Michał Kowacki

Zbiór najważniejszych instrukcji/wiadomości z wykładów bila.

Wykład 1 – Programowanie niskopoziomowe

Języki programowania:

Wysokiego poziomu: Ada, Basic, C, C++, C#, Fortran, Java, Pascal (Delphi), SQL, …

Niskiego poziomu: **asemblery** – odpowiadają kodowi wykonywanemu przez procesor

Historia powstawania procesorów:

Intel 4004

- 4 bitowy powszechnie **uznawany za pierwszy mikroprocesor**
- 3-poziomowy stos

Intel 8008

- **pierwszy mikroprocesor 8-bitowy Intela**
- 3-4 razy więcej mocy obliczeniowej niż procesory 4 bitowe

Intel 8080

- 72 instrukcje
- 8-bitowa szyna danych, pamięć adresowana 16-bitową szyną adresową.

Intel 8086

- 16 bitowy
- Głównym konstruktorem był **Stephen Morse**

Intel 80186

- 16 bitowy
- kilka nowych rozkazów, szybszy zegar
- niektóre instrukcje były wykonywane 10-20 razy szybciej

Intel 80286

- mniej więcej dwa razy bardziej wydajny w porównaniu
- do procesora Intel 8086
- nowy tryb adresowania pamięci (tryb chroniony)

Intel 80386

- **Pierwszy 32 bitowy procesor z rodziny x86**
- **dodanie do procesora jednostki MMU**
- 32-bitowa magistrala adresowa oraz 32-bitowa magistrala
- danych
- rozszerzone do 32-bitów rejestry
- nowe tryby adresowania
- pracować w trzech trybach: rzeczywistym, chronionym i wirtualnym

Intel 80486 nazwa handlowa i486

- 32 bitowy
- cache na dane i instrukcje
- usprawnienia spowodowały, że i486 był mniej więcej
- dwukrotnie szybszy od podobnie taktowanego 80386

Intel Pentium

- architektura superskalarna
- 64-bitowa szyna danych
- jednostka branch prediction do przewidywania skoków
- (80% skuteczność)
- przeprojektowany koprocesor (5-6x wydajniejszy niż w
- i486)

Pentium Pro - mikroprocesor szóstej generacji

- Podział kodu x86 na mikrorozkazy
- Wykonywanie poza kolejnością
- Wykonywanie spekulatywne
- Dodatkowy potok ("pipeline") dla prostych instrukcji.

Pentium II

- dodatkowe instrukcje MMX
- poprawiona obsługa programów 16-bitowych
- cache pierwszego poziomu (L1) dla kodu i danych: 16 kB

Pentium III - procesor w 32-bitowej architekturze Intela (IA-32).

- architektura RISC
- rozmiar pamięci cache pierwszego poziomu (L1) dla kodu: 16 KB
- liczba etapów przetwarzania rozkazu (w potoku): 12
- liczba jednostek zmiennoprzecinkowych: 1 (z potokowaniem)
- liczba jednostek całkowitoliczbowych: 6 potoków
- liczba jednostek MMX:2
- Instrukcje SSE
- możliwość pracy w systemie wieloprocesorowym (do 2 procesorów).

Pentium 4 – siódma generacja procesorów firmy

- architektura NetBurst
- instrukcje SSE2, w nowszych wersjach jądra – SSE3
- niektóre wersje posiadają wbudowaną wielowątkowość (HyperThreading)
- zwiększona pamięć poziomu L2
- pojawia się technologia **EM64T** (2003)
- **pierwszy procesor dwurdzeniowy**

Intel Core 2 to ósma generacja mikroprocesorów firmy Intel w architekturze x86

- mikroarchitektura Intel Core
- wysoki współczynnik **IPC** (Instructions Per Cycle) - **około 3,5**
- wspólna pamięć cache dla obu rdzeni procesora
- EM64T,
- technologia wirtualizacji,
- XD bit,
- ulepszoną technologię SpeedStep,
- wersja czterordzeniowa

Intel Core i7

- modułowa budowa
- ośmiordzeniowy Nehalem ma się składać z **731 milionów tranzystorów**
- SSE 4.2.
- technologia **współbieżnej wielowątkowości**
- dynamiczne zarządzanie zasilaniem
- wbudowanie kontrolera pamięci RAM
- technologia Quick-Path

Zestawienie procesorów

Nazwa procesora	Rok	Maks. częstotliwość taktowania (w momencie wprowadzenia, MHz)	Liczba tranzystorów (mln.)
Intel 8086	1978	8	0,029
Intel 80186	1982	12	
Intel 80286	1982	12,5	0,134
Intel 80386	1985	20	0,275
Intel i486	1989	25	1,2
Pentium	1993	66	3,1
Pentium Pro	1995	200	5,5
Pentium MMX	1995	233	4,5
Pentium II	1997	266	7
Pentium III	1999	500	8,2
Pentium 4	2000	1500	42
EM64T	2003	2200	
Pentium D	2004	3200	230
Intel Core 2	2006	3000	321
Intel Core i7	2008	3400	731

Wykład 2 – Architektura procesora

Rejestry

- **AX** (ang. Accumulator) - jest wykorzystywany głównie do operacji arytmetycznych i logicznych.
- **BX** (ang. Base Registers) - rejestr bazowy, głównie wykorzystywany przy adresowaniu pamięci.
- **CX** (ang. Counter Registers) – rejestr często wykorzystywany jako licznik, np. przy instrukcji LOOP.
- **DX** (ang. Data Register) - rejestr danych, wykorzystywany przy operacjach mnożenia i dzielenia, a także do wysyłania i odbierania danych z portów.
- **SI** (ang. Source Index) - rejestr indeksujący pamięć, wskazuje obszar z którego przesyłane są dane. W połączeniu z DS tworzy adres logiczny DS:SI
- **DI** (ang. Destination Index) - rejestr indeksujący pamięć, wskazuje obszar, do którego przesyłane są dane. W połączeniu z ES, tworzy adres logiczny ES:DI
- **BP** (ang. Base Pointer) - rejestr stosowany do
- adresowania pamięci.
- **SP** (ang. Stack Pointer) - wskaźnik stosu.
- **IP** (ang. Instruction Pointer) – zawiera adres aktualnie wykonywanej instrukcji, może być modyfikowany przez rozkazy sterujące pracą programu.
- FLAGS – rejestr znaczników.

Rejestry - segmentowe

- **CS** (ang. Code Segment) - rejestr informujący o segmencie aktualnie wykonywanego rozkazu. Razem z IP tworzy adres logiczny CS:IP kolejnej instrukcji.
- **DS** (ang. Data Segment) - rejestr informujący o segmencie z danymi.
- **ES** (ang. Extra Segment) - rejestr informujący o segmencie dodatkowym np. przy operacjach przesyłania łańcuchów.
- **SS** (ang. Stack Segment) - rejestr informujący o segmencie stosu

Rejestr flag w architekturze Intel x86			
bit	Skrót/wartość	opis	typ
0	CF	flaga przeniesienia (carry)	S
1	1	zarezerwowany	
2	PF	flaga parzystości (parity)	S
4	AF	flaga wyrównania (adjust)	S
6	ZF	flaga zera (zero)	S
7	SF	flaga znaku (sign)	S
8	TP	flaga umożliwiająca krokowe wykonanie (trap)	X
9	IF	flaga zezwolenia na przerwania (interrupt enable)	X
10	DF	flaga kierunku (direction)	C
11	OF	flaga przepełnienia (overflow)	S
12, 13	IOPL	poziom uprawnień we/wy (I/O privilege level, od 286)	X
14	NT	nested task flag (od 286)	X
16	RF	flaga wznowienia (resume, od 386)	X
17	VM	flaga trybu Virtual 8086 (od 386)	X
18	AC	alignment check (od 486SX)	X
19	VIF	Virtual interrupt flag (od Pentium)	X
20	VIP	Virtual interrupt pending (od Pentium)	X
21	ID	Identification (od Pentium)	X
3, 5, 15, 22-31	0	zarezerwowany	

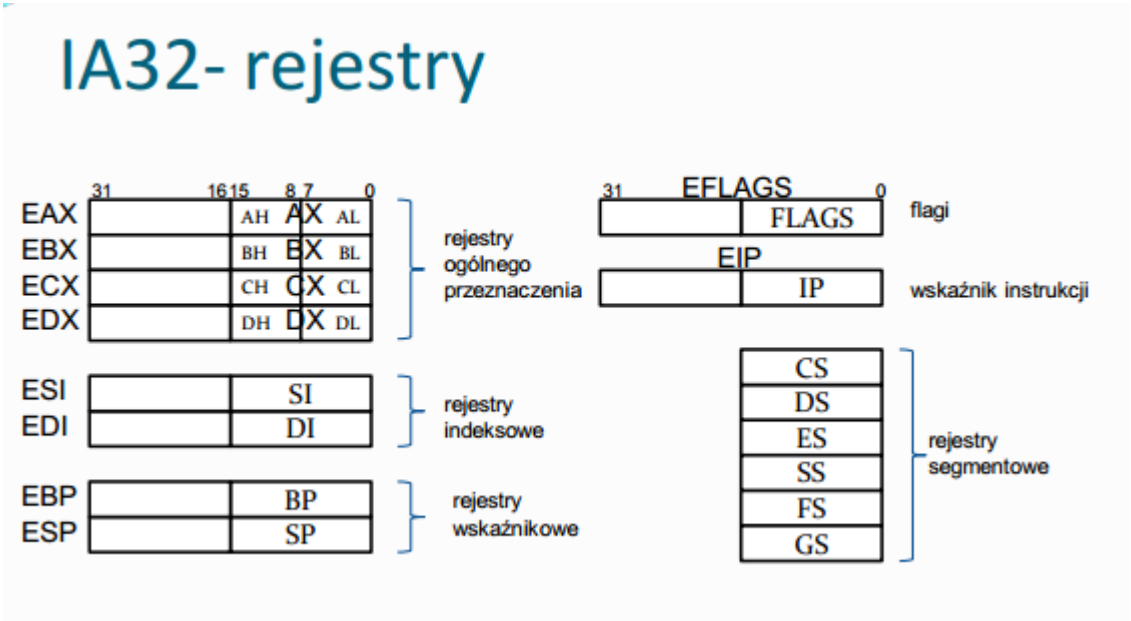
S: Znacznik stanu
C: Znacznik kontrolny
X: Znacznik systemowy

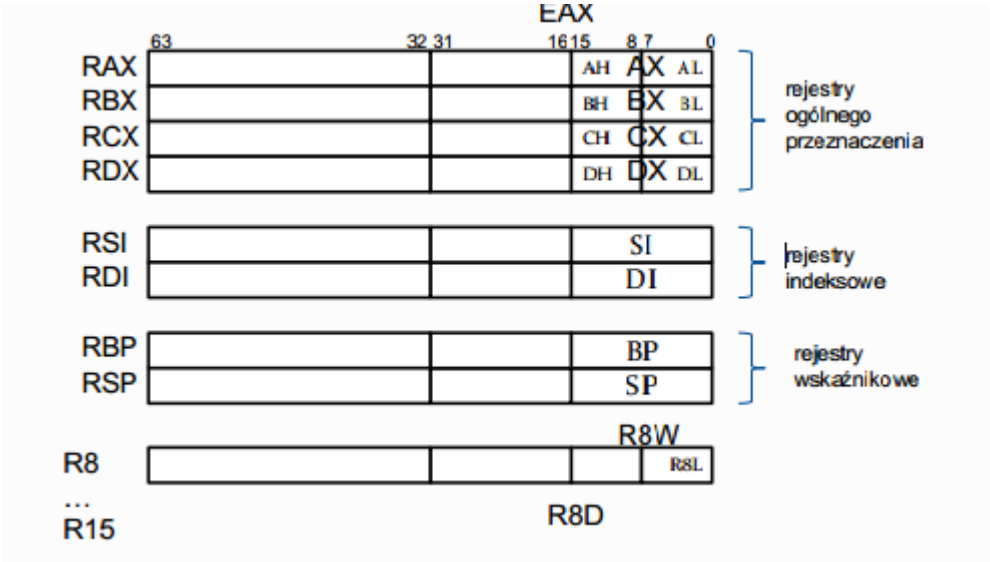
Rejestry MMX

Działają na nich instrukcje **całkowitoliczbowe** SIMD Wykorzystują rejestry koprocesora

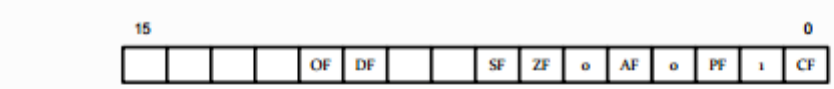
Rejestry XMM

Działają na nich instrukcje **zmiennoprzecinkowe** SIMD





Wykład 3 - Tryby adresowania, Instrukcje przesyłania



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

Tryb adresowania – rejestrowy

Argumentem instrukcji jest **rejestr**.
Przykłady:

```
push ebx;
mov edx, ebx;
inc ecx;
```

Tryb adresowania prosty natychmiastowy

Argumentem instrukcji jest **wartość.**
Przykłady:

```
Mov al, 5;
Mov edi, offset tablea;
Inz petla;
```

Tryb adresowania bezpośredni

Argumentem instrukcji jest **adres** w **pamięci**(**wskaźnik**)
Przykłady:

```
Mov al, [1234ec5fh]
Mov edi, tabale; pobiera pierwszy element
Mov zmienna, edx;
```

Tryb adresowania pośredni rejestrowy

Argumentem instrukcji jest **rejestr** – wskaźnik
Przykłady:

```
Mov al, [ecx]
Mov edi, [ebx]
Mov [edi], edx
```

Tryb adresowania pośredni – bazowy

Argumentem instrukcji jest **wskaźnik**
Przykłady:

```
Mov al, [ebx+5]
Mov edi, [ebx+tablica]
Mov [ebp+8], edx
```

Tryb adresowania indeksowany

Argumentem instrukcji jest **rejestr** – wskaźnik
Przykłady:

```
mov al, [esi]
mov edi, [esi*4+tablica]
mov [edi*8+tablica],  edx
```

Tryby adresowania - pośredni – bazowo-indeksowy

Argumentem instrukcji jest **wskaźnik**:
Przykłady:

```
mov al, [ebx+esi+3]
mov edi, [ebx+eax*4]
mov [ebp+edi*4+tablica],  edx
```

Wielkość danych

Można określić wielkość stosowanych danych:
Przykłady:

```
mov al, byte ptr [ebx+esi+3]
mov cx, word ptr [ebx+eax*4]
mov dword ptr [ebp+edi*4+tablica],  edx
```

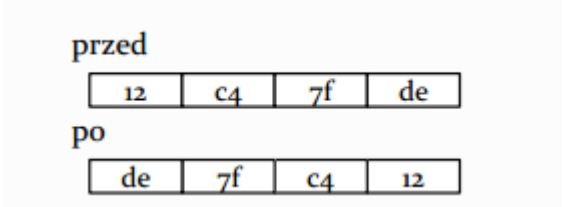
Instrukcje przesyłania

mov Przesyła zawartość źródła do miejsca przeznaczenia (cel). Wpływa na flagi: -

XCHG Zamienia zawartość źródła i celu. Wpływa na flagi: -

Bswap Zamienia bajty w argumencie. Wpływa na flagi: -

bswap eax



XADD Zamienia zawartość źródła i celu, a ich sumę umieszcza w miejscu przeznaczenia (cel)
Wpływa na flagi: **OSZAPC**

CMPXCHG Wpływa na flagi: **OSZAPC**

CMPXCHG8B Wpływa na flagi: **OSZAPC**

PUSH Przesyła zawartość argumentu na stos. Wpływa na flagi: -

POP Przesyła zawartość stosu do celu. Wpływa na flagi: -

PUSHF/PUSHFD Przesyła zawartość Flag/EFlag na stos. Wpływa na flagi: -

POPF/POPFD Pobiera zawartość Flag/EFlag ze stosu. Wpływa na flagi: **OSZAPC**

PUSHA/PUSHAD Przesyła zawartość di,si,bp,bx,dx,cx,ax,edi,esi,ebp,ebx,edx,ecx,eax na stos.
Wpływa na flagi: -

POPA/POPAD Przesyła zawartość stosu do di,si,bp,bx,dx,cx,ax,edi,esi,ebp,ebx,edx,ecx,eax.
Wpływa na flagi: -

CWD/CDQ konwertuje z zachowaniem znaku word na doubleword/doubleword na quadword
(ax na dx:ax, eax na edx:eax)
Wpływa na flagi: -

CBW/CWDE konwertuje byte (AL) na word(AX)/word(AX) na doubleword (EAX) z uwzględnieniem znaku.
Wpływa na flagi: **-**

MOVSX Przesyła zawartość źródła do rejestru celu z uwzględnieniem znaku. Cel posiada 2 /4 razy więcej bitów. Wpływa na flagi: **-**

MOVZX Przesyła zawartość źródła do rejestru celu z dopisaniem na starszych bitach zer. Cel posiada 2 /4 razy więcej bitów Wpływa na flagi: **-**

Wykład 4 Instrukcje arytmetyczne

- ADD dodawanie całkowitoliczbowe
- ADC dodawanie z przeniesieniem
- SUB odejmowanie
- SBB odejmowanie z pożyczką
- MUL mnożenie bez znaku
- IMUL mnożenie ze znakiem
- DIV dzielenie bez znaku
- IDIV dzielenie ze znakiem
- INC inkrementacja (zwiększenie)
- DEC dekrementacja (zmniejszenie)
- NEG zmiana znaku
- CMP porównanie

INC - Wpływa na flagi: **OSZAP**

DEC Wpływa na flagi: **OSZAP**

Cała reszta instrukcji arytmetycznych wpływa na flagi **OSZAPC**

Instrukcje arytmetyczne BCD

Wszystkie instrukcje BCD wpływają na flagi **OSZAPC**

- DAA korekta upakowanego kodu BCD po dodawaniu
- DAS korekta upakowanego kodu BCD po odejmowaniu
- AAA ASCII korekta po dodawaniu
- AAS ASCII korekta po odejmowaniu
- AAM ASCII korekta po mnożeniu
- AAD ASCII korekta przed dzieleniem

Nie działają w trybie 64 bitowym !!!

Przykład dodawania liczb BCD

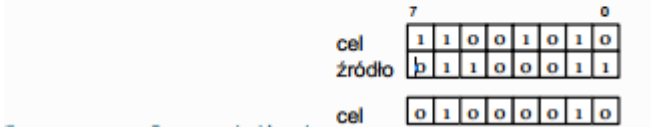
```
mov al,0
add al,al ;CF=0
petla: mov al,[esi] ;pobierz cyfrę źródła
adc al,ds:[edi] ;dodaj cyfrę celu z przeniesieniem
aaa ;korekta
mov ds:[edi],al ;zapamiętaj cyfrę
inc esi ;następna cyfra
inc edi
dec ecx
jnz petla ;CF nie zmieniło się od AAA!!!
```

Wykład 5 Instrukcje logiczne, przesunięć i rotacji

Instrukcje logiczne

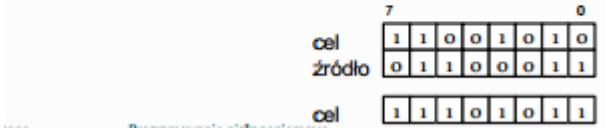
AND bitowa funkcja AND Wpływa na flagi: **OSZAPC** 0SZxP0

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



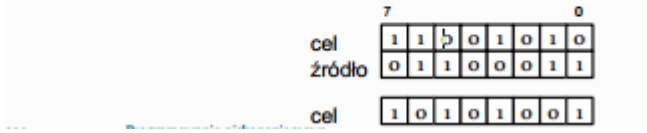
OR bitowa funkcja OR Wpływa na flagi: **OSZAPC** 0SZxP0

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



XOR bitowa funkcja OR Wpływa na flagi: **OSZAPC** 0SZxP0

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



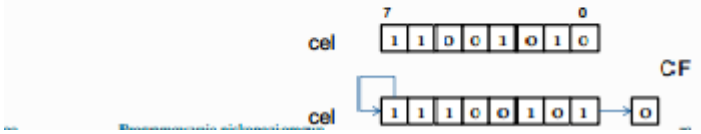
NOT bitowa funkcja NOT Wpływa na flagi: **brak**

```
not    eax
not    [ebx+esi*4]
```

Instrukcje przesunięć i rotacji

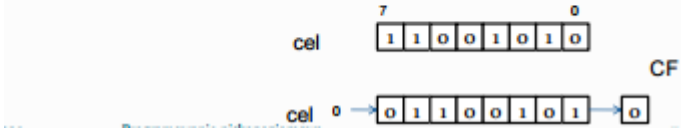
SAR przesunięcie arytmetyczne w prawo Wpływa na flagi: **OSZAPC** (0x)SZxPC

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



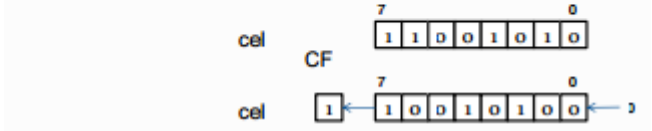
SHR przesunięcie logiczne w prawo Wpływa na flagi: **OSZAPC** (0x)SZxPC

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



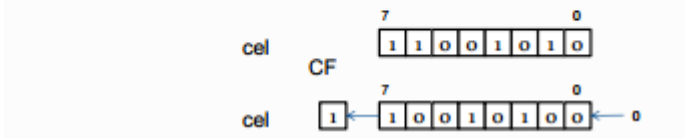
SAL przesunięcie arytmetyczne w lewo Wpływa na flagi: **OSZAPC** (0x)SZxPC

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



SHL przesunięcie logiczne w lewo Wpływa na flagi: **OSZAPC** (0x)SZxPC

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



SHRD przesunięcie w prawo double Wpływa na flagi: **OSZAPC** (Ox)SZxPC

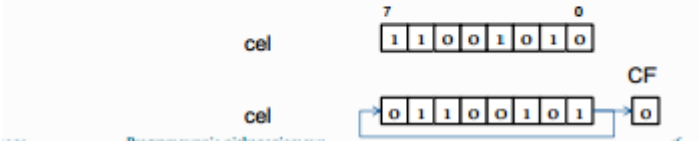
```
shrd  eax,ecx,15
shrd  [ebx+esi*4],edx,cl
```

SHLD przesunięcie w lewo double Wpływa na flagi: **OSZAPC** (Ox)SZxPC

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

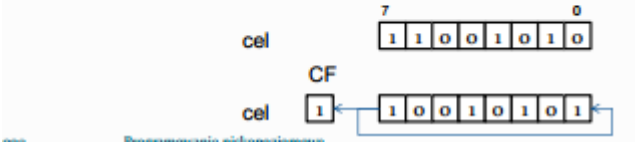
ROR rotacja w prawo Wpływa na flagi: **OSZAPC** (0x)----C

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



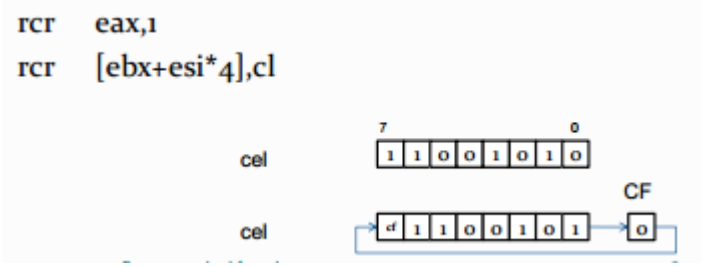
ROL rotacja w lewo Wpływa na flagi: **OSZAPC** (0x)----C

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



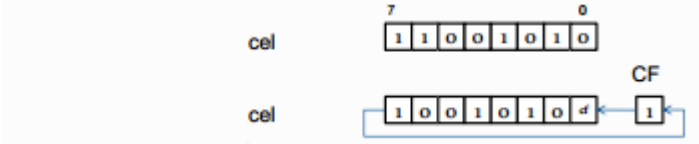
RCR rotacja w prawo przez przeniesienie Wpływa na flagi: **OSZAPC** (0x)----C

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



RCL rotacja w lewo przez przeniesienie Wpływa na flagi: **OSZAPC** (0x)----C

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



Wykład 6 Instrukcje warunkowe i skoku

Warunki dotyczące flag:

• E/Z	equal/ zero	ZF=1
• NE/NZ	not equal/ not zero	ZF=0
• C	carry	CF=1
• NC	not carry	CF=0
• O	overflow	OF=1
• NO	not overflow	OF=0
• S	sign (negative)	SF=1
• NS	not sign (non-negative)	SF=0
• P/PE	parity/ parity even	PF=1
• NP/PO	not parity/ parity odd	PF=0

Warunki porównania liczb:

• E/Z	equal/ zero	ZF=1
• NE/NZ	not equal/ not zero	ZF=0

Dla liczb bez znaku:

• A/NBE	above/ not below or equal	CF=0 i ZF=0
• AE/NB	above or equal/ not below	CF=0
• B/NAE	below/ not above or equal	CF=1
• BE/NA	below or equal/ not above	CF=1 lub ZF=1

Dla liczb ze znakiem

• G/NLE	greater/ not less or equal	ZF=0 i SF=OF
• GE/NL	greater or equal/ not less	SF=OF
• L/NGE	less/ not greater or equal	SF<>OF
• LE/NG	less or equal/ not greater	ZF=1 lub SF<>OF

Skoki warunkowe Jcc:

JE/JZ	Skocz jeśli equal/zero
JNE/JNZ	Skocz jeśli not equal/not zero
JA/JNBE	Skocz jeśli above/not below or equal
JAE/JNB	Skocz jeśli above or equal/not below
JB/JNAE	Skocz jeśli below/not above or equal
JBE/JNA	Skocz jeśli below or equal/not above
JG/JNLE	Skocz jeśli greater/not less or equal
JGE/JNL	Skocz jeśli greater or equal/not less
JL/JNGE	Skocz jeśli less/not greater or equal
JLE/JNG	Skocz jeśli less or equal/not greater
JC	Skocz jeśli carry
JNC	Skocz jeśli not carry
JO	Skocz jeśli overflow
JNO	Skocz jeśli not overflow
JS	Skocz jeśli sign (negative)
JNS	Skocz jeśli not sign (non-negative)
JPO/JNP	Skocz jeśli parity odd/not parity
JPE/JP	Skocz jeśli parity even/parity

Instrukcje sterujące przebiegiem programu

JMP	Skok bezwarunkowy
JCXZ/JECXZ	Skok jeśli zero w rejestrze CX/ECX
LOOP	Pętla z licznikiem ECX
LOOPZ/LOOPE	Pętla z licznikiem ECX i zero/equal
LOOPNZ/LOOPNE	Pętla z licznikiem ECX i not zero/not equal
CALL	Wywołanie podprogramu
RET	Powrót z podprogramu
IRET	Powrót z podprogramu obsługi przerwania
INT	Przerwanie programowe
INTO	Przerwanie przy przekroczeniu zakresu
BOUND	sprawdzenie ograniczeń indeksu tablicy
ENTER	Wysokopoziomowe wejście do podprogramu – utworzenie ramy stosu
LEAVE	Wysokopoziomowe wyjście z podprogramu – usunięcie ramy stosu

Instrukcje sterujące nie wpływają na żadne flagi.

Wykład 7 Operacje na znacznikach bitach i bajtach

Operacje na flagach

- **STC** Ustawienie CF Wpływa na flagi: **C**
- **CLC** Zerowanie CF Wpływa na flagi: **C**
- **CMC** Zanegowanie CF Wpływa na flagi: **C**
- **CLD** Zerowanie DF – f lagi kierunku Wpływa na flagi: -
- **STD** Ustawienie DF Wpływa na flagi: **D**
- **LAHF** Przesłanie f lag do rejestru AH Wpływa na flagi: -
- **SAHF** Przesłanie rejestru AH do f lag Wpływa na flagi: **SZAPC**
- **PUSHF/PUSHFD** Wysłanie f lag na stos Wpływa na flagi: -
- **POPF/POPFD** Pobranie f lag ze stosu Wpływa na flagi: **OSZAPC**
- **STI** Ustawienie IF – f lagi przerwań Wpływa na flagi: **I**
- **CLI** Zerowanie IF Wpływa na flagi: **I**

Operacje na bitach

- **BT** Testowanie bitu Wpływa na flagi: **OSZAPC** xxxxxC
- **BTS** Testowanie bitu z ustawianiem Wpływa na flagi: **OSZAPC** xxxxxC
- **BTR** Testowanie bitu z zerowaniem Wpływa na flagi: **OSZAPC** xxxxxC
- **BTC** Testowanie bitu z negacją Wpływa na flagi: **OSZAPC** xxxxxC
- **BSF** Przeszukiwanie bitów w przód Wpływa na flagi: **OSZAPC** xxZxxx
- **BSR** Przeszukiwanie bitów wstecz Wpływa na flagi: **OSZAPC** xxZxxx
- **TEST** Porównanie logiczne Wpływa na flagi: **OSZAPC** 0SZxP0

Operacje na łańcuchach

MOVS/MOVSB	Prześlij łańcuch/bajtów
MOVS/MOVSW	Prześlij łańcuch/słów
MOVS/MOVSD	Prześlij łańcuch/podwójnych słów
CMPS/CMPSB	Porównaj łańcuchy/bajtów
CMPS/CMPSW	Porównaj łańcuchy/słów
CMPS/CMPSD	Porównaj łańcuchy/podwójnych słów
SCAS/SCASB	Skanuj łańcuch/bajtów
SCAS/SCASW	Skanuj łańcuch/słów
SCAS/SCASD	Skanuj łańcuch/podwójnych słów
LODS/LODSB	Ładuj łańcuch/bajtów
LODS/LODSW	Ładuj łańcuch/słów
LODS/LODSD	Ładuj łańcuch/podwójnych słów
STOS/STOSB	Zapamiętaj łańcuch/bajtów
STOS/STOSW	Zapamiętaj łańcuch/słów
STOS/STOSD	Zapamiętaj łańcuch/podwójnych słów
REP	Powtarzaj dopóki ECX nie jest zerem
REPE/REPZ	Powtarzaj dopóki equal/zero
REPNE/REPNZ	Powtarzaj dopóki not equal/not zero

Instrukcja CMPS/CMPSB - Wpływa na flagi: OSZAPC
Instrukcja CMPS/CMPSW Wpływa na flagi: OSZAPC
Instrukcja CMPS/CMPSD Wpływa na flagi: OSZAPC
Instrukcja SCAS/SCASB Wpływa na flagi: OSZAPC
Instrukcja SCAS/SCASW Wpływa na flagi: OSZAPC
Instrukcja SCAS/SCASD Wpływa na flagi: OSZAPC
Reszta instrukcji nie wpływa na żadne flagi!

Przykłady użycia

Szuka wartości 77 w bufora. ZF=1 oznacza znalezienie żądanej wartości.

```
mov al,77
mov ecx,100
mov edi,bufor
repnz ds:scasb
```

Kopiuje zawartość bufora1 do bufora2.

```
mov ecx,100
mov esi,bufor1
mov edi,bufor2
rep movsb
```

Szuka wartości <>0 w buforze. ZF=0 oznacza znalezienie żądanej wartości.

```
mov al,0
mov ecx,100
mov edi,bufor
repz ds:scasb
```

Zeruje zawartość bufora.

```
mov eax,0
mov ecx,100
mov edi,bufor
rep ds:stosd
```

Wykład 8 Operacje zmiennoprzecinkowe

Operacje przesyłania danych

- FLD załadowanie argumentu zmiennoprzecinkowego
- FST zapisanie wartości z wierzchołka stosu
- FSTP zapisanie wartości z wierzchołka stosu i usunięcie go za stosu
- FILD załadowanie liczby całkowitej
- FIST zapisanie liczby całkowitej
- FISTP zapisanie liczby całkowitej ze zdjęciem ze stosu
- FBLD załadowanie liczby BCD
- FBSTP zapisanie liczby BCD i zdjęcie jej ze stosu
- FXCH zamiana zawartości rejestrów
- FCMOVE przesłanie warunkowe (jeśli równe)
- FCMOVNE przesłanie warunkowe (jeśli nie równe)
- FCMOVB przesłanie warunkowe (jeśli poniżej)
- FCMOVBE przesłanie warunkowe (jeśli poniżej lub równe)
- FCMOVNB przesłanie warunkowe (jeśli nie poniżej)
- FCMOVNBE przesłanie warunkowe (jeśli nie poniżej lub równe)
- FCMOVU przesłanie warunkowe (jeśli nieuporządkowane)
- FCMOVNU przesłanie warunkowe (jeśli uporządkowane)

Operacje arytmetyczne

- FADD dodawanie
- FADDP dodawanie ze zdjęciem ze stosu
- FIADD dodawanie liczby całkowitej
- FSUB odejmowanie
- FSUBP odejmowanie ze zdjęciem ze stosu
- FISUB odejmowanie liczby całkowitej
- FSUBR odejmowanie odwrotne
- FSUBRP odejmowanie odwrotne ze zdjęciem ze stosu
- FISUBR odejmowanie odwrotne liczby całkowitej
- FMUL mnożenie
- FMULP mnożenie ze zdjęciem ze stosu
- FIMUL mnożenie liczby całkowitej
- FDIV dzielenie
- FDIVP dzielenie ze zdjęciem ze stosu
- FIDIV dzielenie przez liczbę całkowitą
- FDIVR dzielenie odwrotne
- FDIVRP dzielenie odwrotne ze zdjęciem ze stosu
- FIDIVR dzielenie odwrotne liczby całkowitej
- FPREM obliczenie reszty (częściowej) z dzielenia
- FPREM1 obliczenie reszty (częściowej) z dzielenia zgodne z IEEE
- FABS obliczenie wartości bezwzględnej
- FCHS zmiana znaku
- FRNDINT zaokrąglenie do liczby całkowitej
- FSCALE skalowanie przez potęgę 2
- FSQRT obliczenie pierwiastka kwadratowego
- FXTRACT obliczenie wykładnika i mantysy

Operacje ładowania stałych

- FLD1 zapisanie +1.0 na wierzchołku stosu
- FLDZ zapisanie +0.0 na wierzchołku stosu
- FLDPI zapisanie π na wierzchołku stosu
- FLDL2E zapisanie $\log_2 e$ na wierzchołku stosu
- FLDLN2 zapisanie $\log_e 2$ (ln2) na wierzchołku stosu
- FLDL2T zapisanie $\log_2 10$ na wierzchołku stosu
- FLDLG2 zapisanie $\log_{10} 2$ na wierzchołku stosu

Operacje funkcji przestępnych

- FSIN Oblicza sinus
- FCOS Oblicza cosinus
- FSINCOS Oblicza sinus i cosinus
- FPTAN Oblicza (częściowy) tangens
- FPATAN Oblicza (częściowy) arcus tangens
- F2XM1 Oblicza $2^x - 1$
- FYL2X Oblicza $y*\log_2 x$
- FYL2XP1 Oblicza $y*\log_2(x+1)$

Operacje porównania

- FCOM porównanie liczb zmiennoprzecinkowych
- FCOMP porównanie liczb zmiennoprzecinkowych i zdjęcie ze stosu
- FCOMPP porównanie liczb zmiennoprzecinkowych i podwójne zdjęcie ze stosu
- FUCOM nieuporządkowane porównanie liczb zmiennoprzecinkowych
- FUCOMP nieuporządkowane porównanie liczb zmiennoprzecinkowych i zdjęcie ze stosu
- FUCOMPP nieuporządkowane porównanie liczb zmiennoprzecinkowych i podwójne zdjęcie ze stosu
- FICOM porównanie z liczbą całkowitą
- FICOMP porównanie z liczbą całkowitą i zdjęcie ze stosu
- FCOMI porównanie liczb zmiennoprzecinkowych i ustawienie EFLAGS
- FUCOMI nieuporządkowane porównanie liczb zmiennoprzecinkowych i ustawienie EFLAGS
- FCOMIP porównanie liczb zmiennoprzecinkowych, ustawienie EFLAGS i zdjęcie ze stosu
- FUCOMIP nieuporządkowane porównanie liczb zmiennoprzecinkowych, ustawienie EFLAGS i zdjęcie ze stosu
- FTST porównanie z liczbą 0.0
- FXAM sprawdzenie liczby zmiennoprzecinkowej

Instrukcja **FCOM/FCOMP/FCOMPP** Wpływa na flagi: **C3 C2 C0**
Instrukcja **FICOM/FICOMP** Wpływa na flagi: **C3 C2 C0**
Instrukcja **FCOMI/FCOMIP/FUCOMI/FUCOMIP** Wpływa na flagi: **ZF PF CF**
Instrukcja **FTST** Wpływa na flagi: **C3 C2 C0**
Instrukcja **FXAM** Wpływa na flagi: **C3 C2 C1 C0**

Operacje sterowania koprocesorem

FINCSTP	zwiększenie rejestru wskaźnika stosu koprocesora
FDECSTP	zmniejszenie rejestru wskaźnika stosu koprocesora
FFREE	zwolnienie rejestru zmiennoprzecinkowego
FINIT	inicjalizacja koprocesora po sprawdzeniu zgłoszenia błędu numerycznego
FNINIT	inicjalizacja koprocesora bez sprawdzenia zgłoszenia błędu numerycznego
FCLEX	zerowanie flag błędów numerycznych po sprawdzeniu zgłoszenia błędu numerycznego
FNCLEX	zerowanie flag błędów numerycznych bez sprawdzenia zgłoszenia błędu numerycznego
FSTCW	zapamiętanie rejestru sterowania po sprawdzeniu zgłoszenia błędu numerycznego
FNSTCW	zapamiętanie rejestru sterowania bez sprawdzenia zgłoszenia błędu numerycznego
FLDCW	wczytanie rejestru sterowania
FSTENV	zapamiętanie środowiska koprocesora po sprawdzeniu zgłoszenia błędu numerycznego
FNSTENV	zapamiętanie środowiska koprocesora bez sprawdzenia zgł. błędu numerycznego
FLDENV	wczytanie środowiska koprocesora
FSAVE	zapamiętanie zawartości koprocesora po sprawdzeniu zgłoszenia błędu numerycznego
FNSAVE	zapamiętanie zawartości koprocesora bez sprawdzenia zgłoszenia błędu numerycznego
FRSTOR	wczytanie zawartości koprocesora
FSTSW	zapamiętanie rejestru stanu po sprawdzeniu zgłoszenia błędu numerycznego
FNSTSW	zapamiętanie rejestru stanu bez sprawdzenia zgłoszenia błędu numerycznego
WAIT/FWAIT	czekanie na koprocesor
FNOP	nic nie robi

Wykład 8 Instrukcje typu SIMD

Single Instruction Multiple Data - przetwarzanych jest wiele strumieni danych przez jeden wykonywany program – cecha tzw. **komputerów wektorowych**.

Instrukcje SIMD dzieli się na:

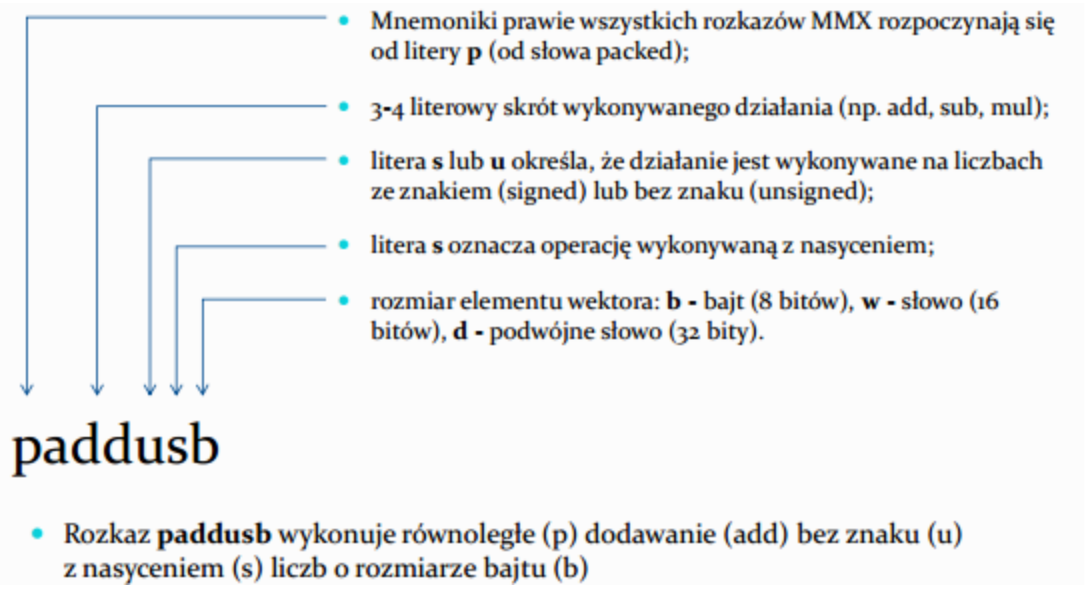
- **MMX** (MultiMedia eXtensions lub Matrix Math eXtensions) - liczby **całkowite**.
- **SSE** (Streaming SIMD Extensions) - liczby **zmiennoprzecinkowe**.

Instrukcje typu MMX

Przykłady zastosowań:

- wyświetlanie grafiki trójwymiarowej: przekształcenia geometryczne, cieniowanie, teksturowanie;
- dekodowanie obrazów JPEG i PNG;
- dekodowanie i kodowanie filmów MPEG (m.in. wyznaczanie transformat DCT i IDCT);
- filtrowanie sygnałów: obrazów statycznych, filmów, dźwięku;
- wyświetlanie grafiki dwuwymiarowej (blue box, maskowanie, przezroczystość);
- wyznaczanie transformat: Haara, FFT

Budowa rozkazów



Operacje przesłania

- **MOVD** przesłanie podwójnego słowa (double)
- **MOVQ** przesłanie poczwórnego słowa (quad)

Operacje konwersji

- PACKSSWB pakowanie z nasyceniem słów ze znakiem do bajtów
- PACKSSDW pakowanie z nasyceniem podwójnych słów ze znakiem do słów
- PACKUSWB pakowanie z nasyceniem słów bez znaku do bajtów
- PUNPCKHBW rozpakowanie z przeplotem starszych bajtów
- PUNPCKHWD rozpakowanie z przeplotem starszych słów
- PUNPCKHDQ rozpakowanie z przeplotem starszych podwójnych słów
- PUNPCKLBW rozpakowanie z przeplotem młodszych bajtów
- PUNPCKLWD rozpakowanie z przeplotem młodszych słów
- PUNPCKLDQ rozpakowanie z przeplotem młodszych podwójnych słów

Operacje arytmetyczne

- PADDB dodawanie wektorów bajtów
- PADDW dodawanie wektorów słów
- PADDD dodawanie wektorów podwójnych słów
- PADDSB dodawanie z nasyceniem wektorów bajtów ze znakiem
- PADDSW dodawanie z nasyceniem wektorów słów ze znakiem
- PADDUSB dodawanie z nasyceniem wektorów bajtów bez znaku
- PADDUSW dodawanie z nasyceniem wektorów słów bez znaku
- PSUBB odejmowanie wektorów bajtów
- PSUBW odejmowanie wektorów słów
- PSUBD odejmowanie wektorów podwójnych słów
- PSUBSB odejmowanie z nasyceniem wektorów bajtów ze znakiem
- PSUBSW odejmowanie z nasyceniem wektorów słów ze znakiem
- PSUBUSB odejmowanie z nasyceniem wektorów bajtów bez znaku
- PSUBUSW odejmowanie z nasyceniem wektorów słów bez znaku
- PMULHW mnożenie wektorów słów i zapamiętanie starszych słów wyniku
- PMULLW mnożenie wektorów słów i zapamiętanie młodszych słów wyniku
- PMADDWD mnożenie i dodawanie wektorów słów

Operacje porównania

- PCMPEQB sprawdzenie równości wektorów bajtów
- PCMPEQW sprawdzenie równości wektorów słów
- PCMPEQD sprawdzenie równości wektorów podwójnych słów
- PCMPGTB sprawdzenie większości wektorów bajtów ze znakiem
- PCMPGTW sprawdzenie większości wektorów słów ze znakiem
- PCMPGTD sprawdzenie większości wektorów podwójnych słów ze znakiem

Operacje logiczne

- PAND bitowy iloczyn logiczny
- PANDN bitowy iloczyn logiczny z negacją
- POR bitowa suma logiczna
- PXOR bitowa suma modulo 2

Operacje przesunięć

- PSLLW logiczne przesunięcie w lewo wektora słów
- PSLLD logiczne przesunięcie w lewo wektora podwójnych słów
- PSLLQ logiczne przesunięcie w lewo wektora poczwórnych słów
- PSRLW logiczne przesunięcie w prawo wektora słów
- PSRLD logiczne przesunięcie w prawo wektora podwójnych słów
- PSRLQ logiczne przesunięcie w prawo wektora poczwórnych słów
- PSRAW arytmetyczne przesunięcie w prawo wektora słów
- PSRAD arytmetyczne przesunięcie w prawo wektora podwójnych słów

Operacje sterujące

- FXSAVE zapisanie stanu x87 FPU i rejestrów SIMD
- FXRSTOR wczytanie stanu x87 FPU i rejestrów SIMD
- EMMS zwalnia wszystkie rejestry koprocesora
- LDMXCSR wczytanie rejestru MXCSR
- STMXCSR zapisanie rejestru MXCSR

Instrukcja EMMS

Zwalnia wszystkie rejestry koprocesora wpisując do pól TAG[i] rejestru stanu zawartości rejestrów stosu wartość 11b (rejestr pusty). Wszystkie instrukcje MMX wpisują do pól TAG[i] 0, co oznacza liczbę prawidłową! **Operacje MMX wprowadzone z SSE.**

- PAVGB — oblicza średnią z elementów wektorów bajtów bez znaku
- PAVGW — oblicza średnią z elementów wektorów słów bez znaku
- PEXTRW — wydobycie słowa
- PINSRW — wstawienie słowa
- PMAXUB — oblicza maksimum z elementów wektorów bajtów bez znaku
- PMAXSW — oblicza maksimum z elementów wektorów słów ze znakiem
- PMINUB — oblicza minimum z elementów wektorów bajtów bez znaku
- PMINSW — oblicza minimum z elementów wektorów słów ze znakiem
- PMOVMSKB — przesłanie maski bajtów
- PMULHUW — mnożenie wektorów słów bez znaku i zapamiętanie starszych słów wyniku
- PSADBW — oblicza sumę wartości bezwzględnych różnic
- PSHUFW — tasuje słowa w rejestrze MMX