**CPU Registers & Flags**

**What is a register?**

Think of it as a variable, specific to the CPU

CPU only has a set number of registers

* Only so much data that can be physically held in each one of these

Each have their own use and function for the CPU to use

* Can manipulate register as you want
* But putting wrong thing in wrong register can cause the system to crash and reboot

**Registers (32-apps)**

* EAX
* EBX
* ECX
* EDX
* ESI
* EDI
* EBP
* ESP
* EIP

EAX

* Most commonly used register (storage) for a return value
  + Process a function, the return value is: True,
  + True is stored in EAX

EBX

* No special purpose specifically
* Most commonly used to hold a value while performing calculations
* This will commonly hold the value of 0 for quick access to that value
  + Takes 3 op calls just to get to 0 so it’s easier and quicker just to make one call to access EBX to get 0 for the calculation

ECX

* Counting register
* When you perform a loop, this will often be the iterator that’s used
  + Variable (i)
* Can be used to count up to a digit or down to 0

**EDX**

* Holds parameters passed by functions
* Hold temporary data – related to functions

**ESI**

* Often used as a pointer
* Points to source of data (memory addr) so it can be referenced without changing it

**EDI**

* Similar to ESI but points to actual destination of data, so it can be modified

**EBP**

* Can be used for temporary storage
* Used commonly with compilers
  + Keeps track of stack pointer when a compiled application is not set to optimised
  + When it is set to optimised, it will not be used for any compilation of an application

**ESP**

* Special purpose register
* Points to top of stack at that point in time
* ESP isn’t normally changed unless required to
* Is not normally used for temporary storage
* When manipulating a binary to create your own backdoors keeping this value will be important

**EIP**

* Instruction pointer
  + Most important
* Whatever is in this register is what will be executed next
* This is commonly the target of memory based exploits
* Controlling this register often means a compromise of the system

**High and Low Registers**

Each register is 32-bits

There a 3 sub-versions of each register

Using EAX as an example:

* EAX can hold a full 32-bit value
* AX: only handles the last 26
* AL: handles the lower 8 bits
* AH: handles the bits 8-15

Important to understand sub-registers as you may not want to use the whole 32 bits for something that sonly 5 bits

Assembly uses sub-register spaces, which is important because when doing multiplication or division, your remainder will end up in AL or AH and not EAX

**CPU Flags**

**What are flags?**

Special type of register that holds a bitmap of data

Holds a simple 1 or 0 in a specific bit that will change what the CPU is doing

Understanding these are important when you are working with AMS instructions

**Common flags**

* Carry Flag
  + Commonly used for division, when there is a carry or remainder, this will be flagghed
* Zero Flag
  + This is used within division and other arithmetic operations
  + Used when controlling the flow of logic
  + Flags when the return value is 0
* Overflow Flag
  + When something doesn’t fit into the 32-bit register, it will be overflown into the next register