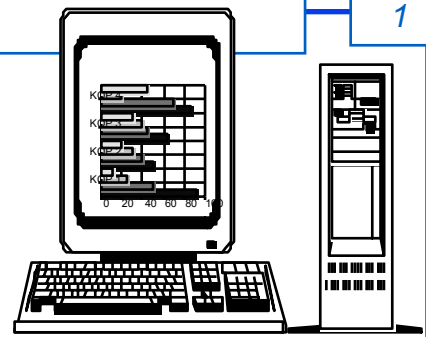
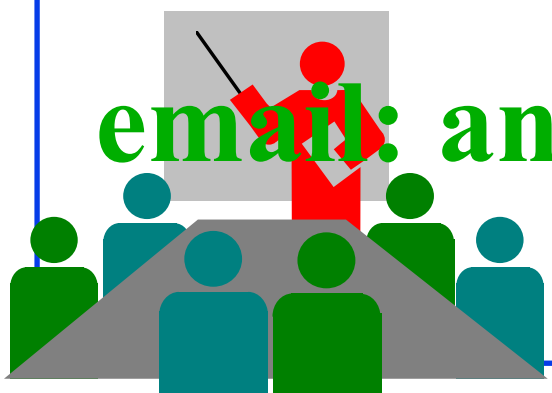


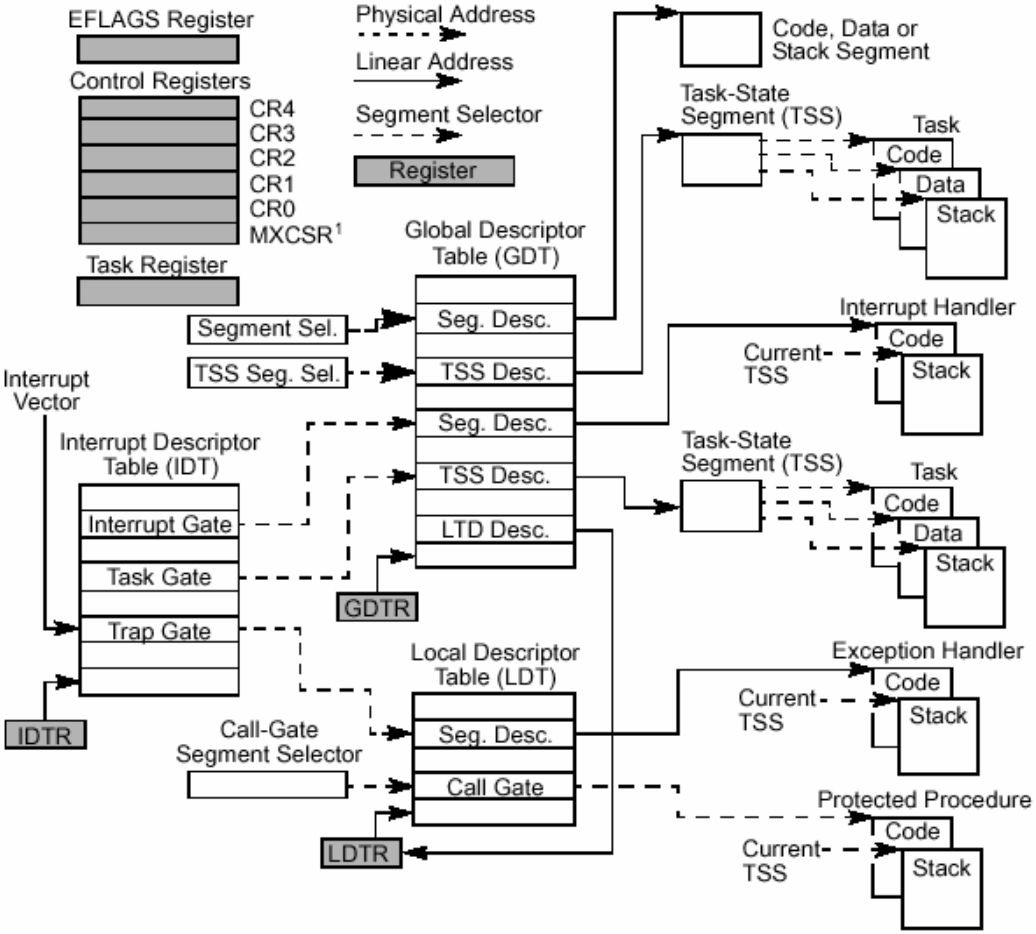
Temat: Tryb chroniony mikroprocesorów x86

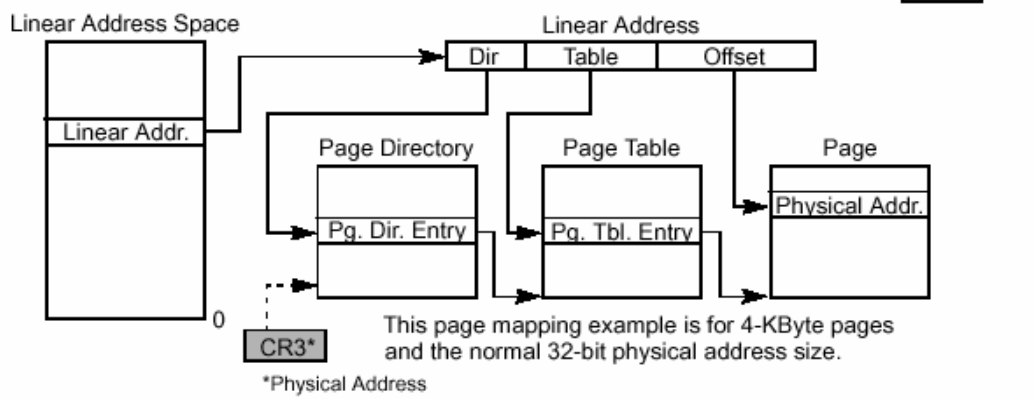
email: andrzej.stasiak@gmail.com



Rejestry mikroprocesora I80386

- Rejestr flag
- Rejestry zarządzania pamięcią
- Rejestry sterujące
- Rejestry uruchomieniowe

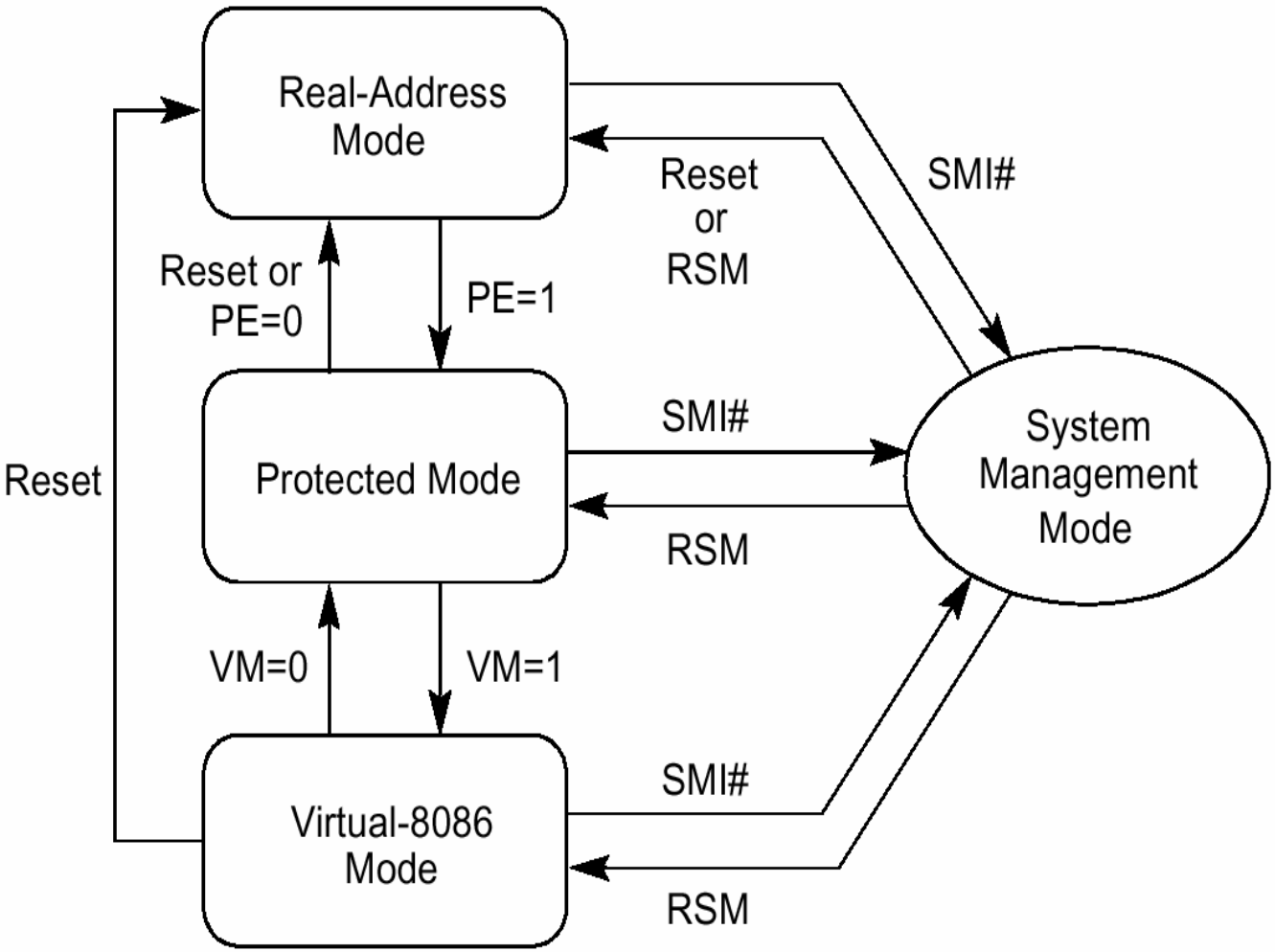




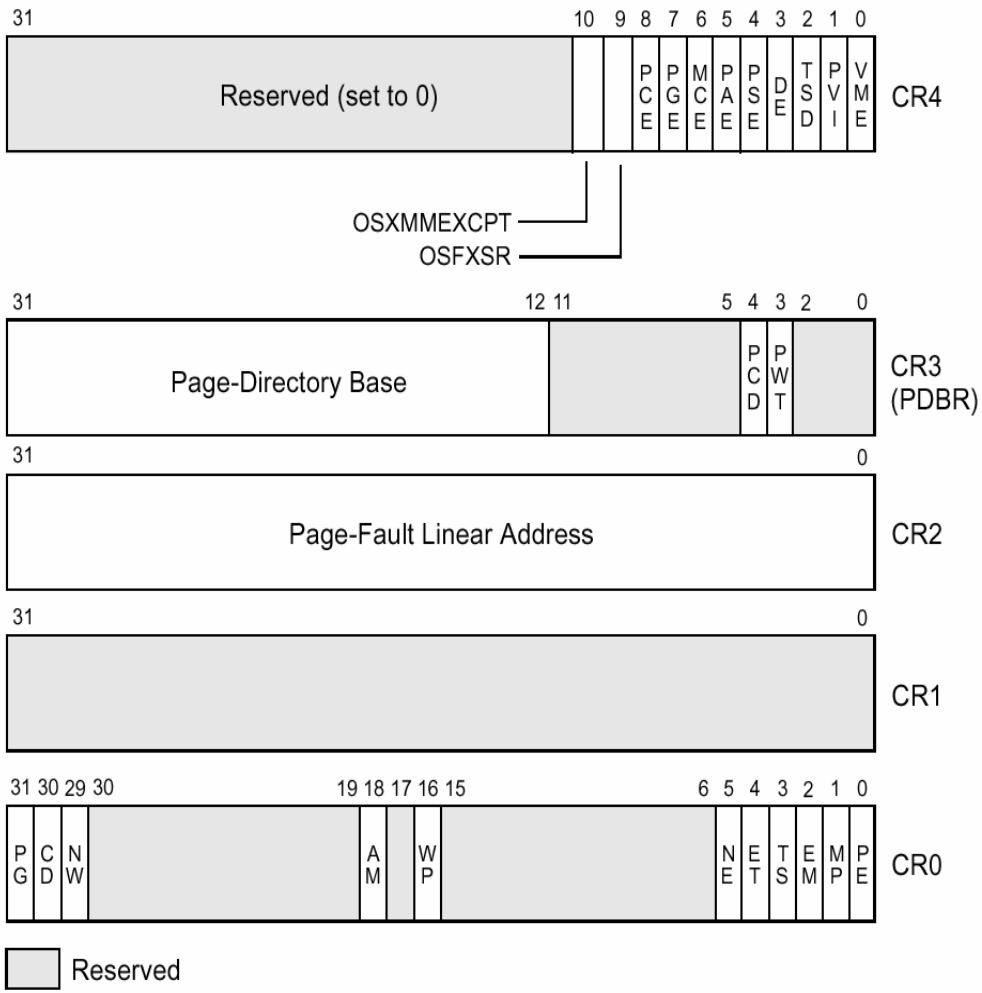
1. MXCSR is new control/status register in the Pentium® III processor.

Poziom architektury procesorów pentium

Model przejść stanów procesorów x86



Rejestry sterujące



Rejestry sterujące

- PG Paging (bit 31 of CR0)
- **CD Cache Disable (bit 30 of CR0)**
- **NW Not Write-through (bit 29 of CR0).**
- **AM Alignment Mask (bit 18 of CR0).**
- **WP Write Protect (bit 16 of CR0).**
- **NE Numeric Error (bit 5 of CR0).**
- **ET Extension Type (bit 4 of CR0).**
- **TS Task Switched (bit 3 of CR0).**
- **EM Emulation (bit 2 of CR0).**
- **MP Monitor Coprocessor (bit 1 of CR0).**
- PE Protection Enable (bit 0 of CR0).

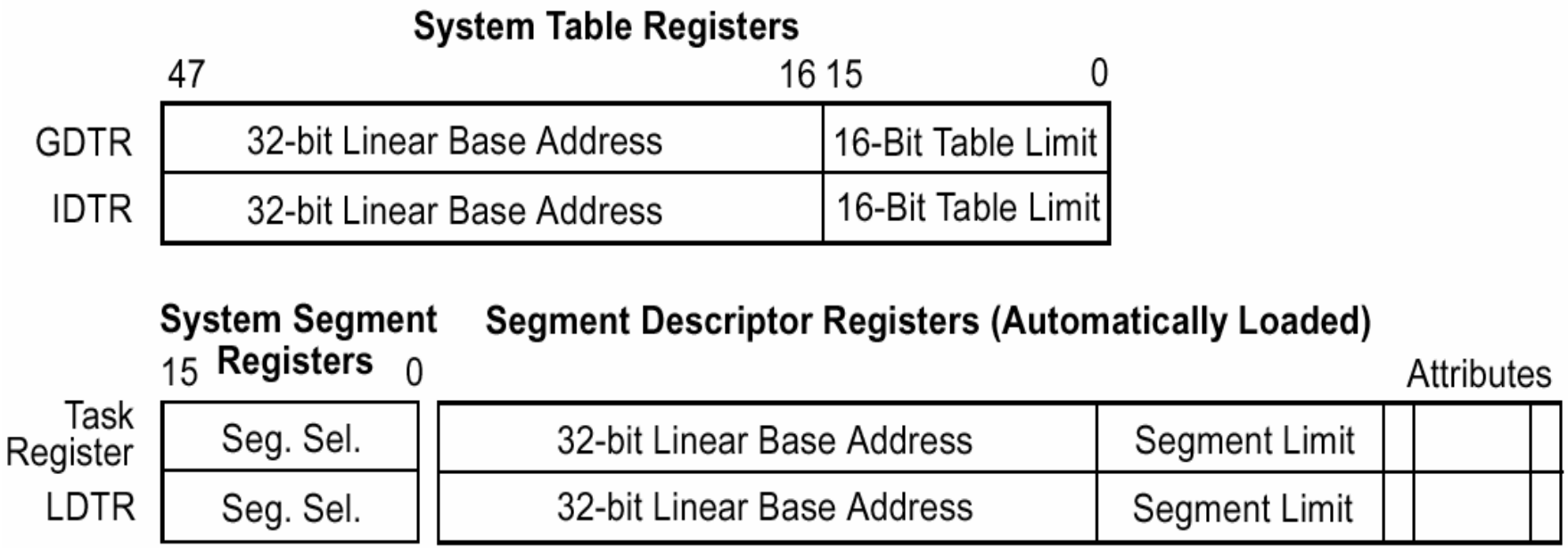
Rejestry sterujące

- **PCD** Page-level Cache Disable (bit 4 of CR3).
- PWT Page-level Writes Transparent (bit 3 of CR3).

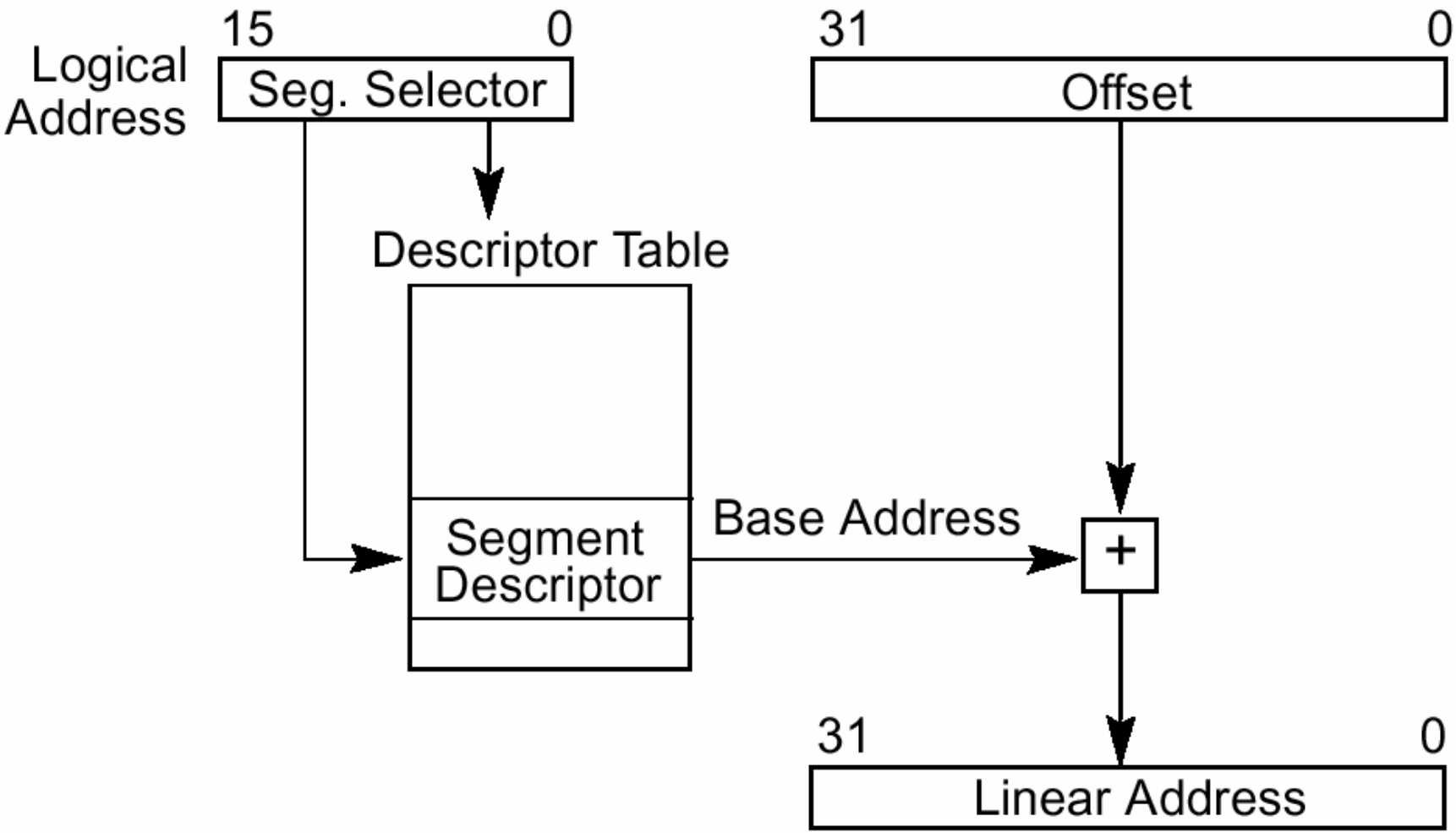
Rejestry sterujące

- **VME** Virtual-8086 Mode Extensions (bit 0 of CR4).
- PVI Protected-Mode Virtual Interrupts (bit 1 of CR4).
- TSD Time Stamp Disable (bit 2 of CR4).
- DE Debugging Extensions (bit 3 of CR4).
- PSE Page Size Extensions (bit 4 of CR4).
- PAE Physical Address Extension (bit 5 of CR4).
- MCE Machine-Check Enable (bit 6 of CR4).
- PGE Page Global Enable (bit 7 of CR4).
- PCE Performance-Monitoring Counter Enable (bit 8 of CR4).
- OSFXSR - Operating Sytsem FXSAVE/FXRSTOR Support (bit 9 of CR4).
- OSXMMEXCPT - Operating System Unmasked Exception Support (bit 10 of CR4).

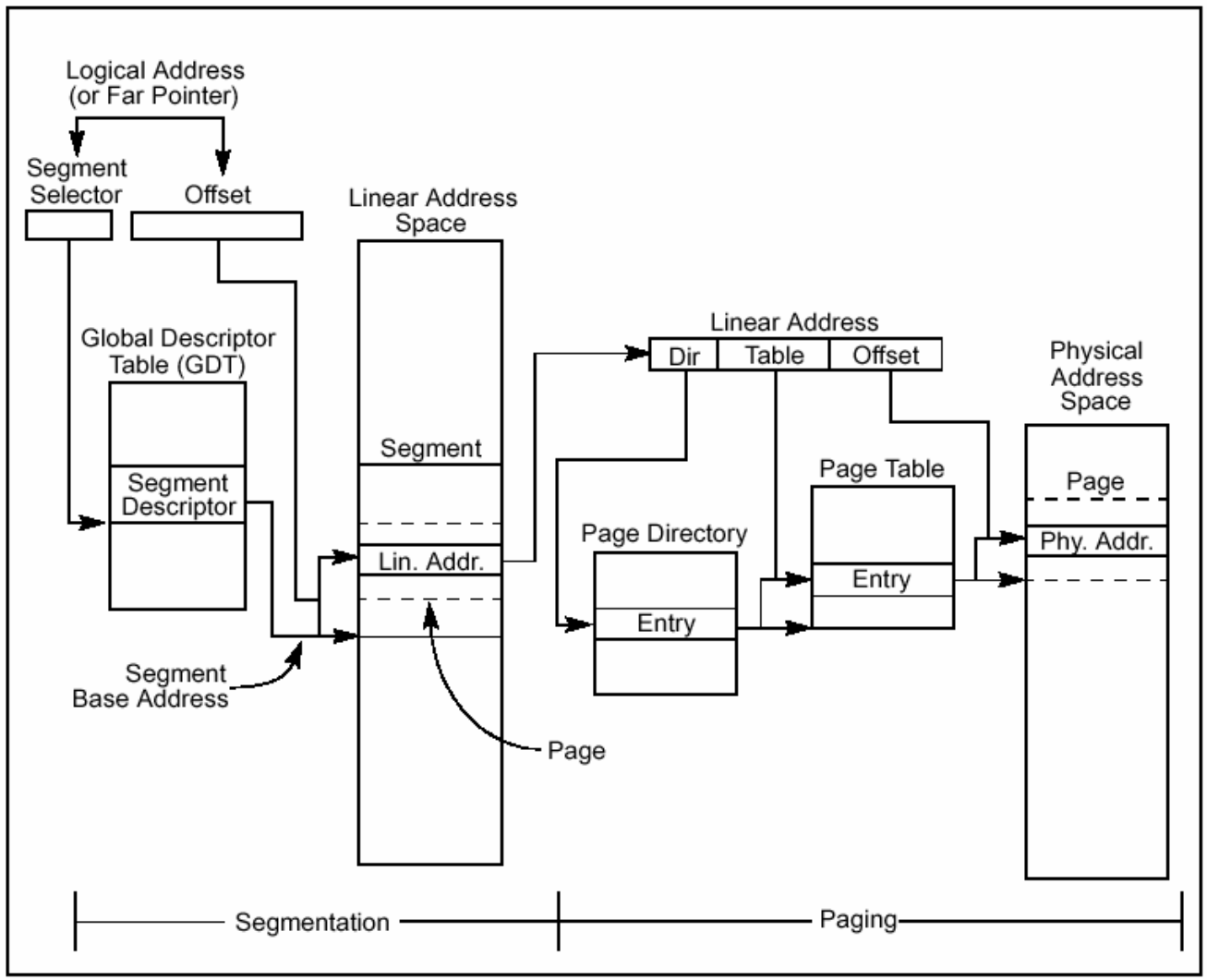
Rejestry zarządzania pamięcią



Translacja adresu logicznego w adres liniowy



Segmentacja i stronicowanie



Struktura rejestrów selektorowych

$2^{13} = 8192$ struktur deskryptorowych

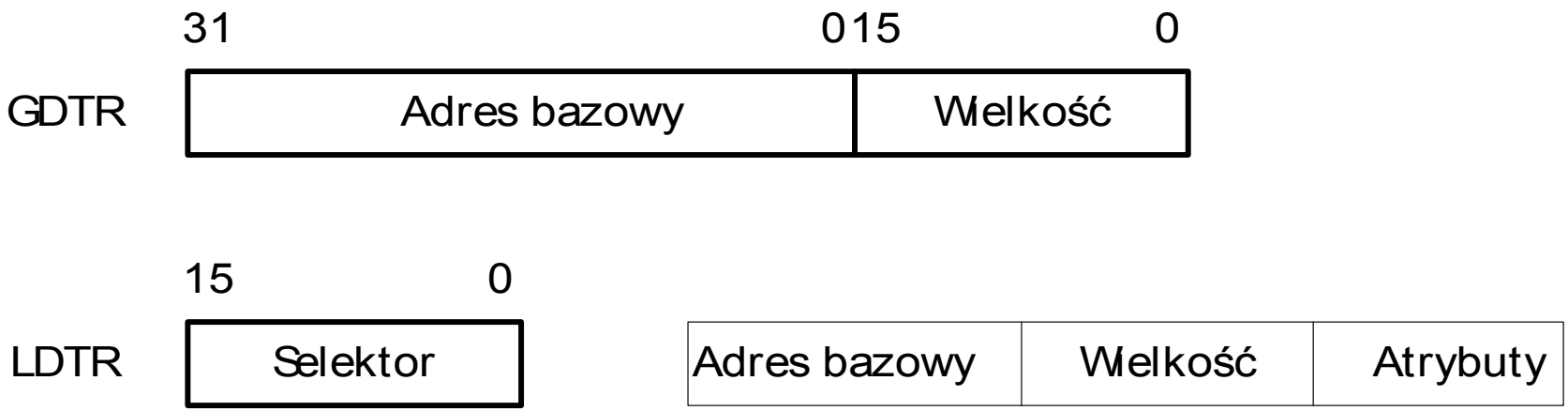
$2^{2} = 4$; poziomy uprzywilejowania

Rejestry selektorowe

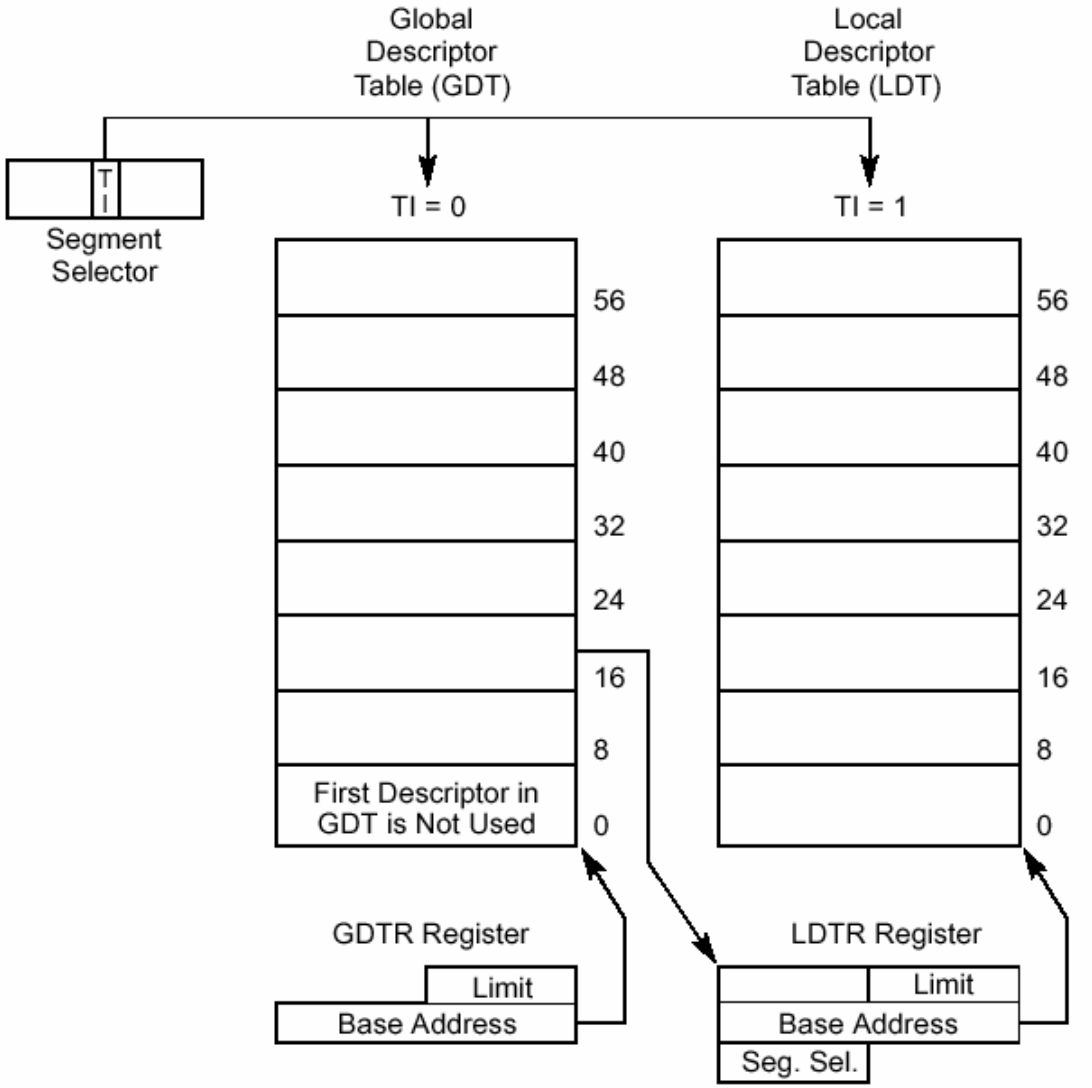
CS
DS
ES
FS
GS
SS

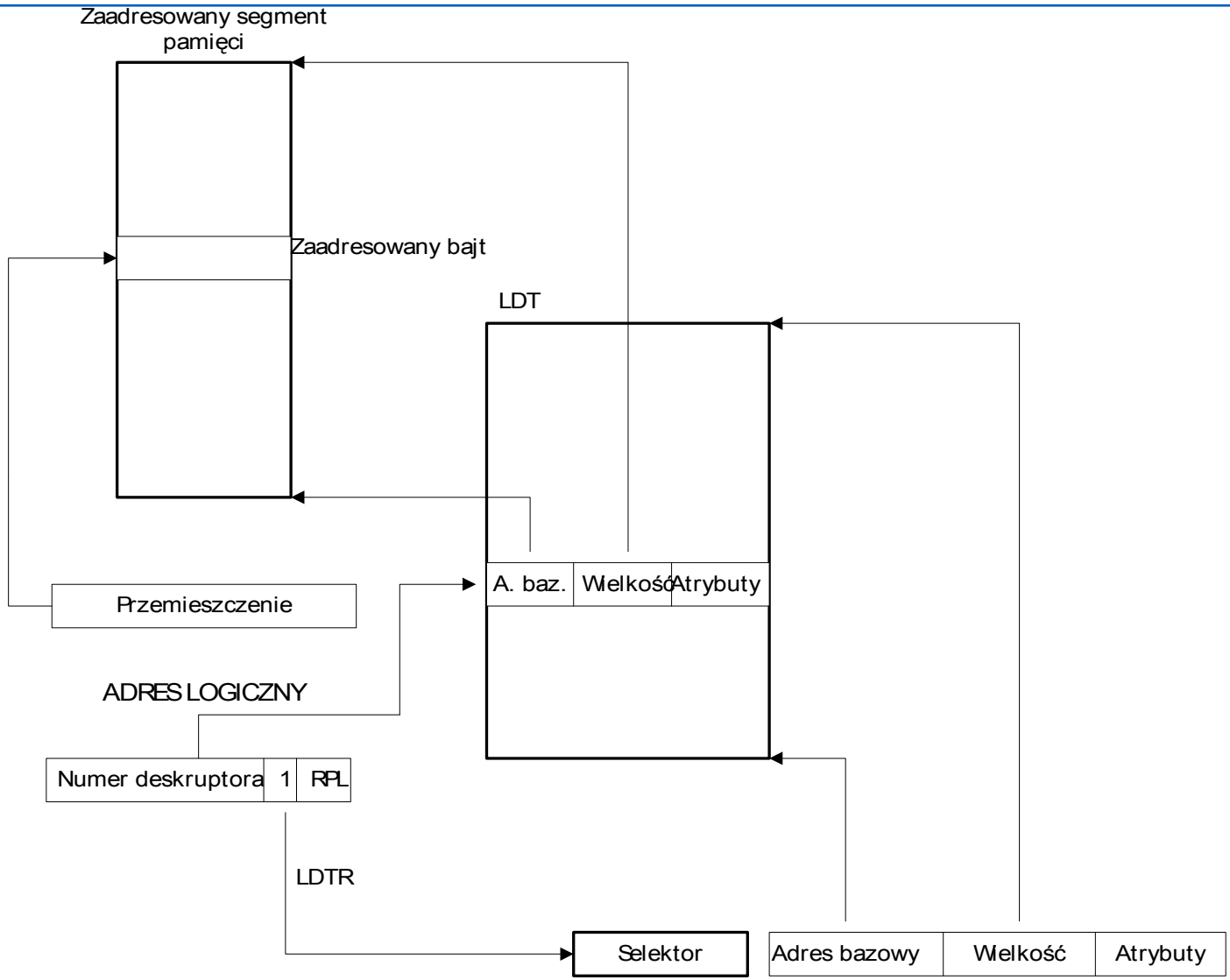
Adres bazowy	Wielkość	Atrybuty
Adres bazowy	Wielkość	Atrybuty
Adres bazowy	Wielkość	Atrybuty
Adres bazowy	Wielkość	Atrybuty
Adres bazowy	Wielkość	Atrybuty
Adres bazowy	Wielkość	Atrybuty

Rejestry tablic globalnych i lokalnych



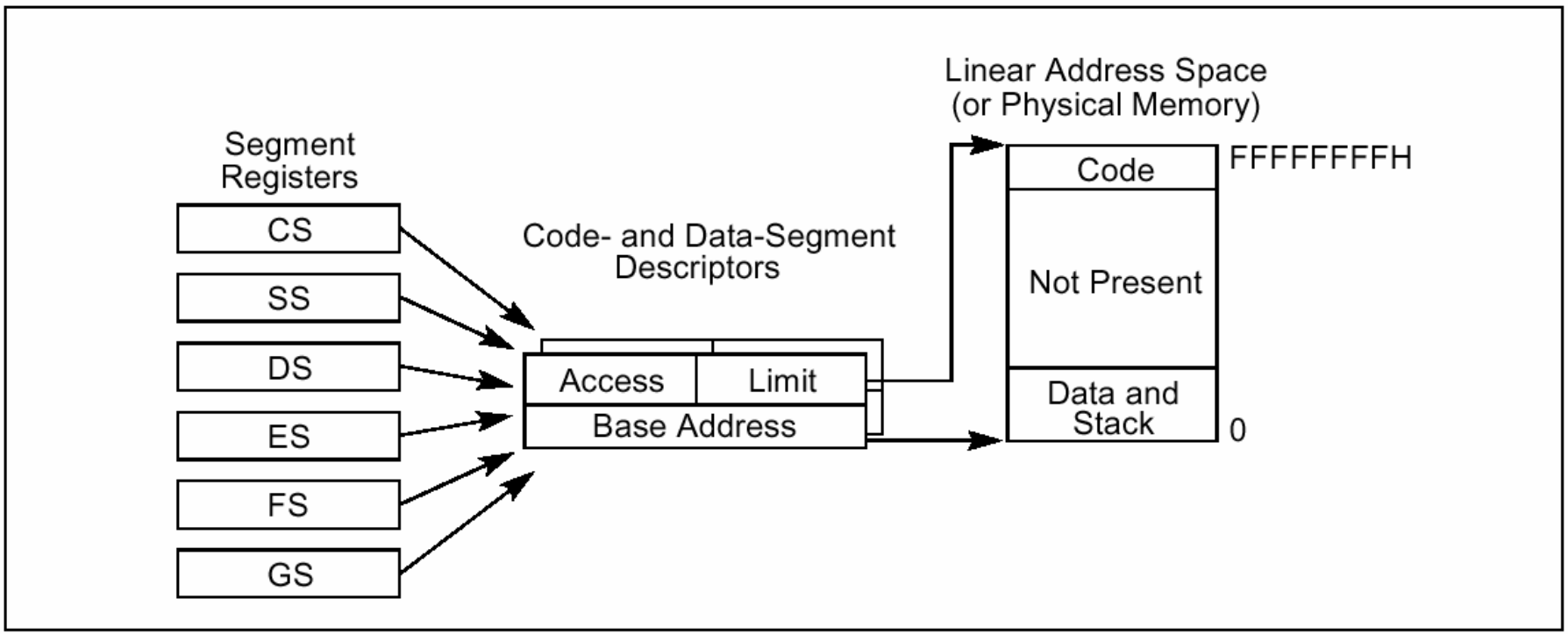
Globalna i lokalne tablice deskryptorów



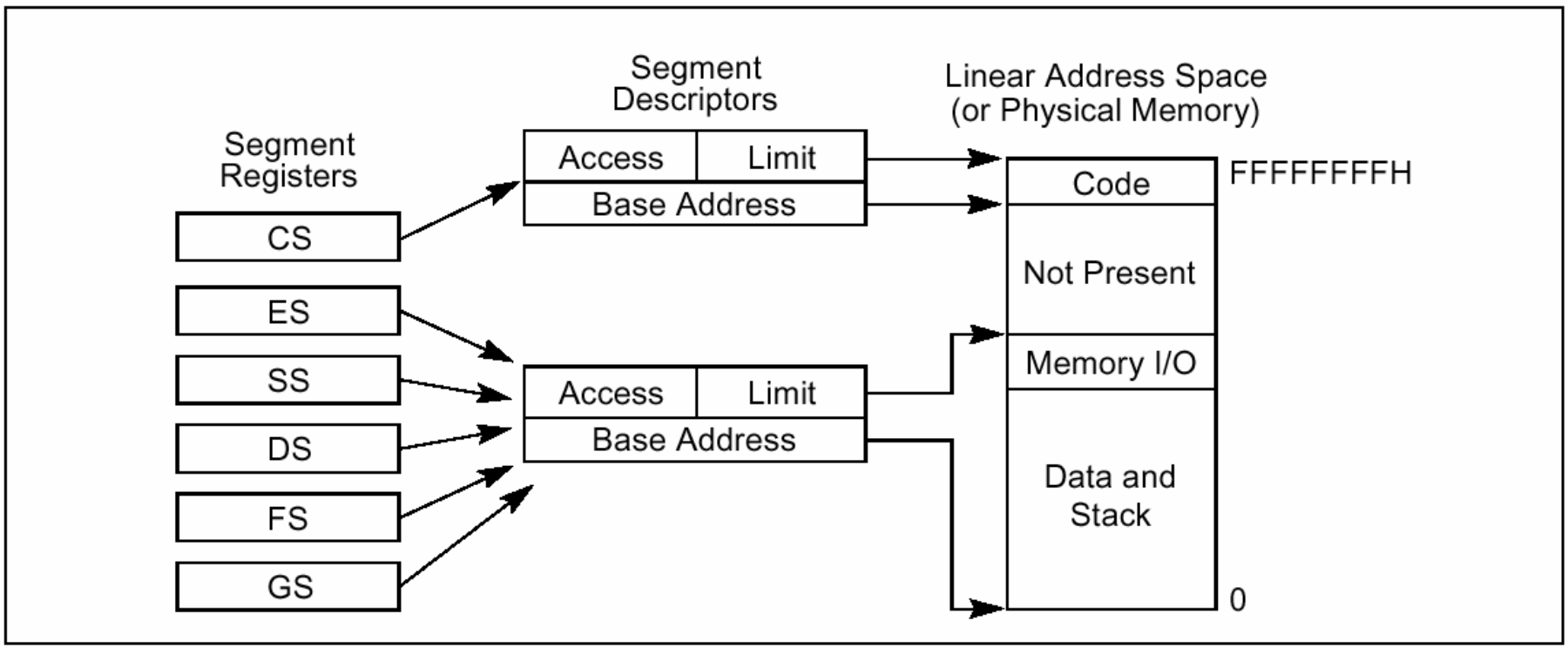


Rejestr lokalnej tablicy deskryptorów (LDTR)

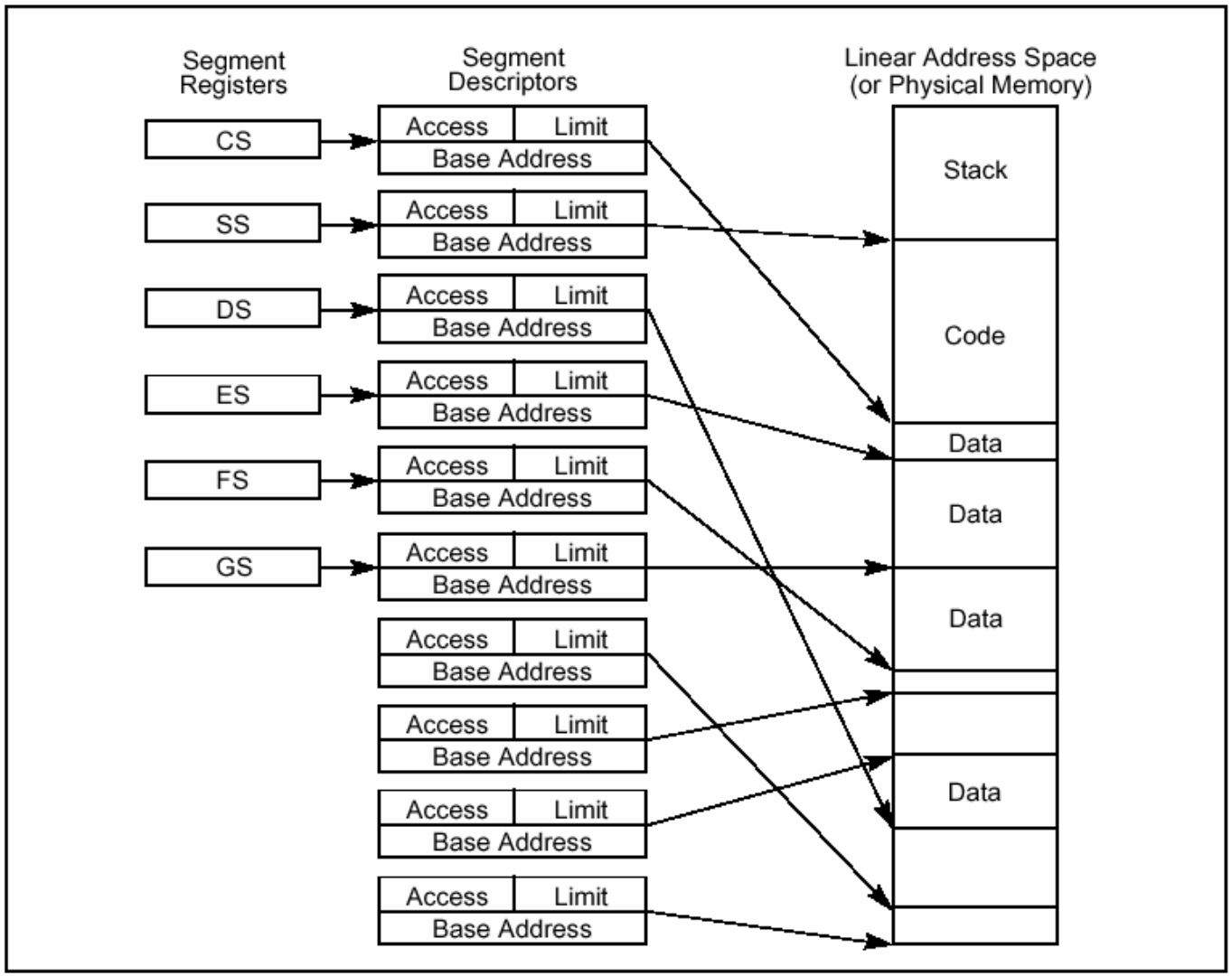
Flat Model


Linear Address Space
(or Physical Memory)
Segment
Registers
CS
SS
DS
ES
FS
GS
Code- and Data-Segment
Descriptors
Access
Limit
Base Address
Code
Not Present
Data and
Stack
FFFFFFFFH
0

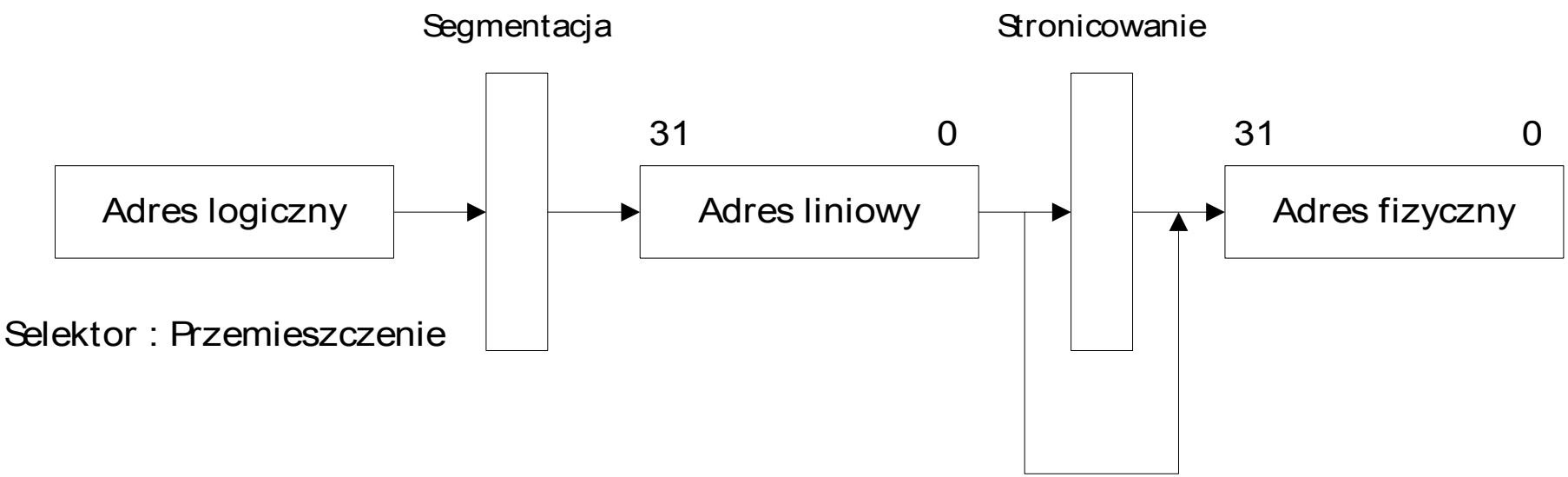
Modele pamięci - *Protected Flat Model*

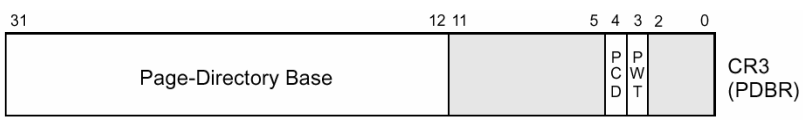


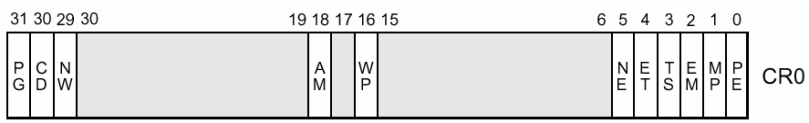
Modele pamięci - Multisegment Model



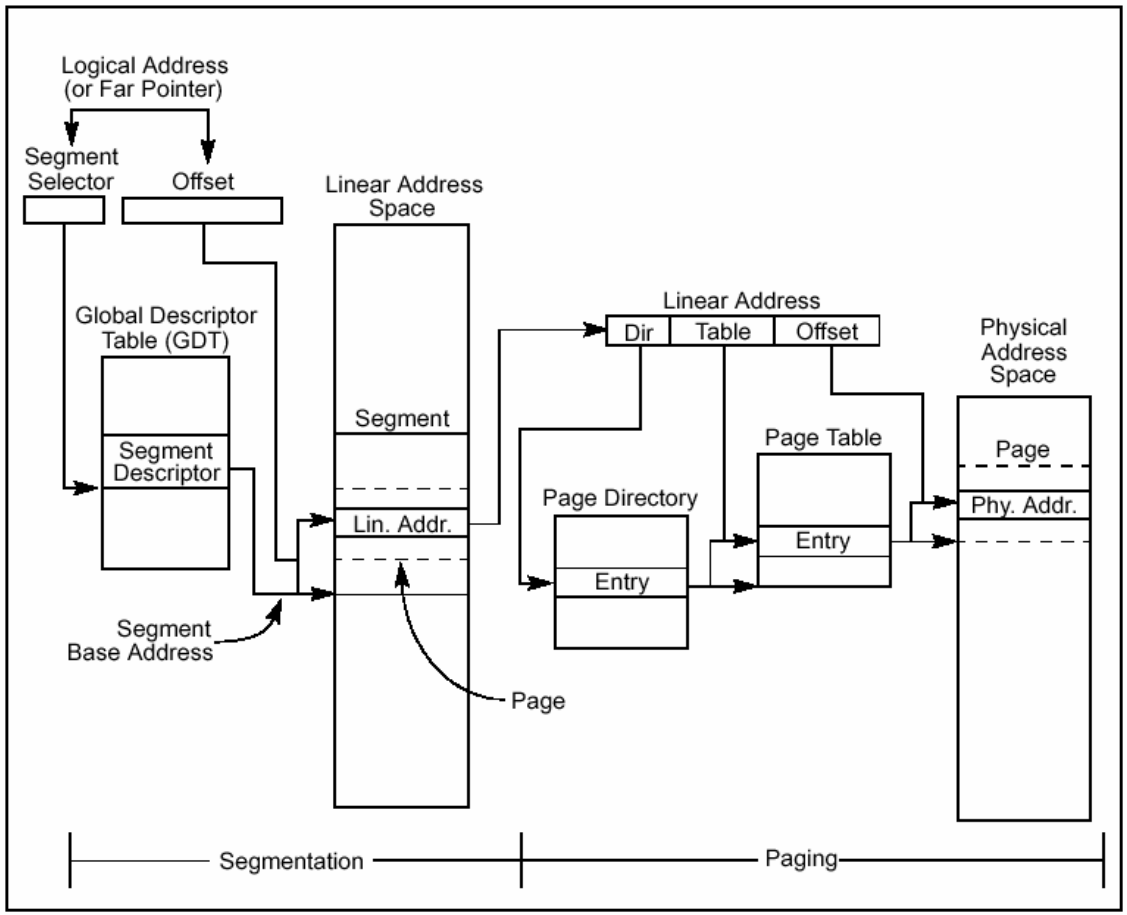
Stronicowanie pamięci

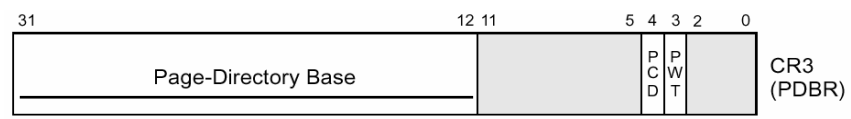


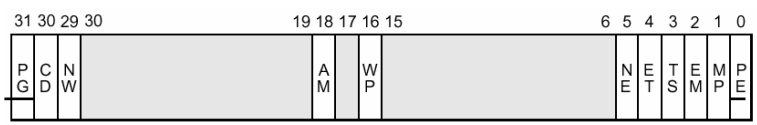


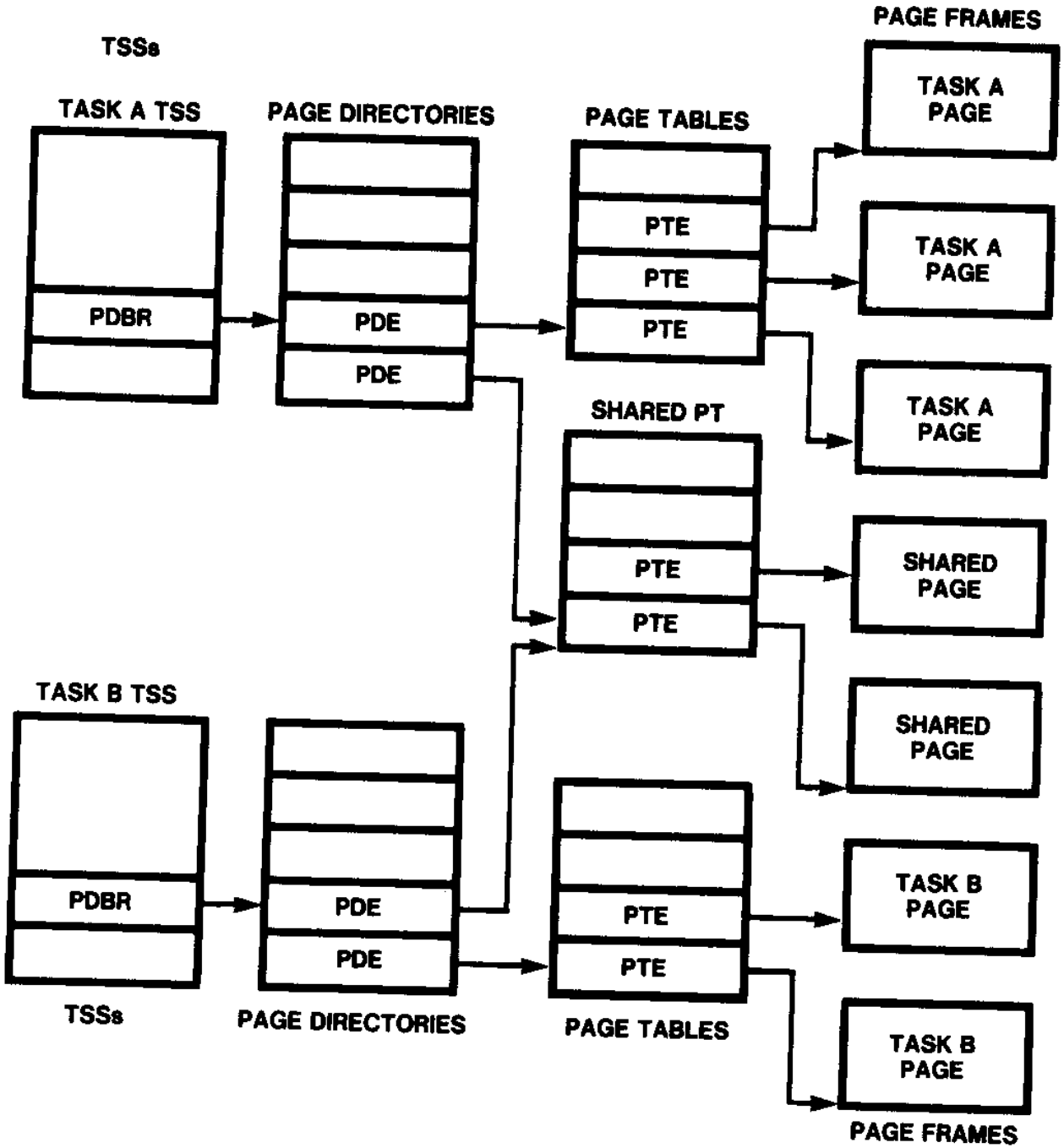


Segmentacja i stronicowanie







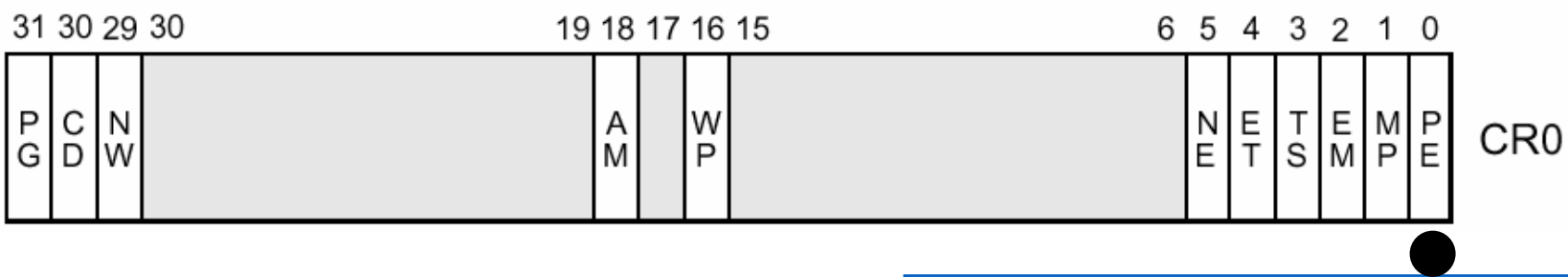

TSSs
TASK A TSS
PDBR
PAGE DIRECTORIES
PDE
PDE
PAGE TABLES
PTE
PTE
PTE
PAGE FRAMES
TASK A PAGE
TASK A PAGE
TASK A PAGE
SHARED PT
PTE
PTE
SHARED PAGE
SHARED PAGE
TASK B TSS
PDBR
TSSs
PDE
PDE
PAGE DIRECTORIES
PTE
PTE
PAGE TABLES
TASK B PAGE
TASK B PAGE
PAGE FRAMES

Struktury deskryptorowe

- **[S=1]-Segmenty pamięci (danych i kodu)**
- **[S=0]-Segmenty systemowe**

Struktury deskryptorowe

Adres bazowy segmentu 31..24								G	0	0	AVL	Wielkość 19..16			
P	DPL			S		typ			Adres bazowy segmentu 23..16						
Adres bazowy segmentu 15..0															
Wielkość segmentu 15..0															



Segmenty danych i kodu (S=1)

1. Segment danych, tylko odczytywanie.
2. Segment danych, odczytywanie i zapisywanie.
3. Segment danych rozszerzany w dół, tylko odczytywanie.
4. Segment danych rozszerzany w dół, odczytywanie i zapisywanie.
5. Segment kodu, tylko wykonywanie.
6. Segment kodu, wykonywanie i odczytywanie.
7. Segment zgodny, tylko wykonywanie.
8. Segment zgodny, wykonywanie i odczytywanie.

Deskryptor systemowy

Adres bazowy segmentu 31..24					G	0	0	AVL	Wielkość 19..16
P	DPL		S	typ	Adres bazowy segmentu 23..16				
Adres bazowy segmentu 15..0									
Wielkość segmentu 15..0									

TYP	1	Dostępny segment stanu zadania (TSS)
	2	Lokalna tablica deskryptorów
	3	Zajęty segment stanu zadania
P	0	Zawartość deskryptora jest nieważna
	1	Deskryptor opisuje segment w pamięci
DPL	0-3	Descriptor Privilege Level
Adres	32-bity	Adres początkowy
Limit	16-bitów	Przesunięcie

Deskryptor systemowy

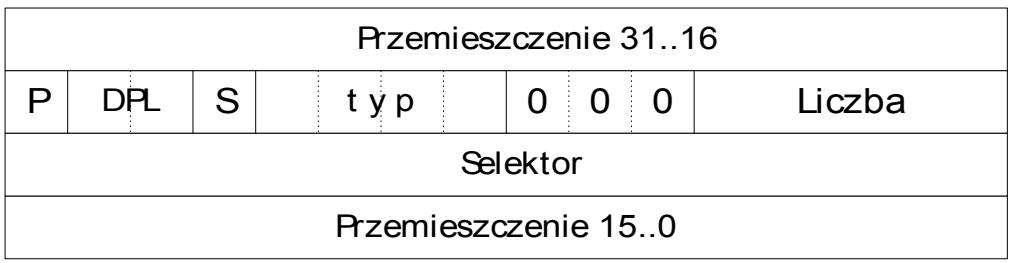
TYP	4	Bramka wywołania
	5	Bramka zadania
	6	Bramka przerwań
	7	Bramka pułapki
P	0	Zawartość deskryptora jest nieważna
	1	Deskryptor opisuje segment w pamięci
DPL	0-3	Descriptor Privilege Level
Licznik słów	0-31	Liczba słów do skopiowania ze stosu
Selektor celu	16-bitów	Selektor wskazujący na deskryptor segmentu kodu lub stanu zadania
Przesunięcie celu	16-bitów	Punkt wejścia wewnątrz segmentu docelowego

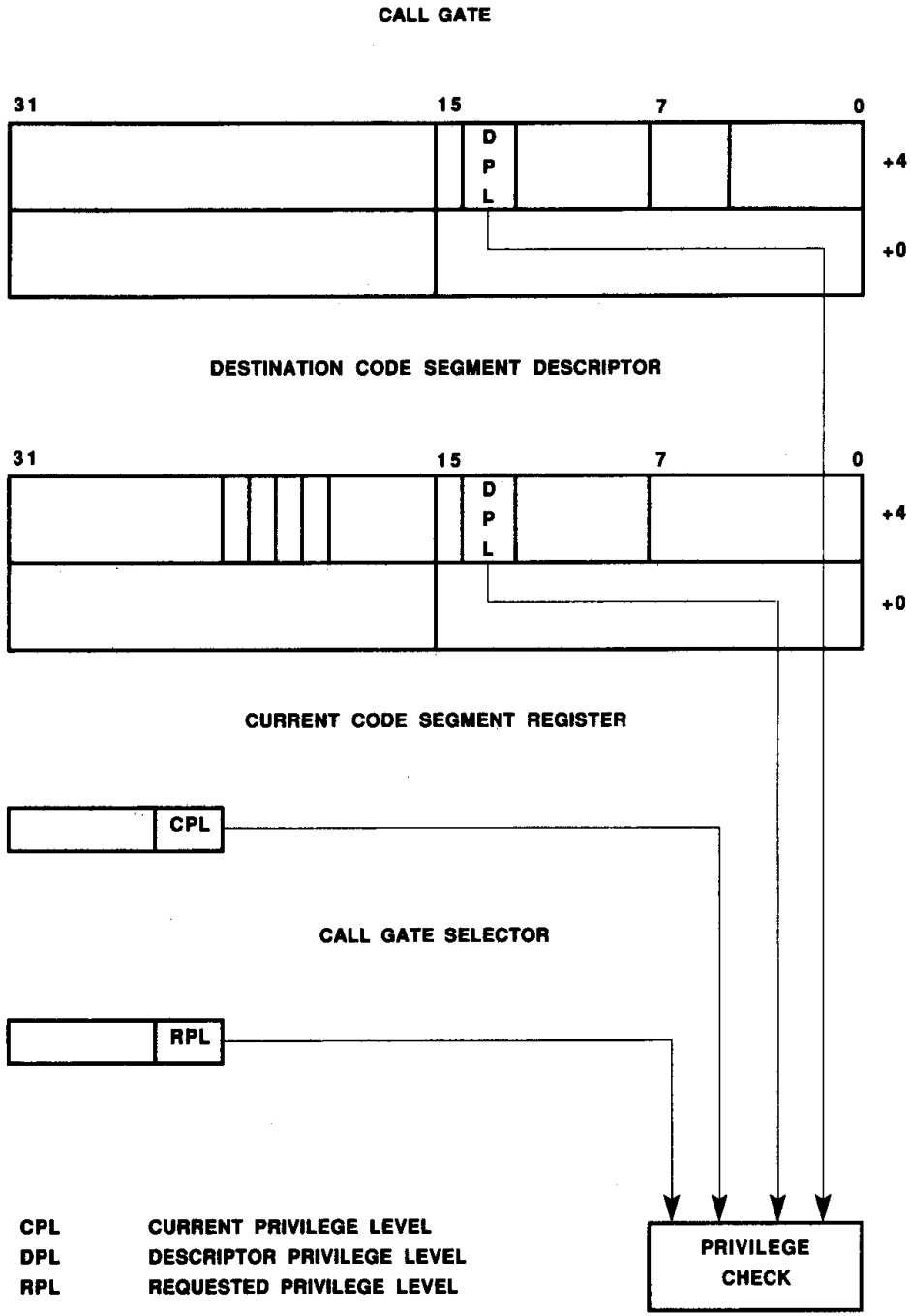
Struktury deskryptorowe

Adres bazowy segmentu 31..24				G	D	0	AVL	Wielkość 19..16
P	DPL	S	typ	A	Adres bazowy segmentu 23..16		
Adres bazowy segmentu 15..0							
Wielkość segmentu 15..0							

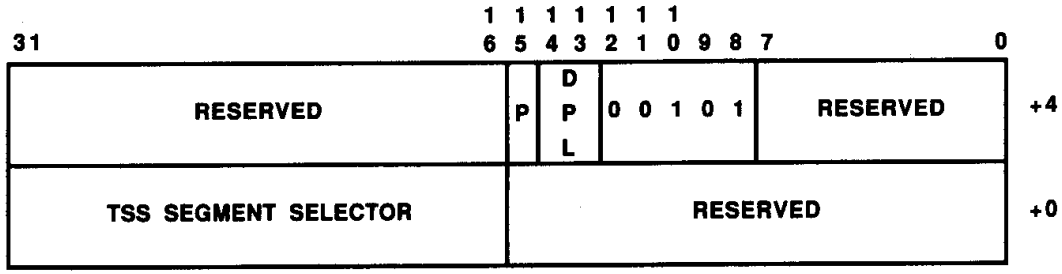
Struktury deskryptorowe

Deskryptor bramki

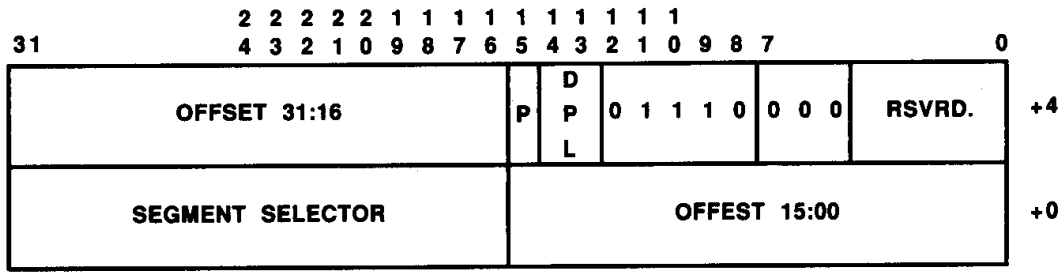



CALL GATE
31
15
7
0
D
P
L
+4
+0
DESTINATION CODE SEGMENT DESCRIPTOR
31
15
7
0
D
P
L
+4
+0
CURRENT CODE SEGMENT REGISTER
CPL
CALL GATE SELECTOR
RPL
CPL CURRENT PRIVILEGE LEVEL
DPL DESCRIPTOR PRIVILEGE LEVEL
RPL REQUESTED PRIVILEGE LEVEL
PRIVILEGE CHECK

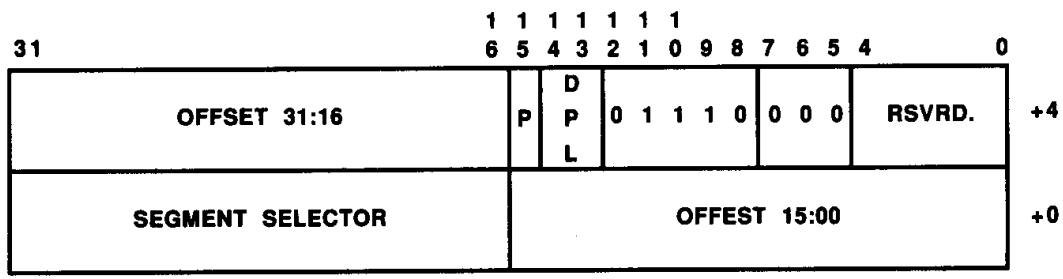
TASK GATE



INTERRUPT GATE



TRAP GATE



DPL	DESCRIPTOR PRIVILEGE LEVEL
OFFSET	OFFSET TO PROCEDURE ENTRY POINT
P	SEGMENT PRESENT BIT
RESERVED	DO NOT USE
SELECTOR	SEGMENT SELECTOR FOR DESTINATION CODE SEGMENT

Przerwania i wyjątki

Sytuacje wyjątkowe - to zdarzenia, których wystąpienie wymaga interwencji systemu operacyjnego. Poszczególne rodzaje sytuacji wyjątkowych maja różne źródła i wymagają nieco odmiennej obsługi. Można wyróżnić trzy klasy sytuacji wyjątkowych:

- Przerwania,
- Pułapki,
- Błędy.

Przerwania i wyjątki

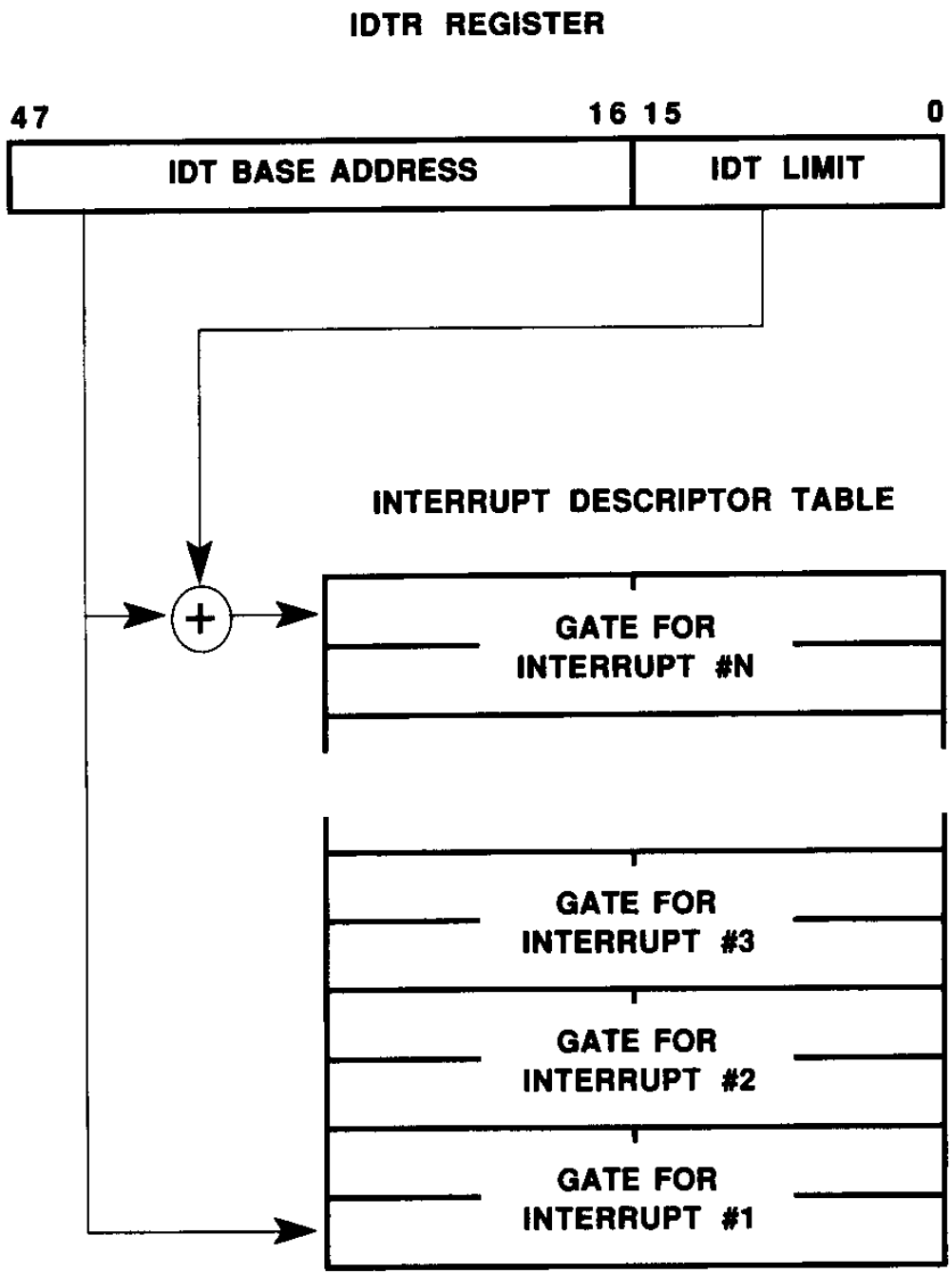
Przerwania są to sytuacje wyjątkowe, których przyczyny leżą poza modułem procesora. Zwykle są one generowane przez sterowniki urządzeń. Typowe źródła przerwań, to np. zegar systemowy, sterowniki pamięci dyskowej, sterowniki transmisji, sterowniki bezpośredniego dostępu do pamięci (DMA). Mechanizm przerwań służy do informowania o gotowości urządzenia do współpracy lub zajściu jakiegoś zdarzenia, np. upłynięciu określonego czasu.

Przerwania i wyjątki

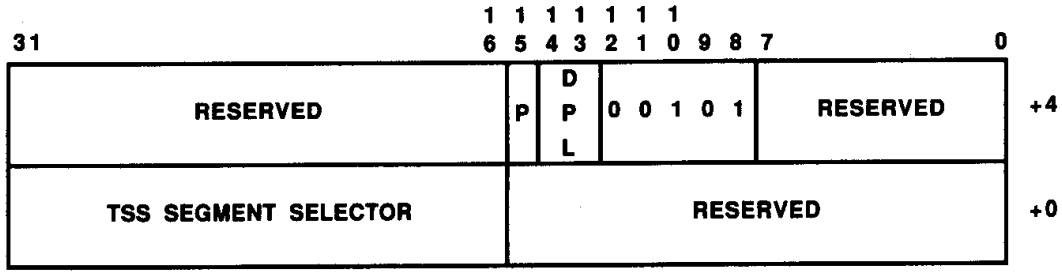
Przerwania są obsługiwane po zakończeniu wykonania instrukcji lub nieprzerywalnego ciągu instrukcji, w czasie wykonania, których przerwanie zostanie wykryte przez procesor.

Układ wykrywania przerwań może być włączany i wyłączany przez system operacyjny na drodze programowej, poprzez modyfikację bitów w rejestrze stanu procesora sterujących działaniem układu przerwań.

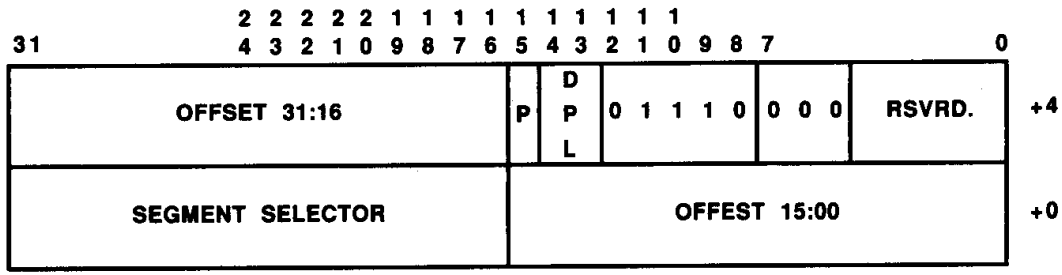
Rejestr przerwań


IDTR REGISTER
47
16 15
0
IDT BASE ADDRESS
IDT LIMIT
INTERRUPT DESCRIPTOR TABLE
GATE FOR INTERRUPT #N
GATE FOR INTERRUPT #3
GATE FOR INTERRUPT #2
GATE FOR INTERRUPT #1

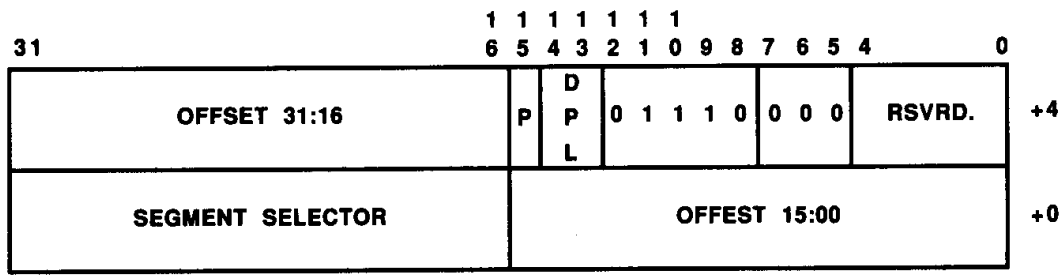
TASK GATE



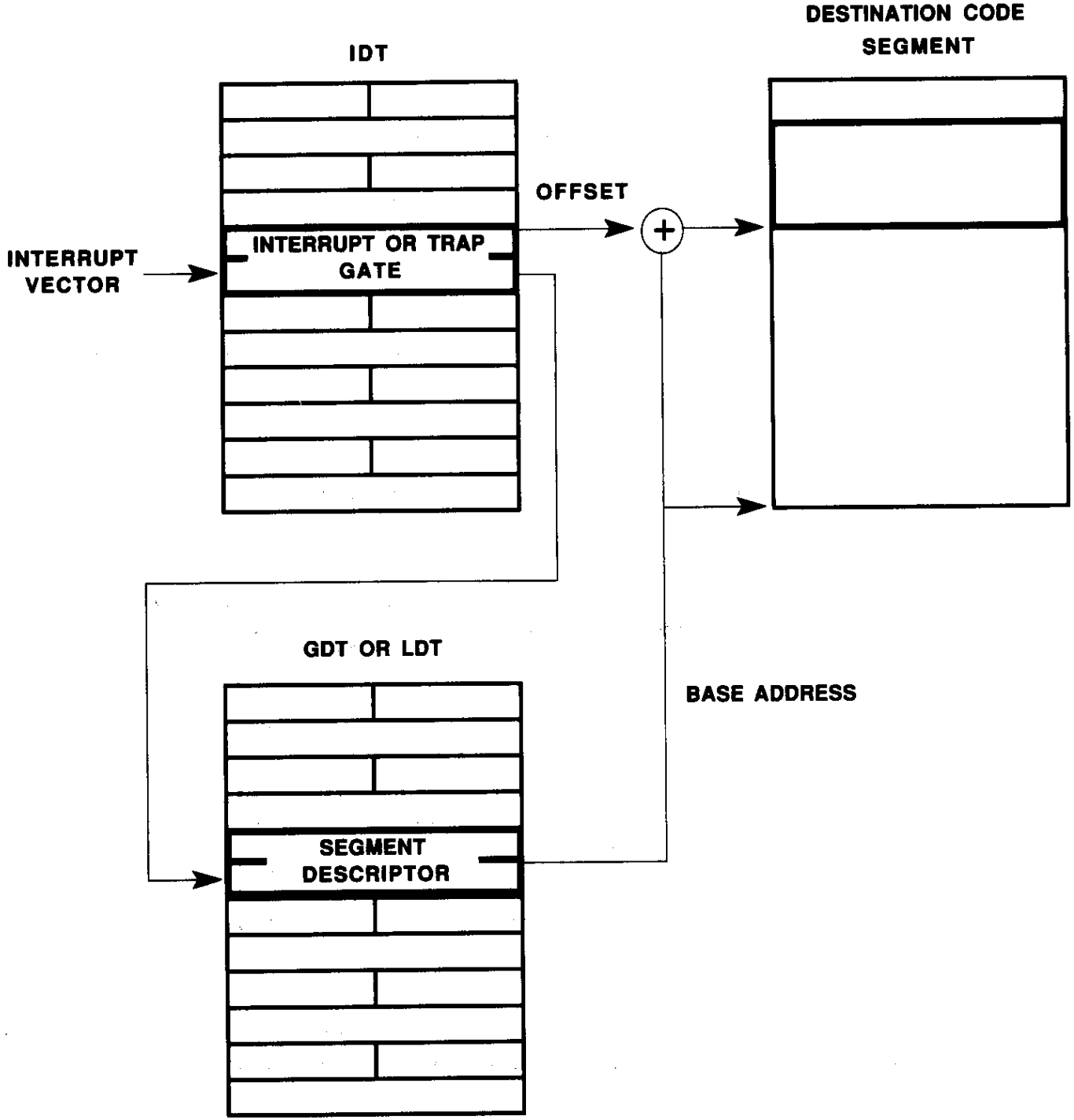
INTERRUPT GATE

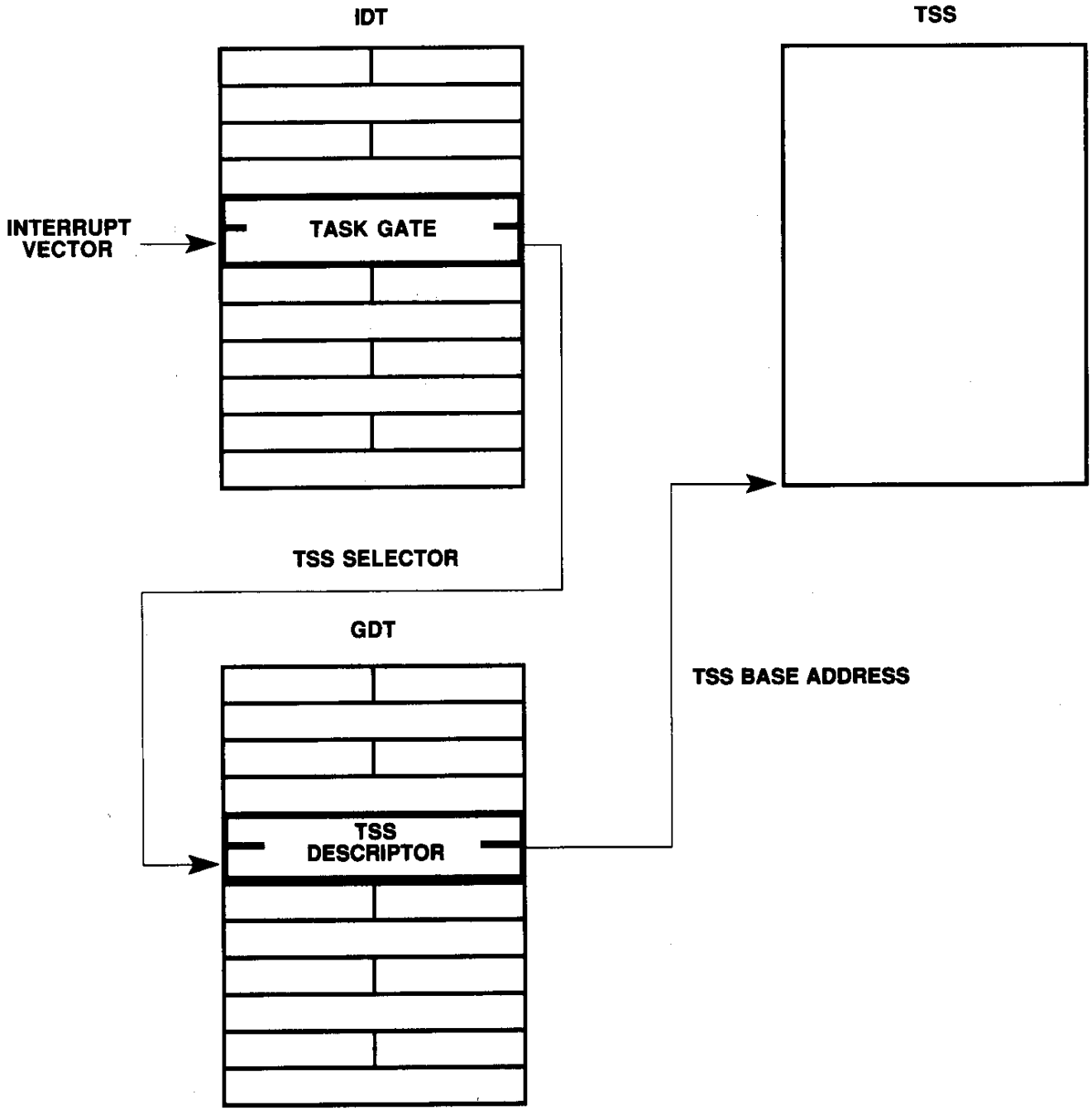


TRAP GATE



DPL	DESCRIPTOR PRIVILEGE LEVEL
OFFSET	OFFSET TO PROCEDURE ENTRY POINT
P	SEGMENT PRESENT BIT
RESERVED	DO NOT USE
SELECTOR	SEGMENT SELECTOR FOR DESTINATION CODE SEGMENT


IDT
DESTINATION CODE SEGMENT
OFFSET
INTERRUPT VECTOR
INTERRUPT OR TRAP GATE
GDT OR LDT
BASE ADDRESS
SEGMENT DESCRIPTOR


IDT
TSS
INTERRUPT VECTOR
TASK GATE
TSS SELECTOR
GDT
TSS BASE ADDRESS
TSS DESCRIPTOR

Pułapki

- Pułapki powstają wewnątrz procesora w związku z wykonaniem instrukcji.
- Pułapki są zaplanowane lub dopuszczone przez programistę lub system operacyjny i stanowią świadome żądanie interwencji systemu operacyjnego.

Pułapki

Podstawowe klasy pułapek to:

- pułapki będące zaprogramowanym wynikiem wykonania instrukcji, czyli tzw. "przerwania programowe", wykorzystywane do wywołania funkcji systemu operacyjnego,
- pułapki wywołane przez wykonanie niektórych instrukcji w sposób uznany za niepoprawny:
  - dzielenie przez zero,
  - wykroczenie indeksu tablicy poza dozwolony zakres
  - pułapki śledzenia (*trace*), wywołane przez uaktywniony wcześniej moduł śledzenia wykonania programów stanowiący część procesora.

Błędy

Błędy mogą być generowane zarówno przez procesor, jak i przez jego otoczenie (błąd transmisji na szynie). Powstają one w związku z wykonaniem programu, ale nie są zaplanowane i nie zawsze są synchroniczne względem strumienia instrukcji. Błędy mogą wymagać natychmiastowej obsługi - niekiedy konieczne jest nawet zaniechanie wykonywania bieżącej instrukcji.

Błędy

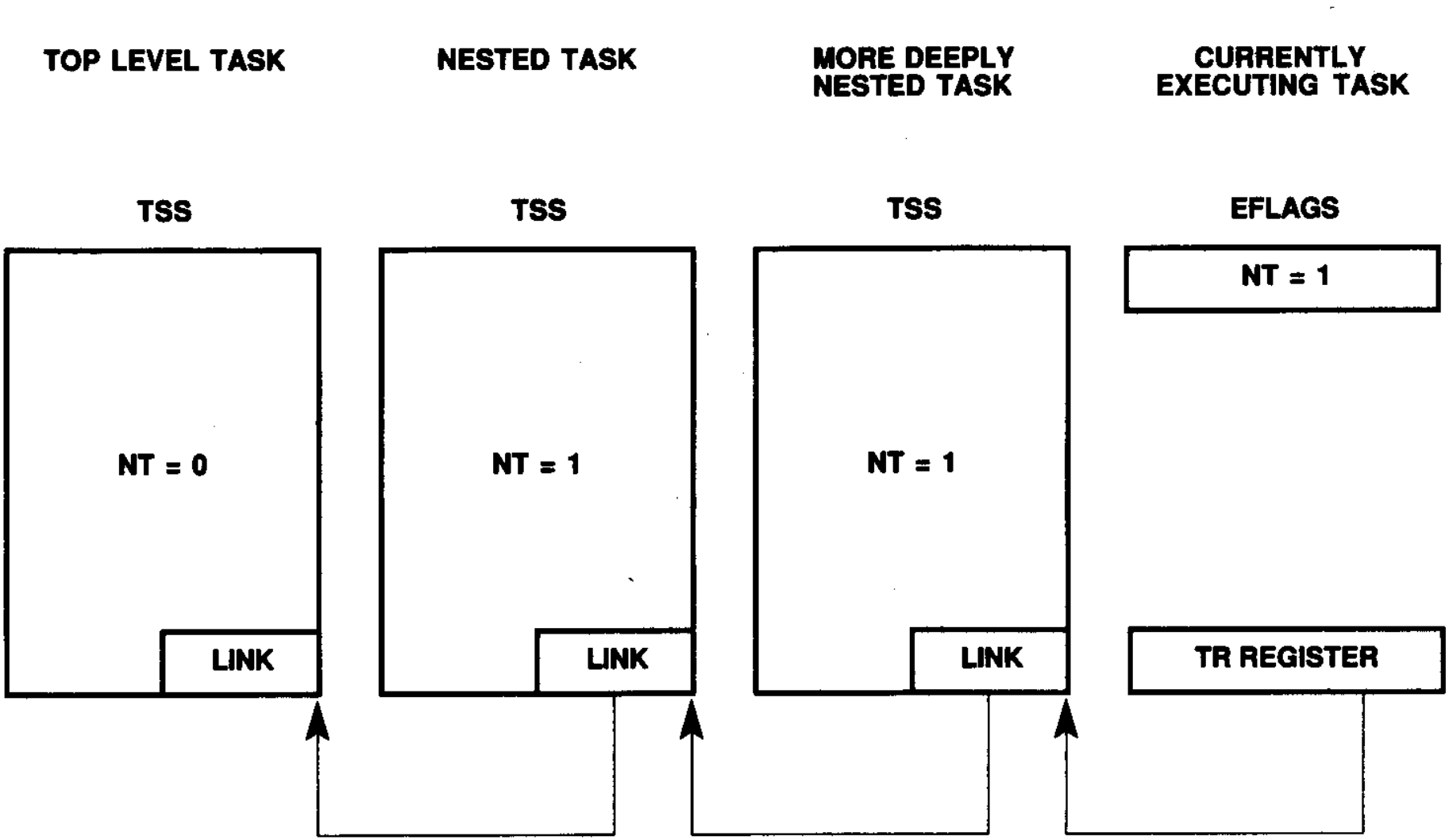
Przykładami błędów są:

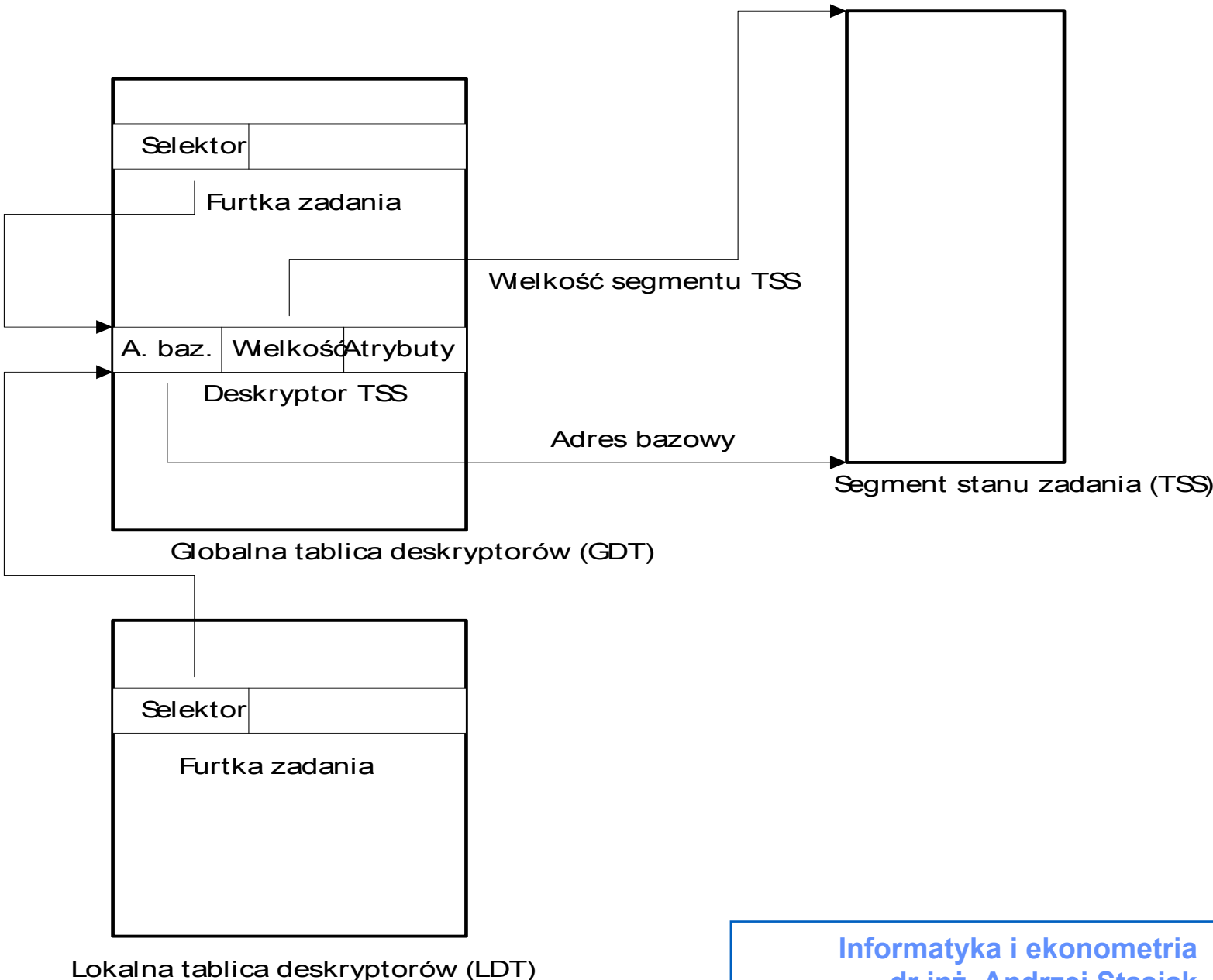
- błąd dostępu do pamięci (błąd strony lub segmentu),
- błąd szyny (np. błąd parzystości transmisji), próba odwołania do nieistniejącego urządzenia lub fragmentu pamięci),
- niezidentyfikowany kod operacyjny instrukcji,
- pogwałcenie zasad ochrony zasobów.

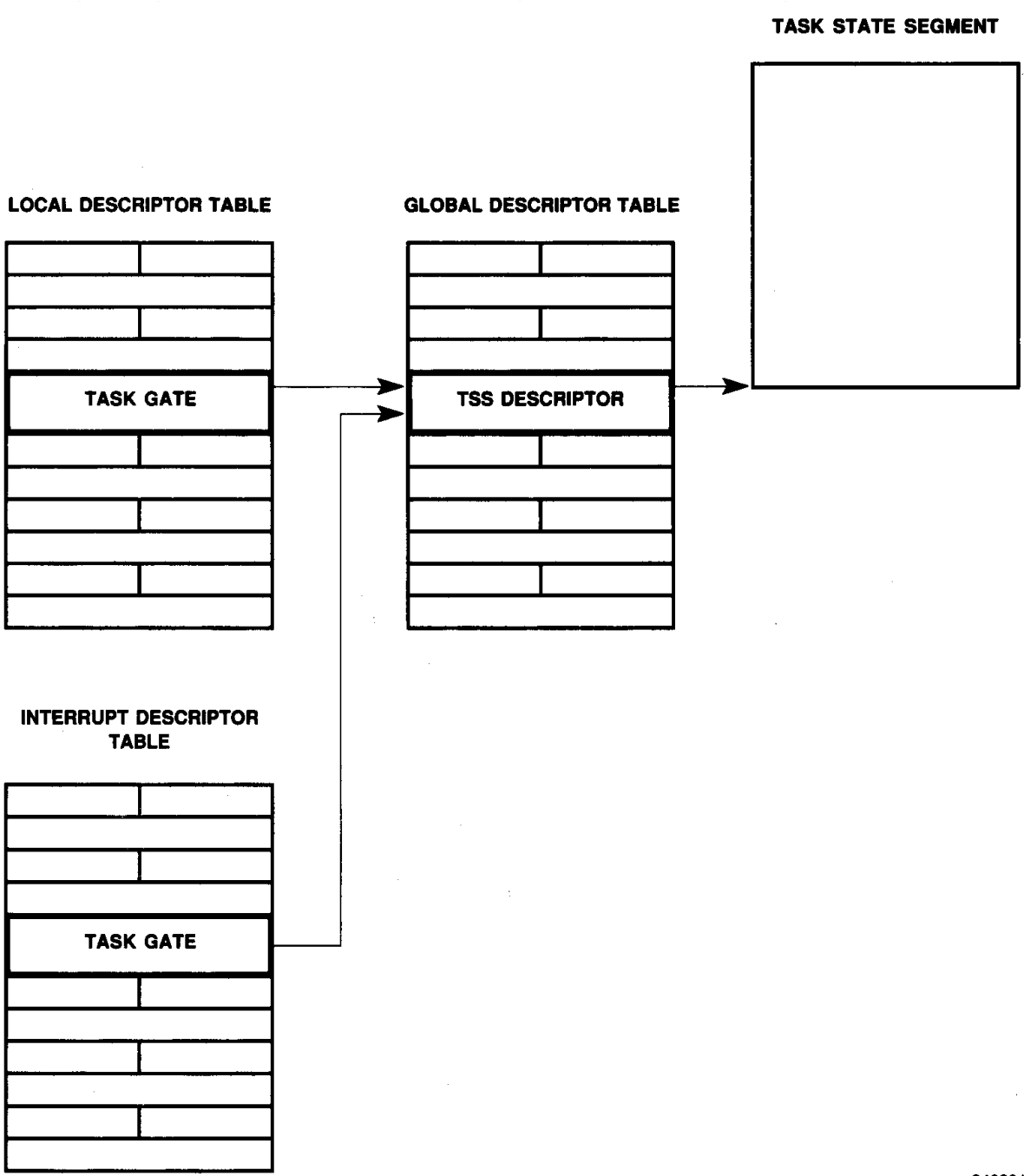
Rejestr stanu zadania

Segment stanu

Położenie tablicy zezwoleń we- wy	0 T
0	LDT
0	GS
0	FS
0	DS
0	SS
0	CS
0	ES
EDI	
ESI	
EBP	
ESP	
EBX	
EDX	
ECX	
EAX	
Rejestr znaczników EFLAGS	
EIP	
CR3	
0	SS (poziom 2)
ESP (poziom 2)	
0	SS (poziom 1)
ESP (poziom 1)	
0	SS (poziom 0)
ESP (poziom 0)	
0	Wskaźnik poprzedniego zadania


TOP LEVEL TASK
NESTED TASK
MORE DEEPLY NESTED TASK
CURRENTLY EXECUTING TASK
TSS
TSS
TSS
EFLAGS
NT = 0
NT = 1
NT = 1
NT = 1
LINK
LINK
LINK
TR REGISTER


Selektor
Furtka zadania
Wielkość segmentu TSS
A. baz.
Wielkość
Atrybuty
Deskryptor TSS
Adres bazowy
Segment stanu zadania (TSS)
Globalna tablica deskryptorów (GDT)
Selektor
Furtka zadania
Lokalna tablica deskryptorów (LDT)


TASK STATE SEGMENT
LOCAL DESCRIPTOR TABLE
GLOBAL DESCRIPTOR TABLE
TASK GATE
TSS DESCRIPTOR
INTERRUPT DESCRIPTOR TABLE
TASK GATE

MECHANIZMY OCHRONY W SYSTEMACH WIELOZADANIOWYCH

- Separacja zadań (każde zadanie dysponuje własną wirtualną przestrzenią adresową).
- Wprowadzenie poziomów ochrony (każdy segment, zarówno danych, jak i kodu ma przypisany poziom ochrony).
- Poziomy ochrony stron (dla adresowania ze stronicowaniem: poziom supervisor i user).

ZASADY OCHRONY ZASOBÓW

Zasady te wynikają z następującego rozumowania:

- z jednej strony nie można programowi na danym poziomie zezwolić na dostęp (odczytywanie, modyfikowanie) danych bardziej chronionych
- z drugiej strony - w realizacji zadań na pewnym poziomie nie można korzystać z procedur „mniej godnych zaufania”

ZASOBY CHRONIONE PRZEZ SYSTEM OPERACYJNY

System operacyjny gospodaruje:

- czasem procesora,
- systemem pamięci,
- urządzeniami zewnętrznymi komputera.

Architektura procesora musi umożliwiać systemowi operacyjnemu sprawne dysponowanie zasobami i ochronę ich integralności.

ZASADY OCHRONY ZASOBÓW

W każdym systemie wieloprocesowym i wieloużytkownikowym istnieje konieczność ochrony zasobów systemu przed użytkownikami i ochrony poszczególnych użytkowników przed innymi użytkownikami. Żadne błędne ani świadomie destrukcyjne zachowanie programu użytkownika nie może powodować awarii systemu ani wpływać na pracę innych użytkowników.

ZASADY OCHRONY ZASOBÓW

W kontekście ochrony zasobów systemu następujące zachowania programu użytkownika uznawane są za błędne i powodują stosowną akcję systemu:

- modyfikacja kodu programu,
- "wykonanie" danych lub odczyt kodu,
- dostęp do pamięci poza obszarem zaalokowanym przez system operacyjny,
- dostęp do urządzeń wejścia-wyjścia,
- modyfikacja kontekstu systemowego (maski przerwań, deskryptorów procedur obsługi sytuacji wyjątkowych itp.).

ZASADY OCHRONY ZASOBÓW

Zasady te wynikają z następującego rozumowania:

- z jednej strony nie można programowi na danym poziomie zezwolić na dostęp (odczytywanie, modyfikowanie) danych bardziej chronionych
- z drugiej strony - w realizacji zadań na pewnym poziomie nie można korzystać z procedur „mniej godnych zaufania”

System ochrony zasobów

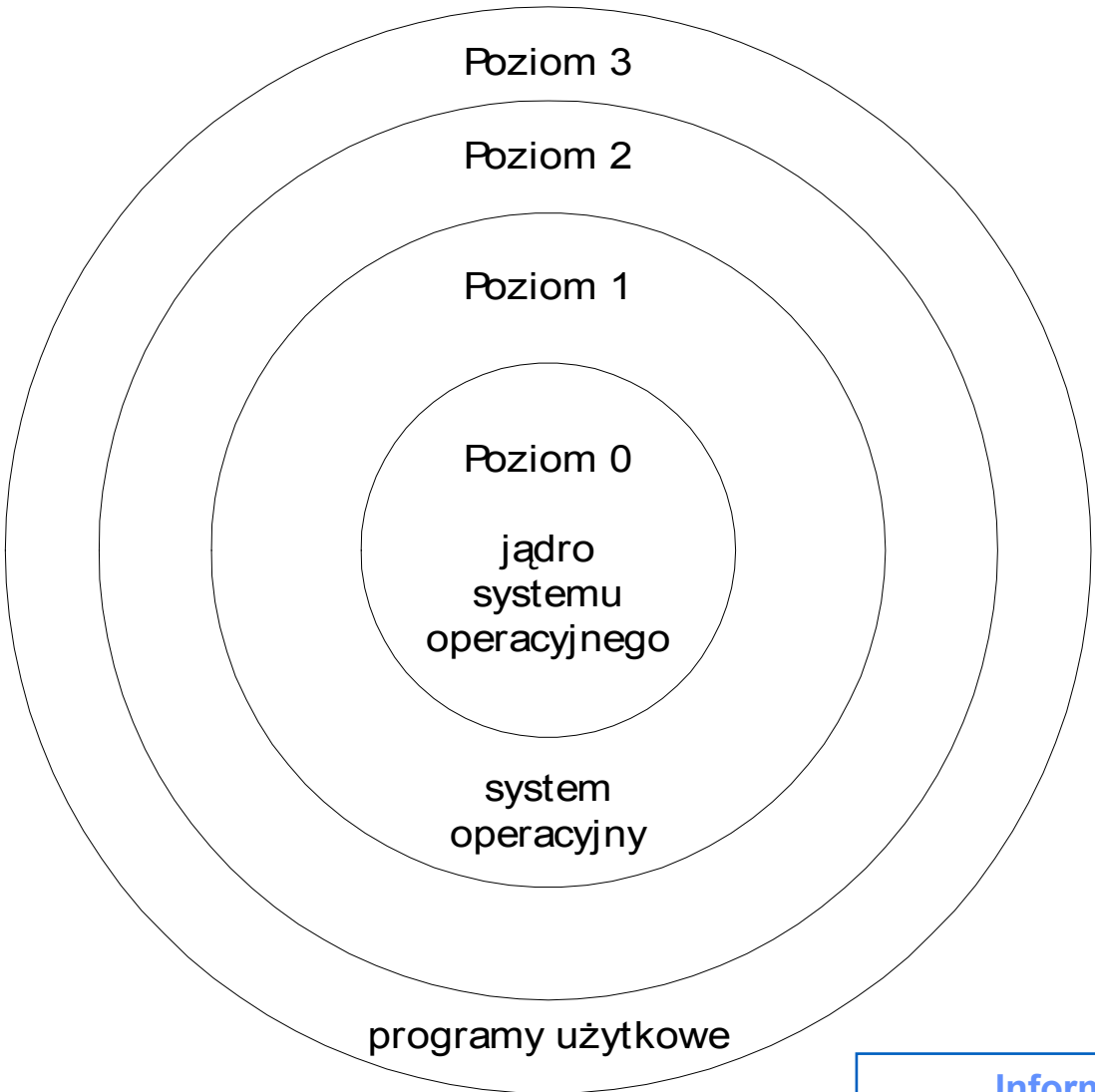
Aby zapewnić ochronę zasobów należy przede wszystkim wyróżnić przynajmniej dwa poziomy uprzywilejowania w systemie:

- poziom systemu
- użytkownika.

Oprogramowanie działające na poziomie systemowym (system operacyjny) jest traktowane jako zaufane, tj. uznane za wolne od błędów mogących spowodować awarię systemu.

Pozostała część programowania (programy użytkowników) ma ograniczone uprawnienia.

Poziomy ochrony



System ochrony zasobów

Konieczne jest więc wyróżnienie dwóch trybów pracy komputera, trybu systemowego i użytkownika, różniących się zakresem uprawnień programów. Instrukcje operujące na zasobach systemowych, a więc instrukcje modyfikujące kontekst systemowy procesora, stan system-użytkownik, poziom zezwolenia na przerwania, rejestry związane z obsługą sytuacji wyjątkowych i zarządzaniem pamięci, są niedozwolone dla programów użytkownika. Próba ich wykonania powoduje powstanie sytuacji wyjątkowej, która jest obsługiwana przez system operacyjny.

Jak określamy poziom ochrony

Poziom ochrony określany jest na podstawie:

- wartości pola **RPL** w selektorze (**CPL to RPL selektora CS**) i
- wartości pola **DPL** w deskryptorze.

Procesor sprawdza relacje między tymi polami i dokonuje zmian poziomu uprzywilejowania lub generuje wyjątek.

Jak określamy poziom ochrony

Poziom ochrony określany jest na podstawie:

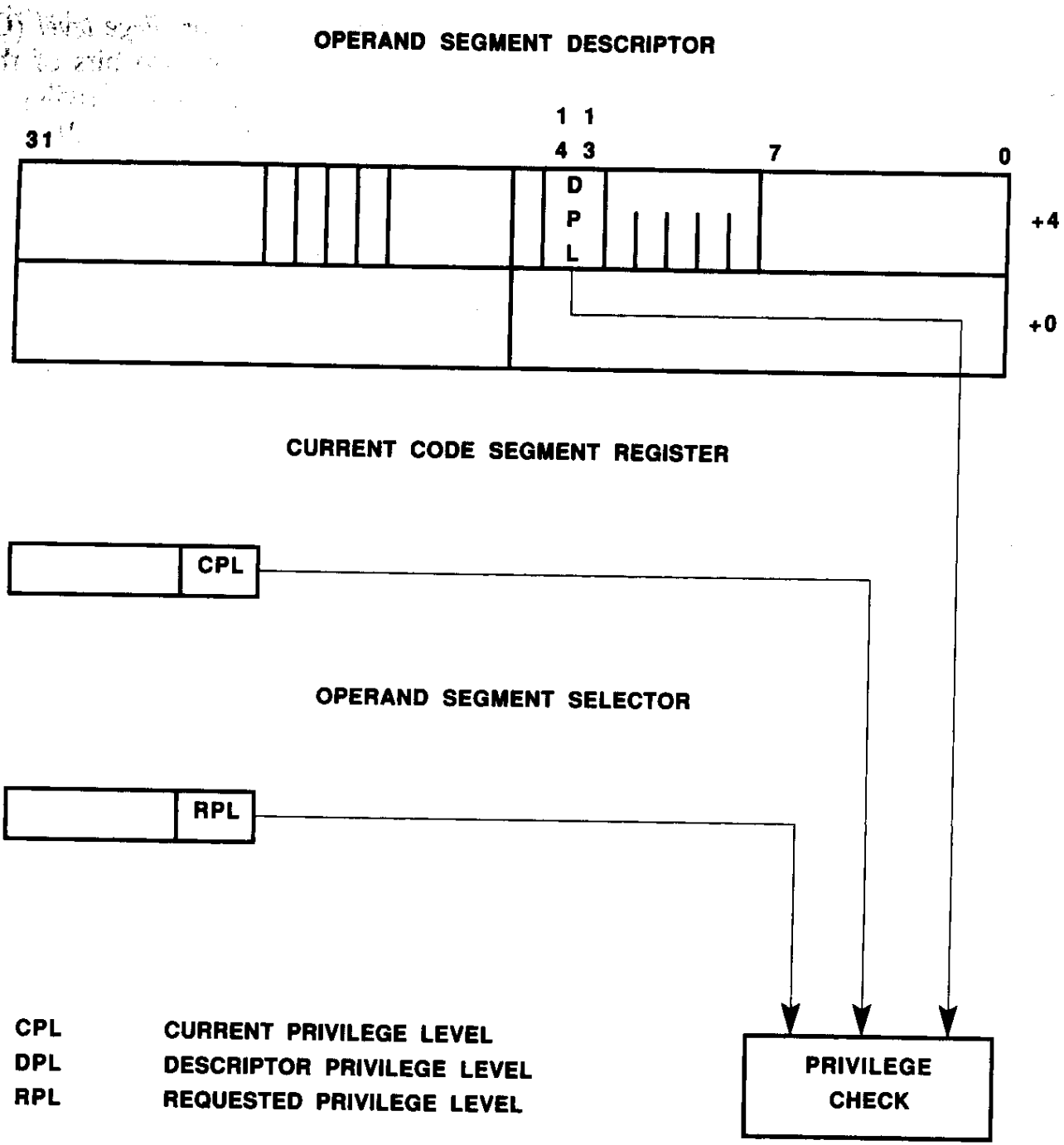
- wartości pola **RPL** w selektorze (**CPL to RPL selektora CS**) i
- wartości pola **DPL** w deskryptorze.

Procesor sprawdza relacje między tymi polami i dokonuje zmian poziomu uprzywilejowania lub generuje wyjątek.

Ochrona danych:

Zmiana poziomu ochrony danych jest możliwa gdy spełnione są warunki:

- CPL<=DPL i RPL<=DPL


OPERAND SEGMENT DESCRIPTOR
31
14
13
7
0
D
P
L
+4
+0
CURRENT CODE SEGMENT REGISTER
CPL
OPERAND SEGMENT SELECTOR
RPL
CPL CURRENT PRIVILEGE LEVEL
DPL DESCRIPTOR PRIVILEGE LEVEL
RPL REQUESTED PRIVILEGE LEVEL
PRIVILEGE CHECK

Ochrona kodu:

zmiana poziomu ochrony nastąpi gdy:

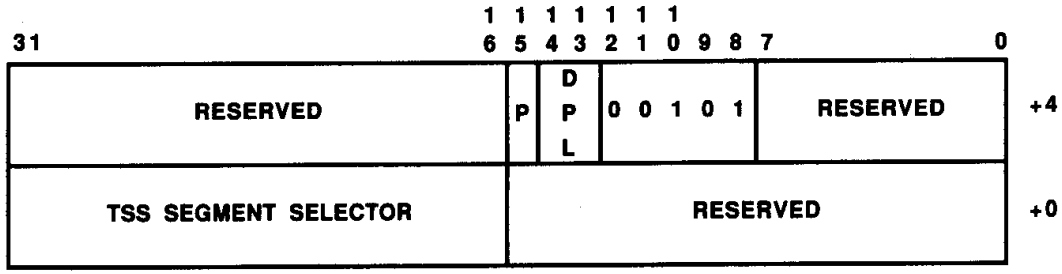
- DPL<=CPL **(czyli sterowanie przenosimy na ten sam lub wewnętrzny poziom ochrony)**
- DPL<CPL **i segment docelowy jest segmentem zgodnym**

dla segmentów zgodnych

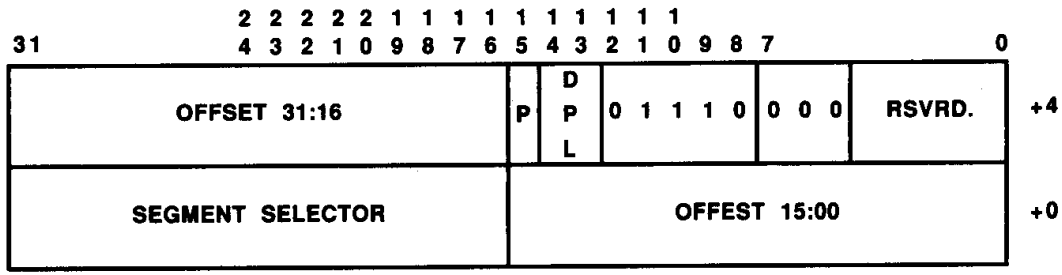
- **dla których kod procedury zawarty w tym segmencie nie wykonuje się na własnym - określonym przez pole DPL deskryptora - poziomie ochrony, lecz na poziomie programu wywołującego**

Deskryptor bramki

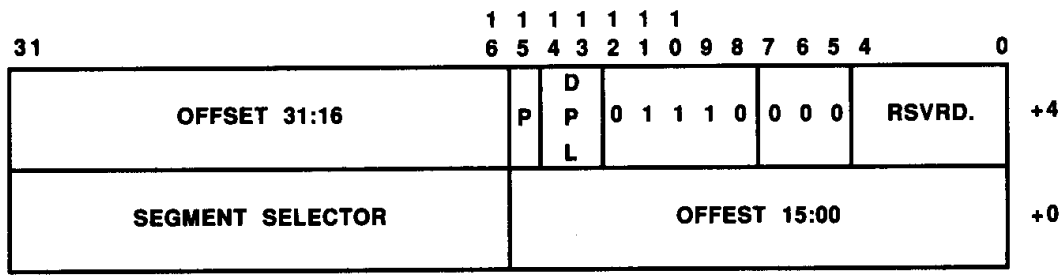
TASK GATE



INTERRUPT GATE

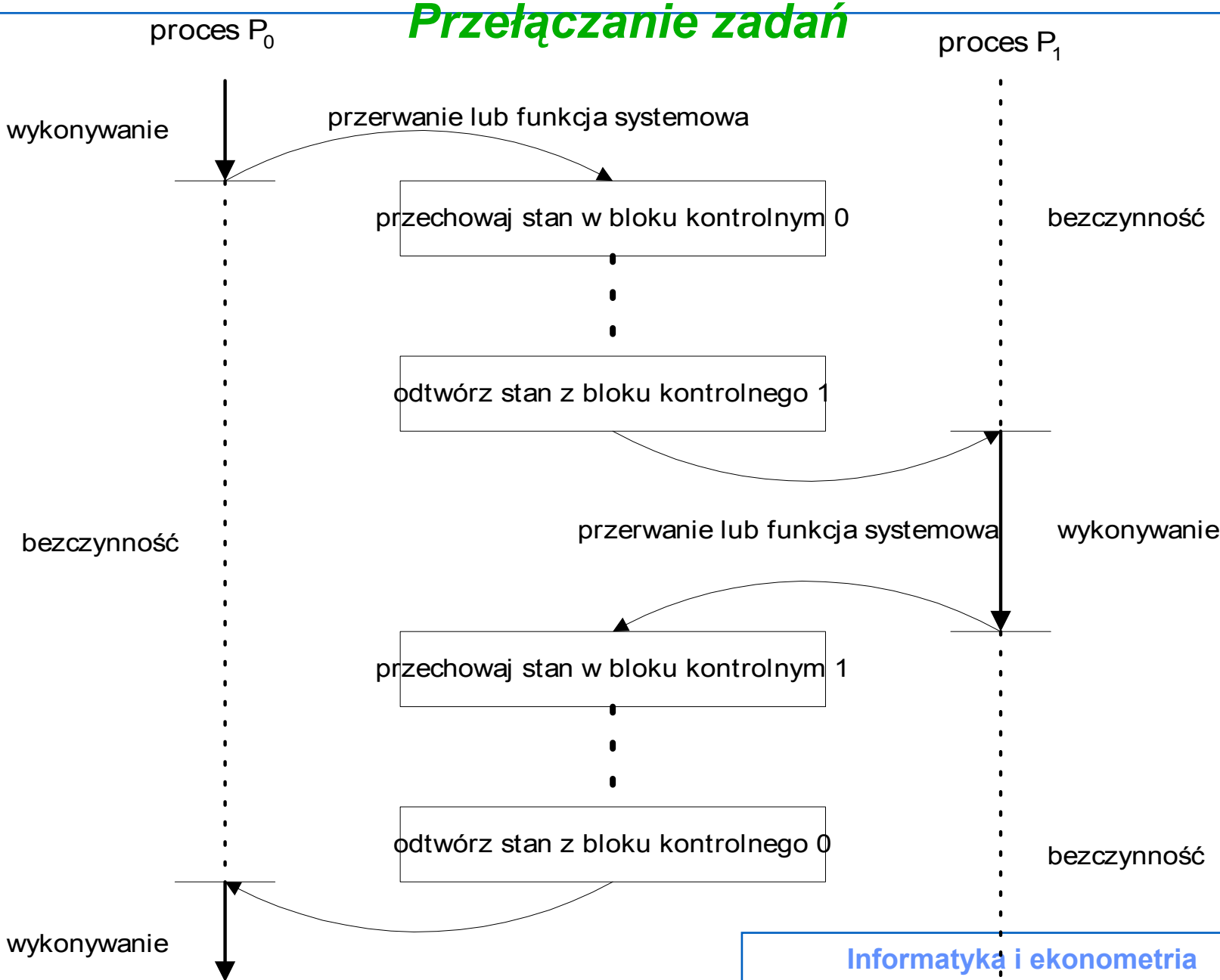


TRAP GATE

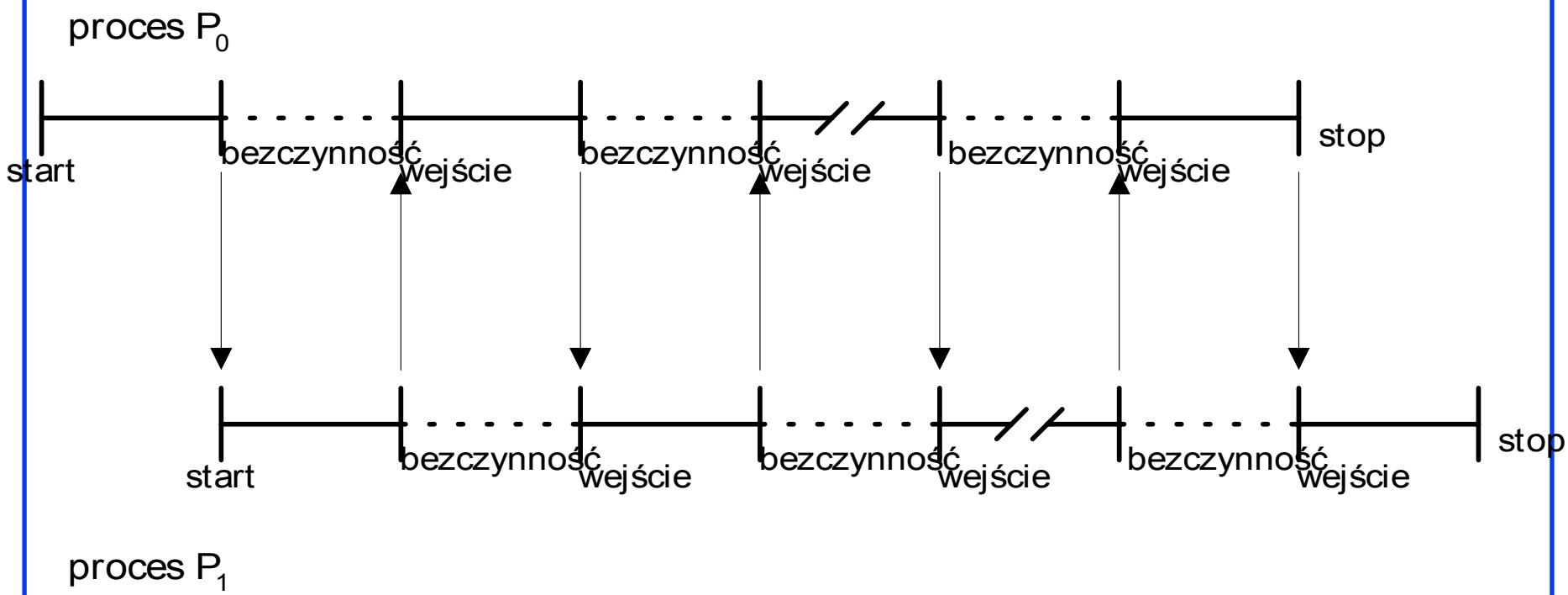


DPL	DESCRIPTOR PRIVILEGE LEVEL
OFFSET	OFFSET TO PROCEDURE ENTRY POINT
P	SEGMENT PRESENT BIT
RESERVED	DO NOT USE
SELECTOR	SEGMENT SELECTOR FOR DESTINATION CODE SEGMENT

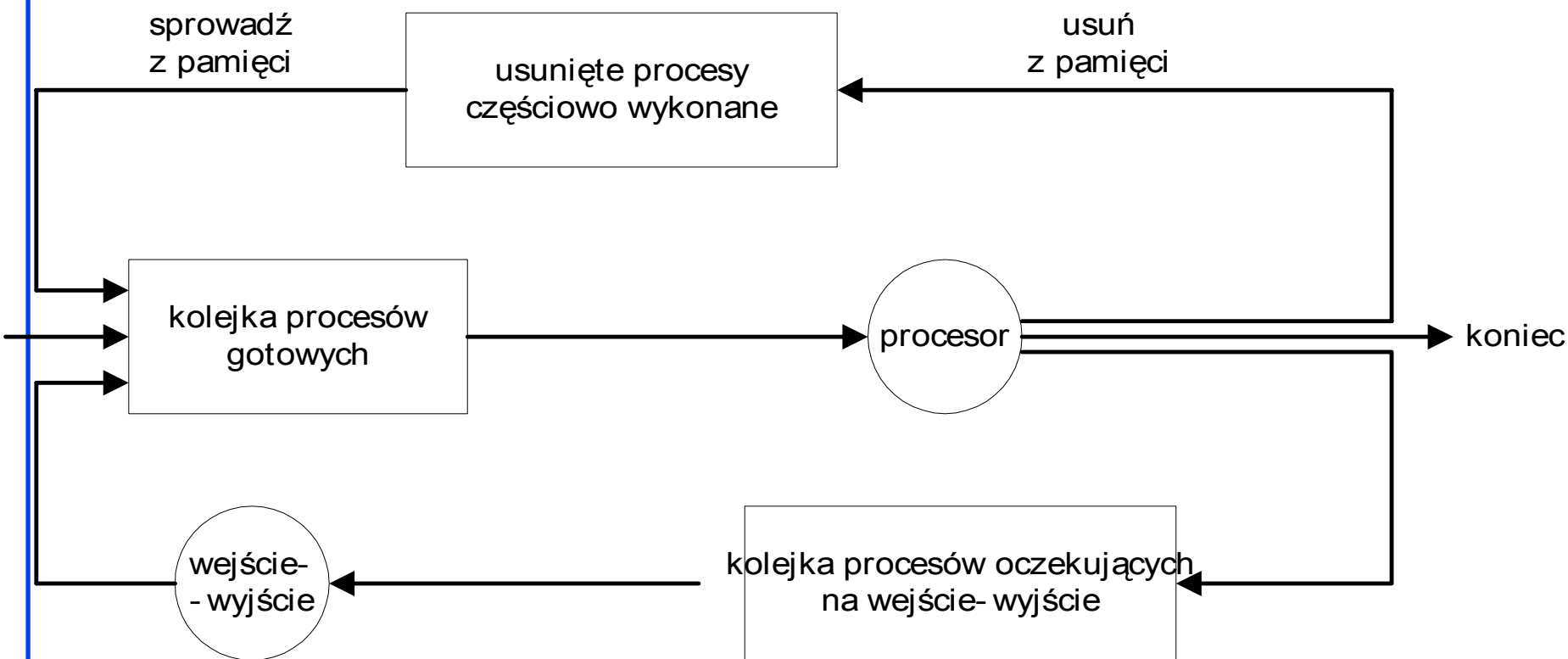
Przełączanie zadań

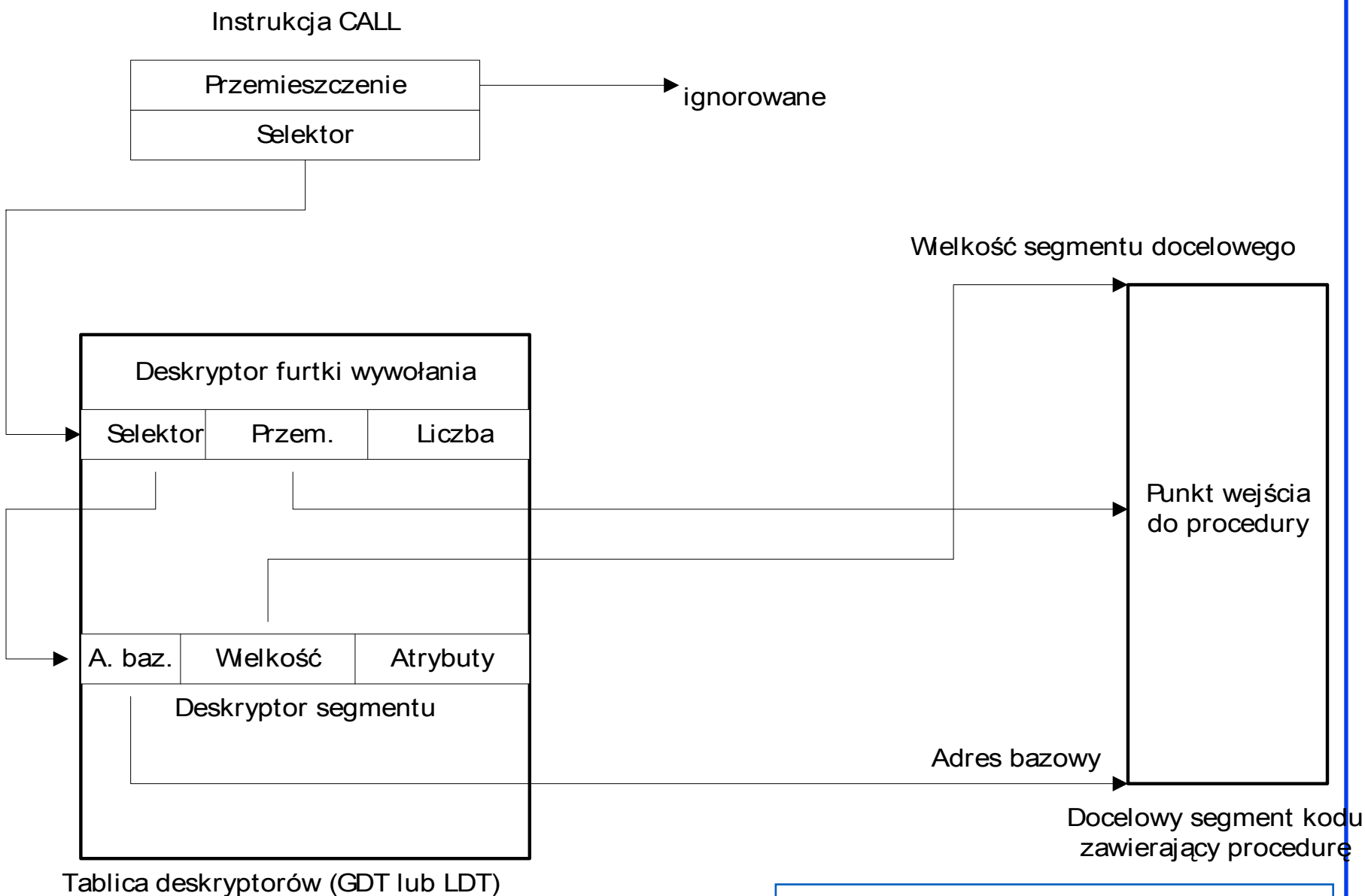


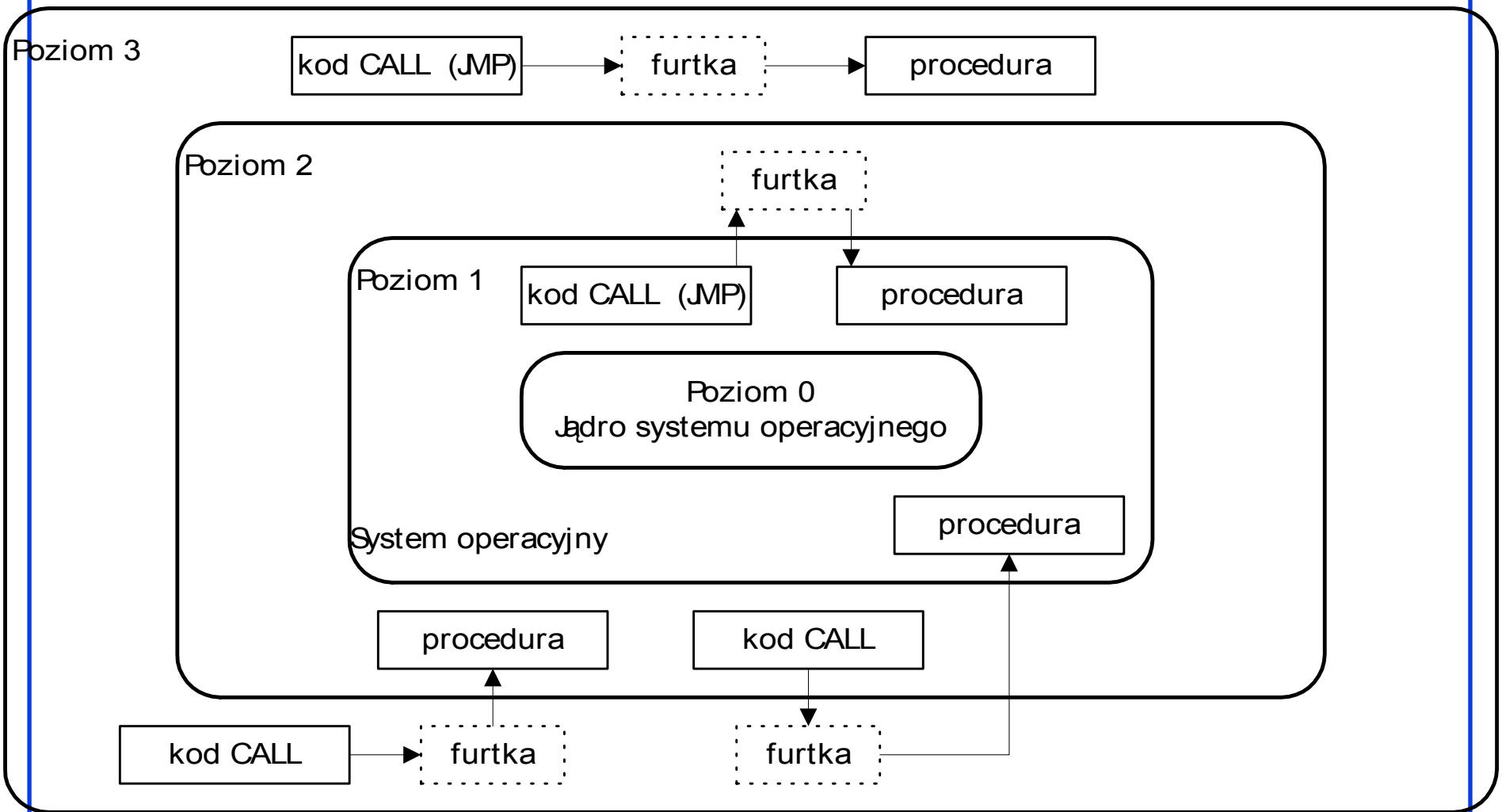
Przełączanie zadań



Przełączanie zadań




Instrukcja CALL
Przemieszczenie
Selektor
ignorowane
Wielkość segmentu docelowego
Deskryptor furtki wywołania
Selektor
Przem.
Liczba
Punkt wejścia
do procedury
A. baz.
Wielkość
Atrybuty
Deskryptor segmentu
Adres bazowy
Docelowy segment kodu
zawierający procedurę
Tablica deskryptorów (GDT lub LDT)

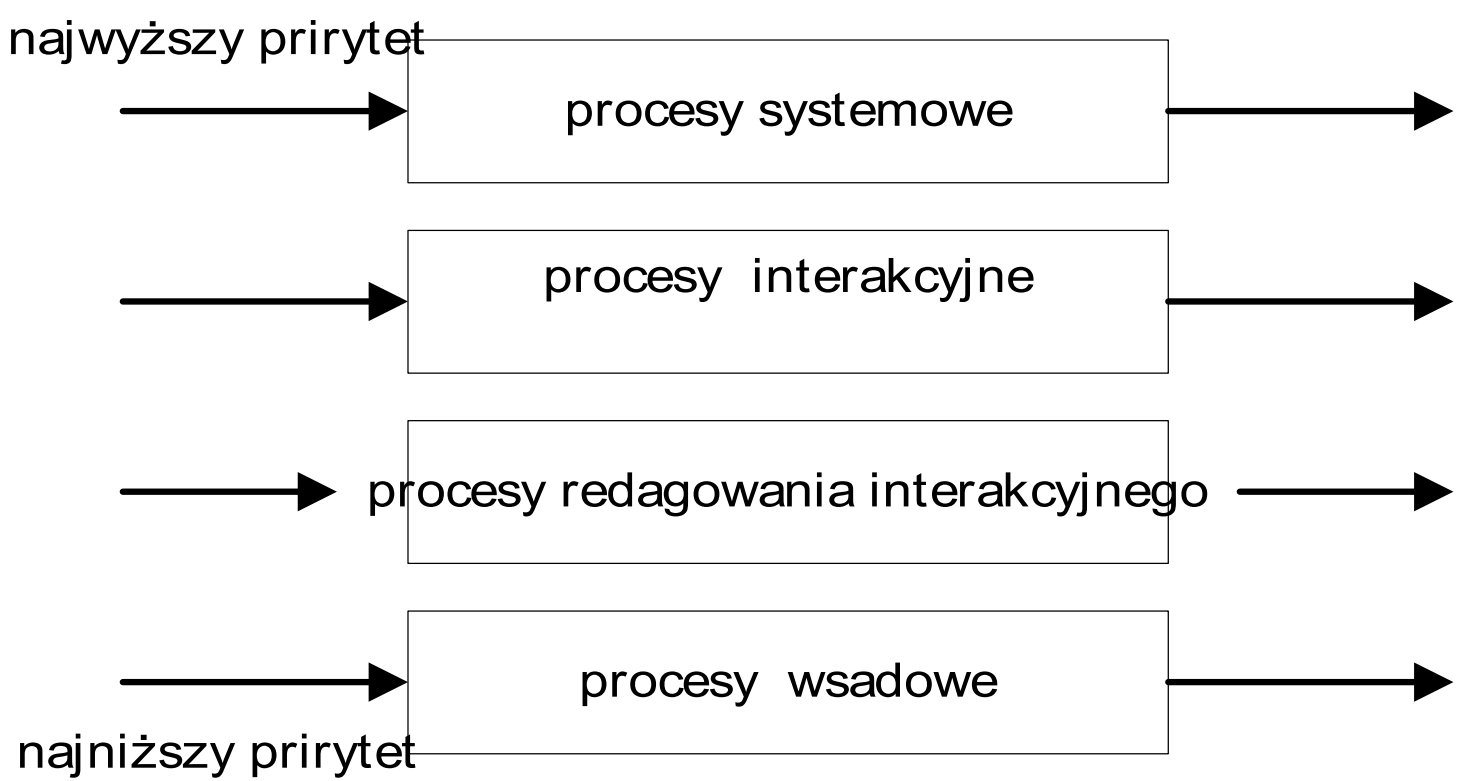

Poziom 3
kod CALL (JMP)
furtka
procedura
Poziom 2
furtka
Poziom 1
kod CALL (JMP)
procedura
Poziom 0
Jądro systemu operacyjnego
System operacyjny
procedura
procedura
kod CALL
kod CALL
furtka
furtka

Poziomy ochrony

- Często w komputerach implementuje się kilka poziomów uprzywilejowania. Pozwala to na wydzielenie kilku warstw ochrony w systemie i w efekcie - na odseparowanie systemu operacyjnego od sprzętu.
- Wprowadzenie pomiędzy oprogramowaniem systemowym i sprzętem tzw. warstwy abstrakcji sprzętu (HAL - Hardware Abstraction Layer) zwiększa przenośność oprogramowania systemowego.

Poziomy ochrony

W tym przypadku dostosowanie systemu operacyjnego do nowego komputera wymaga jedynie rekompilacji systemu i napisania nowego oprogramowania operującego na sprzętowych zasobach komputera i zapewniającego odpowiedni interfejs pomiędzy wyższą warstwą systemu i sprzętem. Taką strukturę posiada np. system operacyjny Windows NT firmy Microsoft.


najwyższy prirytet
procesy systemowe
procesy interakcyjne
procesy redagowania interakcyjnego
procesy wsadowe
najniższy prirytet

Instrukcje systemowe

Rozkazy adresowania tablic deskryptorów	
LGDT	Załadowanie rejetru globalnej tablicy deskryptorów
LLDT	Załadowanie rejetru lokalnej tablicy deskryptorów
SGDT	Umieszczenie zawartości rejestru globalnej tablicy deskryptorów w pamięci
SLDT	Umieszczenie zawartości rejestru lokalnej tablicy deskryptorów w pamięci

Instrukcje systemowe

Rozkazy związane z wielozadaniowością	
LTR	Załadowanie rejestru zadania
STR	Zapamiętanie zawartości rejestru zadania

Instrukcje systemowe

Rozkazy sterowania systemem przerwań	
LIDT	Załadowanie rejetru tablicy deskryptorów przerwań
SIDT	Umieszczenie zawartości rejestru tablicy deskryptorów przerwań w pamięci

Instrukcje systemowe

Rozkazy sterujące pracą mikroprocesora	
LMSW	Załadowanie rejestru sterującego MSW
MOV	Przesłanie danych z/do rejestru specjalnego
SMSW	Umieszczenie zawartości rejestru sterującego MSW w pamięci
HLT	Wstrzymanie pracy procesora

Instrukcje systemowe

Rozkazy weryfikacji adresowania:

ARPL	Zmiana pola RPL selektora
LAR	Umieszczenie atrybutów segmentu w rejestrze
LSL	Umieszczenie rozmiaru segmentu w rejestrze
VERR	Weryfikacja segmentu do odczytu
VERW	Weryfikacja segmentu do zapisu

1. Verification of pointer parameters (see Chapter 6):

Instruction	Description	Useful to Application?	Protected from Application?
ARPL	Adjust RPL	No	No
LAR	Load Access Rights	Yes	No
LSL	Load Segment Limit	Yes	No
VERR	Verify for Reading	Yes	No
VERW	Verify for Writing	Yes	No

2. Addressing descriptor tables (see Chapter 5):

Instruction	Description	Useful to Application?	Protected from Application?
LLDT	Load LDT Register	Yes	No
SLDT	Store LDT Register	Yes	No
LGDT	Load GDT Register	No	Yes
SGDT	Store GDT Register	No	No

3. Multitasking (see Chapter 7):

Instruction	Description	Useful to Application?	Protected from Application?
LTR	Load Task Register	No	Yes
STR	Store Task Register	Yes	No

4. Coprocessing and Multiprocessing (see Chapter 11):

Instruction	Description	Useful to Application?	Protected from Application?
CLTS	Clear TS bit in CR0	No	Yes
ESC	Escape Instructions	Yes	No
WAIT	Wait Until Coprocessor Not Busy	Yes	No
LOCK	Assert Bus-Lock	No	Can be

. Input and Output (see Chapter 8):

Instruction	Description	Useful to Application?	Protected from Application?
IN	Input	Yes	Can be
OUT	Output	Yes	Can be
INS	Input String	Yes	Can be
OUTS	Output String	Yes	Can be

6. Interrupt control (see Chapter 9):

Instruction	Description	Useful to Application?	Protected from Application?
CLI	Clear IF flag	Can Be	Can be
STI	Store IF flag	Can be	Can be
LIDT	Load IDT Register	No	Yes
SIDT	Store IDT Register	No	No

7. Debugging (see Chapter 10):

Instruction	Description	Useful to Application?	Protected from Application?
MOV	LOAD and store debug registers	No	Yes

8. System Control:

Instruction	Description	Useful to Application?	Protected from Application?
SMSW	Store MSW	No	No
LMSW	Load MSW	No	Yes
MOV	Load and Store CR0	No	Yes
HLT	Halt Processor	No	Yes