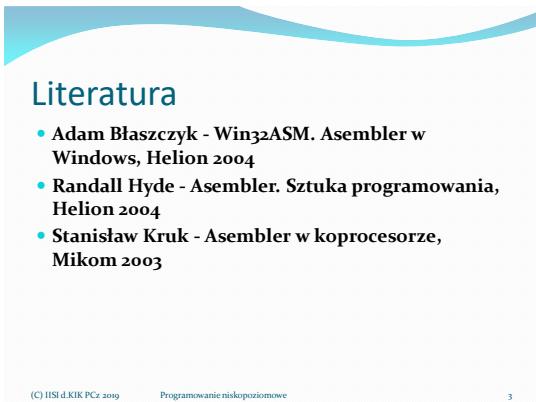


## Struktura przedmiotu

- Wykład
- Laboratorium

(C) IISI d.KIK PCz 2019 Programowanie niskopoziomowe 2



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

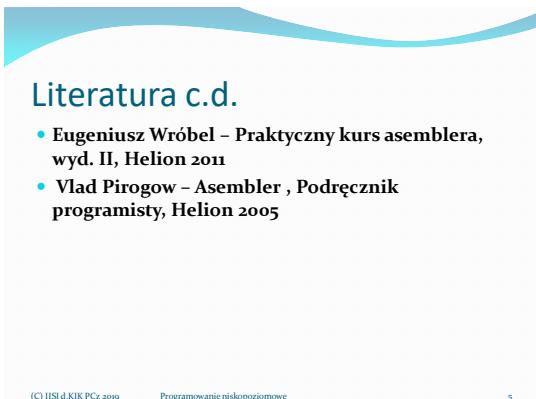
3

## Literatura c.d.

- Ryszard Goczyński, Michał Tuszyński – Mikroprocesory 80286, 80386 i i486, Komputerowa Oficyna Wydawnicza „HELP” 1991
- Michał Tuszyński, Ryszard Goczyński – Koprocesory arytmetyczne 80287 i 80387 oraz jednostka arytmetyki zmiennoprzecinkowej mikroprocesora i486, Komputerowa Oficyna Wydawnicza „HELP” 1992

(C) IISI d.KIK PCz 2019 Programowanie niskopoziomowe

4



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

5

## Literatura c.d.

- Intel® 64 and IA-32 Architectures Software Developer’s Manual
  - Basic Architecture
  - Instruction Set
  - System Programming Guide

[www.intel.com](http://www.intel.com)

(C) IISI d.KIK PCz 2019 Programowanie niskopoziomowe

6

## Narzędzia

- MASM, FASM, ...
- Delphi, Visual Studio, ...

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

7

## Języki programowania

- Wysokiego poziomu – Ada, Basic, C, C++, C#, Fortran, Java, Pascal (Delphi), SQL, ...
- Niskiego poziomu – asemblerы – odpowiadają kodowi wykonywanemu przez procesor

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

8

## Porównanie

	Języki wysokiego poziomu	Języki niskiego poziomu
Trudność programowania	niiska	duża
Przenośność	duża	mała
Wykorzystanie możliwości procesora i sprzętu	średnie	duże
Programowanie zadań wymagających czasowo	słabe	dobre
Przejrzystość kodu źródłowego	duża	mała
Przejrzystość kodu wynikowego	bardzo mała	duża
Szybkość tworzenia	duża	mała
Szybkość działania	mała	duża
Rozmiar kodu źródłowego	mały	średni
Rozmiar kodu wynikowego	duży	maly
Zajętość pamięci/dysku	duża	mała

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

9

## Porównanie

- Iloczyn skalarny

	Rozmiar	Zakres	Stan pośredni	Takty pamięciowe
Lokalny	1024	L_0	00000000	33
Wysoki	256,322791312028	2796	9132	
Delphi	256,322791312028	866	2742	
Assembler	256,322791312028			
PPV1x1	256,322791312028		865	2736
PPV1x2	256,322791312028		432	1383
PPV1x3	256,322791312028		357	1128
SEEDx1	256,322791312028		430	1371
SEEDx2	256,322791312028		228	675
SEEDx3	256,322791312028		184	543
SEEDx4	256,322791312028		290	921
Funkcje w bibliotece DLL				
PPV1x1a	256,322791312028		866	2742
PPV1x1b	256,322791312028		432	1383
PPV1x1c	256,322791312028		358	1134
SEEDx1a	256,322791312028		424	1362
SEEDx1b	256,322791312028		225	678
SEEDx1c	256,322791312028		186	552
Testy AIR				
AVX8x1a	256,322791312028		249	762
AVX8x1b	256,322791312028		167	495
AVX8x1c	256,322791312028		167	507
AVX8x2a	256,322791312028		422	1383
AVX8x2b	256,322791312028		226	681
AVX8x2c	256,322791312028		187	576

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

10

## Porównanie

- Rekonstrukcja obrazu – tomograf komputerowy

Opis programu	Ilość wątków	1 iteracja [ms]	20k iteracji [s]	Przyspieszenie
Oryginalny	1	688	13760 3149m20s	
Asembler x64	1	8,1	162 2m42s	84,938
Asembler x64 wielowątkowy i9-7900X	8	1,119047	22,38095	614,809
i9-7900X	10	0,994545	19,89089	691,774
10r 20t	16	1,017738	20,35476	676,009
	20	0,930187	18,60375	739,636
NVIDIA 1080Ti - 3584r		1,74473	34,8946	394,330
NVIDIA Titan V - 5120r		0,961224	19,22448	715,754

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

11

## Producenci procesorów

- AMD
- Cyrix
- IBM
- Intel**
- Nec
- Siemens
- Transmeta
- VIA
- Acorn Computers
- HP
- MOS Technology
- Motorola
- Silicon Graphix
- Zilog
- Texas Instruments
- Samsung

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

12

## Trocę historii

- F14 CADC (F-14A Central Air Data Computer)** - mikroprocesor zaprojektowany przez Steve'a Gellera i Raya Holta na potrzeby US Navy do myśliwca F-14 Tomcat.
  - Powstał w czerwcu 1970
  - niezwykle zaawansowany, 20-bitowy układ z techniką potokową
  - istnienie F-14 CADC zostało ujawnione dopiero w 1998 (z powodu tajemnicy wojskowej)

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

13



## Historia c.d.

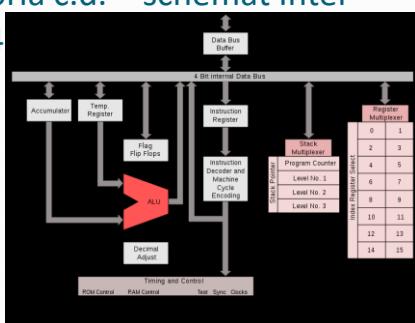
- Intel 4004** - 4-bitowy mikroprocesor zaprojektowany i produkowany przez firmę Intel od 1971
  - Powszechnie uznany za pierwszy mikroprocesor
  - Maksymalna częstotliwość taktowania - 740 kHz.
  - Osobna pamięć dla programu i danych (tzw. "architektura harwardzka").
  - 46 instrukcji.
  - 16 czterobitowych rejestrów.
  - 3-poziomowy stos.
  - 2300 tranzystorów (technologia produkcji 10 µm).

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

14

## Historia c.d. – schemat Intel 4004



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

15



## Historia c.d.

- Intel 8008** – pierwszy mikroprocesor 8-bitowy Intel
  - obudowa DIP18
  - 8-bitowa magistrala
  - dostęp do większej ilości RAM
  - 3-4 razy więcej mocy obliczeniowej niż procesory 4-bitowe.

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

16

## Historia c.d.



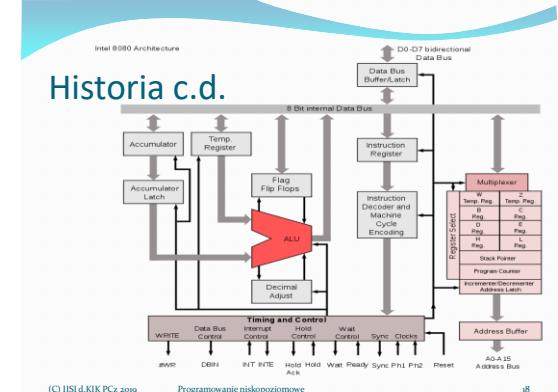
### Intel 8080

- wyprodukowany przez Intela w kwietniu 1974
- 8-bitowa szyna danych, pamięć adresowana 16-bitową szyną adresową.
- slowo 8-bitowe
- 72 instrukcje
- bezpśrednie adresowanie pamięci o pojemności do 64 KB
- arytmetyka dwójkowa i dziesiątna kodowana dwójkowo (BCD)
- 8 rejestrów programowych dostępnych dla programisty cykl pracy zapis,
- zegar zewnętrzny o częstotliwości 2-3 MHz (podstawowy cykl rozkazowy – 4 taktów)

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

17



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

18

## Historia c.d.

- Intel 8086 - procesor 16-bitowy wprowadzony w 1978
  - traktowany jako tymczasowy projekt przejściowy. Intel połknął wówczas swoje nadzieje w znacznie bardziej zaawansowanym 32-bitowym układzie 8800 (iAPX 432)
  - Głównym konstruktorem był Stephen Morse, który specjalizował się w oprogramowaniu. "Gdyby szefostwo Intelu chciało, by architektura ta przetrwała wiele generacji i przerodziła się w dzisiejsze procesory, to nigdy nie zleciłby tego zadania jednej osobie"



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

19

## Historia c.d.

- W 1980 IBM rozpoczyna pracę nad komputerem 5150
- Microsoft ma już gotowy interpreter języka Basic, który działał na układach 8086 i 8088
- IBM 5150 staje się standardem "Pytaliśmy ich, czy chcą mieć komputer od International Business Machines czy od firmy, która swą nazwę wzięła od owocu"



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

20

## Historia c.d.

- rozszerzenie listy rozkazów
- rozszerzenie możliwości adresowania operandów
- wprowadzenie segmentacji obszaru pamięci
- mechanizmy przyspieszenia pracy
- mechanizmy dla pracy wieloprocesorowej

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

21

## Historia c.d.

- Intel 80186 - procesor opracowany w firmie Intel w 1982.
  - posiadał nieco większą wydajność, kilkanaście nowych rozkazów i szybszy zegar
  - niektóre instrukcje były wykonywane 10-20 razy szybciej.
  - procesor wykorzystywany był głównie w systemach wbudowanych jako mikrokontrolery.



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

22

## Historia c.d.



- Intel 80286 - 16-bitowy procesor opracowany przez firmę Intel, pokazany po raz pierwszy 1 lutego 1982
  - mniej więcej dwa razy bardziej wydajny w porównaniu do procesora Intel 8086
  - posiadał 24-bitową szynę adresową mógł adresować aż 16MB pamięci RAM
  - wprowadzono nowe instrukcje,
  - nowy tryb adresowania pamięci (tryb chroniony)

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

23

## Historia c.d.

- Intel 80386 - 32-bitowy procesor opracowany przez firmę Intel w 1986
  - pierwszy 32-bitowy procesor z rodzinę x86. Architektura tego procesora została opracowana jeszcze zanim Intel wypuścił na rynek procesory poprzedniej serii 286, jednak procesor był zbyt skomplikowany, aby go w tamtym czasie wyprodukować.
  - 32-bitowa magistrala adresowa oraz 32-bitowa magistrala danych
  - rozszerzone do 32-bitów rejestry
  - nowe tryby adresowania
  - pracą w trzech trybach: rzeczywistym, chronionym i wirtualnym
  - dodanie do procesora jednostki MMU



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

24

## Historia c.d.



### • Intel 80486 - poprawna nazwa handlowa i486

- kilka (7) dodatkowych instrukcji
- cache na dane i instrukcje
- zintegrowany koprocesor arytmetyczny x87
- poprawiony interfejs szyny danych
- zastosowano pięciostopniowy potok.
- usprawnienia spowodowały, że i486 był mniej więcej dwukrotnie szybszy od podobnie taktowanego 80386

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

25

## Historia c.d.



### • Pentium - mikroprocesor się 22 marca 1993

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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

26

## Historia cd.



### • Pentium Pro - mikroprocesor szóstej generacji należący do rodziny x86 (październik 1995)

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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

27

## Historia c.d.



### • Pentium II - mikroprocesor zaprezentowany 7 maja 1997

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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

28

## Historia c.d.



### • Pentium III - procesor w 32-bitowej architekturze Intel'a (IA-32).

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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

29

## Historia c.d.



### • Pentium 4 – siódma generacja procesorów firmy Intel (od 20 listopada 2000)(wiele wersji)

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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

30

## Historia c.d.

- 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 (Execute Disable - wyłącza możliwość wykonywania instrukcji z oznaczonych stron),
  - ulepszoną technologię SpeedStep,
  - wersja czterordzeniowa



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

31

## Historia c.d.

- Intel Core i7
- modułowa budowa
- ósmiodzienny Nehalem składa się z 731 milionów tranzystorów
- SSE 4.2.
- technologia współbieżnej wielowątkowości
- dynamiczne zarządzanie zasilaniem
- wbudowanie kontrolera pamięci RAM
- technologia Quick-Path



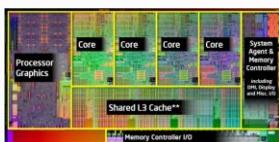
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

32

## Historia c.d.

- Intel Core i7 – 2 generacja - Sandy Bridge
  - modułowa budowa
  - 32-nanometrowy proces
  - wbudowany układ graficzny
  - instrukcje AVX
  - Turbo Boost 2.0
  - pamięć cache L3



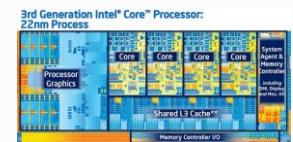
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

33

## Historia c.d.

- Intel Core i7 – 3 generacja - Ivy Bridge
  - modułowa budowa
  - 22-nanometrowy proces (tanzystory 3D)
  - wbudowany układ graficzny Intel HD Graphics
  - instrukcje AVX
  - gen. liczb losowych
  - PCI Express 3.0



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

34

## Historia c.d.

- Intel Core i7 – 4 generacja - Haswell
  - podniesiona wydajność pamięci cache
  - Zwiększoła wydajność i energooszczędność
  - instrukcje AVX2, FMA3
  - rozbudowany układ graficzny
  - wsparcie Direct3D 11.1 i OpenGL 4.0



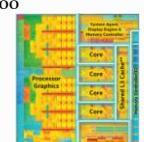
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

35

## Historia c.d.

- Intel Core i7 – 5 generacja - Broadwell
  - technologia 14 nm
  - zwiększoła wydajność i energooszczędność
  - obsługa pamięci DDR3L
  - rozbudowany układ graficzny Iris Pro 6200
  - 128 MB pamięci podrzędnej eDRAM
  - wsparcie Direct3D 11.2 i OpenGL 4.4



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

36

## Historia c.d.

- Intel Core i7 - 6 generacja - Skylake
- technologia 14 nm
- zwiększenie liczby rejestrów ogólnego przeznaczenia do 32
- obsługa pamięci DDR3L i DDR4
- instrukcje AVX 2 i rejestr YMM - 512 bitowe - w wersji XEON
- wsparcie Direct3D 12
- wsparcie dla Thunderbolt 3.0



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

37

## Historia c.d.

- Intel Core i7 - 7 generacja - Kaby Lake
- technologia 14 nm+
- zwiększenie częstotliwości zegara
- zwiększenie wydajności układu graficznego
- wsparcie Intel Optane Memory storage caching



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

38

## Historia c.d.

- Intel Core i7 - 8 generacja - Coffee Lake
- technologia 14++ nm
- zwiększenie liczby rdzeni do 6
- zwiększenie częstotliwości zegara
- zwiększenie wydajności układu graficznego



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

39

## Historia c.d.

- Intel Core i7 - 9 generacja - Coffee Lake
- technologia 14++ nm
- zwiększenie liczby rdzeni do 8
- zwiększenie rozmiaru pamięci cache L3
- AVX512 - rejestr YMM - 512 bitowe - w wersji X



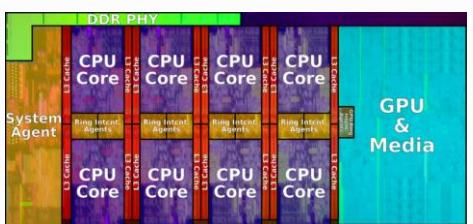
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

40

## Historia c.d.

- Intel Core i9 - 9 generacja - Coffee Lake



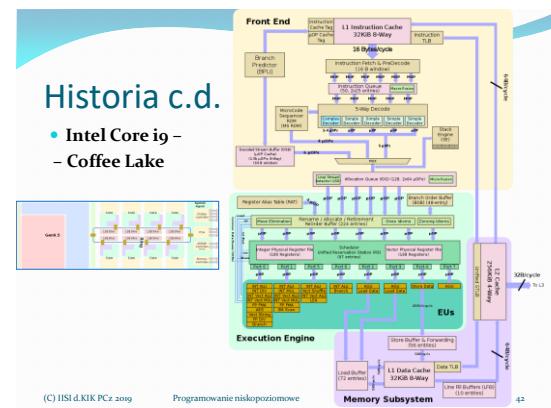
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

41

## Historia c.d.

- Intel Core i9 - Coffee Lake



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

42

## Zestawienie

Nazwa procesora	Rok	Maks. częstotliwość taktowania (w momencie wprowadzenia, MHz)	Liczba tranzystorów (mln.) Dane szacunkowe
Intel 8086	1978	8	0,029
Intel 80186	1982	12	0,055
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	1996	233	4,5
Pentium II	1997	266	7
Pentium III	1999	500	8,3
Pentium 4	2000	1500	42
Pentium 4 z EM64T	2003	2200	228
Pentium D	2004	3200	230
Intel Core 2	2006	3000	321
Intel Core i7	2008	3400	731
Intel Core i7 2600K	2011	3400-3800	995
Intel Core i7 3770K	2012	3500-3900	1400
Intel Core i7 5700K	2015	4000-4400	1750
Intel Core i9 5900K	2018	3600-5000	>3000
Intel Xeon Phi KNM - 72r	2017	1500-1600	8000

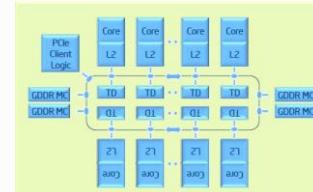
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

43

## Intel Xeon Phi - Knights Corner

- do 61 rdzeni połączonych w dwukierunkowy pierścień

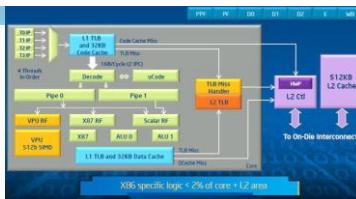


(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

44

## Xeon Phi



- rdzenie wykonują instrukcje SIMD - FMA3 na danych 512 bitowych, co oznacza, że jedna instrukcja przetwarza osiem zestawów danych podwójnej precyzyji albo 16 pojedynczej precyzyji - zwiększa to jeszcze stopień zrównoleglenia wykonywanych operacji.
- każdy rdzeń jest czterowątkowy,
- rdzenie posiadają dużą pamięć podręczną - 512KB.

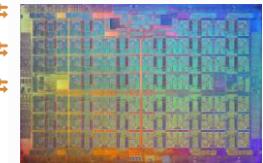
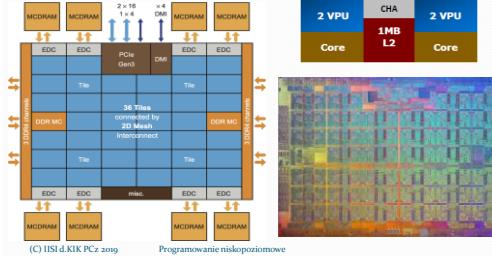
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

45

## Intel Xeon Phi - Knights Landing

- do 72 rdzeni połączonych pierścieniami



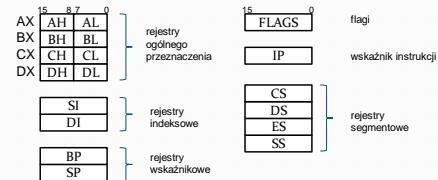
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

46

## Architektura procesora

### Procesor 8086 - rejesty



(C) IISI d.KIK PCz 2019 Programowanie niskopoziomowe

2

## Rejesty

- **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.

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

3

4

## Rejesty c.d.

- **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.

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

4

## Rejesty c.d.

- **IP (ang. Instruction Pointer)** – zawiera adres aktualnie wykonywanej instrukcji, może być modyfikowany przez rozkazy sterujące pracę programu.
- **FLAGS** – rejestr znaczników.

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

5

6

## Rejesty c.d. - 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 przesyłaniałańcuchów.
- **SS (ang. Stack Segment)** - rejestr informujący o segmencie stosu.

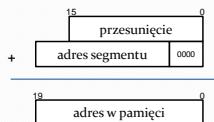
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

6

## Adres w trybie rzeczywistym

powstaje w wyniku sumowania położenia segmentu i przesunięcia w nim.

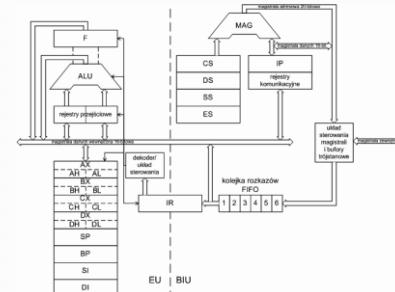


(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

7

## Architektura 8086

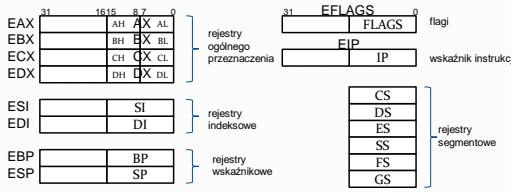


(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

8

## IA32- rejesty



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

9

## Rejestr flag

Rejestr flag w architekturze Intel 8086		
nr	Skrot/wartosc	opuszczenie
0	CF	flaga przeniesienia (carry)
1		zarezerwowany
2	PF	flaga parzystosci (parity)
4	AF	flaga znakowa (adjust)
5	ZF	flaga zera (zero)
7	SF	flaga znaku (sign)
8	TP	flaga umożliwiaca krotkowe wykonywanie (trap)
9	IF	flaga zezwolenia na przerwania (interrupt enable)
10	DF	flaga kierunku przekroju (dir)
11	OF	flaga przepelenia (overflow)
12..13	IOPL	poziom uprawnien we/wy (I/O privilege level, od 286)
14	NT	nested task flag (od 286)
16	RF	flaga wznowienia (resume, od 386)
17	VM	flaga trybu Virtual 8086 (od 386)
18	AC	alignment check (od 486SX)
19	VIF	virtual interrupt flag (od Pentium)
20	VIP	Virtual interrupt pending (od Pentium)
21	ID	Identification (od Pentium)
3..5, 15..23, 31..32..33		zarezerwowany

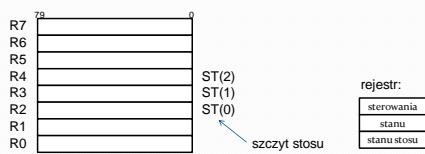
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

10

## Koprocesor

stos rejestrów



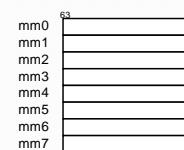
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

11

## Rejestry MMX

Działają na nich instrukcje całkowitoliczbowe SIMD  
Wykorzystują rejesty koprocesora



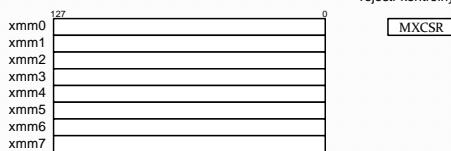
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

12

## Rejestry XMM

Działają na nich instrukcje zmiennoprzecinkowe SIMD

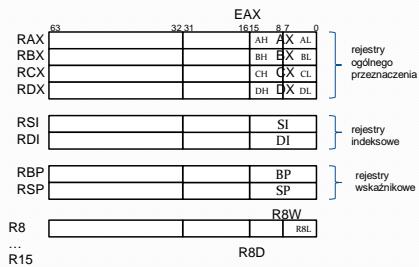


(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

13

## EM64T- rejesty

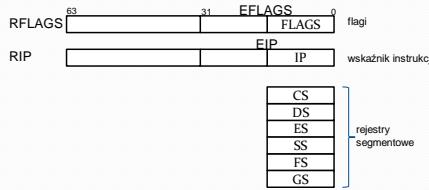


(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

14

## EM64T- rejesty



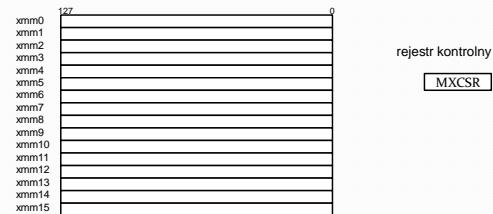
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

15

## EM64T- rejesty XMM

Działają na nich instrukcje zmiennoprzecinkowe SIMD



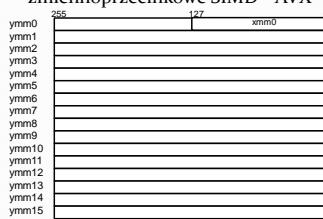
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

16

## AVX- Advanced Vector eXtensions

Rejestry ymm - działają na nich instrukcje zmiennoprzecinkowe SIMD - AVX



(C) IISI d.KIK PCz 2019

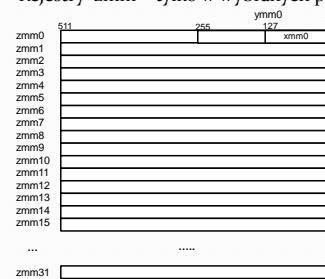
Programowanie niskopoziomowe

17

## AVX512-

## Advanced Vector eXtensions

Rejestry zmm - tylko w wybranych procesorach

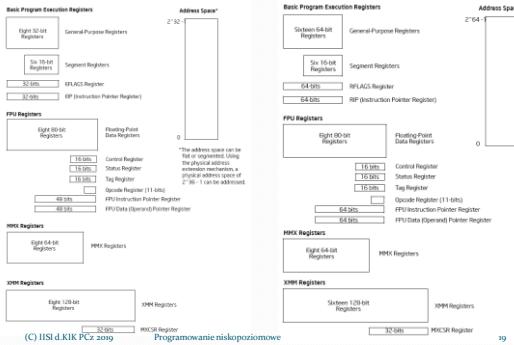


(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

18

## Środowisko 32 i 64 bitowe

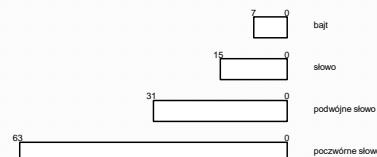


(C) IISI d.KIK PCz 2019

19

## Liczbowe typy danych

Liczby całkowite bez znaku



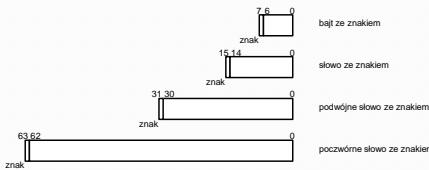
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

20

## Liczbowe typy danych

Liczby całkowite ze znakiem



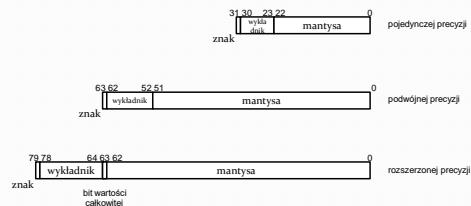
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

21

## Liczbowe typy danych

Liczby zmiennoprzecinkowe

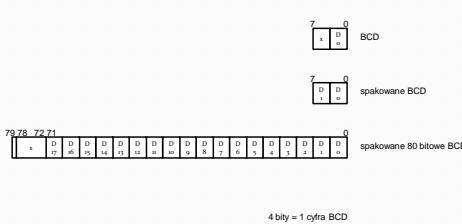


(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

22

## Typy BCD

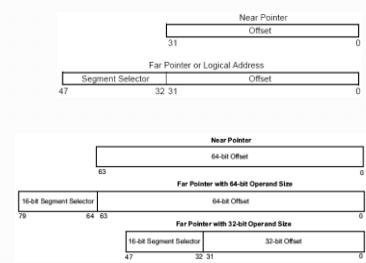


(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

23

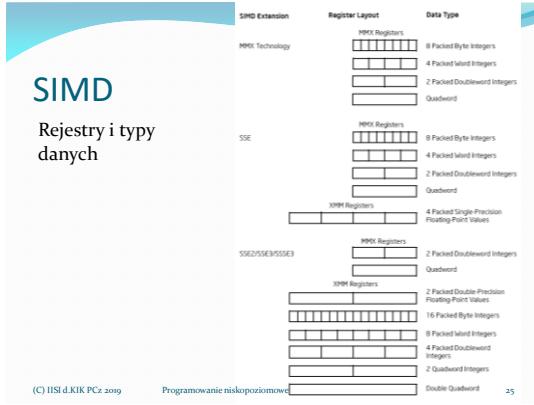
## Wskaźniki w trybie 32 i 64 bitowym



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

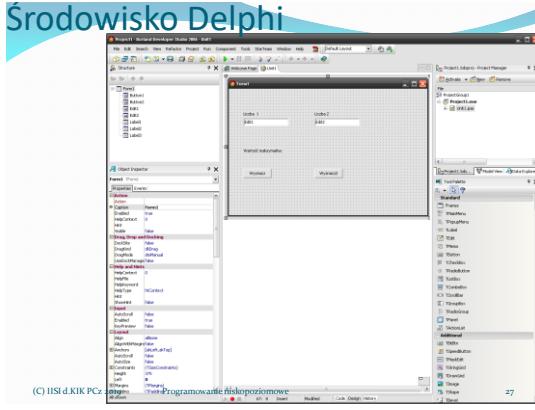
24



### Kilka instrukcji

add	eax,edx
sub	rax,rbx
mul	ecx ;edx:eax=eax*ecx
inc	ecx
dec	rcx
push	bp
pop	eax
jz	etykieta
call	podprogram
ret	
cmp	eax,ecx
jnz	etykieta
jc	etykieta
jnc	etykieta

(C) IISI d.KIK PCz 2019 Programowanie niskopoziomowe 26



### Użycie rejestrów

Rejestry	Windows 32	Windows 64
do użycia	EAX, ECX, EDX, ST(o)-ST(7), Ko-K7, xMMo-xMM7,	RAX, RCX, RDX, R8-R11, ST(o)-ST(7), Ko-K7, xMMo-xMM5, xMM6-xMM15
do zabezpieczenia	EBX, ESI, EDI, EBP	RBX, RSI, RDI, RBP, R12-R15, xMM6-xMM15
parametry funkcji	cdecl, stdcall, pascal, Gnu C: na stosie, fastcall Microsoft/Gnu : ecx, edx fastcall Borland eax, edx, ecx thiscall Microsoft ecx	RCX, RDX, R8, R9 lub xMMo-xMM3 reszta na stosie.
zwracające wartość funkcji	EAX, EDX, ST(o)	RAX, xMMo

**x=X, Y lub Z**

(C) IISI d.KIK PCz 2019 Programowanie niskopoziomowe 28

### Przykład

```
function TForm1.MyMax(x,y:integer):integer;
begin
  mov eax,x
  cmp eax,y
  jnc @_exit
  mov eax,y
 @_exit:
 // mov result, eax
 end;

function TForm1.MyMax2(x,y:integer):integer;
begin
  if x>y then
    MyMax2:=x
  else
    Result:=y;
end;
```

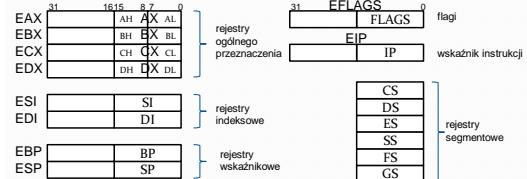
(C) IISI d.KIK PCz 2019 Programowanie niskopoziomowe 29

### Kod wynikowy

(C) IISI d.KIK PCz 2019 Programowanie niskopoziomowe 30

## Tryby adresowania Instrukcje przesyłania

## IA32- rejesty



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

2

## Rejestr flag



Rejestr flag w architekturze Intel x86		
bit	Skrot/wartosc	opis
0	CF	flaga przeniesienia (carry)
2	P	flaga parzystosci (parity)
4	AF	flaga przyrostu (adjust)
6	ZF	flaga zerowa (zero)
7	SF	flaga znaku (sign)
8	IDF	flaga kierunku (direction)
10	OF	flaga przepelenia (overflow)

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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

3

## Format rozkazów

etykieta:	mnemonik	argumenti, argument2	komentarz
etykieta:	mnemonik	cel, źródło	komentarz
etykieta:	mnemonik	argumenti	
	mnemonik	argumenti, argument2	
Np.:			
	ret		
	pop	eax	
	mov	edx,ecx ;zapamiętaj licznik	
	mov	rax,1001	

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

4

## Tryby adresowania - rejestrowy

Argumentem instrukcji jest rejestr:

```
push ebx
mov edx,ebx
inc ecx
dec r9
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

5

## Tryby adresowania - prosty - natychmiastowy

Argumentem instrukcji jest wartość (zawiera się w kodzie rozkazu):

```
mov al,5
mov r10d,32
mov edi,offset tabela
jnz petla
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

6

## Tryby adresowania - bezpośredni

Argumentem instrukcji jest adres w pamięci (wskaźnik):

```
mov al, [1234ec5fh]
mov edi, tabela ;pobiera pierwszy element
mov zmienna, rdx
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

7

## Tryby adresowania - pośredni - rejestrowy

Argumentem instrukcji jest регистр – wskaźnik:

```
mov al, [rcx]
mov edi, [ebx]
mov [edi], edx
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

8

## Tryby adresowania - pośredni - bazowy

Argumentem instrukcji jest wskaźnik:

```
mov al, [ebx+5]
mov edi, [ebx+tablica]
mov [rbp+8], rdx
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

9

## Tryby adresowania - pośredni - indeksowy

Argumentem instrukcji jest register – wskaźnik:

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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

10

## Tryby adresowania - pośredni – bazowo-indeksowy

Argumentem instrukcji jest wskaźnik:

```
mov al, [ebx+esi+3]
mov edi, [ebx+eax*4]
mov [rbp+rdi*8+tablica], rdx
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

11

## Wielkość danych

Mogą określać wielkość stosowanych danych:

```
mov al, byte ptr [ebx+esi+3]
mov cx, word ptr [ebx+eax*4]
mov dword ptr [ebp+edi*4+tablica], edx
mov qword ptr [rbp+rdi*8+tablica], rdx

inc byte ptr [ebx+esi+3]
dec word ptr [ebx+eax*4]
inc dword ptr [ebp+edi*4+tablica]
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

12

## Przedrostki segmentowe

Można podać segment do danych:

```
mov al, es:byte ptr [ebx+esi+3]
mov cx, cs:word ptr [ebx+eax*4]
mov ss:[ebp+4], edx
```

Przyporządkowanie rejestrów

```
esp, ebp: ss
eax, ebx, ecx, edx, edi, esi: ds.
eip: cs
```

Analogicznie rejesty 16 i 64 bitowe.

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

13

14

Wpływ na flagi: -

## Instrukcje przesyłania

- MOV przesyła dane między rejestrami, pamięcią zamień
- XCHG zamień bajty
- BSWAP wymień i dodaj
- XADD porównaj i wymień
- CMPXCHG porównaj i wymień 8(16) bajtów
- CMPXCHG8(i6)B wyslij na stos
- PUSH zdejmij ze stosu
- POP wyslij rejestry na stos
- PUSHF/PUSHFD/PUSHFQ wyslij na stos flagi
- POPF/POPFD/POPFQ zdejmij ze stosu flagi
- PUSHA/PUSHAD wyslij rejesty na stos
- POPA/POPAD zdejmij rejesty ze stosu
- CWD/CDQ/CQO konwertuj word na dword/dword na qword
- CBW/CWDE /CDQE konwertuj byte na word/word na doubleword w rejestrze EAX/ doubleword na quadword w RAX
- MOVSX/MOVSD przeslij i rozszerz znakiem
- MOVZX przeslij i rozszerz zerem

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

14

Wpływ na flagi: -

## Instrukcja MOV

```
mov cel,źródło
```

Przesyła zawartość źródła do miejsca przeznaczenia (cel).

```
mov al,bl
mov [ebp+4], edx
mov zmienna, eax
mov rcx,licznik
mov [ebp+edi*4+tablica], edx
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

15

16

Wpływ na flagi: -

## Instrukcja XCHG

```
xchg cel,źródło
```

Zamienia zawartość źródła i celu.

xchg	ax,zmienna
xchg	ecx,[ebp+4]
xchg	rax,r12

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

16

Wpływ na flagi: -

## Instrukcja BSWAP

```
bswap rejestr
```

Zamienia bajty w argumencie – 32/64 bity.

```
bswap eax
bswap rdx
```

przed

12	c4	7f	de
----	----	----	----

po

de	7f	c4	12
----	----	----	----

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

17

18

Wpływ na flagi: -

## Instrukcja XADD

```
xadd cel,źródło
```

Zamienia zawartość źródła i celu(8/16/32/64 bity), a ich sumę umieszcza w miejscu przeznaczenia (cel).

xadd	al,bl
xadd	eax,zmienna
xadd	edx,[ebx+esi*4]
xadd	rcx,r8

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

18

Wpływ na flagi: OSZAPC

Wpływa na flagi: OSZAPC

## Instrukcja CMPXCHG

CMPXCHG arg1,arg2

Działanie:

```
if acc==arg1 then
    arg1==arg2
else
    acc==arg1
```

acc=al,ax,eax,rax

arg2 - rejestr

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

19

Wpływa na flagi: OSZAPC

## Instrukcja CMPXCHG8(16)B

CMPXCHG8(16)B cel

Działanie:

```
if (E(R)DX:E(R)AX==cel) then
    cel=e(r)cx:e(r)bx
else
    e(r)dx:e(r)ax==cel
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

20

Wpływa na flagi: -

## Instrukcja PUSH

push arg

Przesyła zawartość argumentu na stos.

```
push    eax
push    rdx
push    ds
push    1234
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

21

Wpływa na flagi: -

## Instrukcja POP

pop cel

Przesyła zawartość stosu do celu.

```
pop    bx
pop    ecx
pop    rdx
pop    [edx+edi+4]
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

22

Wpływa na flagi: -

## Instrukcja PUSHF/PUSHFD/PUSHFQ

pushf/pushfd/pushfq

Przesyła zawartość Flag/Eflag/Rflag na stos.

```
pushf
pushfd
pushfq
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

23

Wpływa na flagi: OSZAPC

## Instrukcja POPF/POPFD/POPFQ

popf/popfd/popfq

Pobiera zawartość Flag/Eflag/Rflag ze stosu.

```
popf
popfd
popfq
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

24

Wpływa na flagi: -

## Instrukcja PUSHA/PUSHAD

pusha/pushad

Przesyła zawartość di,si,bp,bx,dx,cx,ax/  
edi,esi,ebp,ebx,edx,ecx,eax na stos.

**Instrukcja nie działa w trybie 64-bitowym.**

```
pusha
pushad
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

25

Wpływa na flagi: -

## Instrukcja POPA/POPAD

popa/popad

Przesyła zawartość stosu do di,si,bp,bx,dx,cx,ax/  
edi,esi,ebp,ebx,edx,ecx,eax.

**Instrukcja nie działa w trybie 64-bitowym.**

```
popa
popad
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

26

Wpływa na flagi: -

## Instrukcja CWD/CDQ/CQO

CWD/CDQ konwertuje z zachowaniem znaku word na doubleword/doubleword na quadword/quadword na octaword (ax na dx:ax, eax na edx:eax, rax na rdx:rax).

```
cwd
cdq
cqo
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

27

Wpływa na flagi: -

## Instrukcja CBW/CWDE/CDQE

CBW/CWDE

konwertuje byte (AL) na word(AX)/word(AX) na doubleword (EAX)/doubleword (EAX) na quadword (RAX) z uwzględnieniem znaku.

```
cbw
cwde
cdqe
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

28

Wpływa na flagi: -

## Instrukcja MOVSX/MOVSDX

movsx cel,źródło

Przesyła zawartość źródła do rejestru celu z uwzględnieniem znaku. Cel posiada 2/4/8 razy więcej bitów.

```
movsx    eax, bl
movsx    cx, al
movsxd   r8,edx ;movsxd tylko dla 32 na 64
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

29

Wpływa na flagi: -

## Instrukcja MOVZX

movzx cel,źródło

Przesyła zawartość źródła do rejestru celu z dopisaniem na starszych bitach zer. Cel posiada 2/4/8 razy więcej bitów. Źródło 8/16 bitów.

```
movzx    eax, bl
movzx    cx,al
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

30

## Przykład

Wypełnienie wartościami od 0 do 255 tabeli bajtów

```
mov  ecx,256  
mov  eax,0  
mov  edi,o  
p1: mov  [edi+tabela],al  
inc  edi  
inc  eax  
dec  ecx  
jnz  p1
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

31

## Przykład

Przepisanie wartości integer (32 bity) z tabeli tab1 do tabeli tab2.

```
mov  rcx,65536  
mov  rdi,o  
p1: mov  eax,[tab1+4*rdi]  
     mov  [tab2+4*rdi],eax  
     inc  rdi  
     dec  rcx  
     jnz  p1
```

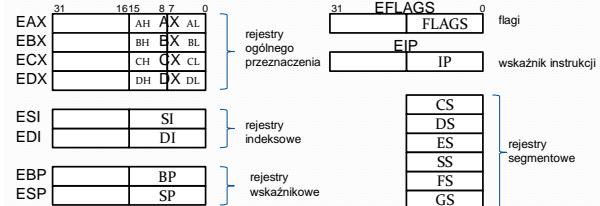
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

32

## Instrukcje arytmetyczne

## IA32- rejesty



(C) IISI d.KIK PCz 2019 Programowanie niskopoziomowe

## Rejestr flag



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

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

Programowanie niskopoziomowe

3

## Instrukcje arytmetyczne

- ADD dodawanie całkowitoliczbowe
- ADC dodawanie z przeniesieniem
- ADCX dodawanie z przeniesieniem bez znaku
- ADOX dodawanie z przeniesieniem bez znaku
- SUB odejmowanie
- SBB odejmowanie z pożyczką
- MUL mnożenie bez znaku
- MULX 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

(C) IISI d.KIK PCz 2019 Programowanie niskopoziomowe

4

Wpływ na flagi: OSZAPC

## Instrukcja ADD

add cel, źródło

Dodaje zawartość źródła i celu, sumę umieszcza w miejscu przeznaczenia (cel).

cel:=cel+źródło

```
add eax, zmienna
add edx, [ebx+esi*4]
add rcx, rbx
```

**Uwaga:**

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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

5

Wpływ na flagi: OSZAPC

## Instrukcja ADC

adc cel, źródło

Dodaje zawartość źródła, celu i przeniesienia, sumę umieszcza w miejscu przeznaczenia (cel).

cel:=cel+źródło+CF

```
adc eax, zmienna
adc edx, [ebx+esi*4]
add rcx, rbx
```

**Uwaga:**

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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

6

Wpływ na flagi: -----C  
Wymagane: ADX

## Instrukcja ADCX

adcx cel, źródło

Dodaje bez znaku zawartość źródła, celu i przeniesienia, sumę umieszcza w miejscu przeznaczenia (cel – rejestr 32|64 bitowy).

cel:=cel+źródło+CF

adcx eax, zmienna

adcx edx, [ebx+esi\*4]

adcx rcx, rbx

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

7

Wpływ na flagi: O-----  
Wymagane: ADX

## Instrukcja ADOX

adox cel, źródło

Dodaje bez znaku zawartość źródła, celu i flagi przepelenia, sumę umieszcza w miejscu przeznaczenia (cel – rejestr 32|64 bitowy).

cel:=cel+źródło+OF

adox eax, zmienna

adox edx, [ebx+esi\*4]

adox rcx, rbx

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

8

Wpływ na flagi: OSZAPC

## Instrukcja SUB

sub cel, źródło

Odejmuje zawartość źródła od celu, różnicę umieszcza w miejscu przeznaczenia (cel).

cel:=cel-źródło

sub ecx, zmienna

sub ebx, [ebx+esi\*4]

sub rax, rdx

**Uwaga:**

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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

9

Wpływ na flagi: OSZAPC

## Instrukcja SBB

sbb cel, źródło

Odejmuje zawartość źródła od celu z uwzględnieniem pożyczki, różnicę umieszcza w miejscu przeznaczenia (cel).

cel:=cel-(źródło+CF)

sbb edx, zmienna

sbb eax, [ebx+esi\*4]

sbb rax, rdx

**Uwaga:**

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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

10

Wpływ na flagi: OxxxxC

## Instrukcja MUL

mul źródło

Mnoży bez znaku zawartość akumulatora (al, ax, eax, rax) i źródła, wynik umieszcza w miejscu przeznaczenia (ax, dx:ax, edx:eax, rdx:rax). Flagi CF i OF są zerem, jeśli starsza połowa bitów wyniku jest równa zero.

wynik:=acc\*źródło

mul zmienna

mul word ptr[ebx+esi\*4]

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

11

Wpływ na flagi: -----  
Wymaga BMI2

## Instrukcja MULX

mulx cel1,cel2,źródło

Mnoży bez znaku zawartość rejestru edx|rdx i źródła, wynik umieszcza w rejestrach celu(cel1:cel2).

cel1:cel2:=e(r)dx\*źródło

mulx ecx, ebx, [tab+esi\*4]

mulx rcx, rbx, rax

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

12

Wpływ na flagi: OxxxxC

## Instrukcja IMUL -1

imul źródło

Mnoży ze znakiem zawartość akumulatora (al, ax, eax, rax) i źródła, wynik umieszcza w miejscu przeznaczenia (ax, dx:ax, edx:eax, rdx:rax). Flagi CF i OF są zerem, jeśli iloczyn mieści się w młodszej połowie bitów wyniku.

wynik:=acc\*źródło

```
imul zmienna
imul ecx
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

13

Wpływ na flagi: OxxxxC

## Instrukcja IMUL -2

imul cel, źródło

Mnoży ze znakiem zawartość rejestru celu (16|32|64 bity) przez źródło, wynik umieszcza w miejscu przeznaczenia (cel). Flagi CF i OF są zerem, jeśli iloczyn mieści się w rejestrze celu.

cel:=cel\*źródło

```
imul eax, zmienna
imul rdx, rcx
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

14

Wpływ na flagi: OxxxxC

## Instrukcja IMUL -3

imul cel, źródło1, źródło2

Mnoży ze znakiem zawartość źródła1 (16|32|64 bity) przez źródło2 (stała), wynik umieszcza w miejscu przeznaczenia (cel). Flagi CF i OF są zerem, jeśli iloczyn mieści się w rejestrze celu.

cel:=źródło1\*źródło2

```
imul eax, zmienna, 5
imul cx, dx, 77
```

**Uwaga:**

Jeśli źródło2 jest adresowane w trybie prostym może mieć do 32 bitów.

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

15

Wpływ na flagi: xxxxxx

## Instrukcja DIV

div źródło

Dzieli bez znaku zawartość AX, DX:AX, EDX:EAX, RDX:RAX przez źródło, iloraz umieszcza w AL, AX, EAX, RAX a resztę w AH, DX, EDX, RDX.

```
div byte ptr zmienna
div ebx
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

16

Wpływ na flagi: xxxxxx

## Instrukcja IDIV

idiv źródło

Dzieli ze znakiem zawartość AX, DX:AX, EDX:EAX, RDX:RAX przez źródło, iloraz umieszcza w AL, AX, EAX, RAX a resztę w AH, DX, EDX, RDX.

```
idiv byte ptr zmienna
idiv ebx
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

17

Wpływ na flagi: OSZAP

## Instrukcja INC

inc cel

Zwiększa zawartość celu o 1.

```
inc zmienna
inc edx
inc rcx
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

18

Wpływ na flagi: OSZAP

## Instrukcja DEC

dec cel

Zmniejsza zawartość celu o 1.

```
dec zmienna
dec edx
dec r8
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

19

Wpływ na flagi: OSZAPC

## Instrukcja NEG

neg cel

Zmienia znak celu w kodzie U2.

cel:=-cel

```
neg ax
neg byte ptr[ebx+esi*4]
neg r11
```

Flaga CF=0, tylko dla argumentu=o.

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

Programowanie niskopoziomowe

20

Wpływ na flagi: OSZAPC

## Instrukcja CMP

cmp źródło1, źródło2

Porównuje zawartość źródła1 i źródła2, wynik nie jest zapamiętywany, tylko są ustawiane flagi.

źródło1-źródło2

```
cmp ax, zmienna
cmp edx, [ebx+esi*4]
cmp rcx, 123
```

Uwaga:

Jeśli źródło2 jest adresowane w trybie prostym może mieć do 32 bitów.

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

21

## Przykład – oblicz sumę kwadratów liczb w tablicy

mov eax, 0	wartość początkowa sumy
mov esi, eax	;indeks
mov ebx, offset wektor	;tablica liczb całkowitych
mov ecx, 123	;licznik
petla: mov edx, [ebx+esi*4]	
imul edx, edx	;kwadrat liczby
add eax, edx	;suma
inc esi	
dec ecx	
jnz petla	wynik w eax

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

22

## Instrukcje arytmetyczne BCD

- DAA korekta upakowanego kodu BCD po dodawaniu  
Decimal adjust after addition
- DAS korekta upakowanego kodu BCD po odejmowaniu  
Decimal adjust after subtraction
- AAA ASCII korekta po dodawaniu  
ASCII adjust after addition
- AAS ASCII korekta po odejmowaniu  
ASCII adjust after subtraction
- AAM ASCII korekta po mnożeniu  
ASCII adjust after multiplication
- AAD ASCII korekta przed dzieleniem  
ASCII adjust before division

**!!! Nie działają w trybie 64 bitowym !!!**

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

23

Wpływ na flagi: OSZAPC

## Instrukcja DAA

daa

Korekta upakowanego kodu BCD (w AL) po dodawaniu. Polega na dodaniu 6 najpierw do młodszej półbjata, a potem do starszego, jeśli ich zawartości były większe od 9 lub wystąpiło przeniesienie AF (CF)

daa

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

24

Wpływa na flagi: OSZAPC

## Instrukcja DAS

das

Korekta upakowanego kodu BCD (w AL) po odejmowaniu. Polega na odjęciu 6 najpierw od młodszej półabajta, a potem od starszego, jeśli ich zawartości były większe od 9 lub wystąpiło przeniesienie AF (CF).

das

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

25

Wpływa na flagi: OSZAPC

## Instrukcja AAA

aaa

Korekta nieupakowanego kodu BCD (w AL) po dodawaniu. Polega na dodaniu 6 do młodszej półabajta i 1 do AH, jeśli zawartość AL była większa od 9 lub wystąpiło przeniesienie AF.

aaa

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

26

Wpływa na flagi: OSZAPC

## Instrukcja AAS

aas

Korekta nieupakowanego kodu BCD (w AL) po odejmowaniu. Polega na odjęciu 6 od młodszej półabajta i 1 od AH, jeśli zawartość AL była większa od 9 lub wystąpiło przeniesienie AF.

aas

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

27

Wpływa na flagi: OSZAPC

## Instrukcja AAM

aam

Korekta nieupakowanego kodu BCD (w AX) po mnożeniu. Polega na jednoczesnym wykonaniu:  
 $AH=AL \text{ div } 10$   
 $AL:=AL \text{ Mod } 10$ .

aam

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

28

Wpływa na flagi: OSZAPC

## Instrukcja AAD

aad

Korekta nieupakowanego kodu BCD (w AX) przed dzieleniem. Polega na jednoczesnym wykonaniu:

$AL:=AH * 10 + AL$ .  
 $AH=0$

aad

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

29

## Przykład – dodawanie liczb BCD

mov	al, o	
add	al, al	;CF=o
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!!!

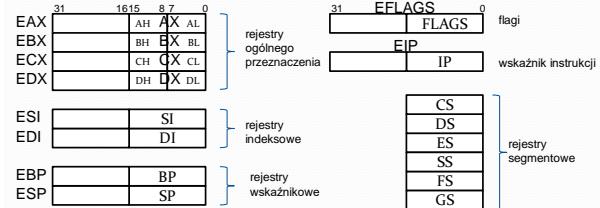
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

30

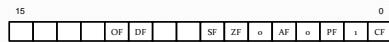
## Instrukcje logiczne, przesunięć i rotacji

### IA32- rejestrysty



(C) IISI d.KIK PCz 2019 Programowanie niskopoziomowe

### 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 wyrownania (adjust)	S
6	ZF	flaga zera (zero)	S
7	SF	flaga znaku (sign)	S
10	DF	flaga kierunku (direction)	C
11	OF	flaga przepchnięcia (overflow)	S

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

Programowanie niskopoziomowe

3

### Instrukcje logiczne

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

(C) IISI d.KIK PCz 2019 Programowanie niskopoziomowe

4

### Instrukcja AND

and cel, źródło

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

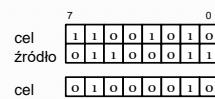
cel:=cel and źródło

and eax, zmienna

and edx, [ebx+esi\*4]

and rax, rdx

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



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

5

Wpływ na flagi: OSZAPC  
OSZxx0  
Wymagane BMI1

### Instrukcja ANDN

andn cel, źródło1, źródło2

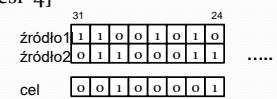
Wyznacza iloczyn logiczny zanegowanego źródła1 i źródła2 (bit po biciu), wynik umieszcza w miejscu przeznaczenia (cel). Cel i źródło1 są rejestrami 32|64 bitowymi.

cel:=(not źródło1) and źródło2

andn edx, eax, zmienna

andn eax, edx, [ebx+esi\*4]

andn r8, rax, rdx



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

6

Wpływ na flagi: OSZAPC  
OSZxP0

## 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.

cel	7	0
źródło	0 1 1 0 0 1 0 1	1
cel	1 1 1 0 1 0 1 1	1

7

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

Wpływ na flagi: OSZAPC  
OSZxP0

## 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.

cel	7	0
źródło	0 1 1 0 0 1 0 1	1
cel	1 0 1 0 1 0 0 1	1

8

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

Wpływ na flagi: -

## Instrukcja NOT

and 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

cel	7	0
cel	1 1 0 0 1 0 1 0	1

9

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

Wpływ na flagi: -

## 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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

10

Wpływ na flagi: OSZAPC  
(0x)SZxPC

## Instrukcja SAR

sar cel, ile

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

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

cel	7	0
cel	1 1 0 0 1 0 1 0	1
cel	1 1 1 0 0 1 0 1	0

CF

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

Wpływ na flagi: OSZAPC  
(0x)SZxPC

## Instrukcja SHR

shr cel, ile

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

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

cel	7	0
cel	1 1 0 0 1 0 1 0	1
cel	0 1 1 0 0 1 0 1	0

CF

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

12

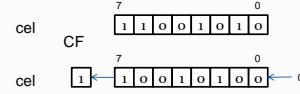


## Instrukcja SAL

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
```



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

13

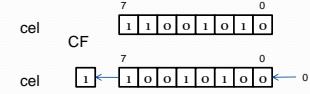


## Instrukcja SHL

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
```



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

14



## 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, rx, rcx
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

15



## 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, rx, rcx
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

16



## 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, rx, rcx
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

17

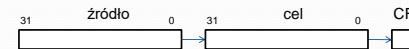


## Instrukcja SHRD

shrd cel, źródło, ile

Przesunięcie źródła:cel 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, rx, cl
```



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

18

Wpływ na flagi: OSZAPC  
(0x)---C

## Instrukcja SHLD

shld cel, źródło, ile

Przesunięcie źródła:celu w lewo o ile bitów. Ille=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
```



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

19

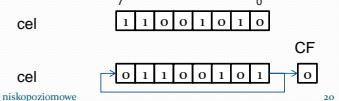
Wpływ na flagi: OSZAPC  
(0x)---C

## Instrukcja ROR

ror cel, ile

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

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



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

20

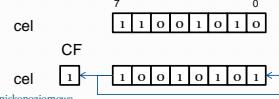
Wpływ na flagi: OSZAPC  
(0x)---C

## Instrukcja ROL

rol cel, ile

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

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



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

21

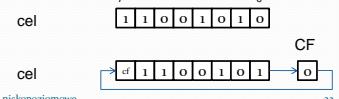
Wpływ na flagi: OSZAPC  
(0x)---C

## Instrukcja RCR

rcr cel, ile

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

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



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

22

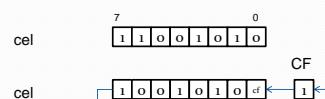
Wpływ na flagi: OSZAPC  
(0x)---C

## Instrukcja RCL

ror cel, ile

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

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



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

23

Wpływ na flagi: -----  
Wymaga BMI2

## Instrukcja RORX

ror cel, źródło, ile

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

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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

24

## Przykłady

```

and  eax, 0fffffefffh ;zeruje 12 bit eax
or   ax, 01000h       ;ustawia 12 bit ax
xor  rxax, 01000h     ;neguje 12 bit rxax
and  eax, eax         ;m. in. zeruje CF
xor  eax, eax         ;zeruje eax

```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

25

## Przykłady

```

and  eax, 07oh      ;maska
sar  eax, 4        ;eax - 3bitowa liczba

mov  ah, al          ;zamienia al na jego
and  ax, 0fo0fh      ;szesnastkową reprezentację
shr  ah, 4           ;ASCII w ah, al.
add  ax, 303oh

```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

26

## Przykłady

```

movsxrd  rax, zmienna    ;typu int - 32 bity
mov      rdx, rax         ;powiel
sal      rax, 2            ;x4
sar      rdx, 2            ;/4
adc      rax, rdx          ;x41/4
mov      zmienna, eax      ;zapisz

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

```

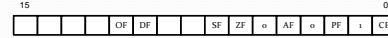
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

27

## Instrukcje warunkowe i skoku

### Rejestr flag



Rejestr flag w architekturze Intel x86		
bit	Skrot/wartość	Opis
0	CF	Flaga przeniesienia (carry)
2	PF	Flaga parzystosci (parity)
4	AF	Flaga wynownia (adjust)
6	ZF	Flaga zera (zero)
7	SF	Flaga znaku (sign)
10	DF	Flaga kierunku (direction)
11	OF	Flaga przepchnienia (overflow)

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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

2

## 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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

3

## 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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

4

Wpływ na flagi: -

## Instrukcja CMOVcc

CMOVcc cel, źródło

Jeśli jest spełniony warunek cc, przesyła źródło do miejsca przeznaczenia (rejestr 16, 32 lub 64 bitowy). Instrukcja wprowadzona w procesorach rodziny P6!

if cc then cel:=źródło

cmovz	eax, zmienna
cmovge	edx, [ebx+esi*4]
cmovna	rax, rdx

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

5

## Instrukcje CMOVcc

• CMOVE/CMOVZ	Przesiąj jeżeli equal/ zero
• CMOVNE/CMOVNZ	Przesiąj jeżeli not equal/ not zero
• CMOVA/CMOVNBE	Przesiąj jeżeli above/ not below or equal
• CMOVAE/CMOVNB	Przesiąj jeżeli above or equal/ not below
• CMOVB/CMOVNAE	Przesiąj jeżeli below/ not above or equal
• CMOVBE/CMOVNB	Przesiąj jeżeli below or equal/ not above
• CMOVG/CMOVNLE	Przesiąj jeżeli greater/ not less or equal
• CMOVG/E/CMOVNL	Przesiąj jeżeli greater or equal/ not less
• CMOVL/CMOVNGE	Przesiąj jeżeli less/ not greater or equal
• CMOVLE/CMOVNG	Przesiąj jeżeli less or equal/ not greater
• CMOCV	Przesiąj jeżeli carry
• CMOVNC	Przesiąj jeżeli not carry
• CMOVO	Przesiąj jeżeli overflow
• CMOVNO	Przesiąj jeżeli not overflow
• CMOVS	Przesiąj jeżeli sign (negative)
• CMOVNS	Przesiąj jeżeli not sign (non-negative)
• CMOV/P/CMOVPE	Przesiąj jeżeli parity/ parity even
• CMOVNP/CMOVPO	Przesiąj jeżeli not parity/ parity odd

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

6

## Przykład

```
MyMax64 proc
movsx  rax, ecx
movsx  rdx, edx
cmp    rax, rdx
cmovl rax, rdx
ret
MyMax64 endp
```

```
function
TForm1.MyMax(x,y:integer):integer;
asm
  mov eax, x
  cmp eax, y
  jnc @@exit
  mov eax, y
@@exit:
end;
```

```
function
TForm1.MyMax2(x,y:integer):integer;
asm
  mov eax, x
  cmp eax, y
  cmovc eax, y ;cmovb eax,y
end;
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

7

## 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/not below
• JAE/JNB	Skocz jeśli below/not above or equal/not below
• JB/JNAE	Skocz jeśli below or equal/not above
• 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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

8

## Instrukcja JZ

jz      przesunięcie

Przeskakuje do podanej etykiety (adres jest wzgledny 16/32bitowy).

EIP := EIP + przesunięcie

jz      dalej

...

dalej: ...

Wpływ na flagi: -

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

9

## Przykład

```
procedure MySort(t:array of integer;n:integer);
asm
  push edi          ;zabezpiecz rejestry
  push esi
  mov esi, n        ;jakość
  dec esi           ;esi ostatni element
  mov edx, t        ;adres tablicy
  mov edi, esi
  dec edi           ;poprzedzający element
  mov eax, [edx+esi*4] ;ostatni do eax
  @w:               ;cmp eax, [edx+edi*4] ;czy >=
  cmp eax, [edx+edi*4]
  jae @@a            ;jاء @ا
  xchg eax, [edx+edi*4] ;zamień elementy
  mov [edx+esi*4], eax
  @@a:              ;jns @@a
  dec edi           ;poprzedzający element
  jns @w             ;ostatni do eax
  dec esi           ;dec esi
  jnz @p             ;jne @p
  pop esi           ;jne @p
  pop edi           ;pop edi
end;
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

10

## Instrukcje sterujące przebiegiem programu +

- JMP               Skok bezwarunkowy
- JCXZ/JECXZ/JRCX Skok jeśli zero w rejestrze CX/ECX/RCX
- LOOP              Pętla z licznikiem CX/ECX/RCX
- LOOPZ/LOOPE      Pętla z licznikiem CX/ECX/RCX i zero/equal
- LOOPNZ/LOOPNE    Pętla z licznikiem CX/ECX/RCX i not zero/not equal
- CALL             Wyzwolenie 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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

11

## Instrukcja JMP

jmp      adres

Przeskakuje do podanej etykiety (adres jest wzgledny 8/16/32bitowy lub bezwzględny).

EIP := EIP + przesunięcie(adres)

CS:=segment(adres); EIP:=EIP+przesunięcie(adres)

jmp      dalej

jmp      eax

jmp      [esi]

jmp      lib1:dalej

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

12

Wpływ na flagi: -

## Instrukcja JCXZ/JECXZ/JRCXZ

JCXZ/JECXZ/JRCXZ      przesunięcie

Skok jeśli zero w rejestrze CX/ECX/RCX do podanej etykiety (adres jest względny 16/32/64 bitowy).

EIP := EIP + przesunięcie

petla: ...

  jecxz   dalej

  ...

  jmp   petla

dalej: ...

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

13

Wpływ na flagi: -

## Instrukcja LOOP

loop   przesunięcie

Pętla z licznikiem CX/ECX/RCX. Zmniejsza CX/ECX/RCX o 1 i jeśli nie uzyskano zera przeskakuje do podanej etykiety (adres jest względny 8 bitowy).

EIP := EIP + przesunięcie(adres)

petla: ...

  ...

  loop   petla

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

14

Wpływ na flagi: -

## Przykład

Silnia

```
function silnia(n:integer):integer;
asm
  mov  ecx, eax
  dec  ecx
  @p: imul eax, ecx
  dec  ecx
  jnz @p
end;
function silnia(n:integer):integer;
asm
  mov  ecx, eax
  @p: dec  ecx
  jecxz @e
  imul  eax, ecx
  jmp  @p
  @e:
end;
function silnia(n:integer):integer;
asm
  mov  ecx, eax
  dec  ecx
  @p: imul eax, ecx
  loop  @p
end;
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

15

Wpływ na flagi: -

## Instrukcja LOOPZ/LOOPE

loopz/loope   przesunięcie

Pętla z licznikiem CX/ECX/RCX i zero/equal. Zmniejsza CX/ECX/RCX o 1 i jeśli nie uzyskano zera w CX/ECX/RCX i flaga ZF=1 przeskakuje do podanej etykiety (adres jest względny 8 bitowy).

EIP := EIP + przesunięcie(adres)

petla: ...

  ...

  loopz   petla

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

16

Wpływ na flagi: -

## Instrukcja LOOPNZ/LOOPNE

loopnz/loopne   przesunięcie

Pętla z licznikiem CX/ECX/RCX i nie zero/equal. Zmniejsza CX/ECX/RCX o 1 i jeśli nie uzyskano zera i flaga ZF=0 przeskakuje do podanej etykiety (adres jest względny 8 bitowy).

EIP := EIP + przesunięcie(adres)

petla: ...

  ...

  loopnz   petla

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

17

Wpływ na flagi: -

## Instrukcja CALL

call   adres

Instrukcja call wywołuje podprogram, wysyła na stos adres powrotu EIP|RIP lub CS:EIP|RIP. Parametr adres wpisuje do EIP|RIP lub CS:EIP|RIP.

push e(r)ip; (push cs);

E(R)IP:= przesunięcie adres; (CS:=segment adres)

call   procedura

call   eax

call   [esi]

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

18

Wpływ na flagi: -

## Instrukcja RET

ret (ile)

Instrukcja ret wraca z podprogramu, pobiera ze stosu adres powrotu do EIP|RIP lub CS:EIP|RIP. Jeśli posiada parametr ile, to dodatkowo usuwa ze stosu ile bajtów (niepotrzebne już parametry aktualne wywołania).

pop e(r)ip; (pop cs); (e(r)sp:=e(r)sp+ile)

ret

ret 6

function silnia3(n:integer):integer;

```
asm
    and eax,eax
    cmovz eax,one
    jz @e
    push eax
    dec eax
    call silnia3
    pop edx
    imul eax,edx
    @e:
end;
```

## Przykład

Silnia

```
one db 1,0,0,0,0,0,0
silnia4 proc
    cmp rcx,1
    cmovbe rax,one
    jbe @e
    push rcx
    dec rcx
    call silnia4
    pop rcx
    imul rax,rcx
    @e:
silnia4 endp;
```

Wpływ na flagi: -

## Instrukcja INT

int nr

Wywołuje przerwanie programowe o numerze nr (0-255). Numery z zakresu 0-31 są zarezerwowane. Instrukcja int działa podobnie do instrukcji call, jednak dodatkowo wysyła na stos flagi i wchodząc do podprogramu część z nich zeruje.

int 21h

## Przerwania i wyjątki zarezerwowane

Wektor Nr	Mnemonik	Opis	Zródło
0	#DE	Błąd dzielenia	Instrukcje DIV i IDIV.
1	#DB	Debugger	Dowolne odwołanie do kodu i danych.
2	#NM	Nieudane NMI	Instrukcja niezakwalifikowane Non-maskable external.
3	#BP	Breakpoint	Instrukcja INT 3.
4	#OF	Overflow	Instrukcja INTO.
5	#BR	BOUND przekroczony	Instrukcja BOUND.
6	#UD	Błędny kod	Instrukcja UD2 lub zarezerwowany kod 1.
7	#NM	Urządzenie nie dostępnego (Brak koprocesora)	Instrukcja zmiennoprzecinkowa lub WAIT/FWAIT.
8	#DF	Segment niepoprawny	Dowolna instrukcja generująca wyjątek, NMII lub INTR.
9	#MF	CoProcessor Segment Overrun (reserved)	Instrukcje zmiennoprzecinkowe.
10	#TS	Błędny TSS	Przeliczanie zadania lub dostępu do TSS.
11	#NP	Segment nieobecny	Dołączanie nowych segmentowych lub dostęp do segmentów systemowych.
12	#SS	Segment stosu uszkodzony	Operacje na stosie i ładowanie rejestru SS.
13	#GP	Ogólna ochrona	Dowolne odwołanie do pamięci i inne zabezpieczenia.
14	#PF	Błąd strony	Dowolne odwołanie do pamięci.
15	#DE	Zarezerwowane	
16	#MF	Błąd zmiennoprzecinkowy (biał matematyczny)	Instrukcje zmiennoprzecinkowe lub WAIT/FWAIT.
17	#AC	Kontrola wywołania	Dowolne odwołanie do danych w pamięci.
18	#MC	Kontrola zmiany	Kod błędu (o ile występuje) i zapisywana od modelu.
19	#XM	Wyjątek SIMD	Instrukcje zmiennoprzecinkowe SIMDs.
20-31		Zarezerwowane	
32-255		Przewrana maskowana	Przewrana zewnętrzne INTR lub instrukcje INT n.

1. Instrukcja UD2 została wprowadzona w procesorze Pentium Pro. 2. Procesory IA-32 po procesorze Intel®386 nie generują tego wyjątku.  
 3. Wyjątek wprowadzony w procesorze Intel486. 4. Wyjątek wprowadzony w procesorze Pentium i poprawiony w procesorach rodziny P6.  
 5. Wyjątek wprowadzony w procesorze Pentium III.

Wpływ na flagi: -

## Instrukcja INTO

into

Wywołuje przerwanie programowe (4) w przypadku ustawienia flagi OF (nadmiaru).

into

## Instrukcja IRET/IRETD/IRETQ

iret/iretd/iretq

Instrukcja iret/iretd/iretq wraca z podprogramu obsługi przerwania, pobiera ze stosu adres powrotu do CS:EIP|RIP oraz flagi zachowane przy wywołaniu przerwania.

pop e(r)ip; pop cs; pop (e(r)flags)

iret



## Instrukcja BOUND

bound idx, gr

Sprawdza, czy indeks tablicy (wartość 16/32 bitowa ze znakiem) zawarty w rejestrze idx nie przekracza jej granic określonych przez strukturę w pamięci gr złożoną z granicy dolnej i górnej. W przypadku przekroczenia granicy generowany jest wyjątek przekroczenia granicy tablicy.

bound eax, granice1

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

25



## Instrukcja ENTER

enter storage, level

Tworzy ramę stosu w podprogramie z uwzględnieniem poziomu zagnieżdżenia podprogramów lokalnych (level) i rozmiaru w bajtach zmennych lokalnych (storage). Na stosie umieszcza wskaźnik ram stosu: podprogramu wywołującego, wszystkich poziomów nadrzednych i własny.

```
PUSH EBP;
FRAME_PTR ← ESP;
IF LEVEL > 0 THEN
    DO (LEVEL - i) times
        EBP ← EBP - 4;
        PUSH Pointer(EBP); (* doubleword wskazywane przez EBP *)
    OD;
    PUSH FRAME_PTR;
FI;
EBP ← FRAME_PTR;
ESP ← ESP - STORAGE;
```

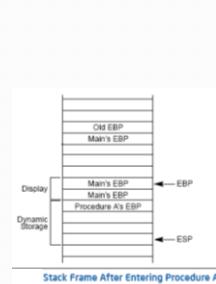
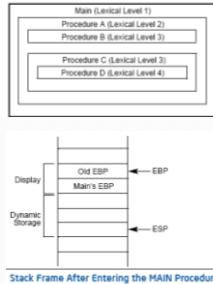
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

26



## Instrukcja ENTER



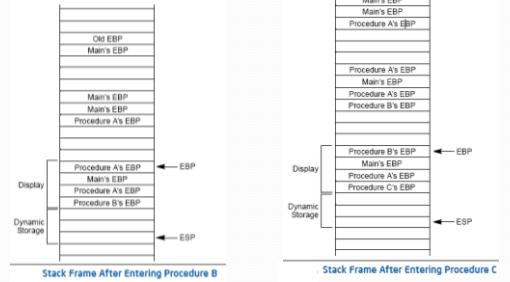
(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

27



## Instrukcja ENTER



(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

28



## Instrukcja LEAVE

leave

Usuwa ramę stosu (utworzoną instrukcją ENTER) przed wyjściem z podprogramu. Przypisuje do wskaźnika stosu rejestr bazowy, następnie zdejmuję rej. basowy ze stosu.

esp:=ebp; pop ebp

leave

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

29



## Przykład

```
procedure Sort(var A: array of Integer);
procedure QuickSort(var A: array of Integer; iLo, iHi: Integer);
var
  Lo, Hi, Mid, T: Integer;
begin
  Lo := iLo;
  Hi := iHi;
  Mid := A[(Lo + Hi) div 2];
  repeat
    while A[Lo] < Mid do Inc(Lo);
    while A[Hi] > Mid do Dec(Hi);
    if Lo <= Hi then
      begin
        T := A[Lo];
        A[Lo] := A[Hi];
        A[Hi] := T;
        Inc(Lo);
        Dec(Hi);
      end;
    until Lo > Hi;
    if Hi > iLo then QuickSort(A, iLo, Hi);
    if Lo < iHi then QuickSort(A, Lo, iHi);
  end;
begin
  QuickSort(A, Low(A), High(A));
end;
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

30

```

procedure mysqrt(var A:Array of integer;Lo,IHi:integer);
asm
    push edi
    push esi
    push ilo
    push ihi
    call @_s
    jmp @_e

@:_s: enter 0,0           //sortowanie eax-A; [ebp+8]=ilo; [ebp+12]=ihi
    mov esi,[ebp+8]        //ilo<hi<ihi
    mov edi,[ebp+12]        //ilo<lo<ihi
    mov exc,esi             //exc=Mid >= A[lo + hi] div 2
    add exc,edi
    sar exc,1
    mov exc,[eax+exc*4]     //mid
    @_r:                   //repeat
    cmp [eax+edi*4],ecx   //while A[lo] < Mid do Inc(Lo)
    jge @_l
    inc edi
    jmp @_r
@:_l: cmp [eax+edi*4],ecx //while A[hi] > Mid do Dec(Hi)
    jle @_s
    dec edi
    jmp @_r

@:_e: cmp edi,esi         //if Lo < Hi then
    jne @_m
    mov [eax+ecx*4],esi    //A[lo]<->A[hi]
    mov ecx,[eax+edi*4]    //A[lo]=A[hi]
    leave
    mov [eax+edi*4],ecx
    mov [eax+edi*4],edx
    inc edi
    dec esi
    @_g: cmp edi,esi        //until Lo > Hi;
    jle @_r
    cmp esi,[ebp+12]        //if Hi > Lo then QuickSort(A, lo, Hi)
    jng @_4
    push edi
    push [ebp+12]
    push esi
    call @_s
    pop edi
    @_h: cmp edi,[ebp+8]      //if Lo < Hi then QuickSort(A, Lo, iHi)
    jge @_g
    jmp @_h
    @_m: leave
    ret 8
@:_p: pop esi
    pop edi
    end;

```

(C) IISI d.RIK F CZ 2019 Programowanie niskopoziomowe

(c) HSGiK i CZ 2019 Programowanie niskopoziomowe

三

## Množenie BCD

$$\begin{array}{r}
 222 \\
 5678 \\
 * \quad 3 \\
 \hline
 17034 \\
 +1122 \\
 \hline
 17034
 \end{array}$$

(C) HSRd.RIK FCZ 2019

Programowanie niskopoziomowe

3

## Množenie BCD

```

void n_BCD(char*a,char*b,int n,int m,char*w) {
    _asm{
        pushad
        pushf
        mov    ecx,m      //ilość cyfr mnoźnika - m
        mov    edi,b      //adres mnoźnika
        add    edi,ecx
        mov    edx,w      //adres wyniku
        add    edx,ecx
        push   ecx         //zabezpiecz m ilość
        //pozostałych cyfr mnoźnika
        dec   edx         //m do o cyfra wyniku
        dec   edi         //m do o cyfra mnoźnika
        mov    bl,[edi]    //do bl
        mov    ecx,p      //ilość cyfr mnoźnej - m
        mov    esi,p      //adres mnożej
        one_n
        mov    al,[esi]    //cyfra mnożej - od najstarszej
        mul   bl          //razy cyfra mnoźnika
        aam
        push   ax         //ab - przeniesienie z
        //mnożenia, al - wynik na stos
        inc    esi
        Loop
        iloczyn
    }
    mov    cx,n
    pop    ax          ///najmłodsze szt osu
    add    al,[edx+ecx],l //i dodajemy cyfrę do wyniku
    aaa
    mov    [edx+ecx],al
    dec    ecx
    jz    one_n         //jeśli tylko jednocyfrowa mnożna
    SumaBCD:
    mov    bl,ah        //przeniesienie do bl
    pop    ax          //na następna cyfra i przeniesienie ze
    stosu
    add    al,[edx+ecx],l //dodajemy kolejną cyfrę wyniku
    aaa
    add    al,bl        //plus poprzednie przeniesienie
    mov    [edx+ecx],al //i do wyniku
    loop
    SumaBCD
    one_n
    mov    [edx],ah     //ostatnie przeniesienie do wyniku
    pop    ecx         //ilość pozostałych cyfr mnoźnika
    loop
    po_m
    popf
    popad
    }
}

```

34

# Operacje na znacznikach, bitach i bajtach

## Rejestr flag

bit	Skrot/wartosc	Opis	typ
15			
14			
13	OF	Flaga przeniesienia (carry)	S
12	DF	Flaga parzystosci (parity)	S
11			
10	SF	Flaga wynowanego (adjust)	S
9	ZF	Flaga zera (zero)	S
8	AF	Flaga znaku (sign)	S
7	PF	Flaga kierunku (direction)	C
6			
5			
4			
3			
2			
1			
0	CF	Flaga przepelenia (overflow)	S

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

(C) IISI d.KIK PCz 2019      Programowanie niskopoziomowe

## 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/          Wysłanie flag na stos  
PUSHFQ
- POPF/POPFD/POPFQ      Pobranie flag ze stosu
- STI                        Ustawienie IF – flagi przerwań
- CLI                        Zerowanie IF

(C) IISI d.KIK PCz 2019      Programowanie niskopoziomowe

## Instrukcja STC

Wplywa na flagi: C

stc

Ustawienie flagi CF.  
CF:=1

stc

(C) IISI d.KIK PCz 2019      Programowanie niskopoziomowe

Wplywa na flagi: C

## Instrukcja CLC

clc

Zerowanie flagi CF.

CF:=0

clc

(C) IISI d.KIK PCz 2019      Programowanie niskopoziomowe

Wplywa na flagi: C

## Instrukcja CMC

cmc

Zanegowanie flagi CF.

CF:=not CF

cmc

(C) IISI d.KIK PCz 2019      Programowanie niskopoziomowe



## Instrukcja STD

std

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

DF:=1

std

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

7



## Instrukcja CLD

cld

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

DF:=0

cld

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

8



## Instrukcja LAHF

lahf

Przesłanie flag do rejestru AH

AH:=lo(FLAGS)

lahf

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

9



## Instrukcja SAHF

sahf

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

lo(FLAGS) :=AH

sahf

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

10



## Instrukcja PUSHF/PUSHFD/PUSHFQ

pushf/pushfd/pushfq

Przesyła zawartość Flag/Eflag/Rflag na stos.

pushf

pushfd

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

11



## Instrukcja POPF/POPFD/POPFQ

popf/popfd/popfq

Pobiera zawartość Flag/EFlag ze stosu.

popf

popfd

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

12



Wpływ na flagi: I

## Instrukcja STI

**sti**

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

IF:=1

**sti**

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

13



Wpływ na flagi: I

## Instrukcja CLI

**cli**

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

IF:=0

**cli**

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

14



## 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
- **BEXT** Wycina ciąg bitów
- **BLSI** Kopiuje najmłodszy ustawiony bit
- **BLSR** Zeruje najmłodszy ustawiony bit
- **BLSMSK** Tworzy maskę do bitu=0
- **BZHI** Zeruje starsze bity

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

15

Wpływ 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**

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

16

Wpływ 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**

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

17

Wpływ 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**

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

18

Wpływ na flagi: OSZAPC  
xxxxx

## 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,esi

btc rcx, 37

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

19

Wpływ na flagi: OSZAPC  
0SZxP0

## Instrukcja TEST

test cel, źródło

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

cel and źródło

test eax,zmienna

test edx,[ebx+esi\*4]

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

20

Wpływ 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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

21

Wpływ 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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

22

Wpływ 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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

23

Wpływ na flagi: OSZAPC  
xxZxxC  
Wymaga BM1

## Instrukcja TZCNT

tzcnt cel, źródło

Zlicza od najmłodszego zerowego bitu 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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

24

Wpływ 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=o, wówczas ZF=1.

```
cel := źródło[start+ile-1:start]
bextr eax, zmienna, edx
bextr edx, esi, eax
bextr rcx, rdx, rax
```

(C) IISI d.KIK PCz 2019      Programowanie niskopoziomowe      25

Wpływ na flagi: OSZAPC  
0xZxxC  
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=o, wówczas CF=o, a cel=o.

```
cel := (-źródło) and źródło
blsi eax, zmienna
blsi edx, esi
blsi rcx, rdx
```

(C) IISI d.KIK PCz 2019      Programowanie niskopoziomowe      26

Wpływ na flagi: OSZAPC  
0xZxxC  
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=o). Jeśli źródło=o, wówczas CF=1, a cel=o.

```
cel := (źródło-i) and źródło
blsr eax, zmienna
blsr edx, esi
blsr rcx, rdx
```

(C) IISI d.KIK PCz 2019      Programowanie niskopoziomowe      27

Wpływ na flagi: OSZAPC  
0x0xxC  
Wymaga BMI1

## Instrukcja BLMSK

**blmsk cel, źródło**

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

```
cel := (źródło-i) xor źródło
blmsk eax, zmienna
blmsk edx, esi
blmsk rcx, rdx
```

(C) IISI d.KIK PCz 2019      Programowanie niskopoziomowe      28

Wpływ na flagi: OSZAPC  
0xSxxC  
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=o). 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
```

(C) IISI d.KIK PCz 2019      Programowanie niskopoziomowe      29

Wpływ 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]
```

(C) IISI d.KIK PCz 2019      Programowanie niskopoziomowe      30

## 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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

31

## Przykład – int na bin(string)

```
procedure sab(var s:string;i:integer);
asm
  push ebx
  bsr edx,ecx //w edx nr najstarszej 1
  jnz @i
  mov word ptr [eax],$3001
  jmp @e
  @i: 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;
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

32

Wpływ na flagi: -

## Instrukcja RDPID

rdpid cel

Czyta 32-bitowy identyfikator procesora do rejestru celu.

cel=PID

rdpid eax

rdpid rdx

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

33

## Instrukcja RDTSC

rdtsc

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

EDX:EAX:=licznik

rdtsc

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

34

Wpływ na flagi: -

## 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;
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

35

## Przykład – funkcja random

```
function MyRandom(a:integer):integer;overload;
asm
  push ebx
  xor ebx,ebx
  imul edx,[ebx+MySeed].$08e088405
  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
  rot a,3
  imul edx,MySeed,$08e088405
  rot ah,3
  inc edx
  rot eax,1
  mov MySeed,edx
  bswap eax
  xor eax,edx
  pop edx
  mul edx
  mov eax,edx
end;
```

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

36

## Operacje na łańcuchach

### Operacje na łańcuchach

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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

2

### Instrukcja MOVS/MOVSB

Wpływ na flagi: -

```
movs byte ptr [(r|e)di],[(r|e)si]
movsb
```

Przesyła bajt z pamięci ds:(r|e)si do pamięci es:(r|e)di. Rejestry (r|e)di/(r|e)si są zwiększone/zmniejszone o 1 w zależności od flagi DF (0/1).

$$\begin{aligned} [(r|e)s:edi] &= [ds:(r|e)si] \\ (r|e)di &:= (r|e)di \pm 1 \\ (r|e)si &:= (r|e)si \pm 1 \end{aligned}$$

movsb

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

3

### Instrukcja MOVS/MOVSD

Wpływ na flagi: -

```
movs dword ptr [(r|e)di],[(r|e)si]
movsd
```

Przesyła podwójne słowo z pamięci ds:esi do pamięci es:edi. Rejestry (r|e)di/(r|e)si są zwiększone/zmniejszone o 4 w zależności od flagi DF (0/1).

$$\begin{aligned} [es:(r|e)di] &= [ds:(r|e)si] \\ (r|e)di &:= (r|e)di \pm 4 \\ (r|e)si &:= (r|e)si \pm 4 \end{aligned}$$

movsd

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

5

### Instrukcja MOVS/MOVSW

Wpływ na flagi: -

```
movs word ptr [(r|e)di],[(r|e)si]
movsw
```

Przesyła słowo z pamięci ds:(r|e)si do pamięci es:(r|e)di. Rejestry (r|e)di/(r|e)si są zwiększone/zmniejszone o 2 w zależności od flagi DF (0/1).

$$\begin{aligned} [es:(r|e)di] &= [ds:(r|e)si] \\ (r|e)di &:= (r|e)di \pm 2 \\ (r|e)si &:= (r|e)si \pm 2 \end{aligned}$$

movsw

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

4

### Instrukcja MOVS/MOVSQ

Wpływ na flagi: -

```
movs qword ptr [(r|e)di],[(r|e)si]
movsq
```

Przesyła poczwórne słowo z pamięci ds:(r|e)si do pamięci es:(r|e)di. Rejestry (r|e)di/(r|e)si są zwiększone/zmniejszone o 8 w zależności od flagi DF (0/1).

$$\begin{aligned} [es:(r|e)di] &= [ds:(r|e)si] \\ (r|e)di &:= (r|e)di \pm 8 \\ (r|e)si &:= (r|e)si \pm 8 \end{aligned}$$

movsq

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

6

Wpływa na flagi: OSZAPC

## Instrukcja CMPS/CMPSB

```
cmps byte ptr [(r|e)si],[(r|e)di]
cmpsb
```

Porównuje bajt z pamięci ds:(r|e)si i z pamięci es:(r|e)di.  
Rejestry (r|e)di/(r|e)si są zwiększane/zmniejszane o 1 w  
zależności od flagi DF (0/1).

$$\begin{aligned} & [ds:(r|e)si]-[es:(r|e)di] \\ & (r|e)di:=(r|e)di \pm 1 \\ & (r|e)si:=(r|e)si \pm 1 \end{aligned}$$

cmpsb

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

7

Wpływa na flagi: OSZAPC

## Instrukcja CMPS/CMPSW

```
cmps word ptr [(r|e)si],[(r|e)di]
cmpsw
```

Porównuje słowo z pamięci ds:(r|e)si i z pamięci es:(r|e)di.  
Rejestry (r|e)di/(r|e)si są zwiększane/zmniejszane o 2 w  
zależności od flagi DF (0/1).

$$\begin{aligned} & [ds:(r|e)si]-[es:(r|e)di] \\ & (r|e)di:=(r|e)di \pm 2 \\ & (r|e)si:=(r|e)si \pm 2 \end{aligned}$$

cmpsw

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

8

Wpływa na flagi: OSZAPC

## Instrukcja CMPS/CMPSD

```
cmps dword ptr [(r|e)si],[(r|e)di]
cmpsd
```

Porównuje podwójne słowo z pamięci ds:(r|e)si i z pamięci  
es:(r|e)di. Rejestry (r|e)di/(r|e)si są zwiększane/zmniejszane o  
4 w zależności od flagi DF (0/1).

$$\begin{aligned} & [ds:(r|e)si]-[es:(r|e)di] \\ & (r|e)di:=(r|e)di \pm 4 \\ & (r|e)si:=(r|e)si \pm 4 \end{aligned}$$

cmpsd

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

9

Wpływa na flagi: OSZAPC

## Instrukcja CMPS/CMPSQ

```
cmps qword ptr [(r|e)si],[(r|e)di]
cmpsq
```

Porównuje poczwórne słowo z pamięci ds:(r|e)si i z pamięci  
es:(r|e)di. Rejestry (r|e)di/(r|e)si są zwiększane/zmniejszane o  
8 w zależności od flagi DF (0/1).

$$\begin{aligned} & [ds:(r|e)si]-[es:(r|e)di] \\ & (r|e)di:=(r|e)di \pm 8 \\ & (r|e)si:=(r|e)si \pm 8 \end{aligned}$$

cmpsq

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

10

Wpływa na flagi: OSZAPC

## Instrukcja SCAS/SCASB

```
scas byte ptr [(r|e)di]
scasb
```

Porównuje bajt akumulatora AL i pamięci es:(r|e)di. Rejestr  
(r|e)di jest zwiększany/zmniejszany o 1 w zależności od flagi  
DF (0/1).

$$\begin{aligned} & AL-[es:(r|e)di] \\ & (r|e)di:=(r|e)di \pm 1 \end{aligned}$$

scasb

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

11

Wpływa na flagi: OSZAPC

## Instrukcja SCAS/SCASW

```
scas word ptr [(r|e)di]
scasw
```

Porównuje słowo akumulatora AX i pamięci es:(r|e)di. Rejestr  
(r|e)di jest zwiększany/zmniejszany o 2 w zależności od flagi  
DF (0/1).

$$\begin{aligned} & AX-[es:(r|e)di] \\ & (r|e)di:=(r|e)di \pm 2 \end{aligned}$$

scasw

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

12

Wpływ na flagi: OSZAPC

## Instrukcja SCAS/SCASD

scas dword ptr [(r|e)di]

scasd

Porównuje podwójne słowo akumulatora EAX i pamięci es:(r|e)di. Rejestr (r|e)di jest zwiększany/zmniejszany o 4 w zależności od flagi DF (0/1).

EAX-[es:(r|e)di]

(r|e)di:=(r|e)di ±4

scasd

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

13

Wpływ na flagi: OSZAPC

## Instrukcja SCAS/SCASQ

scas qword ptr [(r|e)di]

scasq

Porównuje poczwórne słowo akumulatora RAX i pamięci es:(r|e)di. Rejestr (r|e)di jest zwiększany/zmniejszany o 8 w zależności od flagi DF (0/1).

RAX-[es:(r|e)di]

(r|e)di:=(r|e)di ±8

scasq

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

14

Wpływ na flagi: -

## Instrukcja LODS/LODSB

lodsb byte ptr [(r|e)si]

lodsb

Czyta bajt do akumulatora AL z pamięci ds:(r|e)si. Rejestr (r|e)si jest zwiększany/zmniejszany o 1 w zależności od flagi DF (0/1).

AL=[ds:(r|e)si]

(r|e)si:=(r|e)si ±1

lodsb

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

15

Wpływ na flagi: -

## Instrukcja LODS/LODSW

lodsw word ptr [(r|e)si]

lodsw

Czyta słowo do akumulatora AX z pamięci ds:(r|e)si. Rejestr (r|e)si jest zwiększany/zmniejszany o 2 w zależności od flagi DF (0/1).

AX=[ds:(r|e)si]

(r|e)si:=(r|e)si ±2

lodsw

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

16

Wpływ na flagi: -

## Instrukcja LODS/LOSDS

lodsd dword ptr [(r|e)si]

lodsd

Czyta podwójne słowo do akumulatora EAX z pamięci ds:(r|e)si. Rejestr (r|e)si jest zwiększany/zmniejszany o 4 w zależności od flagi DF (0/1).

EAX=[ds:(r|e)si]

(r|e)si:=(r|e)si ±4

lodsd

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

17

Wpływ na flagi: -

## Instrukcja LODS/LODSQ

lodsq qword ptr [(r|e)si]

lodsq

Czyta poczwórne słowo do akumulatora RAX z pamięci ds:(r|e)si. Rejestr (r|e)si jest zwiększany/zmniejszany o 8 w zależności od flagi DF (0/1).

RAX=[ds:(r|e)si]

(r|e)si:=(r|e)si ±8

lodsq

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

18

Wpływa na flagi: -

## Instrukcja STOS/STOSB

stos byte ptr [(r|e)di]

stosb

Zapisuje bajt z akumulatora AL do pamięci es:(r|e)di. Rejestr (r|e)di jest zwiększany/zmniejszany o 1 w zależności od flagi DF (o/1).

[es:(r|e)di]=AL

(r|e)di:=(r|e)di ±1

stosb

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

19

Wpływa na flagi: -

## Instrukcja STOS/STOSW

stos word ptr [(r|e)di]

stosw

Zapisuje słowo z akumulatora AX do pamięci es:(r|e)di. Rejestr (r|e)di jest zwiększany/zmniejszany o 2 w zależności od flagi DF (o/1).

[es:(r|e)di]=AX

(r|e)di:=(r|e)di ±2

stosw

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

20

Wpływa na flagi: -

## Instrukcja STOS/STOSD

stos dword ptr [(r|e)di]

stosd

Zapisuje podwójne słowo z akumulatora EAX do pamięci es:(r|e)di. Rejestr (r|e)di jest zwiększany/zmniejszany o 4 w zależności od flagi DF (o/1).

[es:(r|e)di]=EAX

(r|e)di:=(r|e)di ±4

stosd

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

21

Wpływa na flagi: -

## Instrukcja STOS/STOSQ

stos qword ptr [(r|e)di]

stosq

Zapisuje poczwórne słowo z akumulatora RAX do pamięci es:(r|e)di. Rejestr (r|e)di jest zwiększany/zmniejszany o 8 w zależności od flagi DF (o/1).

[es:(r|e)di]=RAX

(r|e)di:=(r|e)di ±8

stosq

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

22

Wpływa na flagi: -

## Prefiks REP

**REPNZ/REPNE****REPZ/REPE**

Powoduje powtórzenie (R|E)CX razy następującej po nim instrukcji łańcuchowej, jeśli spełniony jest warunek (repnz powtarza dopóty ZF=0, jeśli ZF=1 powtarzanie jest przerywane itd.). Jeżeli (R|E)CX=0, to instrukcja nie zostanie wykonana.

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

23

Wpływa na flagi: -

## Prefiks REP

**REPNZ/REPNE****REPZ/REPE**

rep movsb

rep lodsd

rep stosq

repww cmpsw

repww scasb

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

24

## Przykład

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

Kopiuje zawartość buforai do buforaz.

```
mov rax,o
mov rcx,100
mov rdi,bufor
repz ds:stosq
```

Zeruje zawartość bufor (800B).

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

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

```
mov al,o
mov rcx,100
mov rdi,bufor
repz ds:scash
```

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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

25

## Operacje na rejestrach segmentowych

- LDS Załadowanie pełnego wskaźnika z użyciem DS
- LES Załadowanie pełnego wskaźnika z użyciem ES
- LFS Załadowanie pełnego wskaźnika z użyciem FS
- LGS Załadowanie pełnego wskaźnika z użyciem GS
- LSS Załadowanie pełnego wskaźnika z użyciem SS

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

26

## Instrukcja LDS

lds cel,źródło

Wczytanie pełnego adresu źródła do pary rejestrów  
ds:cel(32).

ds:cel:=wskaźnik do źródła

lds esi,tablica

Wpływ na flagi: -

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

27

## Instrukcja LES

les cel,źródło

Wczytanie pełnego adresu źródła do pary rejestrów  
es:cel(32).

es:cel:=wskaźnik do źródła

les edi,tablica2

Wpływ na flagi: -

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

28

## Instrukcja LFS

lfs cel,źródło

Wczytanie pełnego adresu źródła do pary rejestrów  
fs:cel.

fs:cel:=wskaźnik do źródła

lfs eax,tablica

Wpływ na flagi: -

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

29

## Instrukcja LGS

lgs cel,źródło

Wczytanie pełnego adresu źródła do pary rejestrów  
gs:cel.

gs:cel:=wskaźnik do źródła

lgs eax,tablica

Wpływ na flagi: -

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

30

Wpływa na flagi: -

## Instrukcja LSS

lss cel,źródło

Wczytanie pełnego adresu źródła do pary rejestrów ss:cel.

ss:cel:=wskaźnik do źródła

lss esp,nowy\_stos

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

31

## Inne operacje

- LOCK Powoduje niepodzielne wykonanie następnej instrukcji
- LEA Ładowanie adresu efektywnego
- NOP Nie wykonuje żadnego działania
- UD2 Instrukcja niezdefiniowana
- XLAT/XLATB Tłumaczenie w oparciu o tablicę translacji
- MOVBE Przesłanie po zamianie kolejności bajtów
- CPUID Identyfikacja procesora

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

32

Wpływa na flagi: -

## Prefiks LOCK

lock

Powoduje wystawienie sygnału LOCK procesora i wykonanie w sposób niepodzielny instrukcji:

add, adc, and, brc, btr, bts, cmpxchg, cmpxch8b, cmpxch16b, dec, inc, neg, not, or, sbb, sub, xor, xadd i xchg,

jeśli argument celu jest w pamięci.

lock btr

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

33

## Instrukcja LEA

lea cel,źródło

Wczytanie wyznaczonego adresu źródła do rejestru celu.

cel:=adres źródła

lea eax,[edx+esi*4+12]	; eax=edx+esi*4+12
lea rax,[rdx+rsi*4+12]	; rax=rdx+rsi*4+12

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

34

Wpływa na flagi: -

## Instrukcja NOP

nop

Nic nie robi.

nop

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

35

## Instrukcja UD2

ud2

Generuje wyjątek *instrukcja niezdefiniowana*, nic nie robi, wprowadzona do testów.

ud2

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

36

Wpływa na flagi: -

## Instrukcja XLAT/XLATB

xlat arg

xlatb

Tłumaczenie w oparciu o tablicę translacji.

AL := DS:[(R|E)BX+AL]

xlatb

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

37

Wpływa na flagi: -

## Instrukcja MOVBE

movbe cel, źródło

Przesłanie po zamianie kolejności bajtów. Jeden z argumentów musi być rejestrem (16, 32, 64).

cel:=zamień(źródło)

movbe eax, zmienna

przed

12	c4	7f	de
----	----	----	----

po

de	7f	c4	12
----	----	----	----

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

38

## Rejestr flag

bit	Stan/wartość	Opis	typ
0	CF	Flaga przeniesienia (carry)	S
1	-	zarezerwowany	-
2	PF	Flaga parzystości (parity)	S
4	AF	Flaga wyniesienia (adjust)	S
6	ZF	Flaga zera (zero)	S
7	SF	Flaga znaków (sign)	S
8	TF	Flaga zatrzymania krokowe wykonywanie (trap)	X
9	IF	Flaga zezwolenia na przerwania (interrupt enable)	X
10	DF	Flaga kierunku (direction)	C
11	OF	Flaga przepięśnienia (overflow)	S
12, 13	IOPR	poziom uprawnienia/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	o	zarezerwowany	

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

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

39

Wpływa na flagi: -

## Instrukcja CPUID

cpuid

Identyfikacja procesora jest możliwa, jeśli bit 21 flaga ID w rejestrze flag może być zmieniana. Na podstawie EAX (czasem też ECX) podaje w EAX, EBX, ECX i EDX różne informacje o procesorze.

cpuid

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

40

## Przykład

mov eax,0  
cpuid

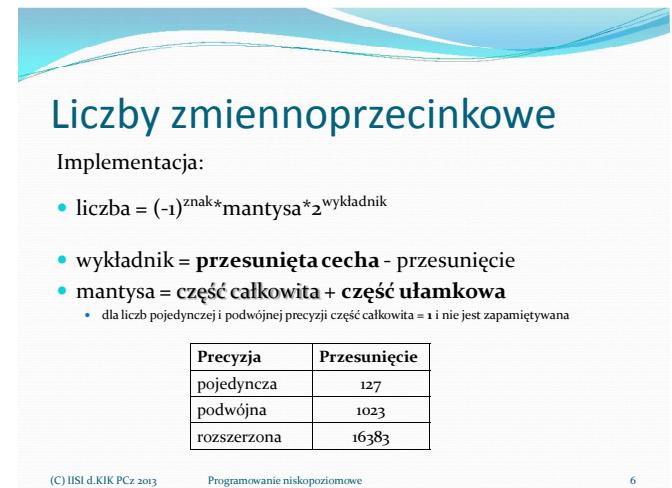
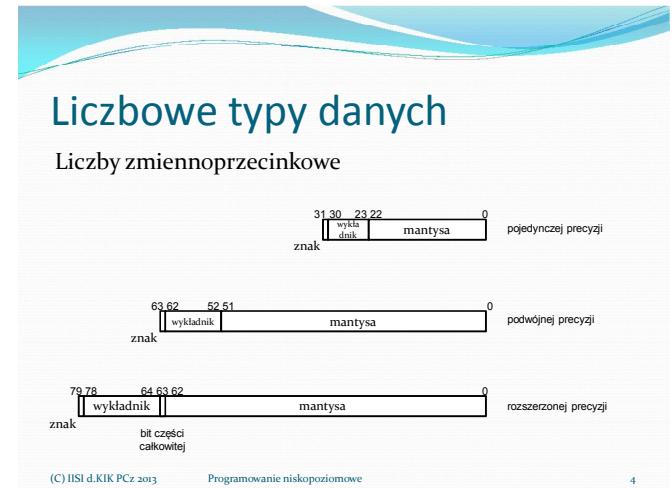
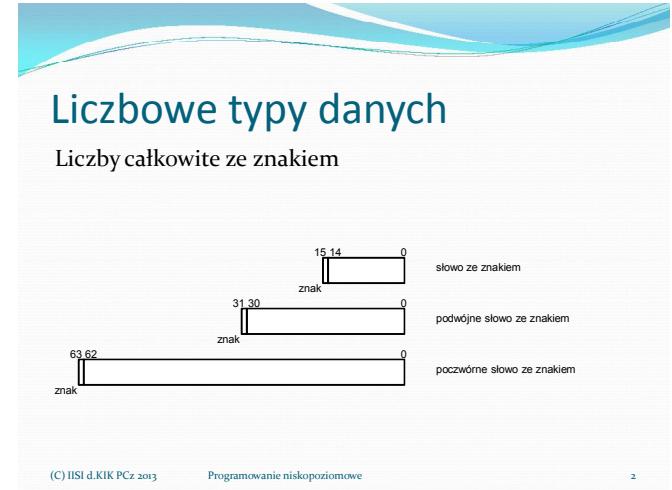
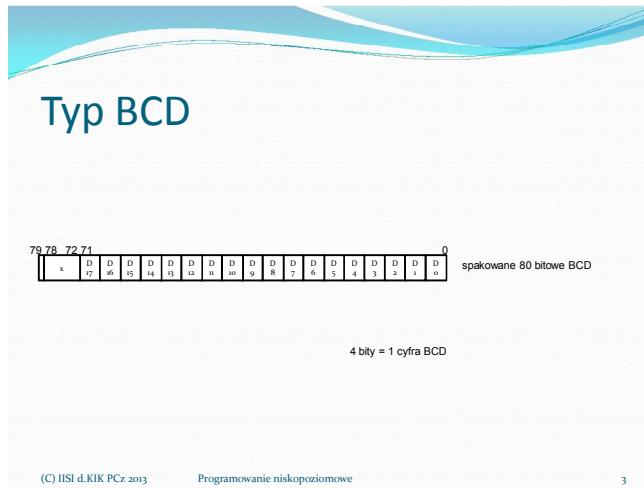
Zwraca wartość maksymalną dla cpuid oraz identyfikator producenta:

eax=max  
ebx='Genu'  
ecx='ntel'  
edx='inel'

(C) IISI d.KIK PCz 2019

Programowanie niskopoziomowe

41



## Wartości

poprawne i niepoprawne zawarte w rejestrach stosu koprocesora.

(C) IISI d.KIK PCz 2013

Klasa	Znak	Przesunięcie cecha	Mantysa	
			Część całkowita	Część ulamek
Dodatnie niewłaściwe	Prawe (obie)	0 11..11	0	11..11
		0 11..11		10..00
Aktywne (stosowe)	0 11..11	0	01..11	
		0 11..11		00..01
Dodatek zmiennoprzecinkowe	Nieskończoność	0 11..11	0	00..00
	Znormalizowane	0 11..10	1	11..11
		0 ..00..01		00..00
	Nienormalizowane	0 11..10	0	11..11
		0 ..00..01		00..00
Pseudonormalizowane	0 00..00	1	11..11	
		0 00..00		00..00
Zero	00..00	0	00..00	
		00..00		00..00
Ujemne zmiennoprzecinkowe	Pseudonormalizowane	1 00..00	1	11..11
		1 00..00		00..00
	Nienormalizowane	1 11..10	0	11..01
		1 ..00..01		00..00
	Znormalizowane	1 11..10	1	11..01
		1 ..00..01		00..00
Minus nieskończoność	1 11..11	0	00..00	
		1 11..11		00..00
Ujemne właściwe	Aktywne (stosowe)	1 11..11	0	01..11
		1 11..11		00..01
	Prawe (obie)	1 11..11	0	11..11
		1 11..11		10..00
Programowanie niskopoziomowe	szczyt stosu	-- 15 bits --	-- 63 bits --	-- 63 bits --

7

## Liczby zmiennoprzecinkowe

precyzja		
pojedyncza	podwójna	rozszerzona
cyfry znaczące	6	15
wartość największa	3,402823466E38	1,7976931348623158E308
wartość najmniejsza	1,175494351E-38	2,2250738585072024E-308
		3,3621031431120935E-4932

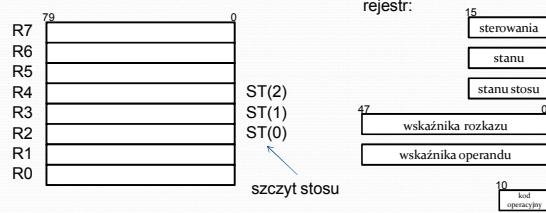
(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

8

## Koprocesor - budowa

stos rejestrów



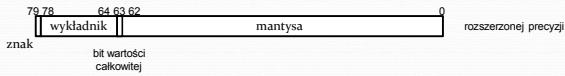
(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

9

## Koprocesor

Rejestry Ro, R1, R2, R3, R4, R5, R6, R7



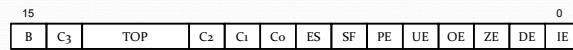
(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

10

## Koprocesor

rejestr stanu



- B – koprocesor zajęty
- Co-C<sub>3</sub> – bity rodzaju wyniku
- TOP – wskaźnik stosu
- ES – znacznik błędu
- SF – znacznik błędu stosu
- PE – błąd niedokładności wyniku
- UE – błąd niedomiaru
- OE – błąd nadmiaru
- ZE – błąd dzielenia przez zero
- DE – błąd znormalizowanego argumentu
- IE – błąd niedozwolonej operacji

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

11

## Koprocesor

rejestr sterowania



- X – interpretacja nieskończoności (tylko 287)
- RC – sterowanie zaokrąglением: do najbliższej(oo), w dół (01), w górę (10), obcięcie (11)
- PC – sterowanie dokładnością obliczeń(23 (oo), 53 (10) i 63 (11) bity)
- PM – maskowanie błędu niedokładności wyniku
- UM – maskowanie błędu niedomiaru
- OM – maskowanie błędu nadmiaru
- ZM – maskowanie błędu dzielenia przez zero
- DM – maskowanie błędu znormalizowanego argumentu
- IM – maskowanie błędu niedozwolonej operacji

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

12

## Koprocesor

rejestr stanu zawartości rejestrów stosu

15	TAG(7)	TAG(6)	TAG(5)	TAG(4)	TAG(3)	TAG(2)	TAG(1)	TAG(0)	0
----	--------	--------	--------	--------	--------	--------	--------	--------	---

TAG – pola określają zawartość poszczególnych rejestrów stosu:

oo – liczba prawidłowa

01 – zero

10 – wartość specjalna (nie liczba NaN, nieskończoność..) lub zdenormalizowana

11 – rejestr pusty

## 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)

## Instrukcja FLD

fld źródło

Przesyła liczbę zmiennoprzecinkową z rejestrów st(i) lub z pamięci na wierzchołek stosu.

*fpush(źródło)*

fld st(3)

fld zmienna

## Instrukcja FST

fst cel

Zapisuje liczbę zmiennoprzecinkową z wierzchołka stosu do rejestrów st(i) lub pamięci.

*cel:=st(o)*

fst st(3)

fst zmienna

## Instrukcja FSTP

fstp cel

Zapisuje liczbę zmiennoprzecinkową z wierzchołka stosu do rejestrów st(i) lub pamięci i zdejmuję ją ze stosu.

*cel:=st(o)*

*fpop*

fstp st(3)

fstp zmienna

## Instrukcja FILD

fild źródło

Przesyła liczbę całkowitą z pamięci na wierzchołek stosu.

*fpush(źródło)*

fild zmienna

## Instrukcja FIST

fist cel

Zapisuje liczbę w formacie całkowitym z wierzchołka stosu do pamięci.

cel:=int(st(o))

fist zmienna

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

19

## Instrukcja FISTP

fistp cel

Zapisuje liczbę w formacie całkowitym z wierzchołka stosu do pamięci i zdejmuje ją ze stosu.

cel:=int(st(o))

fpop

fistp zmienna

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

20

## Instrukcja FBLD

fbld źródło

Przesyła liczbę całkowitą BCD z pamięci na wierzchołek stosu.

fpush(źródło)

fbld zmienna

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

21

## Instrukcja FBSTP

fbstp cel

Zapisuje liczbę w formacie całkowitym BCD z wierzchołka stosu do pamięci i zdejmuje ją ze stosu.

cel:=int(st(o))

fpop

fbstp zmienna

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

22

## Instrukcja FXCH

fxch st(i)

fxch

Zamienia liczbę z wierzchołka stosu z wartością w rejestrze celu. Bez parametru celem jest st(1).

st(i)↔st(o)

fxch st(5)

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

23

## Instrukcja FCMOVcc

FCMOVE	przesłanie warunkowe (jeśli równe, ZF=1)
FCMOVNE	przesłanie warunkowe (jeśli nie równe, ZF=0)
FCMOVB	przesłanie warunkowe (jeśli poniżej, CF=1)
FCMOVBE	przesłanie warunkowe (jeśli poniżej lub równe, CF=1 lub ZF=1)
FCMOVNB	przesłanie warunkowe (jeśli nie poniżej, CF=0)
FCMOVNBE	przesłanie warunkowe (jeśli nie poniżej lub równe, CF=0 i ZF=0)
FCMOVU	przesłanie warunkowe (jeśli nieuporządkowane, PF=1)
FCMOVNU	przesłanie warunkowe (jeśli uporządkowane, PF=0)

fcmovcc st(o),st(i)

Jeśli jest spełniony warunek cc zapisuje do st(o) wartość rejestru źródła st(i).  
if cc then st(o):=st(i)

fcmovnb st(o),st(5)

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

24



## Instrukcja FDIV/FDIVP/FIDIV

Dzieli rejestr celu przez wartość źródła. Dla FIDIV źródłem jest liczba całkowita. FDIVP zdejmuje wierzchołek stosu.

```
st(cel):=st(cel)/st(źródło)|zmienna
```

```
fdiv st,st(4)
```

## Przykład

$$y = ax^3 + bx^2 + cx + d$$

```
fld d ;d
fld x ;x;d
fld st ;x;x;d
fmul st,st(1) ;px;x;d
fld st(1) ;x;xx;x;d
fmul st,st(1) ;xxx;xx;x;d
fmul a ;axxx;xx;x;d
faddp st(3),st ;xx;x;axxx+d
fmul b ;b*xx;x;axxx+d
faddp st(2),st ;x;axxx+b*xx+d
fmul c ;c*x;axxx+b*xx+d
fadd ;axxx+b*xx+c*x+d
fstp y
```

## Instrukcja FDIVR/FDIVRP/FIDIVR

Dzieli źródło przez rejestr celu. Wynik umieszcza w rejestrze celu. Dla FIDIVR źródłem jest liczba całkowita. FDIVRP zdejmuje wierzchołek stosu.

```
st(cel):=st(źródło)|zmienna/st(cel)
```

```
fdivr st,st(3)
```

## Przykład

$$y = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix} \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix}$$

<pre> mov ecx,n mov esi,x mov edi,z fld [esi] ;x fld [edi] ;z; x fmul [edi] ;\$:=x*z dec ecx dec esi,8 @i: add esi,8 add edi,8 fld [esi] ;x; s fmul [edi] ;x*z; s fadd ;\$:=s+x*z dec ecx jnz @i </pre>	<pre> mov esi,x mov edi,y fld [esi] ;x mov ecx,n fmul [edi] ;\$:=x*z dec ecx add esi,8 add edi,8 fld [esi] ;x; s fmul [edi] ;x*z; s dec ecx fadd ;\$:=s+x*z jnz @i </pre>
---	---

## Instrukcja FPREM/FPREM1

Oblicza resztę z dzielenia st/st(1), wynik umieszcza w rejestrze st. Wynik jest dokładny. Jeśli st/st(1) jest zbyt duży ( $c2=1$ ) w st umieszczona zostaje częściowa reszta i trzeba powtórzyć instrukcję. Zakres reszty:

FPREM  $<-|st(1)|, |st(1)|>$

FPREM1  $<-|st(1)/2|, |st(1)/2|>$

$Q := \text{int}|\text{round}(st/st(1))|$

$st:=st - Q*st(1)$

fprem

## Instrukcja FABS

Oblicza wartość bezwzględną liczby z wierzchołka stosu.

$st:=|st|$

fabs

## Instrukcja FCHS

Zmienia znak liczby na wierzchołku stosu.

**st:=-st**

**fchs**

## Instrukcja FRNDINT

Zaokrąga liczbę na wierzchołku stosu.

**st:=round(st)**

**frndint**

## Instrukcja FSCALE

Skalowanie przez potęgę 2. Do wykładnika st dodaje część całkowitą st(1).

**st:=st\*2<sup>int(st(1))</sup>**

**fscale**

## Instrukcja FSQRT

Oblacza pierwiastek kwadratowy z liczby na wierzchołku stosu.

**st:=sqrt(st)**

**fsqrt**

## Instrukcja FXTRACT

Oblicza wykładnik i mantysę liczby z rejestru st.

**st:=wykładnik(st)**

**fpush(mantysa(st))**

**fxtract**

## Przykład

$$y = \frac{\sqrt{|a-b|}}{a+b}$$

fld a	a
fld st(0)	a; a
fld b	b; a; a
fadd st(2),st	b; a; a+b
fsubp st(1),st	a-b; a+b
fabs	a-b ; (a+b)
fsqrt	sqrt( a-b ); (a+b)
fdivr	aqrt( a-b )/(a+b)
fst y	

## Operacje ładowania stałych

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

Instrukcje zapisują stałe na wierzchołku stosu (st).

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

43

## 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  $y * \log_2(x+1)$

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

44

## Instrukcja FSIN

Oblicza sinus liczby zawartej w st(o) i wynik umieszcza w st(o). Jeśli st nie zawiera się w  $<-2^{63}, 2^{63}>$ , wówczas flaga C2 jest ustawiana. Argument można zredukować instrukcją FPREM z dzielnikiem  $2\pi$ .

`st:=sin(st)`

`fsin`

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

45

## Instrukcja FCOS

Oblicza cosinus liczby zawartej w st(o) i wynik umieszcza w st(o). Jeśli st nie zawiera się w  $<-2^{63}, 2^{63}>$ , wówczas flaga C2 jest ustawiana. Argument można zredukować instrukcją FPREM z dzielnikiem  $2\pi$ .

`st:=cos(st)`

`fcos`

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

46

## Instrukcja FSINCOS

Oblicza sinus i cosinus liczby zawartej w st(o) i wynik umieszcza w st(o) i na wierzchołku stosu. Jeśli st nie zawiera się w  $<-2^{63}, 2^{63}>$ , wówczas flaga C2 jest ustawiana. Argument można zredukować instrukcją FPREM z dzielnikiem  $2\pi$ .

`temp:=cos(st)`

`st:=sin(st)`

`fpush(temp)`

`fsincos`

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

47

## Instrukcja FPTAN

Oblicza tangens liczby zawartej w st(o) i wynik umieszcza w st(o) i 1.0 na wierzchołku stosu. Jeśli st nie zawiera się w  $<-2^{63}, 2^{63}>$ , wówczas flaga C2 jest ustawiana. Argument można zredukować instrukcją FPREM z dzielnikiem  $2\pi$ .

`st:=tan(st)`

`fpush(1.0)`

`fptan`

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

48

## Instrukcja FPATAN

Oblicza arcus tangens (kąt) ilorazu st(1)/st i wynik umieszcza w st(1), a st zdejmuje z wierzchołka stosu.

st(1):=arctg(st(1)/st)

fpop

fpatan

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

49

## Instrukcja F2XM1

Oblicza  $2^{\text{st}}$  – 1. st musi być w przedziale  $<-1,1>$ .

st:= $2^{\text{st}-1}$

f2xm1

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

50

## Instrukcja FYL2X

Oblicza  $y \cdot \log_2 x$ . st>0

st(1):=st(1)\* log<sub>2</sub>st

fpop

fyl2x

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

51

## Instrukcja FYL2XP1

Oblicza  $y \cdot \log_2 x$ . Liczba w rejestrze st musi spełniać

$$-(1 - \frac{\sqrt{2}}{2}) < \text{st} < (1 - \frac{\sqrt{2}}{2})$$

st(1):=st(1)\* log<sub>2</sub>(st+1)

fpop

fyl2xp1

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

52

## Przykład

$$a = x^y = 2^{y \cdot \log_2 x}$$

```

fld y          ;y
fld x          ;x; y
fyl2x         ;y*log2x
f2xm1         ;2^(y*log2x)-1    !!!!
fldi           ;1; 2^(y*log2x)-1
fadd           ;a
fstp a

```

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

53

## Przykład

$$a = e^x = 2^{x \cdot \log_2 e}$$

```

fld  x          ;x
fldl2e        ;log2e; x
fmul          ;x*log2e
fld  st(0)     ;x*log2e; x*log2e
frndint      ;round(x*log2e); x*log2e
fsub  st(1), st ;round(x*log2e); x*log2e - round(x*log2e)
fxch  st(1)     ;x*log2e - round(x*log2e); round(x*log2e)
f2xm1         ;2^(x*log2e - round(x*log2e))-1; round(x*log2e)
fldi           ;1,2^(x*log2e - round(x*log2e))-1; round(x*log2e)
fadd           ;2^(x*log2e - round(x*log2e)); round(x*log2e)
fSCALE        ;2^(x*log2e - round(x*log2e))*2^round(x*log2e)
                ;2^(x*log2e - round(x*log2e)+round(x*log2e))= 2^(x*log2e)
fstp a

```

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

54

## Przykład

$$a = \log_b x = \log_2 x / \log_2 b$$

```

fldt      ;t
fld x      ;x; t
fyl2x     ;log2x
fldt      ;t; log2x
fld b      ;b; t; log2x
fyl2x     ;log2b; log2x
fdiv      ;a
fst a

```

## 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 o.o
- FXAM sprawdzenie liczby zmiennoprzecinkowej

## Instrukcja

### FCOM/FCOMP/FCOMPP

fcom/fcomp źródło  
fcompp

Porównanie liczb zmiennoprzecinkowych st i źródła. Źródłem może być rejestr st(i) lub zmienna. Jeśli źródło nie jest podane, to jest nim st(1). Fcomp zdejmuje liczbę z wierzchołka stosu, fcompp zdejmuje dwie liczby.

st(o)<=>? źródło  
fpop ;dla fcomp,fcompp  
fpop ;dla fcompp

fcom st(3)

Wpływ na flagi: C3 C2 C0

## Instrukcja

### FUCOM/FUCOMP/FUCOMPP

fucom/fucomp st(i)  
fucompp

Porównanie liczb zmiennoprzecinkowych st i st(i). Jeśli rejestr nie jest podany, to jest nim st(1). Fucomp zdejmuje liczbę z wierzchołka stosu, fucompp zdejmuje dwie liczby. Nie zgłasza się wyjątku #IA dla niewłaściwych.

st(o)<=>? st(i)  
fpop ;dla fucomp,fucompp  
fpop ;dla fucompp

fucomp st(7)

Wpływ na flagi: C3 C2 C0

## Stan flag po porównaniu

Flagi koprocesora C<sub>3</sub>, C<sub>2</sub>, Co odpowiadają flagom ZF, PF i CF procesora.

relacja	flagi	C <sub>3</sub>	C <sub>2</sub>	Co
	ZF	PF	CF	
st(o)>źródło	o	o	o	
st(o)<źródło	o	o	1	
st(o)=źródło	1	o	o	
nieuporządkowane*	1	1	1	1

\*flagi nie są ustawiane, jeśli wystąpi niezamaskowany wyjątek #IA

Wpływ na flagi: C3 C2 C0

## Instrukcja FICOM/FICOMP

ficom/ficomp zmienna

Porównanie st z liczbą całkowitą w pamięci (16/32). Ficomp zdejmuje liczbę z wierzchołka stosu.

st(o)<=>? zmienna  
fpop ;dla ficomp

ficom liczba\_c

## Instrukcja

### FCOMI/FCOMIP/FUCOMI/FUCOMIP

fcomi/fcomip st(i)

fucomi/fucomip st(i)

Porównanie st z liczbą w st(i). Fcompi/fucomip zdejmuję liczbę z wierzchołka stosu. **Ustawia flagi: ZF, PF i CF.** Fucomi i fucomip nie zgłaszały wyjątku #IA dla nielicznych pasywnych

st(o)<=>? st(i)

fpop ;dla fcomip/fucomip

fucomi st(4)

Wpływ na flagi: ZF PF CF

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

61

Wpływ na flagi: C3 C2 C0

## Instrukcja FTST

ftst

Porównanie liczb zmiennoprzecinkowych st i o.o.

st(o)<=>? o.o

ftst

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

62

## Instrukcja FXAM

fxam

Sprawdza liczbę na wierzchołku stosu. C1 = znak liczby.

znaczenie	flagi	C3	C2	Co
nieznormalizowana, pseudo (nie)liczba		o	o	o
nieliczba		o	o	1
znormalizowana		o	1	o
nieskończoność		o	1	1
zero		1	o	o
rejestr pusty		1	o	1
zdenormalizowana		1	1	o

Wpływ na flagi: C3 C2 C1 C0

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

63

## Operacje sterowania koprocessorem

- FINCSTP zwiększenie rejestru wskaźnika stosu koprocessora
- FDECSTP zmniejszenie rejestru wskaźnika stosu koprocessora
- FFREE zwolnienie rejestru zmiennoprzecinkowego
- FINIT inicjalizacja koprocessora po sprawdzeniu zgłoszenia błędu numerycznego
- FININIT inicjalizacja koprocessora bez sprawdzenia zgłoszenia błędu numerycznego
- FCLEX zerwanie flag błędów numerycznych po sprawdzeniu zgłoszenia błędu numerycznego
- FNCLEX zerwanie 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 koprocessora po sprawdzeniu zgłoszenia błędu numerycznego
- FNSTENV zapamiętanie środowiska koprocessora bez sprawdzenia zgl. błędu numerycznego
- FLDENV wczytanie środowiska koprocessora
- FSAVE zapamiętanie zawartości koprocessora po sprawdzeniu zgłoszenia błędu numerycznego
- FNSAVE zapamiętanie zawartości koprocessora bez sprawdzenia zgłoszenia błędu numerycznego
- FRSTOR wczytanie zawartości koprocessora
- FSTSW zapamiętanie rejestru stanu po sprawdzeniu zgłoszenia błędu numerycznego
- FNSTSW zapamiętanie rejestru stanu bez sprawdzenia zgłoszenia błędu numerycznego
- WAIT/FWAIT czekanie na koprocessor
- FNOP nic nie robi

(C) IISI d.KIK PCz 2013 Programowanie niskopoziomowe

64

## Instrukcja FINCSTP

fincstp

Zwiększenie rejestru wskaźnika stosu koprocessora. Nie usuwa liczb ze stosu.

top := (top+1) mod 8

fincstp

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

65

## Instrukcja FDECSTP

fdecstp

Zmniejszenie rejestru wskaźnika stosu koprocessora.

top := (top-1) mod 8

fdecstp

(C) IISI d.KIK PCz 2013 Programowanie niskopoziomowe

66

## Instrukcja FFREE

ffree st(i)

Zwolnienie rejestru koprocesora. Rejestr wskaźnika stosu koprocesora nie jest zmieniany.

tag(i):= 11 b

ffree st(3)

## Instrukcja FCLEX/FNCLEX

fclex/fnclex

Zerowanie flag błędów numerycznych. Fnflex - bez sprawdzenia zgłoszenia błędu numerycznego.

fnflex

## Instrukcja FLDCW

fldcw źródło

Wczytanie rejestru sterowania. Źródło jest dwubajtową zmienną.

fldcw zmienna

## Instrukcja FINIT/FNINIT

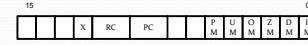
finit/fninit

Inicjalizacja koprocesora. Fninit inicjalizacja koprocesora bez sprawdzenia zgłoszenia błędu numerycznego.

finit

rejestr stanu = 0

rejestr stanu zawartości rejestrów stosu = offff h  
rejestr sterowania = 37f h



X - interpretacja niedokonczona (tryb zbf)  
RC - sterowanie zaokrągleniem do najbliższej(oo), w dół (oo), w górę (oo), okrągłe (o)  
PC - adresowanie niedokładnością i bliskiecią (oo), 53 (oo) i 63 (uu) bity  
FM - maskowanie błędu niedokładności i wyniku  
UM - maskowanie błędu nadmiaru  
OM - maskowanie błędu ujemnego  
ZM - maskowanie błędu zerującego  
DM - maskowanie błędu zdemoralizowanego argumentu  
IM - maskowanie błędu niedowolonej operacji

## Instrukcja FSTCW/FNSTCW

fstcw/fnstcw cel

Zapamiętanie rejestru sterowania. Cel jest dwubajtową zmienną albo rejestr AX. Fnstcw - bez sprawdzenia zgłoszenia błędu numerycznego.

fnstcw

## Instrukcja FSTENV/FNSTENV

fstenv/fnstenv cel

Zapamiętanie środowiska koprocesora . Cel jest 14/28 bajtowym obszarem (tryb 16/32 bitowy). Fnstenv - bez sprawdzenia zgłoszenia błędu numerycznego.

fnstenv fsrodowisko

## Instrukcja FLDENV

fldenv źródło

Wczytanie środowiska koprocesora . Źródło jest 14/28 bajtowym obszarem (tryb 16/32 bitowy).

fldenv fsrodowisko

## Instrukcja FRSTOR

frstor źródło

Wczytanie zawartości koprocesora . Źródło jest 94/108 bajtowym obszarem (tryb 16/32 bitowy).

frstor fstan

## Instrukcja WAIT/FWAIT

wait/fwait

Czekanie przez procesor na gotowość koprocesora (na zakończenie wykonywania instrukcji).

fwait

## Instrukcja FSAVE/FNSAVE

fsave/fnsave cel

Zapamiętanie zawartości koprocesora (środowisko + rejesty zmiennoprzecinkowe). Cel jest 94/108 bajtowym obszarem (tryb 16/32 bitowy). Fnsave - bez sprawdzenia zgłoszenia błędu numerycznego.

fnsave fstan

## Instrukcja FSTSW/FNSTSW

fstsw/fnstsw cel

Zapamiętanie rejestru stanu. Cel jest dwubajtową zmienną albo rejestrem AX. Fnstsw - bez sprawdzenia zgłoszenia błędu numerycznego.

fnstsw

## Instrukcja FNOP

fnop

Nic nie robi.

fnop

## Przykład

$$\sqrt{\Delta} = \sqrt{b^2 - 4ac}$$

fld b	;b	fld b	;b
fld st(o)	;b; b	fld st(o)	;b; b
fmul	;bb	fmul	;bb
fld a	;a; bb	fld a	;a; bb
fmul c	;ac; bb	fmul c	;ac; bb
fadd st,st(o)	;2ac; bb	fadd st,st(o)	;2ac; bb
fadd st,st(o)	;4ac; bb	fadd st,st(o)	;4ac; bb
fsub	;bb-4ac	fsub	;bb-4ac
fst	;porównanie z o,o	fldz	;o; bb-4ac
fstsw ax	;ax-stan	fcomip st,st(1)	;porównanie z o,o
sahf	;stan do flag	jp blad	;flagi już ustawione
jp blad		jz rowne	
jz rowne		jc wieksze	
jc mniejsze		mniejsze: ....	
wieksze: ....			

# Instrukcje typu SIMD

## 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

## MMX

- wprowadzone w 1997 przez Intela dla procesorów Pentium 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.

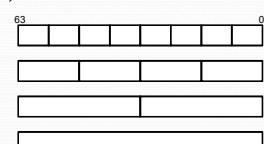
# Rejestry MMX

- 8 rejestrów 64 bitowych oznaczanych jako MM<sub>0</sub>, ..., MM<sub>7</sub>,
- wykorzystują rejesty koprocesora – młodsze 64 bity (mantysa),
- odczyt i zapis wartości, powoduje **wzięcie w użycie zawartości wszystkich rejestrów koprocesora**, nie można mieszać obliczeń MMX z obliczeniami w koprocesorze, po zakończeniu obliczeń MMX należy zwolnić rejesty.

# Typy danych

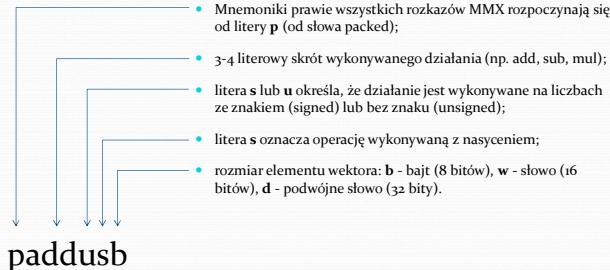
MMX wprowadził nowe wektorowe (macierzowe lub tablicowe) typy danych (ang. packed, czyli dosłownie spakowane, upakowane). „Spakowanie” polega traktowaniu danych 64-bitowych jako składających z odrębnych elementów o tej samej wielkości:

- 8 × 8 bitów (packed byte),
- 4 × 16 bitów (packed word),
- 2 × 32 bity (packed dword),
- 1 × 64 bity (quad word).





## Budowa rozkazów



**paddusb**

- Rozkaz **paddusb** wykonuje równolegle (**p**) dodawanie (**add**) bez znaku (**u**) z nasyceniem (**s**) liczb o rozmiarze bajtu (**b**)

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

7



## Nasycenie

Zakresy liczb (np. dla 8 bitów):

- bez znaku **o ... 2<sup>n-1</sup> (0 ... 255)**
- ze znakiem **- 2<sup>n-1</sup> ... 2<sup>n-1</sup>-1 (-128 ... 127)**
- Jeśli wynik jest mniejszy od najmniejszej liczby z zakresu jest ustawiany na tę liczbę.
- Jeśli wynik jest większy od największej liczby z zakresu jest ustawiany na tę liczbę.
- Dla operacji bez nasycenia wynik jest obcinany do odpowiedniej ilości bitów.

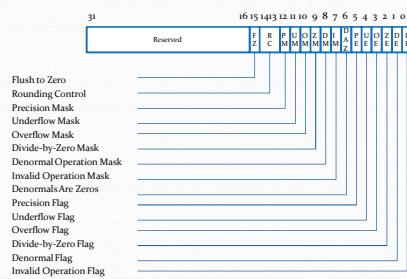
(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

8



## Rejestr MXCSR



(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

9



## Operacje przesłania

- MOVD** przesłanie podwójnego słowa
- MOVQ** przesłanie poczwórnego słowa

movd/movq cel, źródło

Przesłanie podwójnego/poczwórnego słowa do/z rejestru mmx. Przy zapisie podwójnego słowa bity 32-63 rejestru mmx wypełniane są zerami.

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

10



## 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

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

11

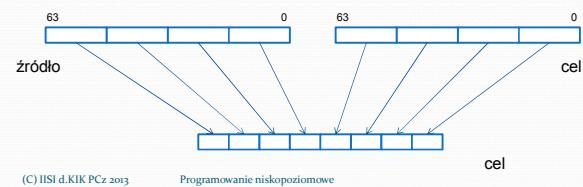


## Instrukcja PACKSSWB/PACKUSWB

**PACKSSWB** cel,źródło

**PACKUSWB** cel,źródło

Pakowanie z nasyceniem słów ze znakiem/bez znaku do bajtów. Cel musi być rejestrem mmx.



(C) IISI d.KIK PCz 2013

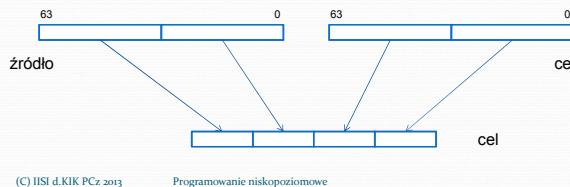
Programowanie niskopoziomowe

12

## Instrukcja PACKSSDW

PACKSSDW cel, źródło

Pakowanie z nasyceniem podwójnych słów ze znakiem do słów. Cel musi być rejestrem mmx.



(C) IISI d.KIK PCz 2013

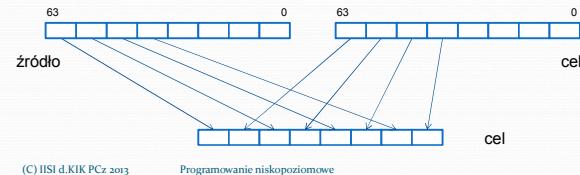
Programowanie niskopoziomowe

13

## Instrukcje PUNPCKHBW PUNPCKHWD PUNPCKHDQ

PUNPCKHBW/PUNPCKHWD/PUNPCKHDQ cel,źródło

Rozpakowanie z przeplotem starszych bajtów/słów/ podwójnych słów bez znaku. Cel musi być rejestrem mmx. Jeśli źródło = o to następuje konwersja do słów, podwójnych i poczwórnego słowa.

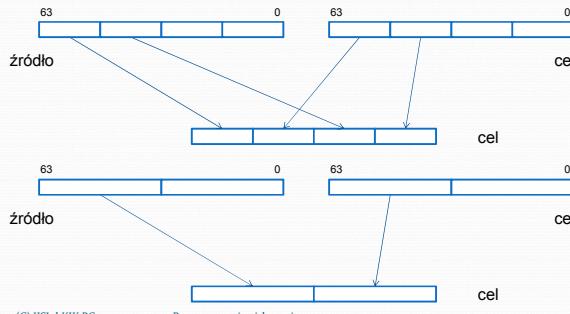


(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

14

## Instrukcje PUNPCKHWD PUNPCKHDQ



(C) IISI d.KIK PCz 2013

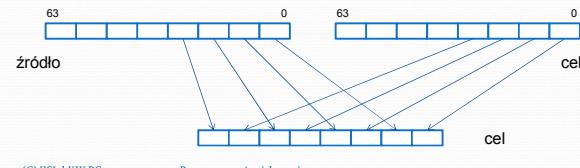
Programowanie niskopoziomowe

15

## Instrukcje PUNPCKLBW PUNPCLWD PUNPCKLDQ

PUNPCKLBW/PUNPCLWD/PUNPCKLDQ cel,źródło

Rozpakowanie z przeplotem młodszych bajtów/słów/ podwójnych słów bez znaku. Cel musi być rejestrem mmx. Jeśli źródło = o to następuje konwersja do słów, podwójnych i poczwórnego słowa.

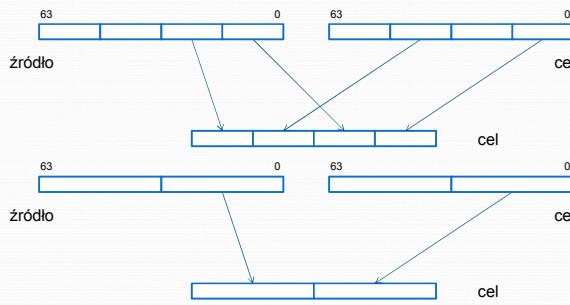


(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

16

## Instrukcje PUNPCLWD PUNPCKLDQ



(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

17

## 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

(C) IISI d.KIK PCz 2013

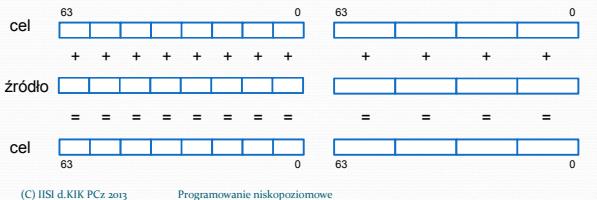
Programowanie niskopoziomowe

18

## Instrukcje PADDB PADDW PADDD

PADDB/PADDW/PADDD cel,źródło

Dodawanie wektorów bajtów/słów/podwójnych słów. Cel musi być rejestrem mmx. Wynik nie uwzględnia przeniesienia.



(C) IISI d.KIK PCz 2013

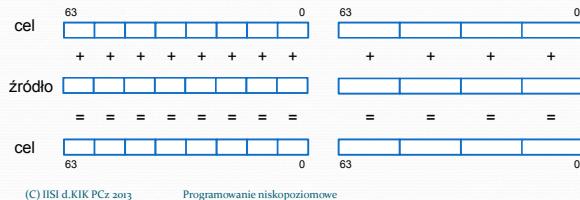
Programowanie niskopoziomowe

19

## Instrukcje PADDSB PADDSW PADDUSB PADDUSW

PADDSB/PADDSW/PADDUSB/PADDUSW cel,źródło

Dodawanie z nasyceniem wektorów bajtów/słów ze znakiem/bez znaku. Cel musi być rejestrem mmx.



(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

20

## Operacje arytmetyczne

- 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
- PSUBUD      odejmowanie z nasyceniem wektorów bajtów bez znaku
- PSUBUSW     odejmowanie z nasyceniem wektorów słów bez znaku

(C) IISI d.KIK PCz 2013

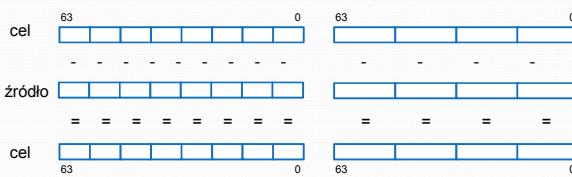
Programowanie niskopoziomowe

21

## Instrukcje PSUBSB PSUBSW PSUBUD PSUBUSW

PSUBSB/PSUBSW/PSUBUD/PSUBUSW cel,źródło

Odejmowanie z nasyceniem wektorów bajtów/słów ze znakiem/bez znaku. Cel musi być rejestrem mmx.



(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

23

## Operacje arytmetyczne

- PMULHW      mnożenie wektorów słów i zapamiętanie starszych słów wyniku
- PMULLW      mnożenie wektorów słów i zapamiętanie młodszych słów wyniku
- PMADDWD     mnożenie i dodawanie wektorów słów

(C) IISI d.KIK PCz 2013

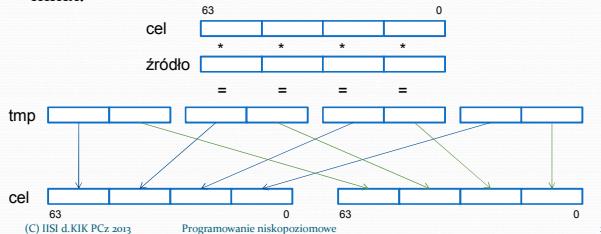
Programowanie niskopoziomowe

24

## Instrukcje PMULHW PMULLW

PMULHW/PMULLW cel,źródło

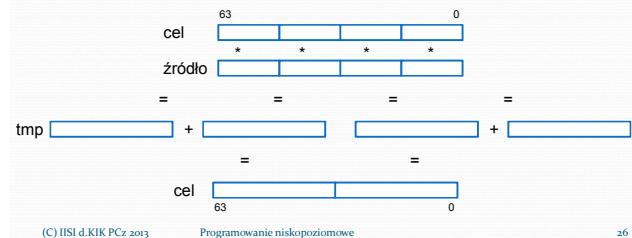
Mnożenie wektorów słów i zapamiętanie starszych/młodszych słów wyniku. Cel musi być rejestrem mmx.



## Instrukcje PMADDWD

PMADDWD cel,źródło

Mnożenie i dodawanie wektorów słów. Cel musi być rejestrem mmx.



## 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

(C) IISI d.KIK PCz 2013 Programowanie niskopoziomowe

27

## Instrukcje PCMPEQB PCMPEQW PCMPEQD

PCMPEQB/PCMPEQW/PCMPEQD cel,źródło

Sprawdzenie równości składowych wektorów bajtów/słów/podwójnych słów. Cel musi być rejestrem mmx.

```
if cel[i] = źródło[i] then cel[i] := offh/offfffh/offfffffh
else cel[i] := 0
```

(C) IISI d.KIK PCz 2013 Programowanie niskopoziomowe

28

## Instrukcje PCMPGTB PCMPGTW PCMPGTD

PCMPGTB/PCMPGTW/PCMPGTD cel,źródło

Sprawdzenie relacji większości składowych wektorów bajtów/słów/podwójnych słów ze znakiem. Cel musi być rejestrem mmx.

```
if cel[i] > źródło[i] then cel[i] := offh/offfffh/offfffffh
else cel[i] := 0
```

(C) IISI d.KIK PCz 2013 Programowanie niskopoziomowe

29

## Operacje logiczne

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

(C) IISI d.KIK PCz 2013 Programowanie niskopoziomowe

30

## Instrukcje

### PAND PANDN POR PXOR

PAND/PANDN/POR/PXOR      cel, źródło

Obliczenie bitowego iloczynu logicznego/bitowego iloczynu logicznego z negacją/bitowej sumy logicznej/bitowej sumy modulo 2. Cel musi być rejestrem mmx.

cel := cel and źródło

cel := (not cel) and źródło

cel := cel or źródło

cel := cel xor źródło

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

31

## Operacje przesunięć

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

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

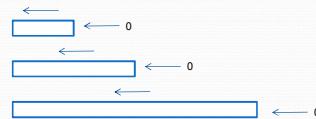
32

## Instrukcje

### PSLLW PSLLD PSLLQ

PSLLW/PSLLD/PSLLQ      cel, ile

Logiczne przesunięcie w lewo elementów wektora słów/podwójnych słów/poczwórnego słowa. Cel musi być rejestrem mmx, ile jest rejestrem mmx, zmienną lub stałą. Młodsze bity wypełniane są zerami.



(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

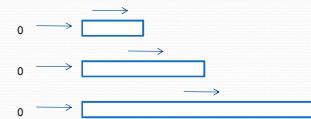
33

## Instrukcje

### PSRLW PSRLD PSRLQ

PSRLW/PSRLD/PSRLQ      cel, ile

Logiczne przesunięcie w prawo elementów wektora słów/podwójnych słów/poczwórnego słowa. Cel musi być rejestrem mmx, ile jest rejestrem mmx, zmienną lub stałą. Starsze bity wypełniane są zerami.



(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

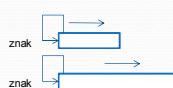
34

## Instrukcje

### PSRAW PSRAD

PSRAW/PSRAD      cel, ile

Arytmetyczne przesunięcie w prawo elementów wektora słów/podwójnych słów. Cel musi być rejestrem mmx, ile jest rejestrem mmx, zmienną lub stałą. Starsze bity wypełniane są znakiem.



(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

35

## Operacje sterujące

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

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

36

## Instrukcje FXSAVE FXRSTOR

FXSAVE/FXRSTOR cel512  
zapisanie/wczytanie stanu  
x87 FPU i rejestrów SIMD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FPU IP	MXCSR							FPU DP	FTW	FSW	FCW				
MXCSR_MASK												16			
Reserved								ST0/MM0				32			
Reserved								ST1/MM1				48			
Reserved								ST2/MM2				64			
Reserved								ST3/MM3				80			
Reserved								ST4/MM4				96			
Reserved								ST5/MM5				112			
Reserved								ST6/MM6				128			
Reserved								ST7/MM7				144			
	XMM0											160			
	XMM1											176			
	XMM2											192			
	XMM3											208			
	XMM4											224			
	XMM5											240			
	XMM6											256			
	XMM7											272			
	XMM8											288			
	XMM9											304			
	XMM10											320			
	XMM11											336			
	XMM12											352			
	XMM13											368			
	XMM14											384			
	XMM15											400			
	Reserved											416			
	Reserved											432			
	Reserved											448			
	Reserved											464			
	Reserved											480			
	Reserved											496			

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

38

## Instrukcje LDMXCSR STMXCSR

LDMXCSR/STMXCSR zmienna

Wczytanie/zapisanie zawartości rejestrów MXCSR.

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

39

40

## Instrukcje PAVGB PAVGW

PAVGB/PAVGW cel, źródło

Oblicza średnią z elementów wektorów bajtów/słów bez znaku. Cel jest rejestrem MMX.

e\_cel := (e\_cel + e\_źródło + 1) &gt;&gt; 1

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

41

42

## Instrukcja EMMS

### EMMS

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

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

38

## 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 wyniku mnożenie wektorów słów bez znaku i zapamiętanie starszych słów
- PSADBW oblicza sumę wartości bezwzględnych różnic
- PSHUFW tasuje słowa w rejestrze MMX

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

40

## Instrukcje PEXTRW PINSRW

PEXTRW cel, źródło, numer

PINSRW cel, źródło, numer

Wydobycie/wstawienie słowa o podanym numerze z/do rejestr MMX do/z rejestru ogólnego przeznaczenia lub pamięci.

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

42

## Instrukcje PMAXUB PMAXSW PMINUB PMINSW

PMAXUB/PMAXSW/PMINUB/PMINSW cel, źródło

Oblicza maksimum/minimum z elementów wektorów bajtów bez znaku/słów ze znakiem.

e\_cel := e\_cel max/min e\_źródło

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

43

## Instrukcja PMOVMSKB

PMOVMSKB cel, źródło

Przesłanie maski bajtów. Celem jest rejestr ogólnego przeznaczenia. Najstarsze bity elementów wektora z rejestrów MMX wpisywane są na bity o ..7 celu. Stosowane w celu określenia znaku lub sprawdzenia wyniku porównania.

(C) IISI d.KIK PCz 2013

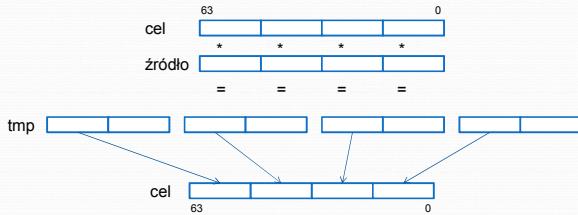
Programowanie niskopoziomowe

44

## Instrukcje PMULHUW

PMULHUW cel,źródło

Mnożenie wektorów słów bez znaku i zapamiętanie starszych słów wyniku. Cel musi być rejestrem mmx.



(C) IISI d.KIK PCz 2013

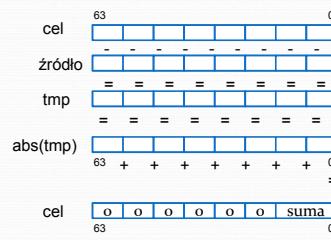
Programowanie niskopoziomowe

45

## Instrukcja PSADBW

PSADBW cel,źródło

Oblicza sumę wartości bezwzględnych różnic.



(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

46

## Instrukcja PSHUFW

PSHUFW cel, źródło, kolejność

Tasuje słowa w rejestrze celu MMX. Źródłem jest rejestr MMX lub pamięć. Bity 7,6; 5,4 ;3,2 i 1,0 stałej kolejności określają numer słowa w źródle, które zostanie umieszczone jako 3,2,1 i 0 w rejestrze celu np.

dla źródła= **0123 4567 89ab cdefh** i kolejności oo oo 10 11b będzie: cel=**ocdef 89ab 4567 0123h**

(C) IISI d.KIK PCz 2013

Programowanie niskopoziomowe

47