Info

x means destination register, sr means source register in a register–register operand, pr means register in a register–pointer operand. n means constant number, nb means constant byte number. * means pointer.
For arguments, R means register, Rax means any ax register (ax, eax, rax), K means a constant of any size, KB means a constant where $0 \leq K \leq 127$, P means memory address.

Destination Registers (x)

`ax` → `0`
`bx` → `3`
`cx` → `1`
`dx` → `2`
`di` → `7`

Source Registers (sr)

`ax` → `Cx`
`bx` → `D(x+8)`
`cx` → `C(x+8)`
`dx` → `Dx`
`di` → `F(x+8)`

Register in a Register–Pointer Op (pr)

`ax` → `04`
`bx` → `1C`
`cx` → `0C`
`dx` → `14`
`di` → `3C`

Size

Goes before every opcode.
`word` → `66`
`dword` → `[none]`
`qword` → `48`

1 mov

1.1 mov

1.1.1 register

```
mov R R ⇒ 89 sr
mov R K ⇒ C7 Cx n
mov R P ⇒ 8B pr 25 *
```

1.1.2 pointer

```
mov P R ⇒ 89 pr 25 *
mov P K ⇒ C7 04 25 * n
```

1.2 movsx

```
movsx R byte P ⇒ 0F BE pr 25 *
movsx R word P ⇒ 0F BF pr 25 *
movsxd R P ⇒ 63 pr 25 *
```

1.3 movzx

```
movzx R byte P ⇒ 0F B6 pr 25 *
movzx R word P ⇒ 0F B7 pr 25 *
```

1.4 cmov

```
cmovz R R ⇒ 0F 44 sr
cmovs R R ⇒ 0F 48 sr
```

2 arithmetic

2.1 add

2.1.1 register

```
add R R ⇒ 01 sr
add R KB ⇒ 83 Cx nb
add Rax K ⇒ 05 n
add R K ⇒ 81 Cx n
add R P ⇒ 01 pr 25 *
```

2.1.2 pointer

```
add P R ⇒ 03 pr 25 *
add P KB ⇒ 83 04 25 * nb
add P K ⇒ 81 04 25 * n
```

2.2 sub

2.2.1 register

`sub R S ⇒ 29 sr`
`sub R KB ⇒ 83 C(x+8) nb`
`sub rax K ⇒ 2D n`
`sub R K ⇒ 81 E(x+8) n`
`sub R P ⇒ 29 pr 25 *`

2.2.2 pointer

`sub P R ⇒ 2B pr 25 *`
`sub P KB ⇒ 83 2C 25 * nb`
`sub P K ⇒ 81 2C 25 * nb`

2.3 mul

`mul R ⇒ F7 Ex`
`mul P ⇒ F7 24 25 *`

2.4 imul

`imul R ⇒ F7 E(x+8)`
`imul P ⇒ F7 2C 25 *`
`imul R R ⇒ 0F AF sr`

2.5 div

`div R ⇒ F7 Fx`
`div P ⇒ F7 34 25 *`

2.6 idiv

`idiv R ⇒ F7 F(x+8)`
`idiv P ⇒ F7 3C 25 *`

2.7 neg

`neg R ⇒ F7 D(x+8)`
`neg P ⇒ F7 1C 25 *`

3 Shift

3.1 shr

`shr R 1 ⇒ D1 E(x+8)`
`shr R KB ⇒ C1 E(x+8) nb`
`shr P 1 ⇒ D1 2C 25 *`
`shr P KB ⇒ C1 2C 25 * nb`

3.2 sar

```
sar R 1 ⇒ D1 F(x+8)
sar R KB ⇒ C1 F(x+8) nb
sar P 1 ⇒ D1 3C 25 *
sar P KB ⇒ C1 3C 25 * nb
```

3.3 shl

```
shl R 1 ⇒ D1 Ex
shl R KB ⇒ C1 Ex nb
shl R 1 ⇒ D1 24 25 *
shl R KB ⇒ C1 24 25 * nb
```

3.4 ror

```
ror R 1 ⇒ D1 C(x+8)
ror R KB ⇒ C1 C(x+8) nb
ror P 1 ⇒ D1 0C 25 *
ror P KB ⇒ C1 0C 25 * nb
```

3.5 rol

```
rol R 1 ⇒ D1 Cx
rol R KB ⇒ C1 Cx nb
rol P 1 ⇒ D1 04 25 *
rol P KB ⇒ C1 04 25 * nb
```

4 Bitwise Logic

4.1 not

```
not R ⇒ F7 Dx
not P ⇒ F7 14 25 *
```

4.2 or

4.2.1 register

```
or R R ⇒ 09 sr
or R KB ⇒ 83 C(x+8) nb
or rax K ⇒ 0D n
or R K ⇒ 81 C(x+8) n
or R P ⇒ 09 pr 25 *
```

4.2.2 pointer

```
or P R ⇒ 0B pr 25 *
or P KB ⇒ 83 0C 25 * nb
or P K ⇒ 81 0C 25 * n
```

4.3 xor

4.3.1 register

```
xor R R ⇒ 31 sr
xor R KB ⇒ 83 Fx nb
xor rax K ⇒ 35 n
xor R K ⇒ 81 Fx n
xor R P ⇒ 33 pr 25 *
```

4.3.2 pointer

```
xor P R ⇒ 31 pr 25 *
xor P KB ⇒ 83 34 25 * nb
xor P K ⇒ 81 34 25 * n
```

4.4 and

4.4.1 register

```
and R R ⇒ 21 sr
and R KB ⇒ 83 Ex nb
and rax K ⇒ 25 n
and R K ⇒ 81 Ex n
and R P ⇒ 23 pr *
```

4.4.2 pointer

```
and P R ⇒ 21 pr 25 *
and P KB ⇒ 83 24 25 * nb
and P K ⇒ 81 24 25 * n
```

4.5 test

4.5.1 register

```
test R R ⇒ 85 sr
test rax K ⇒ A9 n
test R K ⇒ F7 Cx n
test R P ⇒ 85 pr 25 *
```

4.5.2 pointer

```
test P R ⇒ 85 pr 25 *
test P K ⇒ F7 04 25 * n
```

5 jmp

5.1 byte-length

`jns KB` ⇒ `79 nb`
`jnz KB` ⇒ `75 nb`
`jmp KB` ⇒ `EB nb`

6 Miscellaneous

Always 32-bits (dword) (meaning no size code).
`nop` ⇒ `90`
`syscall` ⇒ `0F 05`