Operacje na znacznikach, bitach i bajtach

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

Operacje na flagach

- STC Ustawienie CF
- CLC Zerowanie CF
- CMC Zanegowanie CF
- CLD Zerowanie DF – flagi kierunku
- STD Ustawienie DF
- LAHF Przesłanie flag do rejestru AH
- SAHF Przesłanie rejestru AH do flag
- PUSHF/PUSHFD/ PUSHFQ Wysłanie flag na stos
- POPF/POPFD/POPFQ Pobranie flag ze stosu
- STI Ustawienie IF – flagi przerwań
- CLI Zerowanie IF

Wpływa na flagi: C

Instrukcja STC

```
stc
```

Ustawienie flagi CF.

CF:=1

```
stc
```


Wpływa na flagi: C

Instrukcja CLC

```
clc
```

Zerowanie flagi CF.

CF:=0

```
clc
```


Wpływa na flagi: C

Instrukcja CMC

```
cmc
```

Zanegowanie flagi CF.

CF:=not CF

```
cmc
```

Wpływa na flagi: D

Instrukcja STD

std

Ustawienie flagi kierunku DF. Jeżeli DF=1 instrukcje łańcuchowe zmniejszają rejestr ESI lub EDI.

DF:=1

std

Wpływa na flagi: D

Instrukcja CLD

cld

Zerowanie flagi kierunku DF. Jeżeli DF=0 instrukcje łańcuchowe zwiększają rejestr ESI lub EDI.

DF:=0

cld

Wpływa na flagi: -

Instrukcja LAHF

lahf

Przesłanie flag do rejestru AH

AH:=lo(FLAGS)

lahf

Wpływa na flagi: SZAPC

Instrukcja SAHF

sahf

Przesłanie rejestru AH do flag. Bity 1,3,5 są ignorowane.

lo(FLAGS) :=AH

sahf

Wpływa na flagi: -

Instrukcja PUSHF/PUSHFD/PUSHFQ

pushf/pushfd/pushfq

Przesyła zawartość Flag/Eflag/Rflag na stos.

pushf
pushfd

Wpływa na flagi: OSZAPC

Instrukcja POPF/POPFD/POPFQ

popf/popfd/popfq

Pobiera zawartość Flag/EFlag ze stosu.

popf
popfd

Wpływa na flagi: I

Instrukcja STI

sti

Ustawienie flagi przerwań IF lub VIF. Włącza po następnej instrukcji system przerwań maskowalnych.

IF:=1

sti

Wpływa na flagi: I

Instrukcja CLI

cli

Zerowanie flagi przerwań IF lub VIF. Wyłącza system przerwań maskowalnych.

IF:=0

cli

Operacje na bitach

- BT Testowanie bitu
- BTS Testowanie bitu z ustawianiem
- BTR Testowanie bitu z zerowaniem
- BTC Testowanie bitu z negacją
- TEST Porównanie logiczne
- BSF Przeszukiwanie bitów w przód
- BSR Przeszukiwanie bitów wstecz
- LZCNT Zlicza zerowe bity od najstarszego
- TZCNT Zlicza zerowe bity od najmłodszego
- BEXTR Wycina ciąg bitów
- BLSI Kopiuje najmłodszy ustawiony bit
- BLSR Zeruje najmłodszy ustawiony bit
- BLSMSK Tworzy maskę do bitu=0
- BZHI Zeruje starsze bity

Wpływa na flagi: OSZAPC
xxxxxC

Instrukcja BT

bt baza, nr

Wyznacza wartość bitu nr (rejestr lub wartość) w bazie (rejestr lub zmienna) i umieszcza ją w CF.

CF:=bit bazy numer nr

bt zmienna, eax
bt edx,12
bt rcx, 37

Wpływa na flagi: OSZAPC
xxxxxC

Instrukcja BTS

bts baza, nr

Wyznacza wartość bitu nr (rejestr lub wartość) w bazie (rejestr lub zmienna) i umieszcza ją w CF. Następnie ustawia badany bit.

CF:=bit bazy numer nr
bit bazy numer nr:=1

bts zmienna, eax
bts edx,12
bts rcx, 37

Wpływa na flagi: OSZAPC
xxxxxC

Instrukcja BTR

btr baza, nr

Wyznacza wartość bitu nr (rejestr lub wartość) w bazie (rejestr lub zmienna) i umieszcza ją w CF. Następnie zeruje badany bit.

CF:=bit bazy numer nr
bit bazy numer nr:=0

btr zmienna, eax
btr edx,12
btr rcx, 37

Wpływa na flagi: OSZAPC
xxxxxC

Instrukcja BTC

btc baza, nr

Wyznacza wartość bitu nr (rejestr lub wartość) w bazie (rejestr lub zmienna) i umieszcza ją w CF. Następnie neguje badany bit.

CF:=bit bazy numer nr

bit bazy numer nr:= not bit bazy numer nr

```
btc   zmienna, eax
btc   edx,12
btc   rcx, 37
```


Wpływa na flagi: OSZAPC
0SZxP0

Instrukcja TEST

test cel, źródło

Wyznacza iloczyn logiczny(bit po bicie) zawartości celu i źródła (rejestr lub wartość), wynik jest pominięty, ustawia flagi.

cel and źródło

```
test   eax,zmienna
test   edx,[ebx+esi*4]
```


Wpływa na flagi: OSZAPC
xxZxxx

Instrukcja BSF

bsf cel, źródło

Przeszukiwanie bitów w przód. Szuka w rejestrze lub zmiennej źródła najmłodszego bitu=1, jego indeks umieszcza w rejestrze celu (ZF=0). Jeśli źródło=0, wówczas ZF=1, a cel jest niezdefiniowany

cel :=indeks najmłodszego bitu=1 źródła

```
bsf   eax,zmienna
bsf   edx,esi
bsf   rcx, rdx
```


Wpływa na flagi: OSZAPC
xxZxxx

Instrukcja BSR

bsr cel, źródło

Przeszukiwanie bitów wstecz. Szuka w rejestrze lub zmiennej źródła najstarszego bitu=1, jego indeks umieszcza w rejestrze celu (ZF=0). Jeśli źródło=0, wówczas ZF=1, a cel jest niezdefiniowany

cel :=indeks najstarszego bitu=1 źródła

```
bsr   eax, zmienna
bsr   edx, esi
bsr   rcx, rdx
```


Wpływa na flagi: OSZAPC
xxZxxC
Wymaga LZCNT

Instrukcja LZCNT

lzcnt cel, źródło

Zlicza starsze (wiodące) zerowe bity źródła (16|32|64) i ilość zapisuje do rejestru celu. Dla celu=0 ZF=1. Dla celu=rozmiarowi źródła CF=1.

cel :=liczba wiodących zer w źródle

```
lzcnt  eax, zmienna
lzcnt  edx, esi
lzcnt  rcx, rdx
```


Wpływa na flagi: OSZAPC
xxZxxC
Wymaga BMI1

Instrukcja TZCNT

tzcnt cel, źródło

Zlicza od najmłodszego zerowe bity źródła (16|32|64) i ilość zapisuje do rejestru celu. Dla celu=0 ZF=1. Dla celu=rozmiarowi źródła CF=1.

cel :=liczba końcowych zer w źródle

```
tzcnt  eax, zmienna
tzcnt  edx, esi
tzcnt  rcx, rdx
```

Wpływa na flagi: OSZAPC
0xZxx0
Wymaga BMI1

Instrukcja BEXTR

bextr cel, źródło, st_ile

Wycina z rejestru|zmiennej źródła (32|64) ciąg bitów i umieszcza w rejestrze celu. Początkowy bit określa rejestr st_ile[7:0], a ilość bitów st_ile[15:8]. Jeśli cel=0, wówczas ZF=1.

cel :=źródło[start+ile-1:start]

```
bextr  eax, zmienna, edx
bextr  edx, esi, eax
bextr  rcx, rdx, rax
```


Wpływa na flagi: OSZAPC
0SZxxC
Wymaga BMI1

Instrukcja BLSI

blsi cel, źródło

Izoluje z rejestru lub zmiennej źródła najmłodszy bit=1 i umieszcza w rejestrze celu (CF=1). Zeruje pozostałe bity. Jeśli źródło=0, wówczas CF=0, a cel=0.

cel :=(-źródło) and źródło

```
blsi   eax, zmienna
blsi   edx, esi
blsi   rcx, rdx
```


Wpływa na flagi: OSZAPC
0SZxxC
Wymaga BMI1

Instrukcja BLSR

blsr cel, źródło

Kopiuje bity z rejestru lub zmiennej źródła (32|64) i umieszcza w rejestrze celu, zeruje najmłodszy bit=1 (CF=0). Jeśli źródło=0, wówczas CF=1, a cel=0.

cel :=(źródło-1) and źródło

```
blsr   eax, zmienna
blsr   edx, esi
blsr   rcx, rdx
```


Wpływa na flagi: OSZAPC
0S0xxC
Wymaga BMI1

Instrukcja BLSMSK

blsmsk cel, źródło

Ustawia młodsze bity rejestru celu (32|64) na 1 aż do numeru najmłodszego bitu=1 z rejestru lub zmiennej źródła włącznie (CF=0). Zeruje pozostałe bity. Jeśli źródło=0, wówczas CF=1, a cel=not 0.

cel :=(źródło-1) xor źródło

```
blsmsk   eax, zmienna
blsmsk   edx, esi
blsmsk   rcx, rdx
```


Wpływa na flagi: OSZAPC
0SZxxC
Wymaga BMI2

Instrukcja BZHI

bzhi cel, źródło, idx

Kopiuje bity z rejestru lub zmiennej źródła (32|64) do rejestru celu (32|64) i kasuje starsze bity od numeru z rejestru idx (CF=0). Jeśli idx>31|63, wówczas CF=1.

cel :=źródło; cel[rozmiar-1:idx]=0

```
bzhi   eax, zmienna, edx
bzhi   edx, esi, eax
bzhi   rcx, rdx, rax
```


Wpływa na flagi: -

Instrukcja SETcc

SETcc cel

Jeśli jest spełniony warunek cc, ustawia bajt na 1, w przeciwnym wypadku na 0.

```
if  cc   then cel:=1
         else cel:=0

sets   al
setge  [esi+8]
```

Instrukcje SETcc

- SETE/SETZ — Ustaw bajt jeśli equal/ zero
- SETNE/SETNZ — Ustaw bajt jeśli not equal/ not zero
- SETS — Ustaw bajt jeśli sign (negative)
- SETNS — Ustaw bajt jeśli not sign (non-negative)
- SETO — Ustaw bajt jeśli overflow
- SETNO — Ustaw bajt jeśli not overflow
- SETPE/SETP — Ustaw bajt jeśli parity even/ parity
- SETPO/SETNP — Ustaw bajt jeśli parity odd/ not parity
- SETA/SETNBE — Ustaw bajt jeśli above/ not below or equal
- SETAE/SETNB/SETNC — Ustaw bajt jeśli above or equal/ not below/ not carry
- SETB/SETNAE/SETC — Ustaw bajt jeśli below/ not above or equal/ carry
- SETBE/SETNA — Ustaw bajt jeśli below or equal/ not above
- SETG/SETNLE — Ustaw bajt jeśli greater/ not less or equal
- SETGE/SETNL — Ustaw bajt jeśli greater or equal/ not less
- SETL/SETNGE — Ustaw bajt jeśli less/ not greater or equal
- SETLE/SETNG — Ustaw bajt jeśli less or equal/ not greater

Przykład – int na bin(string)

```
procedure s2b(var s:string;i:integer);
asm
    push ebx
    bsr edx,ecx     //w edx nr najstarszej 1
    jnz @1
    mov word ptr [eax],$3001
    jmp @e
@1: inc edx
    mov [eax],dl    //długość
    dec edx
    inc eax
@p: bt ecx,edx      //testuj
    setc bl
    add bl,$30
    mov [eax],bl    //zapisz znak
    inc eax
    dec edx
    jns @p
@e: pop ebx
end;
```

```
procedure s2b2(var s:string;i:integer);
asm
    bsr edx,ecx          //w edx nr najstarszej 1
    jnz @1
    mov word ptr [eax],$3001
    jmp @e
@1: mov [eax],dl         //długość
    inc [eax]
@p: inc eax
    bt ecx,edx           //testuj
    setc [eax]
    add byte ptr [eax],$30 //zapisz znak
    dec edx
    jns @p
@e:
end;
```


Wpływa na flagi: -

Instrukcja RDPID

rdpid cel

Czyta 32-bitowy identyfikator procesora do rejestru celu.

cel=PID

```
rdpid eax
rdpid rdx
```


Wpływa na flagi: -

Instrukcja RDTSC

rdtsc

Read Time Stamp Counter. Czyta 64-bitowy licznik do rejestrów EDX:EAX.

EDX:EAX:=licznik

```
rdtsc
```


Przykład – funkcja pomiaru czasu

```
function Pomiar(a:integer):integer;
var Cykle_H,Cykle_L:integer;
asm
    rdtsc
    mov Cykle_H,edx
    mov Cykle_L,eax
    ...
    ...
    rdtsc
    sub eax,Cykle_L
    sbb edx,CykleH
    sub EAX,9       ; odliczenie 9?
end;
```


Przykład – funkcja random

```
function MyRandom0(a:integer):integer;overload;
asm
    push ebx
    xor ebx,ebx
    imul edx,[ebx+MySeed],$08088405
    inc edx
    mov [ebx+MySeed],edx
    mul edx
    mov eax,edx
    pop ebx
end;

function MyRandom1(a:integer):integer;overload;
asm
    push eax
    rdtsc
    bswap eax
    pop edx
    mul edx
    mov eax,edx
end;
```

```
function MyRandom2(a:integer):integer;overload;
asm
    push eax
    rdtsc
    ror al,3
    imul edx,MySeed,$08088405
    ror ah,1
    inc edx
    ror eax,1
    mov MySeed,edx
    bswap eax
    xor eax,edx
    pop edx
    mul edx
    mov eax,edx
end;
```