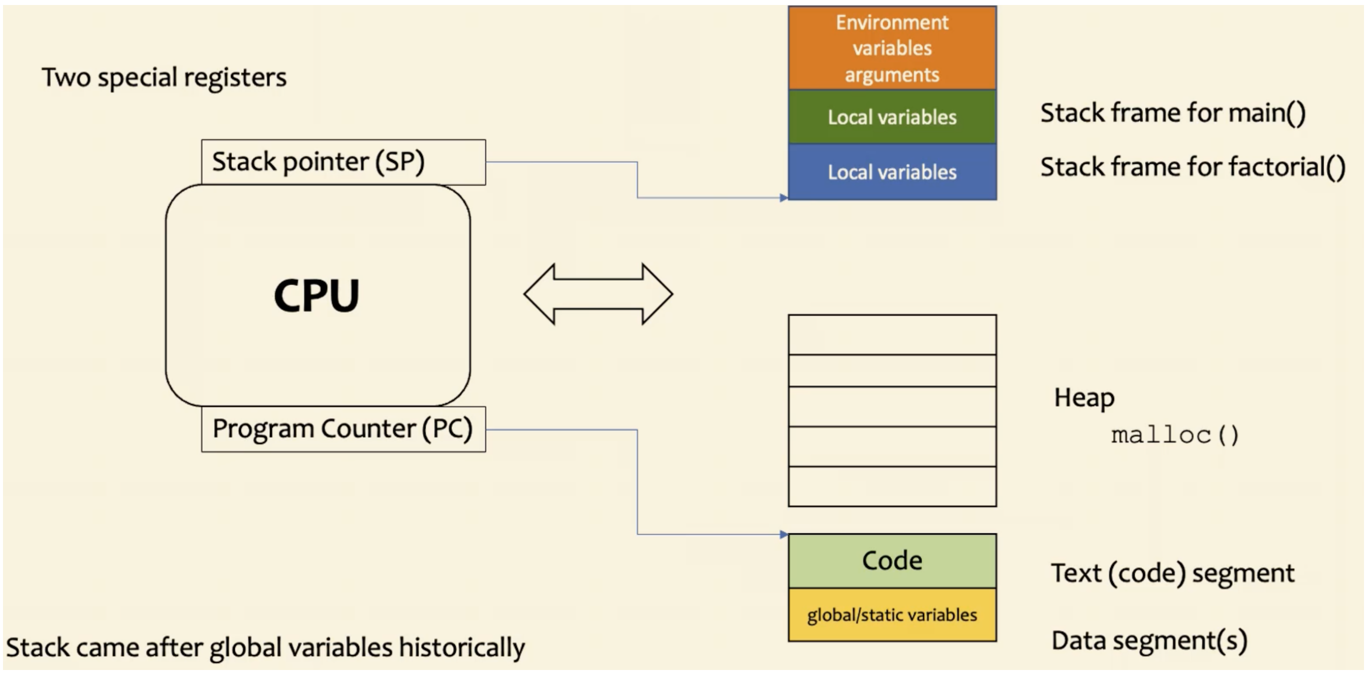
NOTES: C Inner Workings

Inner Workings of C



C Abstract Machine

Memory

- Special registers
  - PC (program counter) is stored in a register; it points to whichever instruction in memory is next to be executed
  - Stack pointer points to the top of the stack memory
- Data is aligned in C; which means, all addresses are a multiple of K bytes
  - `char`: 1 byte
  - `short`: 2 bytes
  - `int`, `float`, `char*`, etc: 4 bytes
  - `double`: 8 bytes
  - `long double`: 12 bytes

> **Note: Structs**
>
> - Structs can have members of different data types, so they > have to be aligned, where each member is "padded" so it starts at a multiple of the size itself
>   - For example, ints need to start at multiples of 4, and (in Windows) doubles need to start at multiples of 8
> - In addition, the struct itself must be aligned to *K* and must be *K* bytes long, where *K* is the size of the largest member
> - Linux and Windows treat padding differently, for example in Linux, `double` is treated as a 4-byte data type, but on Windows, it's an 8-byte data type

Stack

- Stack is used for local variables, and is taken care of by the compiler
- Local variables and function contexts are all stored on the stack
  - When you declare a local variable, an entry is pushed; similarly, when you call a function, the caller function context is pushed
- Stack Frame/Activation Record
  - When a function is called, a chunk of stack is allocated, in which all local variables go

Heap

- Heap is dynamically-allocated variables (malloc), is handled by the programmer
- Heap variables allow you to share variables between functions, since they don't have to be pushed/popped on the stack every time a function is called

Global Variables and Code

- Code and Global/Static Variables go in memory, independent of the stack or heap
  - (due to historical reasons—you don't know how many stack frames there are, so they can't go at the bottom of the stack)

Variables

- Type of a variable defines its size in memory, and the semantics (like what operations you can run)
- Value of variable is the stored value in memory
- Note: Big Endian vs. Little Endian
- Arrays: consecutive blocks of memory are allocated
  - Out-of-range behavior is undefined—you can crash the program or get garbage if you get an index out of bounds
  - Note: 2D arrays
- Structs: contiguously-allocated region of memory

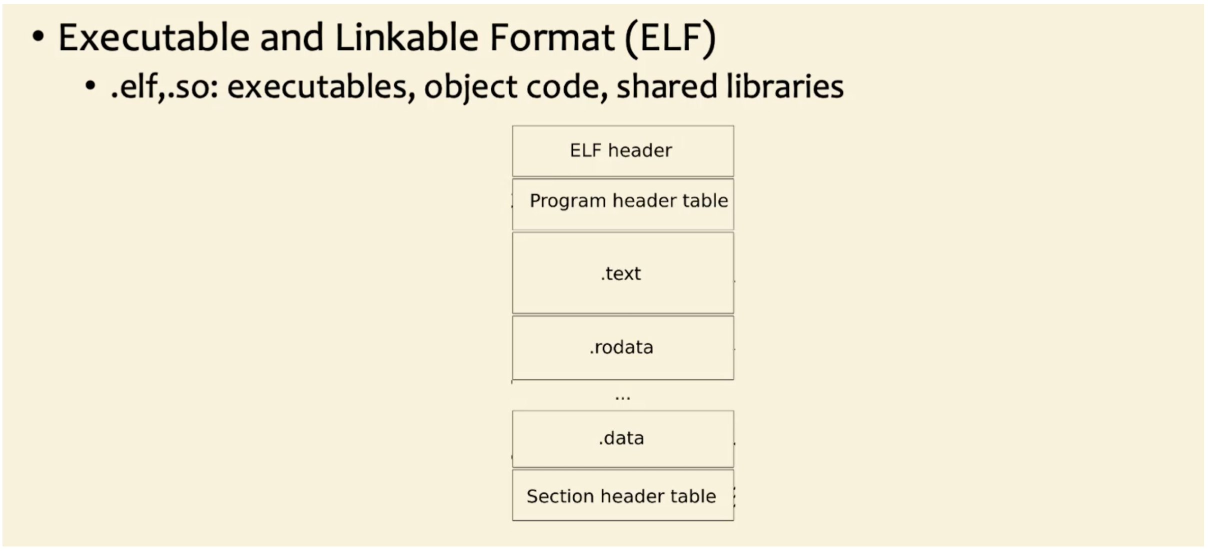
Role of an Operating System

- A memory map is the specification of where RAM, I/O, etc are in memory
- Operating systems abstract the physical memory map; as compared to baremetal programming, Linux et al give you a virtualized address space

Executable Format

- ELF (Executable and Linkable Format) specifies the headers for any executable file
  - Tells the OS how to load into virtual address space

- Section Header Table tells the linker how to link, including relocating memory references

  - Executable and Linkable Format (ELF)
    - .elf,.so: executables, object code, shared libraries

- BIN format is raw binary (no symbols, no debug info)
  - BIN is much smaller than ELF
  - Must be loaded into a specific memory address as assumed by the compiler/linker