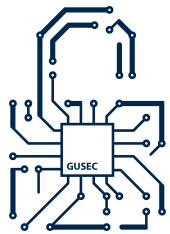
Introduction to x86 Assembly

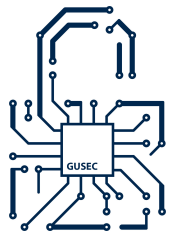
Joe Rose

GUSEC

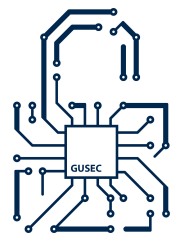
December 17, 2022



Outline of the Workshop

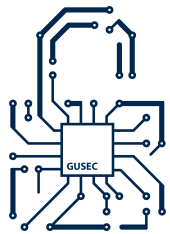


Outline of the Workshop

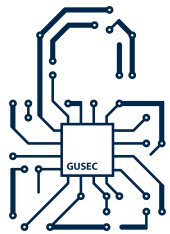


Who am I?

- 3rd Year Comp. Sci. Student
- Secretary of GUSEC
- Big nerd

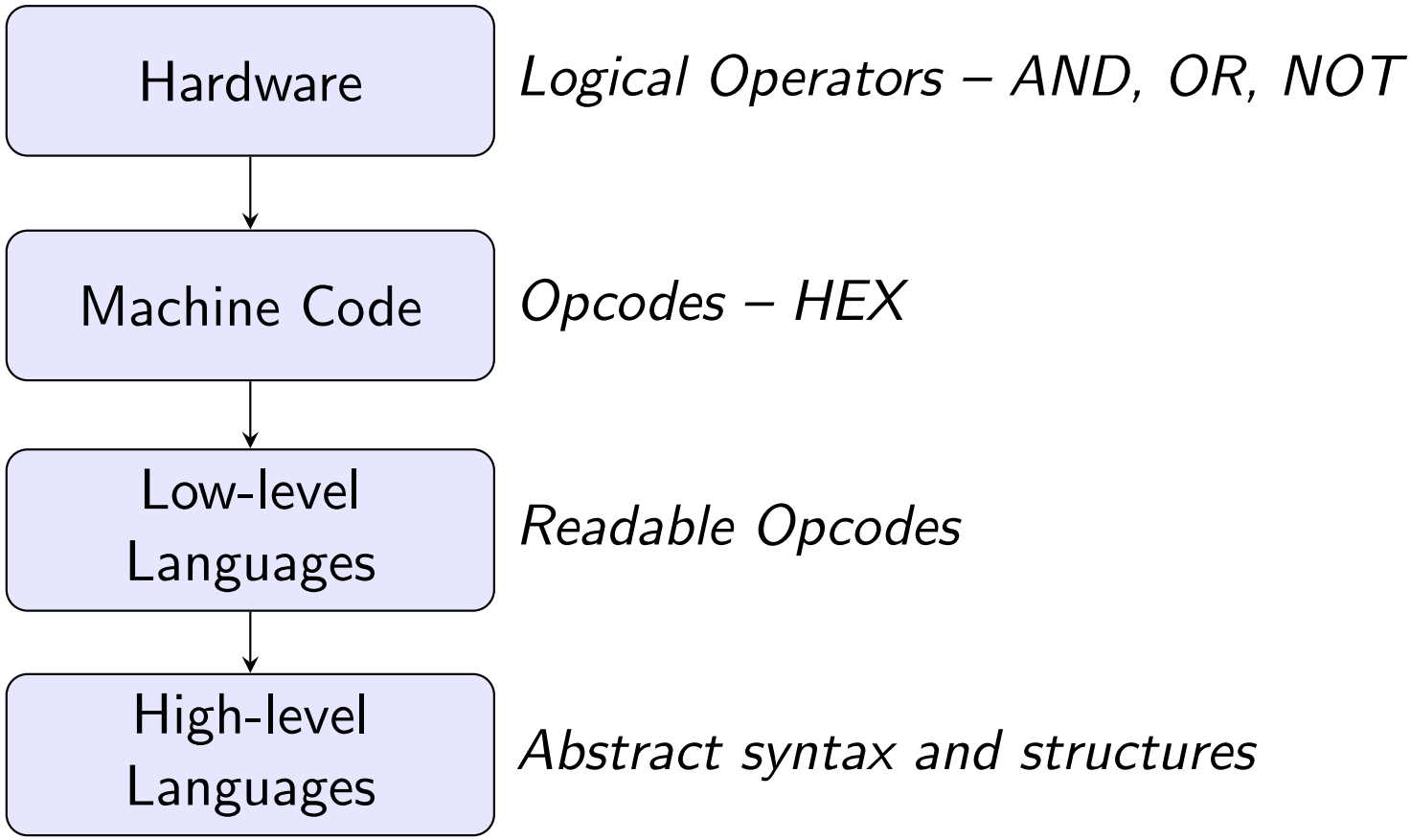


Outline of the Workshop



Abstraction

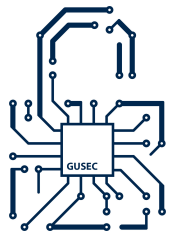
Modern computer systems can be represented as layers of *abstraction*. As software engineers, we generally work in a 'high level' highly abstracted space, which is then converted by our compiler into low level *machine code*.



x86 Memory

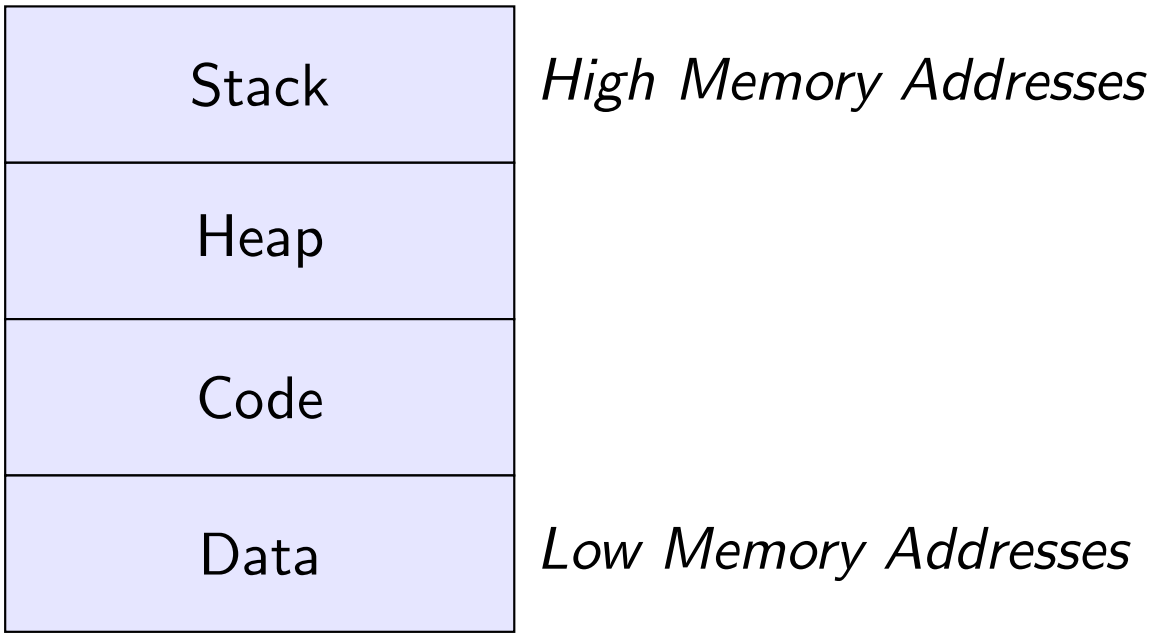
Memory, as it has come to be defined in some current literature, is somewhat of an abstract concept, this is because understanding memory is **really** complex. In this workshop, we're going to be using a *flat memory* model, which is basically just thinking of memory as like a Hash. Each memory location being an 8-**byte** 'bucket' which is mapped to by a 32-**bit** memory address. This allows for a possbile $2^{32} = 4294967295$ possible addresses.

As an aside, this is why 32-bit computers can only really address up to 4GiB of RAM, and FAT32 formatted USB drives can only store up to 4GiB files.



Memory for a Process

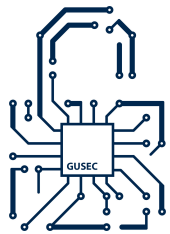
Every process running at a given time is carved out a little piece of virtual memory. It follows roughly this construction



The Registers

In low-level languages we deal with the regsiters. Registers are extremely small (32-**bit**) pieces of memory within your CPU which are quicker than RAM or cache to access. These are what store the pieces of data currently being worked with. The x86 specifies some general purpose registers and some dedicated registers. There are 16 registers in total, but here are some of the most important.

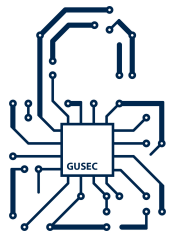
Register	**Purpose**
EAX	Accumulator
ECX	Counter in loops
ESI	Source in string & memory operations
EDI	Destination in string & memory operations
EBP	Stack base pointer
ESP	Stack Pointer
EFLAGS	Collection of flags



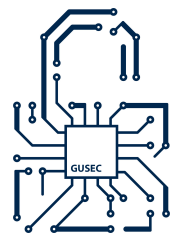
Flags

These are simple way of the computer keeping tracking its current state. All the flags are stored in the EFLAGS register and are set automatically when certain conditions are met. Here is a list of the most important:

Bit	Mnemonic	Usage
0	CF	Carry Flag
2	PF	Parity Flag
6	ZF	Zero Flag
7	SF	Sign Flag
11	OF	Overflow Flag



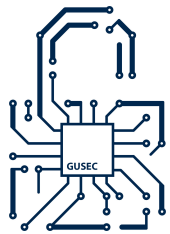
Outline of the Workshop



Data Types

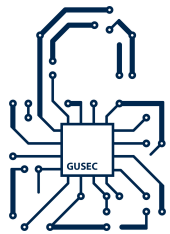
There aren't the traditional, high level data types that we might expect from languages like C and Python. Instead, our data types consist of a certain amout of arbitrary bits:

Name	**Amount of Bits**
Byte	8
Word	16
Double Word	32
Quad Word	64

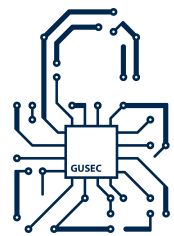


Structure

We're going to be looking at the *Intel* notation for assembly language, this is it's what I learned, it's the most common notation and it's (in my opinion) more clear.



Outline of the Workshop



Sources

These are the books I got the content for this workshop from. However, there is a huge amount of free literature on x86 and these are by no means necessary for you to learn the ropes.

- *Practical Malware Analysis* - Chapter 4. A Crash Course in x86 Disassembly
- *Practical Reverse Engineering* - Chapter 1. x86 and x64
- *Practical Binary Analysis* - Appendix A. A Crash Course in x86 Disassembly
- *Essentials of 80x86 assembly language*