C vs Rust

An introduction to Low Level Programming

Nerdy words (that we will be using in the workshop)

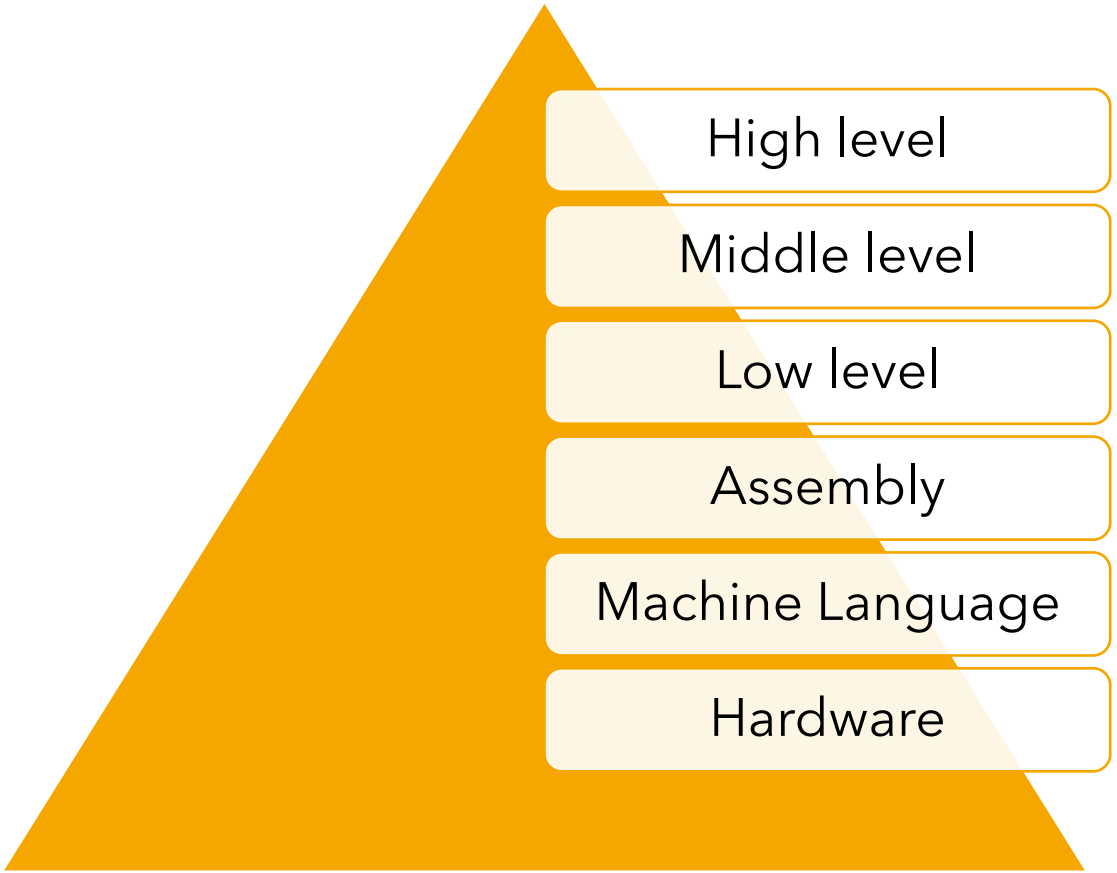
- Abstraction layers
- Statically Typed vs Dynamically typed
- Garbage Collection (In programming)

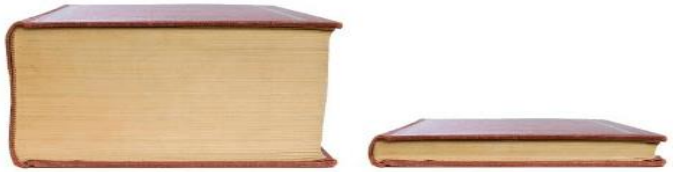
What exactly do we mean by Low Level?

What exactly do we mean by Low Level?

- Low != Negative
- Definitely not in this case!
- Low level in programming just means -- Closer to hardware.

What exactly do we mean by Low Level?





A code in C++

The equivalent in Python

Why *NOT* Low Level?

- Too Verbose
- Complicated memory management
- Follows procedural programming paradigm (no OOP!)
- Speed benefit might not be too beneficial

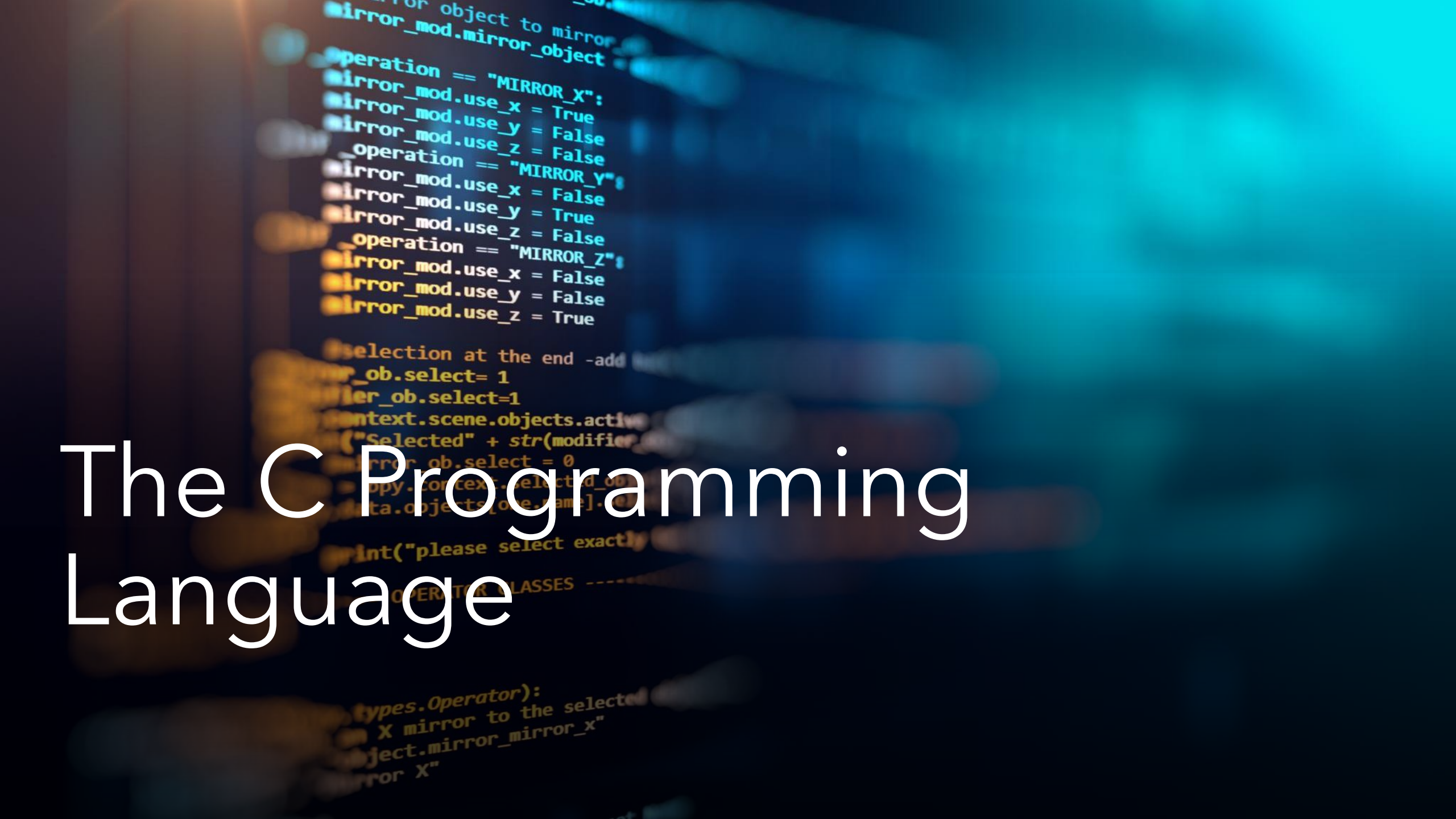
Eg. Execution time reduction from 0.5 ms to 0.1 ms is 5 times, but irrelevant irl.

- Not *many* opportunities.

Opportunities includes: Systems Engineer, Firmware Engineer, HW Engineer.

Why are we even here then?

- Because, nerdy stuff is cool… Actually because its not as irrelevant as it seems.
- Embedded Systems
- Recent heavy Investment from Government in this sector



The C Programming Language

Compared to HLLs

- Compiled language
- Static typed language
- Semicolons must be put at the end of every statement. For e.g., int *ptr = (int*) malloc(n * sizeof(int))**;**
- Does not support object-oriented programming

Why Good for Low-Level Development?

- Very close to the hardware: can use CPU registers, directly manipulate memory using pointers and references
- Highly portable
- Very small language
- Deterministic resource usage

The first C program

```c
#include<stdio.h>

int main(int argc, char* argv)
{
    printf("Hello, World");

    return 0;
}
```

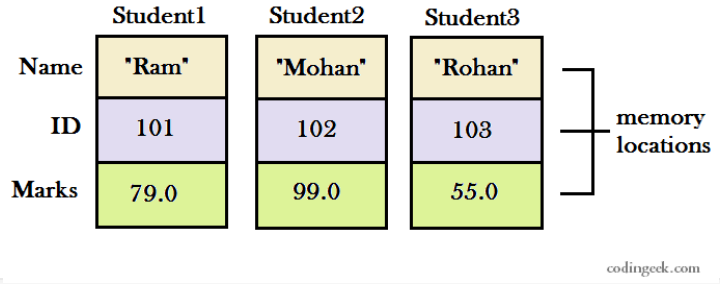
A bit more interesting C program

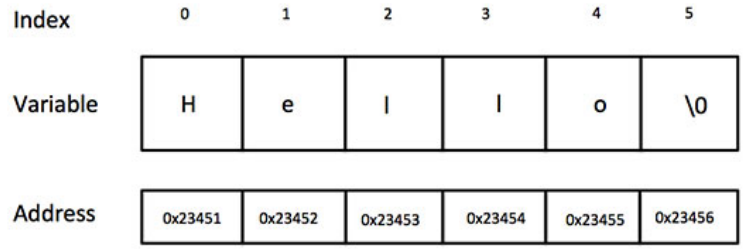
```
int a,b,c;
int count = 1;

for (b = c = 10;
a = "- FIGURE?, UMKC,XYZHello Folks,\
TFy!QJu ROo TNn(ROo)SLq SLq ULo+\
UHs UJq TNn*RPn/QPbEWS_JSWQAIJO^\
NBELPeHBFHT}TnALVlBLOFAkHFOuFETp\
HCStHAUFAgcEAelclcn^r^r\\tZvYxXy\
T|S~Pn SPm SOn TNn ULo0ULo#ULo-W\
Hq!WFs XDt!"[b+++21]; )
  for(; a-- > 64 ; )
    putchar ( ++c=='Z' ? c = c/ 9:33^b&1);
return 0;
}
```

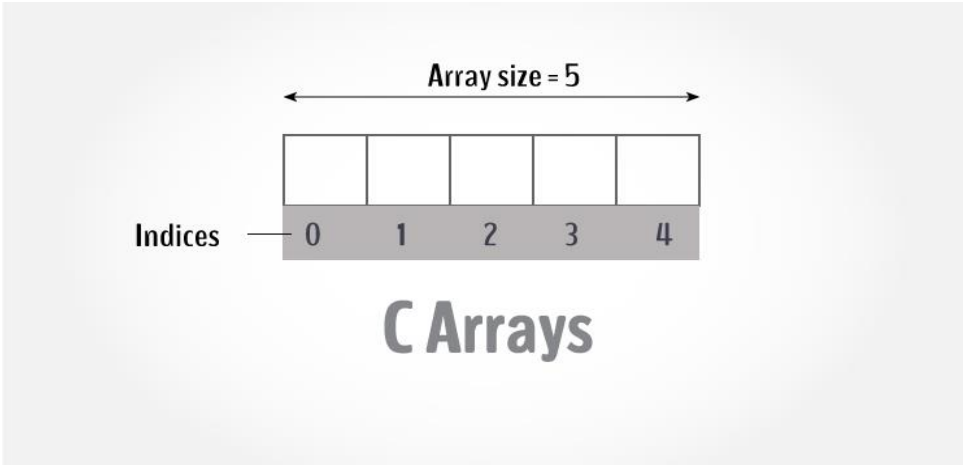
What is in C?

- Basics: Keywords, identifiers, operators, input/output
- Flow Control: if...else, loops, break and continue, switch…case, goto
- Pointers
- Functions
- Arrays
- Strings
- Structs and Unions
- Files

Overview of the components







Low-Level C

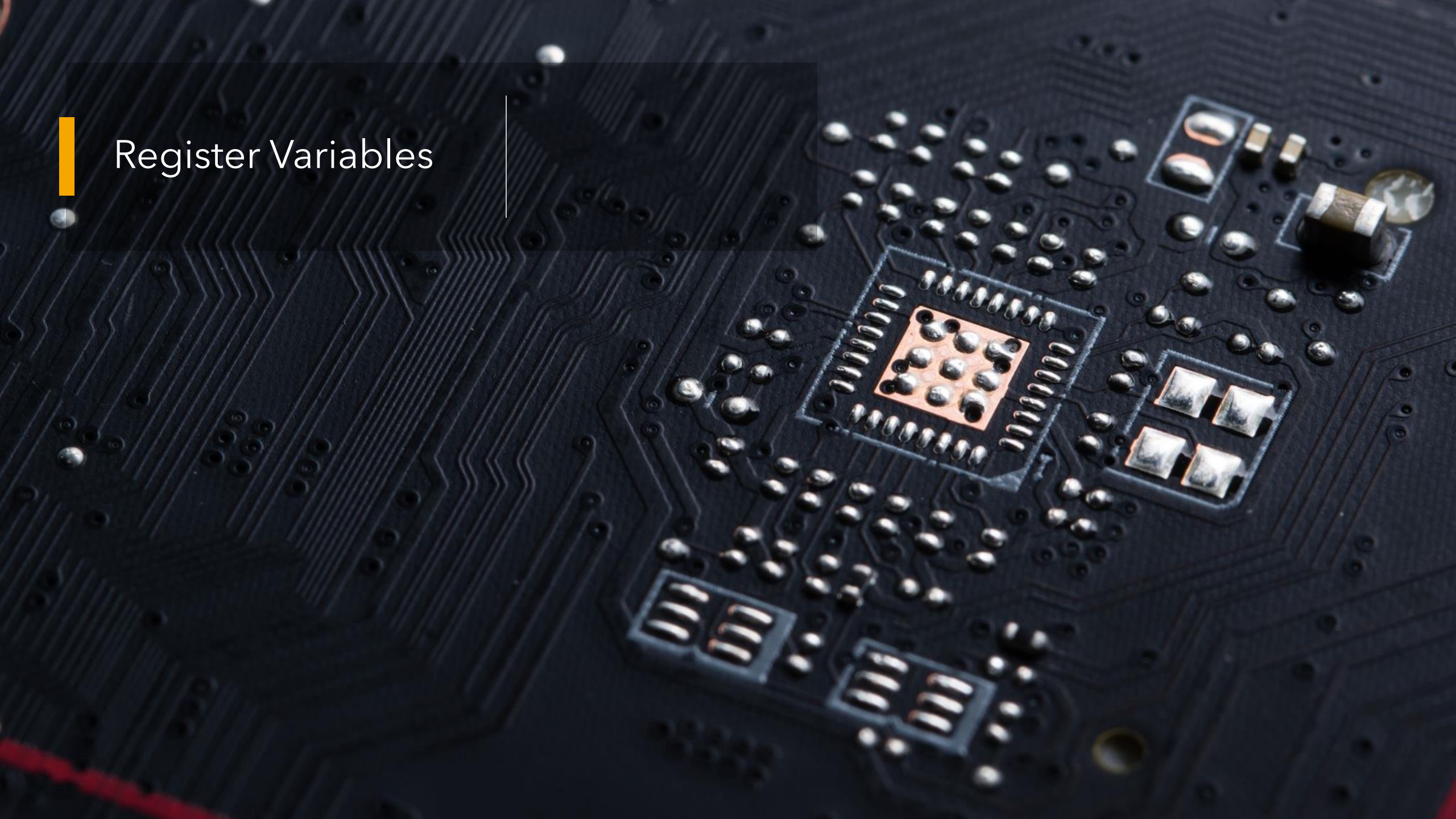
C Components for Low-Level

REGISTER VARIABLES

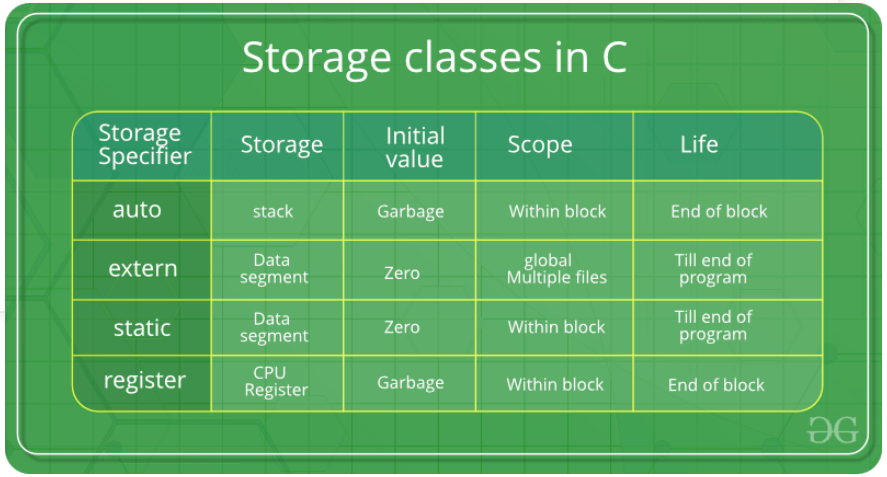
BITWISE OPERATORS

BIT MASKING

Register Variables



Usage



register int a = 10;

“A declaration of an identifier for an object with storage-class specifier `register` suggests that access to the object be as fast as possible. The extent to which such suggestions are effective is implementation-defined”

“A `register` specifier is a hint to the implementation that the variable so declared will be heavily used”

Real World Usage

- Implementing low-level interpreters
- Writing Compilers
- Deeply Embedded systems
- Specific-use algorithms e.g., Heap sort

```
inline  void max_heapify(int *H, int i){
    char OK = FALSE;
    register int l, r, max, hI;
    while(!OK){
        OK = TRUE;
        l = left(i);
        r = right(i);
        max = i;
        if(l <= H[SIZE] && H[l] > H[i]){
            max = l;
        }
        if(r <= H[SIZE] && H[r] > H[max]){
            max = r;
        }
        if(max != i){
            OK = FALSE;
            hI = H[i];
            H[i] = H[max];
            H[max] = hI;
            i = max;
        }
    }
}
```

Bitwise Operator

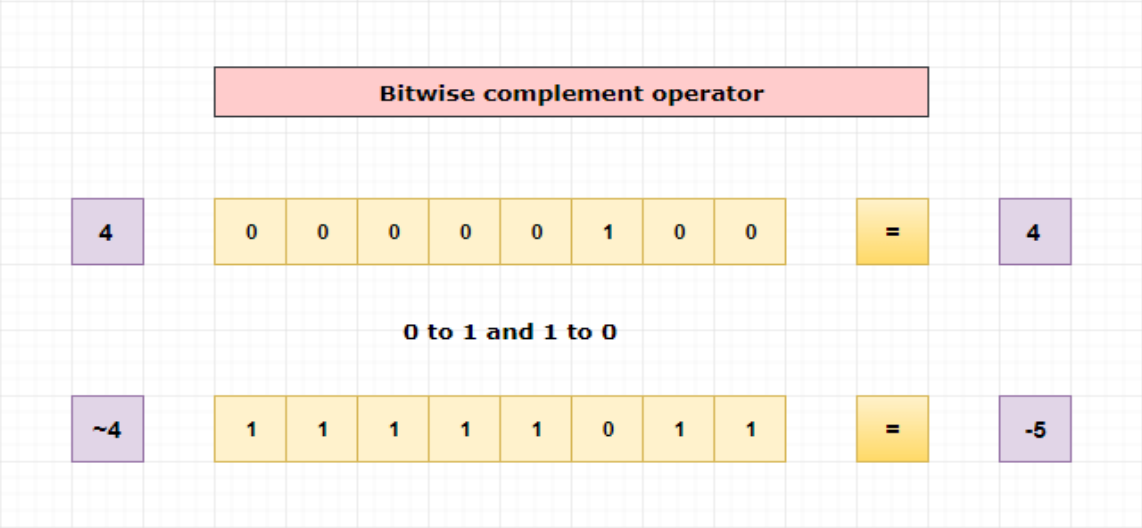
The Operators

One's Complement Operator (~)

Logical Bitwise Operators (&, |, ^)

Shift Operators (<<, >>)

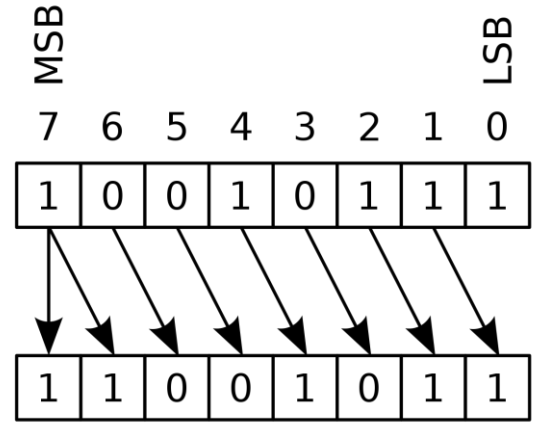
One's Complement



Bitwise

x	y	x&y	x^y	x|y
1	1	1	0	1
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

Shift



Real World Usage

- Bit Fields: Most efficient way of representing states made up of several “yes or no”
- Graphics
- Fast determination of several things e.g.,
  - For checking if odd `(value & 0x1) > 0`
  - For checking if even `(value & 0x1) == 0`

Bit Masking

Usage

Transforming the bit pattern of operands using specifically selected big patters called **masks**

```
Y = X & 0x7f;

X    = 0110 0101 1101 1111
mask = 0000 0000 0111 1111
----------------------------
Y    = 0000 0000 0101 = 0x5f
       (last 7 bit copied as it is)
```

```
Y = X | 0x3ff;

X    = 0110 0101 1101 1111
mask = 0110 0011 1111 1111
----------------------------
Y    = 0110 0111 1111 1111 = 0x67ff
```

Real World Usage

- Performing binary level tasks

```
byte imagePixel = 0xCCDDEE; /* Image in RRGGBB format R=Red, G=Green, B=Blue */

//To only have red
byte redColour = imagePixel & 0xFF0000; /*Bitmasking with AND operator */

//Now, we only want red colour
redColour = (redColour >> 24) & 0xFF;  /* This now returns a red colour between 0x00 and 0x
```

```
/* Decimal to Binary Conversion */

#include <stdio.h>
main()
{
        int i, j, cnt, nbits;
        unsigned mask;

        printf("\nEnter an Integer value: ");
        scanf("%d", &i); fflush(stdin);

        nbits = (8 * sizeof(int));
        mask = 0x1 << (nbits - 1);
        for(cnt = 0; cnt < nbits; cnt++)
        {
                j = (i & mask)? 1: 0;
                printf("%x",j);
                mask >>= 1;
        }
        printf("\n");
}
```

- Database Drivers
- Compiler Implementation
- Calculating valid network addresses for subnet
- Efficient algorithms

Rust

What is Rust?

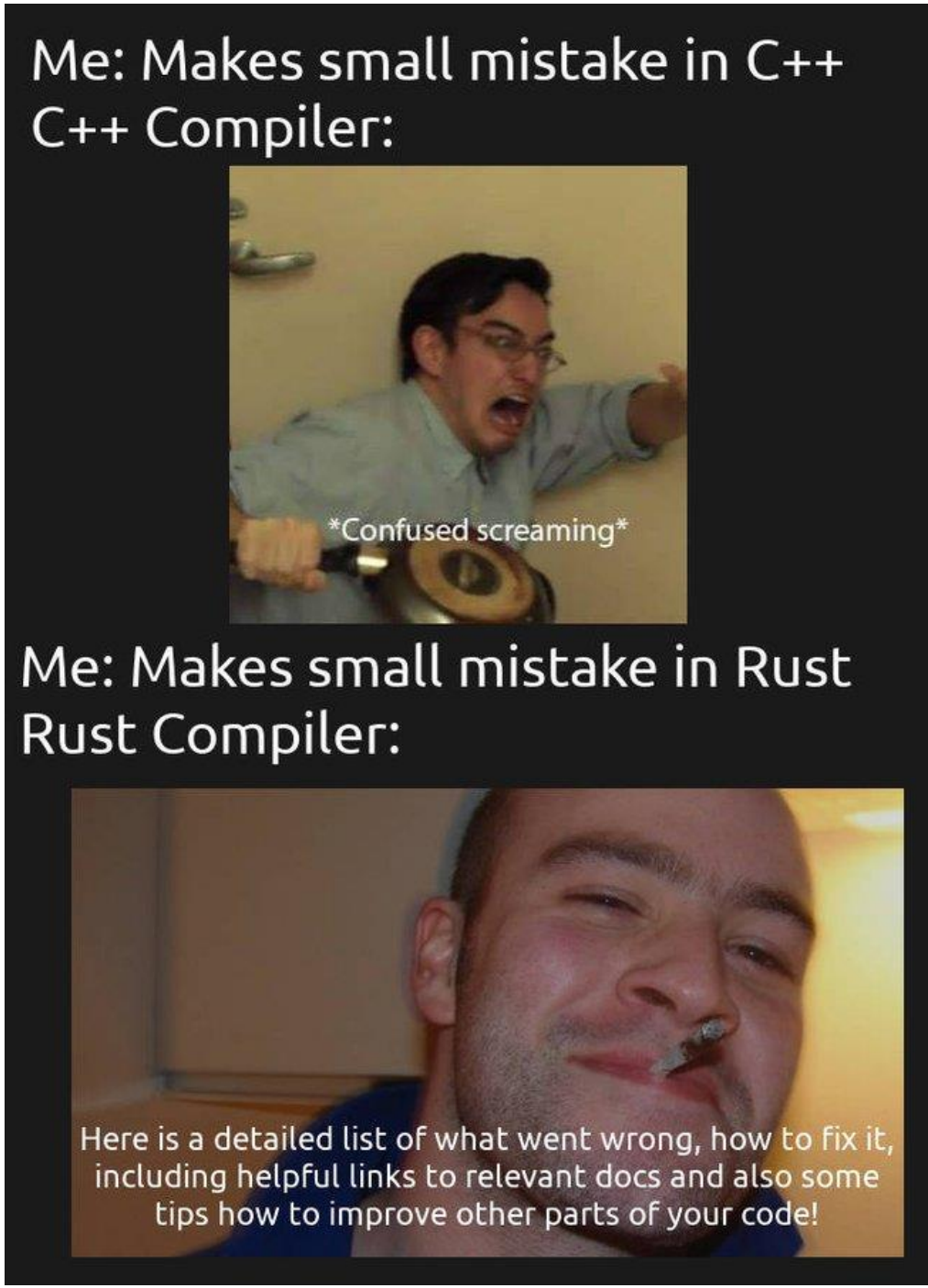
- Rust is a relatively new systems programming language (2010), it tackles the shortcomings of other languages. While providing the same high performance.
- Rust was started by a Mozilla employee as a personal project, and later was sponsored by Mozilla.

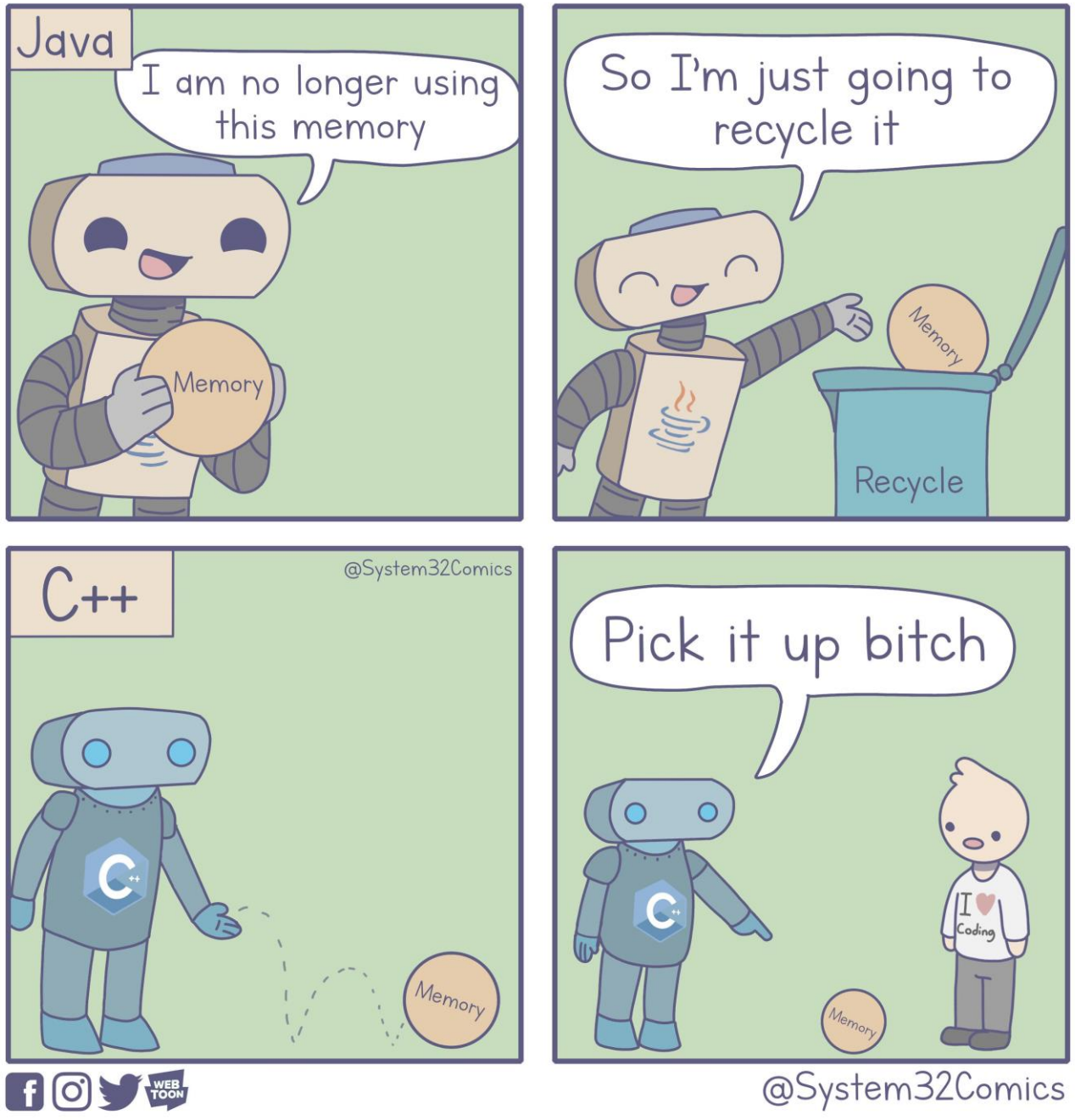
Why Rust?

- Solves many of the problems faced by even the most experienced C/C++ developers (eg. Null pointer exception)
- Guaranteed Thread-safety, and memory management
- Uses the revolutionary concept of borrow checking for memory management.
- Has a cool click in its name (“I code in C” -- *meh*)
- Trendy, cool (nerdy) bois are already doing it. (Seriously though, don’t pick a language like this pls)

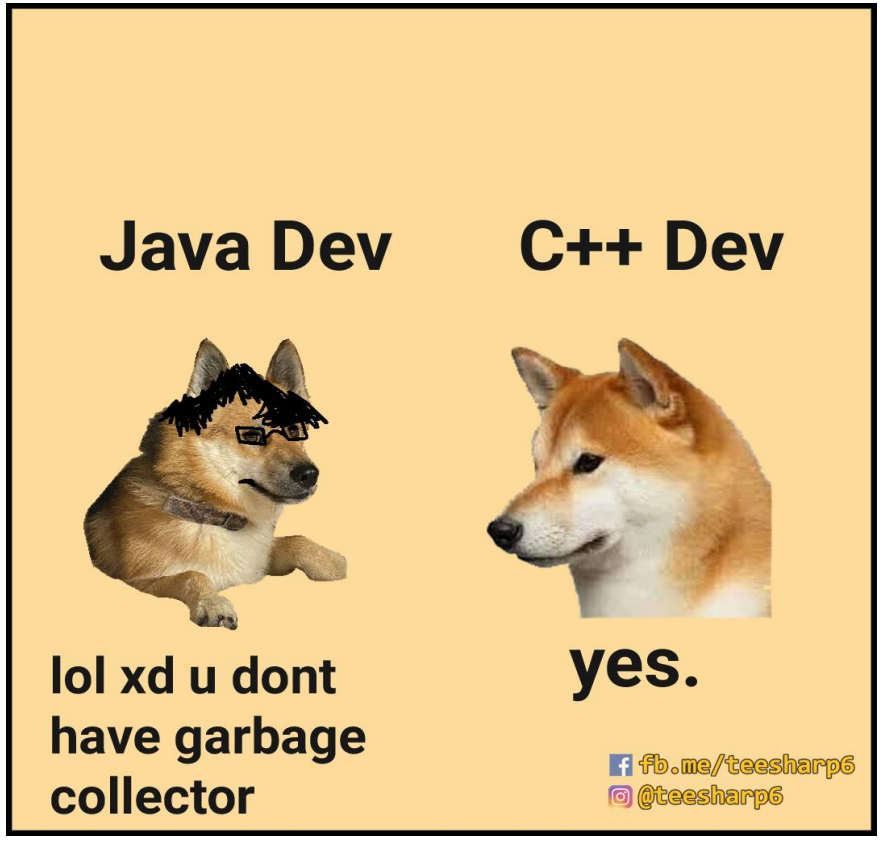
What’s so special in Rust?

- C expects the programmer to manage the memory in their program. Which is s huge power (and thus comes with great responsibility -– Which is often misused)
- Other HLL languages have a “Garbage Collector”, which reclaims the unused memory from time-to-time, but leads to slowdowns (and memory leaks).
- Rust follows a unique middle way, Rust doesn’t allow referencing of a variable at multiple places at the same time.
- Compiler is very strict and cruel it doesn’t compiles the code if you write garbage code. (lol)





Garbage Collector Appreciation slide



Basic Examples (Syntax is so hawt)

1) Hello World

```
10
11
12  fn main() {
13      println!("Hello, World!");
14  }
15
16
```

2) Factorial of a number

```
10
11
12  fn factorial(i: u64) -> u64 {
13      match i {
14          0 => 1,
15          n => n * factorial(n-1)
16      }
17  }
18
19
```

Rust features

- Like C, has funcs, structs, arrays, vectors, bitwise operators etc.
- Does not have Polymorphism and Classes. But Polymorphism is somewhat possible due to traits
- Traits – Similar to interface in Java and protocols in Swift. Basically ensures and enforces implementation of certain code.

Rust Features (contd.)

```
16
17  fn main() {
18
19      // Create a vector
20      let mut vector:Vec::<i32> = Vec::new(); // "mut" allows us to "mutate" the variable later
21
22      // Another way of creating a vector. Notice we dont use "let" or "let mut" here,
23      // Thats because we are "mutating" the previous vector variable.
24      vector = vec![1,2,3];
25      let a = 1;
26      let b = 1;
27      if a == b {
28          println!("{} is equal to {}", a, b); // prints `1 is equal to 1`
29      } else {
30          println!("Not equal!");
31      }
32  }
33
```

Rust Features (contd.)

```
11
12  struct Student {
13      name: String,
14      year: u16,
15  }
16
17  fn main() {
18
19      let me: Student; // Creating a struct. (Note: we can have a constructor/initializer as well)
20
21      // Rust has 2 String types,
22      // str and String. &str is defined by simply wrapper text in double-quotes
23      // String is of dynamic size and can be assigned/reassigned at runtime.
24      // str needs a size at compile time.
25      me.name = String::from("Raghav Vashisht");
26      me.year = 2;
27
28  }
29
```

struct Student

What we use ***all these*** for

Rust

- Can be and has been used to create OS (microkernels), Device Drivers, etc.

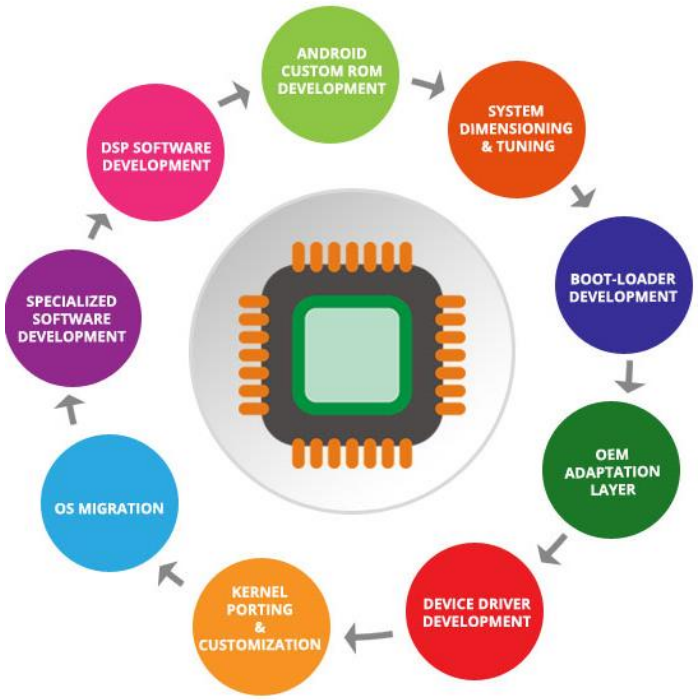
(interesting? Google: `phil opp rust`)

- Can be used to create HTTP/REST Servers

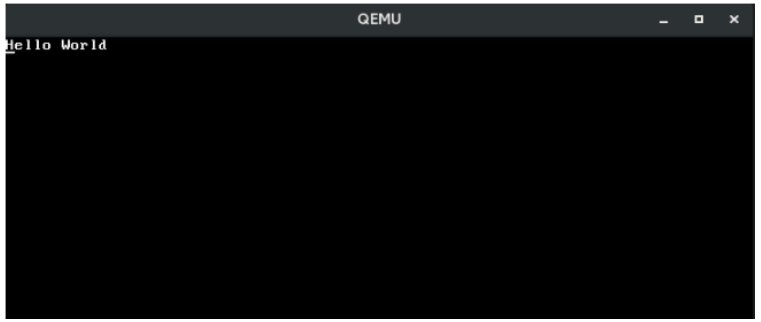
(interesting? Google: `Actix`)

- Can be used with Web Assembly (WASM) to run on browsers!
- Can be used in Kernel Development

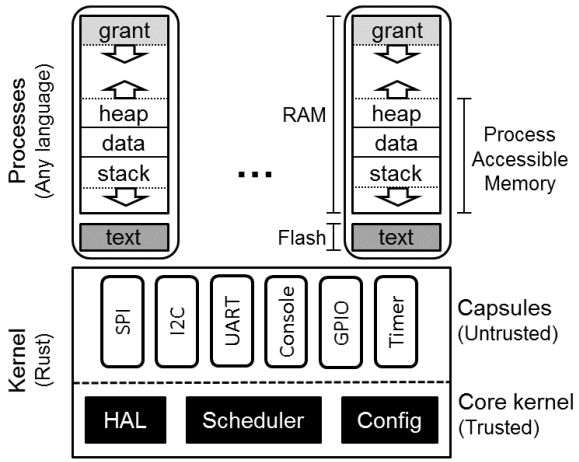
Device Driver and Kernel Development

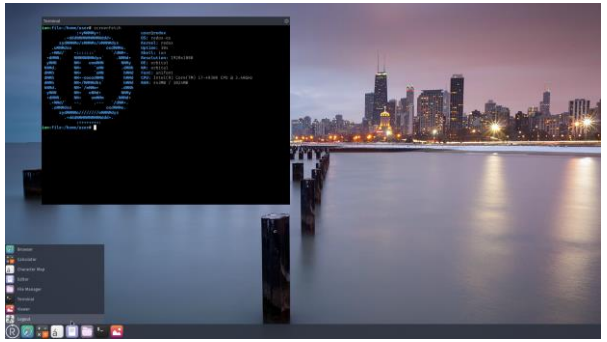



```
#include <linux/module.h>
#include <linux/kernel.h>
int init_module (void)
{
  printk ( "\nHELLO WORLD\n"
  return 0;
}
void cleanup_module (void)
{
  printk ( "\nGOOD BYE\n" );
}
```



Operating system development

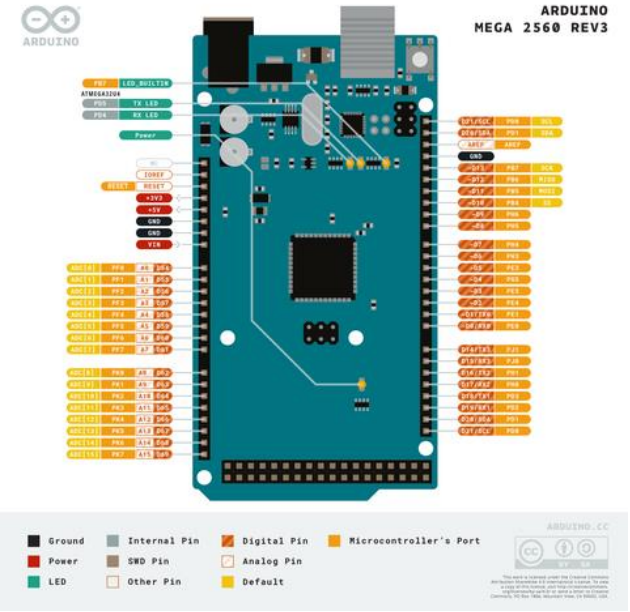


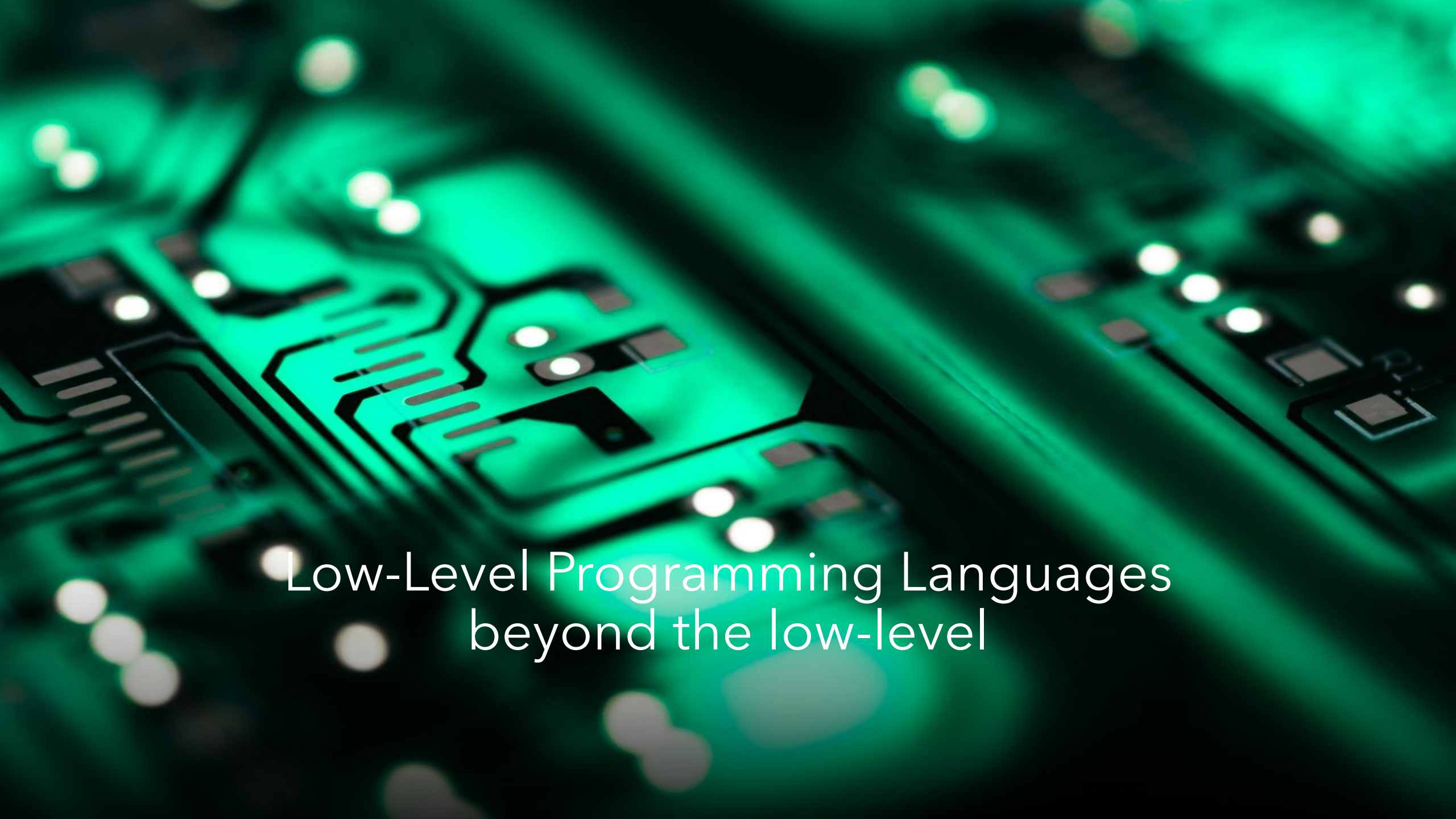


Direct Port Manipulation

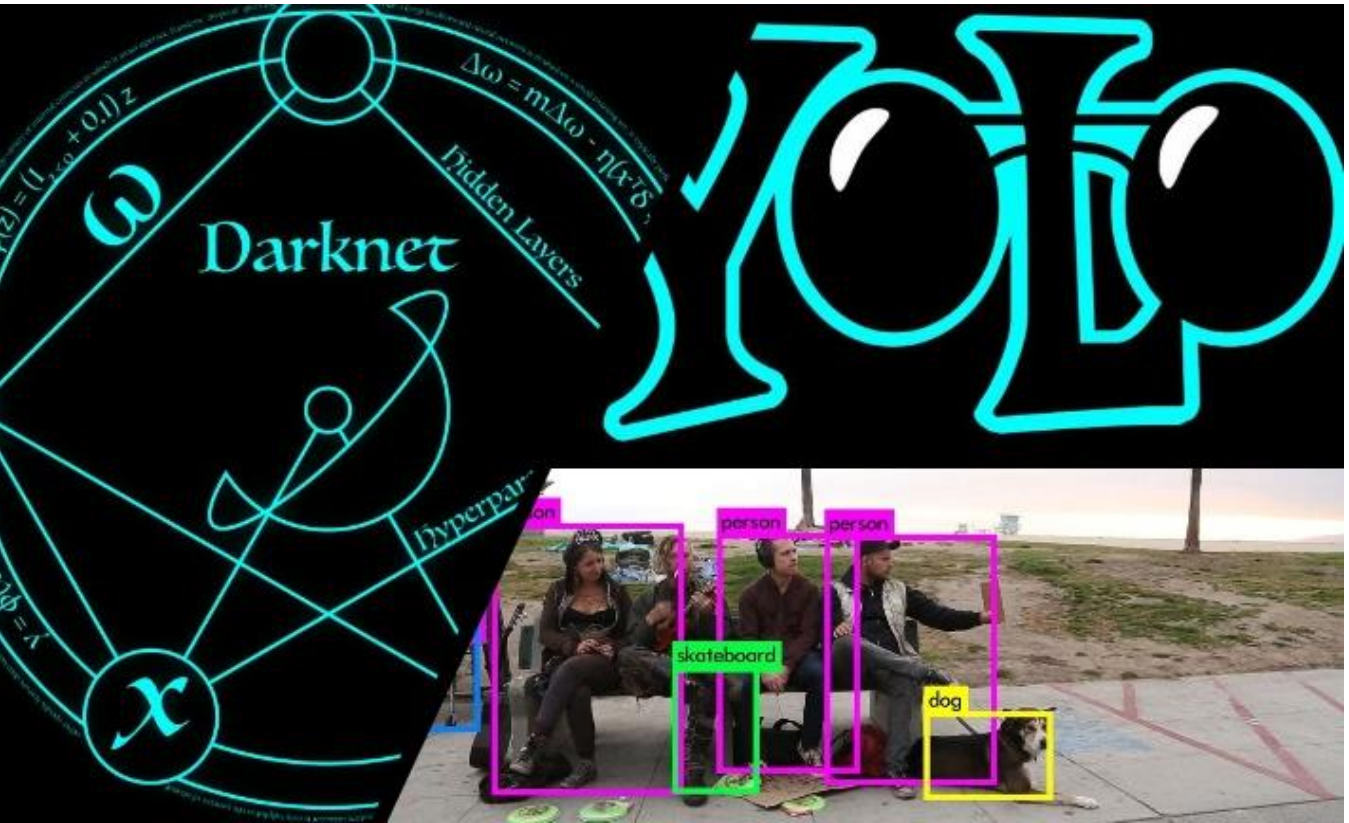
```
digitalWrite(LED_BUILTIN, HIGH);
delay(300);
digitalWrite(LED_BUILTIN, LOW);
delay(700);
```

```
PORTB |= B00100000;
delay(300);
PORTB &= B11011111;
delay(700);
```




Low-Level Programming Languages
beyond the low-level

Use of C in Object Detection – Darknet YOLO



Actions | Projects 7 | Wiki | Security | Insights

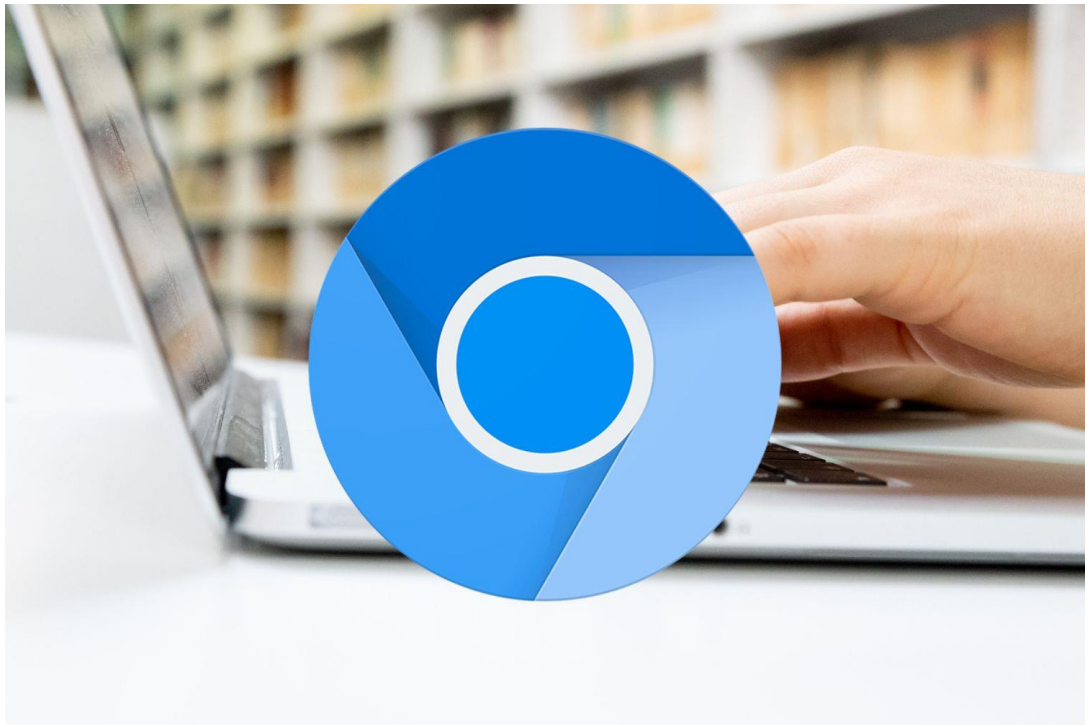
master ▾ darknet / src /

his branch is 1752 commits ahead, 119 commits behind pjreddie:master.

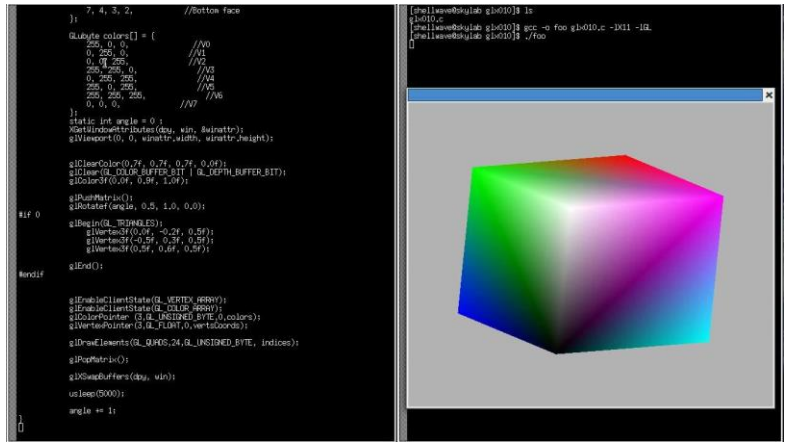
AlexeyAB Fixed equidistant point threshold

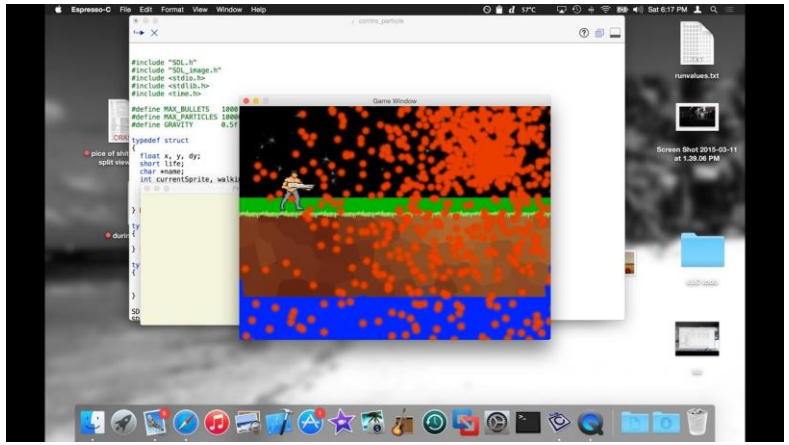
.editorconfig	Just used spaces for indents instead of Tabs
activation_kernels.cu	Hard_mish is fixed
activation_layer.c	fix calloc realloc failure
activation_layer.h	improve compatibility with c++ compilers, prepare for CMake
activations.c	Improved speed of [contrastive] layer. Added Hard-Mish activation. Ad...
activations.h	Improved speed of [contrastive] layer. Added Hard-Mish activation. Ad...
art.c	Some C OpenCV functions replaced by CPP functions
avgpool_layer.c	fix calloc realloc failure
avgpool_layer.h	ZED 3D Camera support added to ./uselib (yolo_console_cpp.exe) example
avgpool_layer_kernels.cu	fixes for ci and included files
batchnorm_layer.c	self-adversarial training
batchnorm_layer.h	final compile fix
blas.c	Added: [net] contrastive_color=0 and [contrastive] contrastive_neg_max
blas.h	Added: [net] contrastive_color=0 and [contrastive] contrastive_neg_max
blas_kernels.cu	Added [convolutional] coordconv=1 for GPU-only
box.c	Fix nms for valid, coco 2014 -> 2017
box.h	Added param to [yolo] iou_thresh_kind=giou iou_thresh=0.213
captcha.c	Fixed many warnings
cifar.c	Moved all OpenCV functions to the image_opencv.cpp/h

Browsers, Libraries of HLLs



Game Programming and 3D Graphics





Questions?

Thank you