Instruction	Operand types	Description	Pseudocode
adc src, src_dest	R, R/M R/M, R I, R/M	add with carry	src_dest = src_dest + src + CF
add src, src_dest	R, R/M R/M, R I, R/M	arithmetic add	src_dest = src_dest + src
and src, src_dest	R, R/M R/M, R I, R/M	bitwise and	src_dest = src_dest and src
bt index, src	R, R/M I, R/M	bit test	CF = src[index]
btc index, src	R, R/M I, R/M	bit test and complement	CF = src[index] src[index] = not(src[index])
btr index, src	R, R/M I, R/M	bit test and reset	CF = src[index] src[index] = 0
bts index, src	R, R/M I, R/M	bit test and set	CF = src[index] src[index] = 1
call label	D	call procedure (relative)	push rip; goto label
call* address	R/M	call procedure (indirect)	push rip; goto address
clc		clear carry	CF = 0
cmp src1, src2	R, R/M R/M, R I, R/M	compare (sets flags based on src2 – src1)	
cnc		complement carry	CF = not(CF)
dec src_dest	R/M	decrement	src_dest = src_dest - 1
div divisor	R/M	divide	rax = rdx:rax/divisor rdx = remainder
idiv divisor	R/M	signed divide	rax = rdx:rax/divisor rdx = remainder
imul factor	R/M	signed multiply	rdx:rax = rax * factor
imul factor, src_dest	R/M, R	signed multiply	src_dest = src_dest * factor
imul factor, src, dest	I, R/M, R	signed multiply	dest = src * factor
inc src_dest	R/M	increment	src_dest = src_dest + 1
jb label **jc** label	D	conditional jump if below (for unsigned), if carry flag set	if CF=1: goto label
jae label **jnc** label	D	conditional jump if above or equal (for unsigned), if carry flag cleared	if CF=0: goto label
je label **jz** label	D	conditional jump if equal, if zero flag set	if ZF=1: goto label
jne label **jnz** label	D	conditional jump if not equal, if zero flag cleared	if ZF=0: goto label
jbe label	D	conditional jump if below or equal (for unsigned)	if CF=1 or ZF=1: goto label
ja label	D	conditional jump if above (for unsigned)	if CF=0 and ZF=0: goto label
jl label	D	conditional jump if less than (for signed)	if SF<>OF: goto label
jle label	D	conditional jump if less than or equal (for signed)	if ZF=1 or SF<>OF: goto label
jg label	D	conditional jump if greater than (for signed)	if ZF=0 and SF=OF: goto label
jge label	D	conditional jump if greater than or equal (for signed)	if SF=OF: goto label

Instruction	Operand types	Description	Pseudocode
js label	D	conditional jump if sign flag set	`if SF=1: goto label`
jo label	D	conditional jump if overflow flag set	`if OF=1: goto label`
jmp label	D	jump relative	`goto label`
jmp* address	R/M	jump indirect	`goto address`
lea src, dest	R/M, R	load effective address	`dest = address_of(src)`
mov src, dest	R, R/M R/M, R I, R/M	move	`dest = src`
mul factor	R/M	multiply	`rdx:rax = rax * factor`
nop		no operation	
not src_dest	R/M	bitwise not	`src_dest = not(src_dest)`
or src, src_dest	R, R/M R/M, R I, R/M	bitwise or	`src_dest = src_dest or src`
pop dest	R/M	pop from stack	
push src	R/M	push onto stack	
rcl count, src_dest	I, R/M %cl, R/M	rotate through carry left	`src_dest = rol(src_dest, count, CF)`
rcr count, src_dest	I, R/M %cl, R/M	rotate through carry right	`src_dest = ror(src_dest, count, CF)`
ret		return	`pop rip`
rol count, src_dest	I, R/M %cl, R/M	rotate left	`src_dest = rol(src_dest, count)`
ror count, src_dest	I, R/M %cl, R/M	rotate right	`src_dest = ror(src_dest, count)`
sal count, src_dest	I, R/M %cl, R/M	shift arithmetic left	`src_dest = src_dest << count`
sar count, src_dest	I, R/M %cl, R/M	shift arithmetic right (with sign extension)	`src_dest = src_dest >> count`
sbb src, src_dest	R, R/M R/M, R I, R/M	sub with borrow	`src_dest = src_dest – src - CF`
shl count, src_dest	I, R/M %cl, R/M	shift left	`src_dest = src_dest << count`
shr count, src_dest	I, R/M %cl, R/M	shift right (without sign extension)	`src_dest = src_dest >> count`
stc		set carry	`CF = 1`
sub src, src_dest	R, R/M R/M, R I, R/M	arithmetic subtract	`src_dest = src_dest - src`
syscall		transfer control to operating system	`syscall_number = rax`
test src1, src2	R, R/M R/M, R I, R/M	test (sets flags based on src2 and src1)	
xor src, src_dest	R, R/M R/M, R I, R/M	bitwise xor	`src_dest = src_dest xor src`

Instruction suffixes: q … 64 bits (quadword), l … 32 bits (long int), w … 16 bits (word), b … 8 bits (byte)

Operand types:

I … Immediate value, R … Register, M … Memory address reference, D … Displacement only, %cl … use value of cl register