

# x86-64 Assembly

CPSC 275  
Introduction to Computer Systems

## x86-64 Integer Registers

%rax	%eax	%r8	%r8d
%rbx	%ebx	%r9	%r9d
%rcx	%ecx	%r10	%r10d
%rdx	%edx	%r11	%r11d
%rsi	%esi	%r12	%r12d
%rdi	%edi	%r13	%r13d
%rsp	%esp	%r14	%r14d
%rbp	%ebp	%r15	%r15d

- Twice the number of registers
- Accessible as 8, 16, 32, 64 bits

## Usage Conventions

%rax	Return value	%r8	Argument #5
%rbx	Callee saved	%r9	Argument #6
%rcx	Argument #4	%r10	Caller saved
%rdx	Argument #3	%r11	Caller Saved
%rsi	Argument #2	%r12	Callee saved
%rdi	Argument #1	%r13	Callee saved
%rsp	Stack pointer	%r14	Callee saved
%rbp	Callee saved	%r15	Callee saved

## x86-64 Registers

- Arguments passed to functions via registers
  - If more than 6 integral parameters, then pass rest on stack
  - These registers can be used as caller-saved as well
- All references to stack frame via stack pointer
  - Eliminates need to update %ebp/%rbp
- Other Registers
  - 6 callee saved
  - 2 caller saved
  - 1 return value (also usable as caller saved)
  - 1 special (stack pointer)

## x86-64 Long Swap

```
void swap_l(long *xp, long *yp)
{
    long t0 = *xp;
    long t1 = *yp;
    *xp = t1;
    *yp = t0;
}

swap:
    movq    (%rdi), %rdx
    movq    (%rsi), %rax
    movq    %rax, (%rdi)
    movq    %rdx, (%rsi)
    ret
```

- Operands passed in registers
  - First (**x**p) in %rdi, second (**y**p) in %rsi
  - 64-bit pointers
- No stack operations required (except ret)
- Avoiding stack
  - Can hold all local information in registers