Y86 Calling Conventions

- Topics
  - Calling conventions: how callers communicate information to callees
- Learning Objectives
  - Define calling convention.
  - Use y86 calling conventions.

Parameter Passing

- How do procedures pass arguments?
  1. On the stack
  2. In registers
  3. In registers and on the stack
- The rules that callers and callees use to communicate information are called **calling conventions** (this video).
- The structure we use to store this information is called a **stack frame** (the next video).

y86 Calling Conventions: Parameters

- Based mostly on the x86 (although not exactly)
- Return value goes in %rax.
- Arguments/parameters are passed in registers, in this order:
  - %rdi, %rsi, %rdx, %rcx, %r8, %r9
  - More than 6 arguments?
    - Push the remaining ones on the stack *in reverse order*!
- What if an argument is too big to fit in a register? (E.g., you are passing a struct?)
  - Pass the too-large argument on the stack
  - If there is more than one too-large argument, these arguments get pushed in reverse order (as do arguments beyond 6)

y86 Parameter Passing Example

- Consider a function with three arguments:
  - A: a quadword
  - B: a struct containing 2 quadwords
  - C: a quadword
- A is placed in register %rdi
- B is pushed onto the stack
- C is placed in register %rsi

- Note that you don't skip register %rsi, because there is a second argument that was placed on the stack.

y86 Calling Conventions: Register Usage

- %rsp is the stack pointer
- %rbp is the frame (base) pointer
- Caller saved registers
  - These are scratch registers – the callee is allowed to scribble all over them, so if the caller cares about their contents, the caller must save them.
  - All the argument and return registers plus %r10, %r11, so
    - %rax, %rdi, %rsi, %rdx, %rcx, %r8, %r9, %r10, %r11
- Callee saved registers
  - If the callee uses them, then the callee must restore the original values before returning.
  - %rbx, %rbp, %r12, %r13, %r14