

Assembly

Review

Register sizes

Operand Restrictions

Review

How does assembly relate to low-level machine code?

How does assembly relate to a high-level language like C code?

Give some examples of x86_64 assembly instructions.

What types of operands can an assembly instruction have in x86_64?

x86_64 instructions

Label the information for the following instructions in terms of address, machine code instruction, op code, and operands

114d: 55 push %rbp

114e: 48 89 e5 mov %rsp,%rbp

1151: 89 7d fc mov %edi,-0x4(%rbp)

x86_64 registers and mechanisms for accessing lower bytes (from *Dive into Systems*)

64-bit Register	32-bit Register	Lower 16 Bits	Lower 8 Bits
%rax	%eax	%ax	%al
%rbx	%ebx	%bx	%bl
%rcx	%ecx	%cx	%cl
%rdx	%edx	%dx	%dl
%rdi	%edi	%di	%dil
%rsi	%esi	%si	%sil
%rsp	%esp	%sp	%spl
%rbp	%ebp	%bp	%bpl
%r8	%r8d	%r8w	%r8b
%r9	%r9d	%r9w	%r9b
%r10	%r10d	%r10w	%r10b
%r11	%r11d	%r11w	%r11b
%r12	%r12d	%r12w	%r12b
%r13	%r13d	%r13w	%r13b
%r14	%r14d	%r14w	%r14b
%r15	%r15d	%r15w	%r15b

Exercise: What are the capacities of these registers?

`%edi`

`%rdi`

`%al`

`%r8w`

Operand Restrictions

- Constant forms cannot serve as destination operands.
- Memory forms cannot serve as *both* the source and destination operand in a single instruction.
- In cases of scaling operations “-0x4(%rbp, %rax, 8)”, the scaling factor is a third parameter in the parentheses. Scaling factors can be one of 1, 2, 4, or 8.

Exercise: Which of these instructions are valid?

`add %rax, (%rbx)`

`add (%rax), %rbx`

`add %rax, $8`

`add (%rax, %rbx, 10), %rcx`

`add (%rax), (%rbx)`

Review: Assembly commands for function calling

Table 2. Stack Management Instructions

Instruction	Translation
<code>push S</code>	Pushes a copy of S onto the top of the stack. Equivalent to: <pre>sub \$0x8, %rsp mov S, (%rsp)</pre>
<code>pop D</code>	Pops the top element off the stack and places it in location D. Equivalent to: <pre>mov (%rsp), D add \$0x8, %rsp</pre>

Review: Assembly commands for function returning

Table 1. Common Function Management Instructions

Instruction	Translation
<code>leaveq</code>	Prepares the stack for leaving a function. Equivalent to: <pre>mov %rbp, %rsp pop %rbp</pre>
<code>callq addr <fname></code>	Switches active frame to callee function. Equivalent to: <pre>push %rip mov addr, %rip</pre>
<code>retq</code>	Restores active frame to caller function. Equivalent to: <pre>pop %rip</pre>