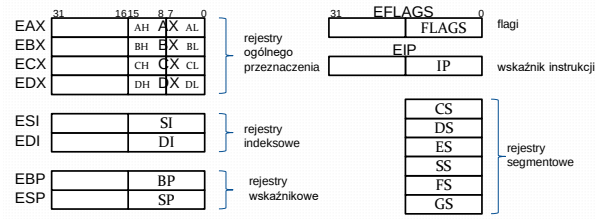
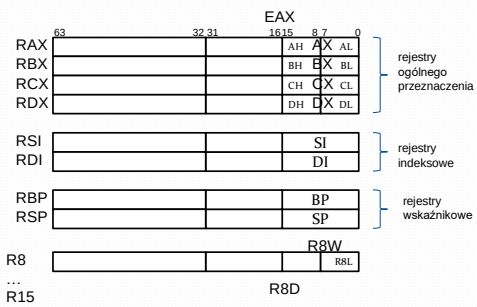
Instrukcje logiczne, przesunięć i rotacji

IA32- rejestry



EM64T- rejestry



Rejestr flag

Rejestr flag w architekturze Intel x86			
bit	Skrót/wartość	opis	typ
0	CF	flaga przeniesienia (carry)	S
2	PF	flaga parzystości (parity)	S
4	AF	flaga wyrównania (adjust)	S
6	ZF	flaga zera (zero)	S
7	SF	flaga znaku (sign)	S
10	DF	flaga kierunku (direction)	C
11	OF	flaga przepełnienia (overflow)	S

S: Znacznik stanu
C: Znacznik kontrolny
X: Znacznik systemowy

Instrukcje logiczne

- AND bitowa funkcja AND
- ANDN bitowa funkcja AND z negacją
- OR bitowa funkcja OR
- XOR bitowa funkcja OR
- NOT bitowa funkcja NOT

Wpływa na flagi: OSZAPC
0SZxP0

Instrukcja AND

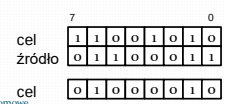
and cel, źródło

Wyznacza iloczyn logiczny zawartości źródła i celu (bit po bicie), wynik umieszcza w miejscu przeznaczenia (cel).

cel := cel and źródło

```
and eax, zmienna
and edx, [ebx+esi*4]
and rax, rdx
```

Uwaga:
Jeśli źródło jest adresowane w trybie prostym może mieć do 32 bitów.



Wpływa na flagi: OSZAPC
0SZxx0
Wymagane BMI1

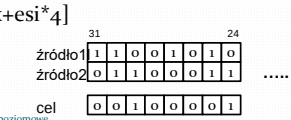
Instrukcja ANDN

andn cel, źródło1, źródło2

Wyznacza iloczyn logiczny zanegowanego źródła1 i źródła2 (bit po bicie), wynik umieszcza w miejscu przeznaczenia (cel). Cel i źródło1 są rejestrami 32|64 bitowymi.

cel := (not źródło1) and źródło2

```
andn  edx, eax, zmienna
andn  eax, edx, [ebx+esi*4]
andn  r8, rax, rdx
```



Wpływa na flagi: OSZAPC
0SZxP0

Instrukcja OR

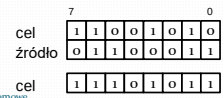
or cel, źródło

Wyznacza sumę logiczną zawartości źródła i celu (bit po bicie), wynik umieszcza w miejscu przeznaczenia (cel).

cel := cel or źródło

```
or    eax, zmienna
or    edx, [ebx+esi*4]
or    rax, r9
```

Uwaga:
Jeśli źródło jest adresowane w trybie prostym może mieć do 32 bitów.



Wpływa na flagi: OSZAPC
0SZxP0

Instrukcja XOR

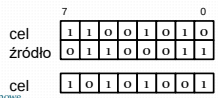
xor cel, źródło

Wyznacza, bit po bicie, sumę modulo 2 zawartości źródła i celu wynik umieszcza w miejscu przeznaczenia (cel).

cel := cel xor źródło

```
xor   eax, zmienna
xor   edx, [ebx+esi*4]
xor   rax, r9
```

Uwaga:
Jeśli źródło jest adresowane w trybie prostym może mieć do 32 bitów.



Wpływa na flagi: -

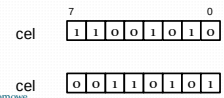
Instrukcja NOT

not cel

Wyznacza negację logiczną zawartości i celu (bit po bicie), wynik umieszcza w miejscu przeznaczenia (cel).

cel := not cel

```
not   eax
not   byte ptr [ebx+esi*4]
not   rdx
```



If ((!a&&b)||(c&&d&&e)) {...}else {...};

```
        mov   al, a          mov   al, a          mov   dl, c
        not   al             andn  al, al, b      mov   al, a
        and   al, b          jnz   then_1         and   dl, d
        mov   dl, c          mov   dl, c          andn  al, al, b
        and   dl, d          and   dl, d          and   dl, e
        and   dl, e          and   dl, e          or    al, dl
        or    al, dl         or    al, dl         jz    else_1
        jz    else_1         jz    else_1   then_1:
then_1:              then_1:
```


Instrukcje przesunięć i rotacji

- SAR przesunięcie arytmetyczne w prawo
- SHR przesunięcie logiczne w prawo
- SAL przesunięcie arytmetyczne w lewo
- SHL przesunięcie logiczne w lewo
- SARX przesunięcie arytmetyczne w prawo
- SHRX przesunięcie logiczne w prawo
- SHLX przesunięcie logiczne w lewo
- SHRD przesunięcie w prawo double
- SHLD przesunięcie w lewo double
- ROR rotacja w prawo
- ROL rotacja w lewo
- RCR rotacja w prawo przez przeniesienie
- RCL rotacja w lewo przez przeniesienie
- RORX rotacja w prawo

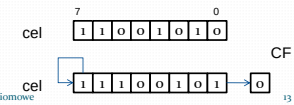
Wpływa na flagi: OSZAPC
(0x)SZxPC

Instrukcja SAR

sar cel, ile

Przesunięcie arytmetyczne celu w prawo o ile bitów. Ile=1, cl lub wartość 0-31|63 .

```
sar   eax, 1
sar   [ebx+esi*4], cl
sar   rdx, cl
```



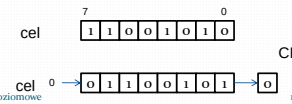
Wpływa na flagi: OSZAPC
(0x)SZxPC

Instrukcja SHR

shr cel, ile

Przesunięcie logiczne w prawo o ile bitów. Ile=1, cl lub wartość 0-31|63.

```
shr   eax, 1
shr   [ebx+esi*4], cl
shr   rdx, cl
```



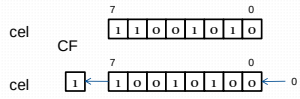
Wpływa na flagi: OSZAPC
(0x)SZxPC

Instrukcja SAL

sal cel, ile

Przesunięcie arytmetyczne celu w lewo o ile bitów. Ile=1, cl lub wartość 0-31|63.

```
sal   eax, 1
sal   [ebx+esi*4], cl
sal   rdx, cl
```



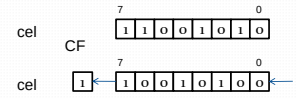
Wpływa na flagi: OSZAPC
(0x)SZxPC

Instrukcja SHL

shl cel, ile

Przesunięcie logiczne celu w lewo o ile bitów. Ile=1, cl lub wartość 0-31|63.

```
shl   eax, 1
shl   [ebx+esi*4], cl
shl   rdx, cl
```



Przykład – oblicz sumę krawędzi prostopadłościanu

```
mov   eax, a          ;wczytaj a
add   eax, b          ;dodaj b
add   eax, c          ;dodaj c
sal   eax, 2          ; ×4
```


Wpływa na flagi: -----
Wymaga BMI2

Instrukcja SARX

sarx cel, źródło, ile

Przesunięcie arytmetyczne źródła w prawo o ile bitów i zapisanie w celu. Cel i ile są rejestrami 32|64 bitowymi .

```
sarx   eax, zmienna, edx
sarx   eax, [ebx+esi*4], ecx
sarx   rdx, rax, rcx
```

cel

Wpływa na flagi: ------
Wymaga BMI2

Instrukcja SHRX

shrx cel, źródło, ile

Przesunięcie logiczne źródła w prawo o ile bitów i zapisanie w celu. Cel i ile są rejestrami 32|64 bitowymi .

```
shrx   eax, zmienna, edx
shrx   eax, [ebx+esi*4], ecx
shrx   rdx, rax, rcx
```


Wpływa na flagi: ------
Wymaga BMI2

Instrukcja SHLX

shlx cel, źródło, ile

Przesunięcie logiczne źródła w lewo ile bitów i zapisanie w celu. Cel i ile są rejestrami 32|64 bitowymi .

```
shlx   eax, zmienna, edx
shlx   eax, [ebx+esi*4], ecx
shlx   rdx, rax, rcx
```

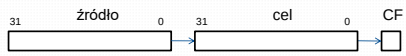

Wpływa na flagi: OSZAPC
(Ox)SZxPC

Instrukcja SHRD

shrd cel, źródło, ile

Przesunięcie źródła:celu w prawo o ile bitów. Ile=cl lub wartość 0-31|64. Rejestr źródła (16,32,64) pozostaje bez zmian.

```
shrd   eax, ecx, 15
shrd   [ebx+esi*4], edx, cl
shrd   zmienna, rax, cl
```



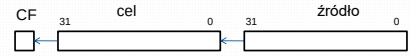
Wpływa na flagi: OSZAPC
(Ox)SZxPC

Instrukcja SHLD

shld cel, źródło, ile

Przesunięcie źródła:celu w lewo o ile bitów. Ile=cl lub wartość 0-31|63. Rejestr źródła (16,32,64) pozostaje bez zmian.

```
shld   eax, ecx, 15
shld   [ebx+esi*4], edx, cl
shld   zmienna, rax, cl
```



Przykład – podziel 256-bitową liczbę a przez 64

```
mov    rcx, offset a       ;wczytaj adres a
mov    rax, [rcx+8]        ;wczytaj drugie 64 bity
shrd   [rcx], rax, 6       ;przesuń pierwsze
mov    rax, [rcx+16]       ;wczytaj trzecie 64 bity
shrd   [rcx+8], rax, 6     ;przesuń drugie
mov    rax, [rcx+24]       ;wczytaj czwarte 64 bity
shrd   [rcx+16], rax, 6    ;przesuń trzecie
shr    rax, 6              ;przesuń czwarte
mov    [rcx+24], rax       ;zapisz
```

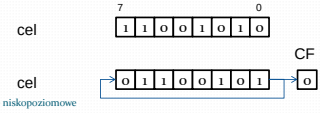

Wpływa na flagi: OSZAPC
(0x)----C

Instrukcja ROR

ror cel, ile

Rotacja (obrót) celu w prawo o ile bitów. Ile=1, cl lub wartość 0-31|63.

```
ror    eax, 1
ror    [ebx+esi*4], cl
ror    rdx, cl
```



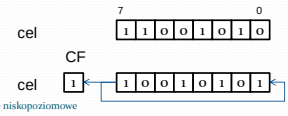
Wpływa na flagi: OSZAPC
(0x)----C

Instrukcja ROL

rol cel, ile

Rotacja (obrót) celu w lewo o ile bitów. Ile=1, cl lub wartość 0-31|63.

```
rol   eax, 1
rol   [ebx+esi*4], cl
rol   rdx, cl
```



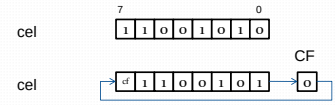
Wpływa na flagi: OSZAPC
(0x)----C

Instrukcja RCR

rcr cel, ile

Rotacja (obrót) przez przeniesienie celu w prawo o ile bitów. Ile=1, cl lub wartość 0-31|63.

```
rcr   eax, 1
rcr   [ebx+esi*4], cl
rcr   rdx, cl
```



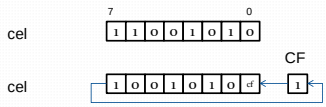
Wpływa na flagi: OSZAPC
(0x)----C

Instrukcja RCL

ror cel, ile

Rotacja (obrót) przez przeniesienie celu w lewo o ile bitów. Ile=1, cl lub wartość 0-31|63.

```
rcl   eax, 1
rcl   [ebx+esi*4], cl
rcl   rdx, cl
```



Wpływa na flagi: ------
Wymaga BMI2

Instrukcja RORX

rorx cel, źródło, ile

Rotacja źródła w prawo o ile bitów i zapisanie w celu. Cel jest rejestrem 32|64 bitowym. Ile przyjmuje wartość 0-31|63.

```
rorx  eax, zmienna, 12
rorx  eax, [ebx+esi*4], 7
rorx  rdx, rax, 44
```


Przykłady

```
and   eax, offffefffh     ;zeruje 12 bit eax

or    ax, 01000h          ;ustawia 12 bit ax

xor   rax, 01000h         ;neguje 12 bit rax

and   eax, eax            ;m. in. zeruje CF

xor   eax, eax            ;zeruje eax
```


Przykłady

```
and   eax, 070h     ;maska
sar   eax, 4        ;eax – 3bitowa liczba


mov   ah, al        ;zamienia al na jego
and   ax, 0f00fh    ; szesnastkową reprezentację
shr   ah, 4         ;ASCII w ah, al.
add   ax, 3030h
```

Przykłady - *4,25

```
movsxd    rax, zmienna      ;typu int – 32 bity
mov       rdx, rax          ;powiel
sal       rax, 2            ;x4
sar       rdx, 2            ;/4
adc       rax, rdx          ;x4¼
mov       zmienna, eax      ;zapisz

movsxd    rdx, eax          ; test
cmp       rax, rdx          ;porównaj
jnz       blad              ;przekroczenie zakresu
```


Przykłady

```
;odwracanie bitów z rcx; rdx liczba odwracanych bitów 1-64
invb    proc
        xor     rax, rax        ;rax:=0
        dec     rdx             ;eliminacja zera
        js      @e
        and     rdx, 3fh        ;ograniczenie zakresu
@p:     shr     rcx, 1          ;lsb->CF
        rcl     rax, 1          ;CF->msb
        dec     rdx             ;licznik--
        jns     @p              ;następny jeśli nieujemne
@e:     ret
invb    endp
```