.1333333$

What about merge sort? $= \frac{1}{\frac{1}{lg(n)}(1 + \frac{1}{2} + \frac{1}{4} + \dots + \frac{1}{p})}$

What if p=1024, n=1M=2³⁰?

$$= \frac{1}{\frac{1}{30}(1 + \frac{1}{2} + \frac{1}{4} + \dots + \frac{1}{1024} + 20 \times \frac{1}{30})} = 14.862119$$

Corollary #4

$$\text{Speedup}_{\text{parallel}}(f_{\text{parallelizable}}, \infty) = \frac{1}{(1 - f_{\text{parallelizable}}) + \frac{f_{\text{parallelizable}}}{\infty}}$$

$$\text{Speedup}_{\text{parallel}}(f_{\text{parallelizable}}, \infty) = \frac{1}{(1 - f_{\text{parallelizable}})}$$

- If we can build a processor with unlimited parallelism
 - The algorithm complexity becomes less important as long as the algorithm can utilize all parallelism
 - That's why bitonic sort or MapReduce works!
- **The future trend of software/application design is seeking for more parallelism rather than lower the computational complexity**

Takeaways: find the right thing to do

- Definition of “Speedup of Y over X” or say Y is n times faster than X:

$$speedup_{Y_over_X} = n = \frac{Execution\ Time_X}{Execution\ Time_Y}$$

- Amdahl's Law — $Speedup_{enhanced}(f, s) = \frac{1}{(1-f) + \frac{f}{s}}$
- Corollary 1 — each optimization has an upper bound
- Corollary 2 — make the common case (the most time consuming case) fast!
- Corollary 3 — Optimization has a moving target
- Corollary 4 — Exploiting more parallelism from a program is the key to performance gain in modern architectures

$$Speedup_{max}(f, \infty) = \frac{1}{(1-f)}$$

$$Speedup_{max}(f_1, \infty) = \frac{1}{(1-f_1)}$$

$$Speedup_{max}(f_2, \infty) = \frac{1}{(1-f_2)}$$

$$Speedup_{max}(f_3, \infty) = \frac{1}{(1-f_3)}$$

$$Speedup_{max}(f_4, \infty) = \frac{1}{(1-f_4)}$$

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1-f_{parallelizable})}$$

**Is it the end of computational
complexity?**

Corollary #5

$$\text{Speedup}_{\text{parallel}}(f_{\text{parallelizable}}, \infty) = \frac{1}{(1 - f_{\text{parallelizable}}) + \frac{f_{\text{parallelizable}}}{\infty}}$$

$$\text{Speedup}_{\text{parallel}}(f_{\text{parallelizable}}, \infty) = \frac{1}{(1 - f_{\text{parallelizable}})}$$

- Single-core performance still matters
 - It will eventually dominate the performance
 - If we cannot improve single-core performance further, finding more “parallelizable” parts is more important
 - Algorithm complexity still gives some “insights” regarding the growth of execution time in the same algorithm, though still not accurate

Takeaways: find the right thing to do

- Definition of “Speedup of Y over X” or say Y is n times faster than X:

$$speedup_{Y_over_X} = n = \frac{Execution\ Time_X}{Execution\ Time_Y}$$

- Amdahl's Law — $Speedup_{enhanced}(f, s) = \frac{1}{(1-f) + \frac{f}{s}}$
- Corollary 1 — each optimization has an upper bound

$$Speedup_{max}(f, \infty) = \frac{1}{(1-f)}$$

- Corollary 2 — make the common case (the most time consuming case) fast!

$$Speedup_{max}(f_1, \infty) = \frac{1}{(1-f_1)}$$

$$Speedup_{max}(f_2, \infty) = \frac{1}{(1-f_2)}$$

$$Speedup_{max}(f_3, \infty) = \frac{1}{(1-f_3)}$$

$$Speedup_{max}(f_4, \infty) = \frac{1}{(1-f_4)}$$

- Corollary 3 — Optimization has a moving target

- Corollary 4 — Exploiting more parallelism from a program is the key

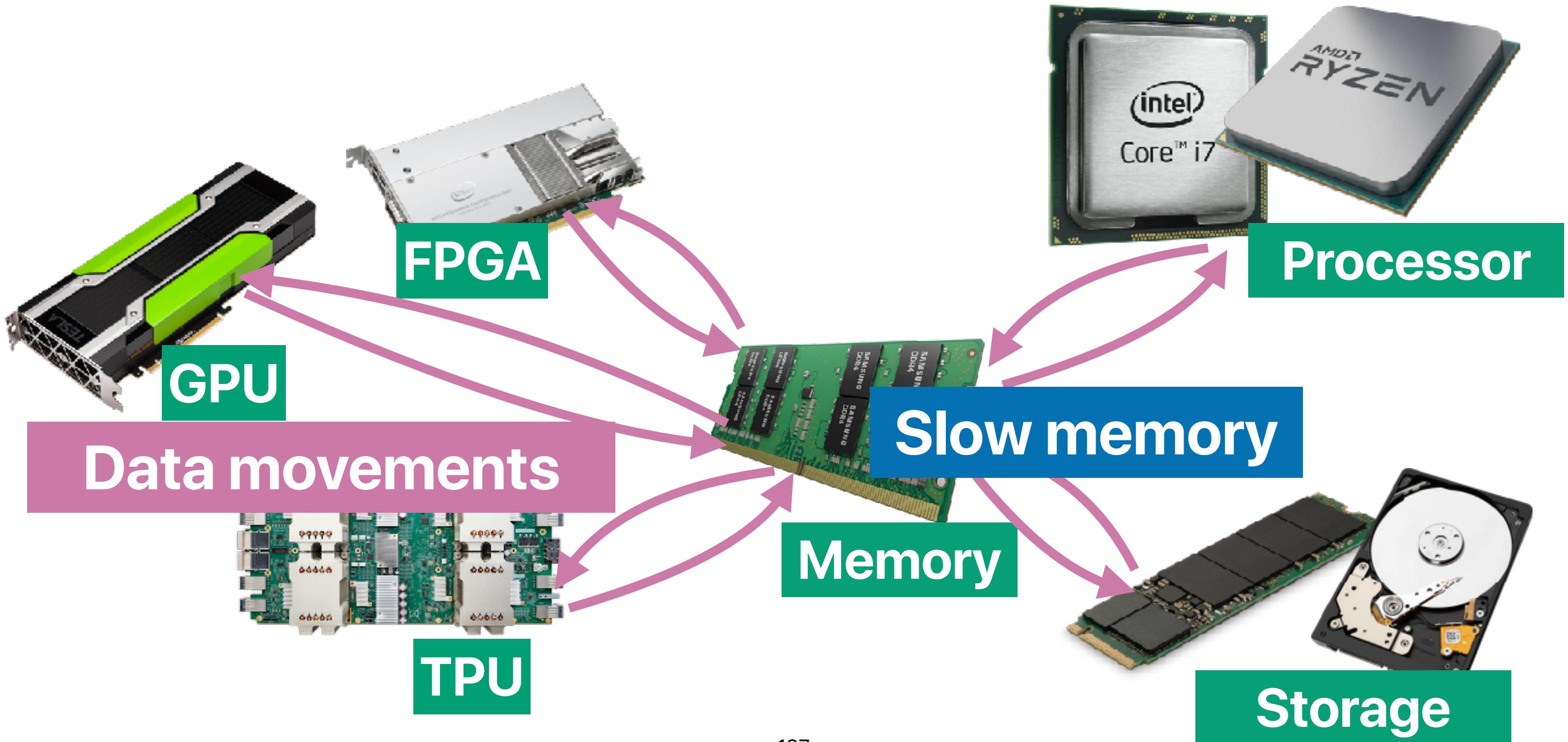
to performance gain in modern architectures

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1-f_{parallelizable})}$$

- Corollary 5 — Single-core performance still matters

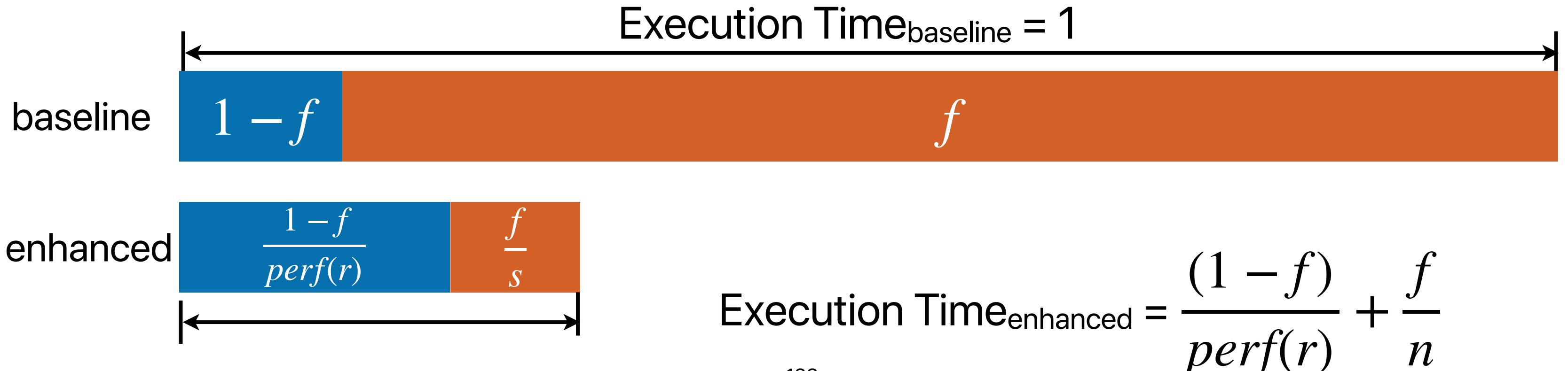
$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1-f_{parallelizable})}$$

Most cases, parallelism is not “tax-free”



What if parallelism is not “tax-free”?

- Parallelization Overhead
 - Preparing/exchanging/synchronizing data
 - Additional function calls/control overhead
- The “ $1 - f$ ” will potentially slowdown by a factor of $perf(r)$



Amdahl's Law considering overhead

$$Speedup_{enhanced}(f, s, r) = \frac{1}{\frac{(1-f)}{perf(r)} + \frac{f}{s}}$$

- r is some other parameter that affects the overhead
 - input size?
 - degree of parallelism?
- $perf(r)$ should be ≤ 1 (i.e., slowdown)
- The overhead may scale differently than the original problem
 - that's why we introduce “ $perf()$ ” function

Corollary #6: Don't hurt non-common part too much

- If the program spend 90% in A, 10% in B. Assume that an optimization can accelerate A by 9x, by hurts B by 10x (i.e., a speedup of $\frac{1}{10}$)...

$$Speedup = \frac{1}{\frac{(1-f)}{perf(r)} + \frac{f}{s}} = \frac{1}{\frac{(1-0.9)}{\frac{1}{10}} + \frac{0.9}{9}} = 0.91 \times$$

Takeaways: find the right thing to do!

- Definition of “Speedup of Y over X” or say Y is n times faster than X:

$$speedup_{Y_over_X} = n = \frac{Execution\ Time_X}{Execution\ Time_Y}$$

- Amdahl's Law — $Speedup_{enhanced}(f, s) = \frac{1}{(1-f) + \frac{f}{s}}$

- Corollary 1 — each optimization has an upper bound

$$Speedup_{max}(f, \infty) = \frac{1}{(1-f)}$$

- Corollary 2 — make the common case (the most time consuming case) fast!

$$Speedup_{max}(f_1, \infty) = \frac{1}{(1-f_1)}$$

$$Speedup_{max}(f_2, \infty) = \frac{1}{(1-f_2)}$$

$$Speedup_{max}(f_3, \infty) = \frac{1}{(1-f_3)}$$

$$Speedup_{max}(f_4, \infty) = \frac{1}{(1-f_4)}$$

- Corollary 3 — Optimization has a moving target

- Corollary 4 — Exploiting more parallelism from a program is the key to performance gain in modern architectures

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1-f_{parallelizable})}$$

- Corollary 5 — Single-core performance still matters

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1-f_{parallelizable})}$$

- Corollary 6 — Don't hurt the non-common case too much

$$Speedup_{enhanced}(f, s, r) = \frac{1}{(1-f) + perf(r) + \frac{f}{s}}$$



Extreme Multitasking Performance

- Dual 4K external monitors
- 1080p device display
- 7 applications

Is this video convincing? Why or
why not?

Outline

- 12 Ways to Fool the Masses When Giving Performance Results on Parallel Computers

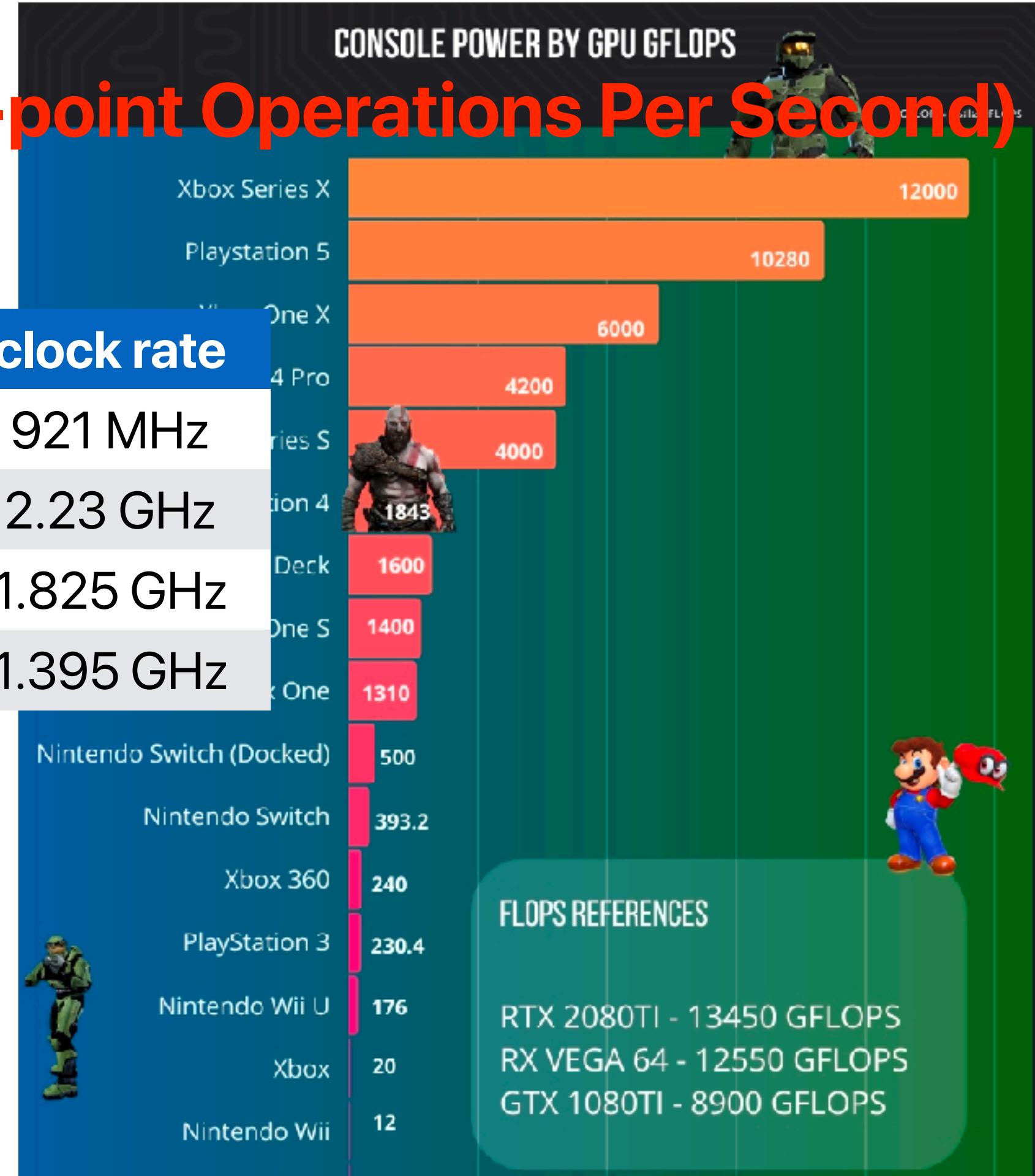
12 Ways to Fool the Masses When Giving Performance Results on Parallel Computers

- Quote only 32-bit performance results, not 64-bit results.
- Present performance figures for an inner kernel, and then represent these figures as the performance of the entire application
- Quietly employ assembly code and other low-level language constructs
- Scale up the problem size with the number of processors, but omit any mention of this fact
- Compare your results against scalar, unoptimized code on Crays
- When direct run time comparisons are required, compare with an old code on an obsolete system
- If MFLOPS rates must be quoted, base the operation count on the parallel implementation, not on the best sequential implementation
- Quote performance in terms of processor utilization, parallel speedups or MFLOPS per dollar.
- Mutilate the algorithm used in the parallel implementation to match the architecture
- Measure parallel run times on a dedicated system, but measure conventional run times in a busy environment
- If all else fails, show pretty pictures and animated videos, and don't talk about performance

Metrics on throughputs

TFLOPS (Tera FLoating-point Operations Per Second)

	TFLOPS	clock rate
Switch	1	921 MHz
PS5	10.28	2.23 GHz
XBox Series X	12	1.825 GHz
GeForce RTX 3090	40	1.395 GHz





Artificial Intelligence Computing Leadership from NVIDIA

CLOUD & DATA CENTER

PRODUCTS ▾

SOLUTIONS ▾

APPS ▾

FOR DEVELOPERS

TECHNOLOGIES ▾

Tesla V100

AI TRAINING

AI INFERENCE

HPC

DATA CENTER GPUs

SPECIFICATIONS

Deep Learning Training in Less Than a Workday



Server Config: Dual Xeon E5-2699 v4 2.6 GHz | 8X NVIDIA® Tesla® P100 or V100 | ResNet-50 Training on MXNet for 90 Epochs with 1.28M ImageNet Dataset.

AI TRAINING

From recognizing speech to training virtual personal assistants and teaching autonomous cars to drive, data scientists are taking on increasingly complex challenges with AI. Solving these kinds of problems requires training deep learning models that are exponentially growing in complexity, in a practical amount of time.

With 640 **Tensor Cores**, Tesla V100 is the world's first GPU to break the 100 teraFLOPS (TFLOPS) barrier of deep learning performance. The next generation of **NVIDIA NVLink™** connects multiple V100 GPUs at up to 300 GB/s to create the world's most powerful computing servers. AI models that would consume weeks of computing resources on previous systems can now be trained in a few days. With this dramatic reduction in training time, a whole new world of problems will now be solvable with AI.

TFLOPS (Tera FLoating-point Operations Per Second)

$$TFLOPS = \frac{\# \text{ of floating point instructions} \times 10^{-12}}{\text{Execution Time}}$$

Let's measure the TFLOPS of matrix multiplications

```
for(i = 0; i < M; i++) {  
    for(j = 0; j < N; j++) {  
        for(k = 0; k < K; k++) {  
            c[i][j] += a[i][k]*b[k][j];  
        }  
    }  
}
```

Floating point operations per second (FLOP"S"):

Floating point operations (FLOP"s"):

$$M \times N \times K \times 2$$

Given $M = N = K = 2048$

$$2^{3 \times 11} \times 2 = 2^{34} \quad \text{FLOPs in total}$$

$$TFLOPS = \frac{2^{34}}{ET_{seconds} \times 2^{12}}$$

TFLOPS (Tera FLoating-point Operations Per Second)

$$TFLOPS = \frac{\# \text{ of floating point instructions} \times 10^{-12}}{\text{Execution Time}}$$

Is TFLOPS (Tera FLoating-point Operations Per Second) a good metric?

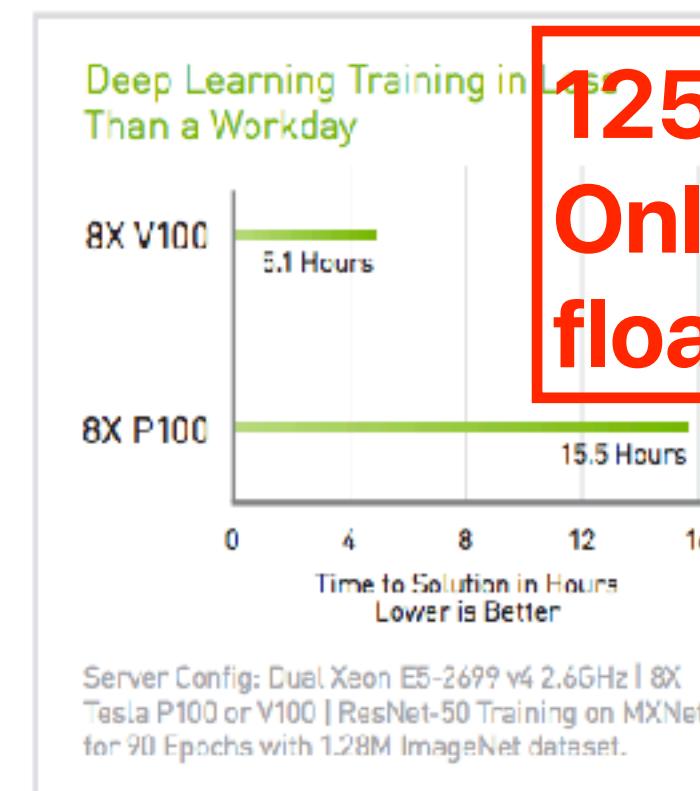
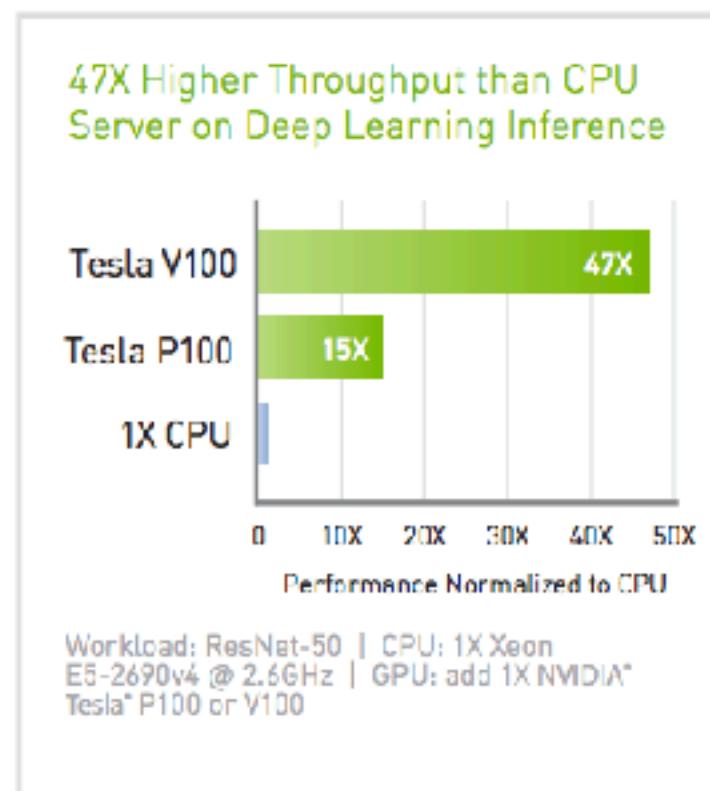
$$\begin{aligned}TFLOPS &= \frac{\# \text{ of floating point instructions} \times 10^{-12}}{\text{Execution Time}} \\&= \frac{IC \times \% \text{ of floating point instructions} \times 10^{-12}}{IC \times CPI \times CT} \\&= \frac{\% \text{ of floating point instructions} \times 10^{-12}}{CPI \times CT}\end{aligned}$$

IC is gone!

- If we have more iterations? Larger datasets? — potentially changes the IC
- What if the hardware trade (cheat) performance with accuracy?
- Cannot compare different ISA/compiler
 - What if the compiler can generate code with fewer instructions?
 - What if new architecture has more IC but also lower CPI?
- If floating point operations are not critical in the target application?

The Most Advanced Data Center GPU Ever Built.

NVIDIA® Tesla® V100 is the world's most advanced data center GPU ever built to accelerate AI, HPC, and graphics. Powered by NVIDIA Volta, the latest GPU architecture, Tesla V100 offers the performance of up to 100 CPUs in a single GPU—enabling data scientists, researchers, and engineers to tackle challenges that were once thought impossible.



**125 TFLOPS
Only @ 16-bit floating point**

SPECIFICATIONS



**Tesla V100
PCIe**



**Tesla V100
SXM2**

GPU Architecture	NVIDIA Volta	
NVIDIA Tensor Cores	640	
NVIDIA CUDA® Cores	5,120	
Double-Precision Performance	7 TFLOPS	7.8 TFLOPS
Single-Precision Performance	14 TFLOPS	15.7 TFLOPS
Tensor Performance	112 TFLOPS	125 TFLOPS
GPU Memory	32GB /16GB HBM2	
Memory Bandwidth	900GB/sec	
ECC	Yes	
Interconnect Bandwidth	32GB/sec	300GB/sec
System Interface	PCIe Gen3	NVIDIA NVLink
Form Factor	PCIe Full Height/Length	SXM2
Max Power	375W	300W

1 GPU Node Replaces Up To 54 CPU Nodes

Node Replacement: HPC Mixed Workload

AI training vs. inference

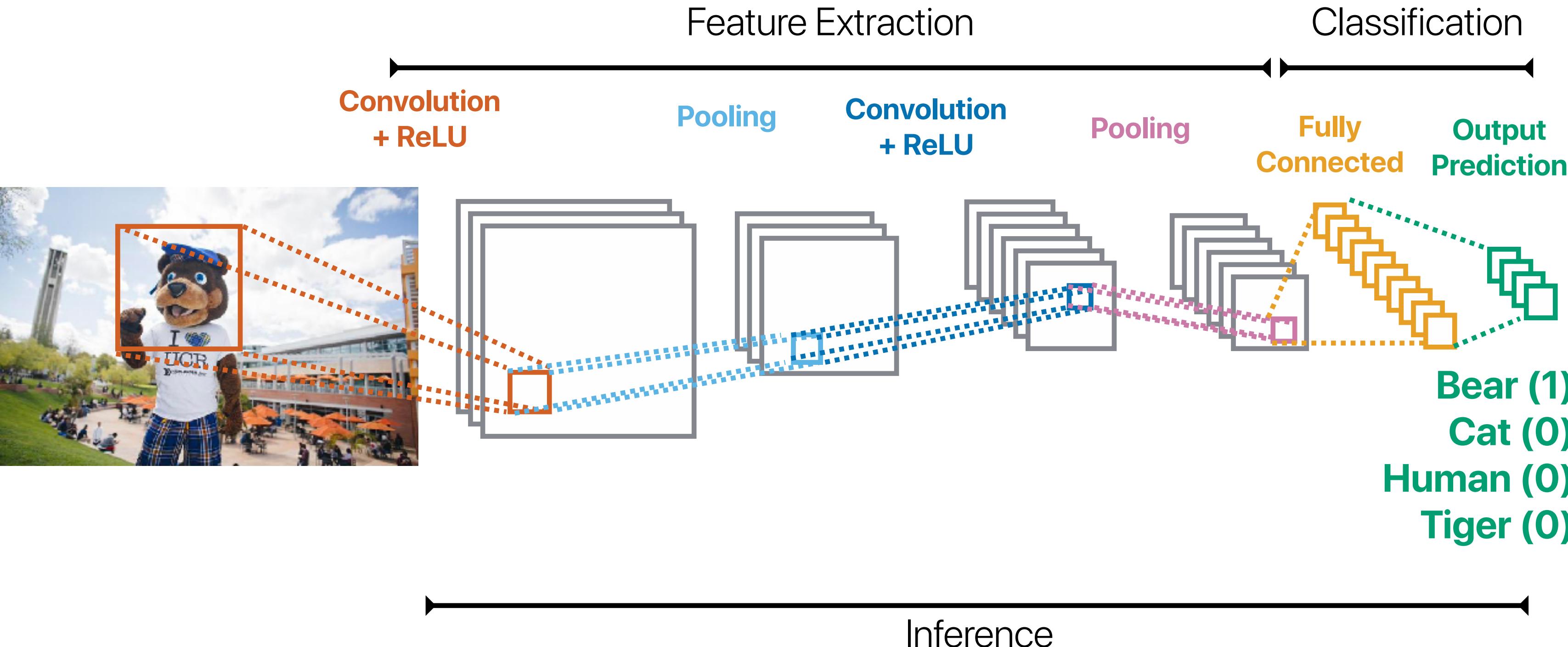
Much of the news coverage recently has been on LLMs, their development and their training – and the high cost and energy consumption required to do so. A recent study¹ estimated that Chat GPT3, comprised of 175 billion parameters, needed 1,287 MWh to train, and also emitted 552 tons of CO₂. This is roughly the equivalent to driving a car 1.3 million miles – but in one hour!

But those huge training workloads running in data centers represent just 15 percent of AI workloads today, according to Omdia's Data Center Compute Intelligence Service.²

The rest – 85 percent of all data center AI workloads – lies in AI inference, and that's not accounting for inference that's happening outside the data center on the edge. AI workloads like inference will remain a key workload in the cloud, but an increasing number will move to the edge as more efficient models continue to evolve. Inference, which is the process of using a trained model that has been deployed into a production environment to make predictions on new real-world

The rest – 85 percent of all data center AI workloads – lies in AI inference, and that's not accounting for inference that's happening outside the data center on the edge. AI workloads like inference will remain a key

The Machine Learning Inference Pipeline



Inferences per second

$$\frac{\text{Inferences}}{\text{Second}} = \frac{\text{Inferences}}{\text{Operation}} \times \frac{\text{Operations}}{\text{Second}}$$
$$= \frac{\text{Inferences}}{\text{Operation}} \times \left[\frac{\text{operations}}{\text{cycle}} \times \frac{\text{cycles}}{\text{second}} \times \#_{_of_PEs} \times \text{Utilization_of_PEs} \right]$$

What about inference per second?



chatgpt.com

ChatGPT 4.0 mini

登入

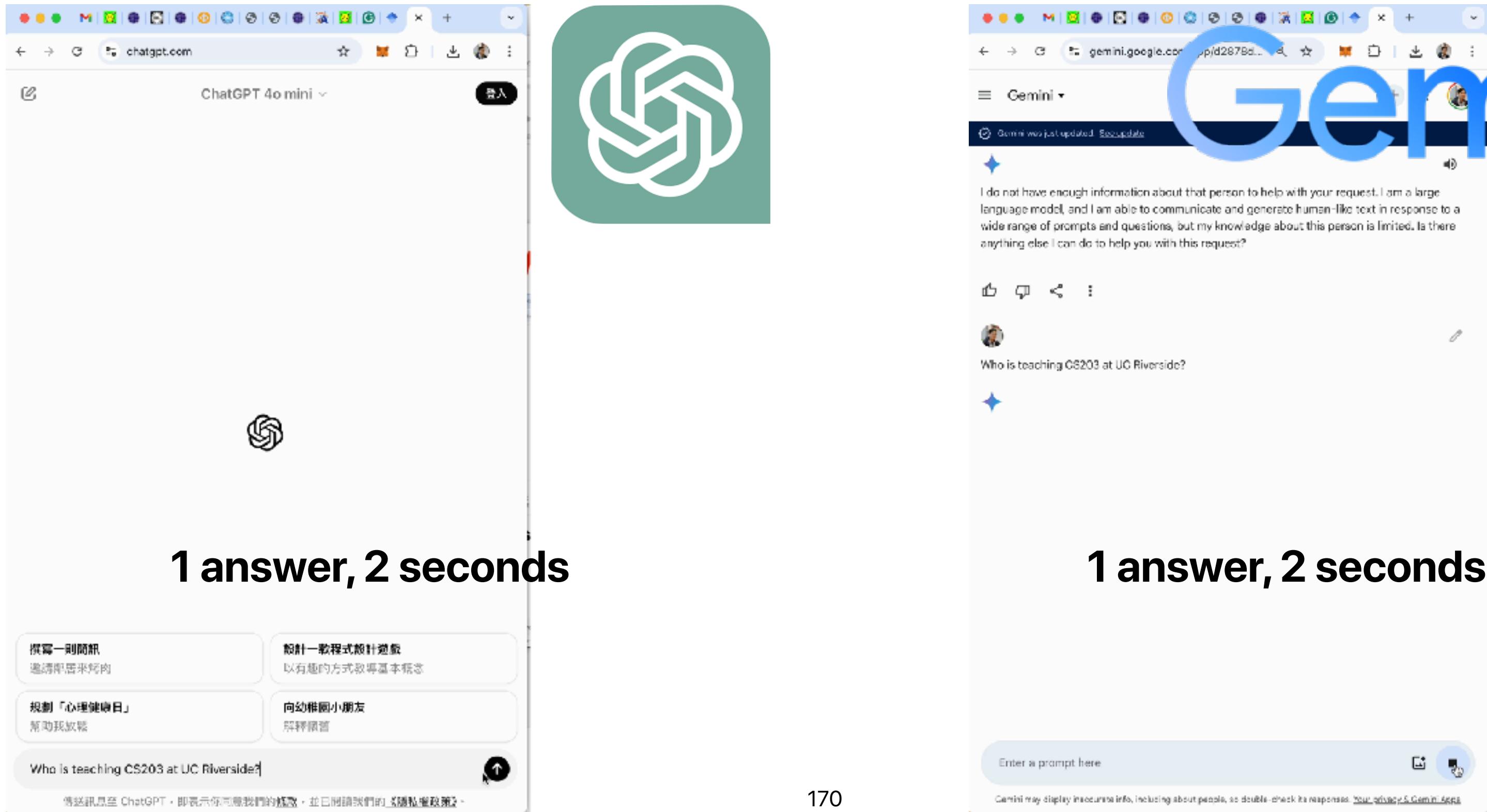
Who is teaching CS203 at UC Riverside?

Enter a prompt here

傳送訊息至 ChatGPT，即表示你同意我們的 [《隱私權政策》](#)，並已閱讀我們的 [《隱私權政策》](#)。



1 answer, 2 seconds



gemini.google.com

Gemini

Gemini was just updated. Escapable.

I do not have enough information about that person to help with your request. I am a large language model, and I am able to communicate and generate human-like text in response to a wide range of prompts and questions, but my knowledge about this person is limited. Is there anything else I can do to help you with this request?

Who is teaching CS203 at UC Riverside?

Enter a prompt here

Gemini may display inaccurate info, including about people, so double-check its responses. [Learn more >](#)

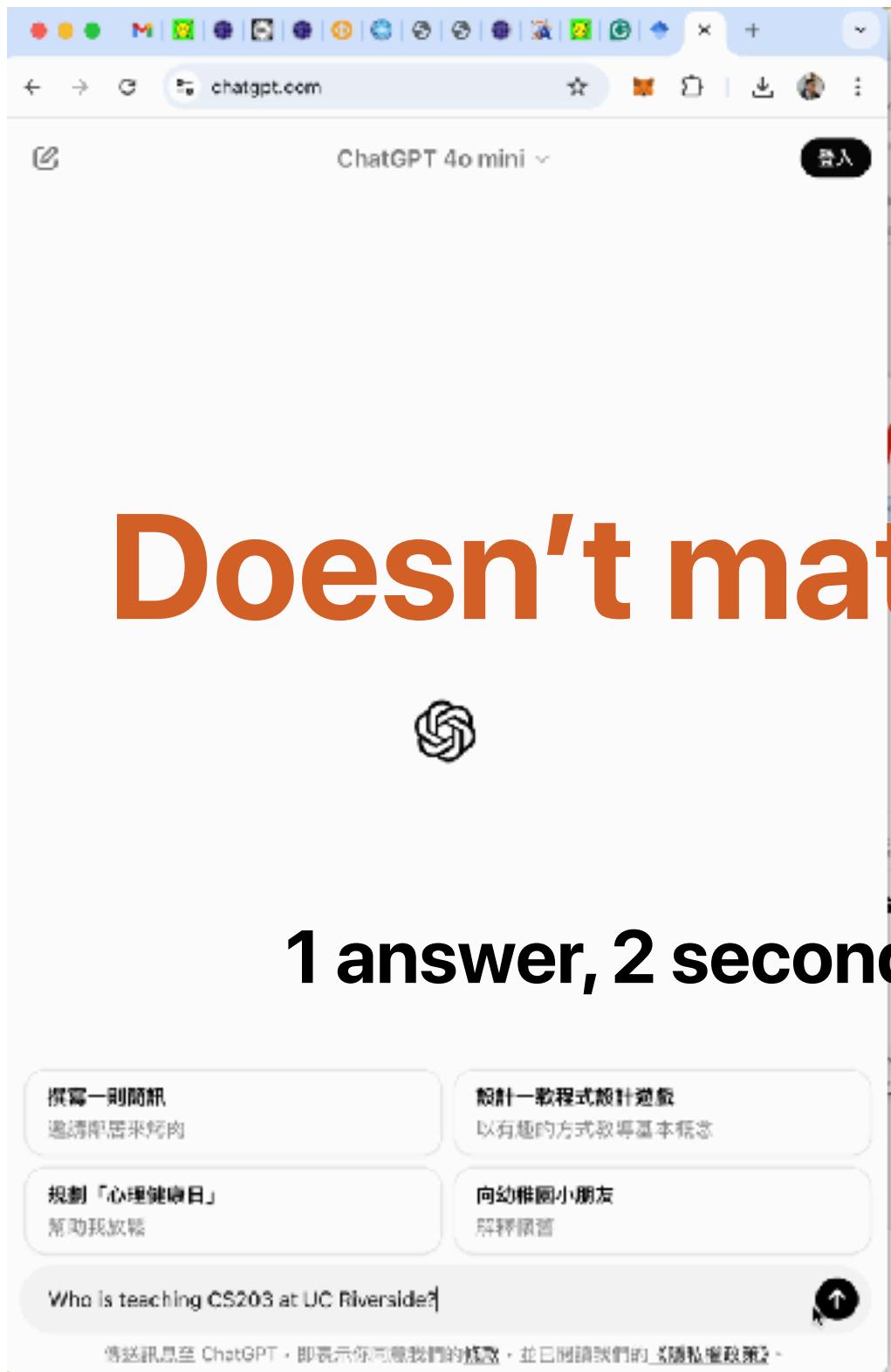
What's wrong with inferences per second?

- There is no standard on how they inference — but these affect!
 - What model?
 - What dataset?
 - Quality?
- That's why Facebook is trying to promote an AI benchmark — MLPerf

- *Pitfall: For NN hardware, Inferences Per Second (IPS) is an inaccurate summary performance metric.*

Our results show that IPS is a poor overall performance summary for NN hardware, as it's simply the inverse of the complexity of the typical inference in the application (e.g., the number, size, and type of NN layers). For example, the TPU runs the 4-layer MLP1 at 360,000 IPS but the 89-layer CNN1 at only 4,700 IPS, so TPU IPS vary by 75X! Thus, using IPS as the single-speed summary is *even more misleading* for NN accelerators than MIPS or FLOPS are for regular processors [23], so IPS should be even more disparaged. To compare NN machines better, we need a benchmark suite written at a high-level to port it to the wide variety of NN architectures. Fathom is a promising new attempt at such a benchmark suite [3].

What about inference per second?



chatgpt.com

ChatGPT 4.0 mini

登入

1 answer, 2 seconds

撰写一则简讯
邀请朋友来烤肉

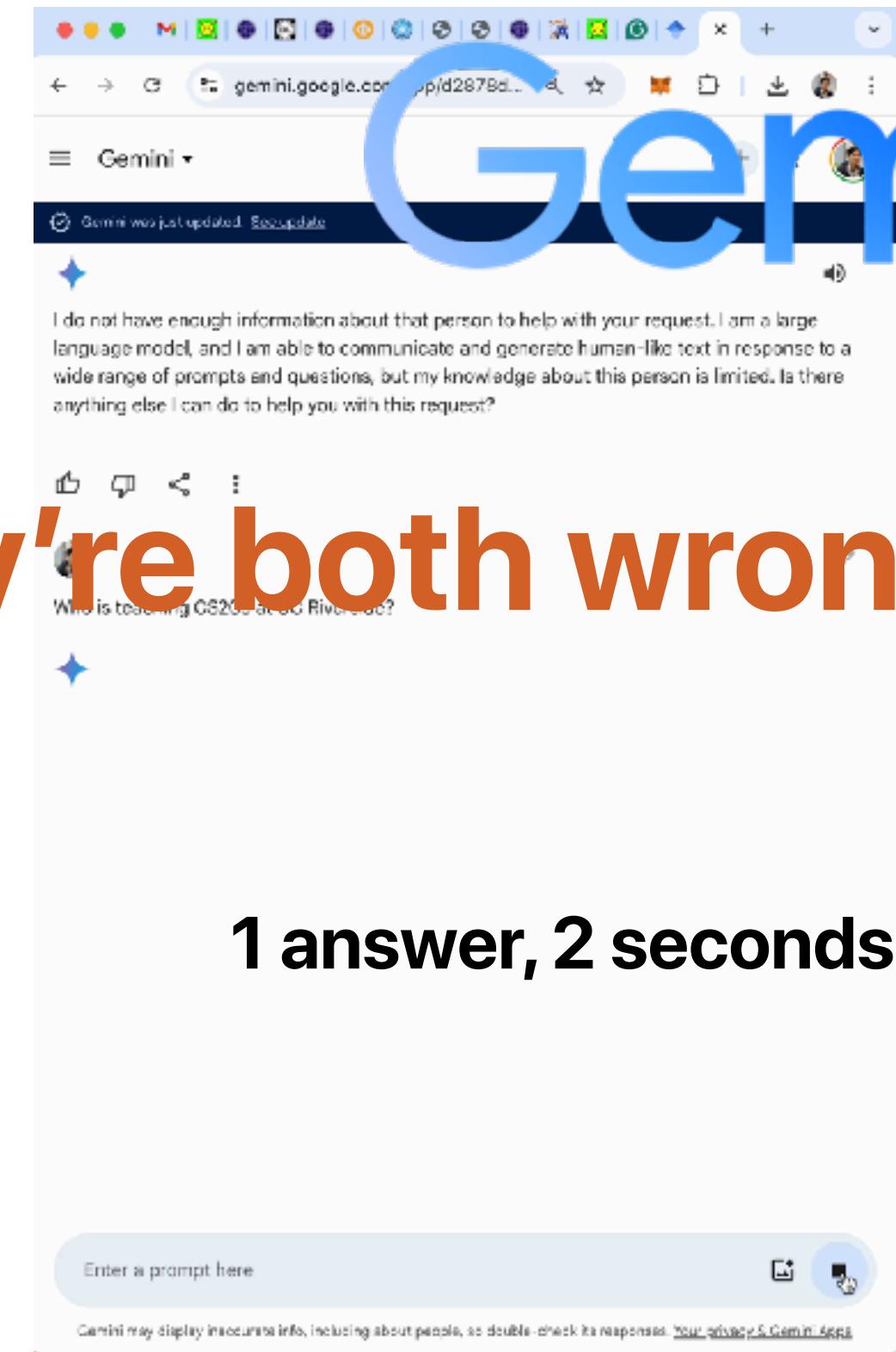
规划「心理健康日」
帮助我放松

预计一款程式设计遊戲
以有趣的方式教导基本概念

向幼稚园小朋友
解释锁首

Who is teaching CS203 at UC Riverside?

傳送訊息至 ChatGPT，即表示你同意我們的 [《隱私權政策》](#)。



gemini.google.com

Gemini

Gemini was just updated. Escapable.

I do not have enough information about that person to help with your request. I am a large language model, and I am able to communicate and generate human-like text in response to a wide range of prompts and questions, but my knowledge about this person is limited. Is there anything else I can do to help you with this request?

Who is teaching CS203 at UC Riverside?

1 answer, 2 seconds

Enter a prompt here

Gemini may display inaccurate info, including about people, so double-check its responses. [Learn more & Get help](#)

Gemini

12 Ways to Fool the Masses When Giving Performance Results on Parallel Computers

- ✓ Quote only 32-bit performance results, not 64-bit results.
 - Present performance figures for an inner kernel, and then represent these figures as the performance of the entire application
 - Quietly employ assembly code and other low-level language constructs
- ✓ Scale up the problem size with the number of processors, but omit any mention of this fact
 - Compare your results against scalar, unoptimized code on Crays
 - When direct run time comparisons are required, compare with an old code on an obsolete system
 - ✓ If MFLOPS rates must be quoted, base the operation count on the parallel implementation, not on the best sequential implementation
- ✓ Quote performance in terms of processor utilization, parallel speedups or MFLOPS per dollar.
- ✓ Mutilate the algorithm used in the parallel implementation to match the architecture
 - Measure parallel run times on a dedicated system, but measure conventional run times in a busy environment
- ✓ If all else fails, show pretty pictures and animated videos, and don't talk about performance

Nvidia accused of cheating in 3DMar

Futuremark, the maker of 3DMark 03, releases a patch for the game to correct the way Nvidia drivers manipulate results in the popular benchmark.

By **David Becker** on May 23, 2003 at 4:27PM PDT

Saratoga, California-based Futuremark on Friday said in a statement that Nvidia tweaked the software needed to run its new GeForce FX 5900 processor to distort performance in Futuremark's 3DMark 03 testing application. The company said drivers for the new Nvidia chip were altered to detect activity characteristic of a benchmark and adjust performance accordingly.

Recently, there have been questions and some confusion regarding 3DMark 03 results obtained with certain Nvidia products, Futuremark said in the statement. **"We have now established that Nvidia's Detonator FX drivers contain certain detection mechanisms that cause an artificially high score when using 3DMark 03,"** the statement read.

Futuremark has released the version 330 patch for 3DMark 03, which prevents the Nvidia drivers from identifying the benchmark. By Futuremark's measure, the performance of the Nvidia GeForce FX 5900 Ultra drops 24 percent with the patch, compared with a drop of less than 2 percent with ATI's Radeon 9800 Pro with the latest ATI drivers.

A representative at Nvidia questioned the validity of Futuremark's conclusions. "Since Nvidia is not part of the Futuremark beta program (a program which costs of hundreds of thousands of dollars to participate in), we do not get a chance to work with Futuremark on writing the shaders like we would with a real applications developer," the representative said. "We don't know what they did, but it looks like they have intentionally tried to create a scenario that makes our products look bad."

<https://www.gamespot.com/articles/nvidia-accused-of-cheating-in-3dmark-03/1100-6028894/>

#:~:text=%22We%20have%20now%20established%20that,drivers%20from%20identifying%20the%20benchmark.

Nvidia is amid a hard-fought battle with rival ATI Technologies to claim the performance lead in PC graphics processors. After years of Nvidia dominating both in market share and performance, ATI

Fair comparison in computer architectures

- Metrics: you must consider the fact that performance is composed of IC, CPI, and CT. — any metric that misses one of them is misleading
- Only one variation in each comparison
 - Only change the processor, but not ISA (related to IC) and others
 - Only change the algorithm, but not others
 - The same dataset, must be the same outcome

The reason of “Benchmark Suites”

- Allowing people evaluate systems with exactly the same program and the same inputs and validate results from different machines
- Popular benchmark suites
 - SPEC — CPU benchmark
 - MLPerf — ML systems

The screenshot shows the homepage of the SPEC Standard Performance Evaluation Corporation. The header features the SPEC logo and navigation links for Home, Benchmarks, Tools, Results, Contact, Blog, Join Us, Search, and Help. A sidebar on the left includes sections for Results (Published Results, Results Search, Fair Use Policy), Information (CPU2017, Documentation Overview, System Requirements, Run & Reporting Rules, Using SPEC CPU2017, Resources, Technical Support, Support, FAQ), and Press & Publications. The main content area highlights the SPEC CPU® 2017 benchmark, describing it as a next-generation, industry-standardized, CPU intensive suite for measuring and comparing compute intensive performance. It mentions a price of \$1000 for new customers, \$250 for qualified non-profit organizations, and \$50 for accredited academic institutions, with contact information at info@spec.org.

The screenshot shows the NVIDIA Cloud & Data Center website. The header includes the NVIDIA logo and navigation links for Cloud & Data Center, Solutions, Products, and Data Center GPUs. The main content area features a section titled "MLPerf Benchmarks" with text about the NVIDIA AI platform showcasing leading performance and versatility in MLPerf Training, Inference, and HPC for demanding real-world AI workloads. A "See Our Results" button is visible at the bottom.

Latency v.s. Bandwidth/Throughput

- Latency — the amount of time to finish an operation
 - End-to-end execution time of “something”
 - Access time
 - Response time
- Throughput — the amount of work can be done within a given period of time (typically “something” per “timeframe” or the other way around)
 - Bandwidth (MB/Sec, GB/Sec, Mbps, Gbps)
 - IOPs (I/O operations per second)
 - FLOPs (Floating-point operations per second)
 - IPS (Inferences per second)
 - FPS (Frames per second)

Demo: matmul on GPU

Size	Latency	Relative Latency	Throughput (Output Numbers Per Second)	Relative Throughput
16x16x16	~ 0.09ms	1	0.09ms/256	1
32x32x32	~ 0.09ms	1	0.09ms/1024	4
64x64x64	~ 0.09ms	1	0.09ms/4096	16

Larger throughput doesn't mean shorter latency!

Takeaways: What does “perfect” mean?

- Latency is the most fundamental performance metric
- Classic CPU performance equation — Instruction count (IC), cycles per instruction (CPI), cycle time (CT) define the latency of execution on CPUs
- Performance metrics without considering all three factors in the classic performance equation can mislead — anything throughput typically miss one of them

Announcement

- Programming assignment 1 **due next Thursday** & Assignment 1 **released**
 - We cannot help you at the last minute — please start early
 - Watch before you start https://youtu.be/m7OoY8y_lsk
 - Please always make sure you follow the exact steps in the readme and the notebook
 - Submit to the right item on Gradescope
 - Please visit an office hour if you need more assistance
- Reading quiz due next Tuesday before the lecture — we will drop two of your least performing reading quizzes
- Check our website for slides, Gradescope for assignments, discord for discussions
- Check your grades at https://www.escalab.org/my_grades
 - If you don't have any grade, you need to make sure your gradescope account is associated with your UCRNetID@ucr.edu
 - You have to submit a course agreement to receive scores
- Youtube channel for lecture recordings:
<https://www.youtube.com/c/ProfUsagi/playlists>

Computer Science & Engineering

203

つづく

