Wykład 5.
Zarządzanie pamięcią

Struktura pamięci

- pamięć -- wielka tablica oznaczona adresami słów lub bajtów
- procesor pobiera rozkazy z pamięci stosownie do wartości licznika rozkazów
- rozkazy mogą posiadać dodatkowe operacje pobrania i przechowania odnoszące się do określonych adresów
- dzielenie pamięci między procesy

Cykl wykonania rozkazu

- pobranie rozkazu z pamięci
- dekodowanie rozkazu
- ewentualne pobranie z pamięci argumentów
- wykonanie operacji
- zapis wyników do pamięci

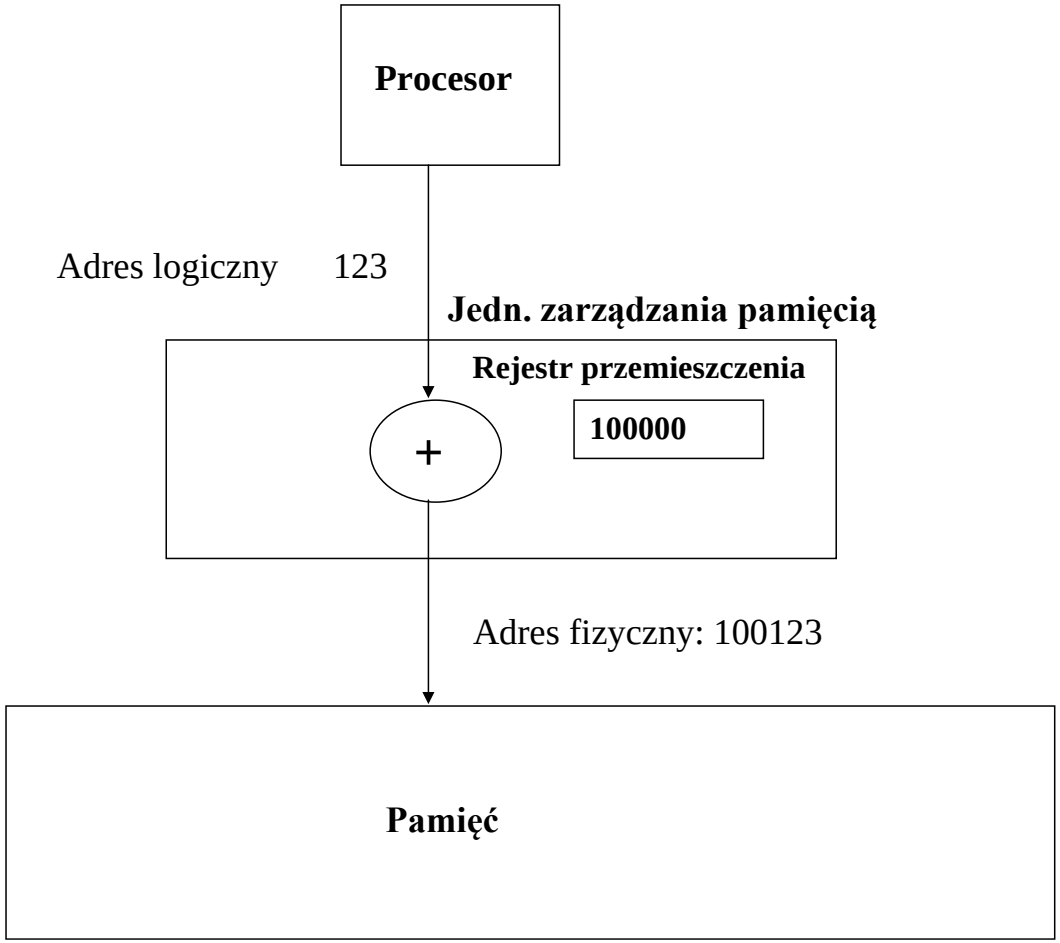
Logiczna i fizyczna przestrzeń adresowa

- **adres logiczny** (ang. logical address) -- adres wytworzony przez procesor,
- **adres wirtualny** (ang. virtual address) -- inna nazwa adresu logicznego
- **adres fizyczny** (ang. physical adress) -- rzeczywiste umiejscowienie w pamięci (adres umieszczony w rejestrze adresowym)
- **logiczna przestrzeń adresowa** (ang. logical address space) -- zbiór wszystkich adresów logicznych generowanych przez program
- **fizyczna przestrzeń adresowa** (ang. physical address space) -- zbiór wszystkich adresów fizycznych odpowiadających adresom logicznym z logicznej przestrzeni adresowej
- **jednostka zarządzania pamięcią** (ang. memory-management unit MMU) -- moduł zajmujący się odwzorowaniem adresów logicznych na fizyczne
- Przykładowe odwzorowanie wykorzystujące schemat rejestru bazowego:
- **rejestr przemieszczenia** (ang. relocation register) -- wartość tego rejestru jest dodawana do każdego adresu logicznego, co daje w rezultacie adres fizyczny.
- Uwaga: program użytkownika operuje na adresach logicznych, a nie fizycznych

Mapowanie logicznej i fizycznej przestrzeni adresowej

- **Podejście wykorzystujące rejestr przemieszczenia**
  - Adres fizyczny= wartość bazy + adres logiczny

Mapowanie logicznej i fizycznej przestrzeni adresowej (2)



Wiązanie adresów

Przygotowanie i ładowanie programu:

- kompilacja → powstaje moduł wynikowy
- konsolidacja → powstaje moduł ładowalny, reprezentowany przez plik binarny na dysku
- ładowanie przez program ładujący (łączenie modułu ładowalnego i bibliotek systemowych) → powstaje obraz binarny w pamięci
- wykonanie programu w pamięci (łączenie obrazu binarnego w pamięci i bibliotek systemowych ładowanych dynamicznie)

Wiązanie adresów → stowarzyszanie symbolicznych wyrażeń (zmiennych) występujących w programie z *adresami względnymi* (liczonymi od początku danego modułu) lub *bezwzględnymi.*

Etapy powiązania rozkazów i danych z adresami pamięci

czas kompilacji --

- kod bezwzględny (ang. absolute code) - jeśli znane jest miejsce, gdzie proces będzie przebywał w pamięci → można odwoływać się do adresów bezwzględnych (np. programy typu com w DOS)

czas ładowania --

- kod przemieszczalny (ang. relocatable code) -- kod wytwarzany przez kompilator, gdy podczas kompilacji nie jest znane umiejscowienie procesu w pamięci
  → wiązanie jest opóźnione do czasu ładowania
  → można ładować kod pod różnymi adresami zmieniając adres początkowy

czas wykonania --

- proces może ulegać przemieszczeniu w pamięci podczas swego wykonania
- konieczne jest czekanie z wiązaniem adresów aż do czasu wykonania

Ładowanie dynamiczne

- **ładowanie dynamiczne** (ang. dynamic loading) -- podprogram jest ładowany do pamięci, gdy zostaje wywołany
- podprogramy są przechowywane w postaciach przemieszczalnych na dysku

Zalety ładowania dynamicznego:

- nie zostanie załadowany do pamięci podprogram, który nie jest używany
- przydatne, gdy trzeba okazjonalnie wykonywać duże fragmenty kodu jak np. podprogramy obsługi błędów
- można jednocześnie przechowywać w pamięci tylko fragment programu
- ładowanie dynamiczne nie wymaga specjalnego wsparcia ze strony systemu operacyjnego → użytkownicy muszą się zatroszczyć o właściwą strukturę swoich programów
- systemy operacyjne dostarczają procedur bibliotecznych

Konsolidacja dynamiczna - biblioteki przyłączane dynamicznie

(ang. dynamic linked libraries), biblioteki dzielone (ang. shared libraries)

Konsolidacja statyczna -

- biblioteki systemowe są traktowane tak samo jak inne moduły wynikowe i są dołączane do binarnego obrazu programu
- w konsekwencji - wszystkie programy muszą mieć dołączone do swoich obrazów binarnych kopie bibliotek języka (lub podprogramów z tych bibliotek, które są przez nie używane)

Konsolidacja dynamiczna -

- konsolidacja zostaje opóźniona,
- w obrazie binarnych w miejscu odwołania bibliotecznego znajduje się jedynie *namiastka procedury* (ang. stub), która wskazuje jak odnaleźć odpowiedni podprogram biblioteczny w pamięci lub jak załadować odpowiednią bibliotekę
- wszystkie procesy wykorzystujące bibliotekę wykonują jedną kopie kodu biblioteki
- aktualizacja bibliotek: biblioteka może zostać zastąpiona przez nową wersję, co spowoduje, że wszystkie używające jej programy będą używać jej nowej wersji (programy ze statyczną konsolidacją musiałyby zostać skonsolidowane ponownie)
- informacje o wersji biblioteki są dołączane do kodu biblioteki i programu
- kons. dyn. wymaga wsparcia systemu operacyjnego (umożliwia dostęp wielu procesów do kodu biblioteki)

Nakładki (ang. overlays)

- w pamięci są przechowywane tylko te rozkazy i dane, które są potrzebne
- kody nakładek są przechowywane na dysku w postaci obrazów pamięci
- niezbędne są specjalne algorytmy przemieszczania i konsolidacji
- nakładki nie wymagają wsparcia systemu operacyjnego
- użycie nakładek spowalnia pracę procesu wskutek zwiększenia liczby operacji we/wy
- trudność - efektywne zaprojektowanie i zaprogramowanie struktury nakładek, konieczna jest znajomość budowy programu (kodu, struktur danych)

Wymiana

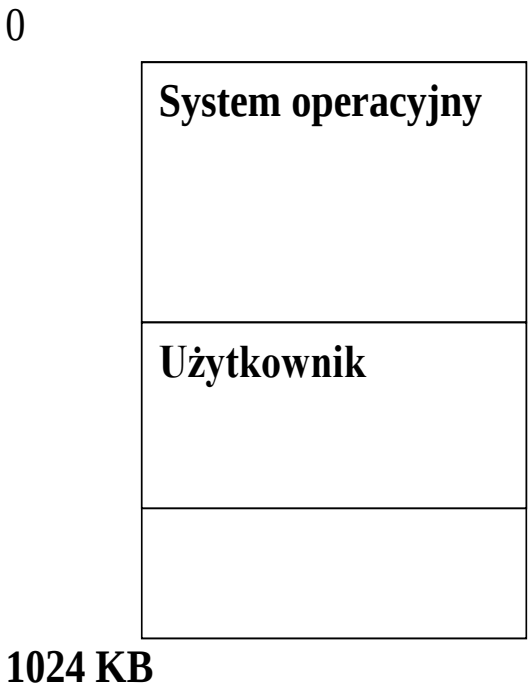
- proces wymieniany (ang. swapped) -- proces przesyłany z pamięci operacyjnej do pomocniczej i odwrotnie
- pamięć pomocnicza powinna udostępniać dostęp do przechowywanych obrazów pamięci
- Przykład: środowisko wieloprogramowe, rotacyjny algorytm planowania przydziału procesora
- po wyczerpaniu kwantu czasu, zarządca pamięci przeprowadza wymianę procesu, który zakończył działanie na proces, który zajmie jego miejsce w pamięci
- proces ulegający wymianie zazwyczaj powraca do pamięci w to samo miejsce, gdzie przebywał poprzednio
- jeśli wiązanie adresów jest wykonywane podczas tłumaczenia lub ładowania, wtedy proces nie może być przesunięty w inne miejsce
- jeśli wiązanie adresów jest dokonywane podczas wykonania, to jest możliwość wprowadzeniu procesu do innego przedziału pamięci

Wymiana (2)

- Wady wymiany: jej realizacja zawiera zbyt dużo czasu
- Obecnie rzadko stosowana jest standardowa realizacja wymiany, stosuje się jej zmodyfikowane wersje
- w systemie Unix standardowo jest zabroniona
- wymiana jest uaktywniana, gdy zajętość pamięci osiąga zadaną wartość progową

Przydział ciągły - wprowadzenie

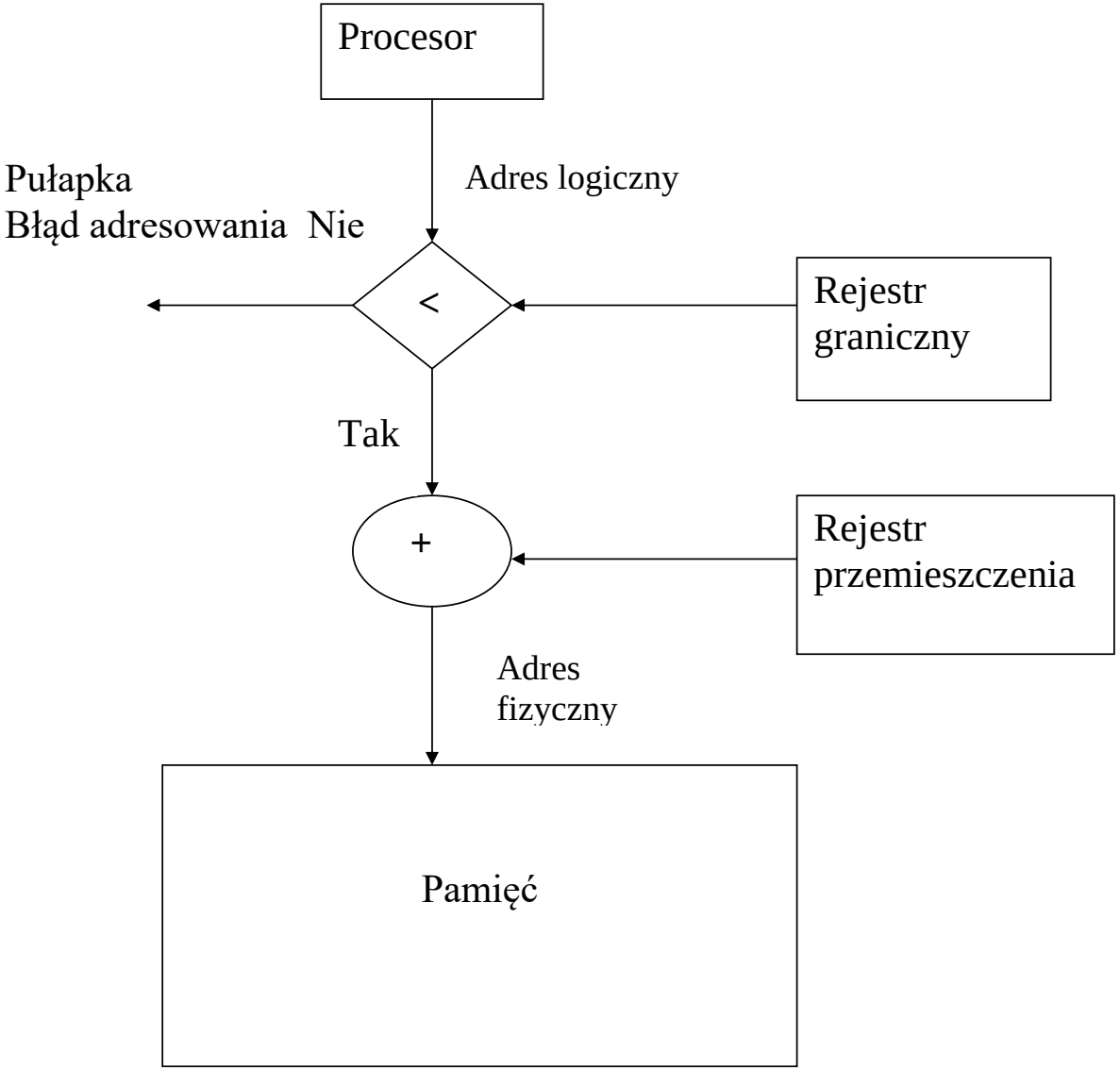
- pamięć operacyjna musi zawierać:
- system operacyjny
- procesy użytkownika
- lokalizacja systemu operacyjnego - - w pamięci dolnej lub górnej
- wpływ na decyzję wywiera lokalizacja wektora przerwań (który często znajduje się w pamięci dolnej



Przydział ciągły:Przydział pojedynczego obszaru

- ochrona danych i kodu procesów przed zmianami
- zastosowanie rejestru przemieszczenia (wartość najmniejszego adresu fizycznego) i rejestru granicznego (zakres adresów logicznych)

Przydział ciągły: Przydział pojedynczego obszaru (2)



Przydział ciągły: Przydział wielu obszarów

- w pamięci w tym samym czasie przebywa kilka procesów
- procesy opuszczają pamięć, nowe procesy przybywają do pamięci

Schematy przydziału pamięci:

- wieloprogramowość ze stałą liczbą zadań (MFT, ang. multiprogramming with with a fixed number of tasks) - podział pamięci na pewną liczbę obszarów (ang. partitions) o stałym rozmiarze, każdy obszar może zawierać dokładnie jeden proces
  - zastosowane po raz pierwszy w systemie IBM OS/360

- wieloprogramowość ze zmienną liczbą zadań (MVT, ang. multiprogramming with with a variable number of tasks)
  - uogólnienie schematu MFT
  - stosowane głównie w środowisku wsadowym
  - system operacyjny przechowuje tablicę z informacjami, które części pamięci są wolne, a które zajęte
  - ciągły obszar wolnej pamięci - dziura (ang. hole)

Przydział ciągły: Strategie przydziału wolnego obszaru: Problem dynamicznego przydziału pamięci

Pierwsze dopasowanie (ang. *first-fit*) --

- przydzielana jest pierwsza dziura o odpowiedniej wielkości
- szukanie może być rozpoczęte od początku wykazu dziur, lub od punku, w którym nastąpiło zatrzymanie
- szukanie zostaje zakończone, gdy zostanie odnaleziona odpowiednio duża dziura

Najlepsze dopasowanie (ang. *best-fit*) --

- przydziela się najmniejszą z dostatecznie dużych dziur
- w poszukiwaniu odpowiedniej dziury, trzeba przejrzeć całą ich listę, o ile nie są one uporządkowane wg. rozmiarów
- zapewnia najmniejsze pozostałości po przydziale

Najgorsze dopasowanie (ang. *worse-fit*) --

- przydziela się największą dziura
- trzeba przeszukać całą listę dziur, o ile nie są one uporządkowane wg. wymiarów
- pozostawia największą dziurę, która może w przyszłości być bardziej użyteczna niż mała pozostałość po najlepszym dopasowaniu

Przydział ciągły:Fragmentacja wewnętrzna i zewnętrzna

Zewnętrzna fragmentacja (ang. external fragmentation)

- wraz z ładowaniem i usuwaniem procesów z pamięci przestrzeń wolnej pamięci zostaje podzielona na małe kawałki
- suma wolnej pamięci wystarcza na spełnienie zamówienia, ale wolne obszary nie stanowią spójnej całości
- analiza statystyczna przewiduje, że w ten sposób 1/3 pamięci może być bezużyteczna (reguła 50 procent, ang. 50-percent rule)

Wewnętrzna fragmentacja (ang. internal fragmentation)

- trzymanie informacji o b. małych dziurach jest nieopłacalne (niezbędne informacje o dziurze w skrajnych przypadkach mogłyby przekraczać jej wielkość)
- wyjściem jest przyłączanie bardzo małych dziur do większych przedziałów -- wewnętrzna fragmentacja
- wewnętrzna fragmentacja powoduje powstawanie bezużytecznych obszarów pamięci wewnątrz przydzielonych obszarów

Upakowanie pamięci

- rozwiązanie problemu zewnętrznej fragmentacji
- takie przemieszczenie zawartości pamięci, by otrzymać całą wolną pamięć w jednym bloku
- przemieszczenie proc. wymaga zmiany ich wewn. adresów

Stronicowanie (ang. paging)

- inna metoda rozwiązania problemu zewnętrznej fragmentacji
- dopuszcza nieciągłości logicznej przestrzeni adresowej procesu

Stronicowanie (ang. paging)

- inna metoda rozwiązania problemu zewnętrznej fragmentacji
- dopuszcza nieciągłości logicznej przestrzeni adresowej procesu

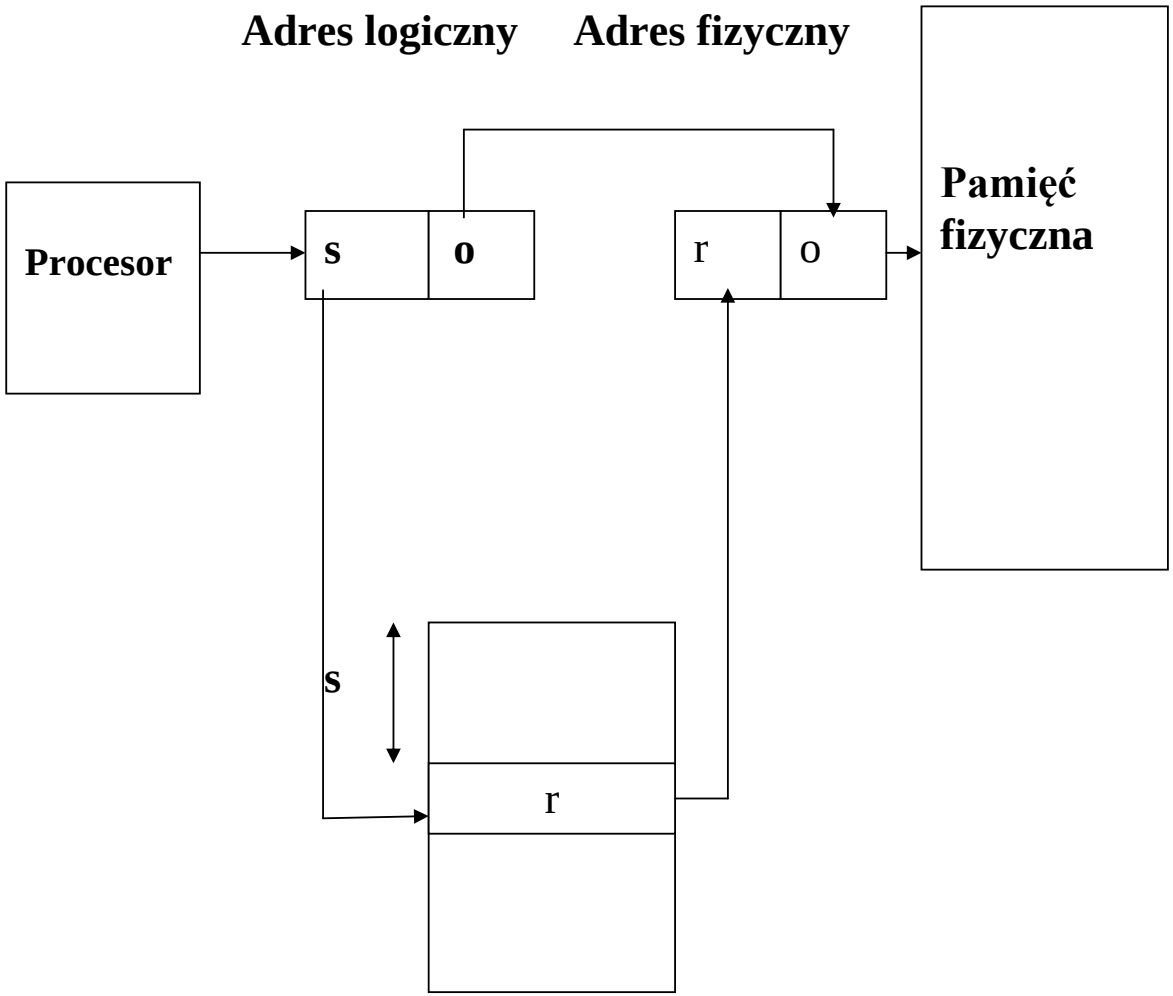
Metoda podstawowa

- pamięć fizyczna dzieli się na bloki stałej długości zwane ramkami (ang. frames)
- pamięć logiczna jest podzielona na bloki o tej samej długości zwane stronami (ang. pages)
- pamięć pomocnicza jest podzielona na bloki o rozmiarze ramek w pamięci operacyjnej
- podczas rozpoczęcia wykonania procesu, strony przebywające w pamięci pomocniczej są wprowadzane w dowolne ramki pamięci operacyjnej
- Adres logiczny składa się z pary (s,o)
  - s -- numer strony (ang. page number), używany jako indeks w tablicy stron
  - o -- odległość na stronie (ang. page offset)

Tablica stron: zawiera adresy bazowe wszystkich stron w pamięci operacyjnej

- rozmiar strony/ramki - określany przez sprzęt
- zwykle jest to potęga 2 - od 512B do 16MB na stronę
- taki wybór ułatwia tłumaczenie adresu logicznego na parę (numer strony, odległość na stronie)

Stronicowanie



Tablica stron

Stronicowanie. Uwagi.

- korzystając ze stronicowania eliminuje się zewnętrzną fragmentację
- stronicowanie nie eliminuje wewnętrznej fragmentacji, straty z nią związane mogą osiągać maksymalnie : rozmiar ramki - 1 słowo/bajt na proces
- mniejszy rozmiar strony zmniejsza wewnętrzną fragmentację, ale zwiększa koszty związane z obsługą tablicy stron
- obecny rozmiar typowej strony: 2-4 KB
- tablica stron procesu zawiera numery ramek przydzielonych danemu procesowi
- stronicowanie oddziela pamięć oglądaną przez użytkownika od pamięci fizycznej
- system operacyjny potrzebuje informacji na temat stanu pamięci fizycznej (wolne i zajęte ramki) → informacje te są przechowywane w tablicy ramek (ang. frame table)
  - tablica ramek ma pozycje dla każdej ramki i zawiera informację:
    - czy ramka jest zajęta. czy wolna
    - jeśli jest zajęta, to do jakiego procesu (lub procesów) jest przydzielona

Przykładowe rozmiary stron

Computer	Page Size
Atlas	512 48-bit words
Honeywell-Multics	1024 36-bit words
IBM 370/XA and 370/ESA	4 Kbytes
VAX family	512 bytes
IBM AS/400	512 bytes
DEC Alpha	8 Kbytes
MIPS	4 Kbytes to 16 Mbytes
UltraSPARC	8 Kbytes to 4 Mbytes
Pentium	4 Kbytes or 4 Mbytes
IBM POWER	4 Kbytes
Itanium	4 Kbytes to 256 Mbytes

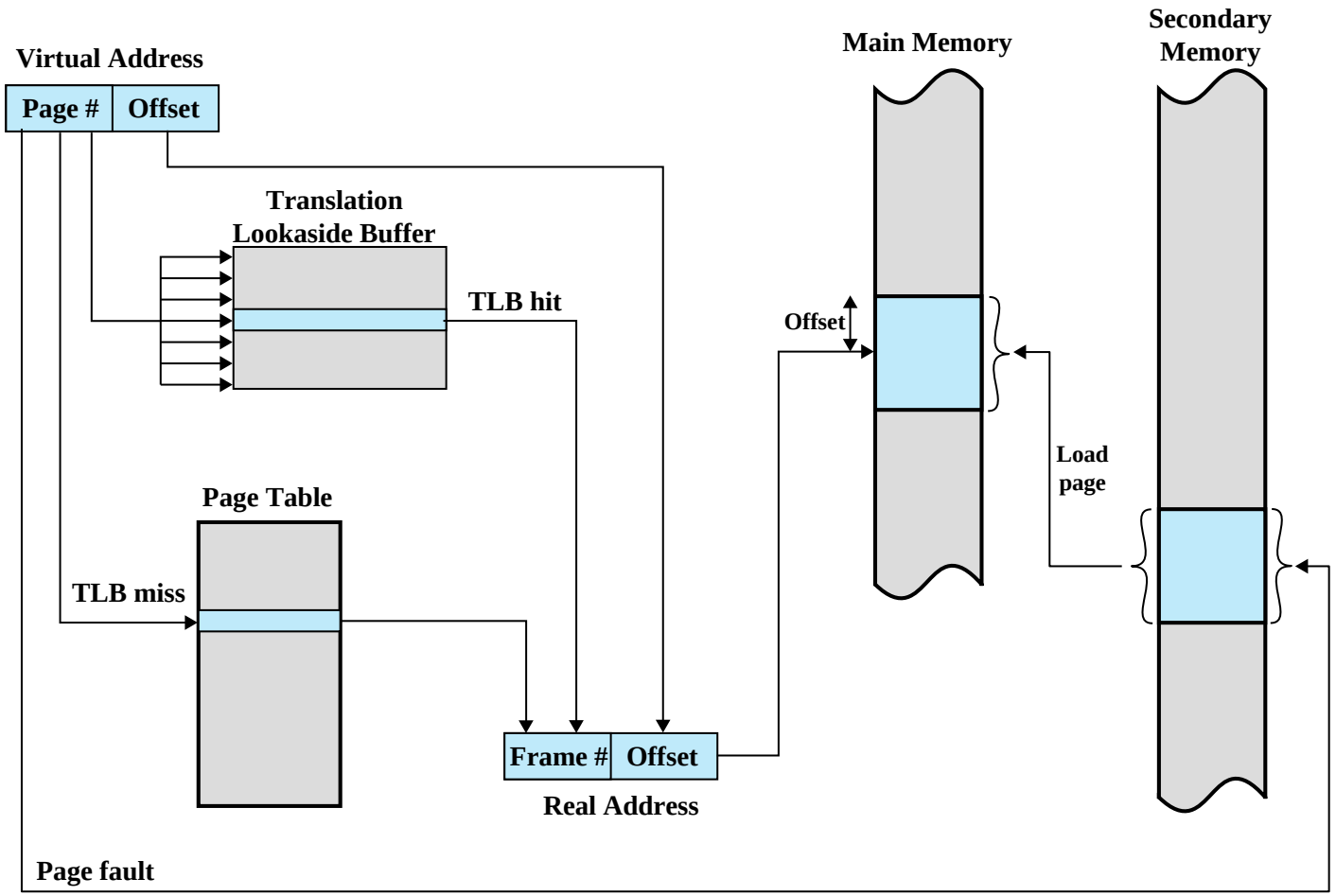
Struktura tablicy stron. Wspomaganie sprzętowe.

- dostęp procesu do tablicy stron: w bloku kontrolnym procesu znajduje się wskaźnik do tablicy stron

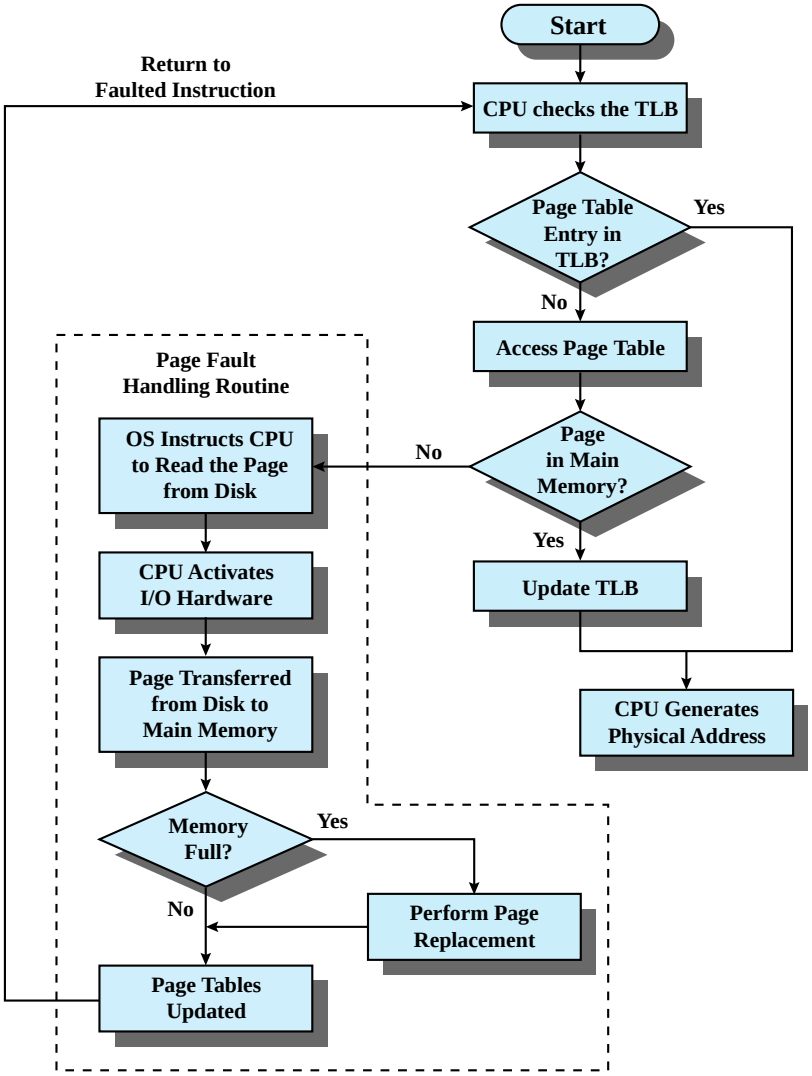
Sposoby realizacji sprzętowej tablicy stron:

- jako zbiór rejestrów specjalnego przeznaczenia o wielkiej szybkości działania → wystarczające dla małej tablicy stron
- tablica stron jest przechowywana w pamięci operacyjnej, jej położenie wskazuje rejestr bazowy tablicy stron (ang. page-table base register, PTBR), takie rozwiązanie wymaga dwóch kontaktów z pamięcią w celu odwołania się do danej
- rozwiązaniem powyższego problemu jest zastosowanie specjalnej sprzętowej pamięci podręcznej -- rejestrów asocjacyjnych (ang. associative registers) lub buforów translacji adresów stron (ang. translation look-aside buffers, TLBs)
  - każdy rejestr jest złożony z klucza i wartości
  - porównywanie obiektu z kluczami odbywa się jednocześnie dla wszystkich kluczy

Zastosowanie bufora translacji adresów (bufora TLB)



Działania stronicowania i TLB



Struktura tablicy stron. Ochrona.

- **bity ochrony -** przypisane każdej ramce
- zazwyczaj znajdują się w tablicy stron
- bit określa stronę jako: dostępną do czytania i pisania, lub przeznaczoną wyłącznie do czytania
- każdy wpis do tablicy stron może być uzupełniony o bit poprawności (ang. valid-invalid bit), który oznacza:
  - stan „*poprawne*” -- strona, z którą jest związany znajduje się w logicznej przestrzeni adresowej procesu
  - stan „*niepoprawne*” -- strona nie należy do logicznej przestrzeni adresowej procesu

Stronicowanie wielopoziomowe

- tablica stron może być b. duża (~ milion wpisów)
- podział tablicy stron na mniejsze części, by uniknąć pełnego ciągłego obszaru w pamięci operacyjnej

- Realizacja:
  - stronicowanie dwupoziomowe -- tablica stron jest podzielona na strony

Odwrócona tablica stron (ang. inverted page table)

- z każdym procesem jest związana tablica stron
- duży rozmiar tablicy stron

Struktura odwróconej tablicy stron:

- jedna pozycja dla każdej rzeczywistej strony pamięci (ramki)
- każda pozycja zwiera adres wirtualny strony przechowywanej w ramce rzeczywistej pamięci, oraz informacje o procesie, do którego strona należy
- w systemie istnieje jedna tablica stron, mająca po jednej pozycji dla każdej strony pamięci fizycznej
- rozwiązanie zmniejsza rozmiar pamięci potrzebnej do pamiętania tablic stron, ale zwiększa czas przeszukania tablicy przy odwołaniu do strony

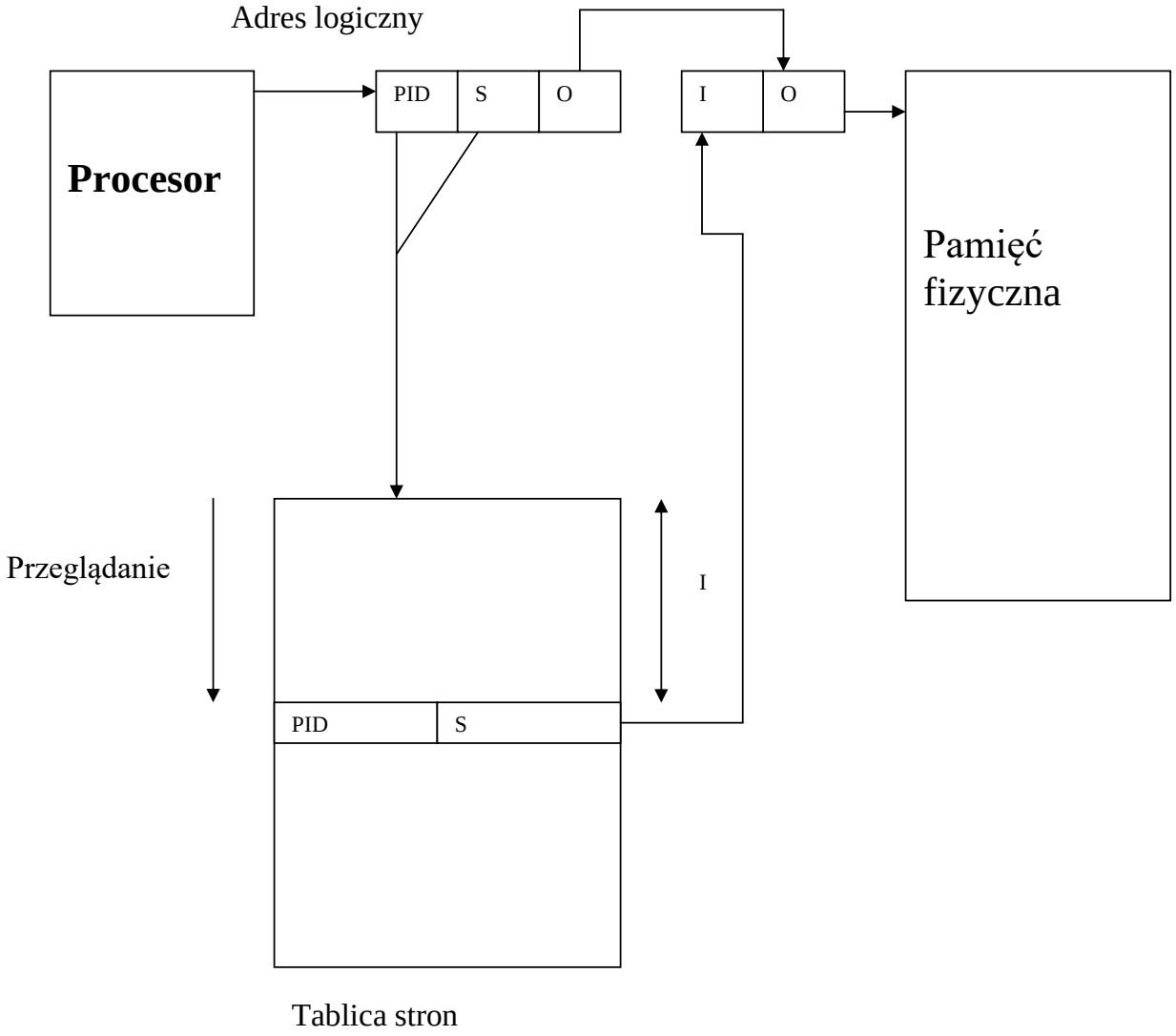
- Adres wirtualny stanowi trójka:
  <identyfikator-procesu, numer-strony, odległość>
- Wpis w odwróconej tablicy stron jest parą:
  <identyfikator-procesu, numer-strony>

Odwrócona tablica stron (ang. inverted page table) (2)

Realizacja odwołania do pamięci:

- część adresu wirtualnego <identyfikator-procesu, numer-strony> jest przekazana podsystemowi pamięci
- przeszukanie odwróconej tablicy stron w celu dopasowania adresu -> znajduje się i-ty element tablicy
- tworzy się adres fizyczny <i, odległość>

Odwrócona tablica stron (ang. inverted page table) (3)



Strony dzielone

- stronicowanie umożliwia dzielenie wspólnego kodu
- kod wznawialny (lub kod czysty) (ang. *reentrant*) -- kod, który nie modyfikuje sam siebie
- kod wznawialny nie może zmienić się podczas wykonania
- kilka procesów może wykonywać ten kod w tym samym czasie

Segmentacja (ang. segmentation) - wstęp

- oddzielenie sposobu widzenia pamięci przez użytkownika
- od rzeczywistej pamięci fizycznej
- użytkownik/programista raczej nie traktuje pamięci jako liniowej tablicy bajtów, ale jako zbiór segmentów o zróżnicowanym wymiarze,
- segmenty nie muszą być uporządkowane wzajemnie
- elementy wewnątrz segmentu są identyfikowane za pomocą ich odległości od początku segmentu

- program z punktu widzenia użytkownika składa się np. z:
  - podprogramów/funkcji
  - stosu
  - tablicy symboli
  - programu głównego itd.

Segmentacja (ang. segmentation) - wstęp

- Segmentacja -- schemat zarządzania pamięcią, w którym:
  - przestrzeń adresów logicznych jest zbiorem segmentów
  - każdy segment ma swoją nazwę i długość
  - adresy logiczne określają nazwę segmentu i odległość wewnątrz segmentu

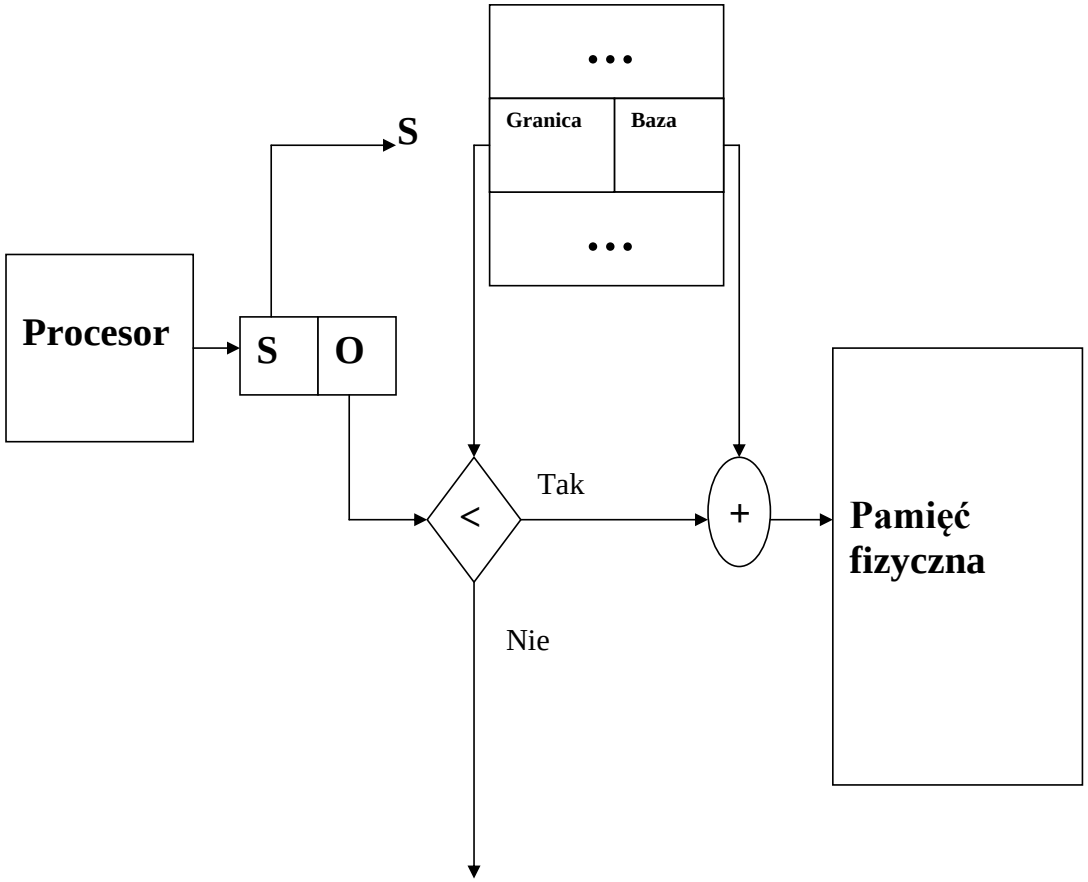
- różnica między stronicowaniem i segmentacją:
  - segmentacja: użytkownik określa każdy adres za pomocą nazwy segmentu i jego długości
  - stronicowanie: użytkownik określa tylko pojedynczy adres, który jest dzielony przez sprzęt na numer strony i odległość w sposób niewidoczny dla użytkownika
  - segmenty są numerowane,

- adres logiczny : <numer-segmentu, odległość>

Segmentacja: Adresowanie

- Realizacja odwzorowania dwuwymiarowych adresów logicznych na jednowymiarowe adresy fizyczne
- zastosowanie tablicy segmentów (ang. *segment table*) każda pozycja w tablicy segmentów składa się z bazy segmentu i granicy segmentu
  - baza segmentu -- adres fizyczny początku segmentu w pamięci operacyjnej
  - granica segmentu -- długość segmentu
- adres logiczny składa się z:
  - numeru segmentu (s), numer segmentu jest indeksem w do tablicy segmentów
  - odległości w segmencie (o), która nie może przekraczać granicy segmentu
  - idea segmentacji jest związana z modelem zarządzania obszarami pamięci
- podstawowa różnica -- jeden program może się składać z kilku segmentów

Realizacja segmentacji



Pułapka.
Błąd adresowania

Implementacja tablica segmentów

- tablica segmentów może być umieszczona w szybkich rejestrach lub pamięci operacyjnej (podobnie jak tablica stron)

- tablica segmentów w pamięci
  - rejestr bazowy tablicy segmentów (ang. *segment-table base register* - STBR) wskazuje na początek tablicy segmentów
  - rejestr długości tablicy segmentów (ang. *segment table length register* - STLR)

- koszt czasowy
  - odwzorowanie adresów poprzez tablicę segmentów wymaga dwóch odwołań do pamięci dla każdego adresu logicznego
  - dla przyspieszenia operacji stosuje się zbiór rejestrów asocjacyjnych, w których są przechowywane ostatnie używane pozycje tablicy segmentów

Segmentacja: Ochrona i wspólne użytkowanie

Ochrona

- zaleta segmentacji: powiązanie ochrony pamięci z segmentami
  - segmenty są określonymi semantycznie fragmentami programów, można zakładać, że wszystkie dane w segmencie będą wykorzystywane w taki sam sposób: segmenty kodu i segmenty danych
  - z każdą pozycją w tablicy segmentów są związane bity ochrony,
    - segment do odczytu/modyfikowania
    - segment danych/do wykonywania

Dzielenie danych

- dzielenie danych występuje na poziomie segmentów
- informacja może być dzielona, jeśli została zdefiniowana jako segment
- dzielone segmenty kodu muszą być określane przez ten sam numer przez wszystkie procesy z nich korzystające

Segmentacja: Fragmentacja

- planista długoterminowy przydziela pamięć wszystkim segmentom programu użytkownika

- różnica w porównaniu ze stronicowaniem -- segmenty mają zmienne rozmiary, w przeciwieństwie do stron

- segmentacja może spowodować zewnętrzną fragmentację, jeśli każdy z bloków wolnej pamięci jest za mały, by pomieścić cały segment → trzeba czekać na zwolnienie pamięci lub zastosować upakowanie

- znaczenie wpływu zewnętrznej fragmentacji zależy od rozmiarów segmentów

Segmentacja ze stronicowaniem: System OS/2

- System operacyjny IBM OS/2 w wersji 32-bitowej działa na procesorze typu Intel 386 (i późniejszych)

- Do zarządzania pamięcią zastosowano segmentację ze stronicowaniem
  - maks. liczba segmentów w jednym procesie wynosi 16K
  - każdy segment może mieć do 4GB
  - rozmiar strony wynosi 4KB

Segmentacja ze stronicowaniem. System OS/2

- Podział przestrzeni adresów logicznych procesu na dwie:
  - do 8K segmentów prywatnych segmentów procesu, informacje przechowywane w *tablicy lokalnych deskryptorów* (ang. *local descriptor table* - LDT)
  - do 8K segmentów wspólnych dla wszystkich procesów - informacje przechowywane w *tablicy globalnych deskryptorów* (ang. *global descriptor table* - GDT)
- pozycje w tablicach LDT i GDT mają po 8 bajtów i zawiera informacje o konkretnych segmentach (m.in. adres początku i długość)

Segmentacja ze stronicowaniem. System OS/2 (2)

Adres logiczny: para (selektor, odległość):

- selektor (16 bitowy), dzieli się na:
  - s (13 bitów) - numer segmentu
  - g (1 bit) - czy segment jest w tablicy LDT czy GDT
  - p (2 bity) - dotyczy ochrony
- odległość (32 bity) - określa położenie bajtu (słowa) w adresowanym segmencie

- 6 rejestrów segmentów w procesorze - proces może jednocześnie adresować do 6 segmentów
- 6 rejestrów mikroprogramowych o rozmiarach 8-bajtów, które służą do przechowywania deskryptorów z tablic LDT i GDT
  - ta pamięć podręczna eliminuje konieczność czytania deskryptora z pamięci podczas każdego do niej odwołania

Tłumaczenie adresu w procesorze Intel 80386

- adres fizyczny - 32 bity

- Tworzenie adresu fizycznego:
  - rejestr segmentu wskazuje na odpowiednią pozycję w tablicy deskryptorów lokalnych lub globalnych
  - adres liniowy (ang. linear address) jest tworzony na podstawie adresu początku segmentu (bazowego) i jego długości
  - Najpierw sprawdza się poprawność adresu ze względu na granicę segmentu, jeśli adres jest niepoprawny, to generowany jest błąd pamięci
  - Jeśli adres jest poprawny, to wartość odległości dodaje się do wartości bazowej, co daje w wyniku liniowy adres 32-bitowy

Tłumaczenie adresu w procesorze Intel 80386 (2)

- Każdy segment jest stronicowany, każda strona ma 4KB
  - Tablica stron mogłaby zawierać do 1 miliona pozycji, ponieważ każda pozycja zajmuje 4B, proces mógłby potrzebować do 4MB fizycznej przestrzeni adresowej na tablicę stron

- tablica stron mogłaby być zbyt duża, dlatego:
  - stronicowanie dwupoziomowe**:**
  - adres liniowy podzielono na 20-bitowy numer strony i 12-bitową odległość na stronie
  - tablica stron podlega stronicowaniu, numer strony jest dzielony na:
    - 10-bitowy wskaźnik do katalogu stron
    - 10-bitowy wskaźnik do tablicy stron

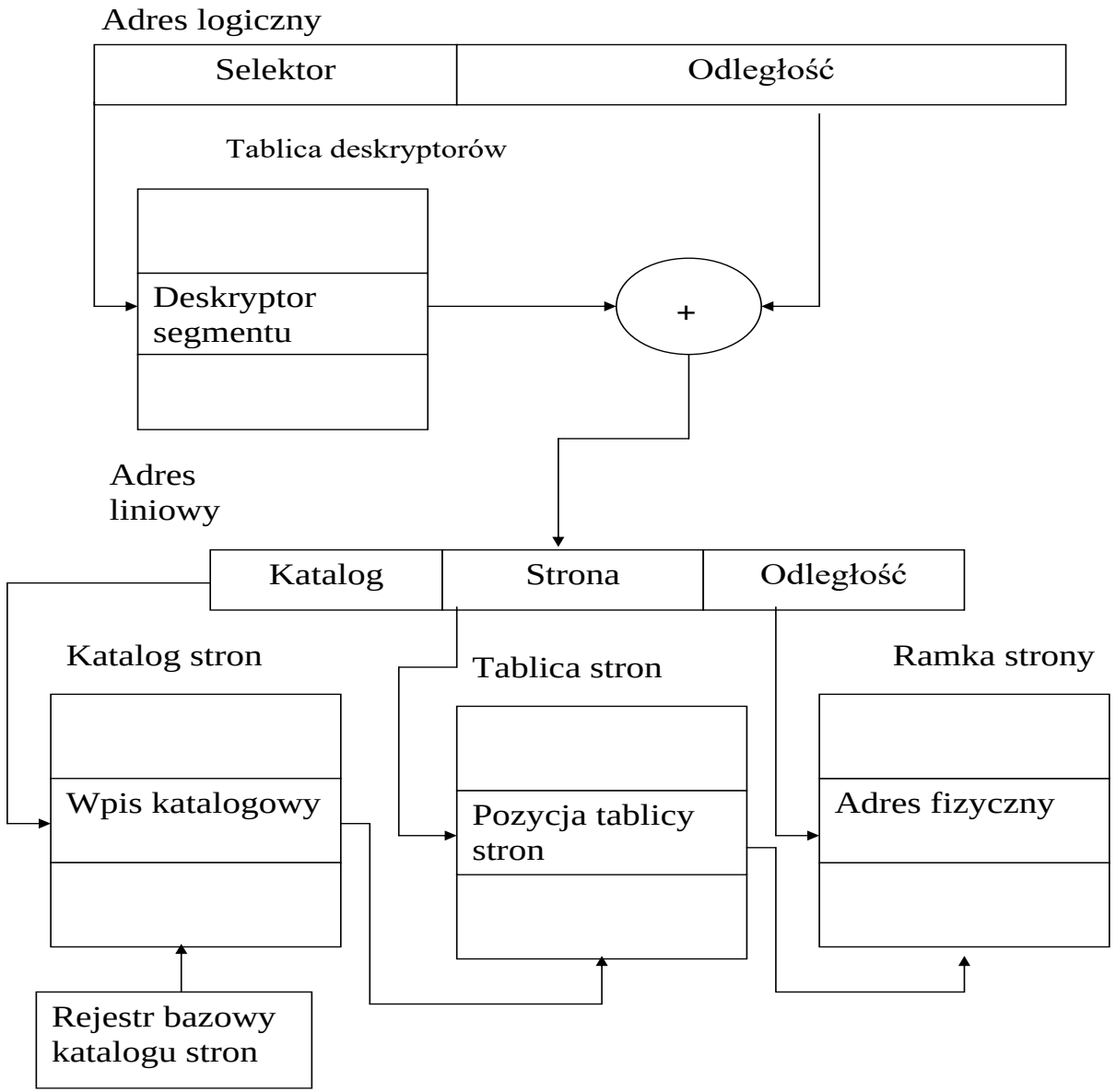
- Uzyskana postać adresu logicznego
  - Wskaźnik do katalogu stron (10 bitów)
  - Wskaźnik do tablicy stron (10 bitów)
  - Odległość na stronie (12 bitów)

Intel 386: Zarządzanie pamięcią tablicy stron

- W celu polepszenia wydajności użytkowania pamięci fizycznej tablice stron procesora Intel 386 można wysłać na dysk

- W pozycjach katalogu stron stosuje się bit poprawności, by zaznaczyć, czy tablica, na którą pozycja wskazuje jest w pamięci operacyjnej czy na dysku
  - Jeśli tablica jest na dysku, to system operacyjny może wykorzystać pozostałe 32 bity do określenia położenia tablicy na dysku
  - Tablicę można sprowadzić na żądanie do pamięci operacyjnej

Tłumaczenie adresu w procesorze Intel 80386



Stronicowanie a segmentacja: porównanie

Problem	Stronicowanie	Segmentacja
Programista powinien brać pod uwagę, że dana technika jest w użyciu	Nie	Tak
Ilość linearnych przestrzeni adresowych	1	Wiele
Pełna przestrzeń adresowa może przekraczać rozmiar pamięci fizycznej	Tak	Tak
Procedury i dane mogą być rozróżniane i oddzielnie chronione	Nie	Tak
Tablice o zmiennym rozmiarze mogą być łatwo realizowane	Nie	Tak
Czy jest wspierane dzielenie procedur między użytkownikami	Nie	Tak
Przyczyna wprowadzenia techniki	Uzyskanie większej linearnej przestrzeni adresowej bez konieczności użycia większej pamięci logicznej	Umożliwienie podziału programów i danych na niezależne przestrzenie adresowe, wspomaganie współdzielenia i ochrony