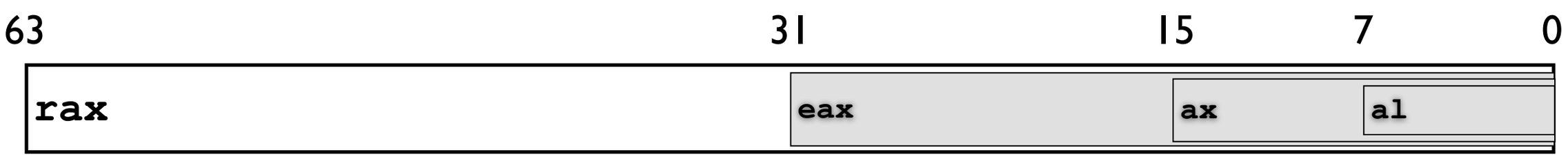
x86-64 registers

rax	**eax**	return value
rbx	**ebx**	callee saved
rcx	**ecx**	4th argument
rdx	**edx**	3rd argument
rsi	**esi**	2nd argument
rdi	**edi**	1st argument
rsp	**esp**	stack pointer
rbp	**ebp**	frame pointer

r8	**r8d**	5th argument
r9	**r9d**	6th argument
r10	**r10d**	caller saved
r11	**r11d**	caller saved
r12	**r12d**	callee saved
r13	**r13d**	callee saved
r14	**r14d**	callee saved
r15	**r15d**	callee saved



Moving data

movq *destination*, *source*

	Source	Destination	Example	C Analog
movq	Immediate	Register	movq rax, $0x1A	t = 0x1A;
		Memory	movq (rax), $42	*p = 42;
	Register	Register	movq rdx, rax	x = y;
		Memory	movq (rdx), rax	*p = y;
	Memory	Register	movq rdx, (rax)	x = *p;

Data types

C	Name	Suffix	Size (bytes)
char	Byte	b	1
short	Word	w	2
int	Double word	l	4
long	Quad word	q	8
char *	Quad word	q	8
float	Single precision	s	4
double	Double precision	l	8

Operand forms

type	form	value	name
immediate	$imm	imm	immediate
register	r_a	R[r_a]	register
memory	imm	M[imm]	absolute
memory	(r_a)	M[r_a]	indirect
memory	imm(r_a)	M[imm + r_a]	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] * s]	scaled indexed
memory	imm(, r_i, s)	M[imm + R[r_i] * s]	scaled indexed
memory	(r_b, r_i, s)	M[R[r_b] + R[r_i] * s]	scaled indexed
memory	imm(r_b, r_i, s)	M[imm + R[r_b] + R[r_i] * s]	scaled indexed

Integer arithmetic operators

Instruction		Effect	Description
lea	D, S	D ← &S	Load effective address
inc	D	D ← D+1	increment
dec	D	D ← D-1	decrement
neg	D	D ← -D	negate
not	D	D ← ~D	compliment
add	D, S	D ← D + S	add
sub	D, S	D ← D - S	subtract
imul	D, S	D ← D * S	integer multiply
xor	D, S	D ← D ^ S	exclusive or
or	D, S	D ← D | S	or
and	D, S	D ← D & S	and
sal	D, k	D ← D << k	left shift
shr	D, k	D ← D << k	left shift (same as sal)
sar	D, k	D ← D $>>_A$ k	arithmetic right shift
shr	D, k	D ← D $>>_L$ k	logical right shift

Condition codes

Flag	Meaning	Description
CF	Carry flag	There was a carry out from the last bit
ZF	Zero flag	The last result was zero
SF	Sign flag	The last result was negative
OF	Overflow flag	The last result had two's compliment overflow

Codes set by

- integer arithmetic operations except lea
- cmp *A*, *B*: subtracts B - A and doesn't store result
- test *A*, *B*: ANDs A & B and doesn't store result

Jump instructions

Instruction	Synonym	Condition	Description
jump *label*			Direct jump
jump *Operand*			Indirect jump
je *Label*	jz	ZF	Equal / zero
jne *Label*	jnz	~ZF	No equal/ not zero
js *Label*		SF	Negative
jns *Label*		~SF	Nonnegative
jg *Label*	*jnle*	~(SF ^ OF)& ~ZF	Greater than
jge *Label*	jnl	~(SF ^ OF)	Greater than or equal
jl *Label*	jnge	(SF ^ OF)	Less than
jle *Label*	*jng*	(SF ^ OF) | ZF	Less than or equal
ja *Label*	jnbe	~CF & ~ZF	Above (unsigned >)
jae *Label*	jnb	~CF	Above or equal
jb *Label*	jnae	CF	Below (unsigned <)
jbe *Label*	jna	CF | ZF	Below or equal