Registers

Saved By	Quad (8 Byte)	Long (4 Byte)	Word (2 Bytes)	Higher Byte (1 Byte)	Lower Byte (1 Byte)	Remarks
Callee	%rax	%eax	%ax	%ah	%al	
Callee	%rbx	%ebx	%bx	%bh	%bl	
Caller	%rcx	%ecx	%cx	%ch	%cl	4th Function Parameter
Caller	%rdx	%edx	%dx	%dh	%dl	3rd Function Parameter
Caller	%rsi	%esi	%si	–	%sil	2nd Function Parameter
Caller	%rdi	%edi	%di	–	%dil	1st Function Parameter
Caller	%rsp	%esp	%sp	–	%spl	Holds memory address of stack
Callee	%rbp	%ebp	%bp	–	%bpl	Used to back up `%rsp`
–	%rip	–	–	–	–	Holds memory address of currently executing code. - DO NOT change directly
Caller	%r8	%r8d	%r8w	–	%r8b	5th Function Parameter
Caller	%r9	%r9d	%r9w	–	%r9b	6th Function Parameter
Caller	%r10	%r10d	%r10w	–	%r10b	
Caller	%r11	%r11d	%r11w	–	%r11b	
Callee	%r12	%r12d	%r12w	–	%r12b	
Callee	%r13	%r13d	%r13w	–	%r13b	
Callee	%r14	%r14d	%r14w	–	%r14b	
Callee	%r15	%r15d	%r15w	–	%r15b	

Condition Codes

Single bit flags that indicate a specific result from the last arithmetic instruction

- `leaq` does not set these codes

Code	Description	Set When an Arithmetic Operation…
CF	Carry Flag	Carries out from most significant bit / Unsigned overflow
ZF	Zero Flag	Result is 0
SF	Sign Flag	Result < 0
OF	Overflow Flag	Signed overflow (sign of operands are matching but result is opposite) `(a > 0 && b > 0 && t < 0) || (a < 0 && b < 0 && t >= 0)`

C-Sizes

Type	Size (Byte)
char	1
short	2
int	4
long	8

Instructions

Suffix `q` can be replaced with: `w`, `l` or `b`

Arithmetic

Op Code	C-Equivalent	Remarks
`incq src`	`dest = dest + 1`	
`decq src`	`dest = dest - 1`	
`negq src`	`dest = -dest`	
`notq src`	`dest = ~dest`	
`addq src, dest`	`dest = dest + src`	
`subq src, dest`	`dest = dest - src`	
`imulq src, dest`	`dest = dest * src`	
`salq k, dest`	`dest = dest << k`	Fills with 0s
`sarq k, dest`	`dest = dest >> k`	Preserves sign bit; fills with sign bit
`shrq k, dest`	`dest = dest >> k`	Does not preserve sign bit; fills with 0s
`xorq src, dest`	`dest = dest ^ src`	
`andq src, dest`	`dest = dest & src`	
`orq src, dest`	`dest = dest | src`	
`leaq src, dest`	-	Stores address calculation into `dest` without accessing it. - Can also be used to compute expressions of the form `x + k * y + c` - Left operand is of the form of memory addressing
`movsXY src, dest`	-	Sets 1st `sizeof(X)` bits to `src` and remainder bits to the zero.
`movzXY src, dest`	-	Sets 1st `sizeof(X)` bits to `src` and remainder bits to the sign bit of `src`. * movzlq = movl

* Valid parameters of
  - `X: b, l`
  - `Y: l, q`
  - `X != Y`

Jump Instructions

Op Code	Description	Condition
`jmp`	Unconditional	`1`
`je`	Equal / Zero	`ZF`
`jne`	Not Equal / Not Zero	`~ZF`
`js`	Negative	`SF`
`jns`	Non-Negative	`~SF`
`jg`	Greater (Signed)	`~(SF^OF)&~ZF`
`jge`	Greater or Equal (Signed)	`~(SF^OF)`
`jl`	Less (Signed)	`SF^OF`
`jle`	Less or Equal (Signed)	`(SF^OF)|ZF`
`ja`	Above (Unsigned)	`~CF&~ZF`
`jb`	Below (Unsigned)	`CF`

Test Instructions

Op Code	Remarks
`testq a, b`	Sets the following **Flag** / **If** ZF / `a&b == 0` SF / `a&b < 0`
`cmpq a, b`	Sets condition codes as if performing `subq a, b` without storing

Stack Instructions

Op Code	Description	Equivalent To
`pushq src`	Pushes data from the register src into the stack	`subq $8, %rsp` `movq src, (%rsp)`
`popq dest`	Pops data from the stack into the register dest	`movq (%rsp), dest` `addq $8, %rsp`