You came early,
We appreciate it
Until we get started, we would like you to do the following

1. Open Linux DSL user
2. If you don’t have a Github account
   1. Create one
3. Connect to the server and share your details
4. If you are still bored waiting,
   1. apply for Github student developer pack

#Pragma

Welcome

Who are we?

Who are you?

- Name
- Course
- Hobby

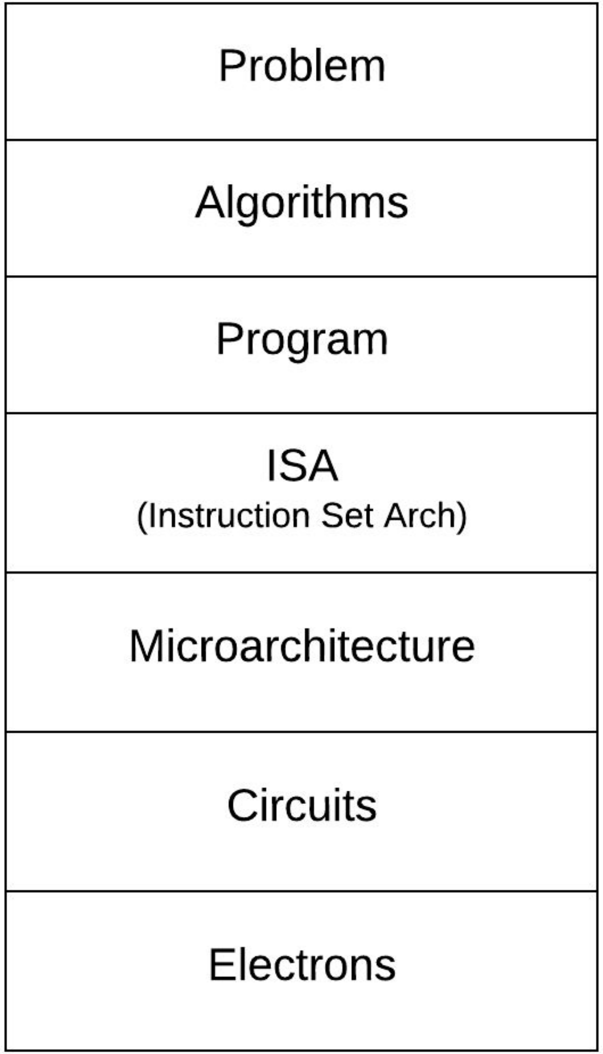
Why are you here?

Why ~~are you here?~~ do you want to give your 2 hours of precious time to us?

What we have in our mind?

Your Favorite Language

This is my favorite images of all times



Problem

Algorithms

Program

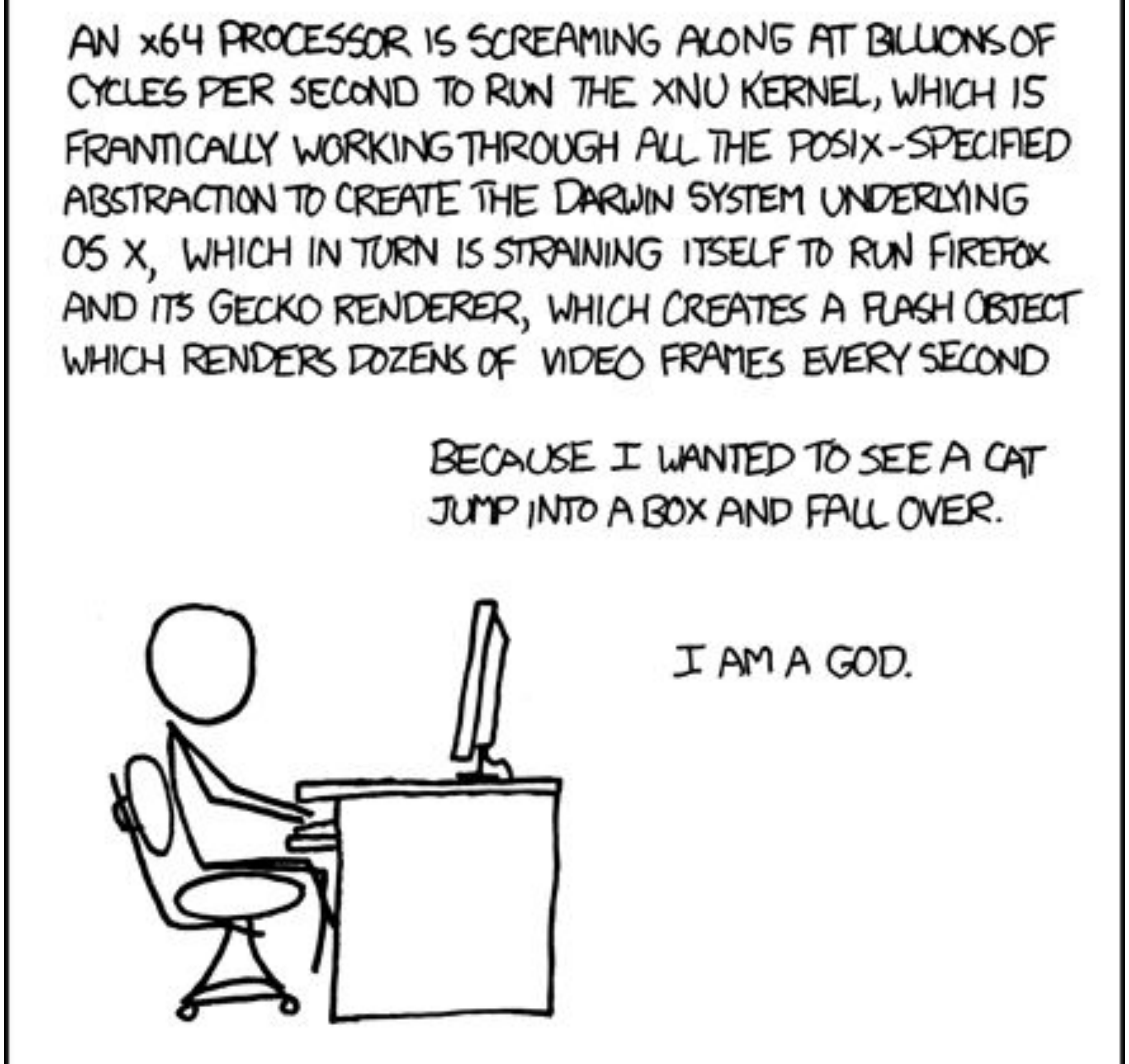
ISA
(Instruction Set Arch)

Microarchitecture

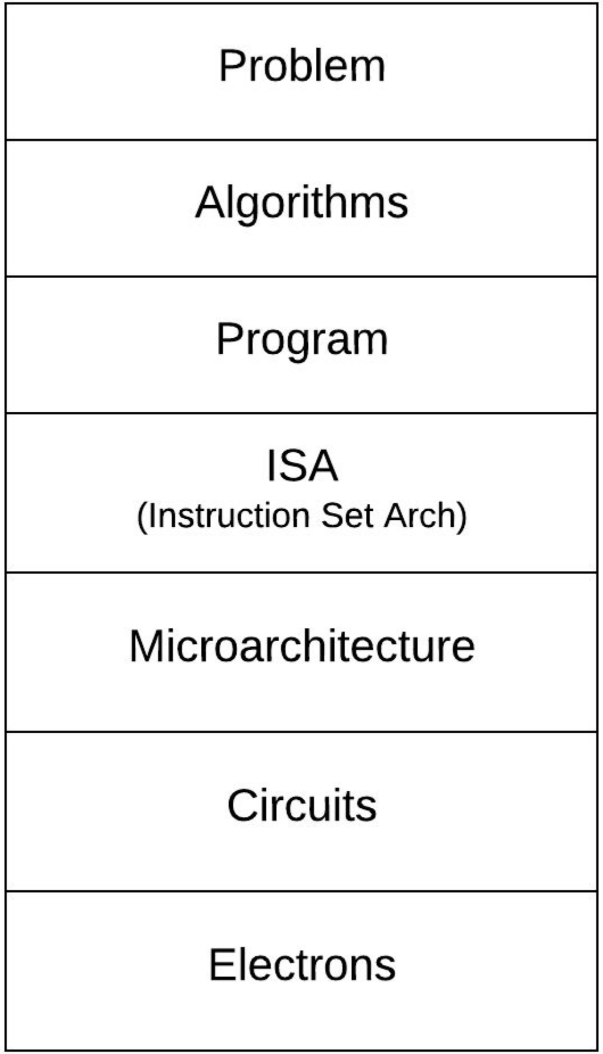
Circuits

Electrons

If you didn’t like that


AN x64 PROCESSOR IS SCREAMING ALONG AT BILLIONS OF CYCLES PER SECOND TO RUN THE XNU KERNEL, WHICH IS FRANTICALLY WORKING THROUGH ALL THE POSIX-SPECIFIED ABSTRACTION TO CREATE THE DARWIN SYSTEM UNDERLYING OS X, WHICH IN TURN IS STRAINING ITSELF TO RUN FIREFOX AND ITS GECKO RENDERER, WHICH CREATES A FLASH OBJECT WHICH RENDERS DOZENS OF VIDEO FRAMES EVERY SECOND
BECAUSE I WANTED TO SEE A CAT JUMP INTO A BOX AND FALL OVER.
I AM A GOD.

Let’s talk code



Problem

Algorithms

Program

ISA
(Instruction Set Arch)

Microarchitecture

Circuits

Electrons

THIS

```
for(int i = 0; i < n; i++){
    a[i] = a[i] + 1;
}
```

THIS

```
for(int i = 0; i < n; i++){
    a[i] = a[i] + 1;
}
```

THAT

```
for(int& i : a){
    i++;
}
```

Fun fact: What are Python Lists?

Fun fact: What are Python Lists?
It is just a dynamic array(think about it)

How many of you know Matrix Multiply?

$C = A \times B$

1	2	3
4	5	6
7	8	9

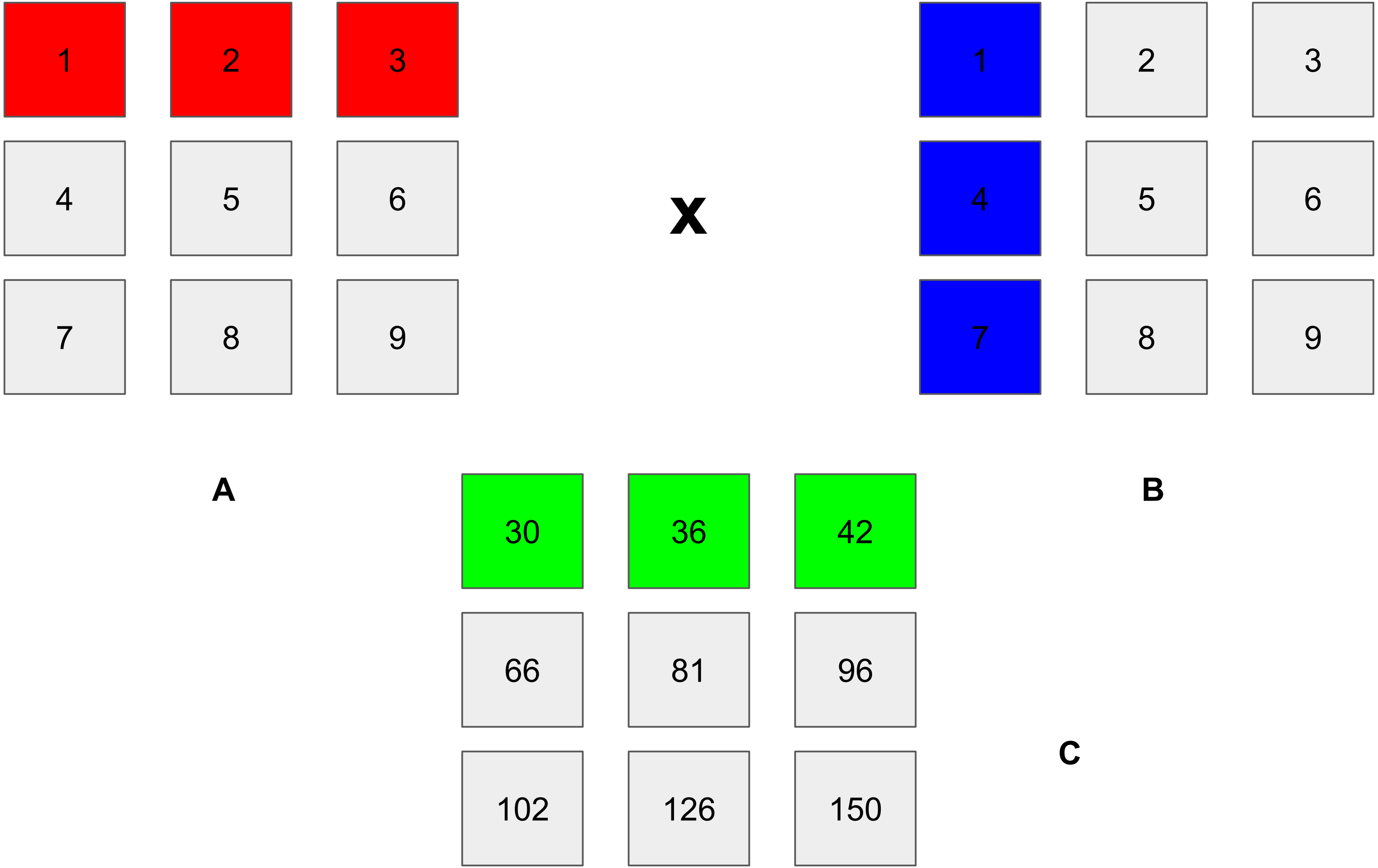
x

1	2	3
4	5	6
7	8	9

30	36	42
66	81	96
102	126	150

$$\underset{\mathbf{A}}{\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}} \quad \mathbf{x} \quad \underset{\mathbf{B}}{\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}}$$

$$\begin{bmatrix}30 & 36 & 42\\ 66 & 81 & 96\\ 102 & 126 & 150\end{bmatrix} \quad \mathbf{C}$$



1	2	3
4	5	6
7	8	9

A

x

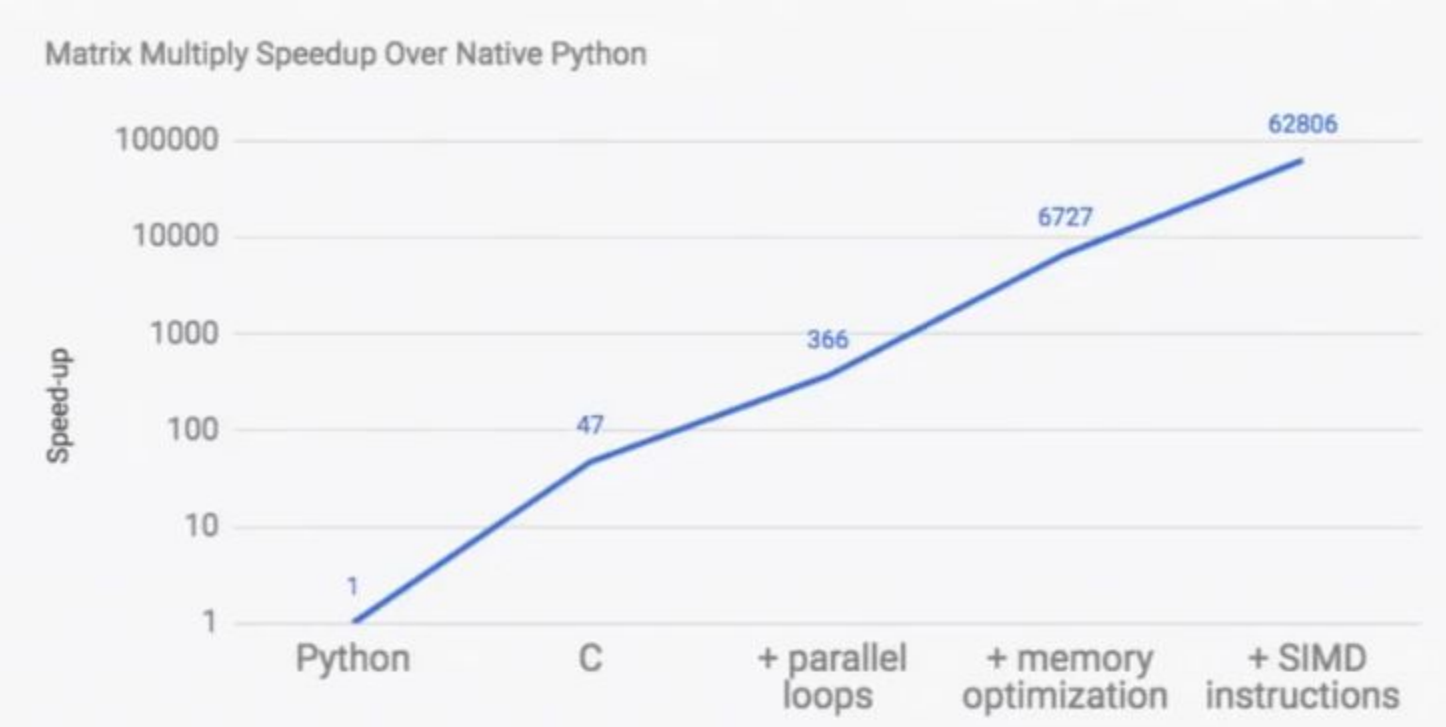
1	2	3
4	5	6
7	8	9

B

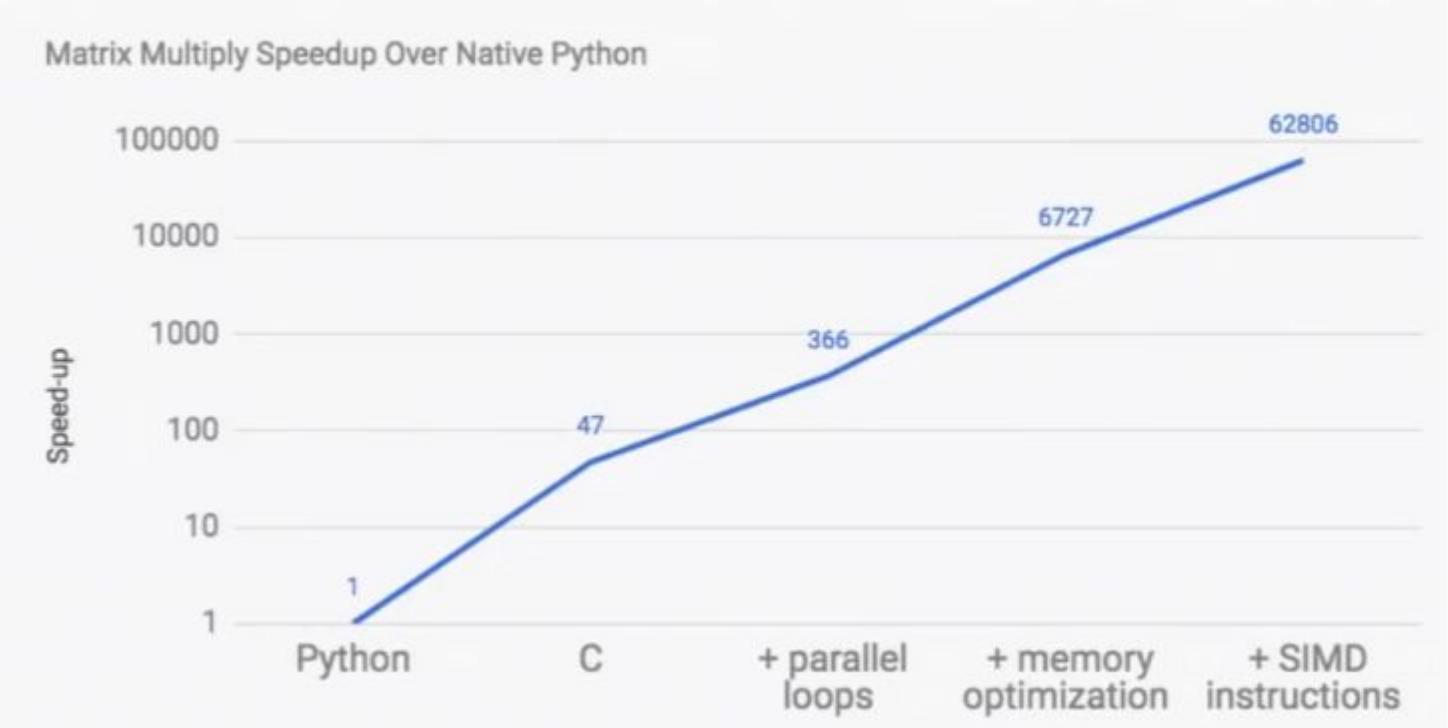
30	36	42
66	81	96
102	126	150

C

Write a code to do Matrix Multiplication


Matrix Multiply Speedup Over Native Python
Speed-up
100000
10000
1000
100
10
1
1
47
366
6727
62806
Python
C
+ parallel loops
+ memory optimization
+ SIMD instructions

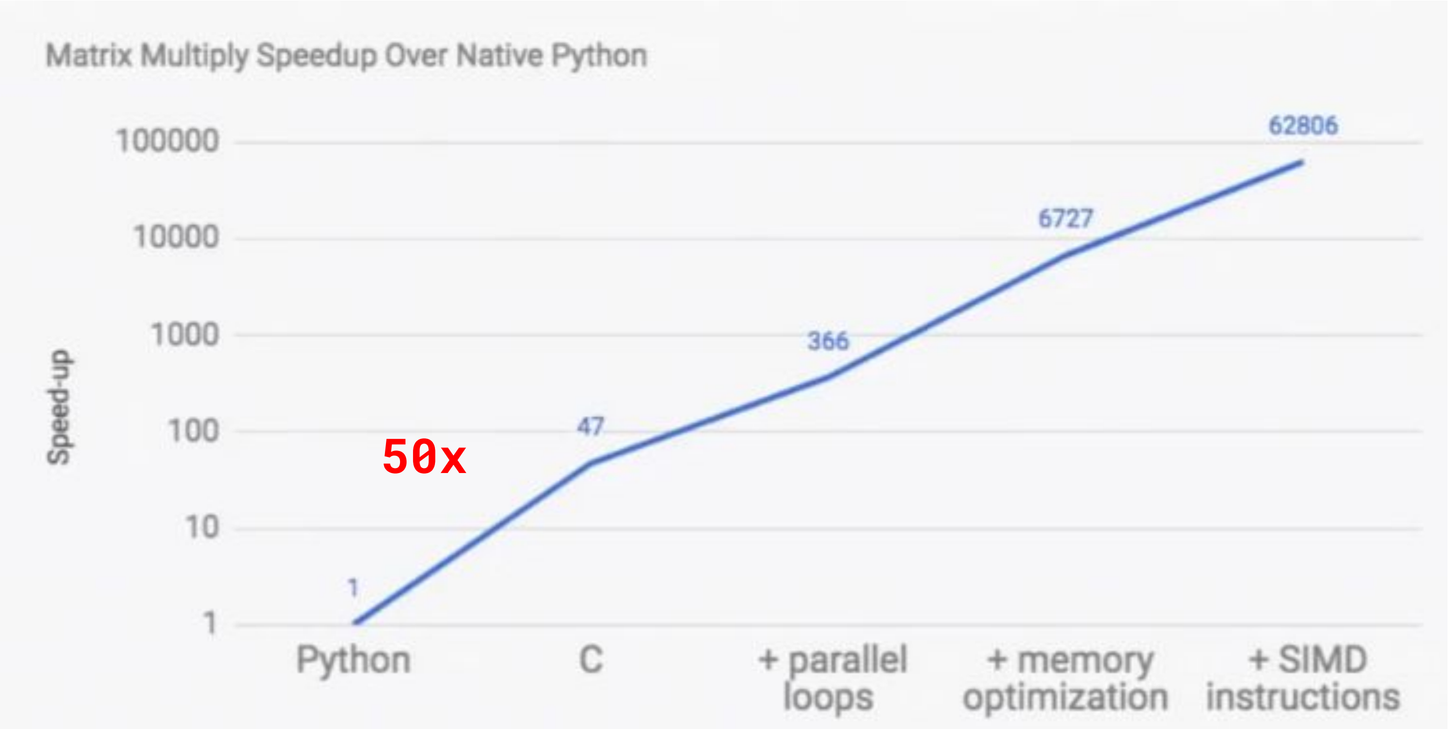
```
def matrix_multiply(A, B, C):
for i in range(len(A)):
   for j in range(len(B[0])):
       for k in range(len(B)):
           C[i][j] += A[i][k] * B[k][j]
```


Matrix Multiply Speedup Over Native Python
Speed-up
100000
10000
1000
100
10
1
1
47
366
6727
62806
Python
C
+ parallel loops
+ memory optimization
+ SIMD instructions

Fun fact: Python2 input vulnerability

Fun fact: Python2 input vulnerability Please refer “Hackin” round 8

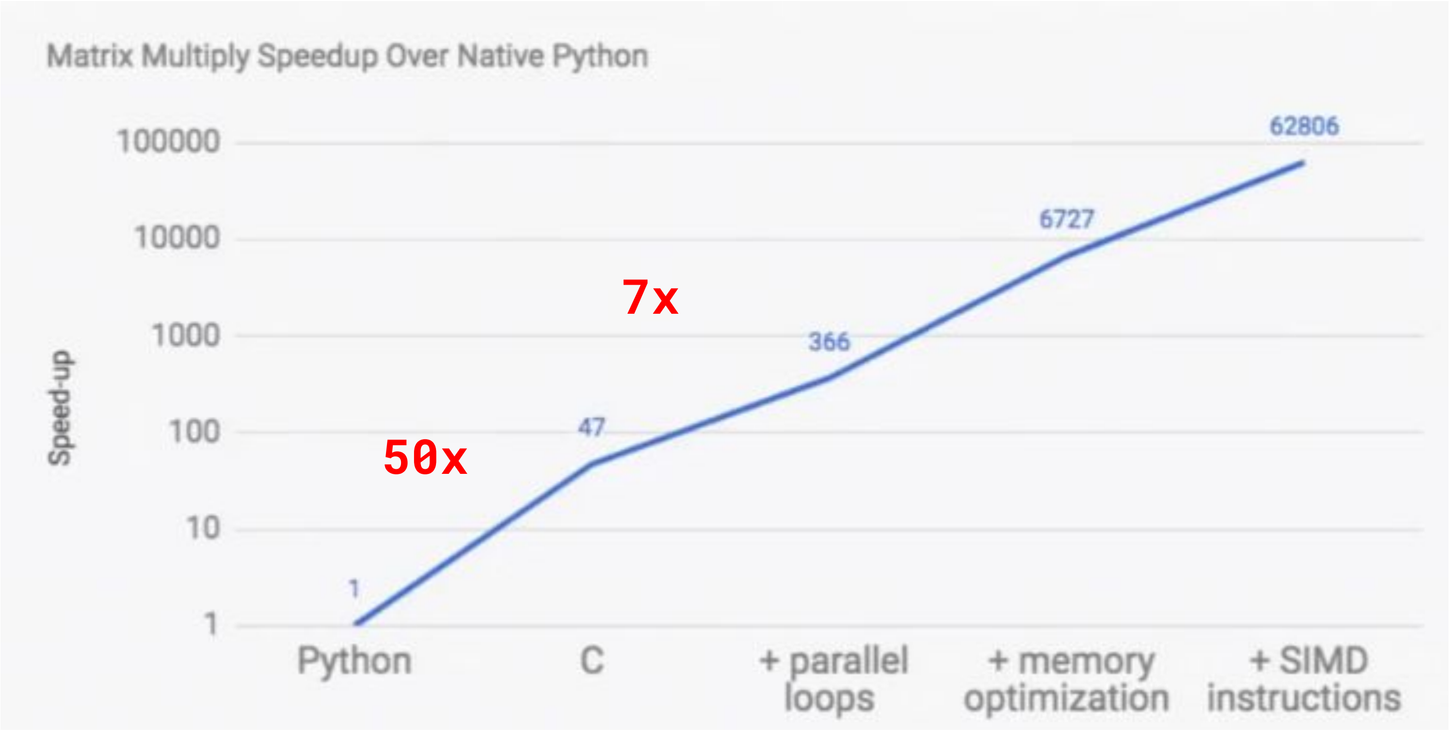
```
int matrix_multiplication(int *A, int *B, int *C, int m,
                          int n, int o){
    #pragma parallel
    for(int i = 0; i < m; i++){
        for(int j = 0; j < n; j++){
            for(int k = 0; k < o; k++){
                C[i * n + j] += A[i * o + k] * B[k * n + j];
            }
        }
    }
}
```


Matrix Multiply Speedup Over Native Python
Speed-up
100000
10000
1000
100
10
1
1
47
366
6727
62806
Python
C
+ parallel loops
+ memory optimization
+ SIMD instructions

50x

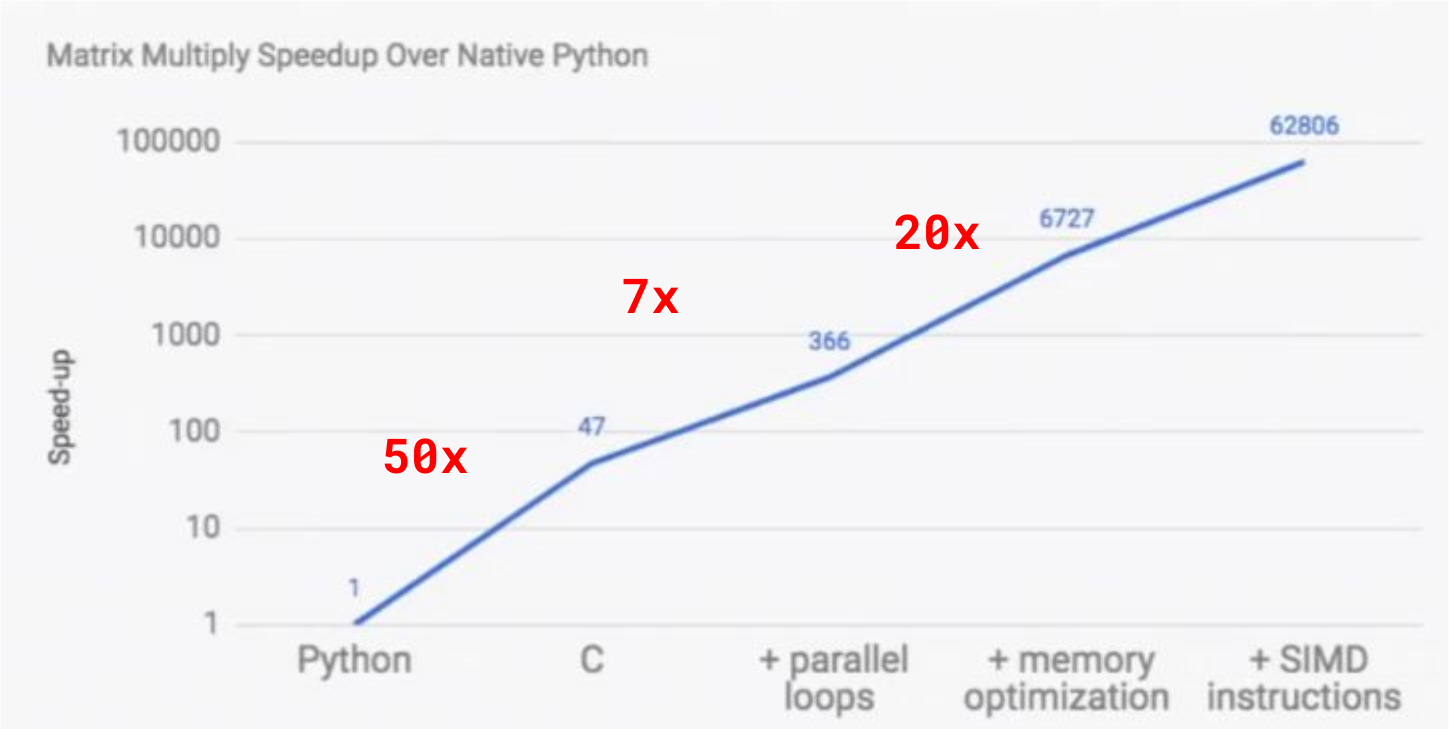
```
int matrix_multiplication(int *A, int *B, int *C, int m,
                          int n, int o){
    #pragma parallel
    for(int i = 0; i < m; i++){
       for(int k = 0; k < o; k++){
           for(int j = 0; j < n; j++){
                C[i * n + j] = A[i * o + k] * B[k * n + j];
            }
        }
    }
}
```

+ Register blocking

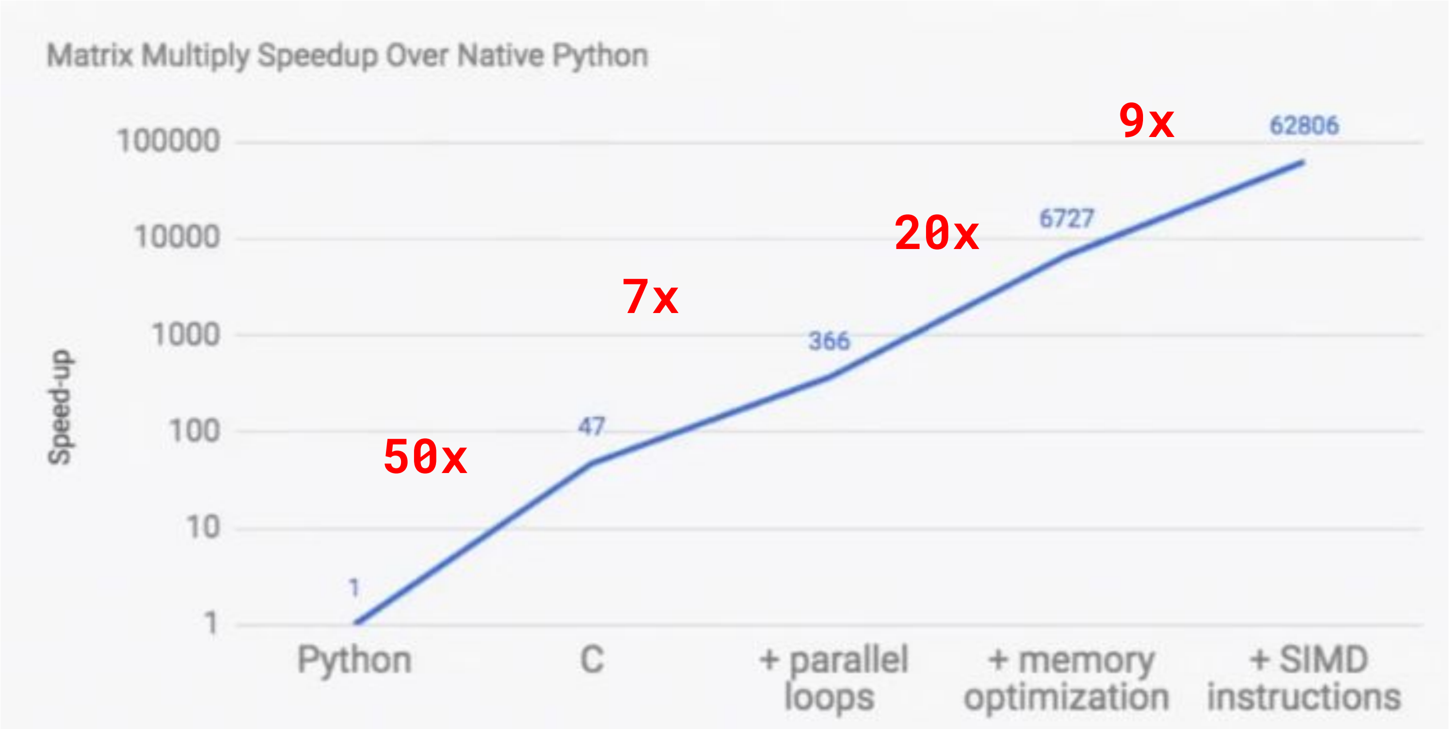

Matrix Multiply Speedup Over Native Python
Speed-up
100000
10000
1000
100
10
1
1
47
366
6727
62806
Python
C
+ parallel loops
+ memory optimization
+ SIMD instructions

50x

7x

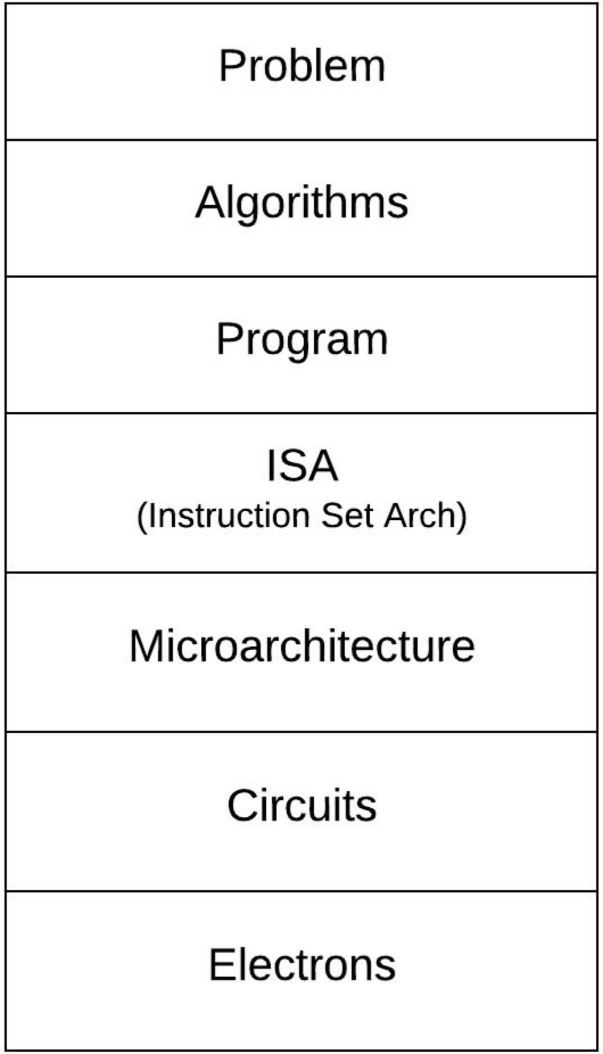


Matrix Multiply Speedup Over Native Python
Speed-up
100000
10000
1000
100
10
1
Python
C
+ parallel loops
+ memory optimization
+ SIMD instructions
1
47
366
6727
62806
50x
7x
20x


Matrix Multiply Speedup Over Native Python
Speed-up
100000
10000
1000
100
10
1
1
47
366
6727
62806
50x
7x
20x
9x
Python
C
+ parallel loops
+ memory optimization
+ SIMD instructions

Now assembly + Sorry out of slide space

We need the right abstraction for performance



Problem

Algorithms

Program

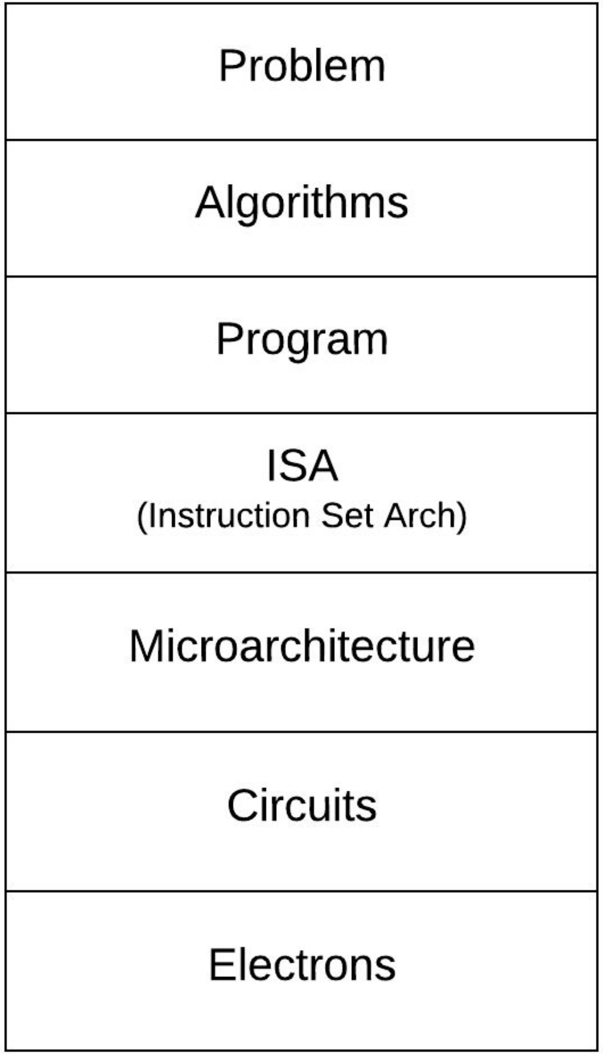
ISA
(Instruction Set Arch)

Microarchitecture

Circuits

Electrons

We need the right abstraction for everything



Problem

Algorithms

Program

ISA
(Instruction Set Arch)

Microarchitecture

Circuits

Electrons