**The Stack & The Heap**

**The Stack**

Graphical user interface, application, Word

Description automatically generated

**What is the Stack?**

A special place in memory that holds temporary data

FIFO structure

* First in, first out
* When two things are pushed on to the stack, you cant get to the second one without taking the first one down first
* Can’t just reach in and grab something 5 items deep into the stack

Can hold ASM instructions, digits and values, even registers – can hold almost anything

This is why ESP is important because you want to keep track of where the top of this stack is so you know where to start pulling from.

**Functions**

* Core stack is used when app is run
* When an app calls a function and creates local variables, they get placed on the stack
* Because these variables are local to that function, it will remove them all off the stack when the function returns
  + Stack will return to ESP state before function, will pop everything off stack to return to state of before function was called
  + Local variables outside of functions don’t get saved

**Endianness**

Big-Endian

* Highes address is at the bottom of the stack

Little-Endian

* Lowest address is at the top of the stack

**Little Endian**

* Most commonly used in OS
* This is required when working with memory as everything is reversed in memory

Graphical user interface, application, Word

Description automatically generated

**The Heap**

**What is the heap?**

The stack takes standard types of data like integers

The heap will handle objects and larger amounts of data

Stack has to keep a neat order of FIFO

The Heap has no particular order

* Completely down to the developer

When an application needs to create something while its running, it will do so in the heap

* This memory in location is managed entirely by developer
* Easy to fragment memory and cause it to fill up
* Anything put in the heap must be removed after the fact, otherwise app will use too much memory, it’ll run out of available memory and the kernel will execute the OMM killer and kill the process