Zarządzanie pamięcią

Wykonywać można jedynie program umieszczony w pamięci głównej.

Wiązanie instrukcji i danych z adresami w pamięci może się odbywać w czasie:

- *kompilacji*: jeśli są znane a priori adresy w pamięci, to generuje się *kod absolutny*; zmiana położenia kodu w pamięci wymaga jego rekompilacji
- *ładowania*: trzeba generować *kod przemieszczalny*
- *wykonania*: jeśli proces może się przemieszczać w pamięci podczas wykonania, to trzeba wiązać *dynamicznie*; wymaga specjalnego sprzętu

Dynamiczne ładowanie - podprogram ładuje się do pamięci dopiero po wywołaniu

Dynamiczne łączenie - łączenie opóźnione do czasu wykonania; zazwyczaj dotyczy bibliotek systemowych

Nakładki - w pamięci przechowuje się tylko te części przestrzeni adresowej, które są niezbędne w danej chwili; umożliwia wykonywanie programów większych od przydzielonego im obszaru pamięci; implem. przez użytk.

Logiczna i fizyczna przestrzeń adresowa:

- *adresy logiczne* (*wirtualne*): generowane przez CPU
- *adresy fizyczne*: widziane przez jednostkę pamięci
- adresy logiczne i fizyczne są:
  - takie same, gdy adresy wiąże się w czasie kompilacji i ładowania
  - różne, gdy adresy wiąże się w czasie wykonania

Jednostka zarządzająca pamięcią (MMU): urządzenie sprzętowe, które odwzorowuje adresy wirtualne na fizyczne

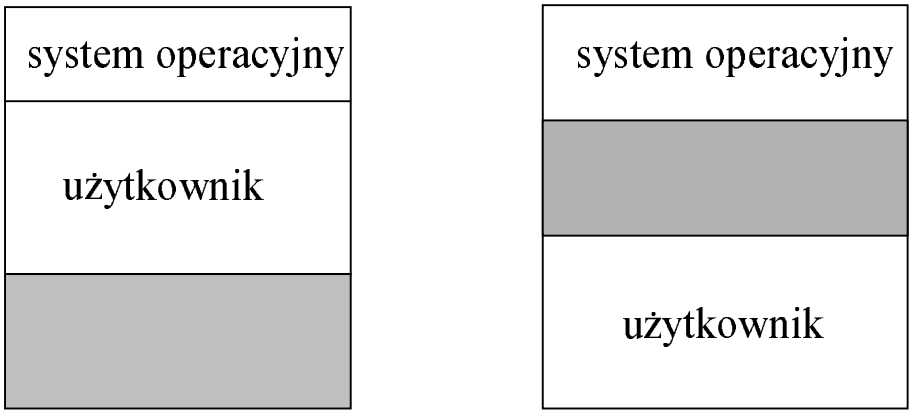
Wymiana (swapping):

- Proces może zostać czasowo wysłany z pamięci głównej do zewnętrznej, a po jakimś czasie sprowadzony ponownie do pamięci głównej
- Jako pamięć zewnętrzna na potrzeby wymiany służy duży szybki dysk z dostępem bezpośrednim
- Główny narzut: czas transmisji
- Różne wersje wymiany są realizowane w wielu systemach, np. w Unixie czy Microsoft Windows

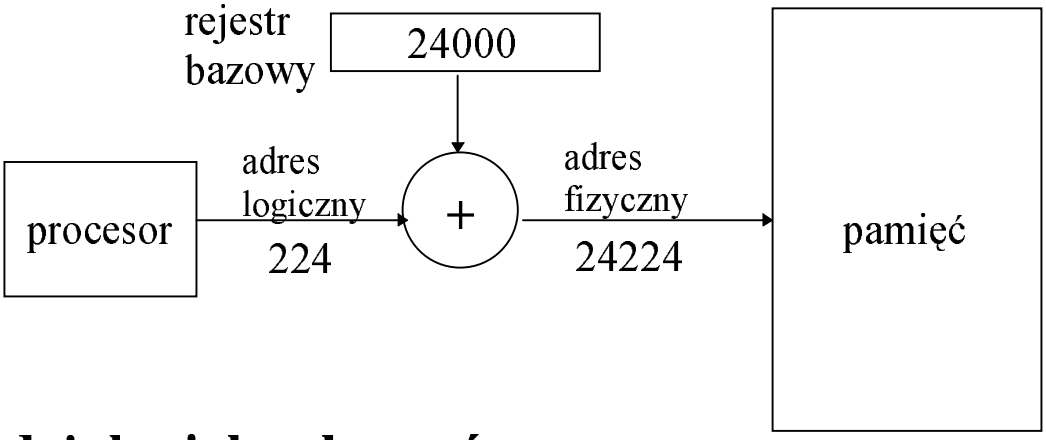
Alokacja ciągła

Przydział pojedynczego obszaru

- Pamięć podzielona na dwa obszary: dla rezydującego systemu operacyjnego i dla użytkownika
- Rejestr *bazowy* (*relokacji*) i *graniczny* służą do ochrony oraz do dynamicznej translacji adresów logicznych na fizyczne



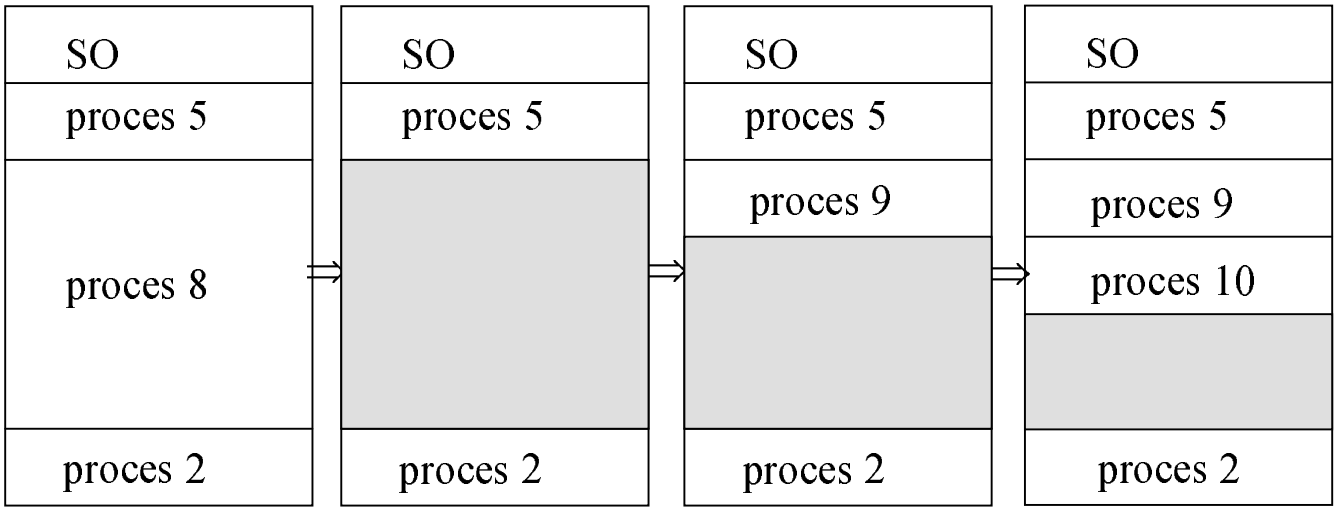
Np. IBM PC: BIOS (programy sterujące urządzeń) w ROMie (górne adresy), SO w RAMie (dolne adresy)



Przydział wielu obszarów

W systemach wieloprogramowych w pamięci równocześnie przebywa wiele programów, każdy we własnym obszarze

Metoda stref statycznych: pamięć jest na stałe podzielona na obszary o ustalonej wielkości (np. IBM OS/360 MFT)
Metoda stref dynamicznych: liczb i wielkość stref zmienia się dynamicznie (np. IBM OS/360 MVT)
Dziura: wolny obszar pamięci; dziury różnych rozmiarów są rozrzucone po pamięci



Strategia wyboru wolnego obszaru:

- *Pierwszy pasujący* (*first fit*): przydziela się pierwszy obszar o wystarczającej wielkości (*następny pasujący*: szukanie rozpoczyna się od miejsca, w którym ostatnio zakończono szukanie); z reguły krótszy czas szukania

- *Najlepiej pasujący* (*best fit*): przydziela się namniejszy z dostatecznie dużych obszarów; wymaga przeszukania całej listy (o ile nie jest uporządkowana wg rozmiaru); pozostająca część obszaru jest najmniejsza
- *Najgorzej pasujący* (*worst fit*): przydziela się największy obszar; wymaga przeszukania całej listy; pozostająca część obszaru jest największa
- Pierwszy pasujący i najlepszy pasujący lepsze w terminach czasu i wykorzystania pamięci

Fragmentacja zewnętrzna: suma wolnych obszarów w pamięci wystarcza na spełnienie żądania, lecz nie stanowi spójnego obszaru

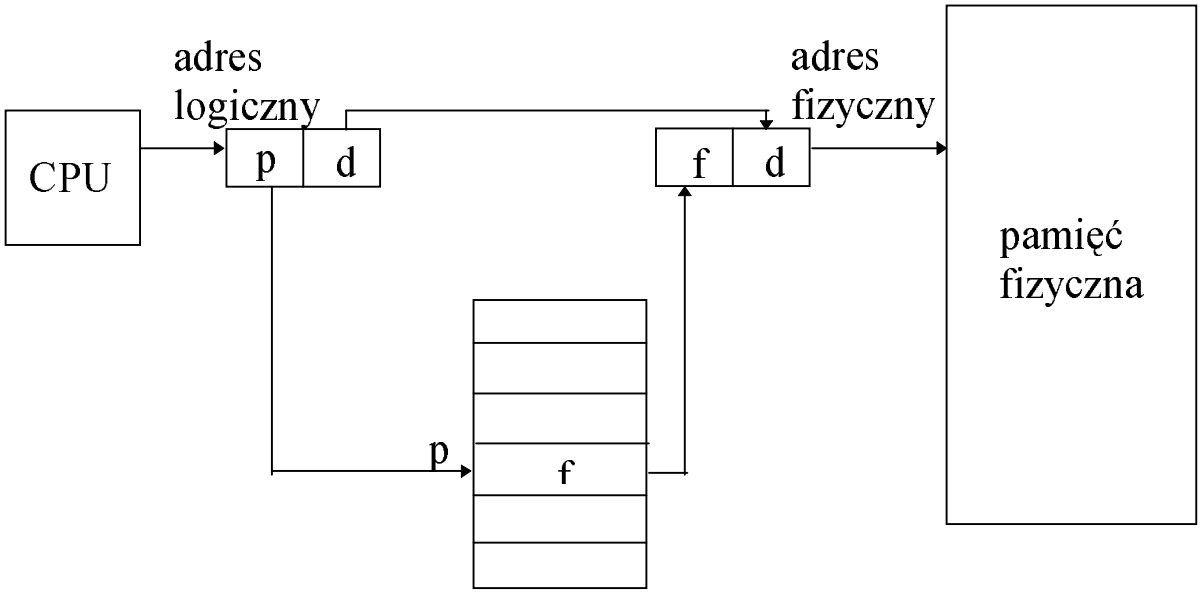
Fragmentacja wewnętrzna: przydzielona pamięć może być nieco większa niż żądana. Różnica między tymi wielkościami znajduje się wewnątrz przydzielonego obszaru, lecz nie jest wykorzystywana

Redukowanie fragmentacji zewnętrznej poprzez kompresję: umieszczenie całej wolnej pamięci w jednym bloku; możliwe tylko w przypadku, gdy adresy są wiązane w czasie wykonania; różne strategie kompresji; kompresja a operacje wejścia-wyjścia

Stronicowanie

- Pamięć fizyczna jest podzielona na bloki jednakowego rozmiaru, zwane *ramkami* (rozmiar jest potęgą 2, między 0.5 K bajtów a 8 K bajtów)
- Logiczna przestrzeń adresowa procesu jest podzielona na bloki o rozmiarze takim jak ramki, zwane *stronami*

- Strony przebywają w pamięci pomocniczej, są sprowadzane do pamięci głównej i umieszczane w wolnych ramkach (program o *n* stronach potrzebuje *n* ramek); strony *nie muszą zajmować ciągłego* obszaru fizycznego
- Fragmentacja wewnętrzna, brak fragmentacji zewnętrznej (małe ramki to mniejsza fragmentacja wewnętrzna, ale większy narzut na tablice stron)
- System przechowuje informacje o wolnych ramkach
- Tablica stron procesu służy do odwzorowywania adresów logicznych w adresy fizyczne
- Adres logiczny składa się z dwóch części:
  numer strony (p) - służy jako indeks w tablicy stron
  przesunięcie w stronie (d) - numer bajtu w stronie



tablica stron

Implementacja tablicy stron:

- Tablica stron jest przechowywana w pamięci głównej
- *Rejestr bazowy tablicy stron* wskazuje jej początek
  Każdy dostęp do danych/instrukcji programu wymaga **dwóch** dostępów do pamięci głównej

Przyśpieszenie translacji adresu jest możliwe dzięki zastosowaniu *rejestrów asocjacyjnych* (zwanych także *buforami translacji bliskiego otoczenia*, ang. *TLB*)

numer strony (klucz)	numer ramki (wartość)

Porównywanie danej pozycji z kluczami w rejestrach asocjacyjnych odbywa się równocześnie dla wszystkich kluczy (szybkie, ale drogie). Wynik: pole wartości. Jeśli nie ma, to trzeba zajrzeć do tablicy stron

Współczynnik trafień (ang. *hit ratio*) - procent numerów stron znajdowanych w rejestrach asocjacyjnych (zależy od liczby rejestrów)

Efektywny czas dostępu do pamięci:

współczynnik trafień = α

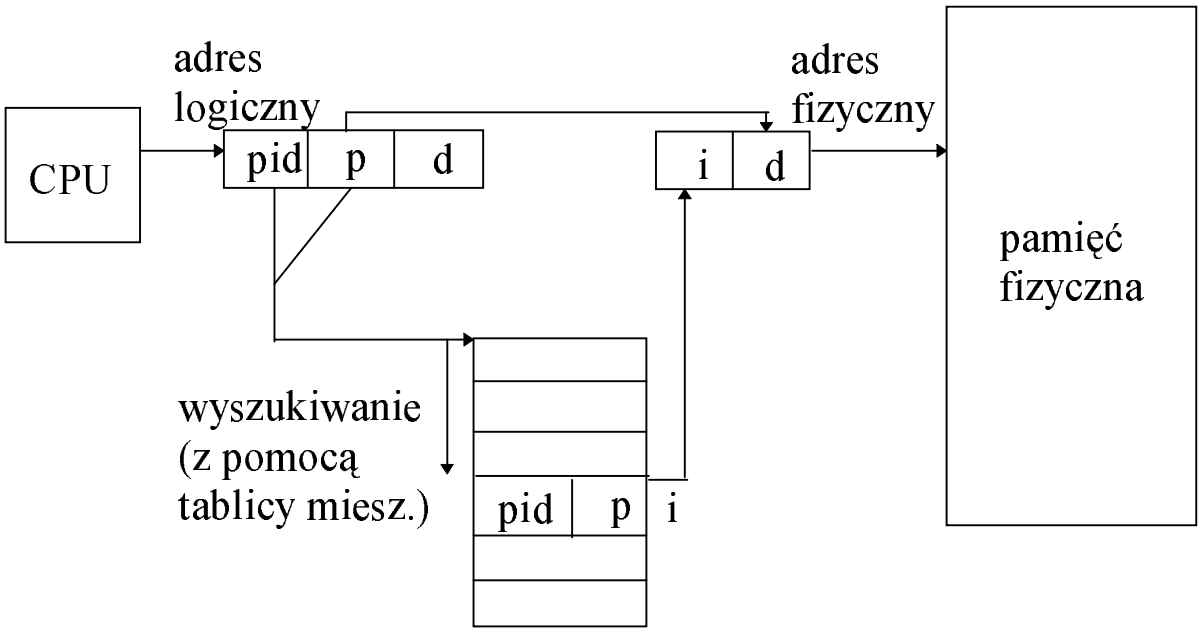
czas przeglądania TLB = ε

czas dostępu do pamięci = 1 mikrosekunda

efektywny czas dostępu = $(1 + \varepsilon) \cdot \alpha + (2 + \varepsilon) \cdot (1 - \alpha)$

- *Odwrotna tablica stron*

  Posiada po jednej pozycji dla każdej ramki; pozycja zawiera adres wirtualny strony umieszczonej w ramce oraz identyfikator procesu będącego właścicielem strony
  - w porównaniu z tradycyjną tablicą stron zmniejsza zużycie pamięci, ale wydłuża czas potrzebny na znalezienie adresu
  - użycie tablicy mieszającej pozwala skrócić czas wyszukiwania



odwrotna tablica stron

- Stronicowanie umożliwia *dzielenie wspólnego kodu* (taki kod musi być *wielowejściowy*, ang. *reentrant*, czyli nie może modyfikować sam siebie); edytory, kompilatory itp.
- Do ochrony pamięci służą *bity ochrony* przypisane każdej ramce i zwykle umieszczone w tablicy stron

Segmentacja

Schemat zarządzania pamięcią zgodny ze sposobem widzenia pamięci przez użytkownika

- ***Program*** jest zbiorem segmentów. ***Segment*** jest jednostką logiczną, taką jak: program główny, procedura, funkcja, stos, tablica symboli, tablice, zmienne lokalne i globalne
- ***Adres logiczny***: <numer_segmentu, przesunięcie>
- ***Tablica segmentów***: każda pozycja zawiera fizyczny adres początku segmentu i rozmiar segmentu
- ***Rejestr bazowy tablicy segmentów*** zawiera adres tablicy
- ***Ochrona***: z każdą pozycją w tablicy segmentów są związane bity ochrony

- ***Dzielenie***: na poziomie segmentów jest bardziej naturalne niż na poziomie stron, ten sam adresów w różnych pozycjach tablic segmentów różnych procesów
- ***Przydział pamięci***: pierwszy/najlepszy pasujący, fragmentacja zewnętrzna

Segmentacja ze stronicowaniem

Multics: każdy segment ma swoją tablicę stron, znika problem fragmentacji zewnętrznej; deskryptor segmentu zawiera adres tablicy stron segmentu (w pamięci), długość segmentu i pomocnicze bity; tablica segmentów też jest stronicowana
Intel 386: segmentacja z dwoma poziomami stronicowania