```
Definitions                                                     ; inversed's concise x86 assembler reference
    D   Destination         r   Register        CF  Carry flag  ; v0.3 : 2015.01.22
    S   Source              m   Memory          ZF  Zero flag
    T   Second source       i   Immediate
                            cl  cl register
    D and S cannot be two memory locations

Data Transfer
    mov         Drm , Srmi      move                        D = S
    mov(z/s)x   Dr  , Srm       move with zero/sign extend  D = S extended with zeroes/sign
    cmovcc      Dr  , Srm       conditional move            D = S if condition cc holds
    lea         Dr  , [S]m      load effective address      D = S
    cdq                         convert double- to quad-word edx:eax = eax extended with sign
    bswap       Dr              byte swap                   Reverse byte order of D
    xchg        Dr  , Sr        exchange                    (D, S) = (S, D)
    push        Srmi                                        Push S onto the stack
    pop         Drm                                         Pop top of the stack into D

Arithmetic
    add/sub     Drm , Srmi      add/subtract                D = D +/- S
    adc/sbb     Drm , Srmi      add/subtract with carry/borrow  D = D +/- (S + CF)
    inc/dec     Drm             increment/decrement         D = D +/- 1, CF not changed
    neg         Drm             negate                      D = -D
    mul         Drm             unsigned multiply           edx:eax = D * eax
    imul        Drm             signed multiply             edx:eax = D * eax
    imul        Dr  , Srmi      signed multiply             D = D * S
    imul        Dr  , Srm , Ti  signed multiply             D = S * T
    (/i)div     Srm             (un/)signed division        eax = edx:eax div S, edx = edx:eax mod S

Logic
    and         Drm , Srmi      bitwise AND                 D = D bitwise and S
    or          Drm , Srmi      bitwise OR                  D = D bitwise or  S
    xor         Drm , Srmi      bitwise XOR                 D = D bitwise xor S
    not         Drm             bitwise NOT                 D = bitwise not D
    popcnt      Dr  , Srm       population count            D = number of nonzero bits in S
    bs(f/r)     Dr  , Srm       bit scan forward/reverse    D = index of least/most significant bit of S,
                                                            ZF = is nonzero bit found
Bit shifting
    sh(l/r)     Drm , Sicl      shift left/right            Shift D to the left/right by S bits
    sa(l/r)     Drm , Sicl      shift arithmetic            Shift D to the left/right by S bits preserving its sign
    sh(l/r)d    Drm , Sr  , Ticl  shift double              Shift D:S to the left/right by T bits
    ro(l/r)     Drm , Sicl      rotate                      Cyclicly shift D to the left/right by S bits
    rc(l/r)     Drm , Sicl      rotate through carry        Cyclicly shift D to the left/right by S bits through CF
```

Control Flow

cmp	Drm , Srmi	Compare	Set flags based on D - S
test	Drm , Sri	Test	Set flags based on D AND S
bt(/s/r/c)	Drm , Sri	Bit test	Copy S-th bit of D into CF and /set/reset/complement it
jmp		Unconditional jump	
jcc		Conditional jump	Jump depending on condition cc
setcc	Drm	Set on condition	D = 0 or 1 depending on condition cc
nop		No operation	
call		Call	
ret	(/S)	Return	Return from routine (/and pop S bytes from the stack)
clc		Clear carry	Clear CF
stc		Set carry	Set CF
cmc		Complement carry	Complement CF

Conditions

(/n)(/g/l/a/b)(/e)

n	not	
e	equal	
g	greater	signed
l	less	signed
a	above	unsigned
b	below	unsigned

(/n)(z/c/s/o/p)

n	not
z	zero
c	carry
s	sign
o	overflow
p	parity

Deprecated

partial registers	false dependencies
string instructions	high overhead, slow without a repeat prefix
xchg Dr , Sm	implicit lock prefix, very slow
xlat	slower than using mov
rc(l/r) Drm , Sicl	much slower than rotate by 1
loop	slower than jcc
jecxz	pointless
inc/dec	false dependencies, possibly slower than add/sub