

Type	Form	Operand value	Name
Immediate	$\$Imm$	Imm	Immediate
Register	r_a	$R[r_a]$	Register
Memory	Imm	$M[Imm]$	Absolute
Memory	(r_a)	$M[R[r_a]]$	Indirect
Memory	$Imm(r_b)$	$M[Imm + R[r_b]]$	Base + displacement
Memory	(r_b, r_i)	$M[R[r_b] + R[r_i]]$	Indexed
Memory	$Imm(r_b, r_i)$	$M[Imm + R[r_b] + R[r_i]]$	Indexed
Memory	$(, r_i, s)$	$M[R[r_i] \cdot s]$	Scaled indexed
Memory	$Imm(, r_i, s)$	$M[Imm + R[r_i] \cdot s]$	Scaled indexed
Memory	(r_b, r_i, s)	$M[R[r_b] + R[r_i] \cdot s]$	Scaled indexed
Memory	$Imm(r_b, r_i, s)$	$M[Imm + R[r_b] + R[r_i] \cdot s]$	Scaled indexed

Figure 3.3 Operand forms. Operands can denote immediate (constant) values, register values, or values from memory. The scaling factor s must be either 1, 2, 4, or 8.

63	31	15	7	
%rax	%eax	%ax	%al	return value
%rbx	%ebx	%bx	%bl	caller owned
%rcx	%ecx	%cx	%cl	4th argument
%rdx	%edx	%dx	%dl	3rd argument
%rsi	%esi	%si	%sil	2nd argument
%rdi	%edi	%di	%dil	1st argument

63	31	15	7	
%rbp	%ebp	%bp	%bpl	caller owned
%rsp	%esp	%sp	%spl	stack pointer
%r8	%r8d	%r8w	%r8b	5th argument
%r9	%r9d	%r9w	%r9b	6th argument
%r10	%r10d	%r10w	%r10b	callee owned
%r11	%r11d	%r11w	%r11b	callee owned

63	31	15	7	
%r12	%r12d	%r12w	%r12b	caller owned
%r13	%r13d	%r13w	%r13b	caller owned
%r14	%r14d	%r14w	%r14b	caller owned
%r15	%r15d	%r15w	%r15b	caller owned

The last column designates the standard programming conventions — we will get to that later, but it denotes how registers manage the stack, passing function arguments, returning from function calls, and storing local and temporary data.

- Because of its 16-bit origins, Intel uses "word" to mean 16-bits (two bytes)
- 32-bit words are referred to as "double words" ("d" suffix)
- 64-bit quantities are referred to as "quad words" ("q" suffix)
- This table shows the x86 primitive data types of C (Figure 3.1 in the textbook)

C declaration	Intel data type	Assembly-code suffix	Size (bytes)
char	Byte	b	1
short	Word	w	2
int	Double word	d	4
long	Quad word	q	8
char *	Quad word	q	8
float	Single precision	s	4
double	Double precision	l	8

- Notice:
 - Pointers are 8-byte quad words
 - The suffixes will become important very soon



```

mov_____ %eax, (%rsp)

mov_____ (%rax), %dx

mov_____ $0xFF, %bl

mov_____ (%rsp,%rdx,4), %dl

mov_____ (%rdx), %rax

mov_____ %dx, (%rax)

```

Answers:



Description

```

copy 4 bytes from %eax into memory at (%rsp)
copy 2 bytes from memory at (%rax) to %dx
set %bl to hold the 1-byte value $0xFF
copy 1 byte from memory at (%rsp + 4*%rdx) to %dl
copy 8 bytes from memory at (%rdx) to %rax
copy 2 bytes from %dx to memory at (%rax)

```

Instruction

```

movl %eax, (%rsp)
movw (%rax), %dx
movb $0xFF, %bl
movb (%rsp,%rdx,4), %dl
movq (%rdx), %rax
movw %dx, (%rax)

```