## Roadmap

#### C:

# car \*c = malloc(sizeof(car)); c->miles = 100; c->gals = 17; float mpg = get\_mpg(c); free(c);

#### Java:

```
Car c = new Car();
c.setMiles(100);
c.setGals(17);
float mpg =
    c.getMPG();
```

# Assembly language:

```
get mpg:
   pushq %rbp
   movq %rsp, %rbp
   ...
   popq %rbp
   ret
```

OS:

Memory & data
Integers & floats
Machine code & C
x86 assembly
Procedures & stacks
Arrays & structs
Memory & caches
Processes
Virtual memory
Memory allocation
Java vs. C

# Machine code:



# Computer system:



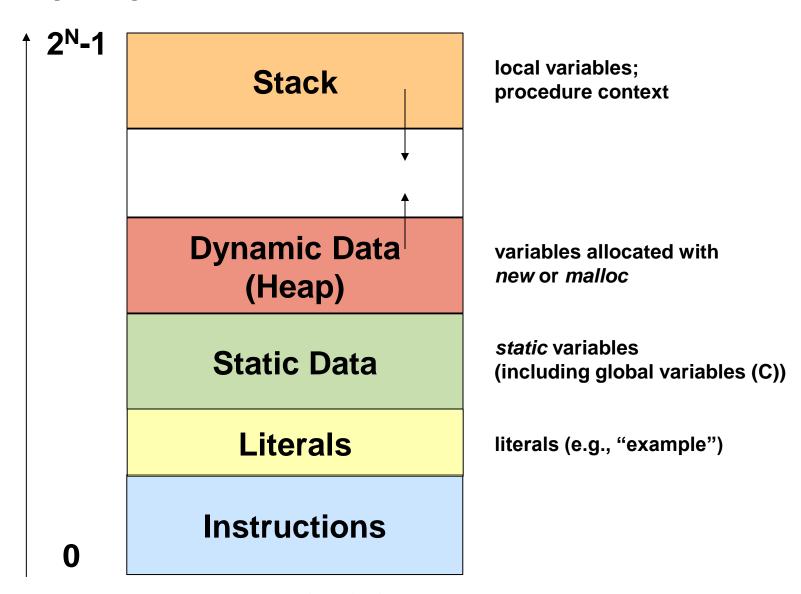




#### **Section 5: Procedures & Stacks**

- Stacks in memory and stack operations
- The stack used to keep track of procedure calls
- Return addresses and return values
- Stack-based languages
- The Linux stack frame
- Passing arguments on the stack
- Allocating local variables on the stack
- Register-saving conventions
- Procedures and stacks on x64 architecture

## **Memory Layout**



## **Memory Layout**

writable; not executable

Stack

Managed "automatically" (by compiler)

writable; not executable

(Heap)

**Dynamic Data** 

Managed by programmer

writable; not executable

**Static Data** 

Initialized when process starts

Read-only; not executable

Literals

Initialized when process starts

Read-only; executable

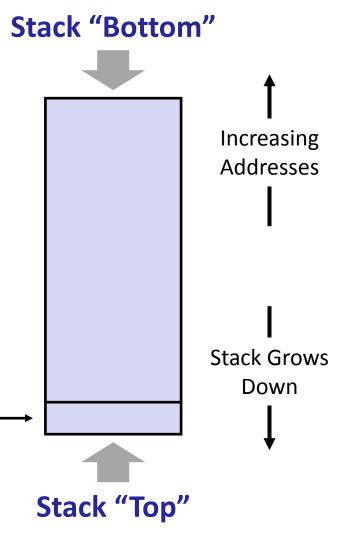
Instructions

Initialized when process starts

#### **IA32 Call Stack**

- Region of memory managed with a stack "discipline"
- Grows toward lower addresses
- Customarily shown "upside-down"
- Register %esp contains lowest stack address= address of "top" element

Stack Pointer: %esp



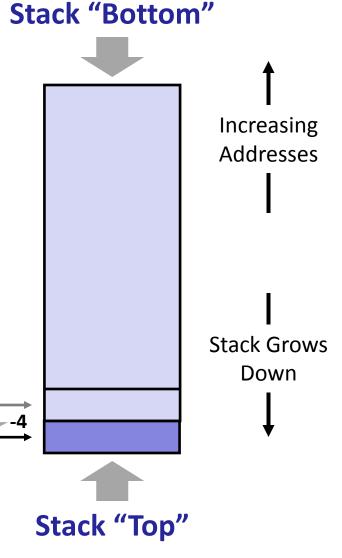
#### **IA32 Call Stack: Push**

Stack "Bottom" ■ pushl Src **Increasing** Addresses **Stack Grows** Down **Stack Pointer: %esp** Stack "Top"

#### **IA32 Call Stack: Push**

#### ■ pushl Src

- Fetch value from Src
- Decrement %esp by 4 (why 4?)
- Store value at address given by %esp



Stack Pointer: %esp

## **IA32 Call Stack: Pop**

Stack "Bottom" ■ popl Dest **Increasing** Addresses **Stack Grows** Down **Stack Pointer: %esp** Stack "Top"

Increasing

## **IA32 Call Stack: Pop**

#### popl Dest

- Load value from address %esp
- Write value to Dest
- Increment %esp by 4

Addresses **Stack Grows Stack Pointer: %esp** Down Stack "Top"

Stack "Bottom"