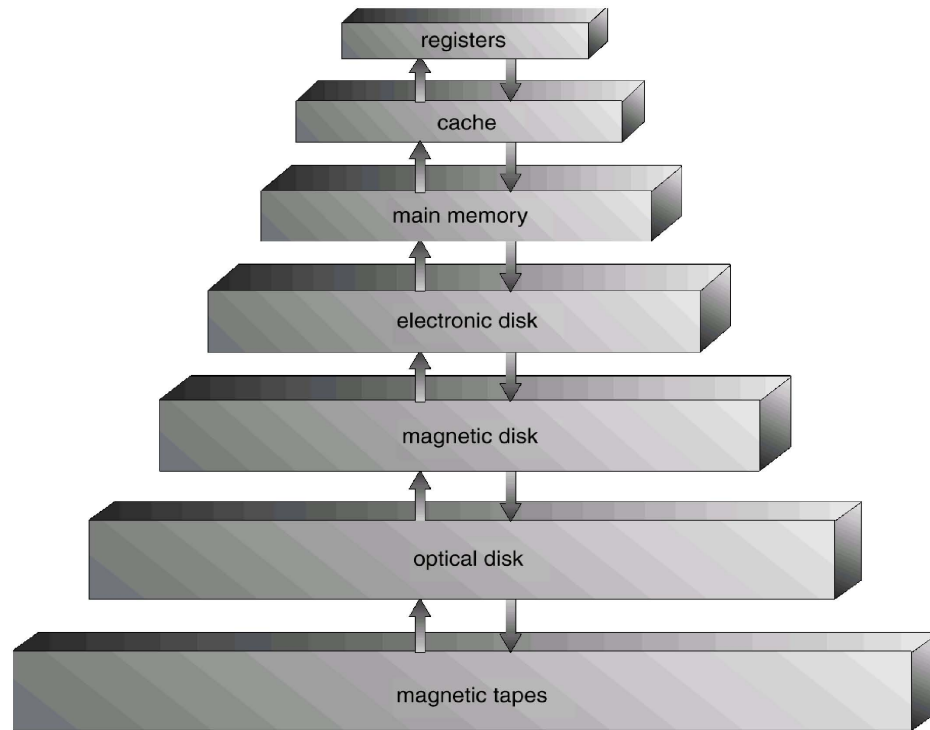


Zarządzanie pamięcią

Sieciowe Systemy Operacyjne

Świat idealny a świat rzeczywisty

- W idealnym świecie pamięć powinna ...
 - Mieć bardzo dużą pojemność.
 - Mieć bardzo krótki czas dostępu
 - Być nieulotna (zawartość nie jest tracona po wyłączeniu zasilania).
- W świecie rzeczywistym pamięć jest ...
 - Szybka
 - O dużej pojemności
 - Tania
 - *Wybierz dowolne dwa z trzech powyższych warunków*
- Cel zarządzania pamięcią:
 - Aby rzeczywistość jak najbardziej przybliżała ideał.



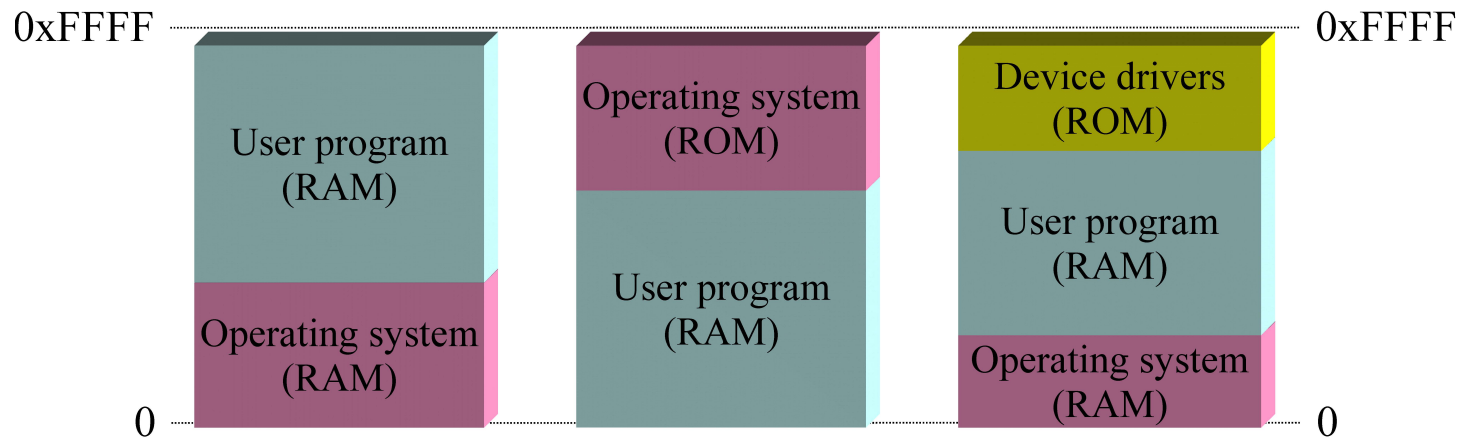
Przemieszczając się w dół hierarchii

- Zwiększamy czas dostępu
- Zmniejszamy koszt jednego bajtu

- Rejestry procesora: dostęp najszybszy, pojemność najmniejsza.
- Pamięć podręczna (ang. cache memory). Na ogół kilka poziomów (L1, L2, L3). Przechowuje najczęściej używane dane z pamięci operacyjnej. Często niewidoczna dla programisty aplikacji.
 - Przykład: AMD Athlon XP.
 - 128 KB L1 cache (64 KB dane + 64 KB program)
 - 512 KB L2 cache (dane + program).
 - L1 znacznie szybsza od L2
- Pamięć operacyjna (DRAM). Średnia szybkość i średni koszt.
- Pamięć dyskowa: Tania, Nieulotna, Duża pojemność, Bardzo duży czas dostępu.

Proste zarządzanie pamięcią w systemie jednoprogramowym

- System operacyjny + sterowniki urządzeń
- Jeden program użytkownika => Ochrona pamięci nie jest potrzebna.



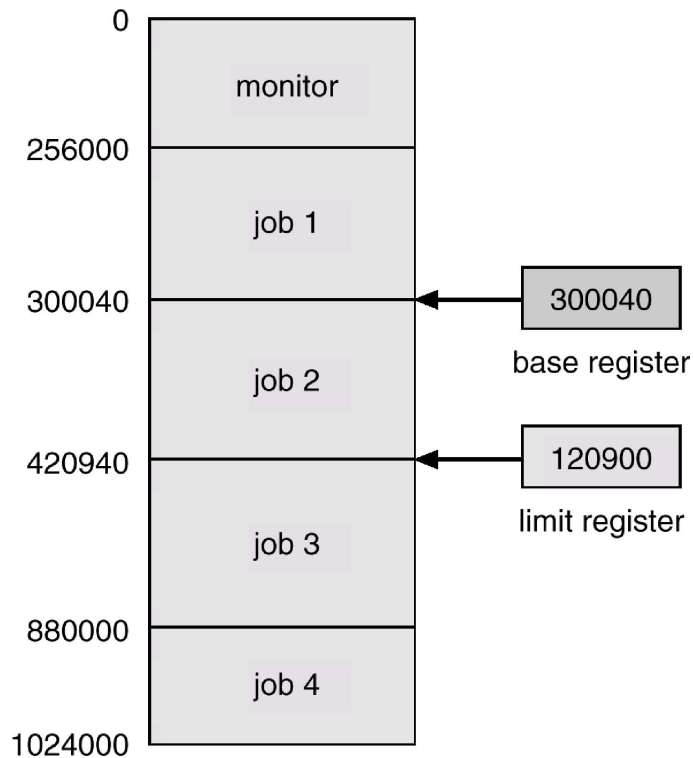
Systemy wieloprogramowe

Partycje o nie zmieniającym się rozmiarze

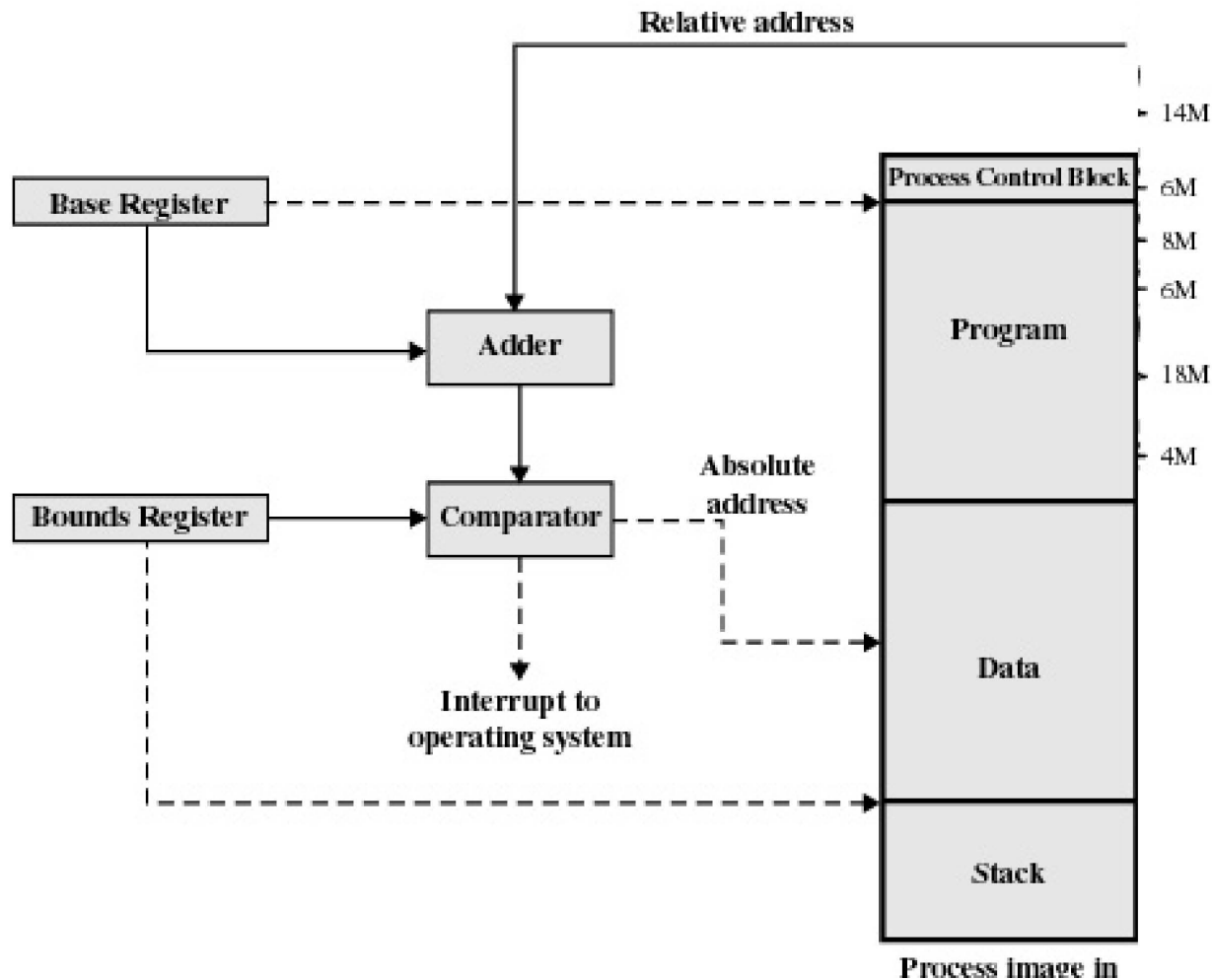
- Pamięć operacyjna podzielona na obszary o stałym rozmiarze.
 - Rozmiary poszczególnych obszarów mogą się różnić
 - Do którego obszaru załadować program ?
- Program ładowany jest do wolnego obszaru.
 - Wspólna kolejka programów
 - Odrębne kolejki dla każdego obszaru
 - Proces trafia do kolejki związanej z najmniejszym obszarem zdolnym zmieścić proces.



- W chwili kompilacji. Problem: Nie znamy adresu pod który program zostanie załadowany.
- W chwili ładowania programu. Generuj kod relokowalny, który może zostać załadowany pod dowolny adres w pamięci. Często wykorzystuje się *tablicę relokacji*. Kod raz załadowany do pamięci nie może być przemieszczany.
 - W rozwiązaniu z tablicą relokacji używa się adresów bezwzględnych, obliczonych np. względem początku kodu programu. Po załadowaniu do wszystkich adresów bezwzględnych (informacje o adresach bezwzględnych przechowuje tablica relokacji) w kodzie dodaje się adres pod który został załadowany kod programu.
- W chwili wykonywania programu: Kod załadowany do pamięci może być przemieszczany.
 - Potrzebne jest wsparcie sprzętowe
 - Np. rejestr bazowy (i opcjonalnie rejestr limitu)



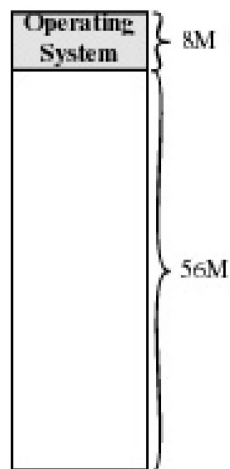
- Logiczna i fizyczna przestrzeń adresowa
- Adres logiczny (zwany czasami adresem wirtualnym): z punktu widzenia procesu
- Adres fizyczny: z punktu widzenia pamięci
- Translacja adresu:
$$AdrFizyczny = RejestrBazowy + AdrLogiczny$$
- Rejestr Limitu służy do realizacji ochrony pamięci. musi być mniejsze niż L
 - $AdresFizyczny < L$



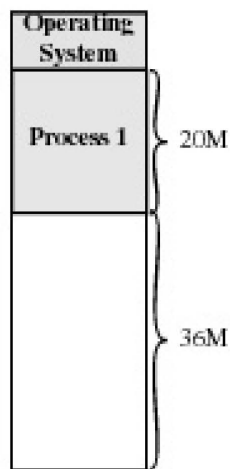
- Fragmentacja wewnętrzna.
 - Obszar pamięci przydzielony dla procesu jest większy niż niezbędny.
 - Poważny problem przy partycjach o stałych rozmiarach.
- Fragmentacja zewnętrzna.
 - Całkowita pojemność wolnej pamięci jest wystarczająca na zaspokojenie żądania procesu, ale wolna pamięć nie stanowi ciągłego bloku

- Rozmiar partycji jest dokładnie równy rozmiarowi procesu
 - Brak fragmentacji wewnętrznej.
- Po pewnym czasie wystąpi fragmentacja zewnętrzna (“dziury w pamięci”)
- Kompakcja: procesy są przesuwane w pamięci tak, aby wolna pamięć tworzyła jeden duży blok.
 - Likwiduje fragmentację
 - Czasochłonna
 - Problemy z wejściem wyjściem.
 - Wymaga określania adresów w chwili wykonywania programu, np. Przy pomocy rejestru bazowego

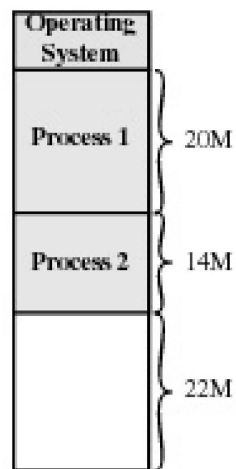
Przykład partycjonowania dynamicznego



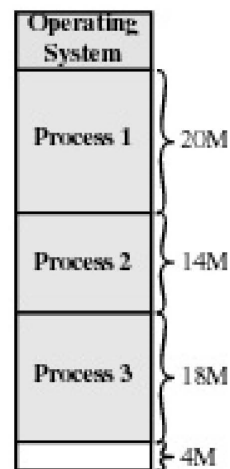
(a)



(b)

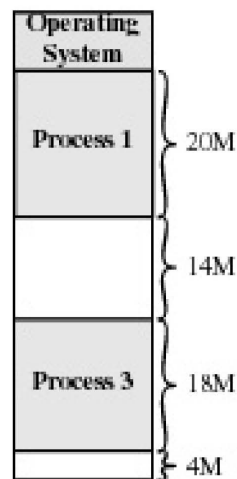


(c)

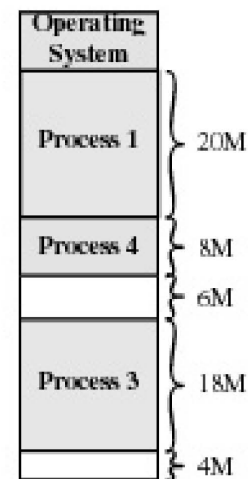


(d)

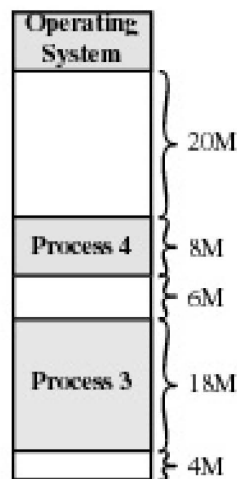
- *Coraz większa fragmentacja zewnętrzna*



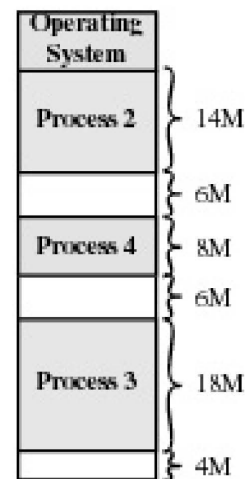
(e)



(f)



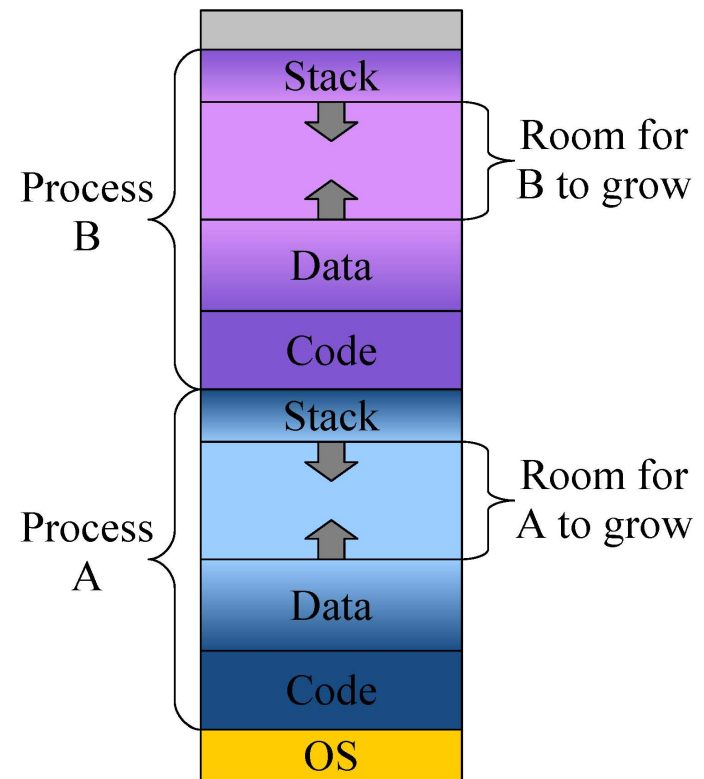
(g)



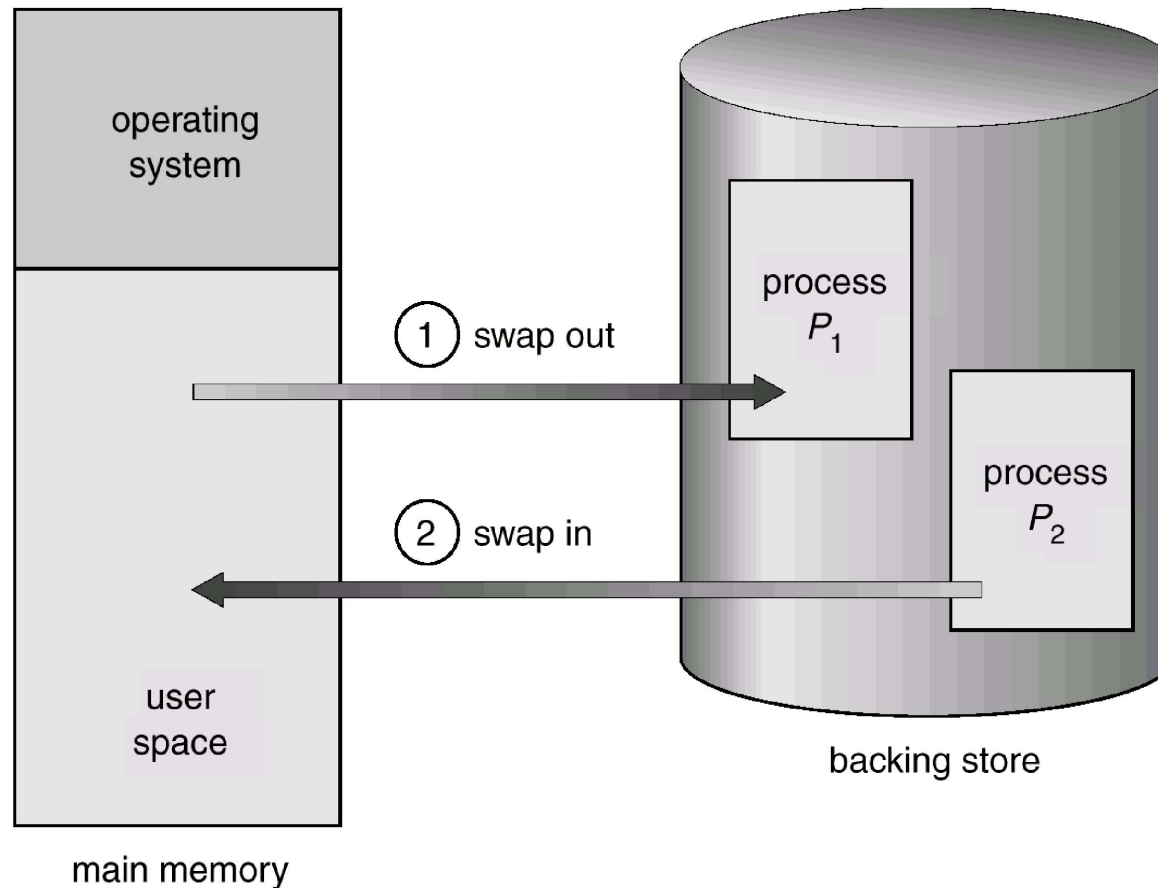
(h)

- First-fit – w pierwszym bloku o wystarczającym rozmiarze
- Best-fit – w największym bloku o wystarczającym rozmiarze
- Next-fit – następny obszar pasujący po ostatnio wybranym
- Worst-fit- w największym bloku o wystarczającym rozmiarze

- Proces może zwiększać swoje wymagania na pamięć.
- Dotyczy to stosu i danych.
- Przydziel większy blok pamięci niż wymagany na początku.
 - Mała wydajność
 - Wrócimy do tego przy opisie stronicowania..



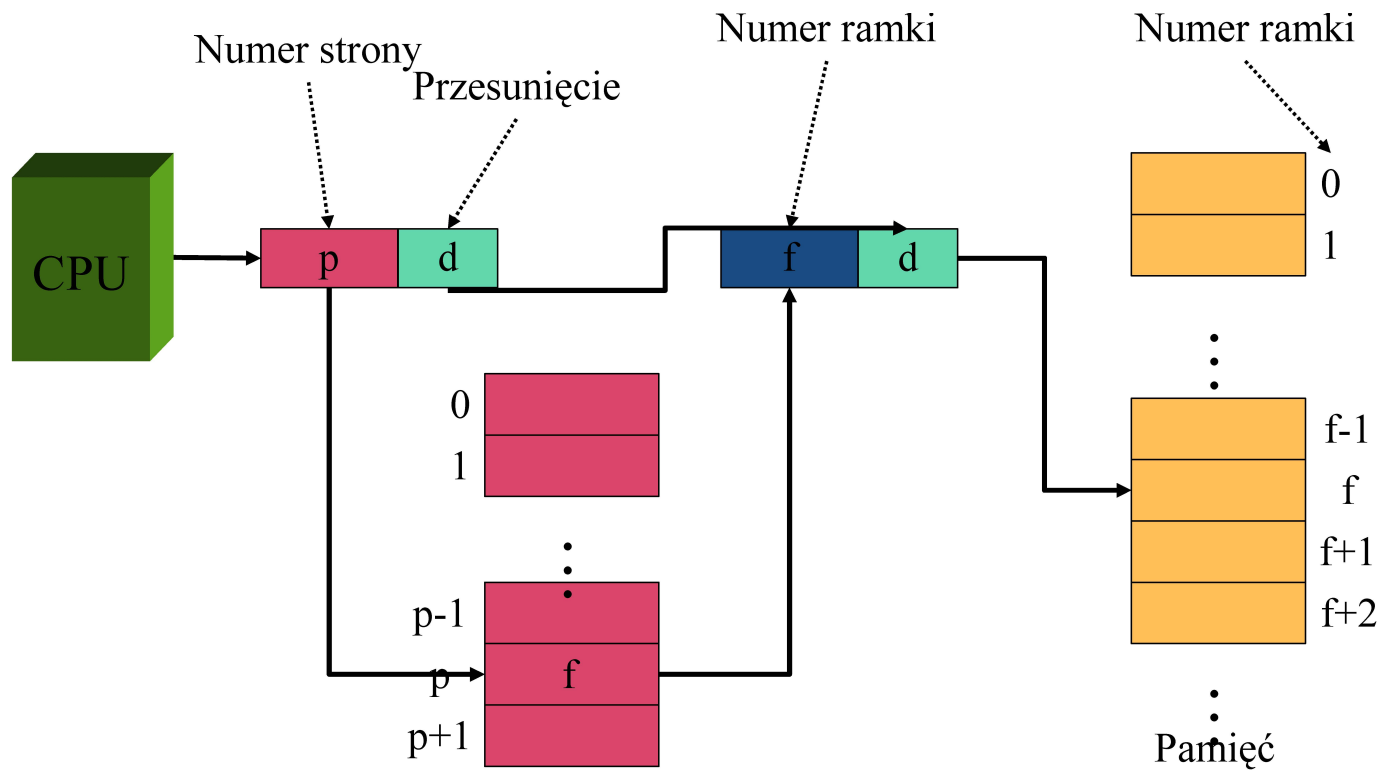
Wymiana (ang. swapping)

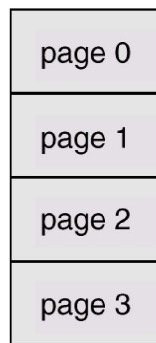


- Proces może być tymczasowo przeniesiony z pamięci operacyjnej do pamięci pomocniczej (ang. backing store).
- Zwolniona pamięć może zostać wykorzystana przez inne procesy
- Wymianie podlegają **kompletne procesy**

- Pamięć fizyczna jest podzielona na *ramki* (ang. frame) o stałym rozmiarze, będącym potęgą dwójki (np. 4KB).
- Przestrzeń adresowa procesu jest podzielona na *strony* (ang. page), rozmiar strony jest równy rozmiarowi ramki.
- I – tej stronie w przestrzeni adresowej procesu odpowiada K -ta ramka. Dla dowolnego I , K może być dowolne.
- Adres generowany przez procesor jest podzielony na dwie części:
 - Numer strony (p) oraz przesunięcie na stronie (d)
 - Przykład strona ma rozmiar $4KB=2^{12}$ bajtów, adres ma 32 bity





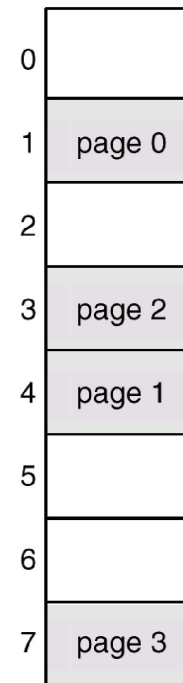


logical
memory

0	1
1	4
2	3
3	7

page table

frame
number



physical
memory

- Każdy proces ma własną tablicę stron.
- Tablica stron jest przechowywana w pamięci.
- Adres fizyczny początku tablicy stron jest przechowywany w *rejestrze bazowym tablicy stron*.
- Liczba elementów w tablicy stron jest przechowywana w *rejestrze długości tablicy stron*.
- Zawartość obydwu rejestrów jest zmieniana przy przełączaniu kontekstu.
- Przy naiwnej sprzętowej implementacji tego schematu każde, odwołanie do pamięci przez proces wymaga wykonania dwóch cykli dostępu do pamięci.
 - Jeden cykl aby znaleźć numer ramki w tablicy stron.
 - Jeden cykl na wykonanie właściwej operacji.
 - Szybkość dostępu do pamięci spada dwukrotnie

Bufor TLB

(ang. Translation lookaside bufer)

- Przechowywany w procesorze.
- Zawiera numer strony oraz odpowiadający mu numer ramki dla k (np $k=32$) najczęściej używanych stron.
- Jeżeli numer strony występuje w buforze TLB, to nie jest wymagany dostęp do tablicy stron.
- Jeżeli nie występuje, wykonywany jest dostęp do tablicy stron a para $\langle \text{Strona}, \text{Ramka} \rangle$ jest wprowadzana do bufora TLB
 - Na którą pozycję ?
- W praktyce zapewniany jest ponad 95% współczynnik trafień.
- Uwaga: przełączenie kontekstu wymaga opróżnienia bufora TLB !!!

Numer strony	Numer ramki
8	3
unused	
2	1
3	0
12	12
29	6
22	11
7	4

Przykładowy bufor
TLB

Main memory	
0	A.0
1	A.1
2	A.2
3	A.3
4	B.0
5	B.1
6	B.2
7	C.0
8	C.1
9	C.2
10	C.3
11	
12	
13	
14	

(d) Load Process C

Main memory	
0	A.0
1	A.1
2	A.2
3	A.3
4	
5	
6	
7	C.0
8	C.1
9	C.2
10	C.3
11	
12	
13	
14	

(e) Swap out B

Main memory	
0	A.0
1	A.1
2	A.2
3	A.3
4	D.0
5	D.1
6	D.2
7	C.0
8	C.1
9	C.2
10	C.3
11	D.3
12	D.4
13	
14	

(f) Load Process D

- (d) W pamięci procesy A,B,C.
- (e) Proces B przesłany do obszaru wymiany
- (f) załadowany proces D

Struktury systemu w chwili (f)

0	0
1	1
2	2
3	3

Process A
page table

0	—
1	—
2	—

Process B
page table

0	7
1	8
2	9
3	10

Process C
page table

0	4
1	5
2	6
3	11
4	12

Process D
page table

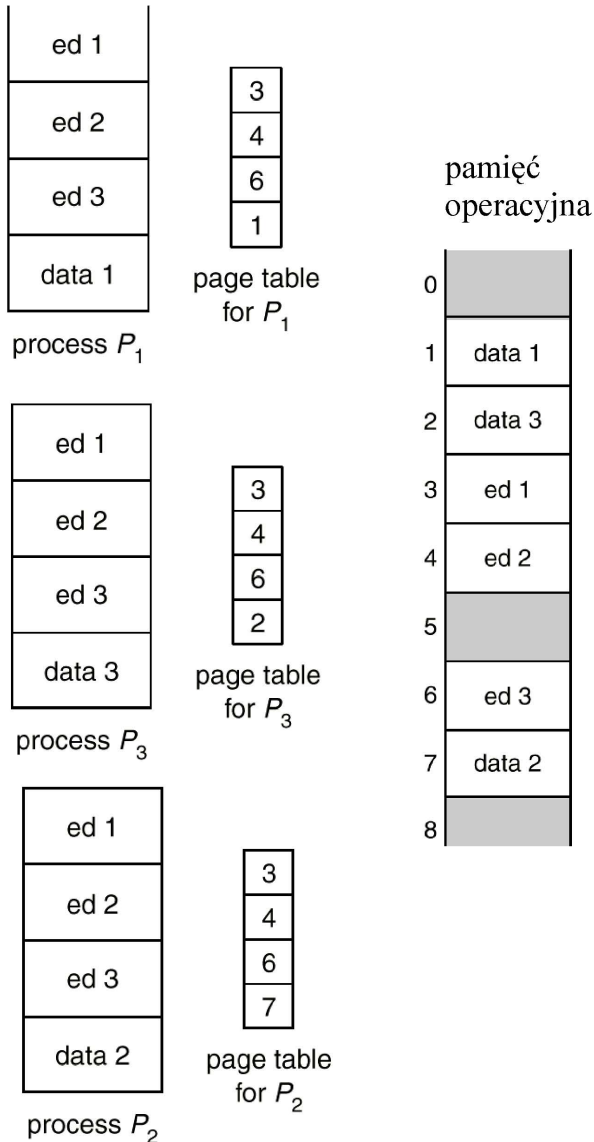
13
14

Free frame
list

Realizacja ochrony przy stronicowaniu

- Sprzętowy mechanizm ochrony, którego naruszenie (np. przez proces wykonywujący się w trybie użytkownika) powoduje zgłoszenie wyjątku (przerwania) stronicowania (ang. *page fault*)
- Możliwość określenia długości tablicy stron (rozmiaru przestrzeni adresów logicznych procesu).
- Bit ważności związany z każdą stroną i przechowywany w tablicy stron
 - Strona ważna => obecna w pamięci
 - Strona nieważna => próba dostępu powoduje wyjątek stronicowania.
 - Możliwość realizacji “dziur” w logicznej przestrzeni adresowej
 - Strony, którym nie odpowiada zaalokowana pamięć mają wyzerowany bit ważności.
 - *Patrz rysunek w jednym z poprzednich slajdów*
- Bit ochrony przed zapisem – próba zmodyfikowania strony z ustawionym bitem powoduje wyjątek stronicowania.
- Bit trybu jądra – próba dostępu do strony z ustawionym bitem trybu jądra z procesu wykonywującego się w trybie użytkownika powoduje wyjątek stronicowania => w ten sposób jądro chroni swoją pamięć przed programami użytkowników.

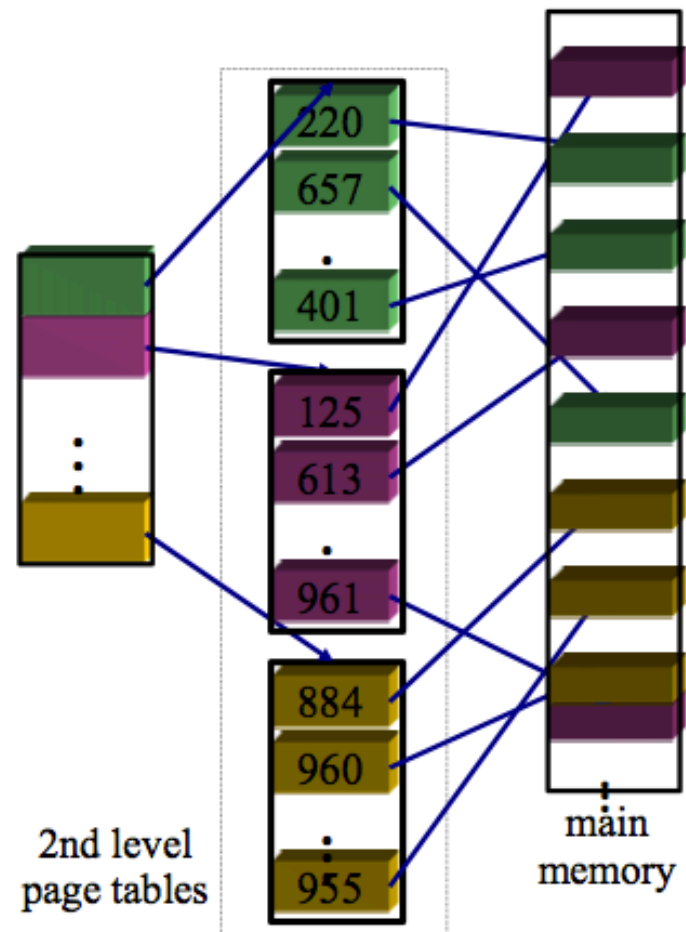
Pamięć współdzielona

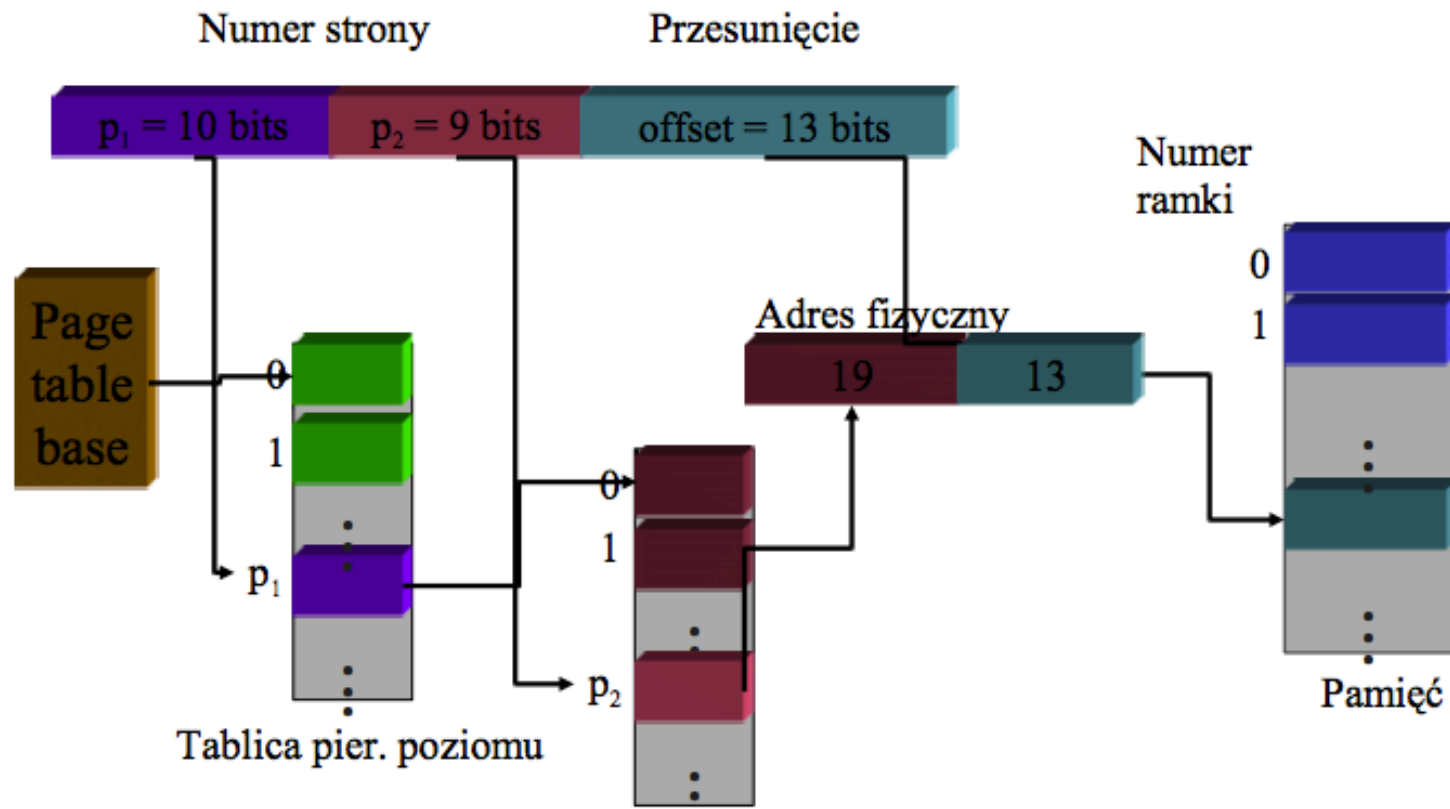


- Niech pamięć edytora tekstu składa się z 3 stron kodu i jednej strony z danymi. Trzech użytkowników uruchomiło edytor.
- Czy musimy trzykrotnie wczytać kod do pamięci (razem dziewięć stron) ?
- Jeżeli kod jest kodem niemodyfikującym się, to nie.
 - trzy ramki z kodem zostały odwzorowane w przestrzeń adresową trzech procesów, każda z nich jest współdzielona przez trzy procesy.
 - Wymagana jest implementacja bitu ochrony przed zapisem – aby nie pozwolić żadnemu z trzech procesów na modyfikację stron z kodem.
- Ponadto pamięć współdzielona może być wykorzystana do bardzo szybkiej wymiany danych pomiędzy procesami – większość systemów posiada niezbędne API.

Stronicowanie dwupoziomowe

- Tablica stron może mieć duży rozmiar.
 - Niech adres logiczny ma 32 bity, Przestrzeń adresowa na rozmiar 4GB.
 - Jeżeli pozycja w tablicy stron ma 4 bajty, to na tablicę stron potrzebujemy 4MB.
- Stronicowanie wielopoziomowe
- Idea: “Tablica stron podlega stronicowaniu”
- Tablica stron pierwszego poziomu przechowuje numery stron w tablicy drugiego poziomu.
- Każdej pozycji w tablicy stron pierwszego poziomu odpowiada tablica stron drugiego poziomu.
- Tablica stron drugiego poziomu przechowuje numery stron





Zalety stronicowania

- Eliminuje fragmentację zewnętrzną
- Fragmentacja wewnętrzna jest ograniczona do rozmiaru strony.
- Łatwa realizacja ochrony pamięci
- Łatwa realizacja dynamicznego wzrostu obszaru przydzielonej pamięci.
 - Wraz z przydzielaniem dodatkowych ramek zaznaczaj bit ważności.
- Możliwość implementacji pamięci wirtualnej.
 - Część przestrzeni adresowej procesu w pamięci, część na dysku
 - Rozmiar przestrzeni adresowej procesu większy od pojemności pamięci RAM

Dziękuję za uwagę



dr inż. Jacek Czerniak
jczerniak@ukw.edu.pl