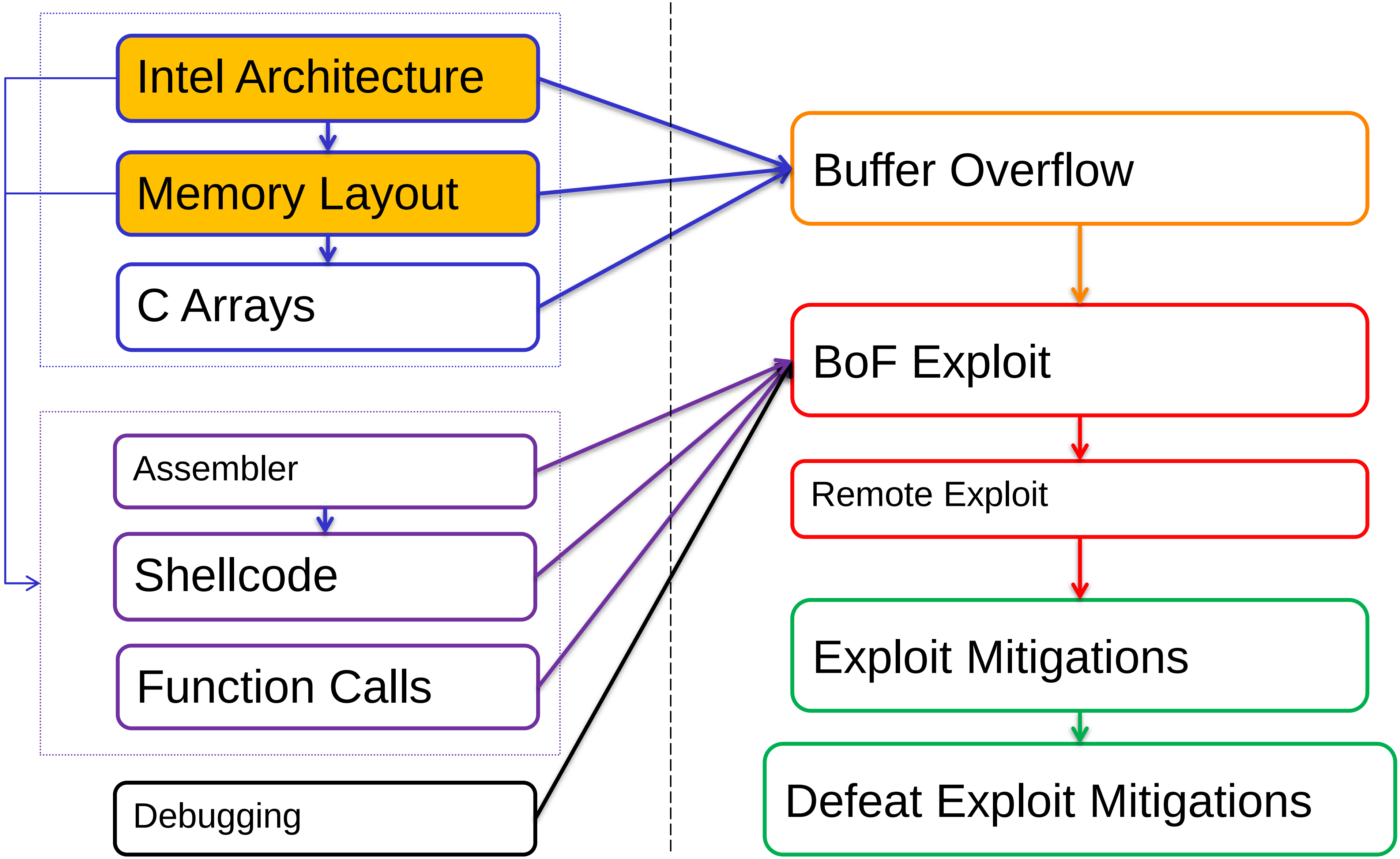
Exploiting & Defense
Day 1 Summary

Content

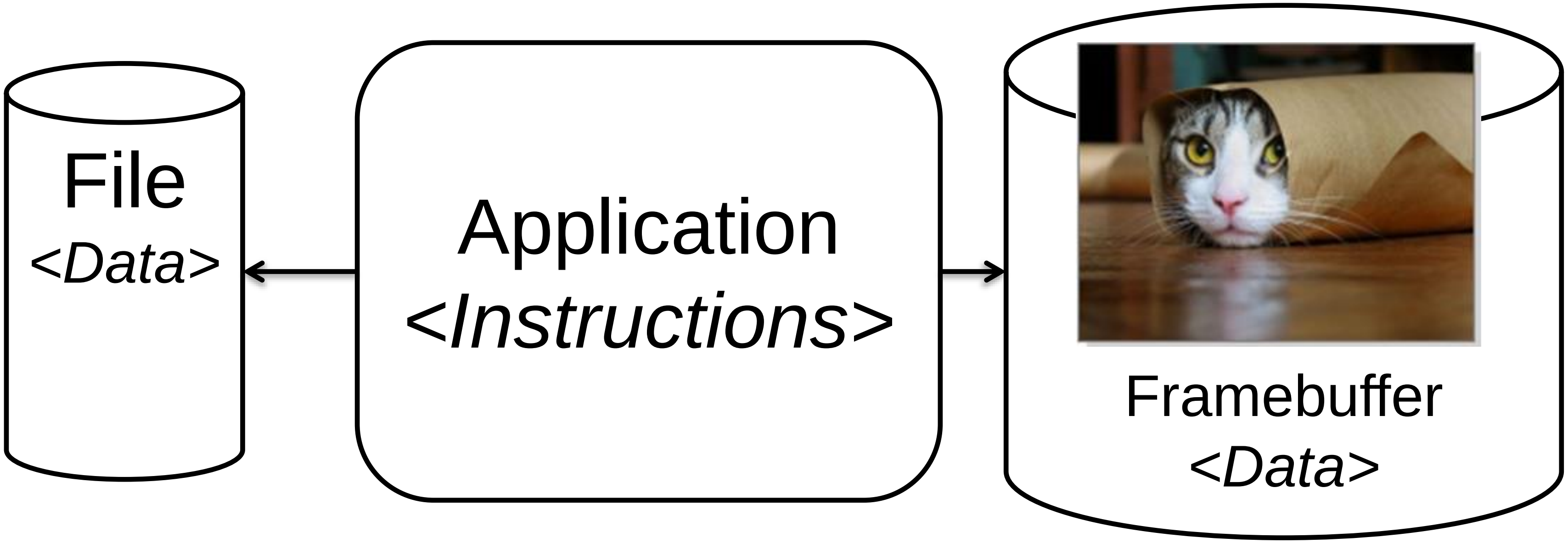

Intel Architecture
Memory Layout
C Arrays
Assembler
Shellcode
Function Calls
Debugging
Buffer Overflow
BoF Exploit
Remote Exploit
Exploit Mitigations
Defeat Exploit Mitigations

0x02_Intro_Technical

Summary

What is a picture?

- Data for the computer
- When interpreted correctly, displays a cat
- When interpreted wrongly, displays garbage / crashes
- When interpreted wrongly in the right way, lets us hack a computer



Vulnerability Types

Memory corruption occurs in a computer program when the **contents of a memory location are unintentionally modified** due to programming errors; this is termed violating memory safety. When the corrupted memory contents are used later in that program, **it leads either to program crash or to strange and bizarre program behavior**

Modern programming languages like C and C++ have powerful features of explicit memory management and pointer arithmetic. These features are designed for developing "efficient" applications and system software.

https://en.wikipedia.org/wiki/Memory_corruption

What is an exploit? Hacking related

to exploit (v): To take advantage of a vulnerability so that the target system reacts in a manner other than which the designer intended.

the Exploit (n): The tool, set of instructions, or code that is used to take advantage of a vulnerability.

(The Shellcoders Handbook, 2nd Edition, p4)

Types of exploits

Local

Server-side

Client-side

What is vulnerable?

What software is affected?

Software developed in unsafe programming languages

- (ASM)
- C
- C++
- Fortran (lol)

Who writes software in C/C++, anyway?

- IE, Chrome, Firefox
- Apache / IIS
- Postfix, Sendmail
- BIND
- MS Office / LibreOffice
- Antivirus
- Other “Security” Software

Programs and Data

Definition of a “program”:

“A program is a set of instructions ~~which modifies data”~~

which is controlled by data”

Or in other words:

Data is manipulating the instruction flow of a program, not the other way round

Recap

Software:

- Important software is written in **C/C++**
- Memory corruption bugs are very, very prevalent

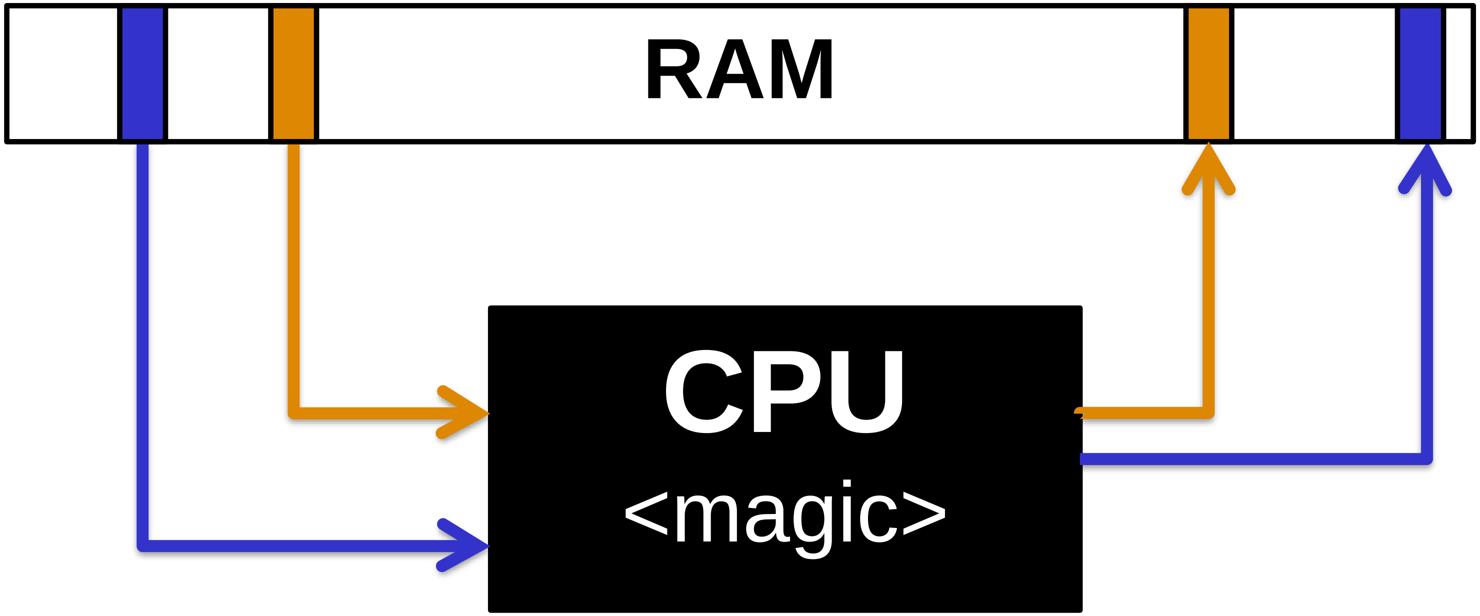
- We are concerned with **memory corruption vulnerabilities**
  - Modify stuff in a program which should not be possible
- A program which misuses a memory corruption vulnerability is called an **exploit**
  - There can be local-, server- and client exploits
- A exploit injects **additional code** into a trusted app and executes it

- For attacker, **data influences execution of code** (weird machines)

0x10_IntelArchitecture

Summary

von Neumann Architecture



Read:

- Data
- Instructions

Write:

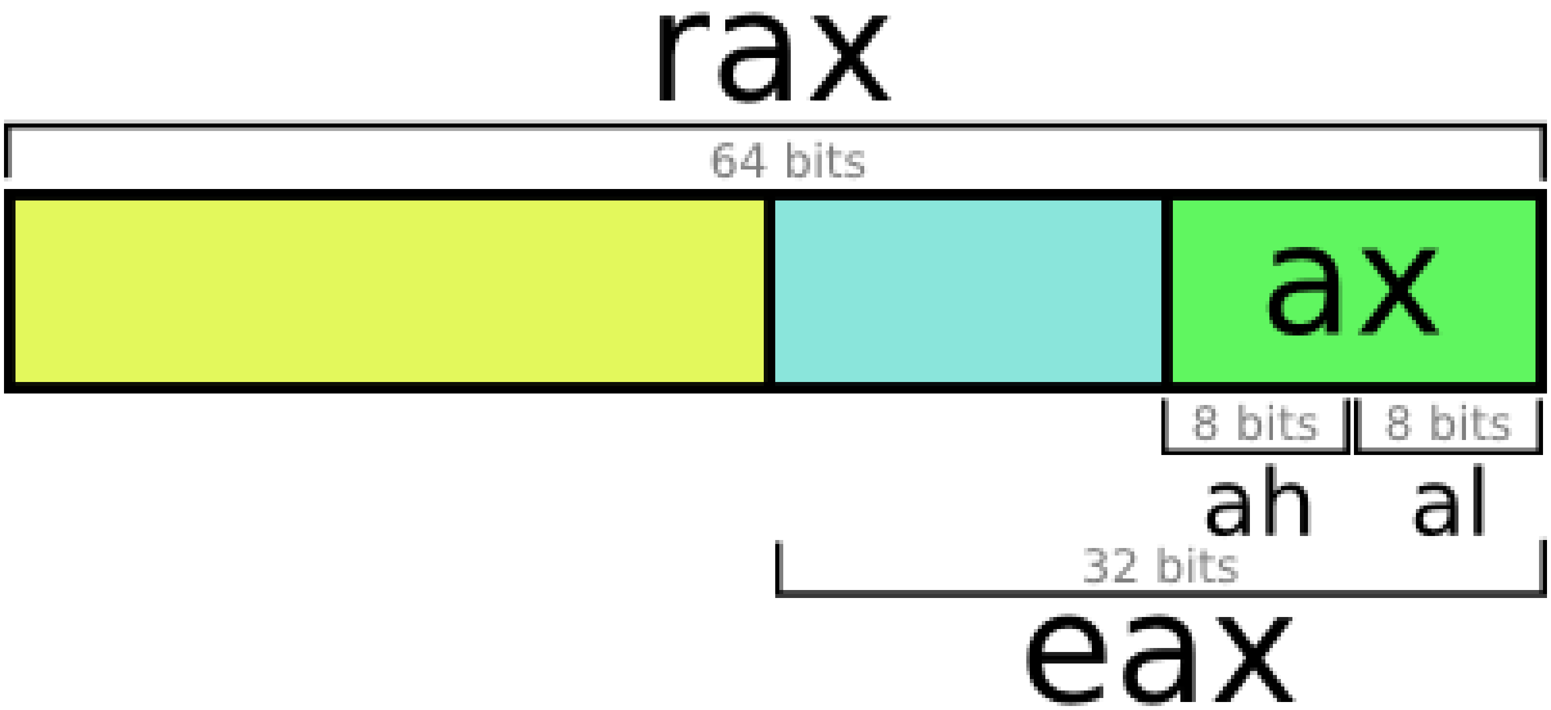
- Data
- Instructions

Overview: CPU Registers

32	64	Acronym	Points to?
EIP	RIP	Instruction Pointer	Next instruction to be executed
ESP	RSP	Stack Pointer	Top of Stack
EBP	RBP	Base Pointer	Current Stack Frame (Bottom)

Print this slide and stick it on your bathroom mirror

Overview: CPU Registers

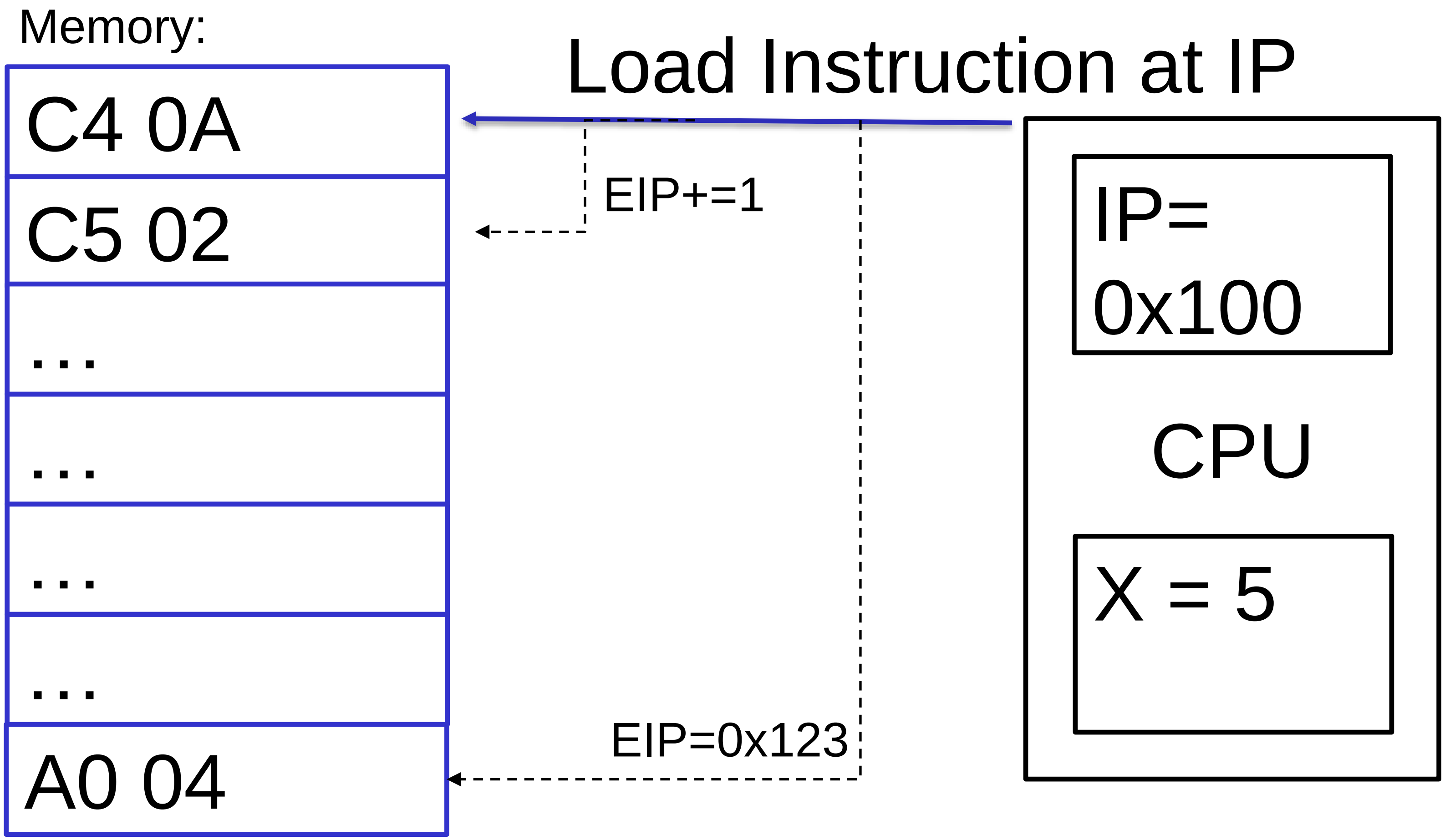


Overview: CPU Registers

Recap:

- CPU work with **registers**
- Registers can hold **data**
- Registers can also hold **addresses** of memory locations (to write/read data)
- They can be 32 bit (**E**AX) or 64 bit (**R**AX)
- Some registers are multi-purpose
- Some registers are special (RIP, RBP, RSP)

Overview: Computerz



Memory:

C4 0A

C5 02

...

...

...

...

A0 04

Load Instruction at IP

EIP+=1

EIP=0x123

IP=
0x100

CPU

X = 5

Hex Numbers

Hex: 0 1 2 3 4 5 6 7 8 9 **A B C D E F**

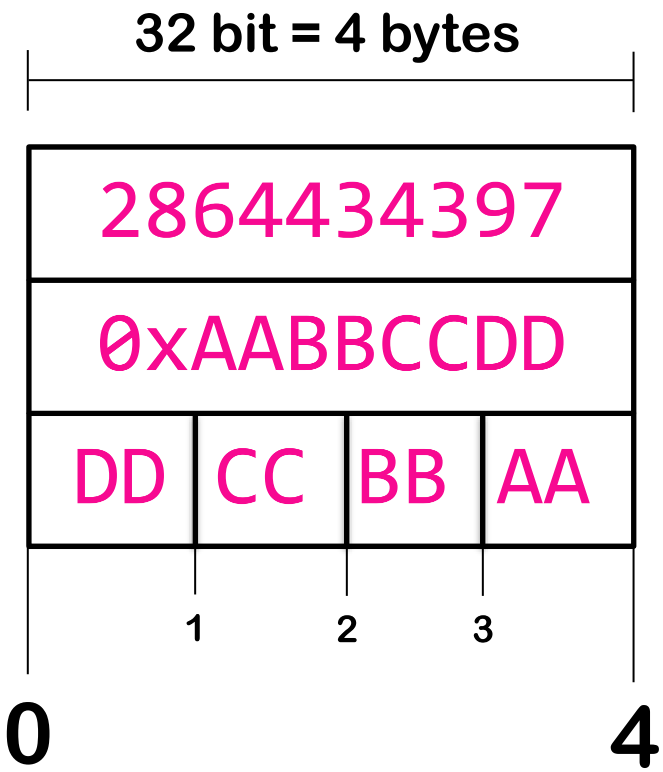
1 hex digit: 16 values (4 bit, 2^4)

*2 hex digits: 256 values (8 bit, 2^8 = 2^4 * 2^4)*

16 * 16 = 256

1 Byte = 8 Bit = 256 values!

Endianness: Little Endian (Intel)



Number in Decimal (10)

Number in Hex (16)

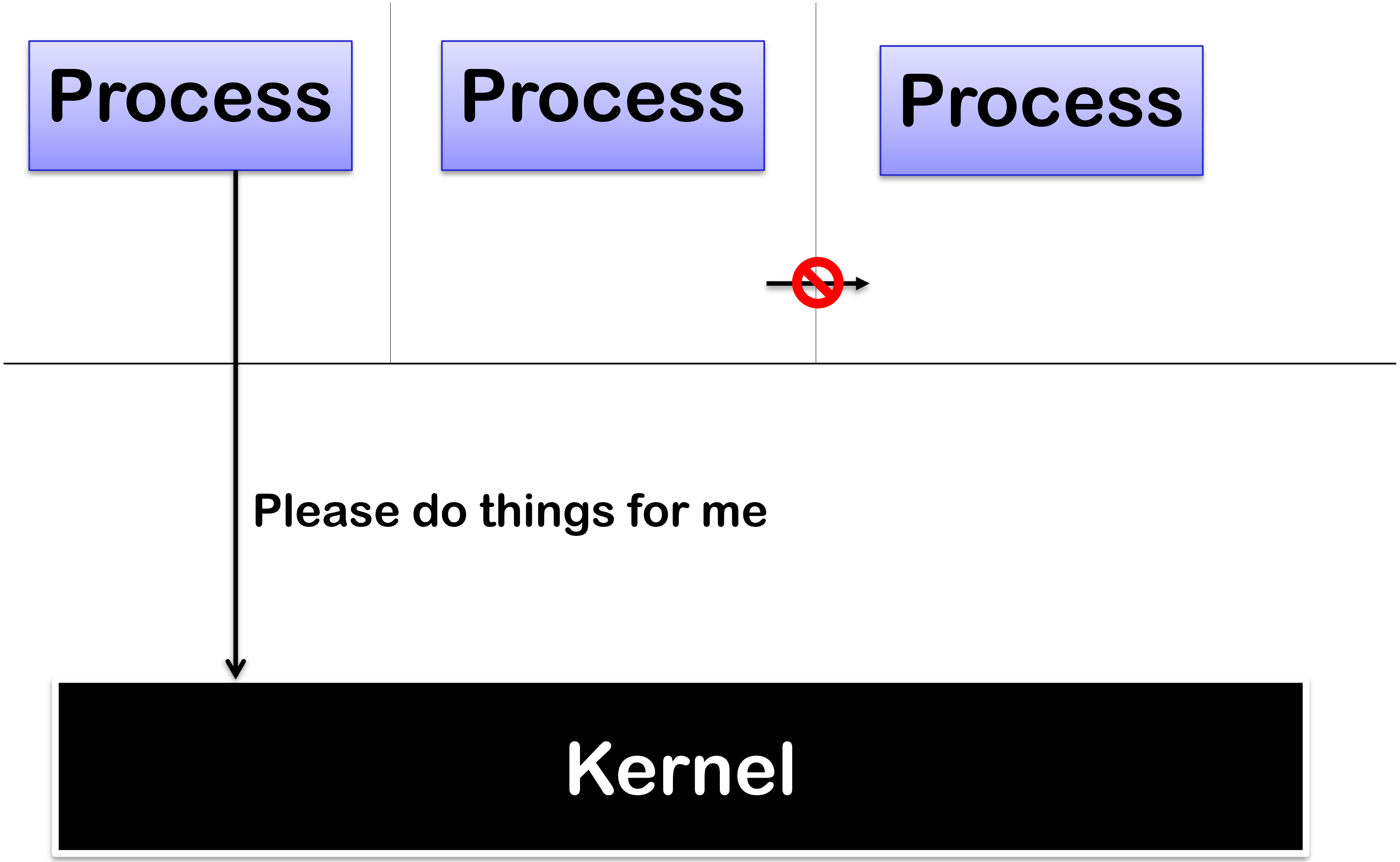
Little Endian Storage

Numbers in memory

Recap:

- Numbers can be displayed in decimal, or hex (0-9, a-f)
- Numbers are stored as 16, 32 or 64 bit values, mostly as little endian
- If we look at little endian numbers as bytes, they are inverted
- If we look at numbers in memory, we can’t know if they are 8, 16, 32 or 64 bit
- We can try to interpret bytes as ASCII

OS Basics: Processes



Process

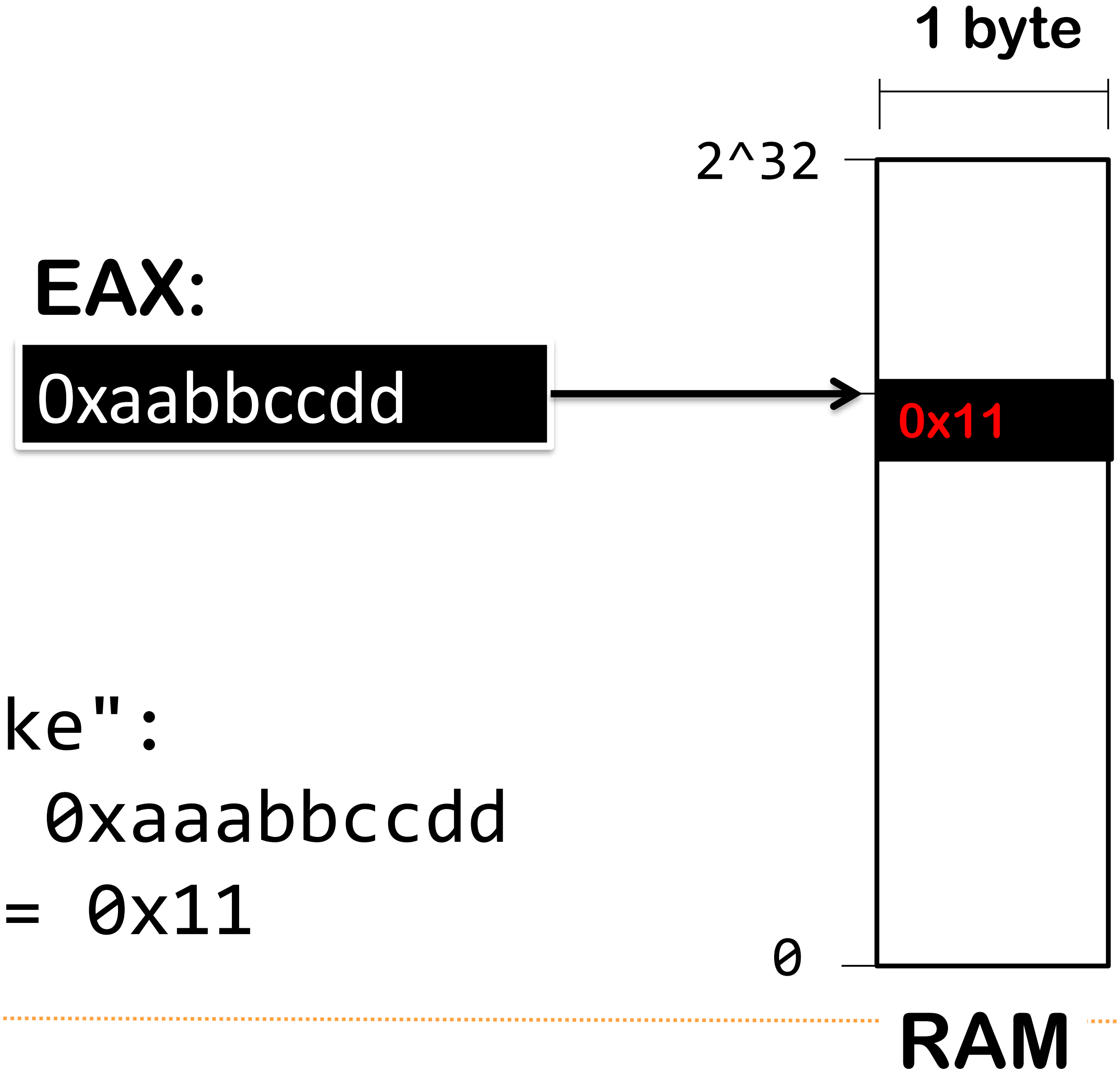
Process

Process

Please do things for me

Kernel

OS Basics: Process and Memory



```
"C like":
eax = 0xaaabbccdd
*eax = 0x11
```

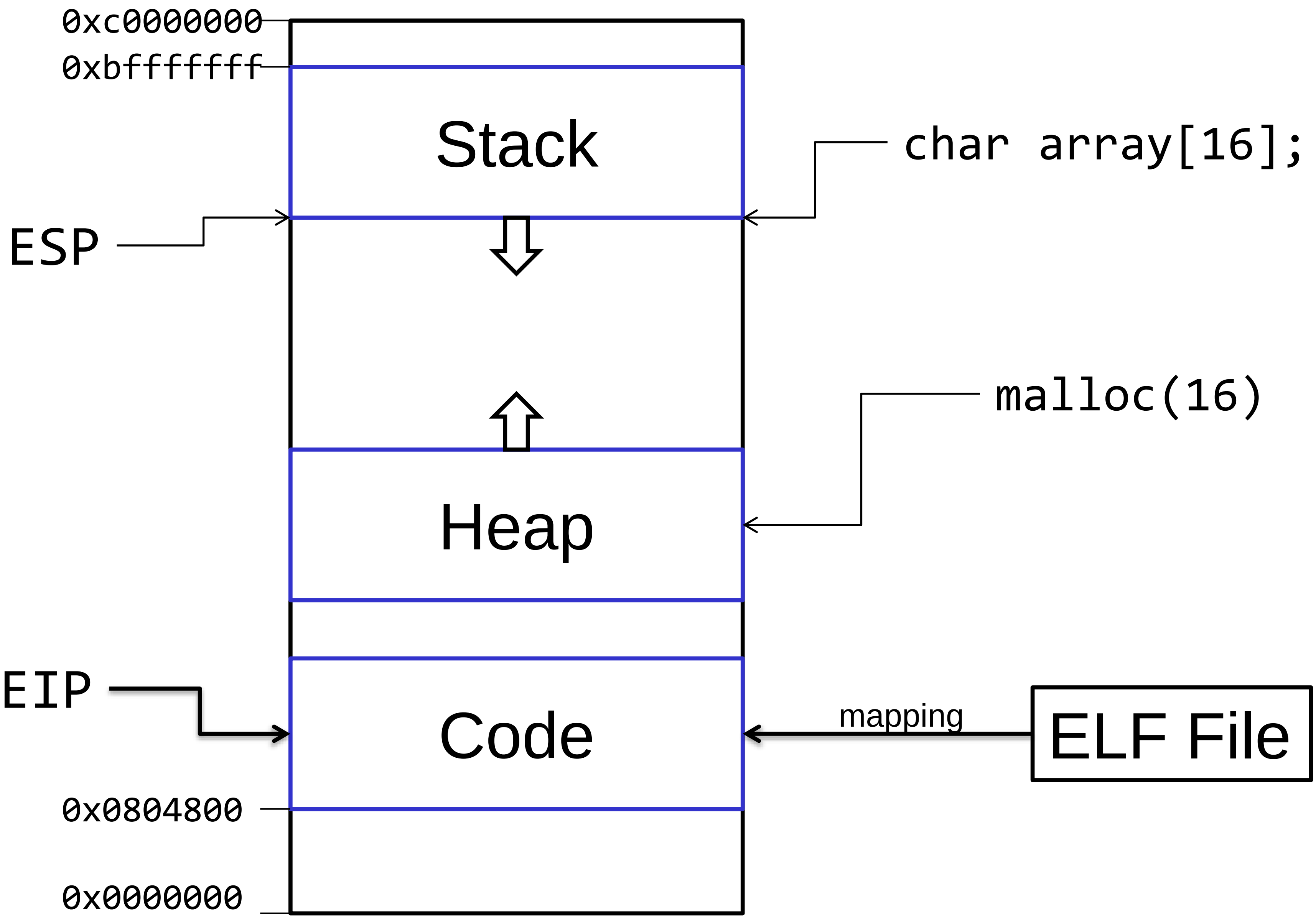
OS Overview

Recap:

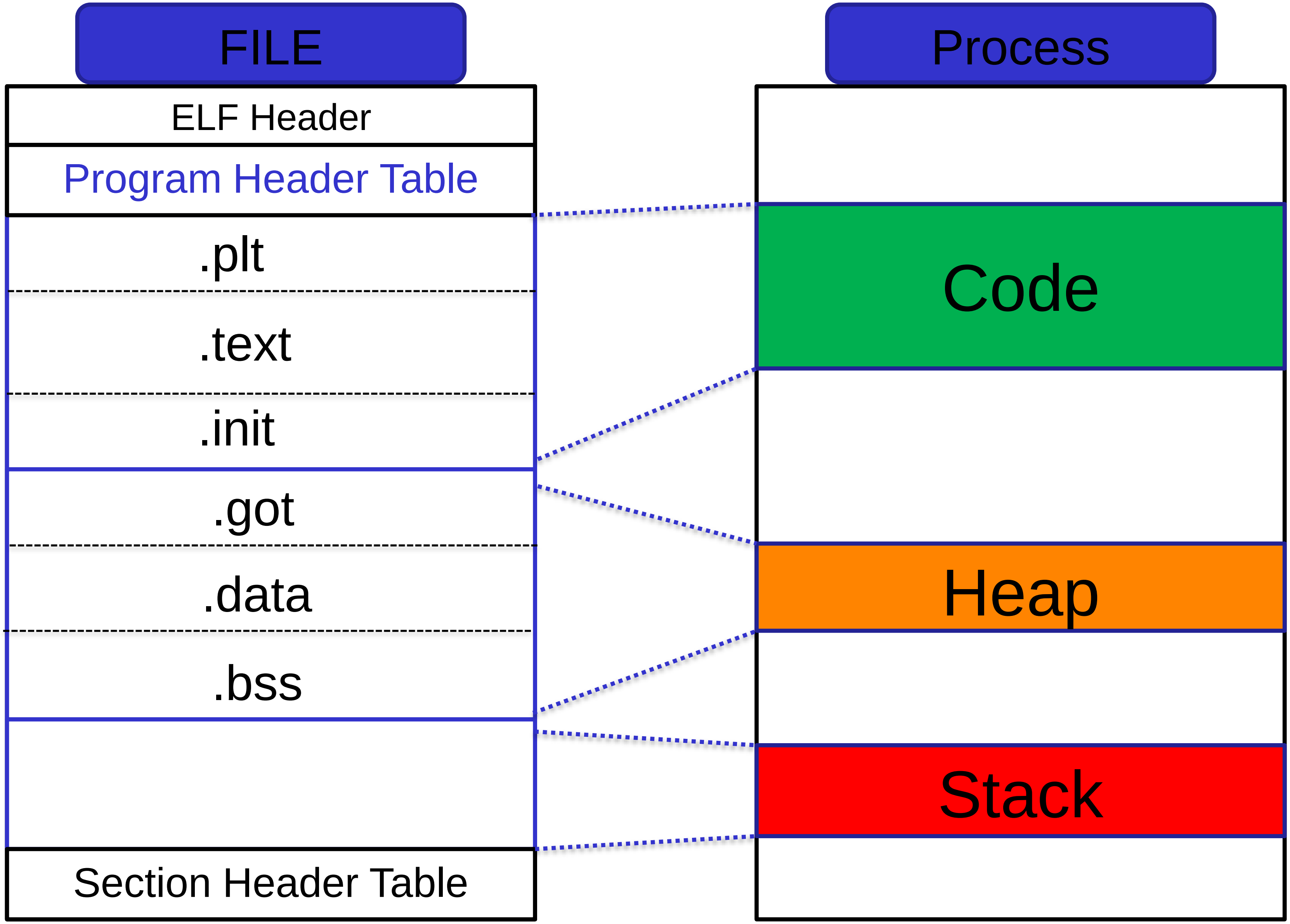
- Processes are programs which are alive in the RAM
- Every process thinks he owns the computer (including all the RAM)
- Every process has access to 2^32 (~4 billion) memory locations of 1 byte size

0x11_MemoryLayout

x32 Memory Layout



ELF Format


FILE
ELF Header
Program Header Table
.plt
.text
.init
.got
.data
.bss
Section Header Table
Process
Code
Heap
Stack

ELF Format

Recap:

- Program Code is stored in ELF Files
- ELF Files contain segments
- Segments are copied 1:1 in the memory to create a process (of that program)
- A process has generally three important segments:
  - Code segment (the actual compiled code)
  - Heap (global allocations with malloc())
  - Stack (local variables of functions)