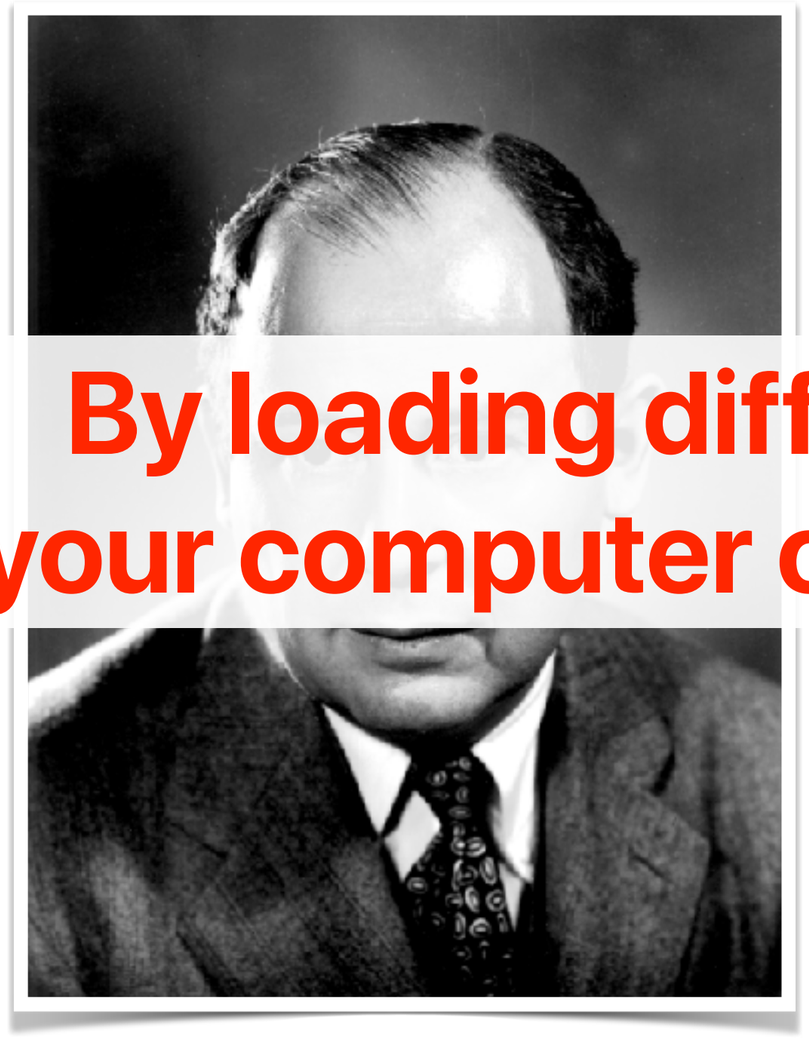
Performance (2): One Thing Right

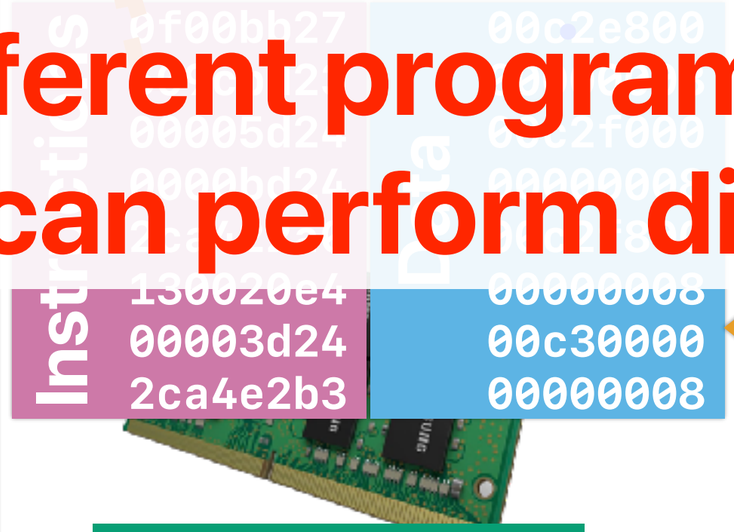
Hung-Wei Tseng

Recap: von Neumman Architecture

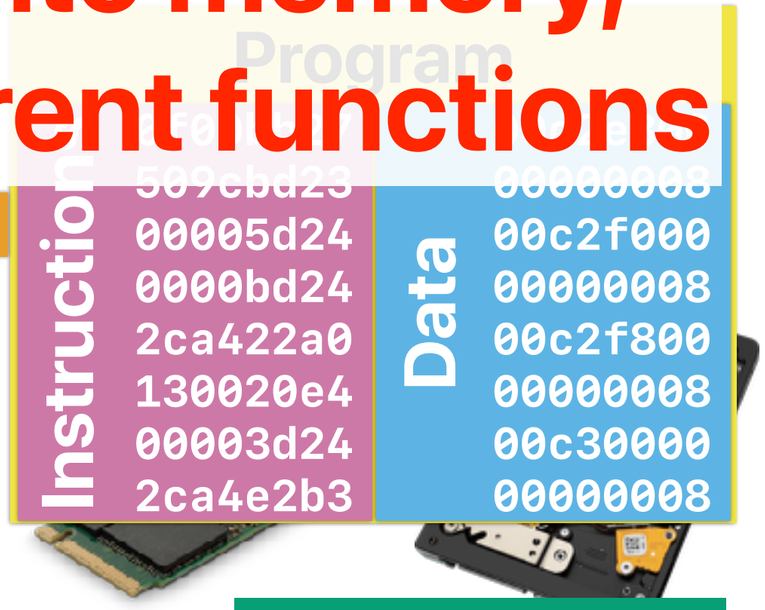


Processor

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



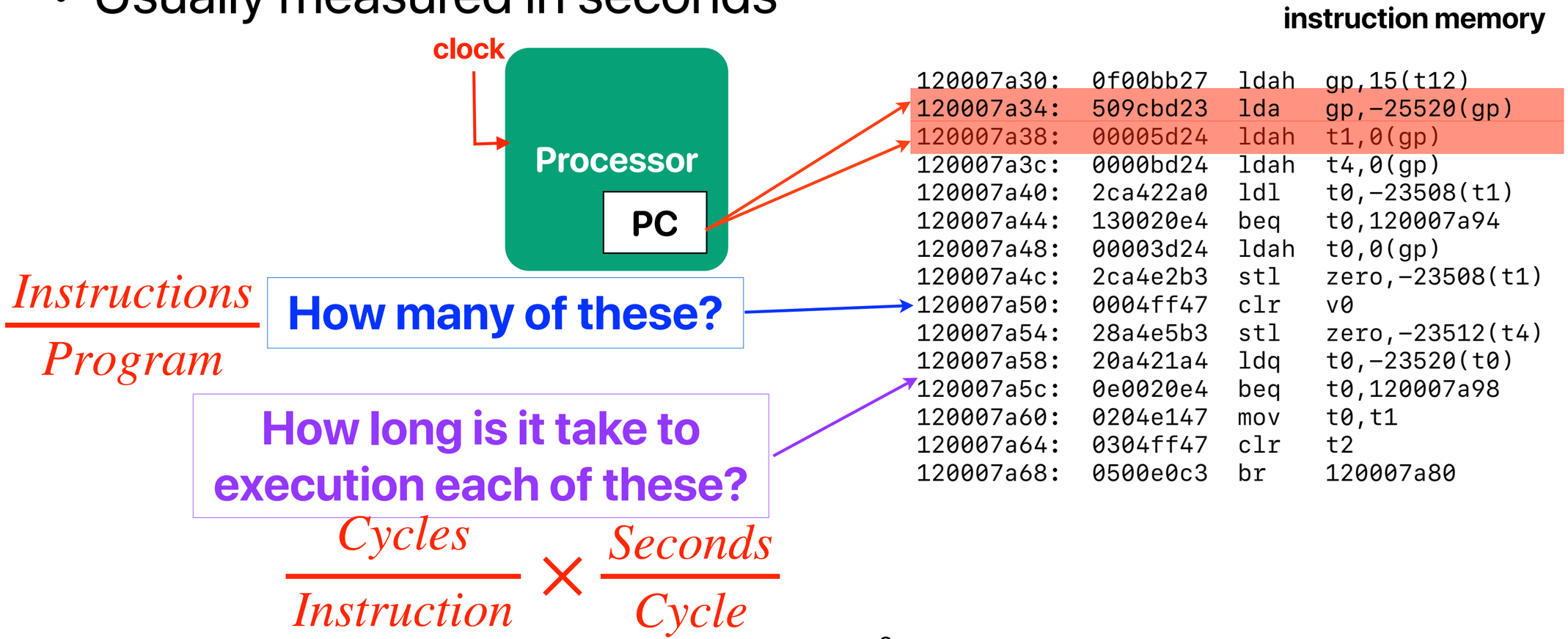
Memory



Storage

Recap: Execution Time

- The simplest kind of performance
- Shorter execution time means better performance
- Usually measured in seconds



Recap: CPU Performance Equation

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

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

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

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

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

Recap: Speedup

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

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

- The speedup of Y over X

$$Speedup = \frac{Execution\ Time_X}{Execution\ Time_Y}$$

Recap: 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

Recap: 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

Recap: 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
```

Recap: How programmer affects performance?

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

  How many can a **programmer** affect?

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

3:00

What change will you make to improve your life? Why?

What do you want your computer to be?

What change will you make to improve your life

- Some ideas
  - Time management
  - Better habits
- Why
  - Because it's important!

Outline

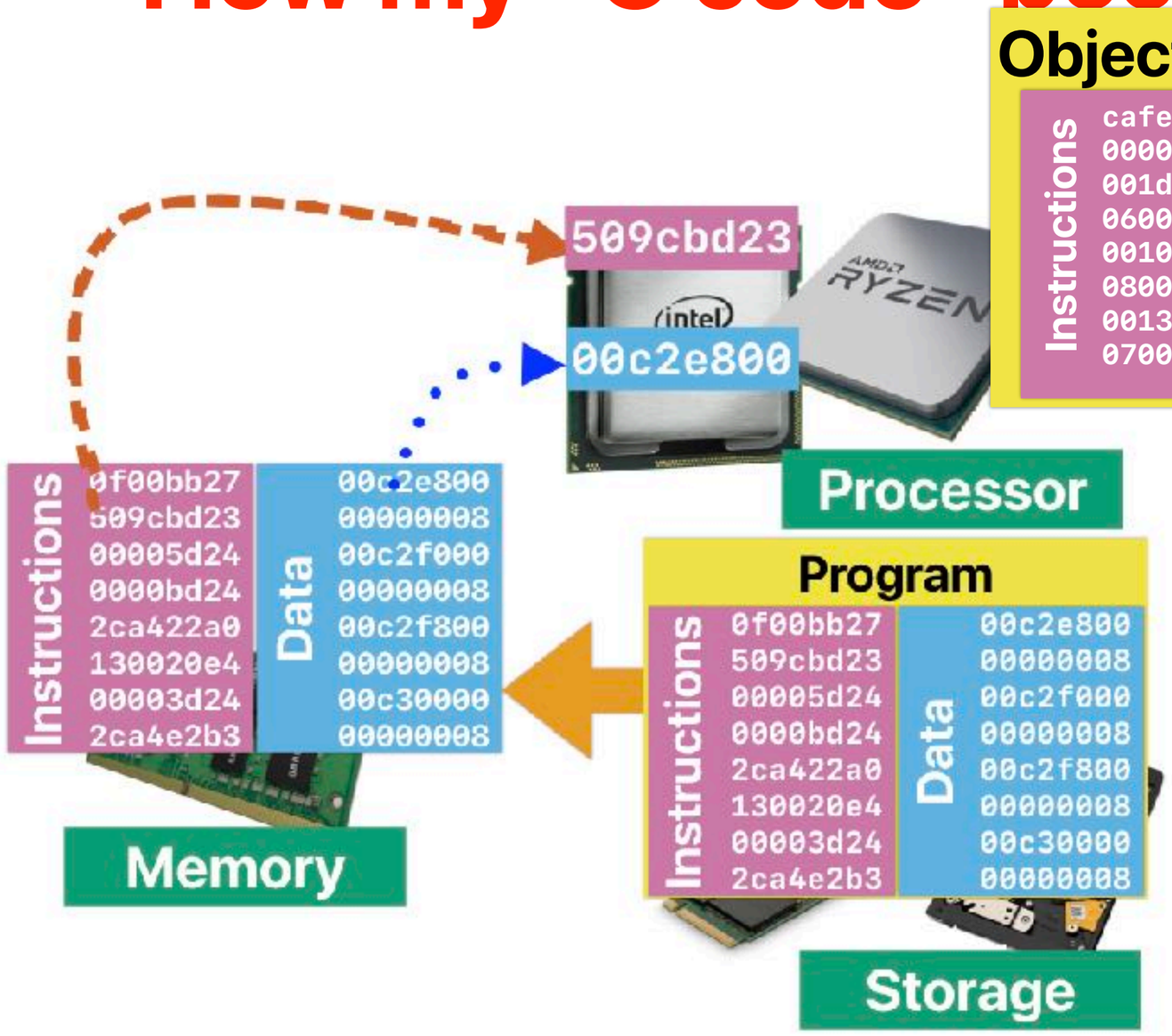
- What affects each factor in “Performance Equation”
  - Programming Languages
  - Compilers
  - Instruction Set Architectures (ISAs)
- Amdahl’s Law and it’s implications

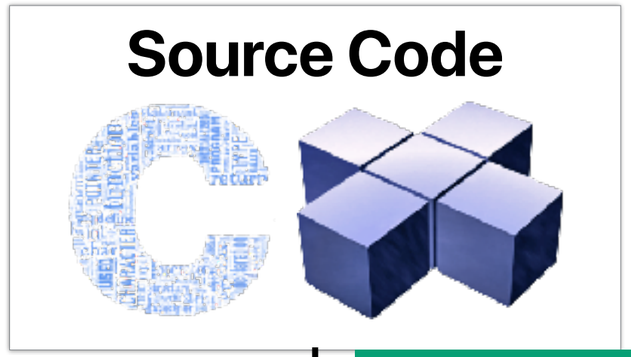
Programming languages

- 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	1.5

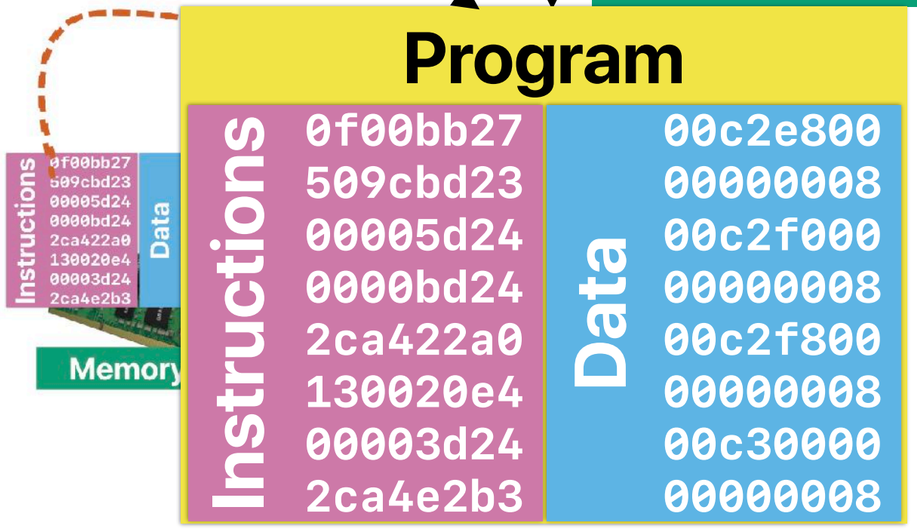
How my "C code" becomes a "program"





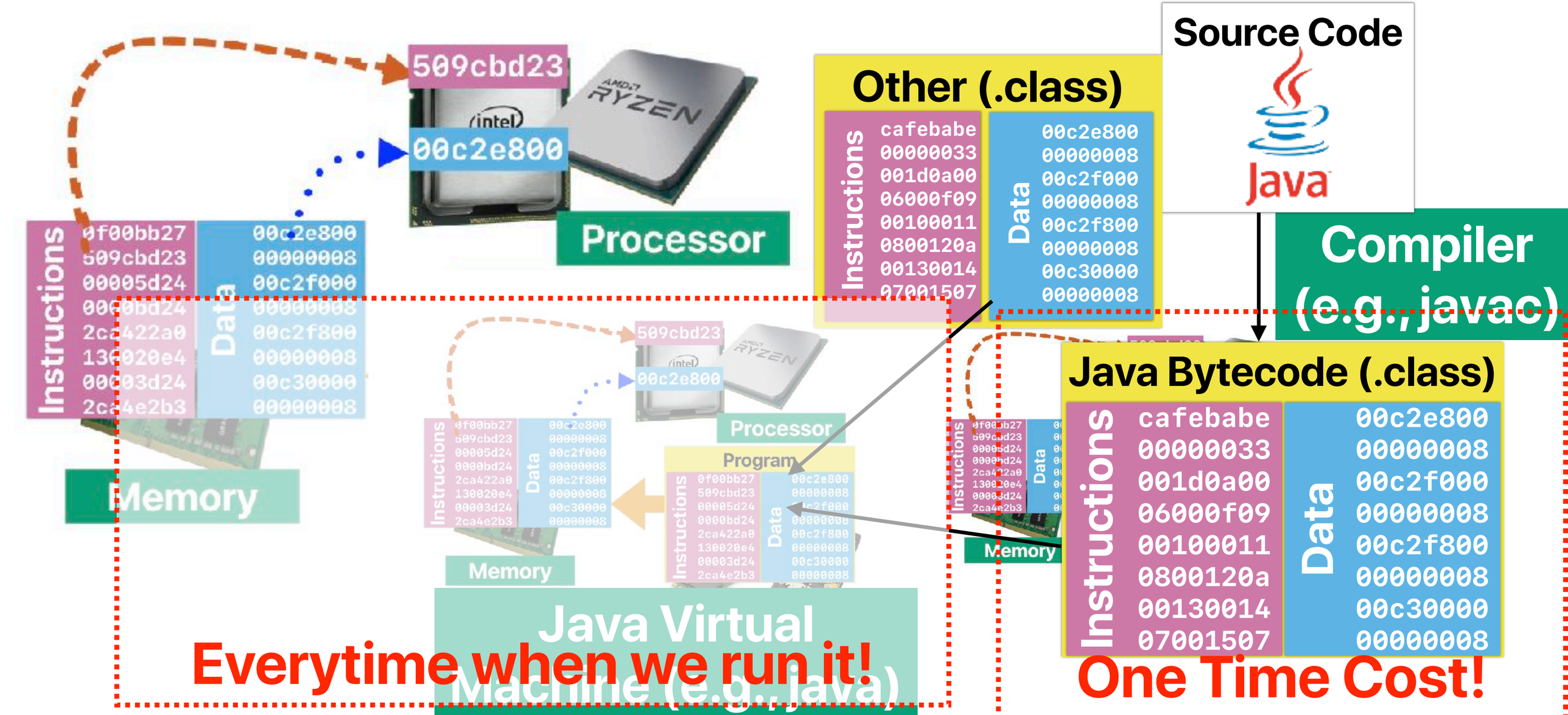
Linker

Compiler (e.g., gcc)

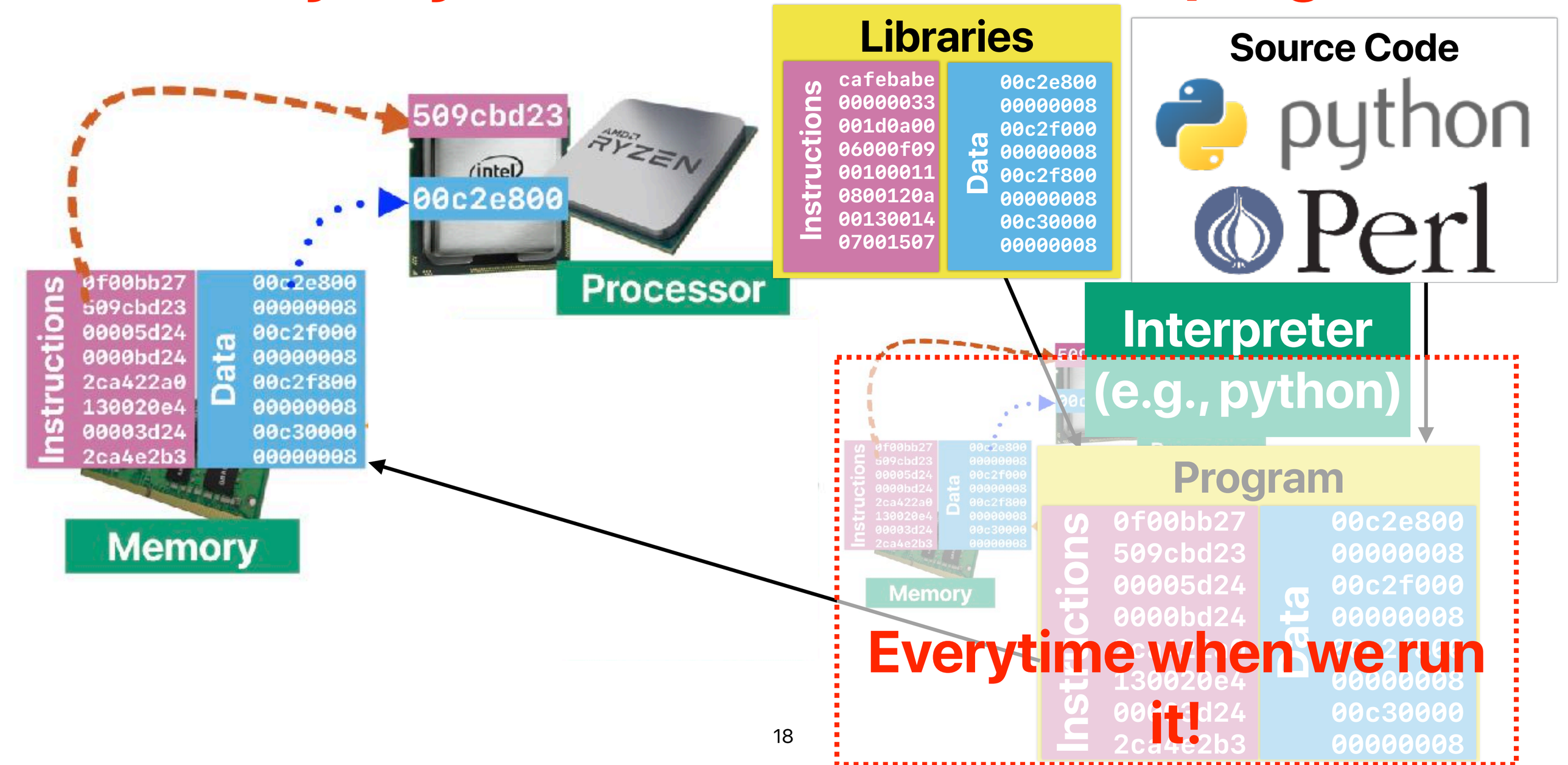


One Time Cost!

How my “Java code” becomes a “program”



How my "Python code" becomes a "program"



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

How compilers affect performance

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

  How many can the **compiler** affect?

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

Revisited the demo with compiler optimizations!

- gcc has different optimization levels.
  - -O0 — no optimizations
  - -O3 — typically the best-performing optimization

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

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

  How many can the **compiler** affect?

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

Instruction Set Architecture (ISA) & Performance

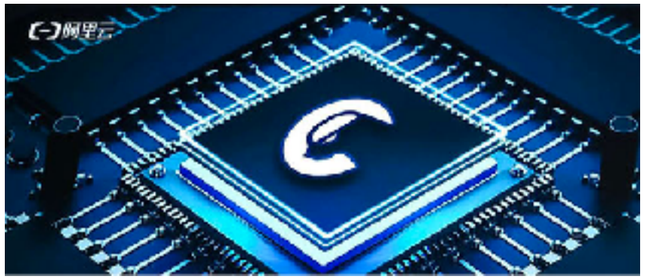
Recap: ISA — the interface b/w processor/software

- Operations
  - Arithmetic/Logical, memory access, control-flow (e.g., branch, function calls)
  - Operands
    - Types of operands — register, constant, memory addresses
    - Sizes of operands — byte, 16-bit, 32-bit, 64-bit
- Memory space
  - The size of memory that programs can use
  - The addressing of each memory locations
  - The modes to represent those addresses

Popular ISAs

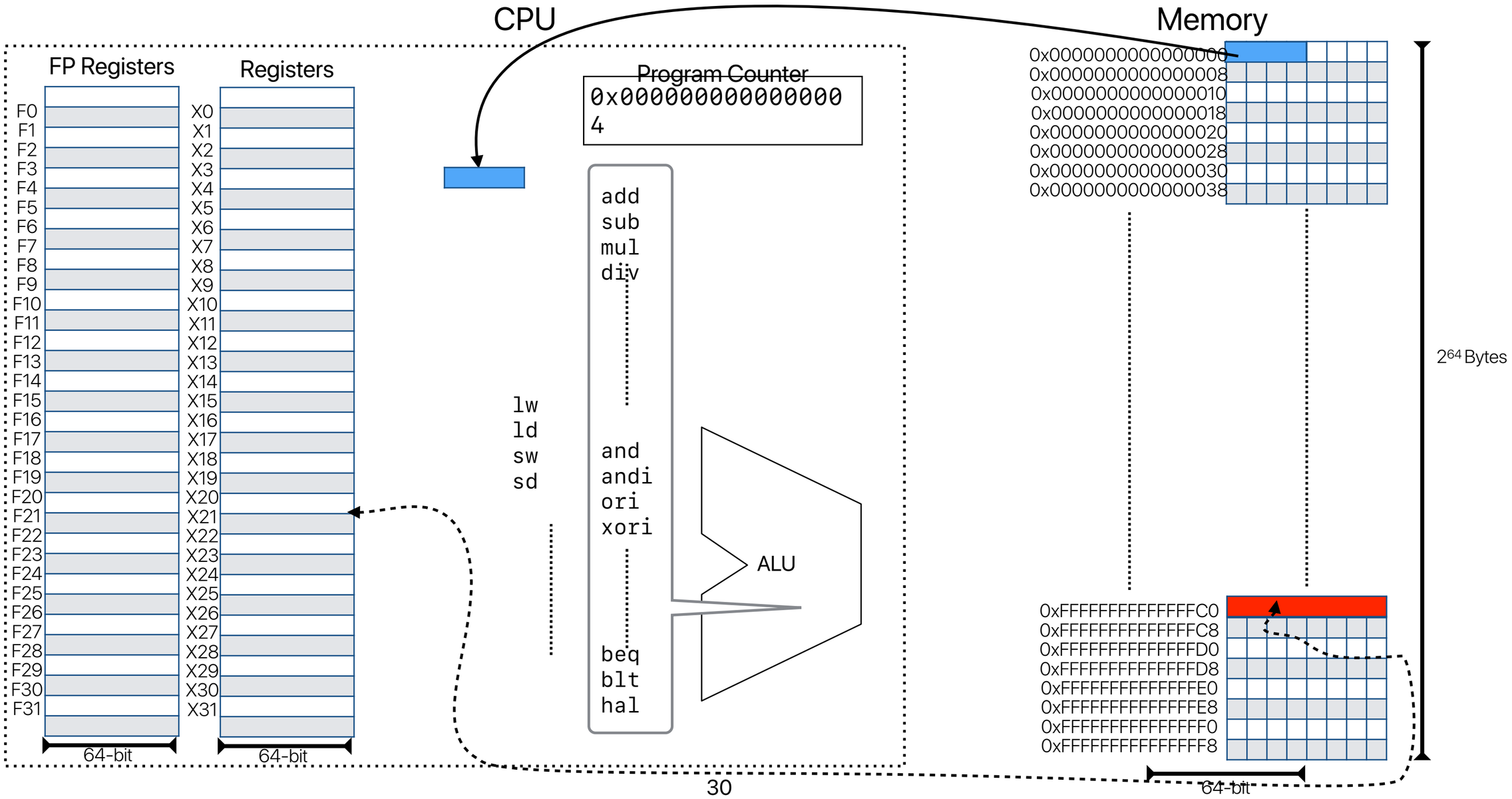
x86

arm



RISC-V

The abstracted “RISC-V” machine

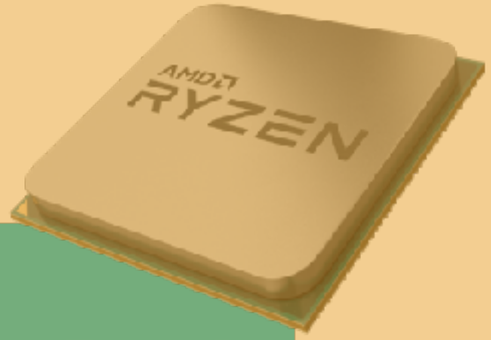


Subset of RISC-V instructions

Category	Instruction	Usage	Meaning
Arithmetic	add	`add  x1, x2, x3`	`x1 = x2 + x3`
	addi	`addi x1,x2, 20`	`x1 = x2 + 20`
	sub	`sub  x1, x2, x3`	`x1 = x2 – x3`
Logical	and	`and  x1, x2, x3`	`x1 = x2 & x3`
	or	`or   x1, x2, x3`	`x1 = x2 | x3`
	andi	`andi x1, x2, 20`	`x1 = x2 & 20`
	sll	`sll  x1, x2, 10`	`x1 = x2 * 2^10`
	srl	`srl  x1, x2, 10`	`x1 = x2 / 2^10`
Data Transfer	ld	`ld   x1, 8(x2)`	`x1 = mem[x2+8]`
	sd	`sd   x1, 8(x2)`	`mem[x2+8] = x1`
Branch	beq	`beq  x1, x2, 25`	`if(x1 == x2), PC = PC + 100`
	bne	`bne  x1, x2, 25`	`if(x1 != x2), PC = PC + 100`
Jump	jal	`jal  25`	`$ra = PC + 4, PC = 100`
	jr	`jr   $ra`	`PC = $ra`

The only type of instructions can access memory

Popular ISAs

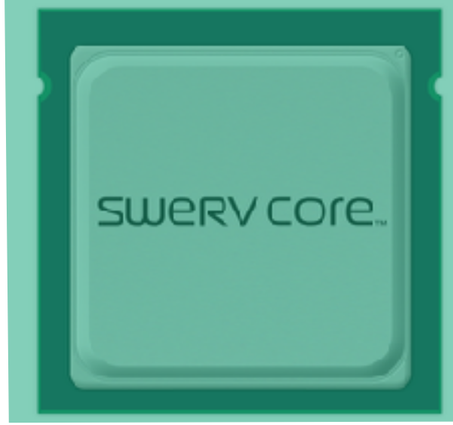


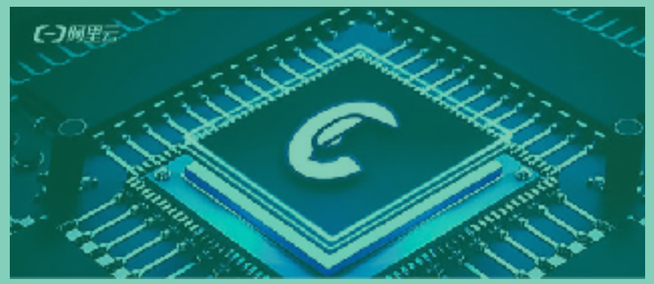
x86

Complex Instruction Set Computers (CISC)

arm

Reduced Instruction Set Computers (RISC)



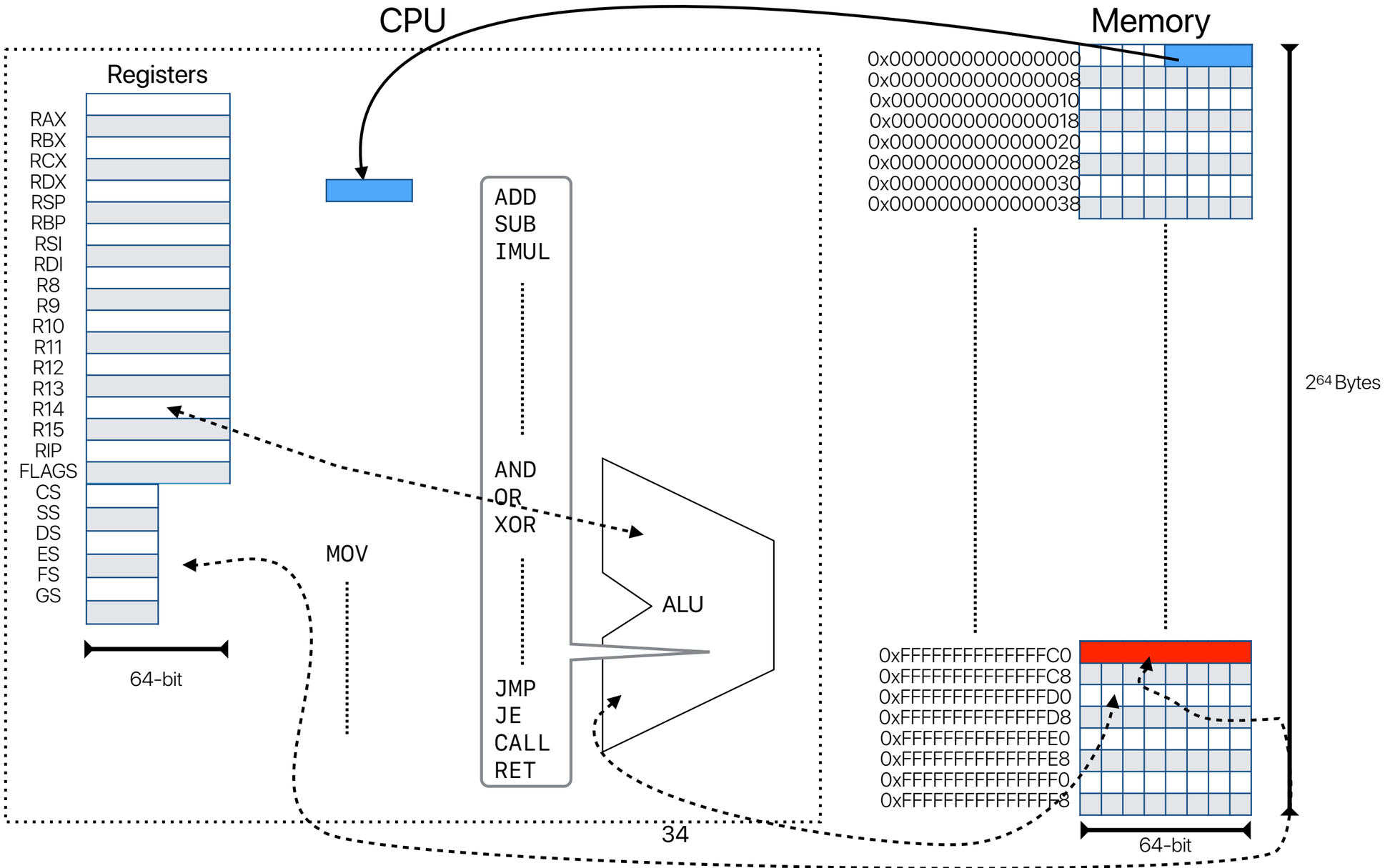


RISC-V

How many operations: CISC v.s. RISC

- CISC (Complex Instruction Set Computing)
  - Examples: x86, Motorola 68K
  - Provide **many powerful/complex** instructions
    - Many: more than 1503 instructions since 2016
    - Powerful/complex: an instruction can perform both ALU and memory operations
    - Each instruction takes more cycles to execute
- RISC (Reduced Instruction Set Computer)
  - Examples: ARMv8, RISC-V, MIPS (the first RISC instruction, invented by the authors of our textbook)
  - Each instruction only performs simple tasks
  - Easy to decode
  - Each instruction takes less cycles to execute

The abstracted x86 machine



RISC-V v.s. x86

	RISC-V	x86
ISA type	Reduced Instruction Set Computers (RISC)	Complex Instruction Set Computers (CISC)
instruction width	32 bits	1 ~ 17 bytes
code size	larger	smaller
registers	32	16
addressing modes	reg+offset	base+offset base+index scaled+index scaled+index+offset
hardware	simple	complex

RISC-V v.s. x86

- Using the same language, the same source code, regarding the compiled program on x86 and RISC-V, how many of the following statements is/are "generally" correct?
  - ① The RISC-V version would contain more instructions than its x86 version
  - ② The RISC-V version tends to incur fewer memory accesses than its x86 version
  - ③ The RISC-V version needs a processor with higher clock rate than its x86 version if the CPI of both versions are similar
  - ④ The RISC-V version needs a processor with lower CPI than its x86 version if the x86 processor runs at the same clock rate

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

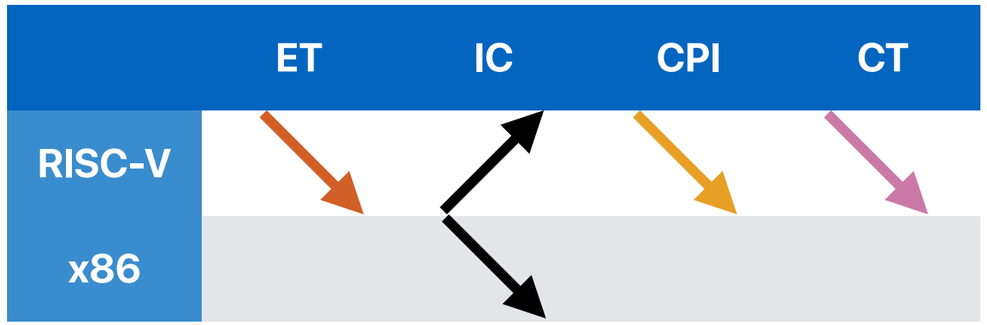
RISC-V v.s. x86

- Using the same language, the same source code, regarding the compiled program on x86 and RISC-V, how many of the following statements is/are "generally" correct?
  - The RISC-V version would contain more instructions than its x86 version
  - The RISC-V version tends to incur fewer memory accesses than its x86 version
  - The RISC-V version needs a processor with higher clock rate than its x86 version if the CPI of both versions are similar
  - The RISC-V version needs a processor with lower CPI than its x86 version if the x86 processor runs at the same clock rate

RISC-V has more registers!

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

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



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

Summary of 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**

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

H&P Chapter 1.9

M. D. Hill and M. R. Marty. Amdahl's Law in the Multicore Era. In Computer, vol. 41, no. 7, pp. 33-38, July 2008.

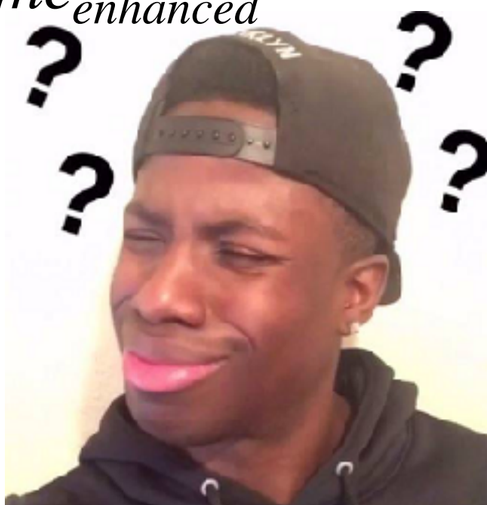
Amdahl's Law

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

f — The fraction of time in the original program

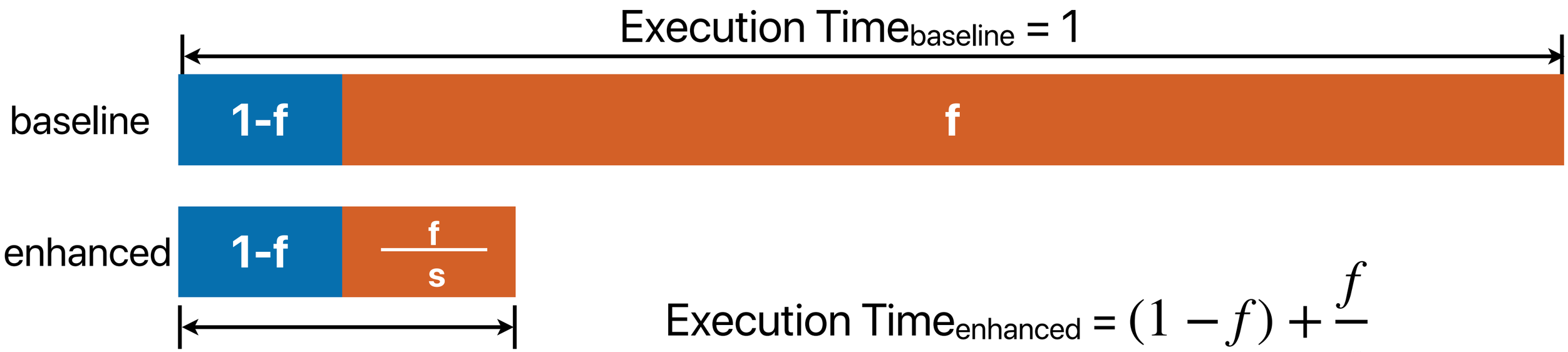
s — The speedup we can achieve on f

$$Speedup_{enhanced} = \frac{Execution\ Time_{baseline}}{Execution\ Time_{enhanced}}$$



Amdahl's Law

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



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

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

Recap: Speedup

- Assume that we have an application composed with a total of 5000000000 instructions, in which 20% of them are the load/store instructions with an average CPI of 6 cycles, and the rest instructions are integer instructions with average CPI of 1 cycle when using a 2GHz processor.
  - If we double the CPU clock rate to 4GHz that helps to accelerate all instructions by 2x except that load/store instruction cannot be improved — their CPI will become 12 cycles. What's the performance improvement after this change?

A. No change
B. 1.25
C. 1.5
D. 2
E. None of the above

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

$$ET_{baseline} = (5 \times 10^9) \times (20\% \times 6 + 80\% \times 1) \times \frac{1}{2 \times 10^9} sec = 5\ sec$$

$$ET_{enhanced} = (5 \times 10^9) \times (20\% \times 12 + 80\% \times 1) \times \frac{1}{4 \times 10^9} sec = 4\ sec$$

$$Speedup = \frac{Execution\ Time_{baseline}}{Execution\ Time_{enhanced}}$$

$$= \frac{5}{4} = 1.25$$

Replay using Amdahl's Law

- Assume that we have an application composed with a total of 5000000000 instructions, in which 20% of them are the load/store instructions with an average CPI of 6 cycles, and the rest instructions are integer instructions with average CPI of 1 cycle when using a 2GHz processor.
  - If we double the CPU clock rate to 4GHz that helps to accelerate all instructions by 2x except that load/store instruction cannot be improved — their CPI will become 12 cycles. What's the performance improvement after this change?

How much time in load/store? $5 \times 10^9 \times (0.2 \times 6) \times 0.5\ ns = 3\ s \rightarrow 60\,\%$

How much time in the rest? $5 \times 10^9 \times (0.8 \times 1) \times 0.5\ ns = 2\ s \rightarrow 40\,\%$

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

$$Speedup_{enhanced}(40\,\%, 2) = \frac{1}{(1-40\%) + \frac{40\,\%}{2}} = 1.25 \times$$

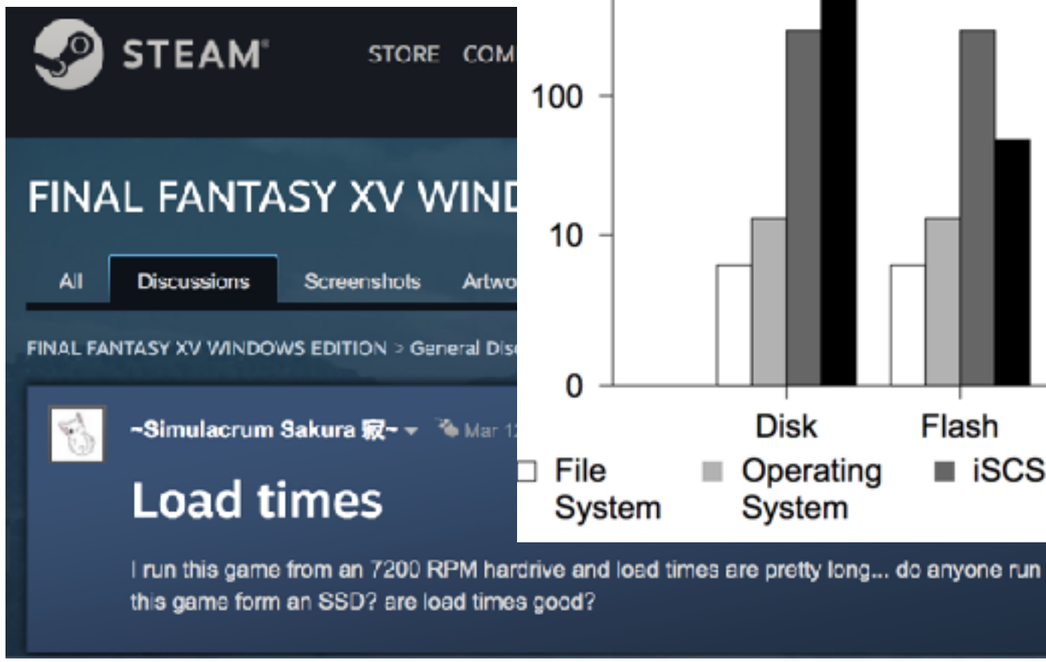
Practicing Amdahl's Law

- Final Fantasy XV spends lots of time loading a map — within which period that 95% of the time on the accessing the H.D.D., the rest in the operating system, file system and the I/O protocol. If we replace the H.D.D. with a flash drive, which provides 100x faster access time. By how much can we speed up the map loading process?
  A. ~7x
  B. ~10x
  C. ~17x
  D. ~29x
  E. ~100x

Practicing Amdahl's Law

- Final Fantasy XV spends lots of time loading a map — within which period that 95% of the time on the accessing the H.D.D., the rest in the operating system, file system and the I/O protocol. If we replace the H.D.D. with a flash drive, which provides 100x faster access time. By how much can we speed up the map loading process?

  A. ~7x
  B. ~10x
  C. ~17x
  D. ~29x
  E. ~100x

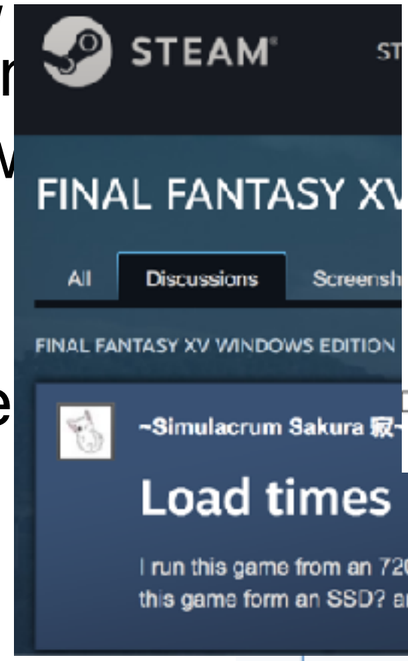


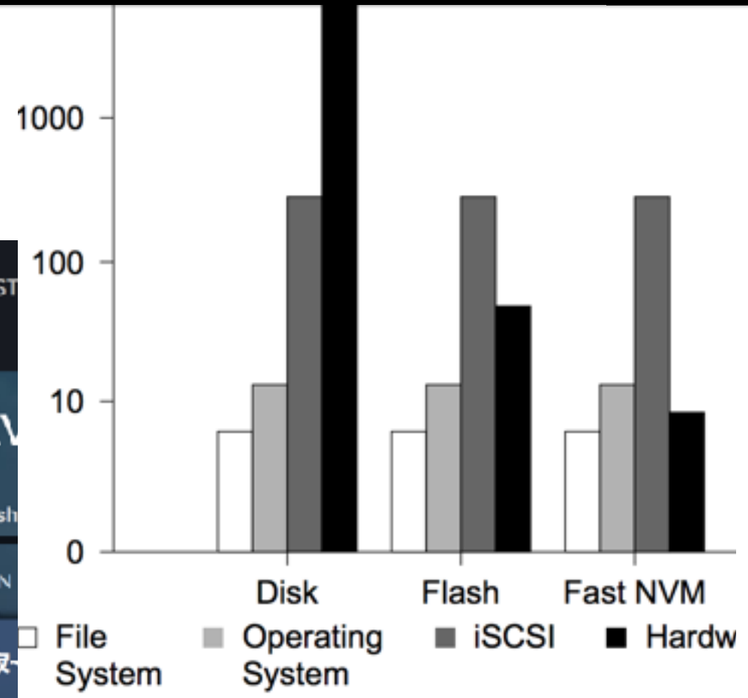
$$Speedup_{enhanced}(95\,\%, 100) = \frac{1}{(1 - 95\%) + \frac{95\,\%}{100}} = 16.81 \times$$

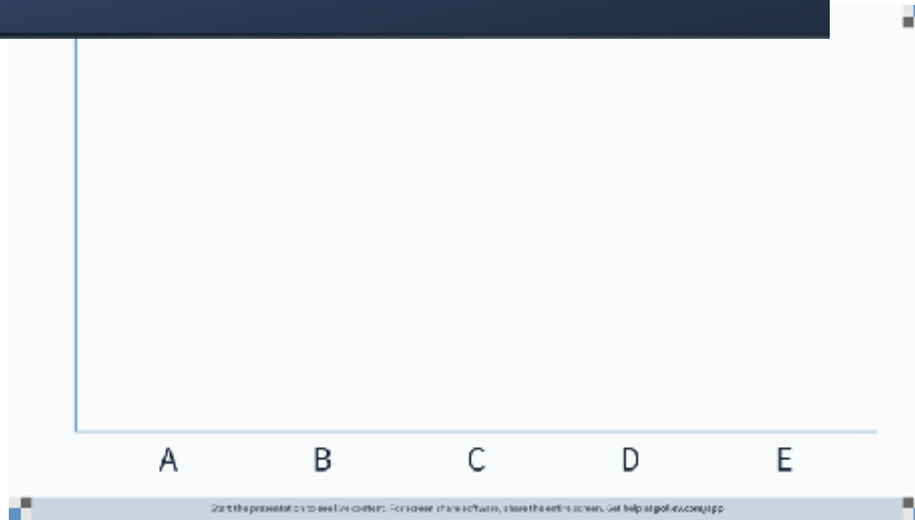
https://www.pollev.com/hungweitseng close in 1:30

Speedup further!

- With the latest flash memory technologies, the system spends 16% of time on accessir the flash, and the software overhead is now 84%. If we want to adopt a new memory technology to replace flash to achieve 2x speedup on loading maps, how much faste the new technology needs to be?
  - A. ~5x
  - B. ~10x
  - C. ~20x
  - D. ~100x
  - E. None of the above



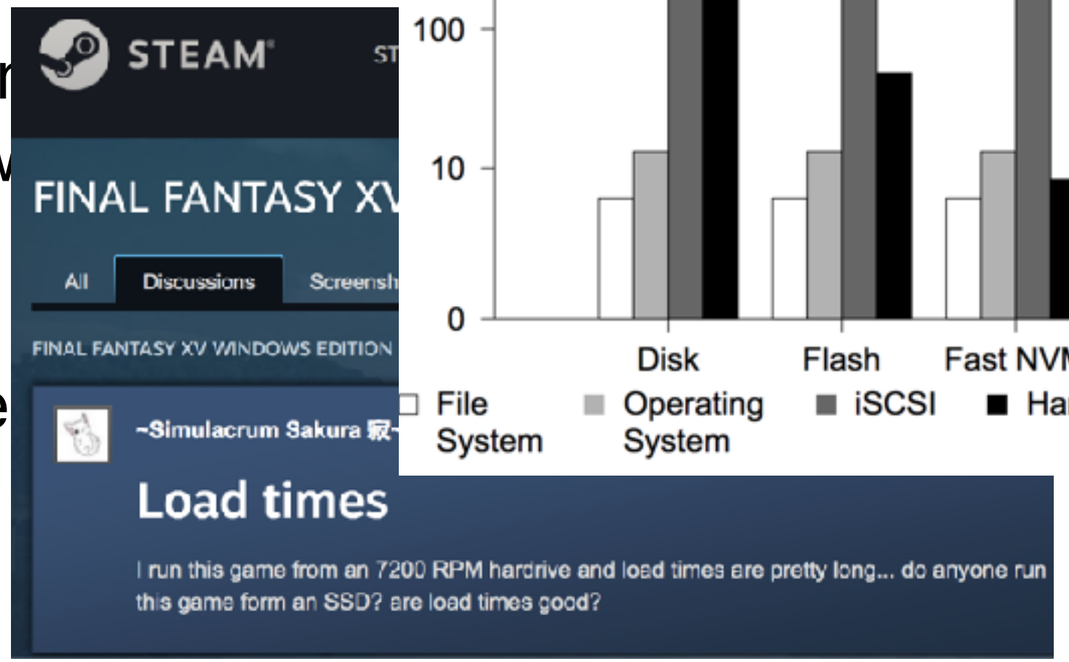




Speedup further!

- With the latest flash memory technologies, the system spends 16% of time on accessing the flash, and the software overhead is now 84%. If we want to adopt a new memory technology to replace flash to achieve 2x speedup on loading maps, how much faster the new technology needs to be?

A. ~5x
B. ~10x
C. ~20x
D. ~100x
E. None of the above



$$Speedup_{enhanced}(16\%, x) = \frac{1}{(1-16\%) + \frac{16\%}{x}} = 2$$

$$x = 0.47$$

Does this make sense?

Amdahl's Law Corollary #1

- The maximum speedup is bounded by

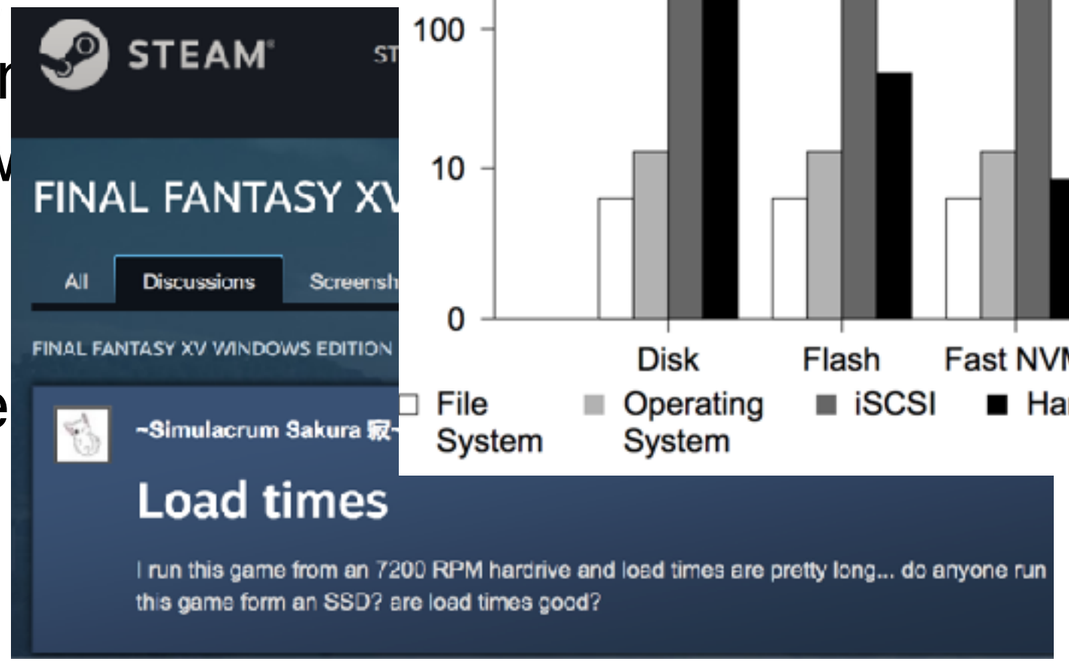
$$Speedup_{max}(f, \infty) = \frac{1}{(1-f) + \frac{f}{\infty}}$$

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

Speedup further!

- With the latest flash memory technologies, the system spends 16% of time on accessing the flash, and the software overhead is now 84%. If we want to adopt a new memory technology to replace flash to achieve 2x speedup on loading maps, how much faster the new technology needs to be?

  A. ~5x
  B. ~10x
  C. ~20x
  D. ~100x
  E. None of the above



$$Speedup_{max}(16\,\%\,,\infty) = \frac{1}{(1-16\%)} = 1.19$$

2x is not possible

Announcement

- Reading quiz due next Wednesday before the lecture
  - We will drop two of your least performing reading quizzes
  - You have two shots, both unlimited time
  - No make-ups
- Assignment #1 released
  - We typically give you two weeks to work on an assignment
  - Due on 10/14
- Assignment #0 due on 10/7
- Check our website for slides, eLearn for quizzes/assignments, piazza for discussions
- Youtube channel for lecture recordings: https://www.youtube.com/c/ProfUsagi/playlists

Computer
Science &
Engineering

203

つづく