Materiały do wykładu

7. Architektura x86

Marcin Peczarski

Instytut Informatyki
Uniwersytet Warszawski

6 maja 2014

Narodziny

- 1978 – Intel 8086
  - architektura 16-bitowa
  - 5 MHz, obudowa DIP40, 29000 tranzystorów
- 1979 – Intel 8088
  - 8-bitowa zewnętrzna szyna danych
- 1980 – Intel 8087
  - koprocesor zmiennoprzecinkowy, 45000 tranzystorów
- 1981 – IBM PC
  - procesor 8088 4,77 MHz
  - maksymalnie 640 KiB pamięci operacyjnej

Zanim powstała architektura x86

- Mikroprocesory 4-bitowe
  - 1971 – Intel 4004
  - 1974 – Intel 4040
- Mikroprocesory 8-bitowe
  - 1972 – Intel 8008
  - 1974 – Intel 8080 – bardzo popularna architektura 8-bitowa w latach 80-tych, wiele klonów, polski odpowiednik to MCY7880
  - 1975 – Intel 8085 – programowo kompatybilny z 8080, tylko dwie nowe instrukcje, zupełnie nowy projekt hardware
  - 1976 – Zilog Z80 – nowe rejestry, istotne poszerzenie repertuaru instrukcji 8080, jeden z najpopularniejszych mikroprocesorów

W kierunku architektury 32-bitowej

- Intel 80186, Intel 80188
  - ulepszone 8086 i 8088
- Intel 80286
  - mikroprocesor 16-bitowy drugiej generacji, tryb chroniony
- Intel 386, AMD Am386
  - architektura 32-bitowa
- Intel 486, AMD Am486
  - zintegrowana jednostka zmiennoprzecinkowa, przetwarzanie potokowe, wbudowana pamięć podręczna L1

Gdy nazwa procesora przestała cokolwiek oznaczać

- Intel Pentium
  - procesor superskalarny, dwa potoki, predykcja skoków, możliwość łączenia dwóch procesorów
- Intel Pentium MMX
  - namiastka przetwarzania SIMD
- Intel Pentium Pro, Cyrix 6x86, AMD K5
  - procesory superskalarne, wewnętrzna architektura RISC, przemianowywanie rejestrów
- Intel Pentium III, AMD K6-2
  - rozwój SIMD o operacje zmiennoprzecinkowe

Wersji 64-bitowej chyba miało nie być

- Wiele firm produkowało klony procesorów x86.
  - Ostały się dwie.
- AMD Athlon 64, AMD Opteron, Intel Pentium 4 (od 2004)
  - architektura 64-bitowa
- Intel Xeon, Intel Core 2, Intel Core i7 Extreme Edition, AMD Phenom, AMD Turion, Third Generation AMD Opteron, ...
  - różne fajne nazwy dla badziewiastej architektury x86
- Ponad $2{,}5 \cdot 10^9$ tranzystorów w procesorze
  - wiele rdzeni, zintegtowany procesor graficzny, pamięci podręczne

Główne cechy architektury x86

- CISC
- Kolejność bajtów little-endian
- Numeracja bitów: 0 = LSB
- Bezpośredni wpływ architektury mikroprocesorów 8080, 8085 i Z80 na architekturę 8086
- Utrzymywanie możliwie pełnej wstecznej kompatybilności
- Brak spójnej wizji rozwoju
- Chaotyczne dodawanie nowych instrukcji

Oznaczenia

- IA-32 – oficjalna nazwa 32-bitowej wersji architektury Intel x86
- x86-64 – 64-bitowe rozszerzenie AMD
- EM64T – nazwa wprowadzona przez Intel dla klonu 64-bitowego rozszerzenia
- IA-32e – nazwa 64-bitowych trybów pracy używana przez Intel
- AMD64 – nazwa handlowa używana przez AMD
- Intel64 – nazwa handlowa używna przez Intel
- Nie mylić z architekturą procesora Itanium oznaczaną IA-64

Legacy Mode

- Real Mode, tryb rzeczywisty
  - emulacja 16-bitowego mikroprocesora 8086
  - tryb startowy po włączeniu zasilania
- Protected Mode, tryb chroniony
  - 32-bitowy z segmentacją i stronicowaniem
  - 16-bitowy tryb chroniony procesora 286
  - Virtual 8086 Mode, programy skompilowane dla Real Mode
- System Management Mode (SMM)
  - implementacja funkcji związanych np. z zarządzaniem energią i trybami uśpienia

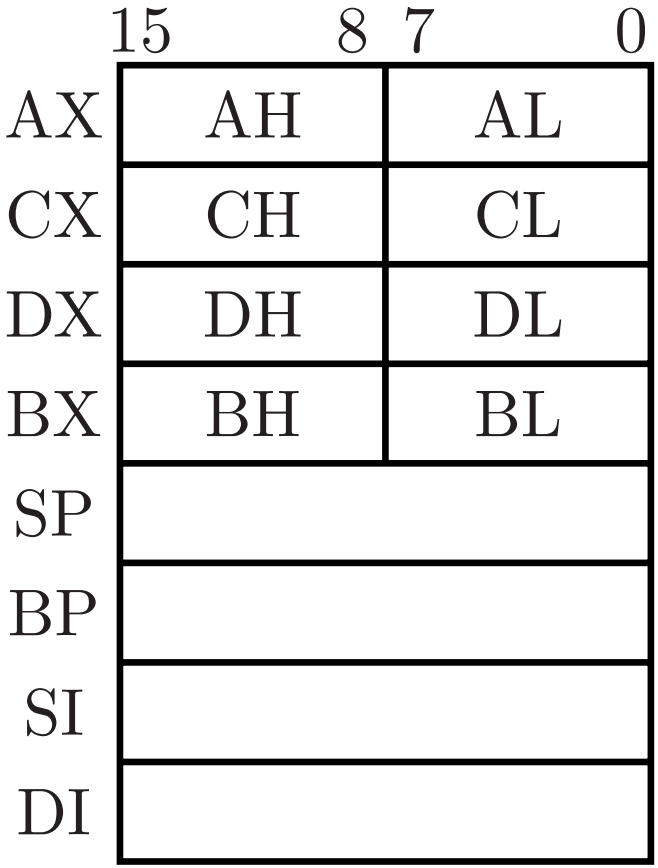
Long Mode

- Compatibility Mode
  - uruchamianie programów skompilowanych dla Legacy Protected Mode w środowisku 64-bitowego systemu operacyjnego
- 64-bit Mode
  - pełny tryb 64-bitowy

Formaty danych

- Liczby całkowite
  - 8-, 16-, 32- i 64-bitowe bez zanku NKB i ze znakiem U2
  - BCD (z wyjątkiem 64-bit Long Mode)
  - spakowane BCD w zakresie od $-10^{18}+1$ do $10^{18}-1$
- Liczby wymierne
  - 32-bitowy format pojedynczej precyzji
  - 64-bitowy format podwójnej precyzji
  - 80-bitowy format chwilowy
- Tablice bitów
- Ciągi bajtów
- Wektory

Rejestry ogólnego przeznacznia w trybie 16-bitowym



Rejestry ogólnego przeznacznia w trybie 32-bitowym

	31 … 16	15 … 8	7 … 0
EAX		AH	AX AL
ECX		CH	CX CL
EDX		DH	DX DL
EBX		BH	BX BL
ESP		SP	
EBP		BP	
ESI		SI	
EDI		DI	

Rejestry ogólnego przeznacznia w trybie 64-bitowym

- 64-bitowe: RAX, RCX, RDX, RBX, RSP, RBP, RSI, RDI, R8, R9, R10, R11, R12, R13, R14, R15
- 32-bitowe: EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI, R8D, R9D, R10D, R11D, R12D, R13D, R14D, R15D
- 16-bitowe: AX, CX, DX, BX, SP, BP, SI, DI, R8W, R9W, R10W, R11W, R12W, R13W, R14W, R15W
- 8-bitowe: AL, CL, DL, BL, SPL, BPL SIL, DIL, R8B, R9B, R10B, R11B, R12B, R13B, R14B, R15B
- Dziwna semantyka operacji
  - Operacje 8- i 16-bitowe nie modyfikują starszej części rejestru.
  - Operacje 32-bitowe zerują starszą część rejestru.

Rejestry zmiennoprzecinkowe i wektorowe

- Przez zbiór instrukcji f87 widziane jako stos ośmiu rejestrów 80--bitowych ST0 – ST7
  - chwilowy format zmiennopozycyjny
- Przez zbiór instrukcji MMX widziane jako 8 rejestrów 64-bitowych MM0 – MM7
  - wektor 8 wartości 8-bitowych
  - wektor 4 wartości 16-bitowych
  - wektor 2 wartości 32-bitowych
  - wartość 64-bitowa

Rejestry wektorowe

- Używane w zbiorze instrukcji SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, SSE5, AVX (Advanced Vector Extensions)

- 8 rejestrów 128-bitowych XMM0 – XMM7 (tryb 32-bitowy) lub 16 rejestrów 128-bitowych XMM0 – XMM15 (tryb 64-bitowy):
  - 16 wartości 8-bitowych całkowitych
  - 8 wartości 16-bitowych całkowitych
  - 4 wartości 32-bitowych całkowitych lub zmiennoprzecinkowych
  - 2 wartości 64-bitowych całkowitych lub zmiennoprzecinkowych

- 16 rejestrów 256-bitowych YMM0 – YMM15

- 32 rejestry 512-bitowe ZMM0 – ZMM31

Zestaw instrukcji

- Typowy dla architektury CISC
- Np. skrócone kody dla instrukcji ładowania akumulatora AL/AX/EAX/RAX
- Opis wszystkich instrukcji to ponad 1500 stron
- Mniej więcej co dwa lata kilkadziesiąt nowych instrukcji

Instrukcje arytmetyczno-logiczne

- Zestaw typowy dla architektury CISC
- Mnożenie i dzielenie z użyciem rejestrów AL/AX/EAX/RAX i AH/DX/EDX/RDX
- Rejestr CL w operacjach przesuwania i rotacji bitów
- Wykonywanie operacji na argumentach w pamięci

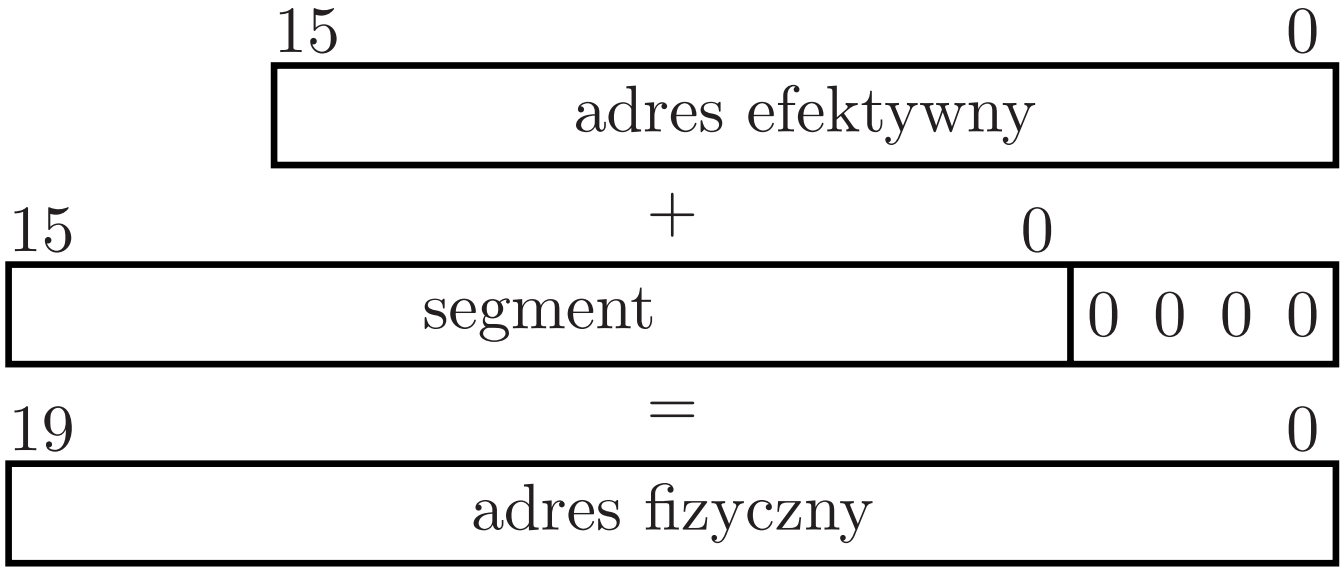
Instrukcje warunkowe

- Dwuetapowe
- Rejestr znaczników FLAGS/EFLAGS/RFLAGS
- Skok krótki (short), zmiana wskaźnika instrukcji IP/EIP/RIP w zakresie od -128 do $+127$ bajtów
- Skok bliski (near) w dowolne miejsce bieżącego segmentu kodu
- Przypisanie warunkowe SET
- Przepisanie warunkowe CMOV
- Instrukcje warunkowe z rejestrem CX/ECX/RCX jako licznikiem

Tryby adresowania

- Natychmiastowy
- Bezpośredni
- Rejestrowy
- Rejestrowy pośredni, indeksowy
- Stosowy
- Względny w instrukcjach skoku i wołania procedury
- Rejestrowy pośredni z postinkrementacją
- Rejestrowy pośredni z postdekrementacją

Obliczanie adresu w Real i Virtual 8086 Mode



adres efektywny := rejestr bazowy + rejestr indeksowy + stała

adres efektywny := rejestr bazowy + stała

adres efektywny := rejestr indeksowy + stała

adres efektywny := stała

rejestr bazowy := BX | BP

rejestr indeksowy := SI | DI

Rejestry segmentowe w Real i Virtual 8086 Mode

- CS – segment kodu
- DS – segment danych
- ES – dodatkowy segment danych
- SS – segment stosu
- Rozmiar segmentu to $2^{16} = 65536$ bajtów
- Powszechna konwencja programowa: DS = SS
- Programowanie dawało „wiele radości” – sześć modeli pamięci: tiny, small, medium, compact, large, huge

Wybór rejestru segmentowego

- Sterowanie (np. skoki i wołanie procedur) używa domyślnie segmentu kodu CS.
- Operacje na stosie używają domyślnie segmentu stosu SS.
- Adresowanie z użyciem rejestru bazowego BP używa domyślnie segmentu stosu SS.
- Operacje na ciągach bajtów używają domyślnie jako źródła segmentu danych DS, a jako celu dodatkowego segmentu danych ES.
- Pozostałe instrukcje używają domyślnie segmentu danych DS.
- Użyty segment może być zmodyfikowany przez dodanie do instrukcji prefiksu zmiany segmentu.

Translacja adresu w 32-bit Protected Mode

- Adres efektywny to 32-bitowe przemieszczenie w obrębie segmentu

  adres efektywny := rej. bazowy + skala * rej. indeksowy + stała

  adres efektywny := stała

  skala := 0 | 1 | 2 | 4 | 8

  rej. bazowy := EAX | ECX | EDX | EBX | ESP | EBP | ESI | EDI

  rej. indeksowy := EAX | ECX | EDX | EBX | EBP | ESI | EDI

- Mechanizm segmentacji zamienia adres efektywny na adres liniowy.

- Zasady wyboru segmentów są analogiczne jak w Real Mode.

- Adres liniowy jest zamieniany na adres fizyczny przez mechanizm stronicowania, który można wyłączyć.

Rejestry segmentowe w 32-bit Protected Mode

- CS – segment kodu
- DS – segment danych
- ES, FS, GS – dodatkowe segmenty danych
- SS – segment stosu
- Maksymalny rozmiar segmentu to 2^{32} bajtów
- Popularna konwencja to płaski model pamięci:
  - są dwa segmenty (kodu i danych) obejmujące całą pamięć liniową (wirtualną),
  - adres bazowy segmentów CS, DS, ES, SS wynosi 0,
  - rejestrów FS i GS nie używa się.

Segmentacja w 32-bit Protected Mode

- Rejestr segmentowy składa się z 16-bitowego widocznego dla programisty selektora i niewidocznego dla programisty deskryptora.

- Selektor zawiera:
  - 13-bitowy indeks w tablicy deskryptorów,
  - 1-bit określający tablicę deskryptorów (0 = GDT, 1 = LDT),
  - 2-bitowy żądany poziom ochrony RPL.

- Podczas ładowania widocznej części rejestru segmentowego selektorem, jego niewidoczna część jest ładowana deskryptorem o danym indeksie z odpowiedniej tablicy deskryptorów.

- Położenie GDT określa deskryptor zapisany w rejestrze GDTR.

- Położenie LDT określa selektor w rejestrze LDTR.

Rodzaje i zawartość deskryptorów (1)

- Każdy deskryptor zawiera
  - poziom ochrony deskryptora DPL
  - bit obecności (present)
- Segment danych
  - adres bazowy i wielkość segmentu
  - prawa dostępu: tylko do odczytu, do odczytu i zapisu
  - rozszerzalność: w górę, w dół (segment stosu)
- Segment kodu
  - adres bazowy i wielkość segmentu
  - prawa dostępu: tylko do wykonywania, do wykonywania i odczytu
  - bit zgodności

Rodzaje i zawartość deskryptorów (2)

- Lokalna tablica deskryptorów (ang. *LDT*)
  - adres bazowy i wielkość tablicy
  - Deskryptory lokalnych tablic deskryptorów (LDT) są przechowywane w globalnej tablicy deskryptorów (GDT).
- Dostępny segment stanu zadania (ang. *available TSS*)
  - adres bazowy i wielkość segmentu
- Zajęty segment stanu zadania (ang. *busy TSS*)
  - adres bazowy i wielkość segmentu

Rodzaje i zawartość deskryptorów (3)

- Furtka zadania (ang. *task gate*)
  - selektor segmentu stanu zadania
  - Selektor bieżącego zadania przechowywany jest w widocznej części rejestru TR, a odpowiedni deskryptor w jego niewidocznej części.
- Furtka wywołania (ang. *call gate*)
  - selektor segmentu kodu
  - przemieszczenie – adres początku kodu w segmencie kodu
  - liczba słów, które trzeba przekopiować na nowy stos

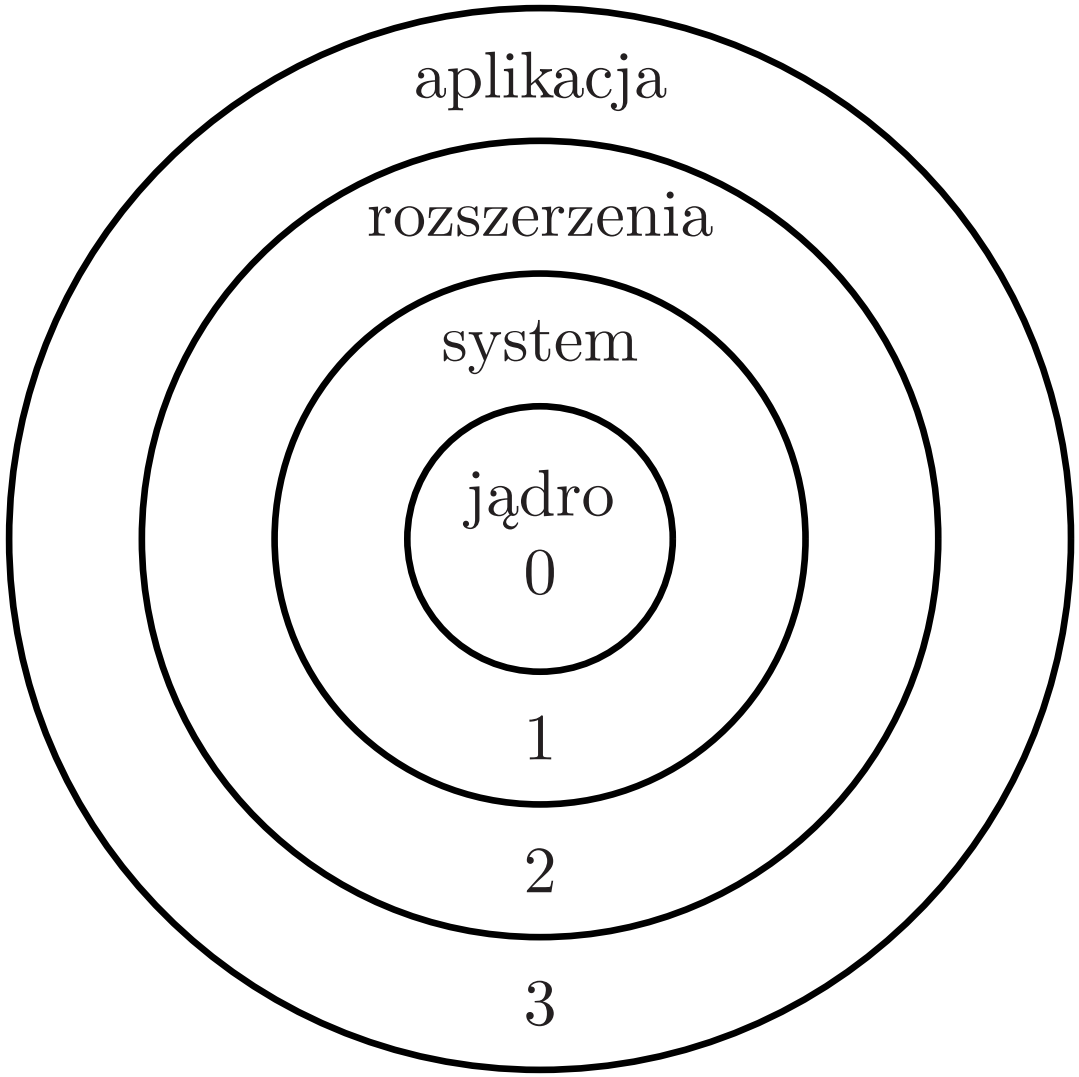
Rodzaje i zawartość deskryptorów (4)

- Furtka przerwania (ang. *interrupt gate*)
  - selektor segmentu kodu
  - przemieszczenie – adres początku kodu w segmencie kodu
- Furtka potrzasku (ang. *trap gate*)
  - selektor segmentu kodu
  - przemieszczenie – adres początku kodu w segmencie kodu
- Deskryptory przerwań i potrzasków są umieszczane w tablicy przerwań, której położenie określa deskryptor w rejestrze IDTR.
- Tablica przerwań może też zawierać deskryptory furtki zadania.

Wykonywane sprawdzenia

- Czy selektor wskazuje na istniejący deskryptor (bit obecności)?
- Czy odwołanie jest do właściwego rodzaju deskryptora?
- Czy zachodzą właściwe nierówności między bieżącym poziomem ochrony (CPL), żądanym poziomem ochrony (RPL) i poziomem ochrony deskryptora (DPL)?
- Czy odwołanie nie przekracza wielkości segmentu?
- Naruszenie któregoś z warunków, powoduje zgłoszenie wyjątku.

Poziomy ochrony



Ochrona dostępu do danych

- CPL – bieżący poziom ochrony, pamiętany w dwóch najmłodszych bitach rejestru CS
- RPL – żądany poziom ochrony, dwa najmłodsze bity selektora użytego do wyspecyfikowania segmentu zawierającego operand, dla segmentu stosu RPL = CPL
- DPL – poziom ochrony deskryptora segmentu zawierającego operand
- Dostęp możliwy, gdy

$$\mathrm{DPL} \geqslant \max\{\mathrm{CPL}, \mathrm{RPL}\}$$

Przekazywanie sterowania

- Skok krótki (short), zmiana wskaźnika instrukcji IP/EIP/RIP w zakresie od -128 do $+127$ bajtów
- Skok i wołanie procedury bliskie (near), w dowolne miejsce bieżącego segmentu kodu
- Skok i wołanie procedury dalekie (far), do innego segmentu kodu

Przekazywanie sterowania do innego segmentu

- Bez zmiany poziomu ochrony, przez selektor wskazujący na deskryptor segmentu kodu

$$\mathrm{DPL} = \mathrm{CPL}$$

lub

segment docelowy zgodny i $\mathrm{DPL} \leqslant \mathrm{CPL}$

- Z ewentualną zmianą poziomu ochrony, przez furtkę

$$\max\{\mathrm{CPL}, \mathrm{RPL}\} \leqslant \mathrm{DPL}\ \text{furtki}$$

i

DPL docelowego segmentu kodu $\leqslant \mathrm{CPL}$

Zmiana poziomu ochrony

- Zadanie może zmieniać poziom ochrony.
- Dla każdego poziomu ochrony jest używany osobny stos.
- Przy zmianie poziomu ochrony jest przełączany stos.
- Powrót z procedury możliwy jest tylko w kierunku bardziej zewnętrznego poziomu ochrony.

Szybka zmiana poziomu ochrony

- Szybkie wołanie usług systemu operacyjnego
- Zmiana między poziomami ochrony 3 a 0
- Para instrukcji SYSENTER, SYSEXIT
- Para instrukcji SYSCALL, SYSRET

Segmentacja w 64-bit Long Mode

- CS – Używany tylko dla określenia poziomu ochrony kodu.
- DS, ES, SS – Zawartość jest ignorowana.
- FS, GS – Używane tylko dla określenia adresu bazowego segmentu.
- Rejestry FS i GS pozostawiono, bo są używane w Windows.
- Adres bazowy segmentów CS, DS, ES, SS wynosi 0.
- Nie jest sprawdzany limit.
- Deskryptory segmentów i furtek są okrojone i służą do implementacji mechanizmów ochrony, przełączania zadań i są potrzebne dla zachowania wstecznej kompatybilności.

Stronicowanie w Protected Mode

adres liniowy	adres fizyczny	liczba poziomów tablic stron (liczby bitów)	rozmiar strony (liczba bitów)
32	32	2 (10, 10)	4 KiB (12)
32	32	1 (10)	4 MiB (22)
32	40	1 (10)	4 MiB (22)
32	52	3 (2, 9, 9)	4 KiB (12)
32	52	2 (2, 9)	2 MiB (21)

- ▷ Podany rozmiar adresu fizycznego jest maksymalnym przewidzianym dla danego trybu stronicowania.
- ▷ Poszczególne modele procesorów implementują krótsze adresy fizyczne: 32 bity, 36 bitów, 40 bitów, ...

Stronicowanie w Long Mode

adres liniowy	adres fizyczny	liczba poziomów tablic stron (liczby bitów)	rozmiar strony (liczba bitów)
48	40, 52	4 (9, 9, 9, 9)	4 KiB (12)
48	40, 52	3 (9, 9, 9)	2 MiB (21)

▷ Adres liniowy jest rozszerzany do 64 bitów przez powielenie bitu znaku.

Stronicowanie

- Rejestr CR3 – adres fizyczny tablicy stron pierwszego poziomu
- Atrybuty strony
  - read/write
  - read-only
- Poziomy ochrony
  - nadzorcy, supervisor level, CPL = 0, 1, 2
  - użytkownika, user level, CPL = 3
- Możliwość ignorowania atrybutu read-only na poziomie nadzorcy
- Bit NX – no execute

Identyfikacja procesora (1)

- Instrukcja CPUID
- Identyfikacja producenta
- Wersja procesora family.model.stepping
- Zaimplementowane instrukcje
- Informacje o pamięci podręcznej
- Różne dziwne informacje, np. jaki sposób ładowania rejestru XMM z pamięci działa efektywniej MOVUPS czy MOVLPS/MOVHPS.

Identyfikacja procesora (2)


```
cat /proc/cpuinfo

vendor_id       : GenuineIntel
cpu family      : 6
model           : 15
model name      : Intel(R) Core(TM)2 CPU  6400  @ 2.13GHz
stepping        : 2
flags           : fpu vme de pse tsc msr pae mce cx8 apic
                  sep mtrr pge mca cmov pat pse36 clflush
                  dts acpi mmx fxsr sse sse2 ss ht tm
                  syscall nx lm constant_tsc pni monitor
                  ds_cpl vmx est tm2 cx16 xtpr lahf_lm
cache size      : 2048 KB
cache_alignment : 64
address sizes   : 36 bits physical, 48 bits virtual
```