NOTES: Software Basics

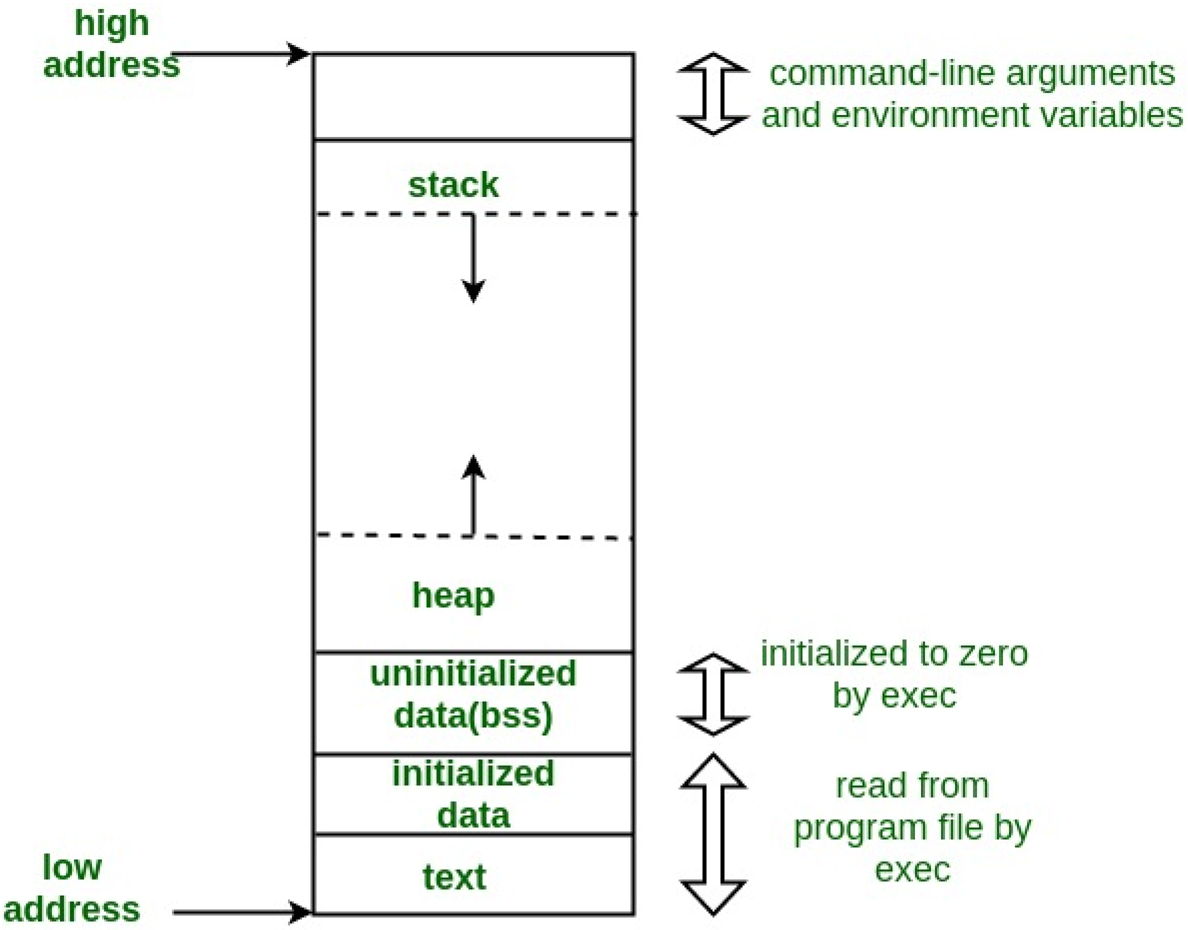
Software Basics

Operations

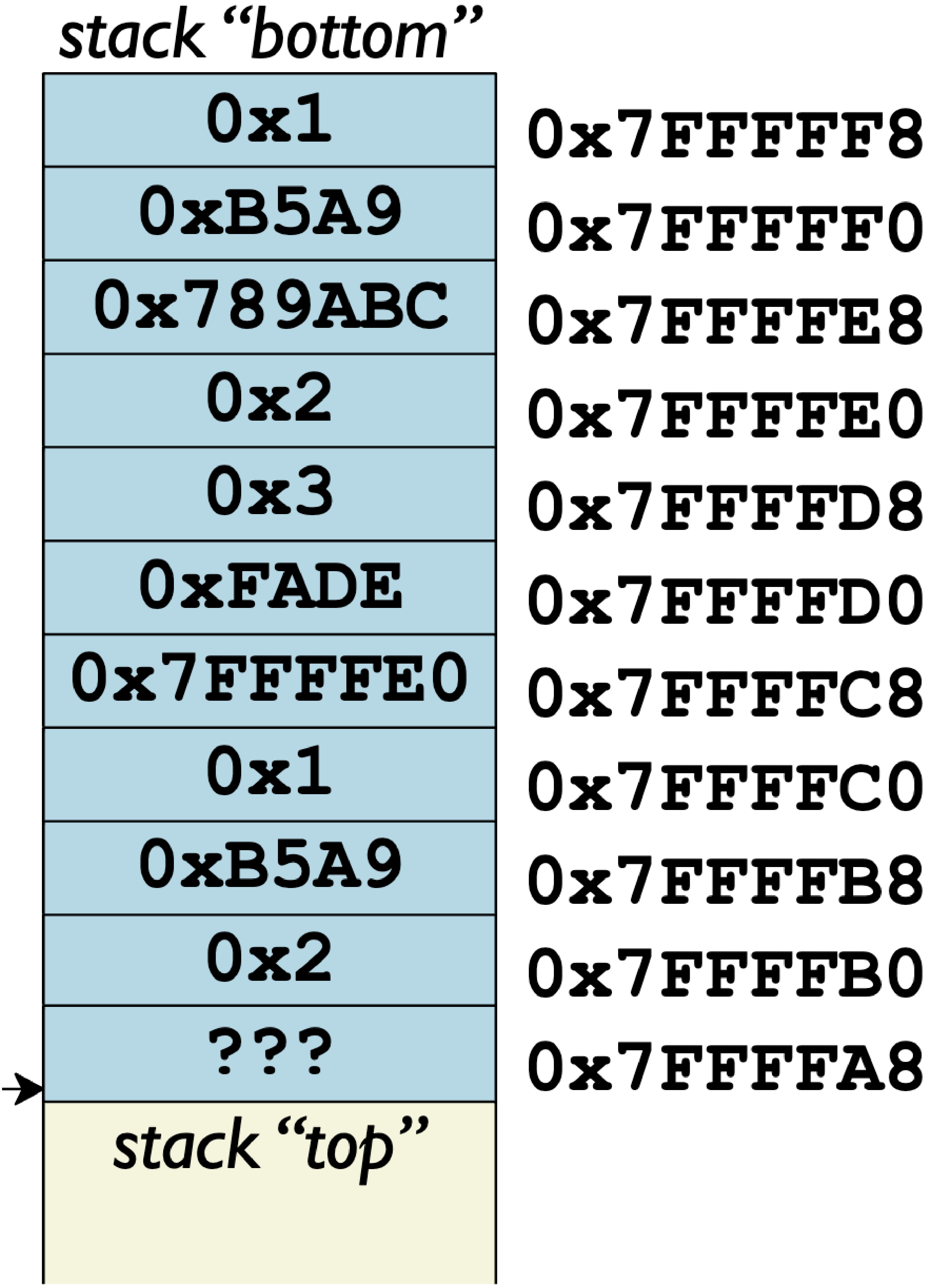
- A program is a list of instructions
    - The opcode and data bytes are passed to the ALU (Arithmetic Logic Unit)

Memory

- Output from each instruction is redirected to memory, which is then fed back into the CPU
- **Von Neumann Architecture**: the program data is stored in the same place as memory
- DRAM is cheap, but slow; flip-flops are expensive, but fast. So, there are Flip Flop registers for fast access
- SRAM (between speed and cost) is used for a cache


high address
command-line arguments and environment variables
stack
heap
uninitialized data(bss)
initialized to zero by exec
initialized data
text
read from program file by exec
low address

Stack



stack “bottom”

0x1	0x7FFFFF8
0xB5A9	0x7FFFFF0
0x789ABC	0x7FFFFE8
0x2	0x7FFFFE0
0x3	0x7FFFFD8
0xFADE	0x7FFFFD0
0x7FFFFE0	0x7FFFFC8
0x1	0x7FFFFC0
0xB5A9	0x7FFFFB8
0x2	0x7FFFFB0
???	0x7FFFFA8

stack “top”

- The stack grows towards lower addresses
- Stack frames are used to store local variables and function arguments

- The stack pointer, `%rsp`, is used to keep track of the top of the stack

Heap

- The heap is used for dynamic memory allocation

Data

- This is where global and static variables that are initialized are stored
- There is a read-only segment (for things like const string definitions), and a read-write segment (for things like global variables that can be changed)

BSS

- "Block Started by Symbol"
- This is where global and static variables that are uninitialized or set to zero are stored
- This segment is zeroed out by the OS on program start

Text

- This is where the program is stored

Architectures

- Load-store architectures mean that instructions can only access registers; only special commands (`load` and `store`) can access memory
- Architectures (like ARM and x86) define the instruction set
  - Micro-architectures (like Cortex-A53) define the pipeline, layout, size of caches, and other small details
- RISC vs CISC
  - RISC (Reduced Instruction Set Computing) has a small number of more simple instructions.
    - Load & store
    - Numerous registers
    - Constant-length instructions
    - Less compact code and simple hardware, but complex compiler
    - Like ARM, RISC-V
  - CISC (Complex Instruction Set Computing) has specialized instructions.
    - Direct memory operation
    - Few fegisters
    - Variable-length instructions
    - Compact code, but complex hardware
    - Like x86