

### Common register usage

Register	Special usage	Called function preserves contents
rax	1st function return value.	No
rbx	Optional base pointer.	Yes
rcx	Pass 4th argument to function.	No
rdx	Pass 3rd argument to function. 2nd function return value.	No
rsp	Stack pointer.	Yes
rbp	Optional frame pointer.	Yes
rdi	Pass 1st argument to function.	No
rsi	Pass 2nd argument to function.	No
r8	Pass 5th argument to function.	No
r9	Pass 6th argument to function.	No
r10	Pass function's static chain pointer.	No
r11		No
r12		Yes
r13		Yes
r14		Yes
r15		Yes

### Registers

bits 63-0	bits 31-0	bits 15-0	bits 15-8	bits 7-0
rax	eax	ax	ah	al
rbx	ebx	bx	bh	bl
rcx	ecx	cx	ch	cl
rdx	edx	dx	dh	dl
rsi	esi	si		sil
rdi	edi	di		dil
rbp	ebp	bp		bpL
rsp	esp	sp		spL
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

### Register argument order

Argument	Register
first	rdi
second	rsi
third	rdx
fourth	rcx
fifth	r8
sixth	r9

### Common Jumps

instruction	meaning	immediately after a cmp ...
ja	jump above	jump if destination is above source in sequence
jae	jump above or equal	jump if destination is above or in same place as source in sequence
jb	jump below	jump if destination is below source in sequence
jbe	jump below or equal	jump if destination is below or in same place as source in sequence

Table 10.2: Conditional jump instructions for unsigned values.

### More Common Jumps

instruction	meaning	immediately after a cmp ...
jg	jump greater	jump if destination is greater than source
jge	jump greater or equal	jump if destination is greater than or equal to source
jl	jump less	jump if destination is less than source
jle	jump less or equal	jump if destination is less than or equal to source

Table 10.3: Conditional jump instructions for signed values.

### Flow Control

program flow control:			
opcode	location	action	see page:
call	label	call function	173
iret		return from kernel function	388
ja	label	jump above (unsigned)	239
jae	label	jump above/equal (unsigned)	239
jb	label	jump below (unsigned)	239
jbe	label	jump below/equal (unsigned)	239
je	label	jump equal	239
jg	label	jump greater than (signed)	240
jge	label	jump greater than/equal (signed)	240
jl	label	jump less than (signed)	240
jle	label	jump less than/equal (signed)	240
jmp	label	jump	241
jne	label	jump not equal	239
jno	label	jump no overflow	239
joc	label	jump on condition codes	192
leave		undo stack frame	192
ret		return from function	192
syscall		call kernel function	201
sysret		return from kernel function	390

cc = condition codes

### Jump List

instruction	action	condition codes
ja	jump if above	$(CF = 0) \cdot (ZF = 0)$
jae	jump if above or equal	$CF = 0$
jb	jump if below	$CF = 1$
jbe	jump if below or equal	$(CF = 1) + (ZF = 1)$
jc	jump if carry	$CF = 1$
jcxz	jump if cx register zero	$ZF = 1$
jecxz	jump if ecx register zero	$(ZF = 0) \cdot (SF = OF)$
jrcxz	jump if rcx register zero	$SF = OF$
je	jump if equal	$SF \neq OF$
jg	jump if greater	$(ZF = 1) + (SF \neq OF)$
jge	jump if greater or equal	$(CF = 1) + (ZF = 1)$
jl	jump if less	$CF = 1$
jle	jump if less or equal	$CF = 0$
jna	jump if not above	$(CF = 0) \cdot (ZF = 0)$
jnae	jump if not above or equal	$ZF = 0$
jb	jump if not below	$CF = 1$
jnb	jump if not below or equal	$(CF = 0) \cdot (ZF = 0)$
jnc	jump if not carry	$CF = 0$
jne	jump if not equal	$ZF = 0$
jng	jump if not greater	$(ZF = 1) + (SF \neq OF)$
jnge	jump if not greater or equal	$SF \neq OF$
jnl	jump if not less	$SF = OF$
jnle	jump if not less or equal	$(ZF = 0) \cdot (SF = OF)$
jno	jump if not overflow	$OF = 0$
jnp	jump if not parity or equal	$PF = 0$
jns	jump if not sign	$SF = 0$
jnz	jump if not zero	$ZF = 0$
jo	jump if overflow	$OF = 1$
jp	jump if parity	$PF = 1$
jpe	jump if parity even	$PF = 1$
jpo	jump if parity odd	$PF = 0$
js	jump if sign	$SF = 1$
jz	jump if zero	$ZF = 1$

Table 10.1: Conditional jump instructions.

### Arithmetic functions

arithmetic/logic:				
opcode	source	destination	action	see page:
adds	$\text{Simm}/\text{sreg}$	$\text{sreg}/\text{mem}$	add	214
addb	$\text{mem}$	$\text{sreg}$	add	214
ands	$\text{Simm}/\text{sreg}$	$\text{sreg}/\text{mem}$	bit-wise and	290
andb	$\text{mem}$	$\text{sreg}$	bit-wise and	290
cmpr	$\text{Simm}/\text{sreg}$	$\text{sreg}/\text{mem}$	compare	237
cmpsb	$\text{mem}$	$\text{sreg}$	compare	237
dec	$\text{sreg}/\text{mem}$		decrement	249
div	$\text{sreg}/\text{mem}$		unsigned divide	315
idiv	$\text{sreg}/\text{mem}$		signed divide	317
imul	$\text{sreg}/\text{mem}$		signed multiply	310
inc	$\text{sreg}/\text{mem}$		increment	248
leaw	$\text{mem}$	$\text{sreg}$	load effective address	191
mul	$\text{sreg}/\text{mem}$		unsigned multiply	309
neg	$\text{sreg}/\text{mem}$		negate	322
or	$\text{Simm}/\text{sreg}$	$\text{sreg}/\text{mem}$	bit-wise inclusive or	290
orb	$\text{mem}$	$\text{sreg}$	bit-wise inclusive or	290
sals	$\text{Simm}/\text{scl}$	$\text{sreg}/\text{mem}$	shift arithmetic left	302
sars	$\text{Simm}/\text{scl}$	$\text{sreg}/\text{mem}$	shift arithmetic right	301
shl	$\text{Simm}/\text{scl}$	$\text{sreg}/\text{mem}$	shift left	302
shr	$\text{Simm}/\text{scl}$	$\text{sreg}/\text{mem}$	shift right	301
sub	$\text{Simm}/\text{sreg}$	$\text{sreg}/\text{mem}$	subtract	215
subb	$\text{mem}$	$\text{sreg}$	subtract	215
test	$\text{Simm}/\text{sreg}$	$\text{sreg}/\text{mem}$	test bits	238
testb	$\text{mem}$	$\text{sreg}$	test bits	238
xor	$\text{Simm}/\text{sreg}$	$\text{sreg}/\text{mem}$	bit-wise exclusive or	290
xorb	$\text{mem}$	$\text{sreg}$	bit-wise exclusive or	290

s = b, w, l, q; w = L, q

### x87 Floating Point

x87 floating point:				
opcode	source	destination	action	see page:
fadds	$\text{mem}/\text{float}$		add	373
faddp			add/pop	373
fchs			change sign	373
fcom	$\text{mem}/\text{float}$		compare	373
fcomp			compare/pop	373
fcos			cosine	373
fdiv	$\text{mem}/\text{float}$		divide	373
fdivp			divide/pop	373
fild	$\text{mem}/\text{int}$		load integer	373
fist	$\text{mem}/\text{int}$	$\text{mem}/\text{int}$	store integer	373
filds	$\text{mem}/\text{int}$	$\text{mem}/\text{int}$	load floating point	373
fml	$\text{mem}/\text{float}$		multiply	373
fmlp			multiply/pop	373
fsin			sine	373
fsqrt			square root	373
fsts	$\text{mem}/\text{float}$	$\text{mem}/\text{int}$	floating point store	373
fstps	$\text{mem}/\text{float}$		subtract	373
fstpb			subtract/pop	373

s = b, w, l, q; w = L, q

### SSE floating point

SSE floating point conversion:				
opcode	source	destination	action	see page:
cvtsd2si	$\text{simm}/\text{sreg}/\text{mem}$	$\text{sreg}$	scalar double to signed integer	368
cvtsd2ss	$\text{simm}/\text{sreg}$	$\text{simm}/\text{sreg}/\text{sreg}$	scalar double to single float	368
cvtsi2sd	$\text{sreg}$	$\text{simm}/\text{sreg}/\text{mem}$	signed integer to scalar double	368
cvtsi2sdq	$\text{sreg}$	$\text{simm}/\text{sreg}/\text{mem}$	signed integer to scalar double	368
cvtsi2ss	$\text{sreg}$	$\text{simm}/\text{sreg}/\text{mem}$	signed integer to scalar single	368
cvtsi2ssq	$\text{sreg}$	$\text{simm}/\text{sreg}/\text{mem}$	signed integer to scalar single	368
cvtsd2sd	$\text{simm}/\text{sreg}$	$\text{simm}/\text{sreg}/\text{mem}$	scalar single to scalar double	368
cvtsd2si	$\text{simm}/\text{sreg}/\text{mem}$	$\text{sreg}$	scalar single to signed integer	368
cvtsd2siq	$\text{simm}/\text{sreg}/\text{mem}$	$\text{sreg}$	scalar single to signed integer	368

