Tryb chroniony cz. 1

Zarządzanie pamięcią

Moduł **zarządzania pamięcią** w trybie chronionym (z ang. PM - Protected Mode) procesorów IA-32 udostępnia:
- segmentację,
- stronicowanie.

Segmentacja – mechanizm umożliwiający odizolowanie obszarów kodu, danych i stosu należących do różnych programów. W rezultacie żaden program nie może ingerować w obszar innego programu.

Stronicowanie – wsparcie dla systemów pamięci wirtualnej, gdzie fragmenty programów są ładowane do pamięci fizycznej tylko wtedy, gdy są potrzebne. W przeciwnym wypadku pozostają na dysku twardym. Stronicowanie jest opcjonalne. Włączamy/wyłączamy je odpowiednio ustawiając/zerując bit 31 w rejestrze kontrolnym CR0. (patrz Rys. 2.2).

```
mov eax, cr0
or al, 01h
mov cr0, eax
```
Włącz PM

```
mov eax, cr0
and al, 0feh
mov cr0, eax
```
Wyłącz PM

Rys. 2.1 Fragment kodu włączający/wyłączający PM

```
mov eax, cr0
or eax, 80000000h
mov cr0, eax
```
Włącz stronicowanie

```
mov eax, cr0
and eax, 7fffffffh
mov cr0, eax
```
Wyłącz stronicowanie

Rys. 2.2 Fragment kodu włączający/wyłączający stronicowanie

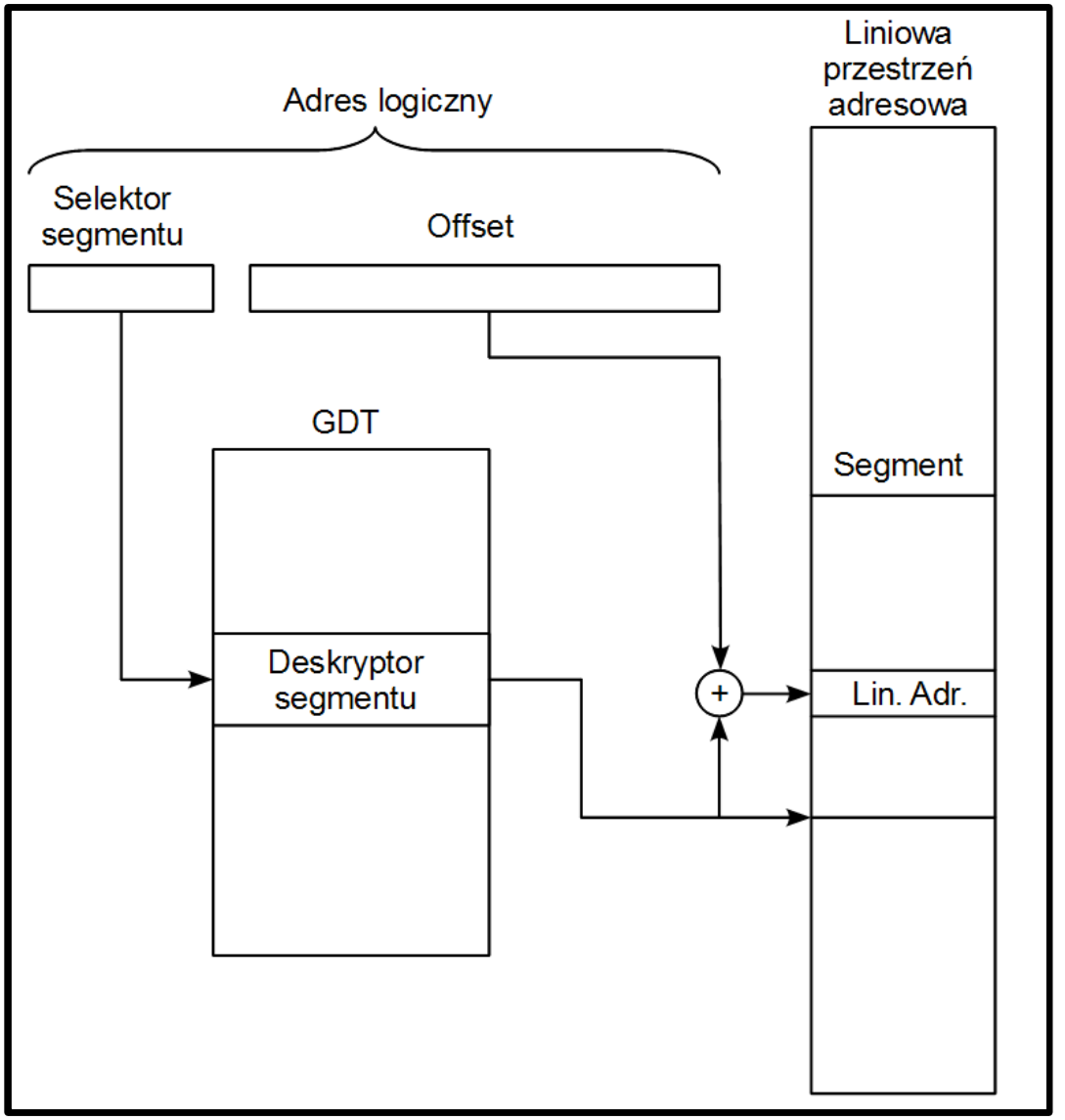
Z kolei za pomocą bitu 0 rejestru CR0 włączamy/wyłączamy tryb chroniony (patrz Rys. 2.1).

Segmentacja

Segmentacja dostarcza mechanizm podziału przestrzeni adresowej procesora (zwanej **liniową przestrzenią adresową**) na chronione obszary zwane **segmentami**. Segmenty mogą zawierać: kod, dane i stos programu, a także wszelkie struktury systemowe.

Segmenty pamięci mają przypisane pewne atrybuty: poziom **uprzywilejowania**, **adres bazowy**, **limit** oraz **przeznaczenie**. Informacje te zapisane są w tzw. **deskryptorze segmentu**.

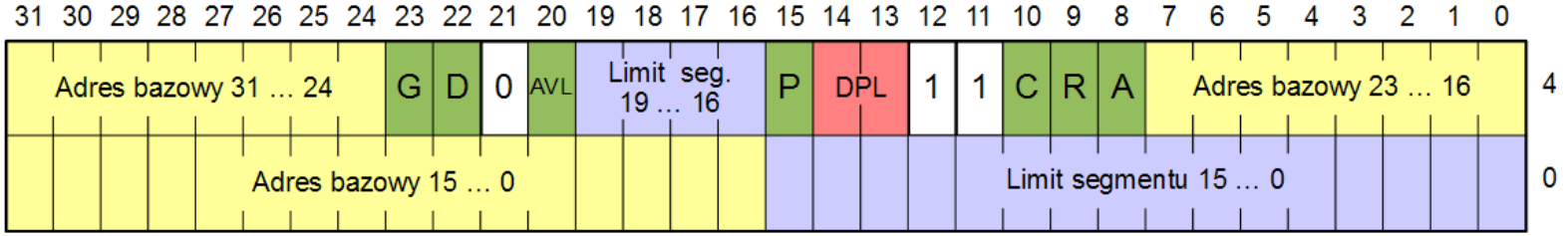
Deskryptory przechowywane są w tablicach deskryptorów: **globalnej** (GDT) i **lokalnych** (LDT). Ich położenie identyfikowane jest przez **selektor segmentu**.



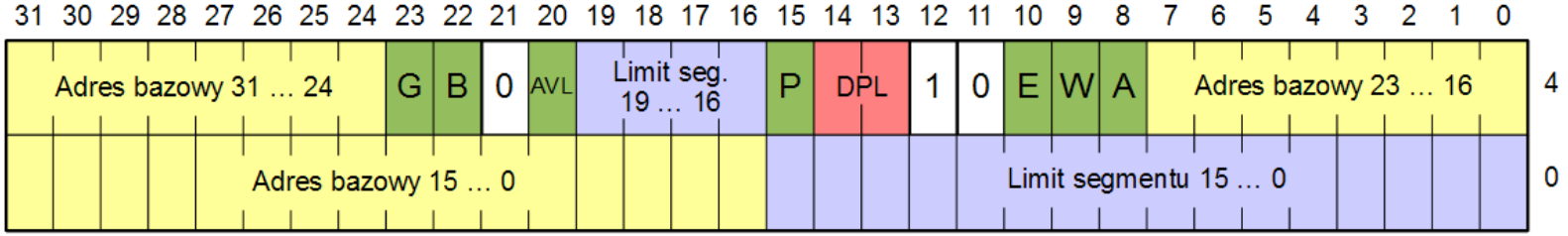
Rys. 2.3 Segmentacja

Format deskryptora segmentu

Deskryptor segmentu to struktura 8 bajtowa zawierająca: adres bazowy (32-bity), limit (20-bitów) i poziom uprzywilejowania (patrz Rys. 2.4 i 2.5).



Rys. 2.4 32-bit deskryptor segmentu kodu



Rys. 2.5 32-bit deskryptor segmentu danych/stosu

Opis pól deskryptora:

1. **Adres bazowy segmentu** – 32 bitowa wartość będąca adresem bazowym segmentu w liniowej przestrzeni adresowej.

2. **Limit segmentu** – 20-bitowe pole zawierające największą dozwoloną wartość offsetu w ramach segmentu. Limit może być liczony zgodnie z granulacją 1 bajta (G=0) lub 4kB (G=1).

3. **G** (Granularity) – wskazuje granulację dla limitu segmentu:jeden bajt (G=0) lub 4kB (G=1). Z granulacją 1B offset zmienia się z rozdzielczością 1B. Stąd maksymalny rozmiar segmentu to 1MB. Z granulacją 4kB jednostkowa zmiana offsetu zmienia adres liniowy o 4kB. Zatem w tym przypadku maksymalny rozmiar segmentu to 4GB.

4. **D** (Default) – wskazuje, czy segment jest 32-bitowy (D=1), czy też 16-bitowy (D=0).

5. **E** (Expansion Direction) – wykorzystywany z segmentami danych, wskazuje, czy segment rozciąga się od adresu bazowego do adresu bazowego+limit (E=0), czy od maksymalnego offsetu do limitu (E=1). Stos jest często segmentem danych typu *expand-down* (E=1), co umożliwia dynamiczną zmianę rozmiaru stosu.

6. **B** (Big) – dla segmentów danych, wskazuje maksymalną wartość offsetu jako 0ffffffffh (B=1) oraz 0000ffffh (B=0). Bit ten ma znaczenie w przypadku segmentów typu *expand-down*.

7. Prawa dostępu:

   a. **P** (Present) – segment jest obecny w pamięci fizycznej (P=1) lub nie (P=0). Do wykorzystania przez menedżery pamięci wirtualnej.

   b. **A** (Accessed) – wskazuje na to, czy do danego segment odwoływano się (A=1) od czasu, kiedy bit A został wyzerowany. Dla menedżerów pamięci wirtualnej bit ten może posłużyć do tworzenia statystyki korzystania z segmentów.

c. **DPL** (Descriptor Privilege Level) – określa poziom uprzywilejowania segmentu jako liczbę 0, 1, 2 lub 3.

DPL	Opis
00	Poziom 0 najbardziej uprzywilejowany (jądro systemu)
01	Poziom 1
10	Poziom 2
11	Poziom 3 najmniej uprzywilejowany (aplikacja użytkownika)

Tab. 2.1 Poziomy uprzywilejowania

DPL dla aktualnego segmentu kodu określa wartość Current Privilege Level (**CPL**).

d. **R** (Readable) – w przypadku segmentu kody wskazuje, czy do dany segment można czytać (R=1), czy też nie (R=0). Segmenty kodu są zawsze wykonywalne.

e. **C** (Conforming) – dla segmentów kodu, określa, czy CPL ulega zmianie, gdy segment jest wywołany z kodu o niższym poziomie uprzywilejowania (C=0), czy też pozostaje niezmieniony (C=1).

f. **W** (Writable) – w przypadku segmentów danych określa przyzwolenie na zapis (W=1) lub jego brak (W=0). Segmenty danych są zawsze możliwe do odczytu.

g. **AVL** – do wykorzystania przez system operacyjny.

Przykłady deskryptorów

Segment kodu

Adres bazowy w trybie rzeczywistym: 4b10h:0000h
Rozmiar segmentu: 1000h B
Prawa dostępu: readable, conforming, present, granularity 1 B, 32-bity, poziom jądra systemu

Liniowy (32-bity) adres bazowy: 16*4b10h+0000h=0004b100h
Limit: 1000h – 1=0fffh

Deskryptor: **dw 0fffh, b100h, 9a04h, 0040h**

Segment danych

Adres bazowy w trybie rzeczywistym: 5cf0h:0000h
Rozmiar segmentu: 4000h B
Prawa dostępu: writable, present, granularity 1 B, 32-bity, poziom użytkownika, expand up

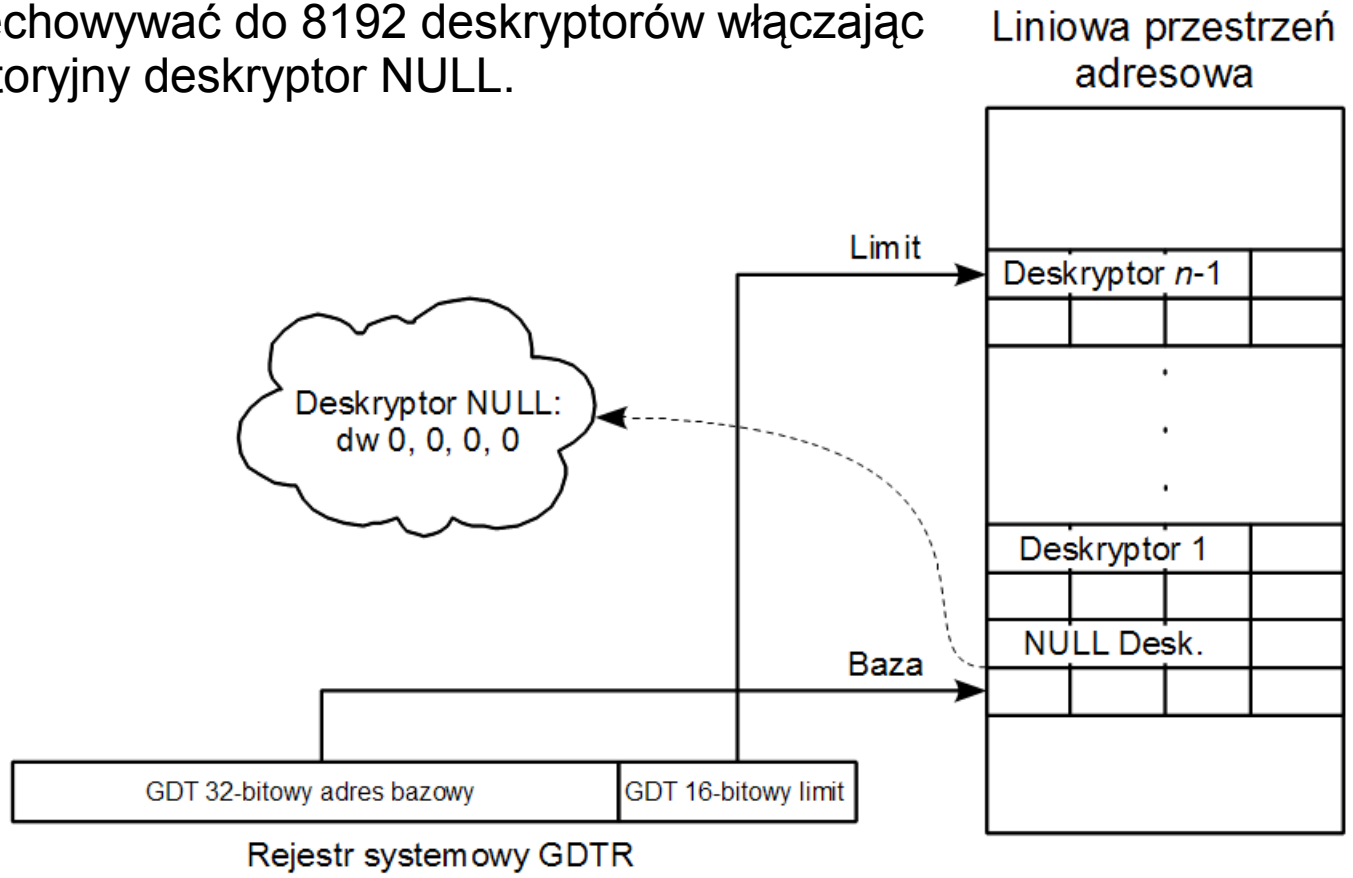
Liniowy (32-bity) adres bazowy: 16*5cf0h+0000h=0005cf00h
Limit: 4000h – 1=3fffh

Deskryptor: **dw 3fffh, 0cf00h, 0f205h, 0040h**

Globalna Tablica Deskryptorów

Globalna Tablica Deskryptorów jest tablicą zawierającą deskryptory. Adres bazowy i limit GDT są zapisane w rejestrze **GDTR**, który można zapisywać za pomocą instrukcji **LGDT**. Istnieje tylko jedna tablica GDT.

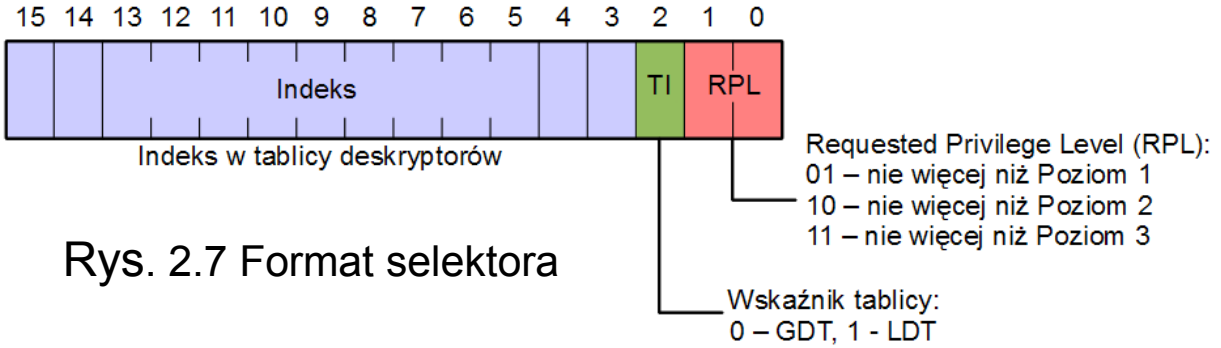
GDT może przechowywać do 8192 deskryptorów włączając pierwszy obligatoryjny deskryptor NULL.



Rys. 2.6 Globalna Tablica Deskryptorów

Selektor

Deskryptory są identyfikowane poprzez 16-bitowe selektory postaci

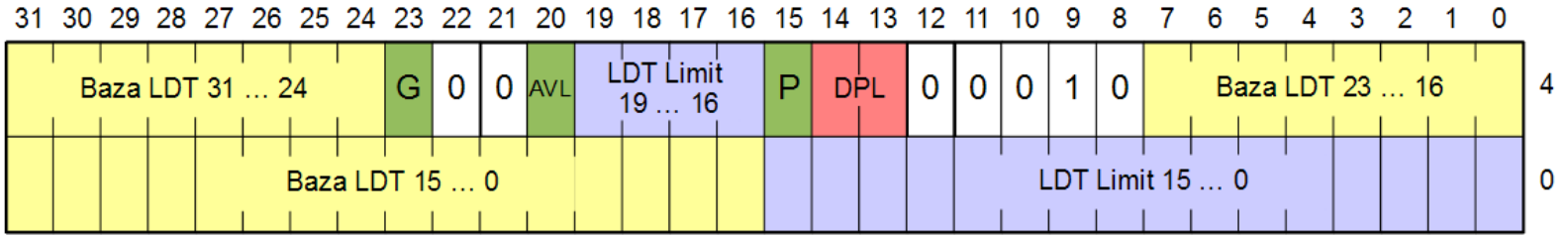


Rys. 2.7 Format selektora

Lokalna Tablica Deskryptorów

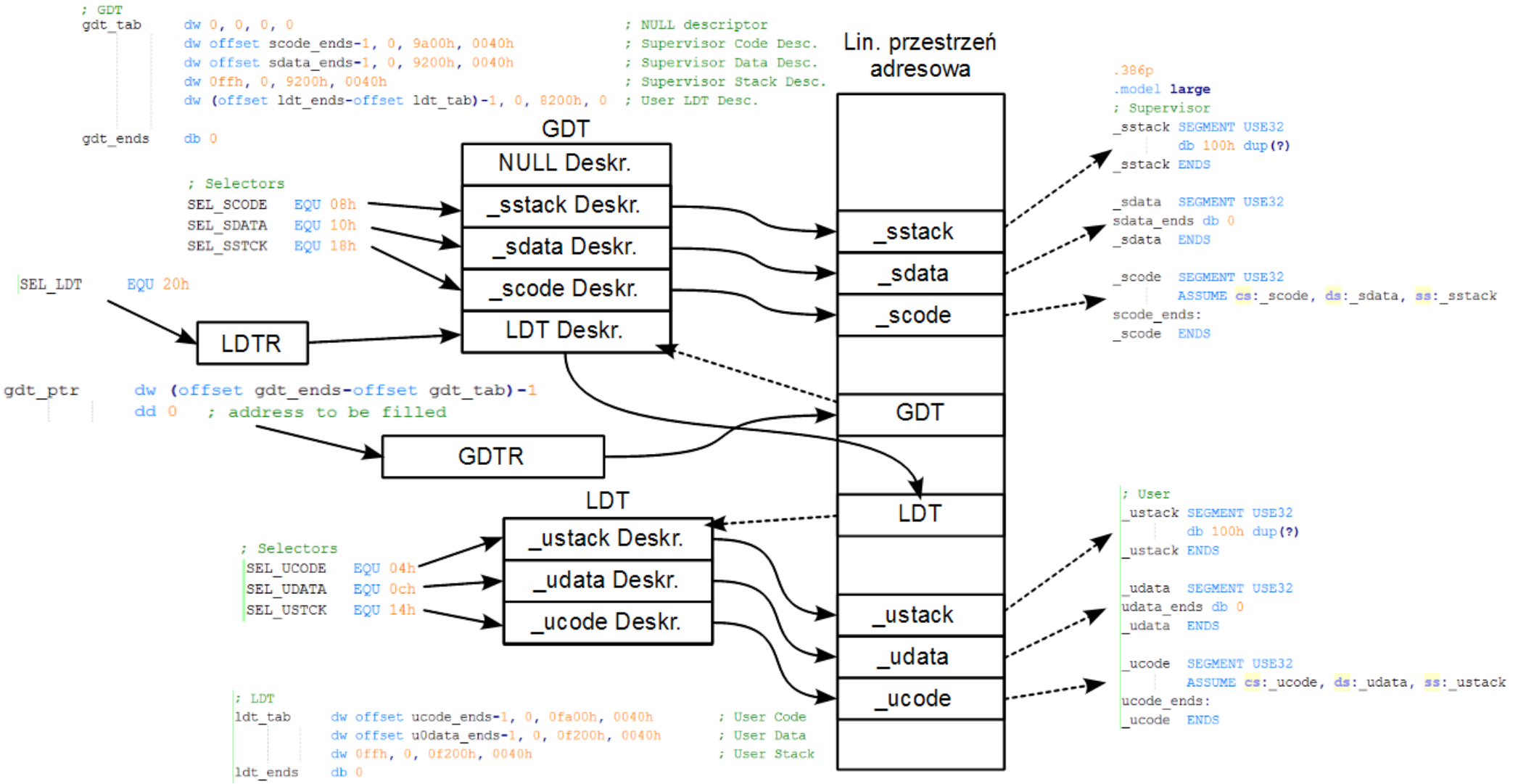
Każde zadanie (ang. *task*) może mieć własną przestrzeń adresową opisaną deskryptorami zawartymi w Lokalnej Tablicy Deskryptorów (**LDT**).

Aktualnie wybrana tablica LDT wskazywana jest (jej obszar musi być opisany deskryptorem w GDT) poprzez selektor załadowany do rejestru **LDTR** instrukcją **LLDT**.



Rys. 2.8 Deskryptor LDT

Przykładowa aplikacja – struktura deskryptorów



Rys. 2.9 Przykładowa struktura deskryptorów dla aplikacji system/użytkownik

Wskaźniki poziomów uprzywilejowania

CPL (Current Privilege Level) – poziom uprzywilejowania kodu aktualnie wykonywanego programu,

DPL (Descriptor Privilege Level) – poziom uprzywilejowania segmentu. Interpretacja DPL zależy od typu segmentu, tzn.:

- segment danych: jest to liczbowo największa wartość poziomu uprzywilejowania jaką może posiadać program, aby uzyskać dostęp do segmentu.

- segment kodu (conforming): wskazuje liczbowo najmniejszą wartość jaką może posiadać program, aby uzyskać dostęp do segmentu.

RPL (Requested Privilege Level) – stanowi nadrzędny wskaźnik dla wskaźnika CPL. Podczas sprawdzania uprawnień procesor bierze pod uwagę RPL i CPL. Jeżeli RPL jest liczbowo większy od CPL, to brany jest RPL i vice versa.

RPL może być użyty dla zapewnienia, iż mniej uprzywilejowany kod nie uzyska dostępu do bardziej uprzywilejowanych segmentów w momencie, kiedy korzystać on będzie z kodu uprzywilejowanego, a selektory do wspomnianych segmentów przekazywane będą jako argumenty wywołania (patrz *Intel IA-32, Intel 64 Manual)*.

Sprawdzanie praw dostępu podczas odwołań do pamięci

Odczyt/zapis do pamięci
- cel musi znajdować się w segmencie do odczytu/zapisu ,
- adres celu musi znajdować się w limicie offsetów dla danego segmentu,
- adresowanie deskryptorem NULL nie jest dozwolone,
- w przypadku przekazywania sterowania cel musi mieścić się w segmencie wykonywalnym (to zagadnienie będzie dokładnie omówione na kolejnym wykładzie),
- podczas operacji odczytu/zapisu RPL i CPL <= DPL.

Ładowanie rejestrów segmentowych selektorami
- indeks selektora musi mieścić się w limicie tablicy deskryptorów,
- w przypadku SS: segment wskazywany przez selektor nie może być NULL, RPL=CPL, DPL=CPL, segment musi być obecny (P=1) i musi być segmentem danych do zapisu,
- dla DS, ES, FS, GS: segment musi być segmentem danych, RPL, CPL <= DPL, segment musi być obecny (P=1).

Naruszenie którejkolwiek z powyższych zasad skutkuje pojawieniem się wyjątku General Protection (GP).

Wymagania dla segmentów danych, stosu i kodu w trybie rzeczywistym

Podczas powrotu do trybu rzeczywistego należy zapewnić selektory segmentów: 16-bitowy, do odczytu, obecny, 1B granulacji, maksymalny limit 0ffffh, wykonywalny, poziom 0 w przypadku kodu (do CS), 16-bitowy, obecny, do zapisu, 1B granulacji, 0ffffh limit, expand up, poziom 0 dla segmentów danych (do DS, ES, FS, GS and SS).