x86_64 (amd64) Reference Sheet

Instructions

`mov S, D`	D = S
`add S, D`	D = D + S
`sub S, D`	D = D - S
`neg D`	D = -D
`imul S, D`	D = D * S
`idiv S`	%rax = %rax / S %rdx = remainder
`sal c, D`	D = D << c
`shr c, D`	D = D >> c (logical)
`sar c, D`	D = D >> c (arithmetic)
`and S, D`	D = D & S
`or S, D`	D = D | S
`xor S, D`	D = D ^ S
`not D`	D = ~D
`inc D`	D = D + 1
`dec`	D = D - 1
`lea S, D`	D = address of S
`cmp O1, O2`	Sets CCs: O2 - O1
`test O1, O2`	Sets CCs: O2 & O1
`br address`	PC = address
`cbz R, label`	If R == 0, PC = addr of label
`cbnz R, label`	If R != 0, PC = addr of label
`jmp label`	jump to label (PC = address of label)
`je label`	jump if equal
`jne label`	jump if not equal
`js label`	jump if negative
`jns label`	jump if non-negative
`jg label`	jump if greater than
`jge label`	jump if greater or equal
`jl label`	jump if less than
`jle label`	jump if less or equal
`ja label`	jump above (unsigned jg)
`jb label`	jump below (unsigned jl)
`push S`	%rsp = %rsp - 8 mem[%rsp] = S
`pop D`	D = mem[%rsp] %rsp = %rsp + 8
`callq label`	push address of next instr jmp label
`leaveq`	mov %rbp, %rsp pop %rbp
`retq`	pop address of next instr

Addressing Modes

Immediate (constant)
A number prefixed with $. Can be decimal or hex:
`Examples: $8  $0x1F  $-32`

Register
A register name prefixed with %:

`Examples: %rax  %rbp  %r15`

Memory (normal)
Access memory at the address stored in a register:
`(%reg)`

`Examples: (%rax)  (%rbp)`

Memory (Displacement)
Access memory at the address stored in a register plus a constant, C:
`C(%reg)`

`Examples: 8(%rbp)  -0x10(%rsp)`

Memory (Indexed)
Access memory at the address stored in a register (base) plus a constant, C, plus a scale * a register (index):
`C(%base, %index, scale)`

```
Examples:
(%rax, %rcx)
0x8(%rbp, %rax, 8)
```

Instruction Suffixes

b	byte	
w	word	(2 bytes)
l	long	(4 bytes)
q	quad	(8 bytes)

Condition Codes

ZF: Zero Flag
SF: Sign Flag (negative)
CF: Carry Flag (unsigned overflow)
OF: Overflow Flag (signed overflow)

32-bit Registers

Registers prefexed with **e** rather than **r** represent the lower 32-bits of a register. (e.g., `%eax` vs. `%rax`)