



CS773-2022-Autumn: Computer Architecture for Performance and Security

Lecture 2: What is the CATCH? Microarchitecture

Remembering Tushar



For all the hard work in online and offline infrastructure facilities that we have, including classrooms, space development and maintenance, and data center development and maintenance



ON SILENT MODE PLEASE

Phones on
silence, please

Thank You

The background of the slide is a dense, chaotic web of thin, tangled lines in various shades of pink, magenta, and orange. The lines are of varying lengths and orientations, creating a complex, almost abstract pattern that fills the entire frame. The overall effect is one of intense energy and complexity.

Let's know all@CS773



Before I start
Carpe Diem!

The background of the slide is a dense, chaotic web of thin, tangled lines in various shades of pink, magenta, and orange. The lines are of varying lengths and orientations, creating a complex, almost abstract pattern that fills the entire frame. The colors are vibrant and saturated, with some lines appearing darker or more saturated than others, giving the background a sense of depth and movement.

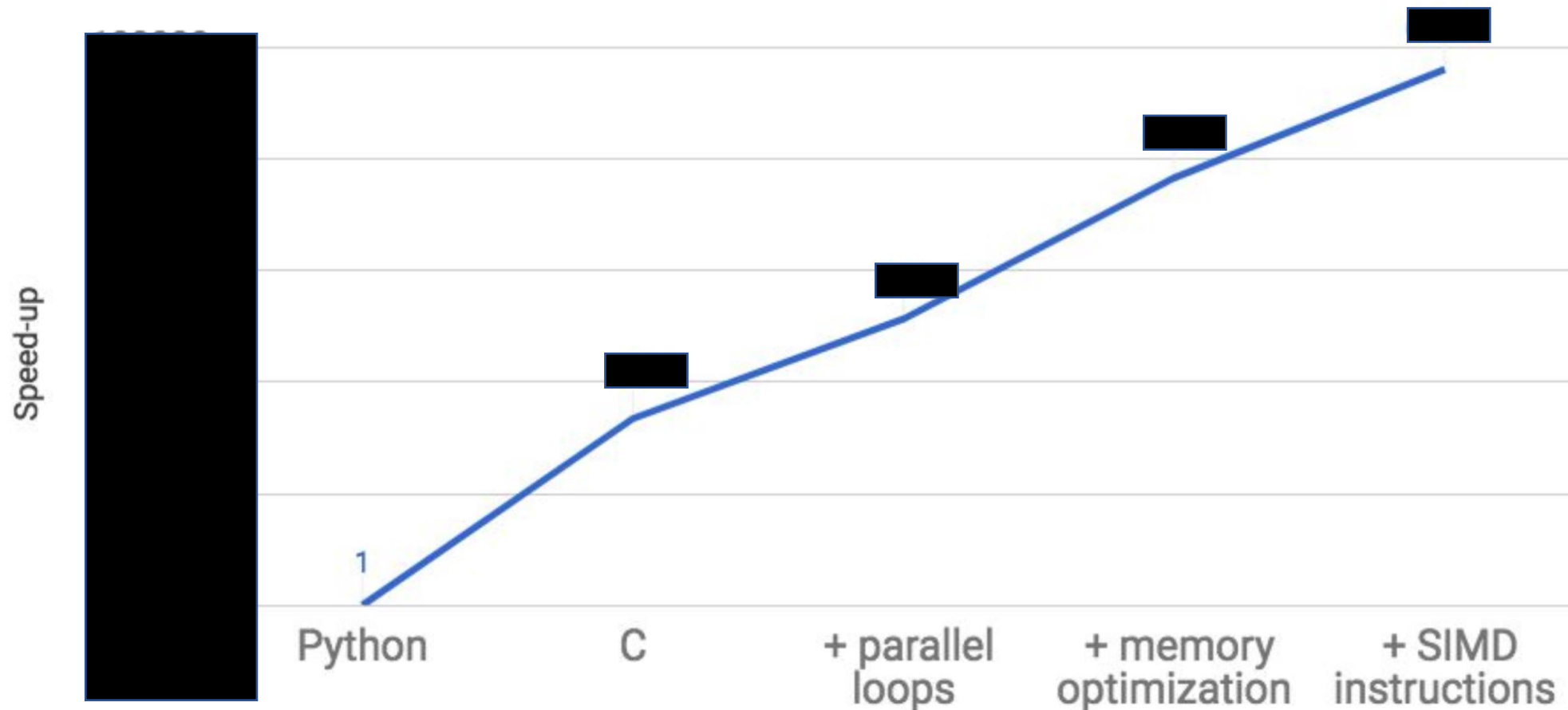
Abstraction Time to see a real chip

The background of the slide is a dense, chaotic web of thin, tangled lines in various shades of pink, magenta, and red. The lines are of varying lengths and orientations, creating a complex, abstract pattern that fills the entire frame. The text is centered over this background.

Abstraction Good or bad for CS773?

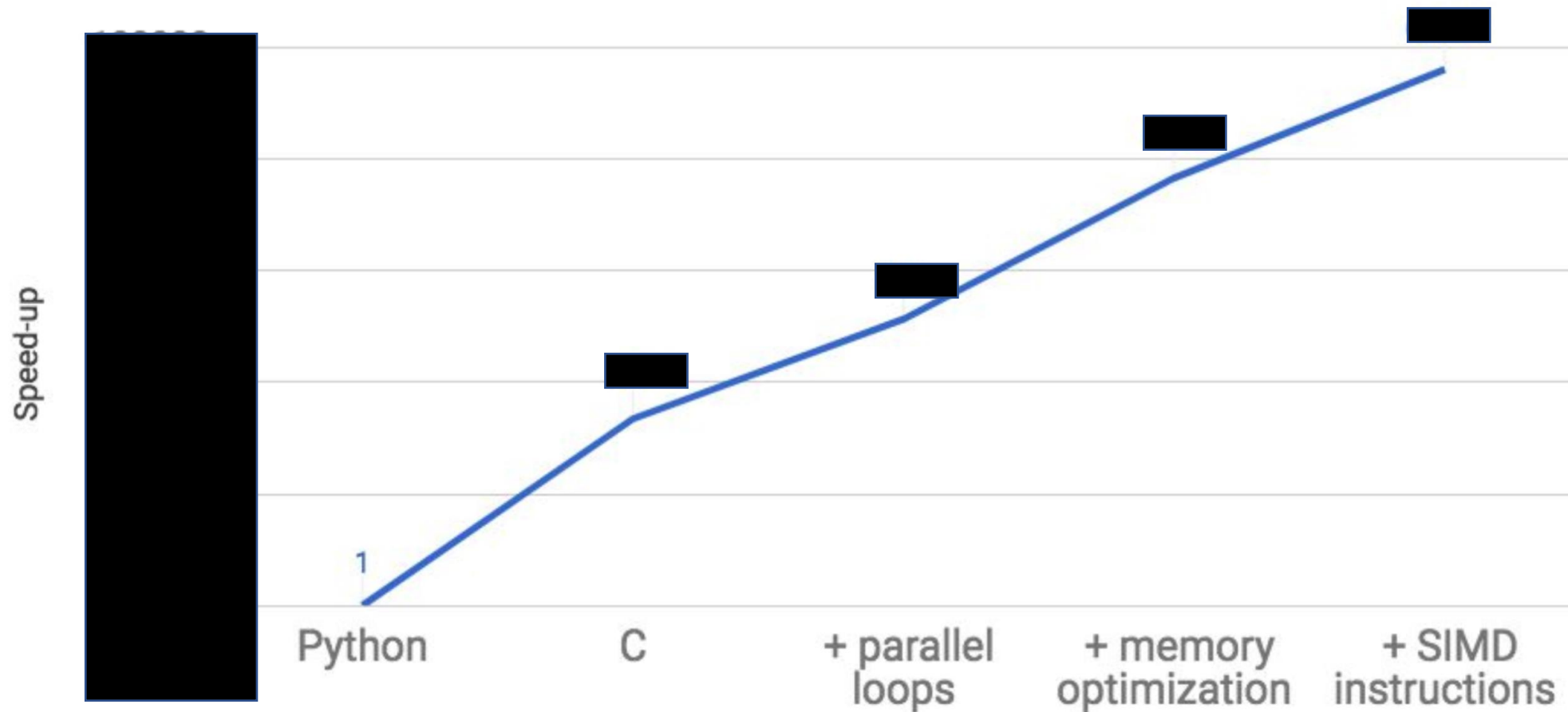
Abstraction is the enemy ☹️

Matrix Multiply Speedup Over Native Python



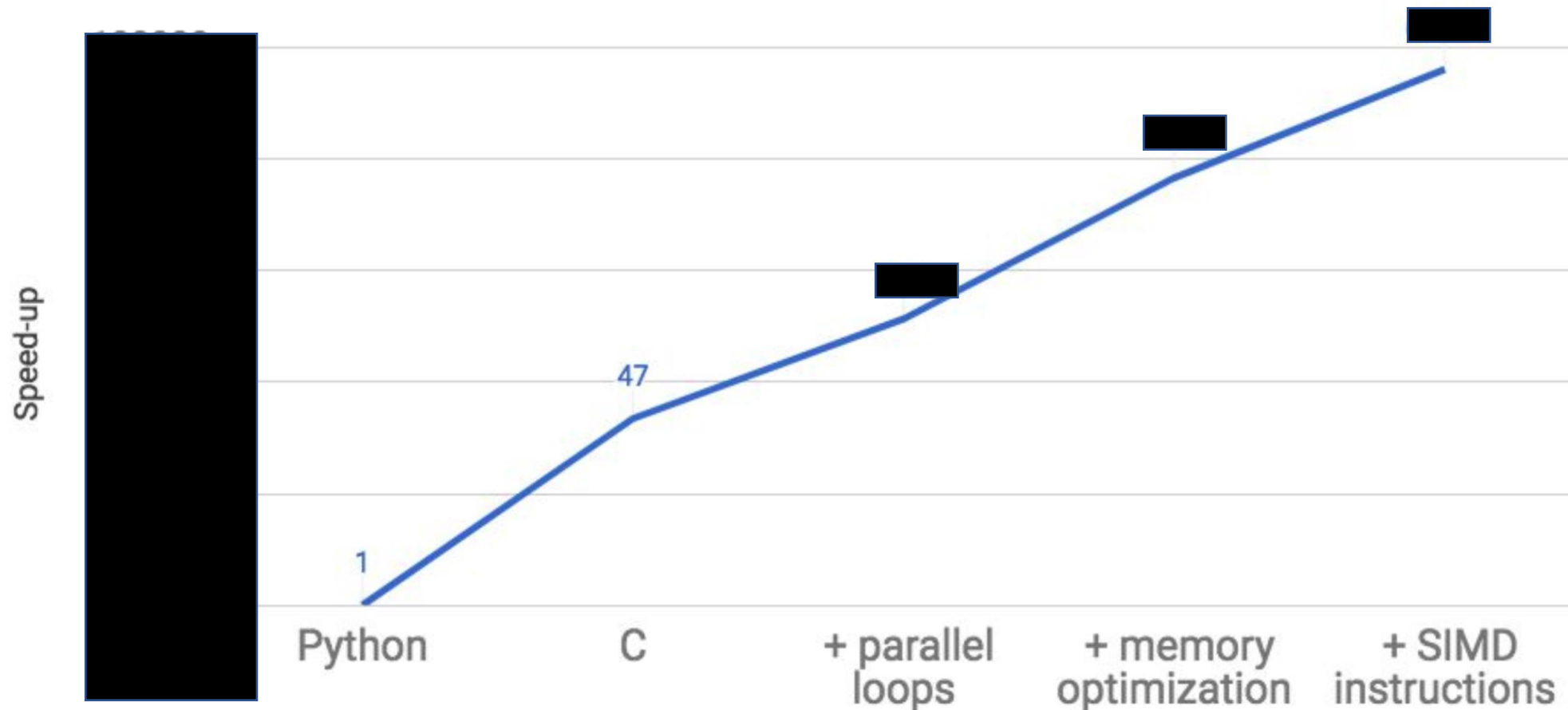
OK. I am ready

Matrix Multiply Speedup Over Native Python



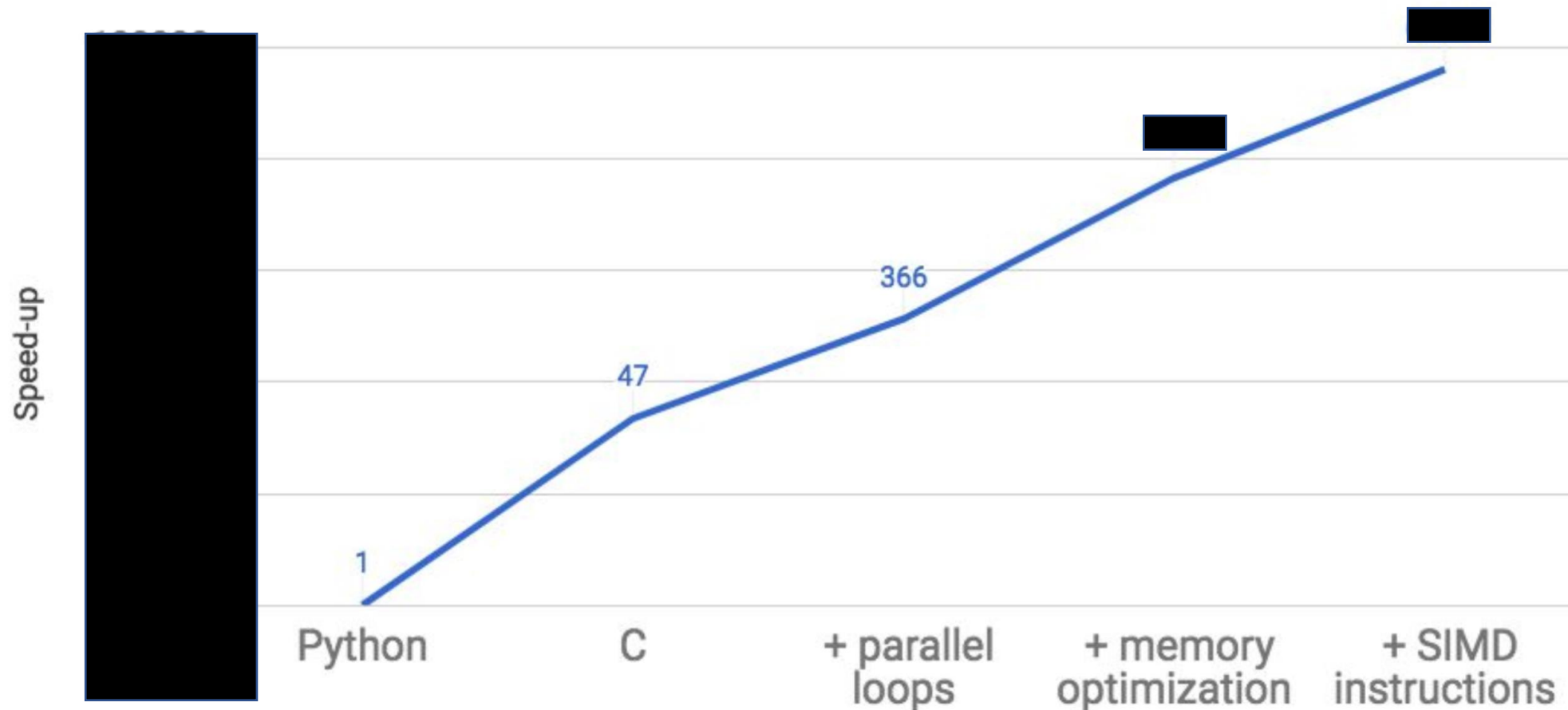
Seriously?

Matrix Multiply Speedup Over Native Python



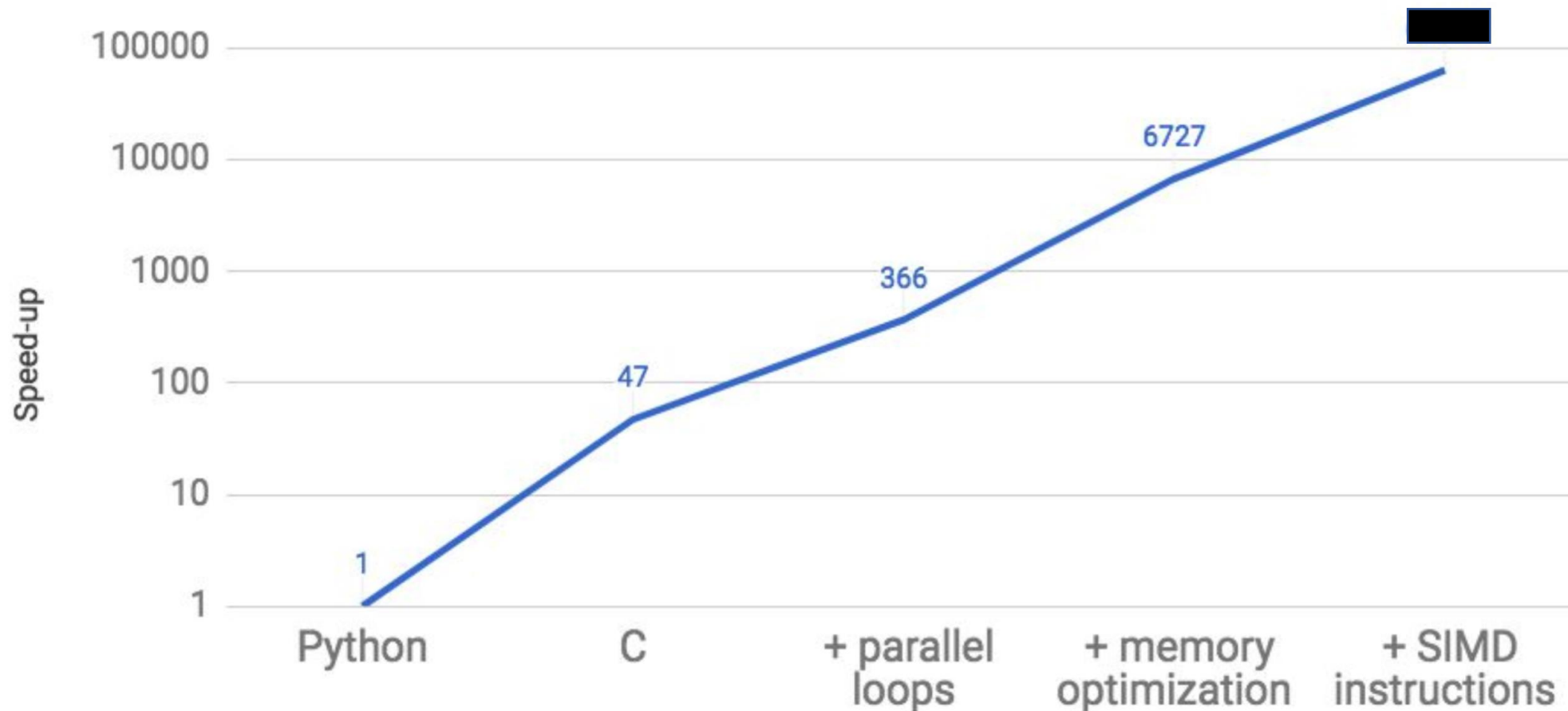
That's it. You can't do better. Period !!

Matrix Multiply Speedup Over Native Python



This is Insane

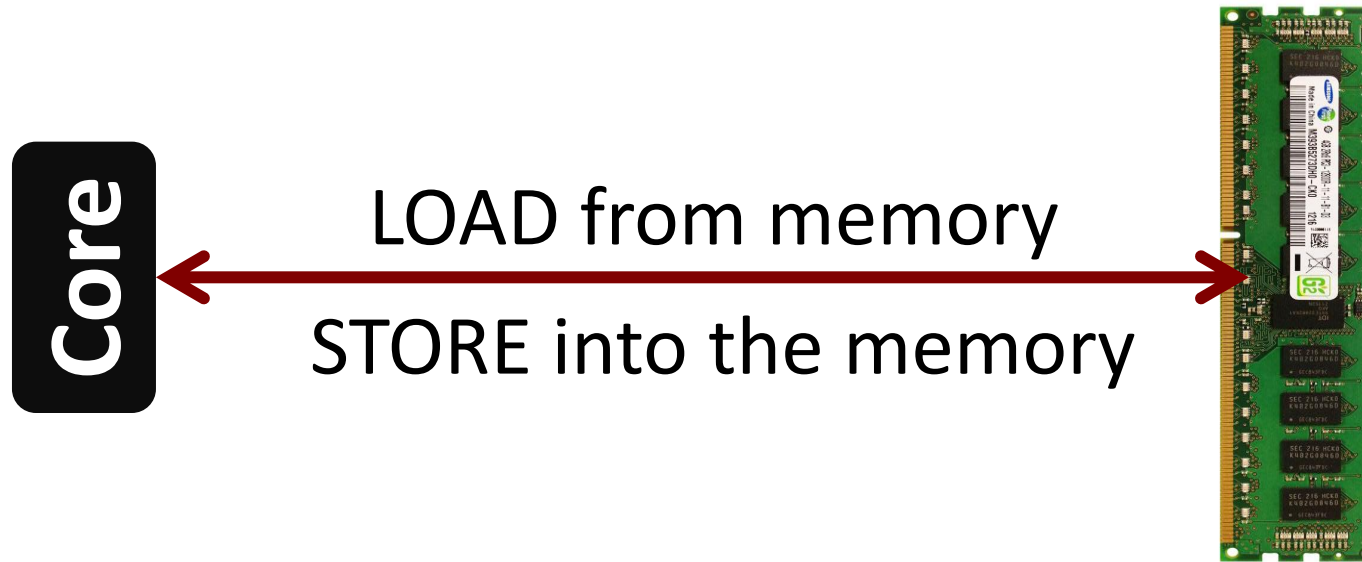
Matrix Multiply Speedup Over Native Python





62,806X

Let's get started: Stored Program World



How to access instructions: Program Counter (PC)

A register that stores the address of the instruction

32-bit processor: addresses are of width 32 bits

So the processor fetches PC, PC+4, PC+8, in a
sequential order

Both instructions and data from memory

$g = h + A[8];$

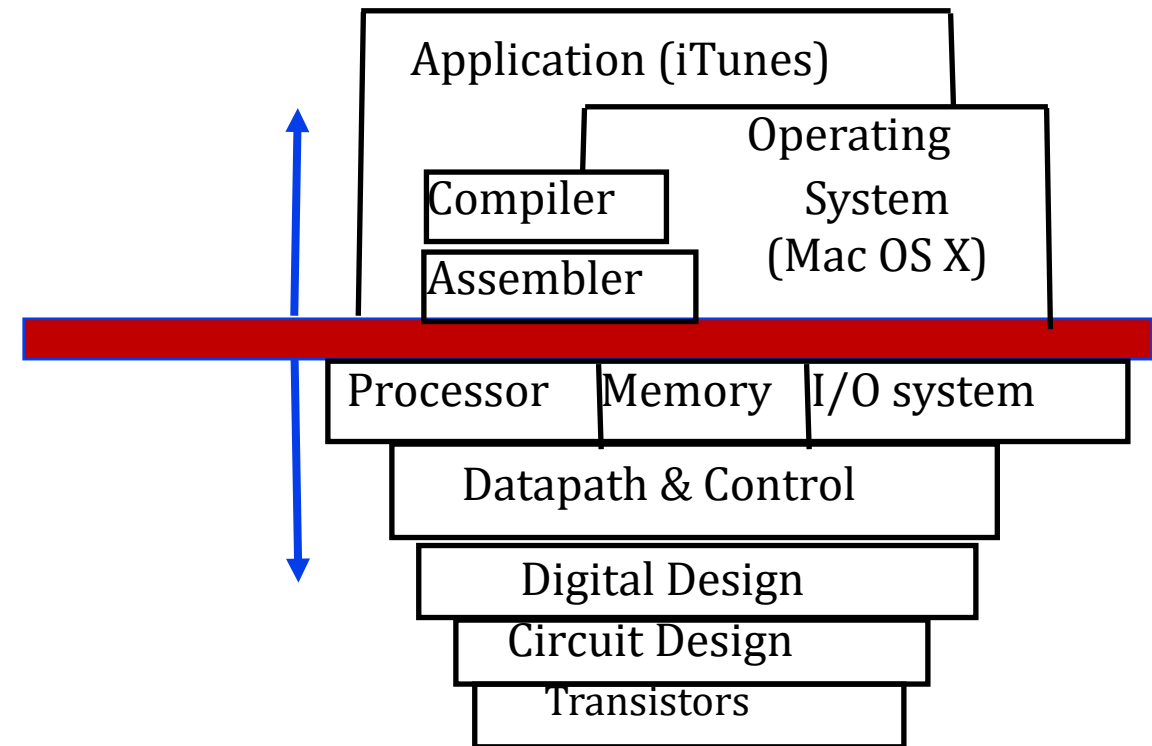
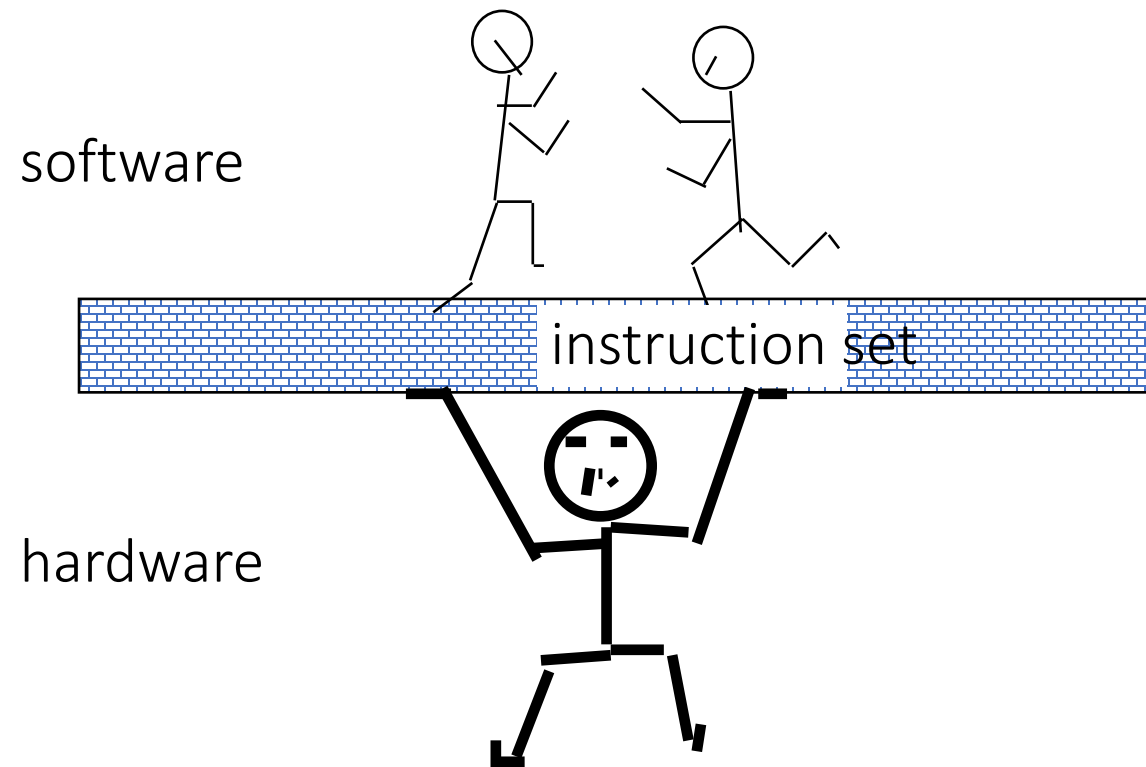
PCX: lw \$t0, 8(\$3) # A[8]

PCY: add \$s1, \$s2, \$t0 # $g = h + t0$

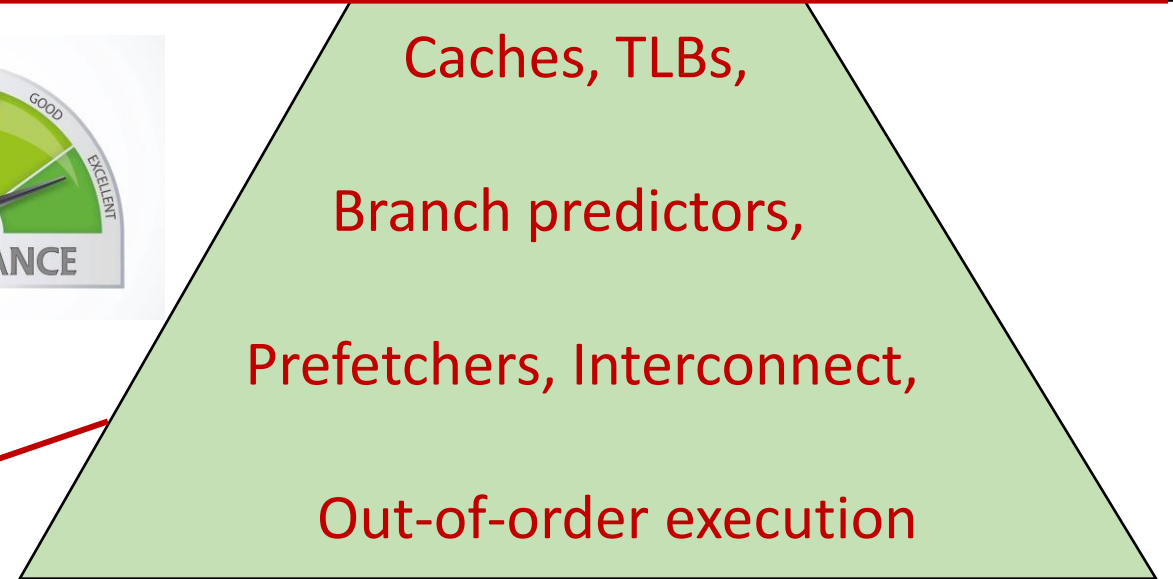
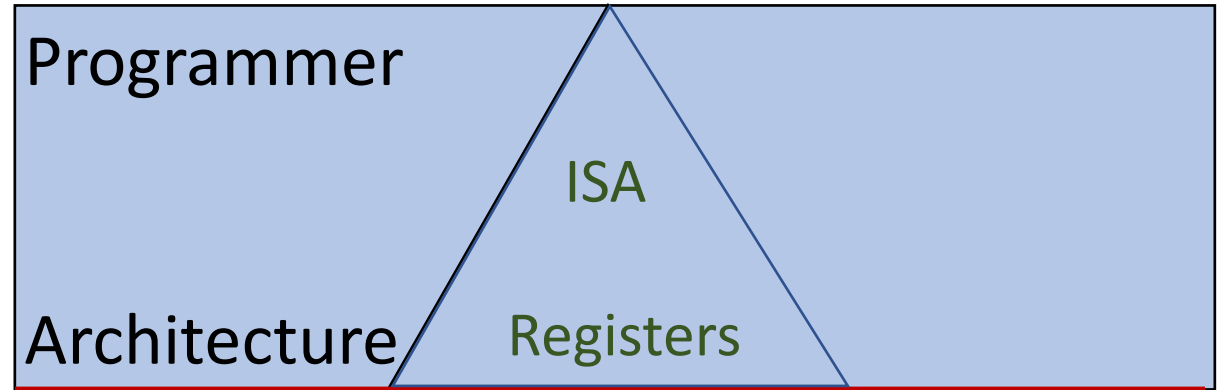
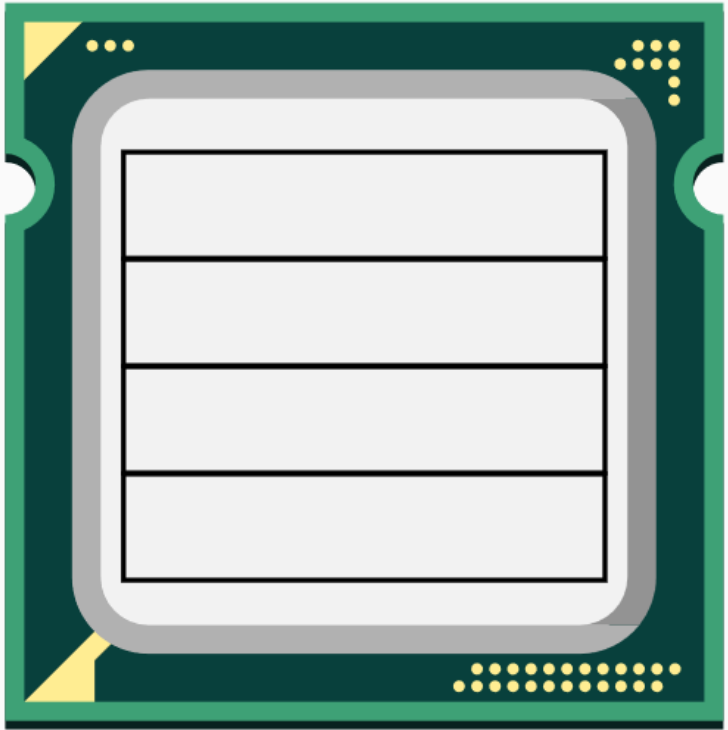
PCY = PCX+4



ISA



Let's dig deep



Not exposed to programmer

Memory

Discussion time!

Microarchitecture

Processor is in state S

Processor moves to state SS

What is what?

#registers:

#cycles to access a register:

#Width of the register (32/64 bit) :

#Instruction that uses register to access memory:

#cycles to access memory:

What is what?

#registers: **ISA**

#cycles to access a register: **Microarch.**

#Width of the register (32/64 bit) : **ISA**

#Instruction that uses register to access memory: **ISA**

#cycles to access memory: **Microarch.**

Where and what is the cost?

A billion-dollar idea 😊

- i) requires changes to microarchitecture.
- ii) requires changes to ISA.
- iii) both (i) and (ii)

Think about the trade-offs 😊

Does it affect the system stack?

The RISKY way

ISA: How to make it high performance and secure?

RISC: Reduced Instruction Set of Computers

Very few simple instructions

Example: MIPS

CISC: Complex Instruction Set of Computers

Lots of complicated instructions

Example: x86 kind of ☺ [x86 is CISC with RISC mysteries]

The RISKY way

Intel *converts* CISC ones into RISC ones and generate microoperations.

Intelligent CISC-RISC decoder

Summary

Conventional definition:

Architecture: ISA + Microarchitecture

ISA: Instruction set architecture, talks to compiler and OS, Example: x86

Microarchitecture: Not visible to programmer, compiler, and also OS upto 99%. Why 99%?

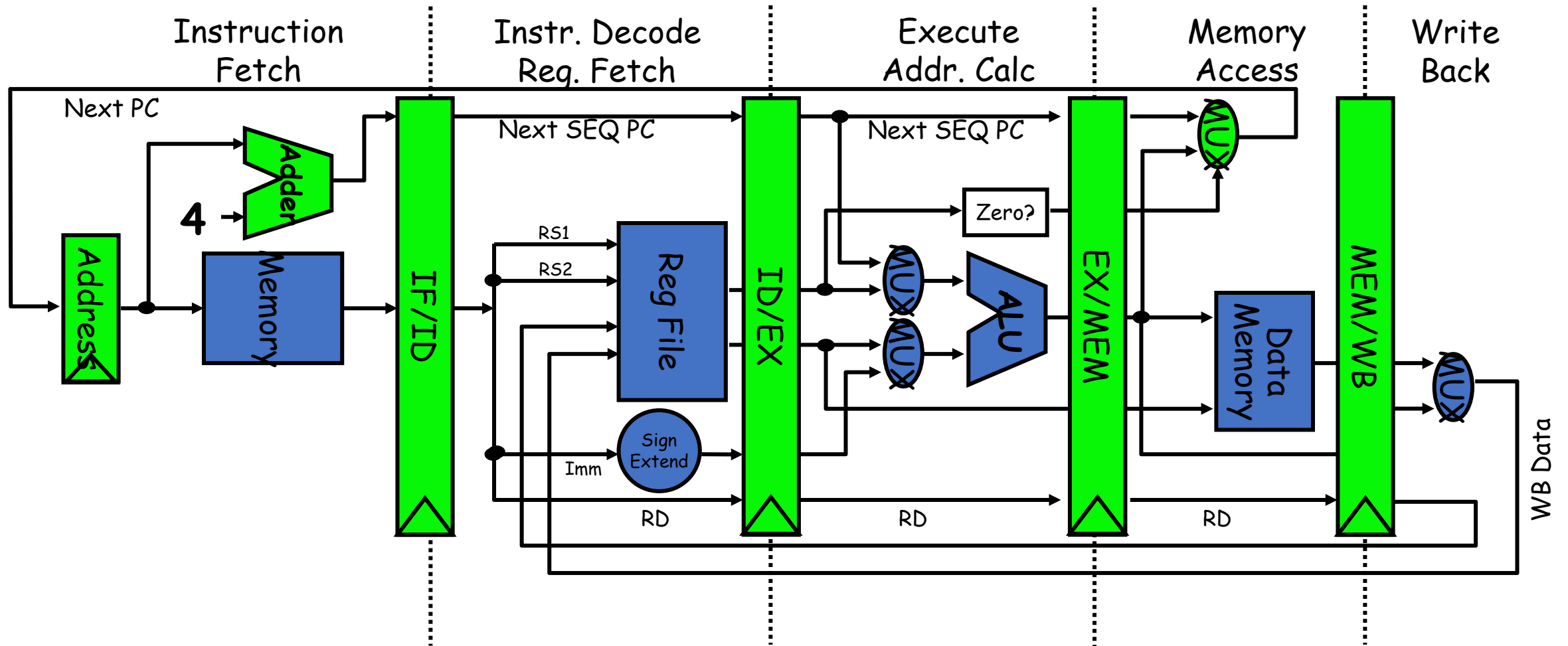


PAUSE

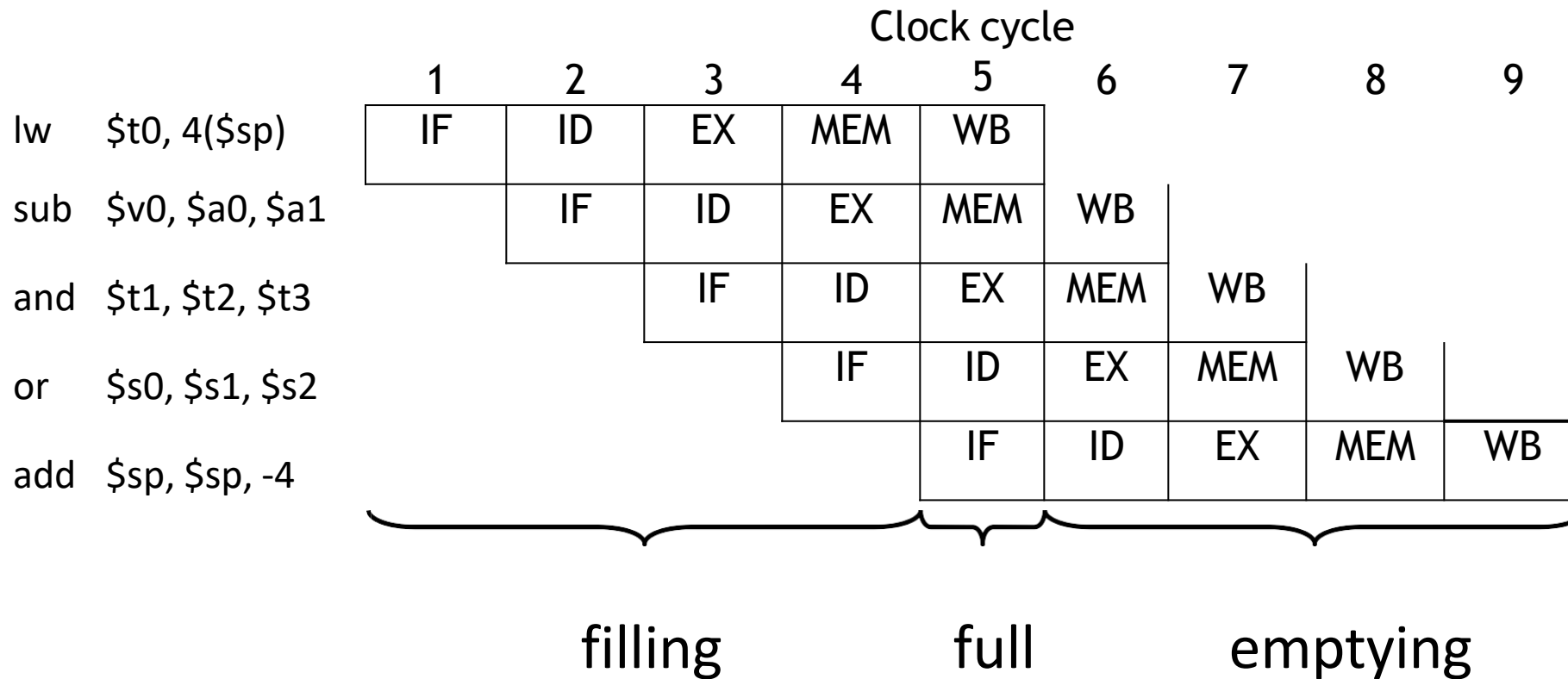
CASPER

26

Vanilla 5-stage pipeline



Visualizing Pipeline: Execution time




For a k-stage pipeline executing N instructions

first instruction: K cycles

Next N-1 instructions: N-1 cycles, total = K + (N-1) cycles

But...Data dependences (hazards)


```
add  R1, R2, R3
sub   R2, R4, R1
or    R1, R6, R3
```



read-after-write
(RAW)

True dependence


```
add  R1, R2, R3
sub   R2, R4, R1
or    R1, R6, R3
```



write-after-read
(WAR)

anti dependence

```
add  R1, R2, R3
sub   R2, R4, R1
or    R1, R6, R3
```

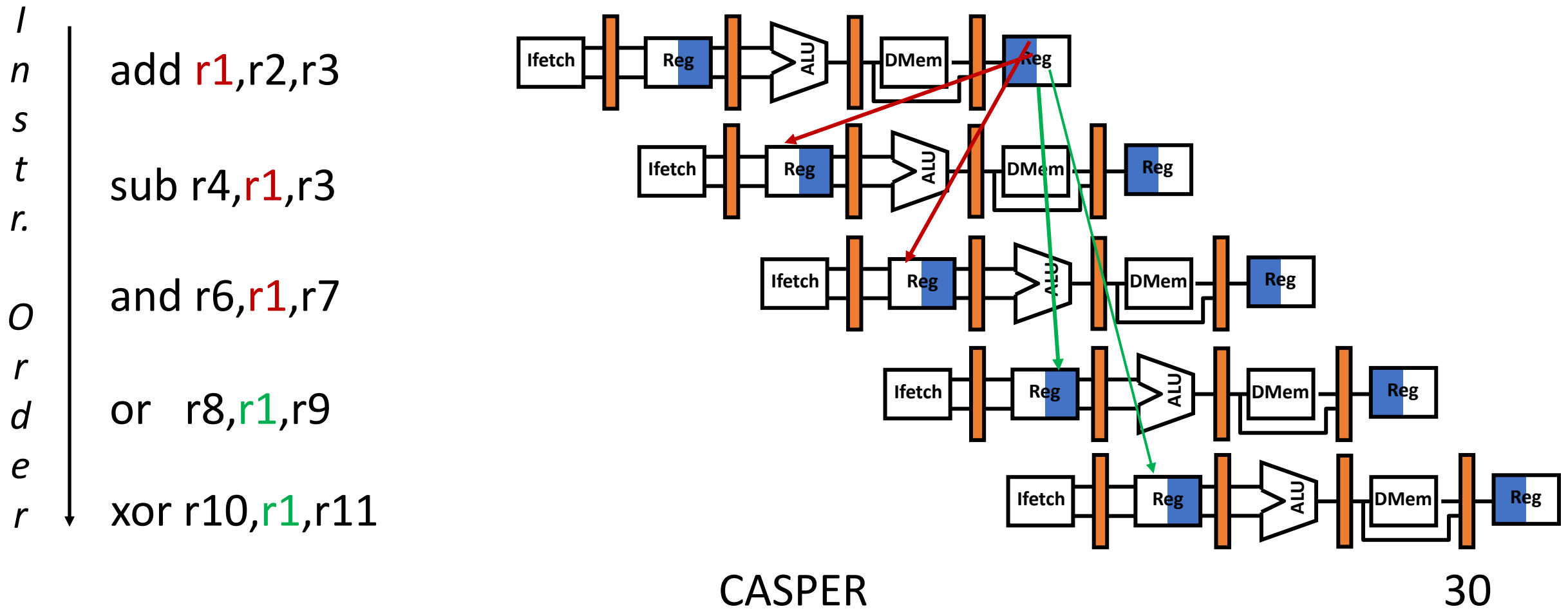


write-after-write
(WAW)

output dependence

Data Hazards (Examples)

Time (clock cycles)



Control Hazard

10: beq r1,r3,36

14: and r2,r3,r5 ☹️

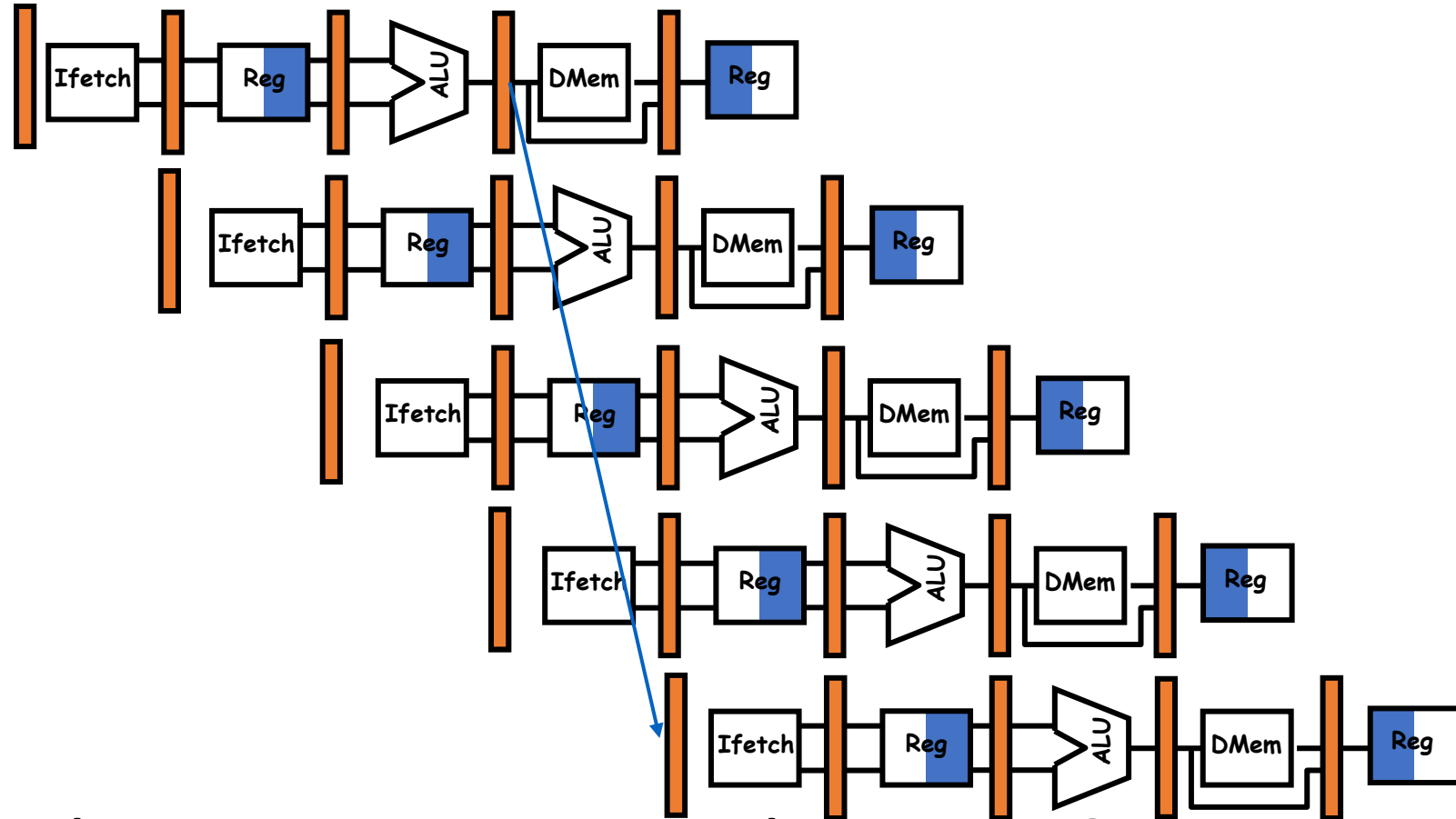
18: or r6,r1,r7 ☹️

22: add r8,r1,r9 ☹️

50: xor r10,r1,r11

What do you do with the 3 instructions in between?

How do you do it?





PAUSE

CASPER

32



Next lecture: Bring your laptop