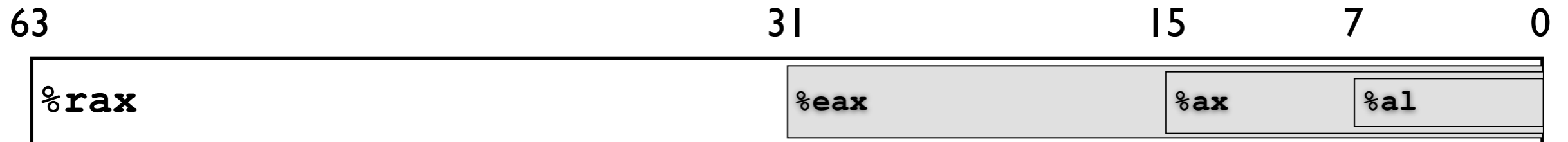
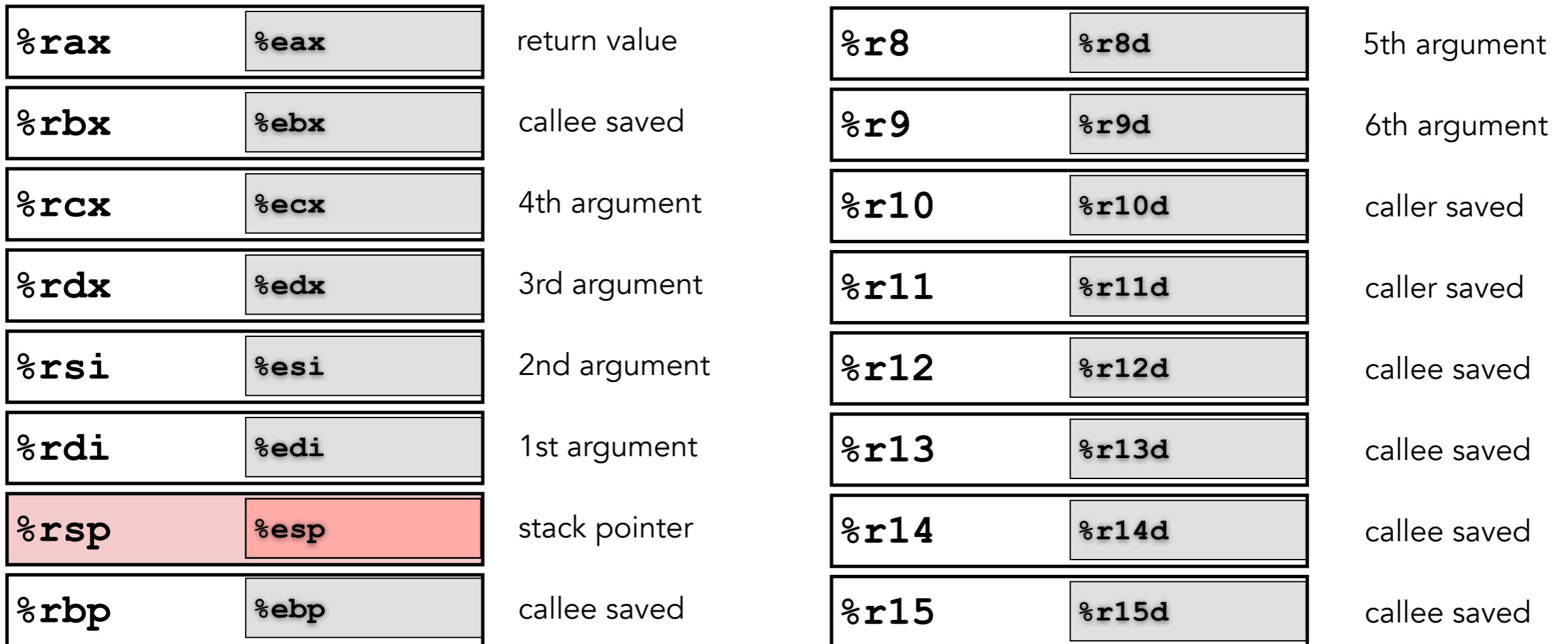


Data types

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

x86-64 registers



Our first instruction: Moving data

`movq source, destination`

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

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