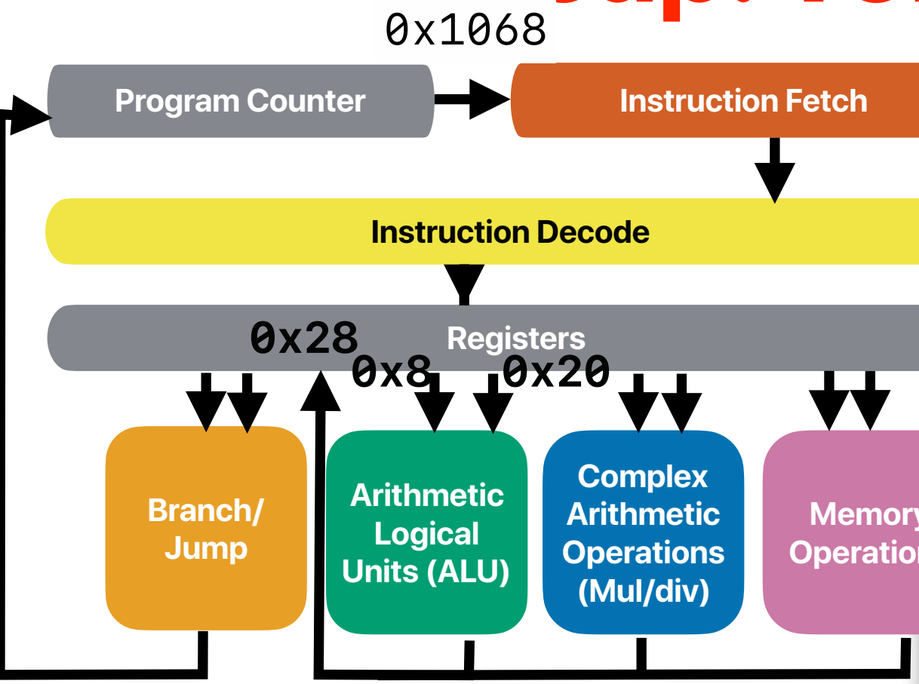
Performance (2): What can I change?

Hung-Wei Tseng

Recap: von Neumann architecture

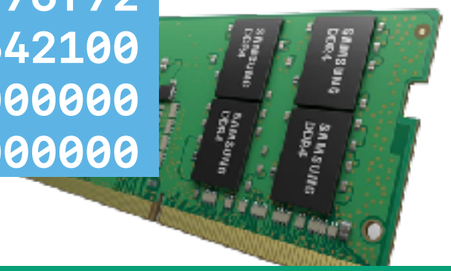


Processor

4883ec08

sub $0x8,%rsp

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



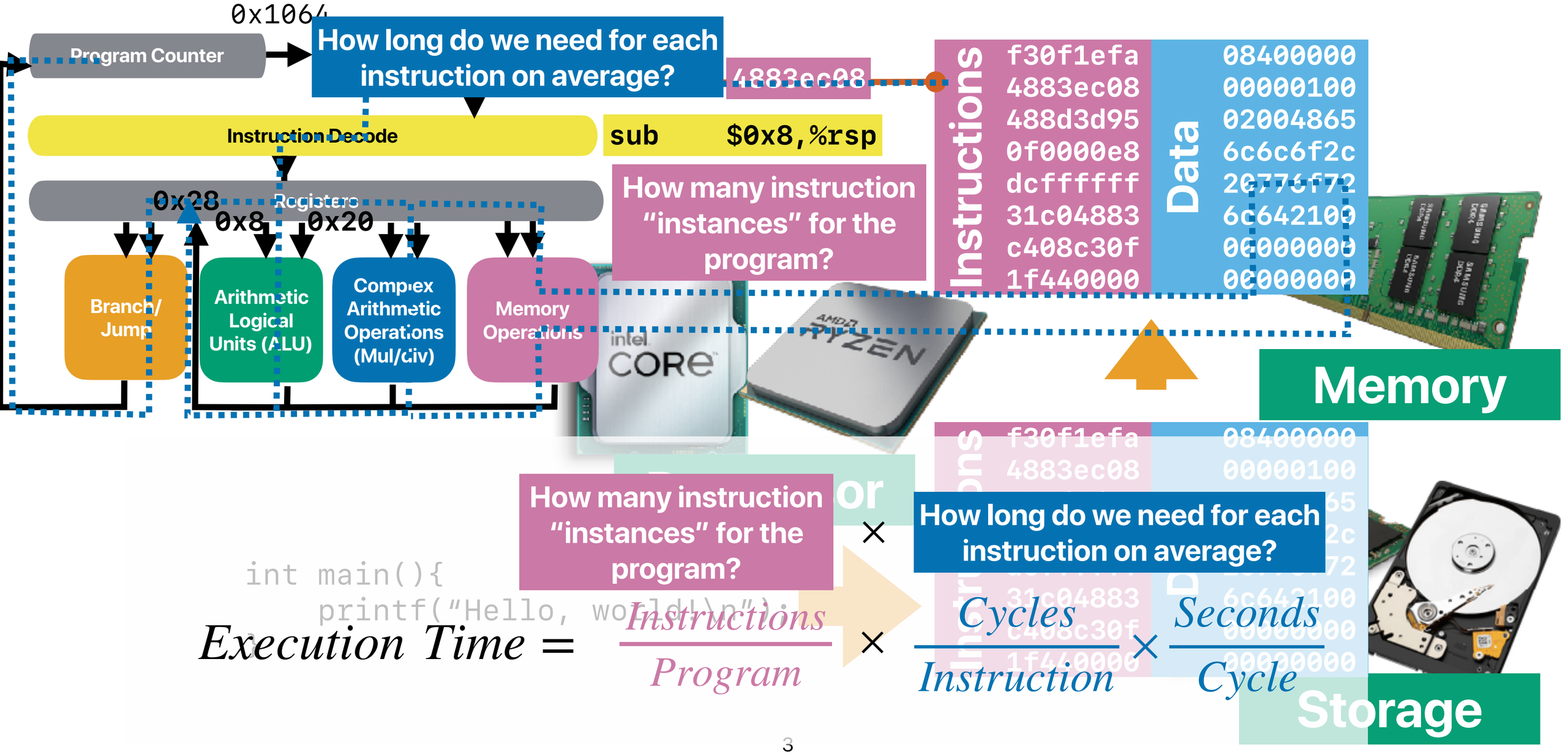
Memory

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

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

Storage

Execution time of a program in the von Neumann model



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;`
`}`
	`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`

memory inst.

branch inst.

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

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

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

1000000000x

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

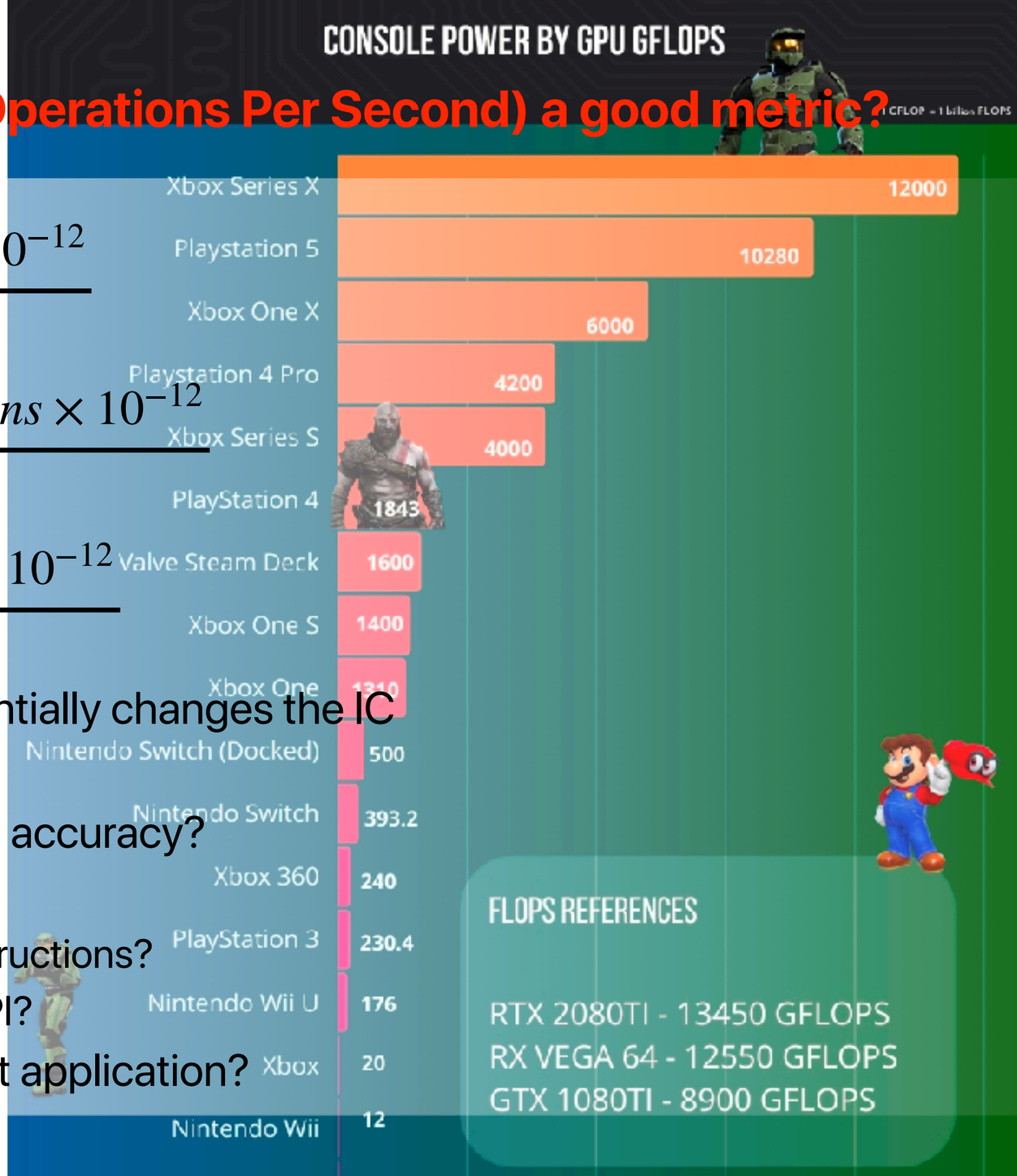
$$TFLOPS = \frac{\#\ of\ floating\ point\ instructions \times 10^{-12}}{Exection\ Time}$$

$$= \frac{IC \times \%\ of\ floating\ point\ instructions \times 10^{-12}}{IC \times CPI \times CT}$$

IC is gone!

$$= \frac{\%\ of\ floating\ point\ instructions \times 10^{-12}}{CPI \times CT}$$

- If we have more iterations? Larger datasets? — potentially changes the IC
- Different applications definitely have different ICs!
- 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?



Recap: 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

3:00

What's your favorite programming language and why?

Outline

- What can affect each factor in the classic CPU performance equation?
  - Programming languages
  - Programmers
  - Compilers
  - Complexity

Programming Languages

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

How programming languages affect performance

- Performance equation consists of the following three factors
  ① IC
  ② CPI
  ③ CT

  How many can the **programming language** affect?

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

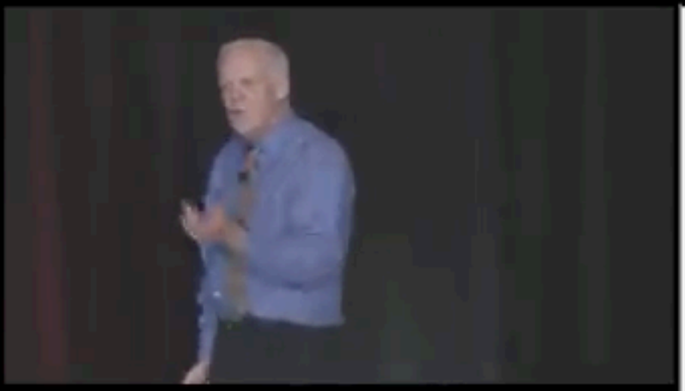
https://www.pollev.com/hungweitseng close in 1:30

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. Perl
  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

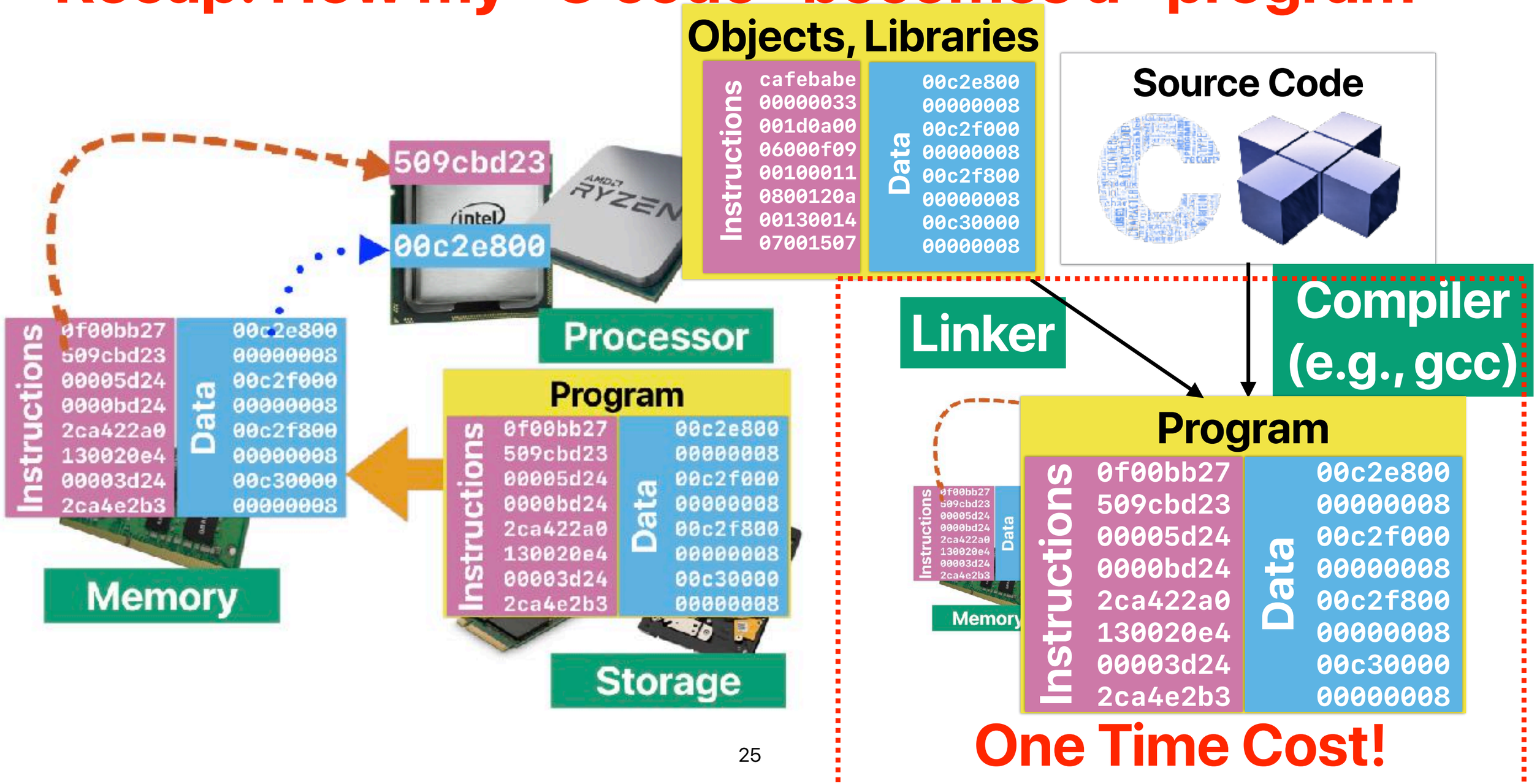
- How many instructions are there in “Hello, world!”

	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	

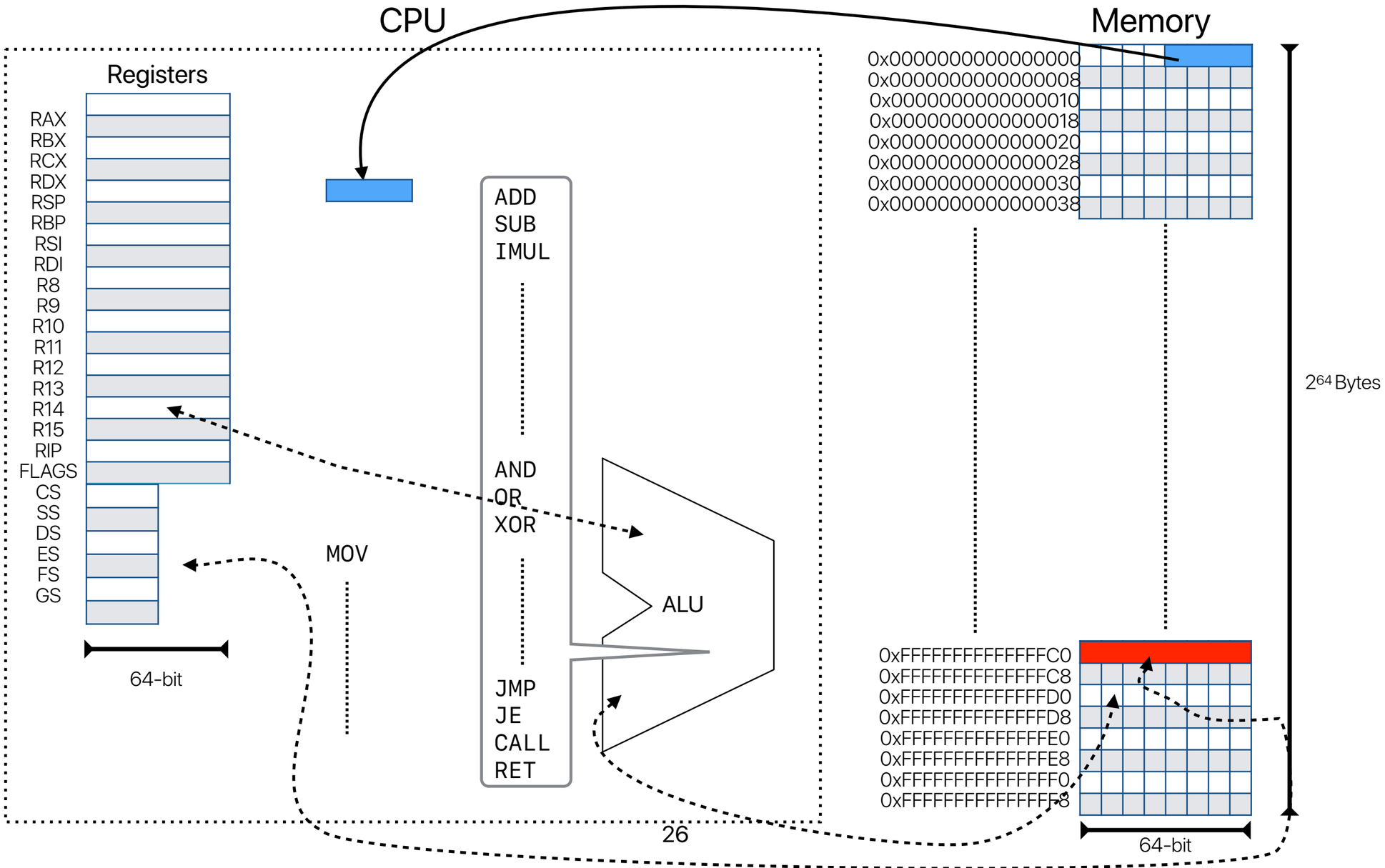
Programming languages

- 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"



x86 ISA: the abstracted machine



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

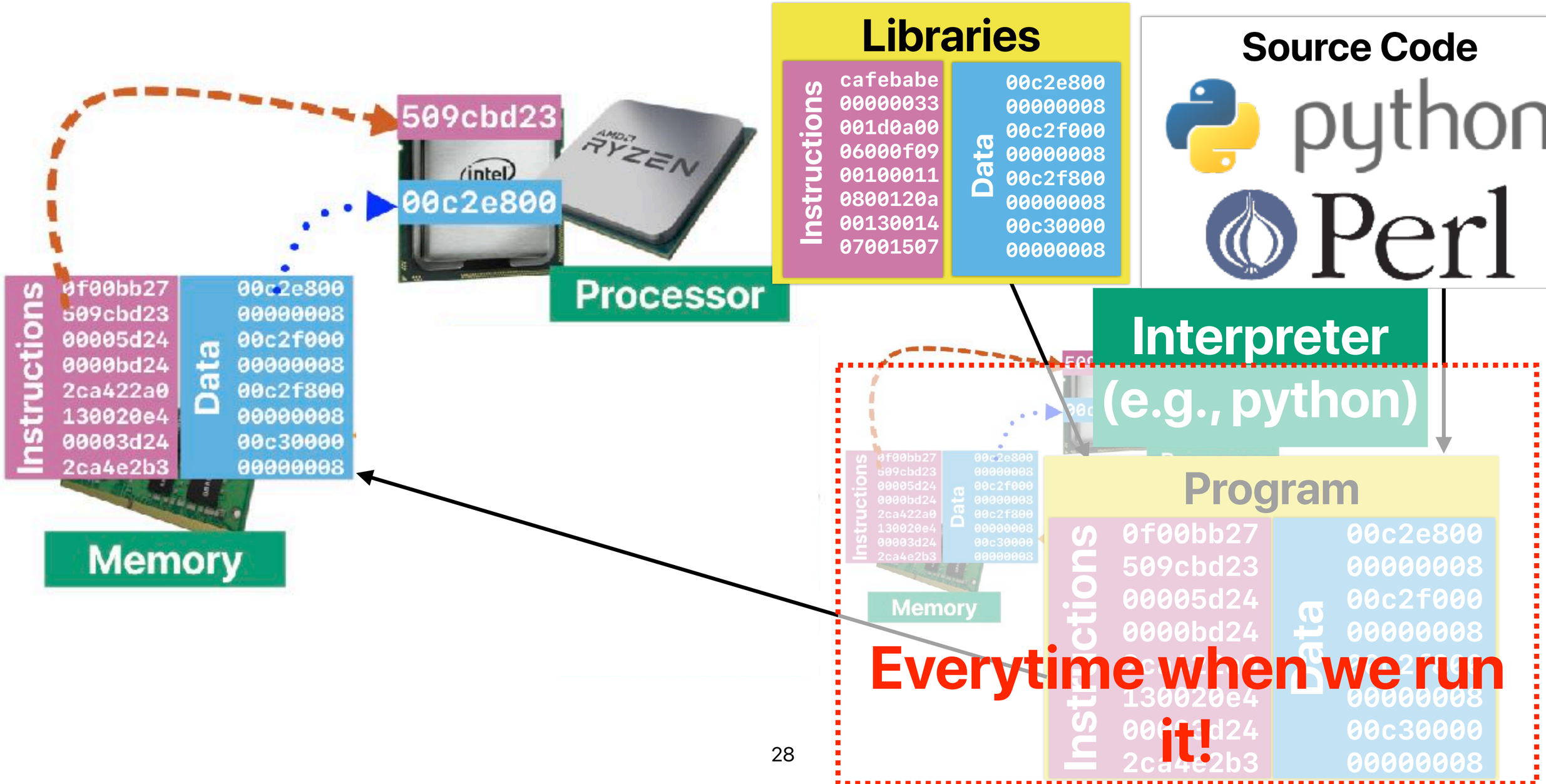
```
main:
.LFB0:
    endbr64
    pushq   %rbp
    movq %rsp, %rbp
    movl $0, -8(%rbp)
    movl $0, -4(%rbp)
    movl $0, -8(%rbp)
    jmp  .L2
.L3:
    movl -8(%rbp), %eax
    cltq
    leaq 0(,%rax,4), %rdx
    leaq A(%rip), %rax
    movl (%rdx,%rax),
%eax
    addl %eax, -4(%rbp)
    addl $1, -8(%rbp)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

Compiler

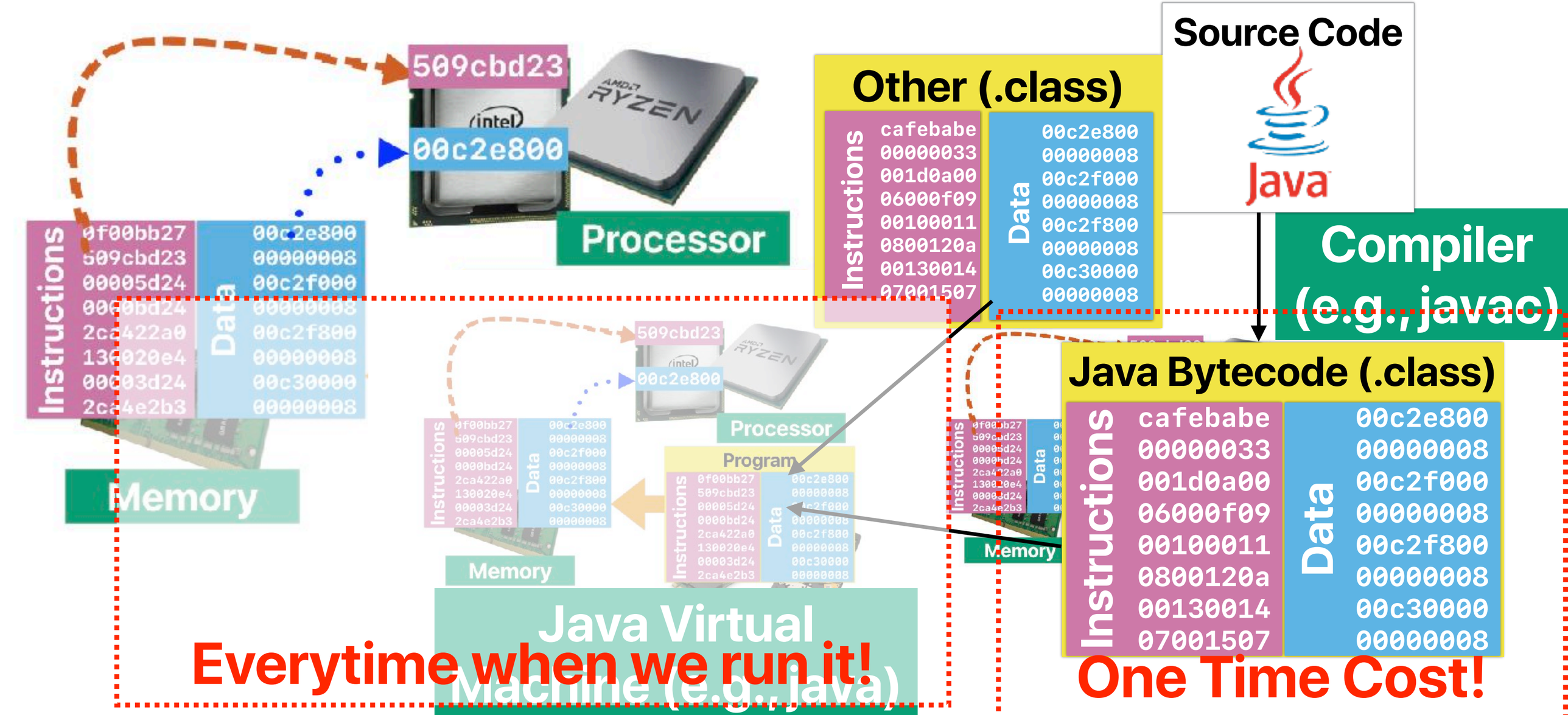
```
Contents of section .text:
 0000 f30f1efa 554889e5 c745f800 000000c7
 0010 45fc0000 0000c745 f8000000 00eb1e8b
 0020 45f84898 488d1425 00000000 488d0500
 0030 0000008b 04020145 fc8345f8 01837df8
 0040 137edcb8 00000000 5dc3
```

Instructions

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

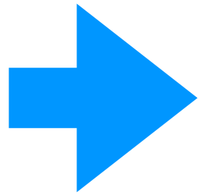


Recap: How my "Java code" becomes a "program"



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);
}
```

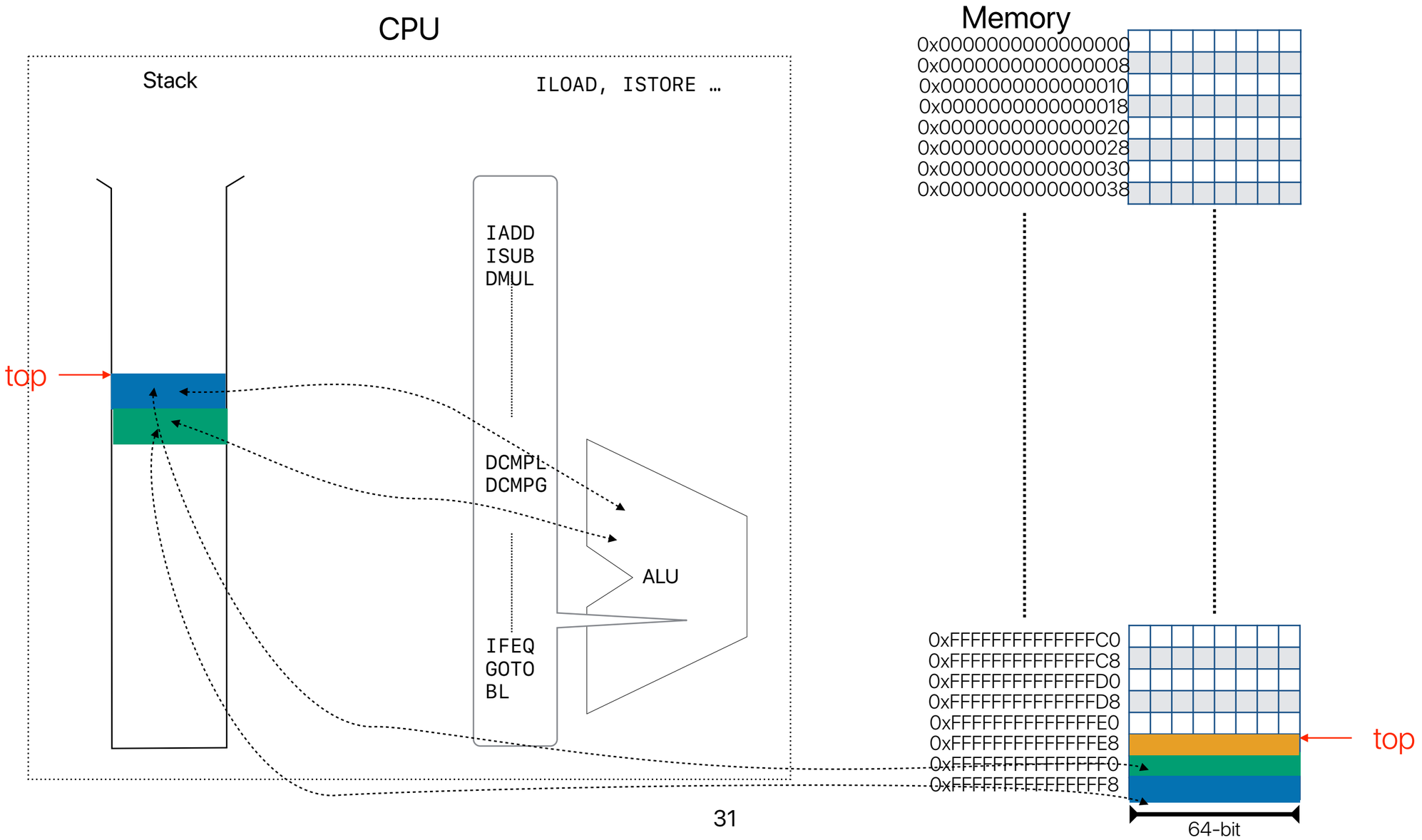


```
 0: iload_0
 1: ifne          6
 4: iconst_0
 5: ireturn
 6: iload_0
 7: iconst_1
 8: if_icmpne     13
11: iconst_1
12: ireturn
13: iload_0
14: iconst_1
15: isub
16: invokestatic  #2
19: iload_0
20: iconst_2
21: isub
22: invokestatic  #2
25: iadd
26: ireturn
```

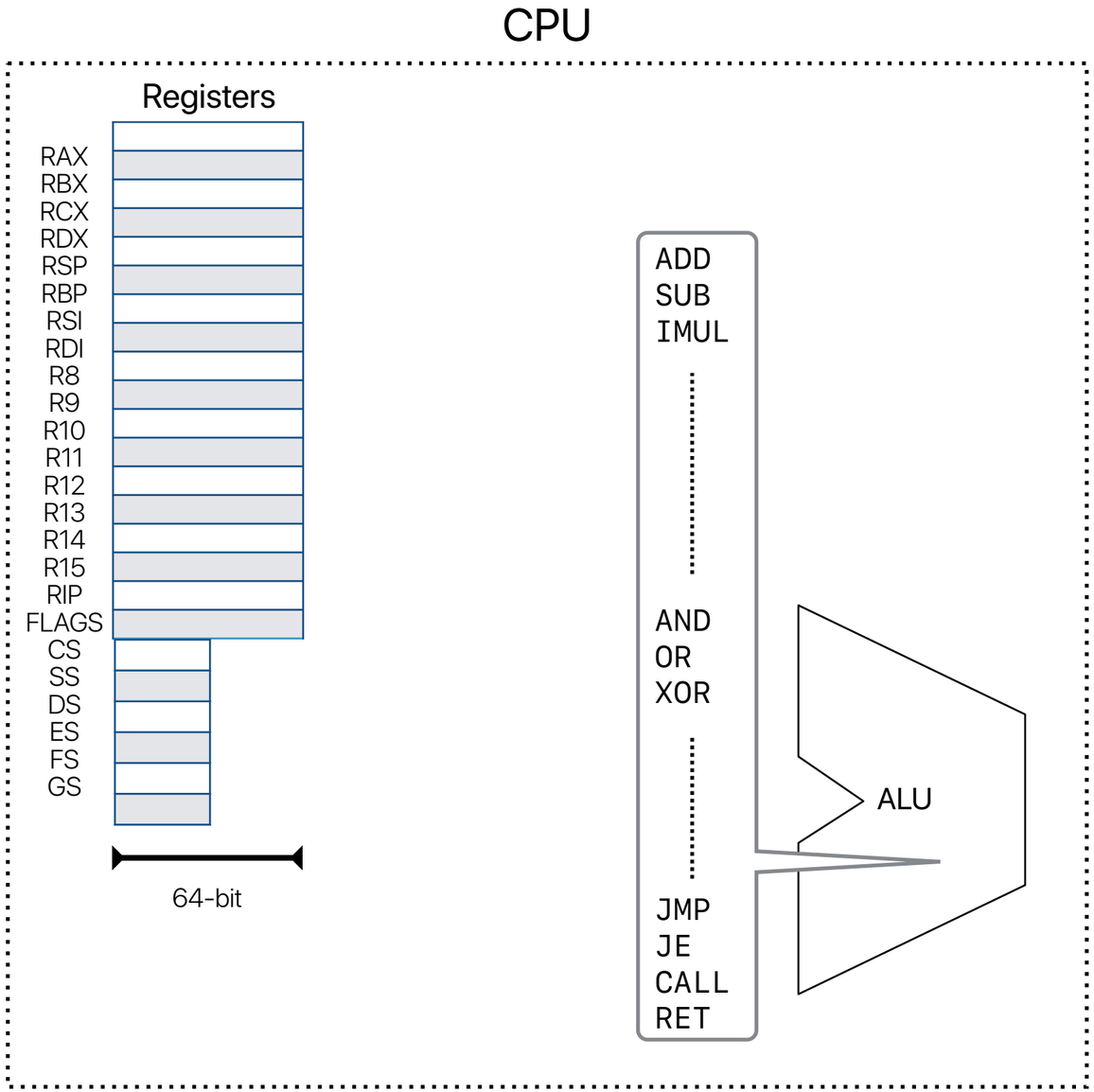
labels

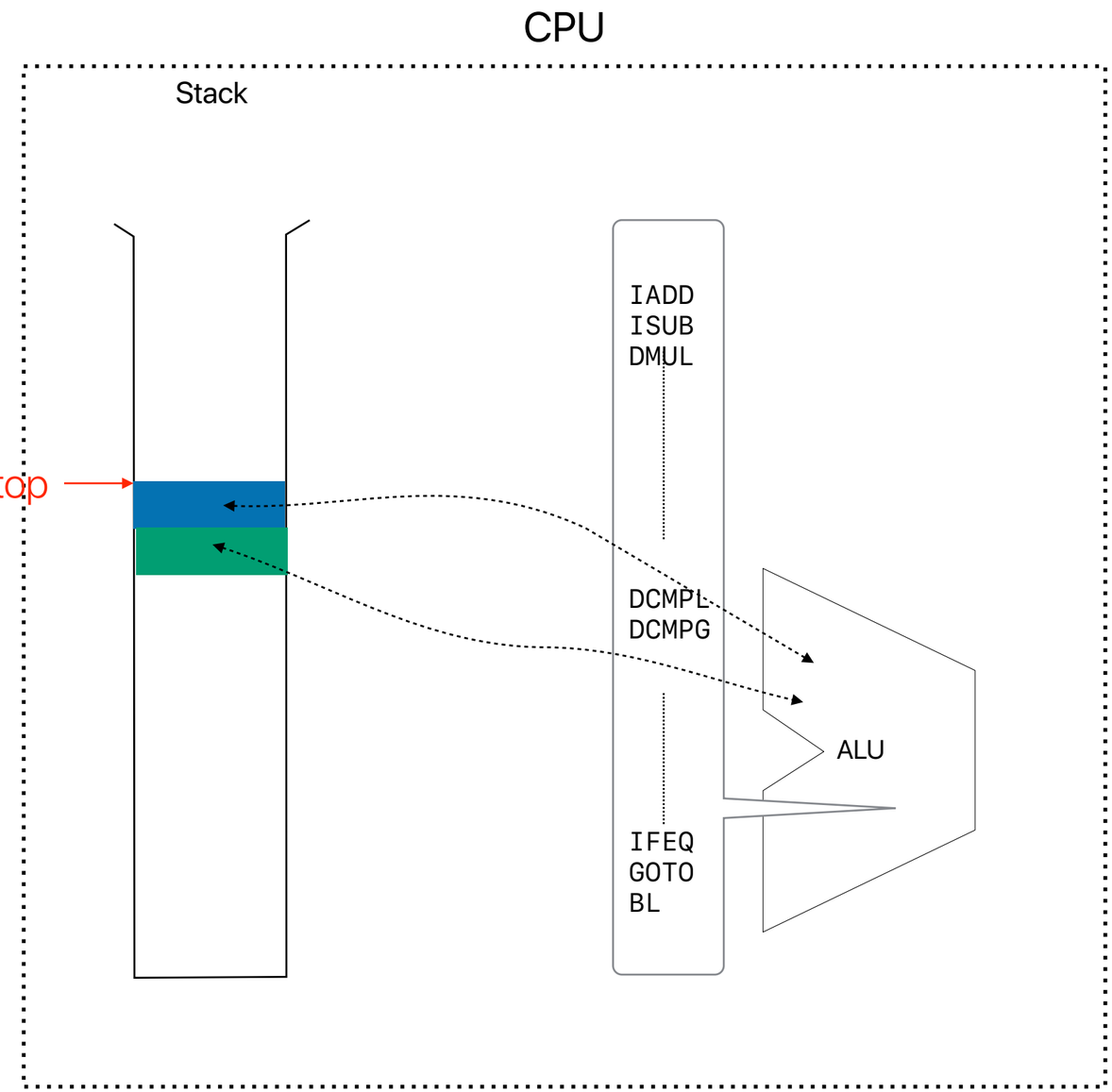
Most instructions doesn't have an argument!

“Abstracted” Java Architecture

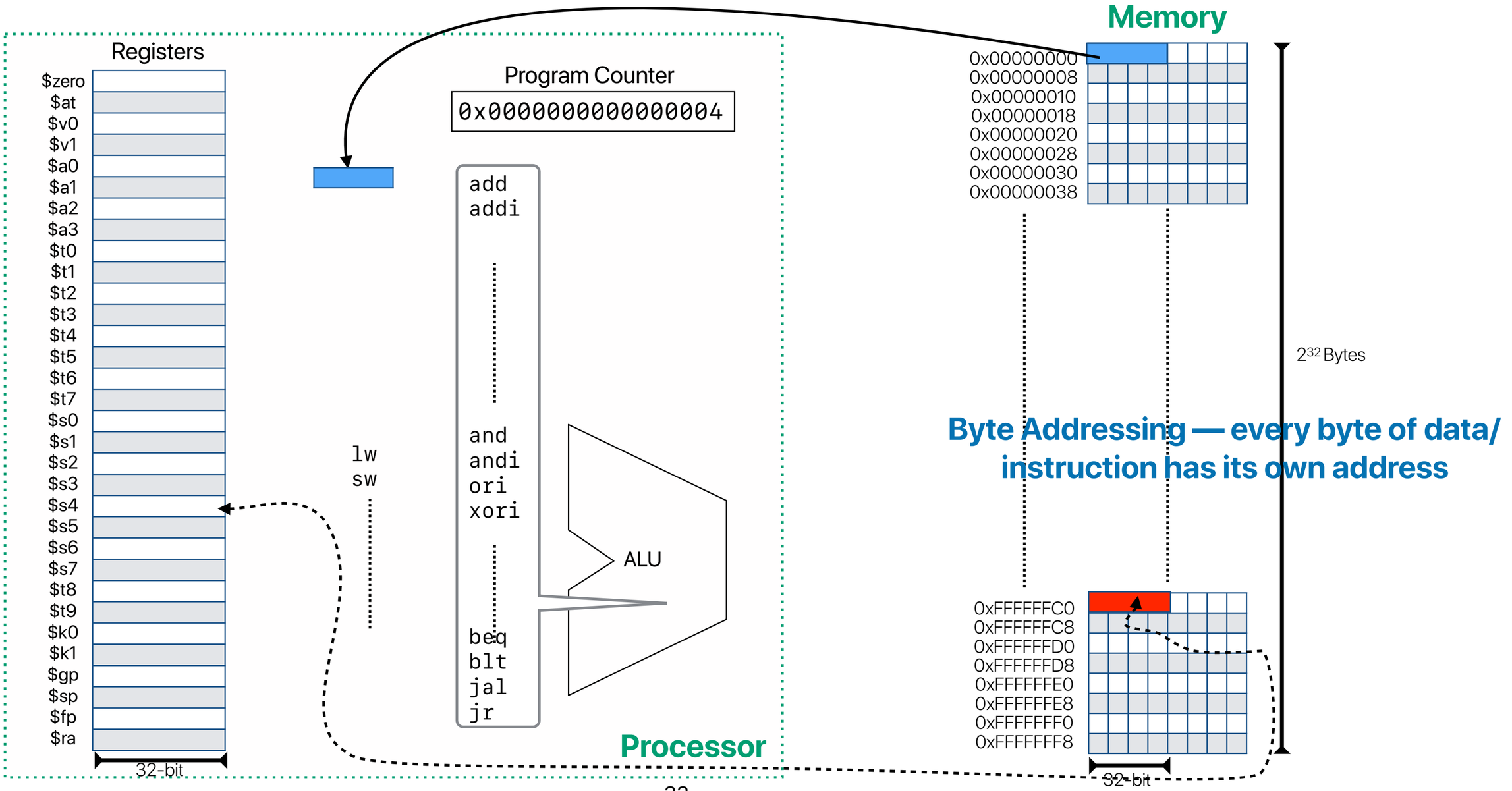


Mismatching x86 abstraction vs JVM





The abstracted MIPS machine



How programming languages affect performance

- Performance equation consists of the following three factors
  1. IC
  2. CPI
  3. CT ← **Programmer uses programming languages to create library/ programs that changes the CT, not the programming language itself makes the change**

How many can the **programming language** affect?

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

Takeaways: What matters?

- Different programming languages can generate machine operations with different orders of magnitude performance — programmers need to make wise choice of that!

Programmers

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
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

Change the algorithm implementation to achieve better CPI

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 — programmer changes IC as well, but not in the positive direction
B. IC ← More IC does not necessarily mean lower performance! Remember
C. CT
D. IC & CPI
E. CPI & CT

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

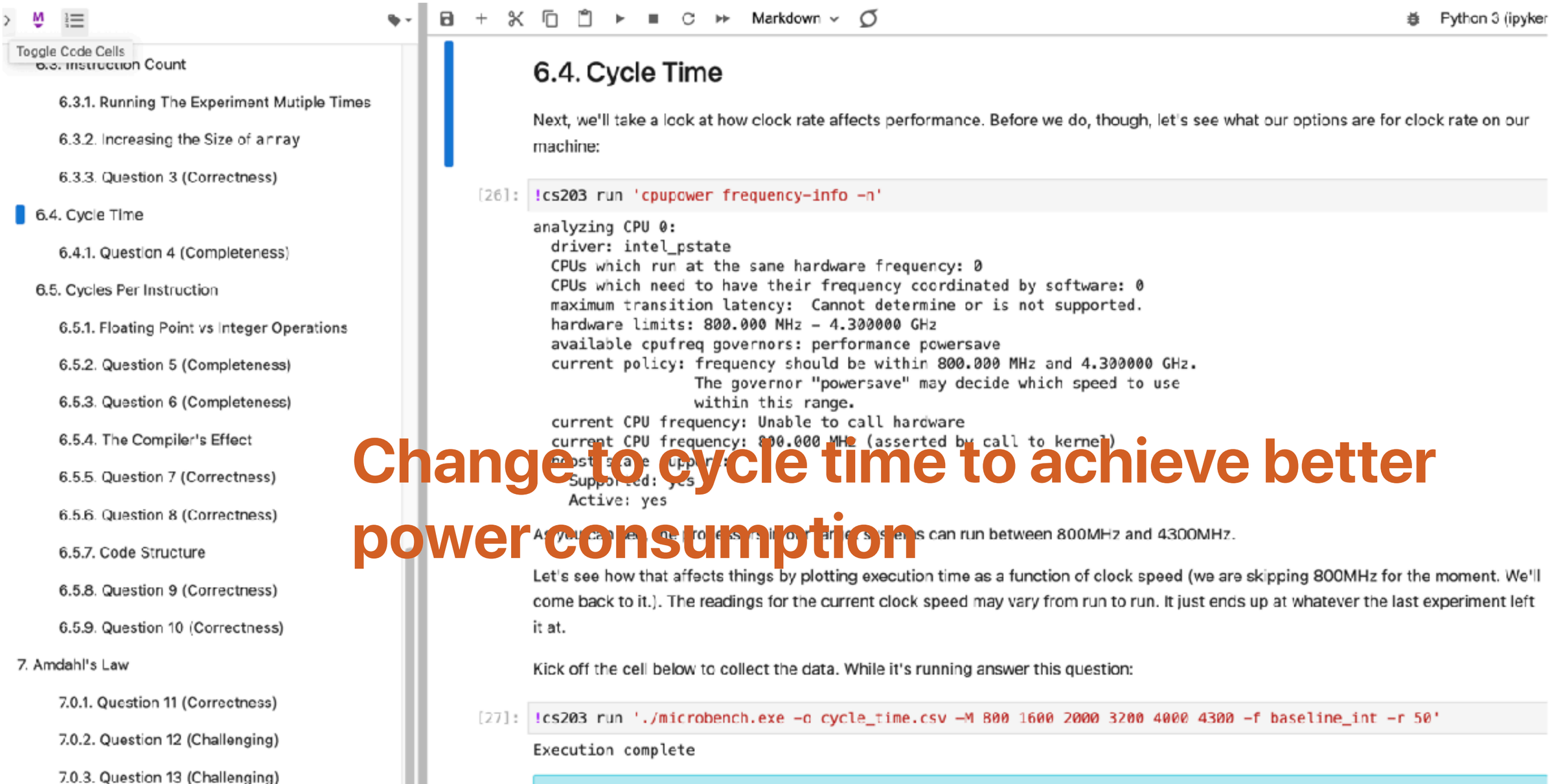
Adding more operations to achieve better CPI

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 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!



Change to cycle time to achieve better power consumption

Takeaways: What matters?

- 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
  - Change the algorithm implementation to achieve better CPI
  - Change the data alignment to achieve better CPI
  - Change the frequency to achieve better power/energy efficiency

Compilers

How compilers affect performance

- If we turn on "-O3" flag when using gcc to compile 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" flag when using gcc to compile 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 matters?

- 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
  - Change the algorithm implementation to achieve better CPI
  - Change the data alignment to achieve better CPI
  - Change the frequency to achieve better power/energy efficiency
- 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!

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 matters?

- 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
  - Change the algorithm implementation to achieve better CPI
  - Change the data alignment to achieve better CPI
  - Change the frequency to achieve better power/energy efficiency
- 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

- Reading quiz 2 due next **tomorrow before the lecture** — we will drop two of your least performing reading quizzes
- Assignment 1 **due this upcoming Sunday**
  - 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
- Book your
  - Practice test with Triton Testing Center on either 8/7, 8/11, and 8/12 through https://us.prairietest.com/pt/course/10381
  - Midterms on slots for 8/18-8/21 (If you've booked other dates, those reservations are cancelled since they're not reasonable timeframes. Please rebook.)
  - Finals on slots for 9/5
- Check our website for slides, Gradescope for assignments, discord for discussions
- Check your grades at https://www.escalab.org/my_grades
- Youtube channel for lecture recordings: https://www.youtube.com/profusagi
- Podcast: podcast.ucsd.edu

Computer
Science &
Engineering

142

つづく