CSE351: The HW/SW Interface

Calling Conventions in x86-64, Stack Frames, Phase 2 Demo
Lab 2 Hints

● Some useful GDB commands
  – disas : shows current function's disassembly
  – info registers : prints all register contents
  – x/s : examine string

● Remember, you can dereference addresses stored in registers.
  – ex. “print *(int *) $rsp”
Memory Layout

- Stack: local variables; procedure context
- Dynamic Data (Heap): variables allocated with `new` or `malloc`
- Static Data: `static` variables (including global variables (C))
- Literals: literals (e.g., “example”)
- Instructions
IA32 vs x86-64

- We teach both in lecture, there are many differences between the two types of assembly, but we will only use x86 for the labs.
- IA32 uses the stack consistently for argument passing, while x86-64 tries to avoid the stack as much as possible (Why would it be more advantageous to use registers instead of the stack?)
Calling Conventions IA32

- Parameter values are passed directly on the stack and make use of the %ebp register.
- Local variables are referenced in relation to the stack pointer
Disassembled *swap*

080483a4 <swap>:

```
080483a4:  55        push    %ebp
080483a5:  89 e5     mov      %esp, %ebp
080483a7:  53        push    %ebx
080483a8:  8b 55 08  mov      0x8(%ebp), %edx
080483ab:  8b 4d 0c  mov      0xc(%ebp), %ecx
080483ae:  8b 1a     mov      (%edx), %ebx
080483b0:  8b 01     mov      (%ecx), %eax
080483b2:  89 02     mov      %eax, (%edx)
080483b4:  89 19     mov      %ebx, (%ecx)
080483b6:  5b        pop      %ebx
080483b7:  c9        leave
080483b8:  c3        ret
```

Calling Code

```
08048409:  e8 96 ff ff ff  call 80483a4 <swap>
0804840e:  8b 45 f8     mov      0xfffffffff8(%ebp), %eax
```

relative address (little endian)
x86-64 swap + differences

• Notice how the stack isn't used at all

```assembly
movq (%rdi), %rdx
movq (%rsi), %rax
movq %rax, (%rdi)
movq %rdx, (%rsi)
ret
```
Another Example

• We'll go over Phase 2 of the bomb (under the assumption that anyone not done with Phase 2 should probably be worried)

• Illustrates how local arguments are stored and referenced.
Register Cheat Sheet x86-64

- Applicable to the labs, notice how %rbp isn’t used as a base pointer
- Caller Saved Registers:
  - %r10
  - %r11
  - All argument registers
- Callee Saved Registers
  - %rbx
  - %rbp
  - %r12-%r15
- Return Register: %rax
- Stack Pointer: %rsp
- Instruction Pointer: %rip
- Argument Registers:
  - %rdi, %rsi, %rdx, %rcx, %r8, %r9