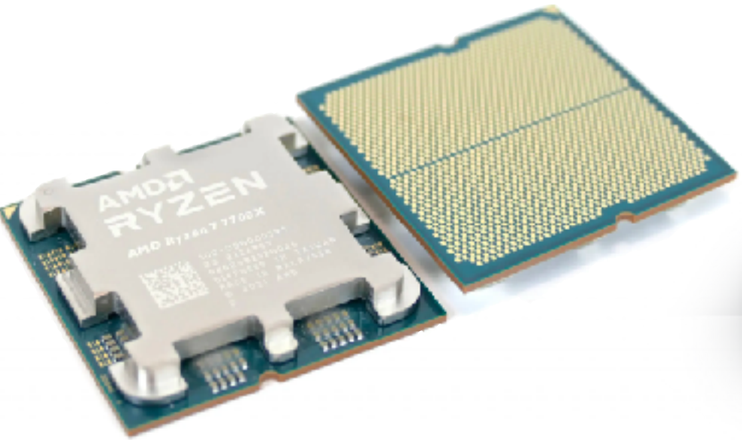
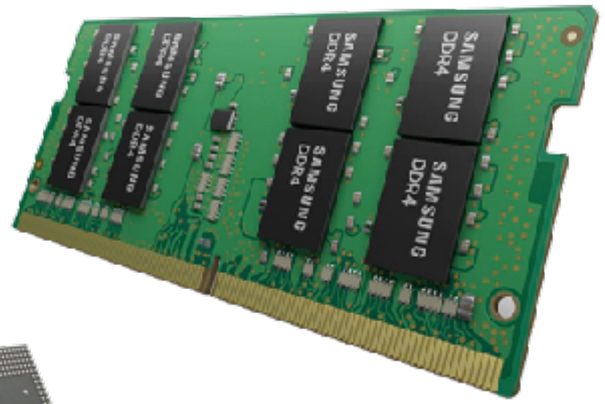
Performance (1): What does "perfect" mean?

Hung-Wei Tseng

Recap: Processors and memory modules are everywhere!

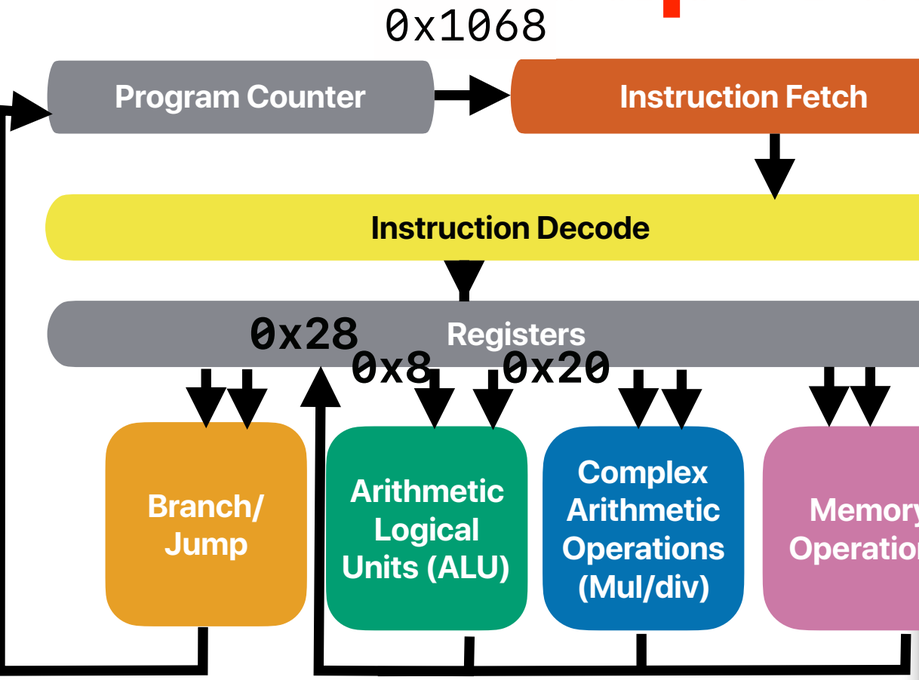
Processors





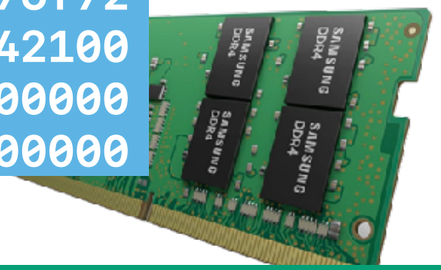
Memory

Recap: von Neumann architecture



Processor

Instructions	Data
f30f1efa	08400000
4883ec08	00000100
488d3d95	02004865
0f0000e8	6c6c6f2c
dcffffff	20776f72
31c04883	6c642100
c408c30f	00000000
1f440000	00000000



Memory

```
int main(){
    pr
}
```

By loading different programs into memory, your computer can perform different functions

Instructions	Data
f30f1efa	08400000
4883ec08	00000100
488d3d95	02004865
0f0000e8	6c6c6f2c
dcffffff	20776f72
31c04883	6c642100
c408c30f	00000000
1f440000	00000000

Storage

Recap: Demo

```
if(option)
    std::sort(data, data + arraySize);

for (unsigned c = 0; c < arraySize*1000; ++c) {
    int t = std::rand();

    (data[c%arraySize] >= t)
            sum ++;
    }
}
```

$O(nlog_2n)$

$O(n)$

if option is set to 1: $O(nlog_2n)$ **— 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];
  }
}
```

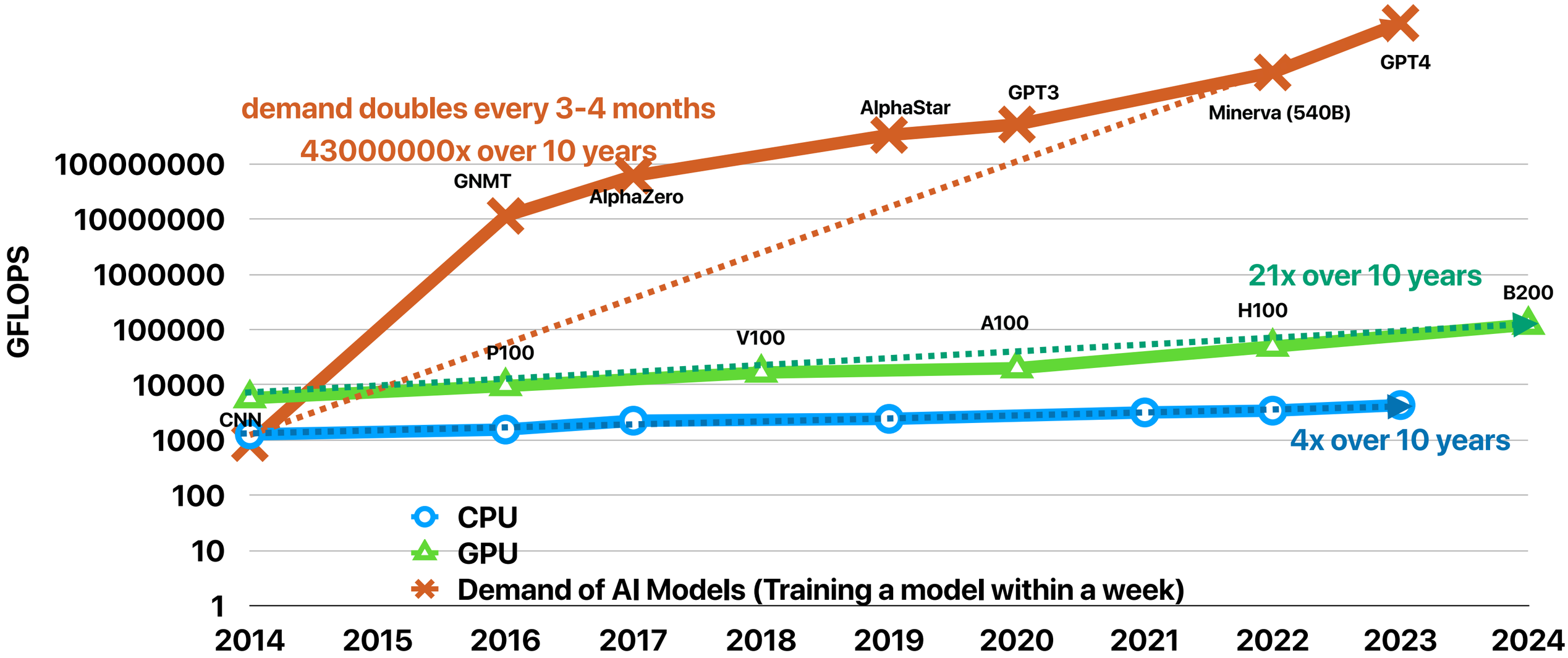
B

```
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** $O(n^2)$

A Lot Better! **Performance?** **Worse**

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



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

puter a

ChatGPT

chat.openai.com/c/640...

ChatGPT 3.5

threats, knowledge of security principles and cr
crucial for protecting data and systems from un
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.

You

What is computer architecture?

ChatGPT

Message ChatGPT...

ChatGPT can make mistakes. Consider checking important information.

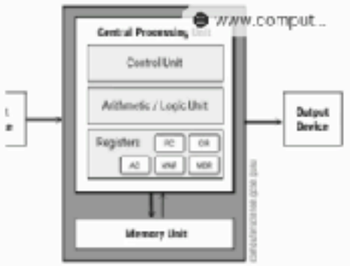
Gemini

gemini.google.com...

Gemini

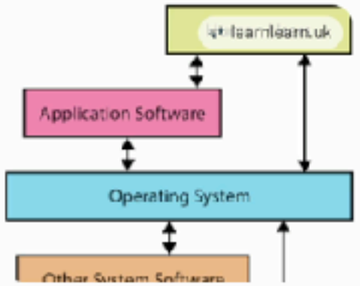
See the latest updates to the Gemini Privacy Hub

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?

Submit

Gemini may display inaccurate info, including about people, so double-check its responses. Your privacy & Gemini Apps

3:00

Which side do you feel the performance is better?

Best Nation

Schools in the National Univers
a full range of undergraduate r
producing groundbreaking res

To unlock full rankings, SAT/A

SUMMARY

443

Scho

Loca

Rank

Nat

Best Compute

Best Computer Science Schools

Ranked in 2022, part of Best Science S

Earning a graduate degree in computer
technology companies and colleges ar
reflects its average rating on a scale fr
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

https://www.pollev.com/hungweitseng close in 1:30

Performance equation

- Consider the following c code snippet and x86 instructions implement the code snippet
  If (1) `data_size` is set to 1,000,000,000,
  (2) a memory instruction takes 5 cycles,
  (3) a branch/jump instruction takes 2 cycles,
  (4) other instructions takes 1 cycle on average,
  and (5) the processor runs at 4 GHz,
  how much time is it take to finish executing the code snippet?
  A. 0.5 sec
  B. 1 sec
  C. 2.5 sec
  D. 3.75 sec
  E. 4 sec

C

```
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;
}
```

x86

```
.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
```

Performance equation

- Consider the following c code snippet and x86 instructions implement the code snippet
  If (1) `data_size` is set to 1,000,000,000,
  (2) a memory instruction takes 5 cycles,
  (3) a branch/jump instruction takes 2 cycles,
  (4) other instructions takes 1 cycle on average,
  and (5) the processor runs at 4 GHz,
  how much time is it take to finish executing the code snippet?
  A. 0.5 sec
  B. 1 sec
  C. 2.5 sec
  D. 3.75 sec
  E. 4 sec

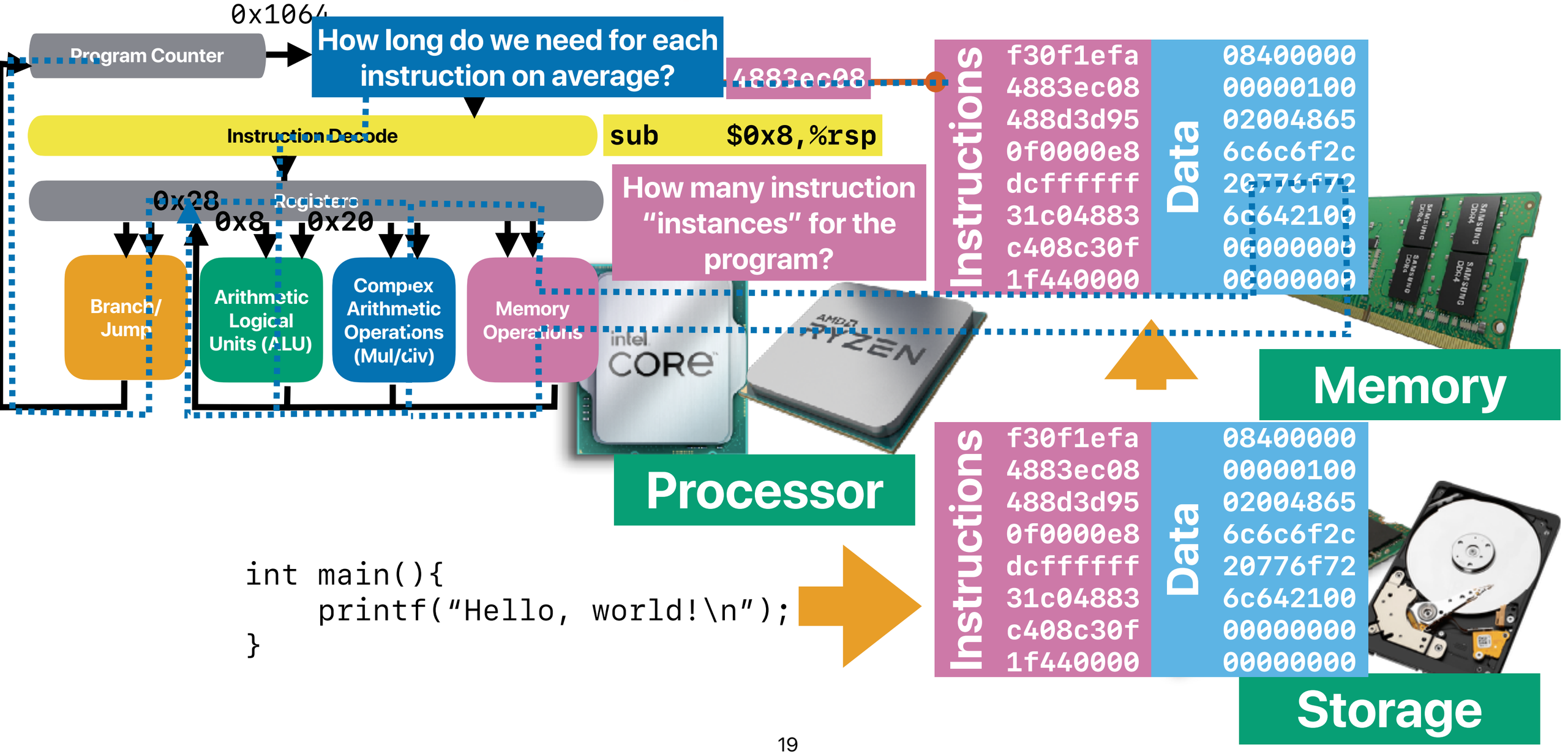
C

```c
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;
}
```

x86

```
.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
```

Execution time of a program in the von Neumann model




```
int main(){
    printf("Hello, world!\n");
}
```

CPU Performance Equation

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

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

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

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

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

Classic CPU Performance Equation (ET of a program)

How many instruction "instances" for the program? × How long do we need for each instruction on average?

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

C Code	x86 instructions

```
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;
}

int main(int argc, char **argv) {
    int *data = malloc(8000000000);
    init_data(data, 1000000000)
    return 0;
}
```

1000000000x

```
init_data:
.LFB16:
    endbr64
    testl    %esi, %esi
    jle  .L2
    leal     -1(%rsi), %ecx
    xorq     %rax, %rax
.L3:
    movslq   (%rdi), %rdx   ←memory inst.
    addq     $4, %rdi
    addq     %rdx, %rax
    cmpq     %rcx, %rdi
    jne      .L3            ←branch inst.
.L2:
    xorlq    %rax, %rax
    ret
```

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

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

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

Performance equation

- Consider the following c code snippet and x86 instructions implement the code snippet

C	x86

```
for(i = 0; i < count; i++) {
    s += a[i];
}
```

```
.L3:
movslq  (%rdi), %rdx
addq    $4, %rdi
addq    %rdx, %tax
cmpq    %rcx, %rdi
jne     .L3
```

If (1) count is set to 1,000,000,000, (2) a memory instruction takes 4 cycles, (3) a branch/jump instruction takes 3 cycles, (4) other instructions takes 1 cycle on average, and (5) the processor runs at 4 GHz, how much time is it take to finish executing the code snippet?

A. 0.5 sec
B. 1 sec
C. 2.5 sec
D. 3.75 sec
E. 4 sec

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

$$ET = (5 \times 10^9) \times (20\% \times 5 + 20\% \times 2 + 60\% \times 1) \times \frac{1}{4 \times 10^9}\ sec = 2.5\ sec$$

total # of dynamic instructions

average CPI

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

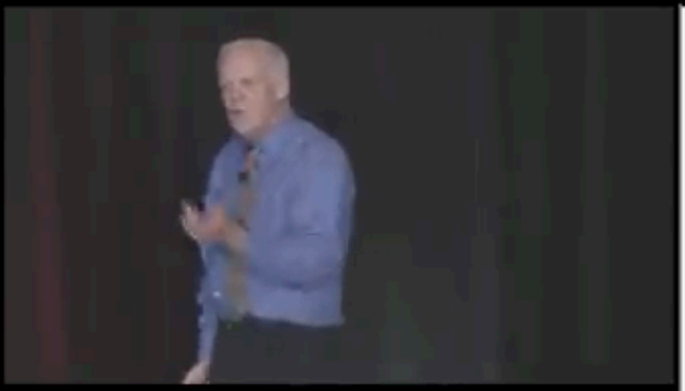
https://www.pollev.com/hungweitseng close in 1:30

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. Rust
  E. Python

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 & performance

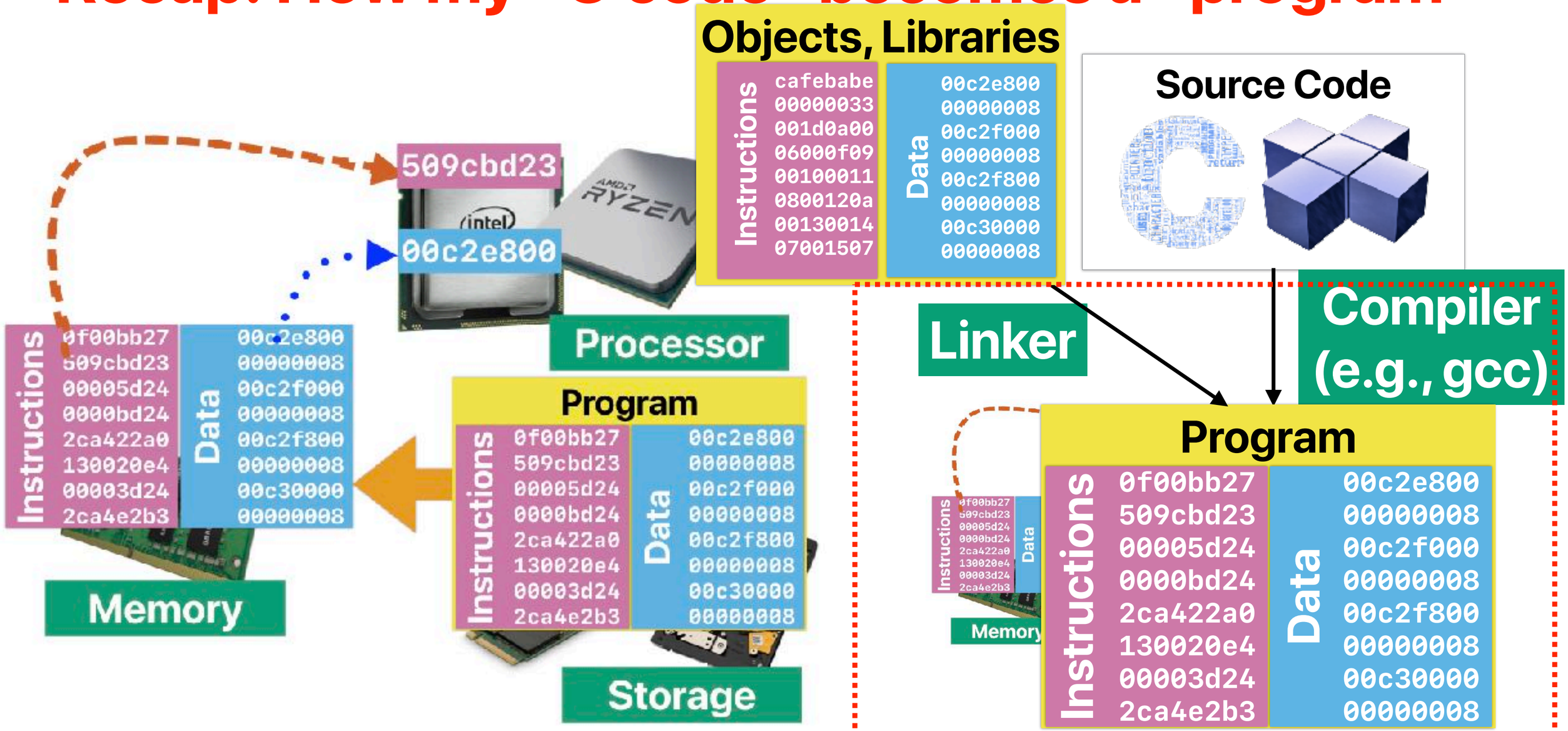
- How many instructions are there in “Hello, world!”

	Instruction count	LOC	Ranking	CPI
C	~1M	6	1	1.04
C++	~3.7M	6	3	0.76
Java	~154M	8	5	0.66
Python	~52M	1	4	0.47
GO (Interpreter)	~1400M	1	6	0.59
GO (Compiled)	~3M	1		1.33
Rust	~1.6M	1	2	1.15

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. Rust
  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 01000000 02000000
 0030 03000000 04000000 05000000 06000000
 0040 07000000 08000000 09000000 0a000000
```

Data

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

control flow operations

logical operations

memory access

arithmetic operations

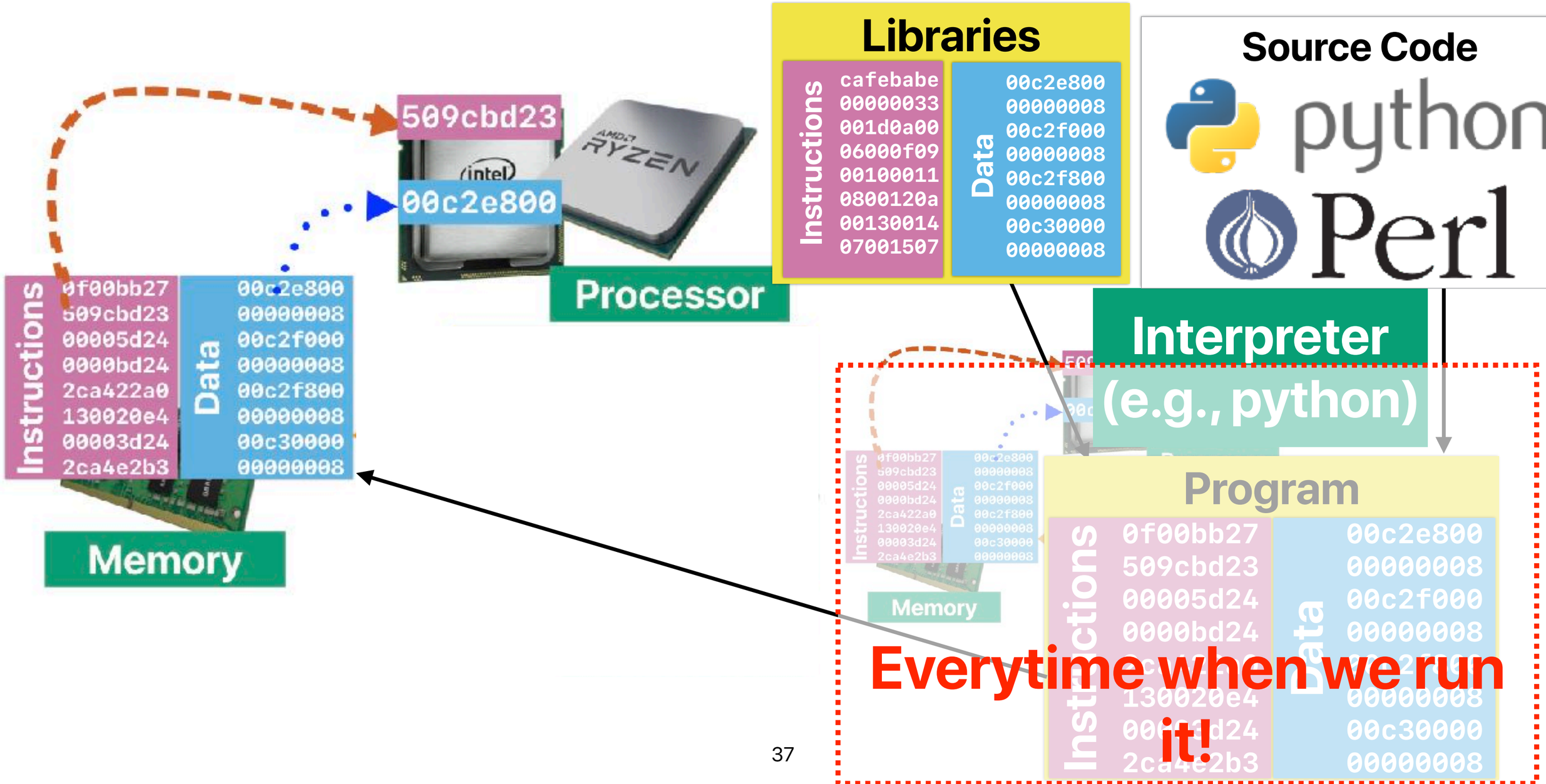
```
main:
.LFB0:
    endbr64
    pushq   %rbp
    movq %rsp, %rbp
    movl $0, -8(%rbp)
    movl $0, -4(%rbp)
    movl $0, -8(%rbp)
    jmp  .L2
.L3:
    movl -8(%rbp), %eax
    cltq
    leaq 0(,%rax,4), %rdx
    leaq A(%rip), %rax
    movl (%rdx,%rax),
%eax
    addl %eax, -4(%rbp)
    addl $1, -8(%rbp)
.L2:
    cmpl $19, -8(%rbp)
    jle  .L3
    movl $0, %eax
    popq %rbp
    ret
```

Compiler

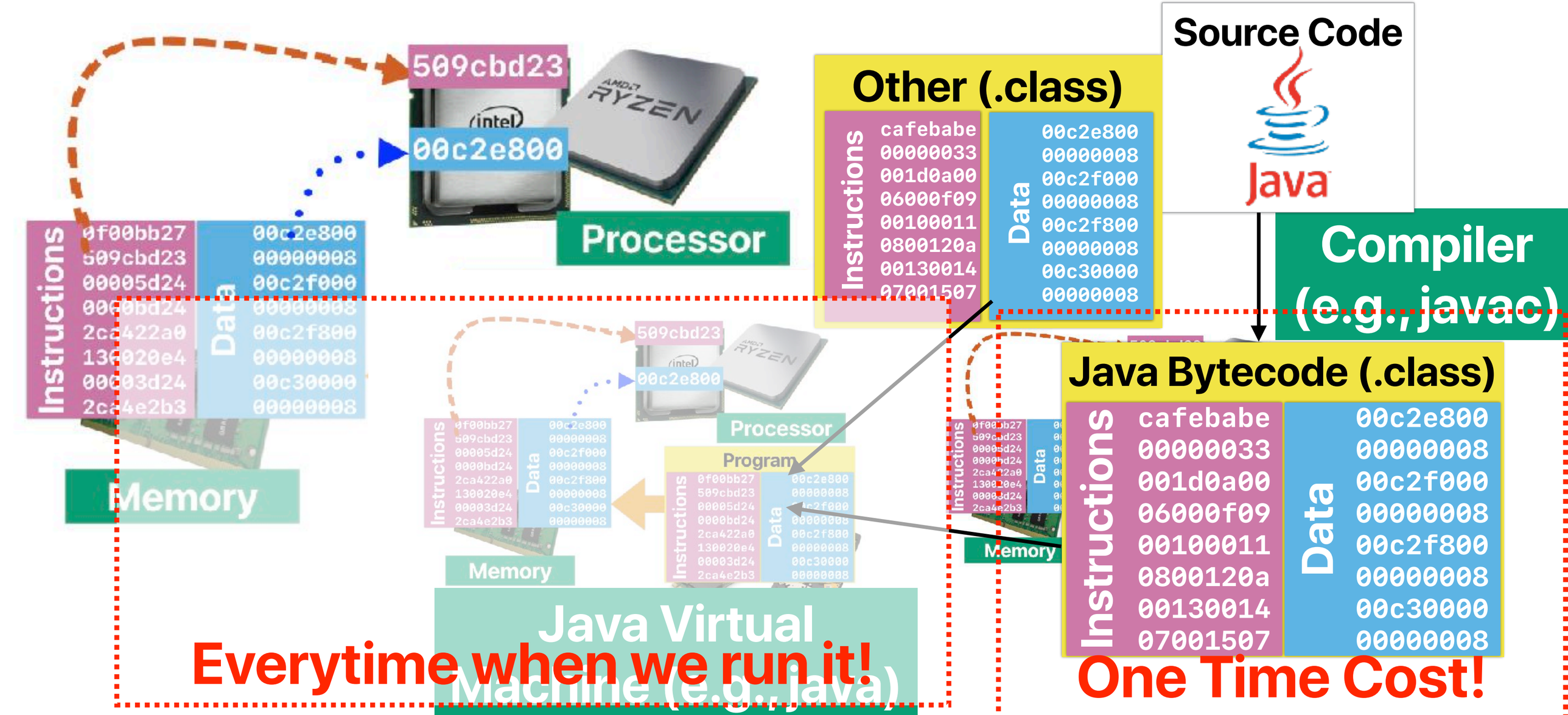
```
Contents of section .text:
 0000 f30f1efa 554889e5 c745f800 000000c7
 0010 45fc0000 0000c745 f8000000 00eb1e8b
 0020 45f84898 488d1425 00000000 488d0500
 0030 0000008b 04020145 fc8345f8 01837df8
 0040 137edcb8 00000000 5dc3
```

Instructions

Recap: How my "Python code" becomes a "program"

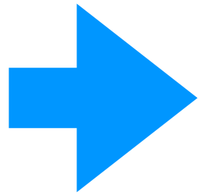


Recap: How my "Java code" becomes a "program"



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);
}
```

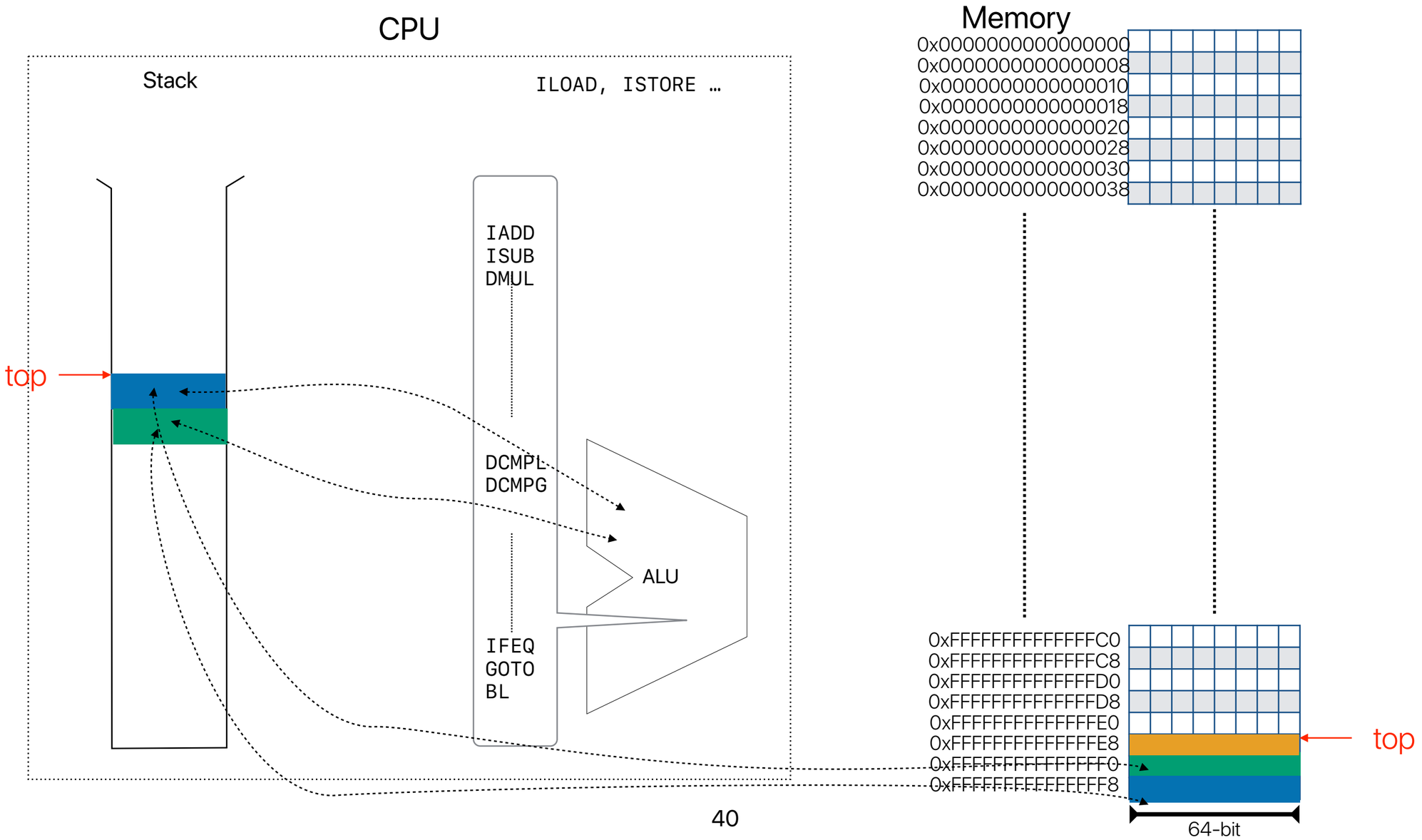


```
 0: iload_0
 1: ifne          6
 4: iconst_0
 5: ireturn
 6: iload_0
 7: iconst_1
 8: if_icmpne     13
11: iconst_1
12: ireturn
13: iload_0
14: iconst_1
15: isub
16: invokestatic  #2
19: iload_0
20: iconst_2
21: isub
22: invokestatic  #2
25: iadd
26: ireturn
```

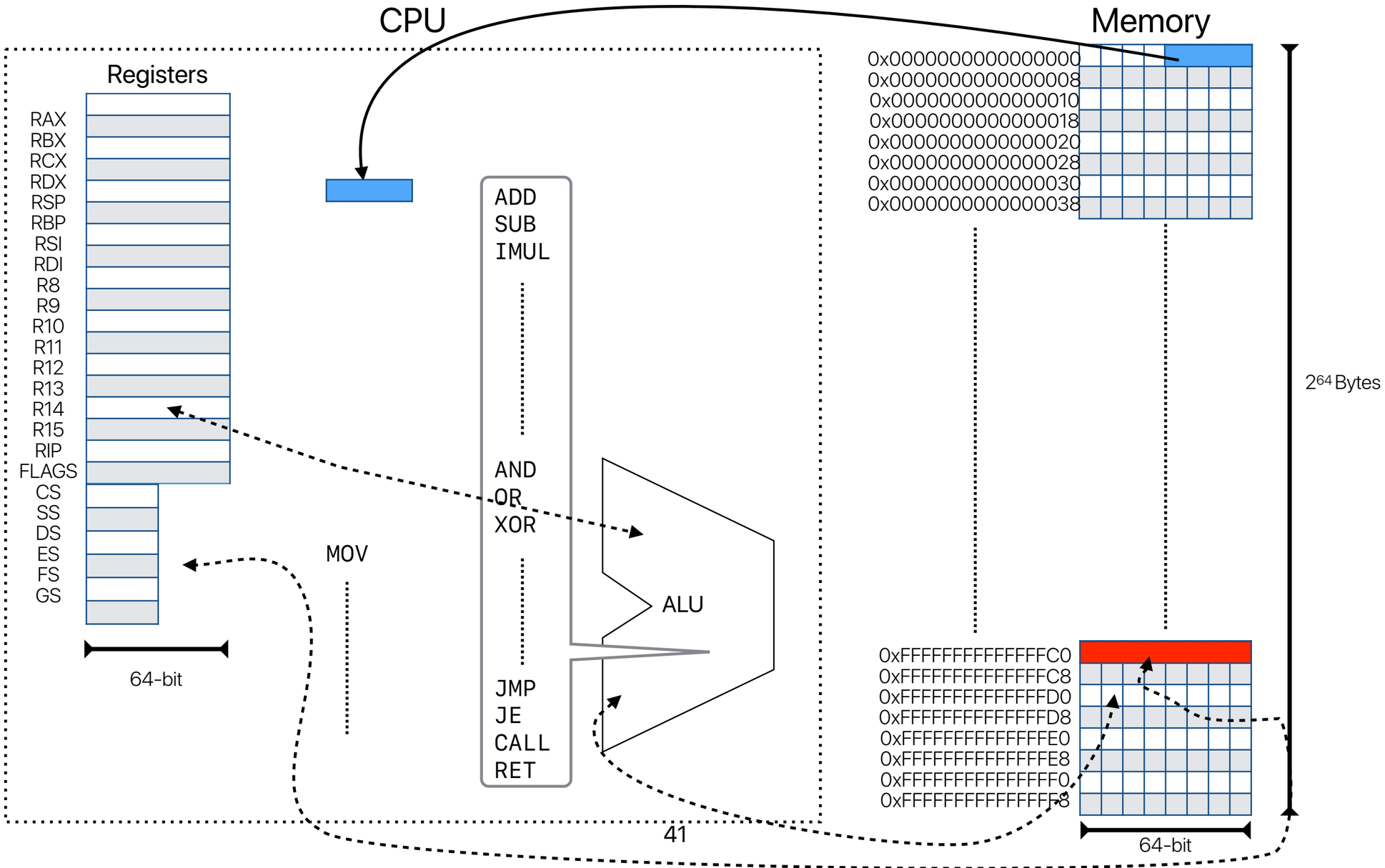
labels

Most instructions doesn't have an argument!

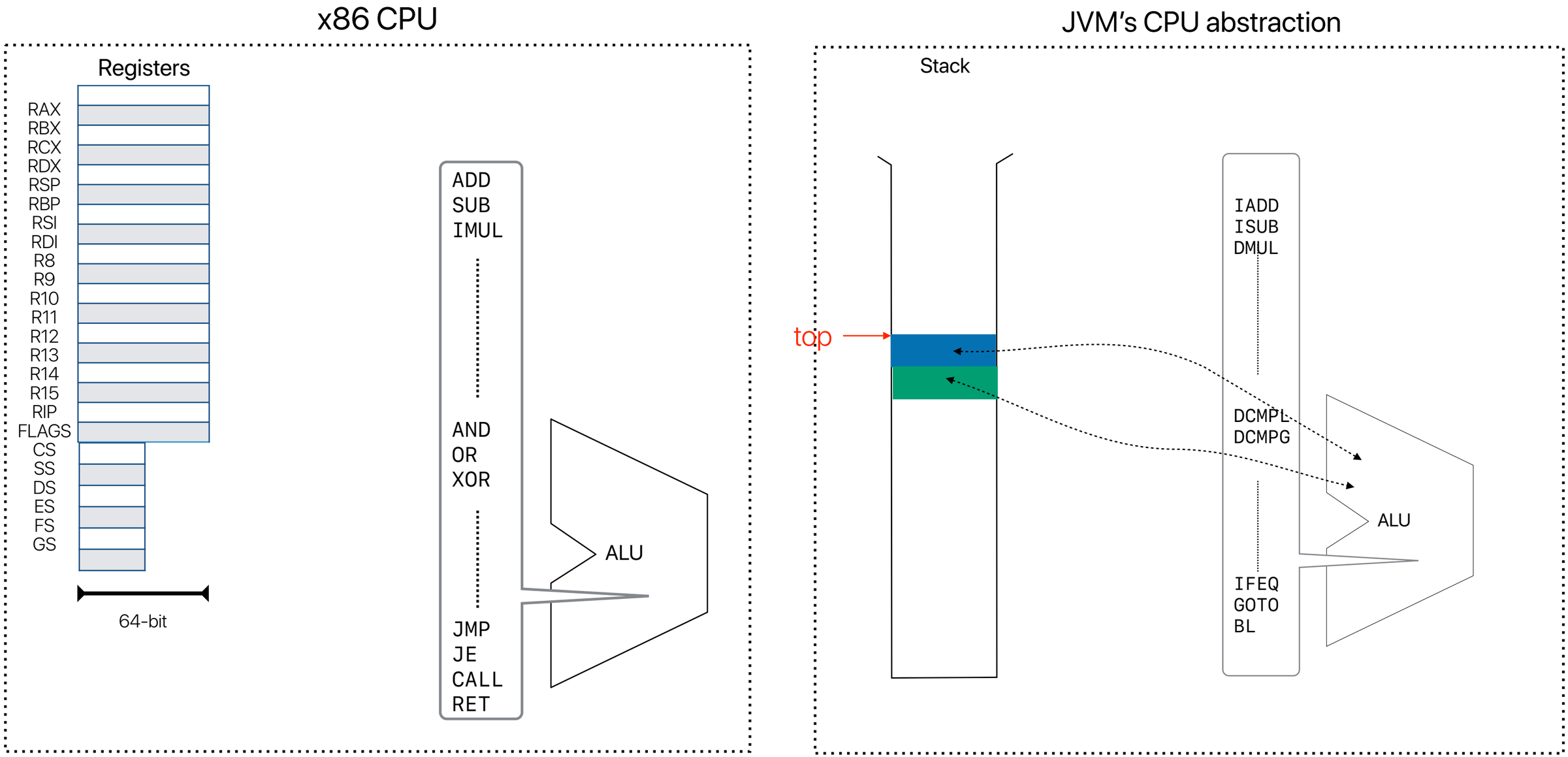
“Abstracted” Java Architecture



x86 ISA: the abstracted machine



Mismatching real architectures vs JVM 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

https://www.pollev.com/hungweitseng close in 1:30

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];
  }
}
```

How many of the following make(s) the performance different between version A & version B?

① IC

② CPI

③ CT

A. 0

B. 1

C. 2

D. 3

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	$O(n^2)$
Same	Instruction Count?	Same
Same	Clock Rate	Same
???	CPI	???

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	$O(n^2)$
Same	Instruction Count?	Same
Same	Clock Rate	Same
Better	CPI	Worse

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];
  }
}
```

How many of the following make(s) the performance different between version A & version B?

① IC

② CPI

③ CT

A. 0

B. 1

C. 2

D. 3

Programmer's impact

- By adding the "sort" in the following code snippet, what the programmer changes in the performance equation to achieve **better** performance?

```
std::sort(data, data + arraySize);

for (unsigned c = 0; c < arraySize*1000; ++c) {
        if (data[c%arraySize] >= INT_MAX/2)
            sum ++;
    }
}
```

A. CPI
B. IC
C. CT
D. IC & CPI
E. CPI & CT

Programmer's impact

- By adding the "sort" in the following code snippet, what the programmer changes in the performance equation to achieve **better** performance?

```
std::sort(data, data + arraySize);

for (unsigned c = 0; c < arraySize*1000; ++c) {
    if (data[c%arraySize] >= INT_MAX/2)
        sum ++;
}
}
```

A. CPI

B. IC ← **programmer changes IC as well, but not in the positive direction**

C. CT

D. IC & CPI

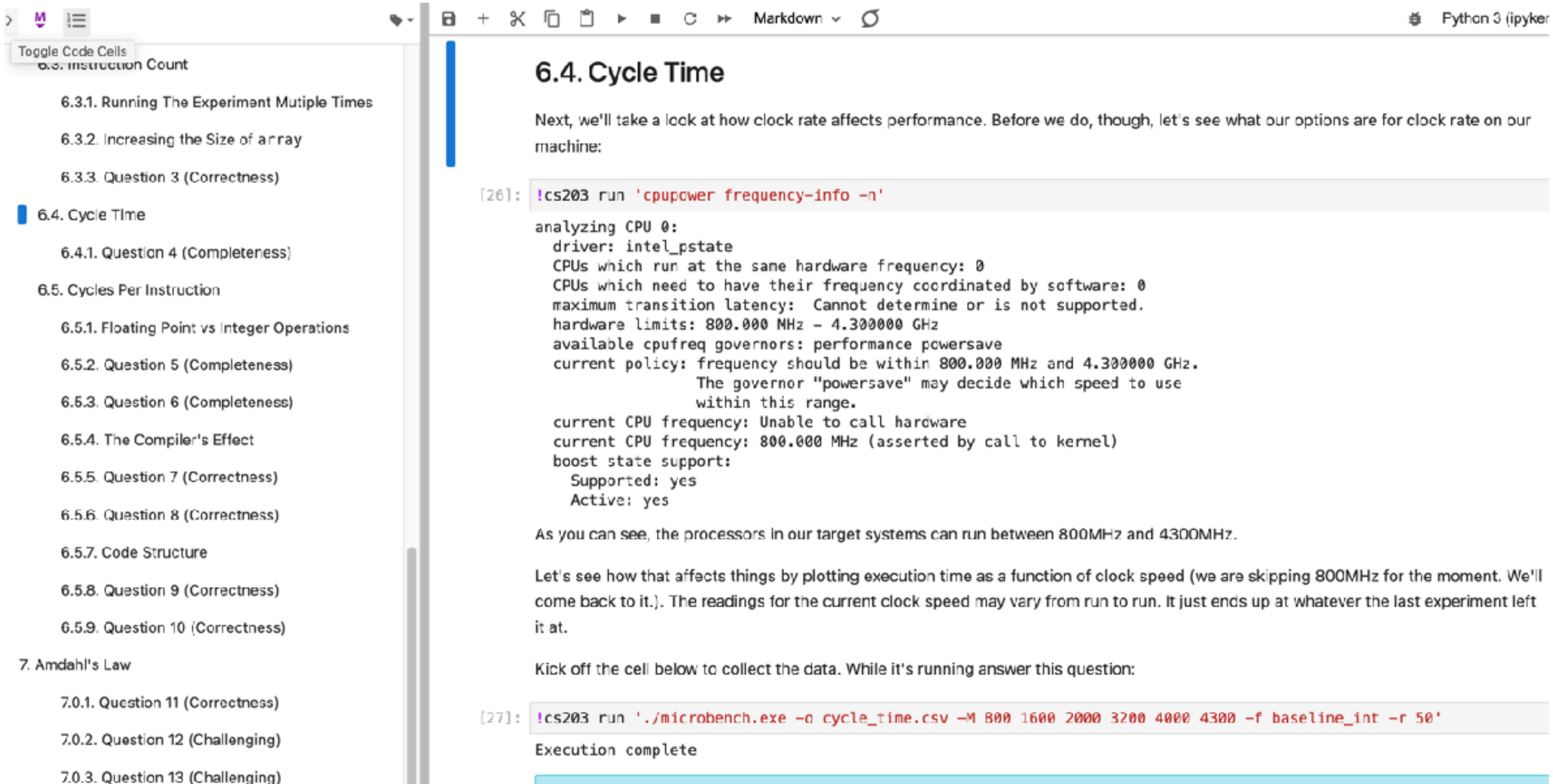
E. CPI & CT

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 159600
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`
$ newSpeedLow=$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:
$ newSpeedLow=$newSpeedTop  # make them the same in this example
```

Check your assignment 1 regarding your power in cycle time!



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

How compilers affect performance

- If we turn on "-O3" in gcc when compiling both code snippets **A** and **B**, how many of the following can we expect?
  - ① Compiler optimizations can reduce IC for both
  - ② Compiler optimizations can make the CPI lower for both
  - ③ Compiler optimizations can make the ET lower for both
  - ④ Compiler optimizations can transform code B into code A

A. 0
B. 1
C. 2
D. 3
E. 4

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];
  }
}
```

How compilers affect performance

- If we turn on "-O3" in gcc when compiling both code snippets **A** and **B**, how many of the following can we expect?
  - ① Compiler optimizations can reduce IC for both
  - ② Compiler optimizations can make the CPI lower for both
  - ③ Compiler optimizations can make the ET lower for both
  - ④ Compiler optimizations can transform code B into code A

Compiler can apply loop unrolling, constant propagation naively to reduce IC

Reduced IC does not necessarily mean lower CPI — compiler may pick one longer instruction to replace a few shorter ones

Compiler cannot guarantee the combined effects lead to better performance!

Compiler will not significantly change programmer's code since compiler cannot guarantee if doing that would affect the correctness

A. 0
B. 1
C. 2
D. 3
E. 4

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];
  }
}
```

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?

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

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

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

- 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

Announcement

- Programming assignment 1 **due this Thursday** & Assignment 1 **released**
  - We cannot help you at the last minute — please start early
  - Watch before you start https://youtu.be/m7OoY8y_Isk
  - 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
    - Hung-Wei Tseng — TuW 1p-2p @ WCH 406
    - Hongrui Zhang — W 4p-5p @ Zoom; Th 2p-4p @ WCH 110
- Reading quiz due next Tuesday before the lecture
  - We will drop two of your least performing reading quizzes
  - Once you closed, you cannot restart
- Check our website for slides/quiz links, Gradescope to submit 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