For next day – Isaac Shoebottom

Regions of memory:

Executable: This stores the executable code section of the program. Mostly assembly
Data: Strings and literal data are stored here. If you write a printf() with a string literal that character data is stored here.
Heap: This is where dynamically allocated variables are stored. Any time you use malloc(), the amount you allocate is stored here.
Stack: This is where local variables are stored. Any time you declare an int or a char or any primitive, it goes here, assuming it is in a function.

How the stack works:

The stack works in the way that it sounds, each new stack frame is put on top of each other, being popped off of it when they are done. When a function is called, it gets a return value, that tells it where to go when the function is finished. When the function finished the stack frame is removed and the address pointer is moved down. In the stack, generally variables are stored starting at locations higher in the address space. For example, if you were to assign 4 chars to an address space of 0xFFFF it would go 0xFFFF to 0xFFFE to 0xFFFD to 0xFFFC. Each of these addresses would hold a char's size (1 byte). After the program ends the stack is eventually deallocated by the operating system.

Stack frame of program:

Frame	Symbol	Address	Value
main	iArr	0xFFE3	garbage
	d	0xFFF3	garbage
	i	0xFFFB	garbage
	c	0xFFFF	garbage