ZARZĄDZANIE PAMIĘCIĄ

WYKŁAD 3 SYSTEMY OPERACYJNE

AUTOR: DR INŻ JOANNA KOŁODZIEJCZYK

ZARZĄDZANIE PAMIĘCIĄ

Cele zarządzania pamięcią

- Aby zapewnić wygodną abstrakcję do programowania.
- Aby przydzielić ograniczone zasoby pamięci między konkurującymi procesami.
- Aby zmaksymalizować wydajność przy ograniczonach zasobach.

Mechanizmy

- Adresowanie fizyczne i wirtualne
- Techniki: partycjonowanie, stronicowanie, segmentacja
- Zarządzanie tabelami stron, TLB, sztuczki maszyn wirtualnych

Zasady

- Algorytmy zastępowania stron

ZAŁOŻENIA

System komputerowy wykonuje programy, a dane znajdują się głównie w pamięci pomocniczej.

W pamięci operacyjnej przechowywana jest pewna liczba procesów (wielozadaniowość). Procesy dzielą pamięć.

Wydajność systemu komputerowego zależy od sposobów zarządzania pamięcią.

Wybór metody zarządzania musi uwzględniać cechy sprzętu.

CO TO JEST PAMIEĆ

Pamięć jest zasobem służący do przechowywania danych i programów.

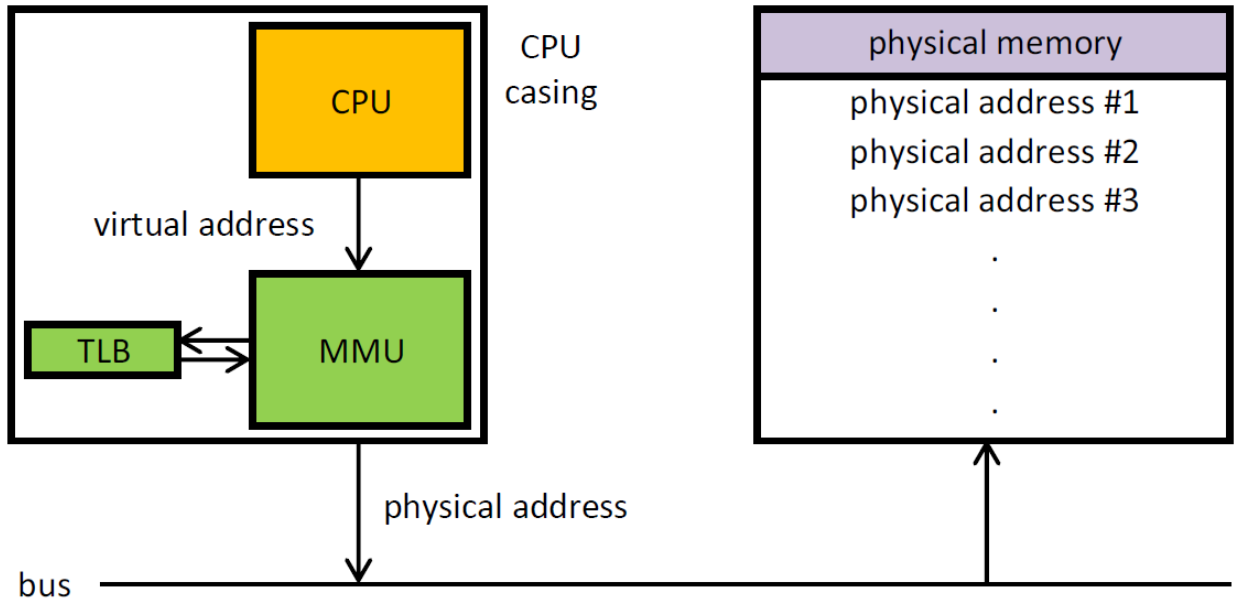
Z punktu widzenia systemu pamięć jest zasobem o strukturze hierarchicznej (począwszy od rejestrów procesora, przez pamięć podręczną, pamięć główną, skończywszy na pamięci masowej), w której na wyższym poziomie przechowywane są dane, stanowiące fragment zawartości poziomu niższego.

Z punktu widzenia procesu (również procesora) pamięć jest zbiorem bajtów identyfikowanych przez adresy, czyli tablicą bajtów, w której adresy są indeksami.

PRZESTRZEŃ ADRESOWA

- Przestrzeń adresowa jest to zbiór dopuszczalnych adresów w pamięci.
  - Przestrzeń fizyczna – zbiór adresów przekazywanych do układów pamięci
  - Przestrzeń logiczna – zbiór adresów generowanych w kontekście wykonywanego procesu
- Rozróżnienie przestrzeni fizycznej i logicznej oznacza, że w procesie komórka pamięci jest inaczej identyfikowana, niż to wynika z jej fizycznego adresu, co wymaga odpowiedniego przekształcenia adres logicznego na fizyczny, zwanego transformacją adresu. Za transformację odpowiada układ ściśle współpracujący z procesorem — jednostka zarzadzania pamięcią (ang. memory management unit — MMU).

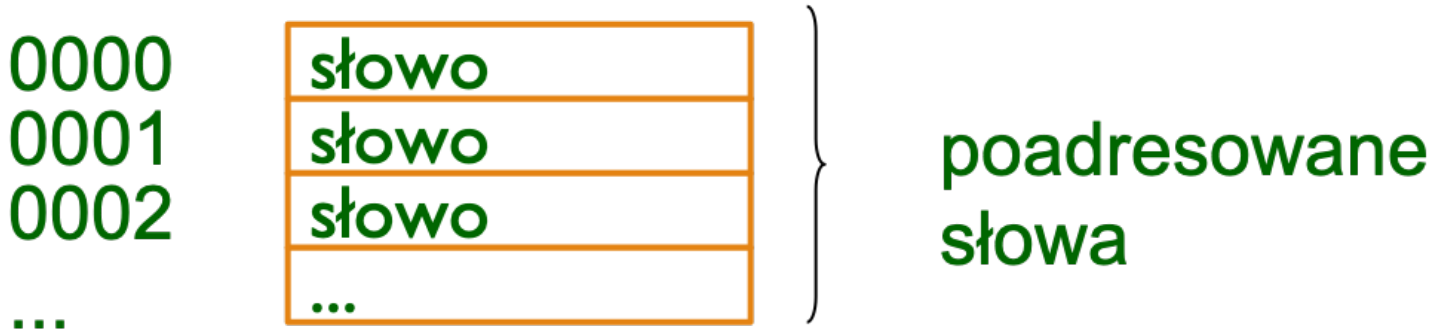
MMU



CPU: Central Processing Unit
MMU: Memory Management Unit
TLB: Translation lookaside buffer

PAMIĘĆ JAKO TABLICA

- Procesor pobiera i przechowuje dane w pamięci.
- Współpraca z pamięcią to:
  - operacje czytania spod konkretnego adresu,
  - operacje pisania pod konkretnym adresem.



CYKL WYKONANIA ROZKAZU PROCESORA

1. Pobranie rozkazu z **pamięci**.
2. Dekodowanie rozkazu.
3. Pobranie z **pamięci** argumentów.
4. Wykonanie rozkazu na argumentach.
5. Zapisanie wyniku w **pamięci**.

WIĄZANIE ADRESÓW

- Proces może rezydować w dowolnym miejscu pamięci fizycznej.
- Program przed wykonaniem przechodzi przez kilka faz, podczas których jego adres może ulegać zmianie.
- Wiązanie adresów to odwzorowanie jednej przestrzeni adresowej (np.. obowiązującej w programie) na inną przestrzeń adresową (np.. adresy fizyczne).
- Wiązanie rozkazów i danych z adresami pamięci może nastąpić:
  - **podczas kompilacji** – kod bezwzględny: z góry określone miejsce w pamięci np.. pliki *.com w MS-DOS;
  - **podczas ładowania** – kompilator tworzy kod przemieszczalny (przy zmianie adresu startowego nie trzeba przeadresowywać pozostałych adresów);
  - **podczas wykonania** – proces ulega przemieszczeniom w pamięci podczas wykonania.

ADRESOWANIE – ZAŁOŻENIA

- Aby ułatwić zarządzanie pamięcią procesów uruchomionych w systemie, zmusimy je do używania adresów wirtualnych (adresów logicznych).
  - Adresy wirtualne są niezależne od rzeczywistej fizycznej lokalizacji przywoływanych danych.
  - System operacyjny określa lokalizację danych w pamięci fizycznej.
  - Instrukcje wykonywane przez procesor posługują się adresami wirtualnymi.
  - Adresy wirtualne są tłumaczone sprzętowo na adresy fizyczne (z pomocą systemu operacyjnego).
  - Zestaw adresów wirtualnych, które mogą być używane przez proces, obejmuje jego wirtualną przestrzeń adresową.

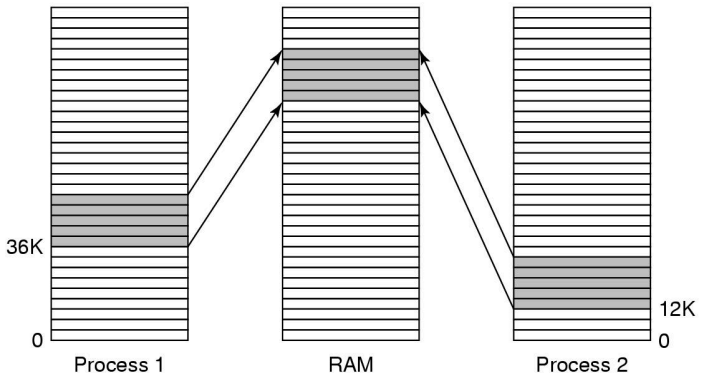
ŁADOWANIE DYNAMICZNE

Podprogramy nie są wprowadzane do pamięci dopóki nie są wywołane.

Program główny jest wczytywany. Jeżeli wywołuje podprogram, to sprawdza czy przypadkiem nie znajduje się on w pamięci. Gdy tak, to zostaje wykonany, gdy nie, zostaje wczytany do pamięci.

Zaleta: podprogramy nie wywołane nie zajmą miejsca w pamięci operacyjnej

ŁĄCZENIE DYNAMICZNE BIBLIOTEK



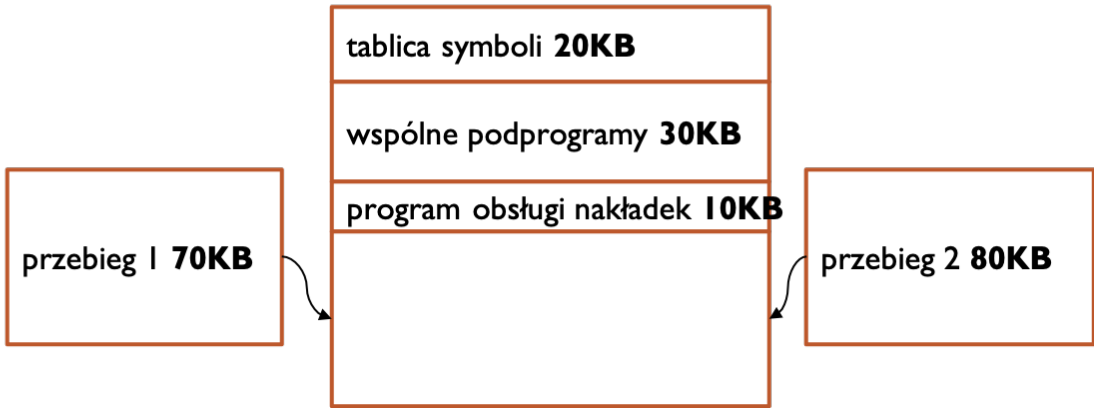
- Ładownie opóźniane jest aż do czasu wykorzystania biblioteki.
- W miejscu odwołania do programu bibliotecznego znajduje się zakładka, wskazującą jak odnaleźć w pamięci odpowiedni program biblioteczny. Wszystkie programy korzystają wówczas z jednej kopii programu bibliotecznego.
- Dzięki takiej metodzie aktualizacja biblioteki spowoduje, że wszystkie programy będą korzystać od razu z nowej biblioteki bez ponownej kompilacji.
- Przy łączeniu statycznym każdy program miał swoją kopię biblioteki.

NAKŁADKI

- Przechowywanie w pamięci tylko tych danych i rozkazów, które są w danej chwili wymagane. Przykład
- W przypadku programu mozżna zastosowć technikę, zwaną nakładkowaniem. Nakładkowanie polega na podziale kodu (nie danych) na niezależne od siebie częsci i wymianie w miarę potrzeb jednej części — nakładki (ang. overlay) — na inną.

kod przebiegu 1 70KB
kod przebiegu 2 80KB
tablica symboli 20KB
wspólne podprogramy 30KB

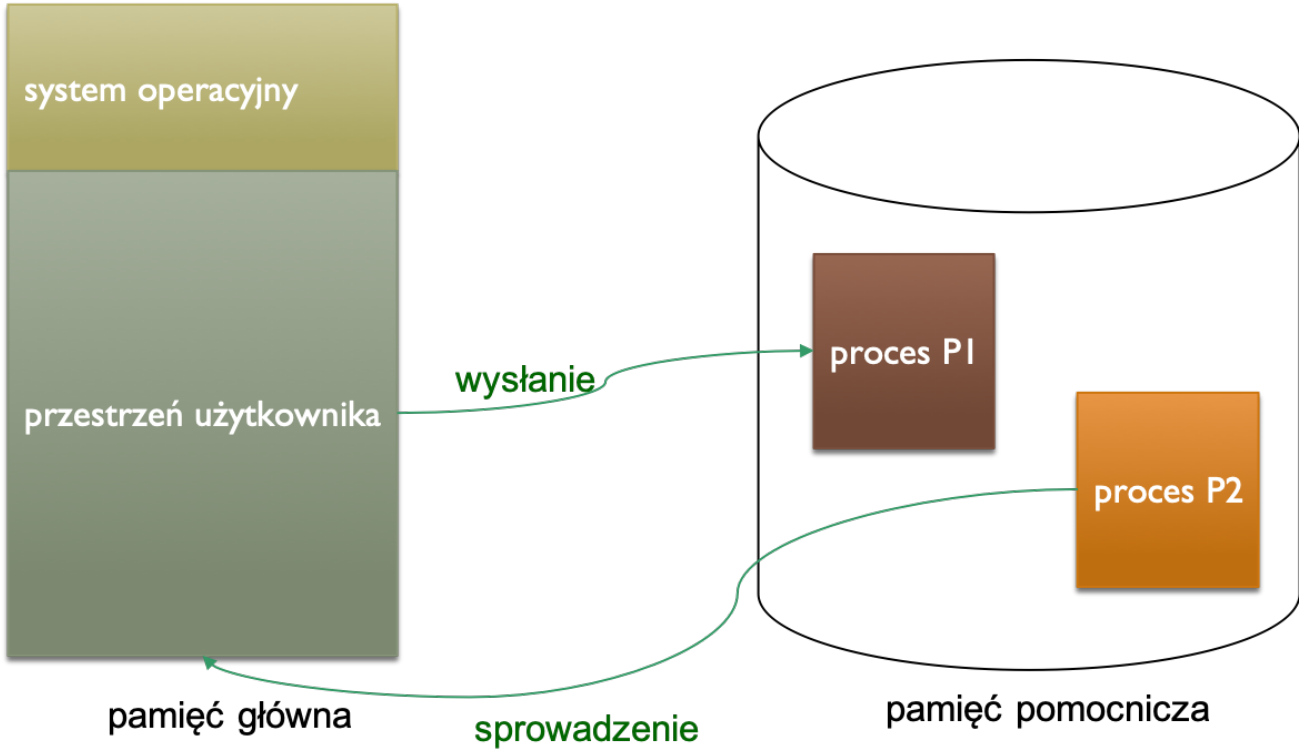
załadowanie całości 200KB
ograniczenie pamięci 150KB



WYMIANA (SWAP)

- Wymiana polega na tymczasowym odsyłaniu procesów z pamięci operacyjnej do pamięci pomocniczej (**dysk**), by nie wykonywane procesy nie zajmowały pamięci.
- Zakładamy system wieloprogramowy z przydziałem procesora opartym na algorytmie rotacyjnym z kwantem czasu. Proces, który zużył swój kwant zostaje podmieniony z innym procesem. Przy dostatecznie dużym kwancie czasu na obliczenia, nie ma potrzeby trzymania wszystkich procesów w pamięci.
- Wywłaszczenie - Przy planowaniu priorytetowym proces o niższym priorytecie może zostać usunięty w celu załadowani procesu o wyższym priorytecie.

OGÓLNY SCHEMAT WYMIANY

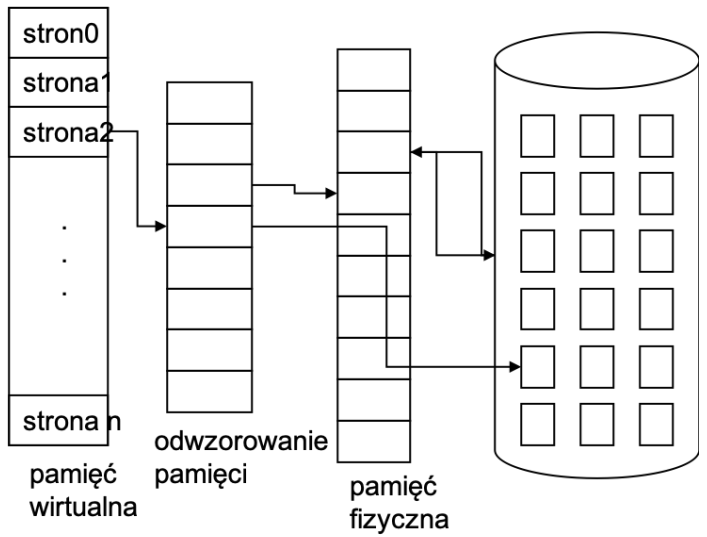


OGRANICZENIA WYMIANY

Proces dynamicznego zarządzania pamięcią realizowany jest przez funkcje systemowe. System operacyjny posiada funkcje do zamawiania i zwalniania pamięci.

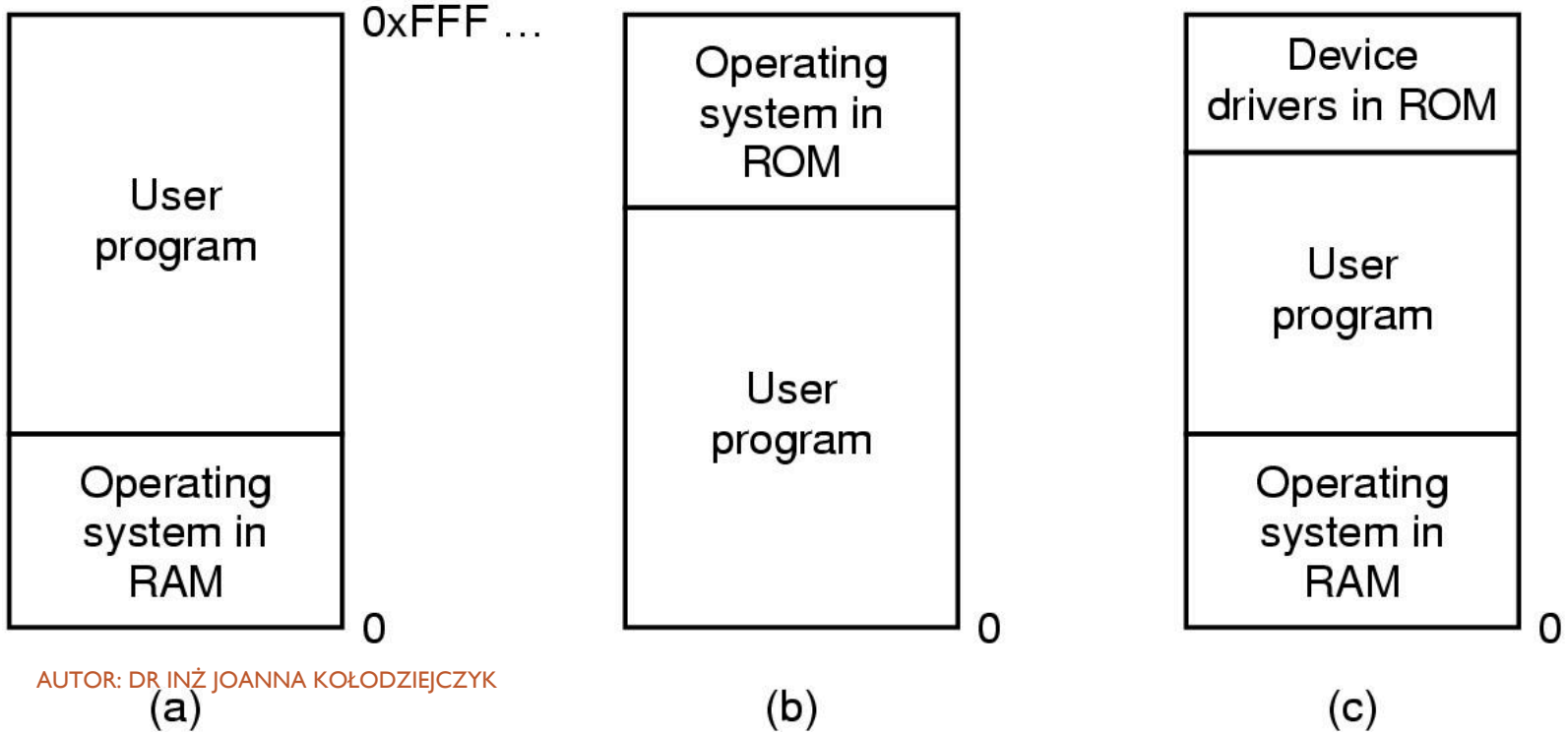
Proces usuwany z pamięci musi być bezczynny (procesy oczekujące na operacje wejścia-wyjścia nie są bezczynne).

DLACZEGO PAMIĘĆ WIRTUALNA?



- Pamięć wirtualna (VM) jest abstrakcją, dzięki której system operacyjny zapewnia zarządzanie pamięcią.
  - Umożliwia wykonanie programu z mniejszą ilością danych niż w dostępnej pamięci fizycznej.
    - Program może działać na komputerze z mniejszą ilością pamięci niż „potrzebuje”.
    - Wiele programów nie potrzebuje całego swojego kodu i danych jednocześnie - nie ma potrzeby przydzielania mu pamięci.
  - Procesy nie widzą pamięci innych procesów
  - System operacyjny dostosuje ilość pamięci przydzielonej do procesu na podstawie jego zachowania.
  - Maszyna wirtualna wymaga wparcia sprzętu i algorytmów zarządzania z poziomu systemu operacyjnego, aby ją uruchomić.
- Jak to było na początku…

BEZ PAMIĘCI WIRTUALNEJ – TYLKO SO I JEDEN PROCES UŻYTKOWNIKA



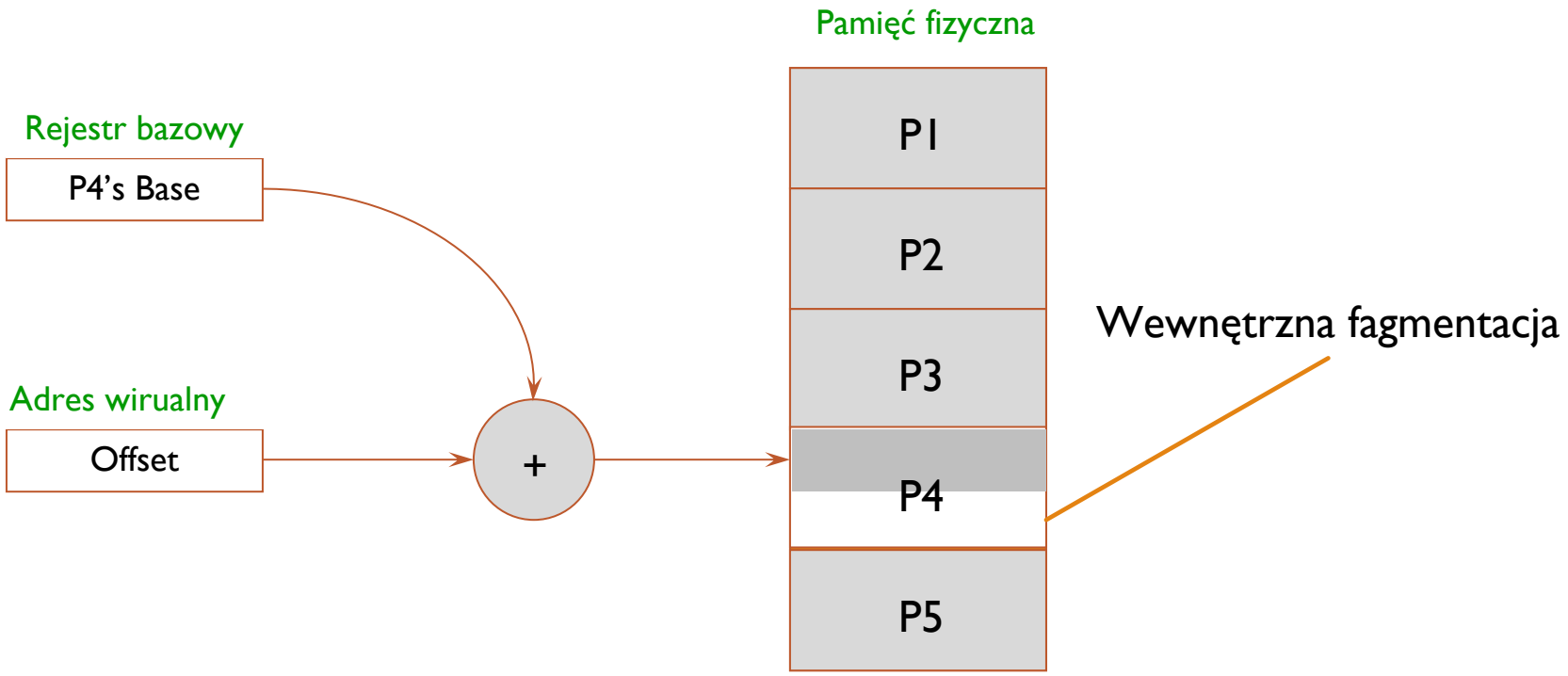
1. Podział pamięci na równe obszary o ustalonym rozmiarze.
   - Każdy obszar może zawierać jeden proces – ograniczenie liczby programów do liczby obszarów.
   - Procesy z kolejki zajmują wolne obszary.
   - Procesy po wykonaniu się zwalniają obszar pamięci.

PRZYDZIELANIE WIELU OBSZARÓW – SYSTEMY WIELOPROGRAMOWE

STAŁE MIEJSCE W PAMIĘCI CD.

- Pamięć fizyczna jest podzielona na stałe partycje
  - Wymagania sprzętowe: rejestr bazowy
  - Adres fizyczny = adres wirtualny + rejestr bazowy
  - Rejestr podstawowy ładowany przez system operacyjny po przełączeniu na proces
  - Rozmiar każdej partycji jest taki sam
- Zalety
  - Łatwy do implementacji,
  - Szybki przełącznik kontekstowy
- Problemy
  - Fragmentacja wewnętrzna: pamięć w partycji nieużywanej przez proces nie jest dostępna dla innych procesów.
  - Rozmiar partycji: jeden rozmiar nie pasuje do wszystkich (bardzo duże procesy?).
  - Nie można rozszerzyć pamięci systemu podczas wykonania programu użytkownika, bo nie można zmienić bazy.

FRAGMENTACJA WEWNĘTRZNA

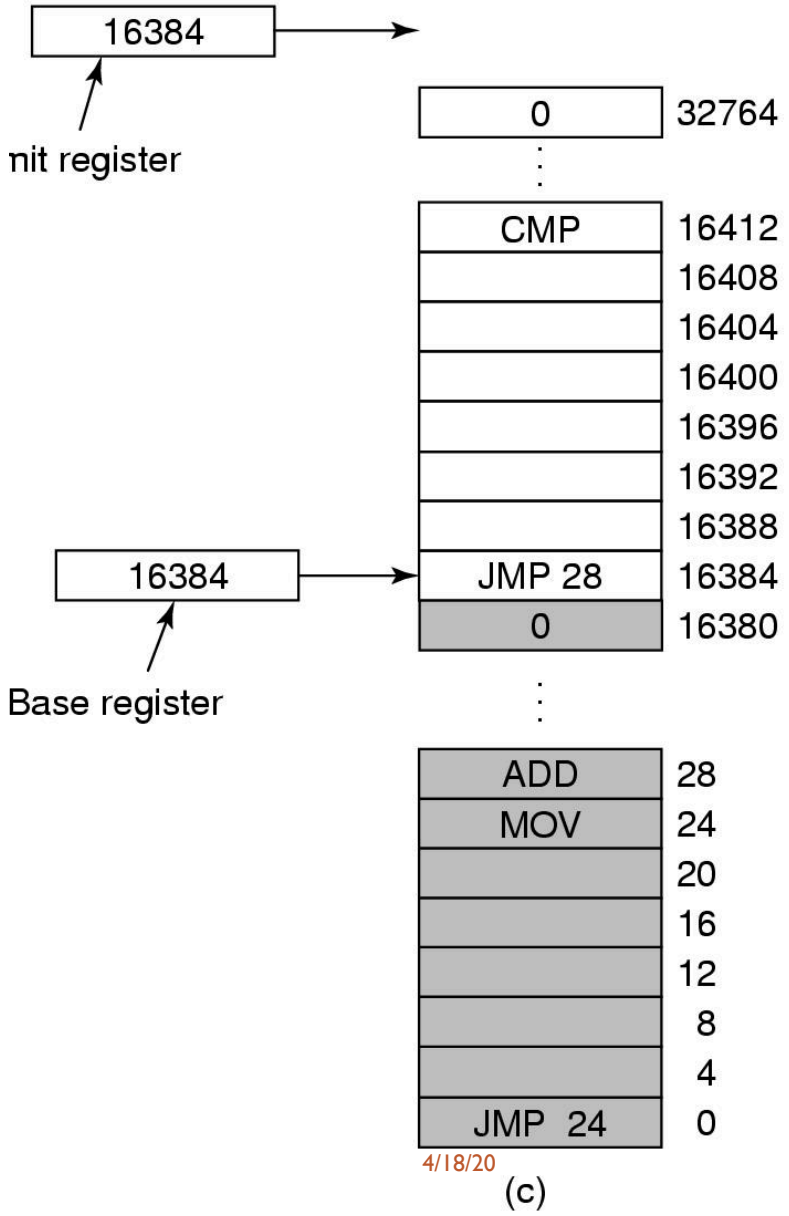


PARTYCJE W PAMIĘCI O ZMIENNEJ WIELKOŚCI

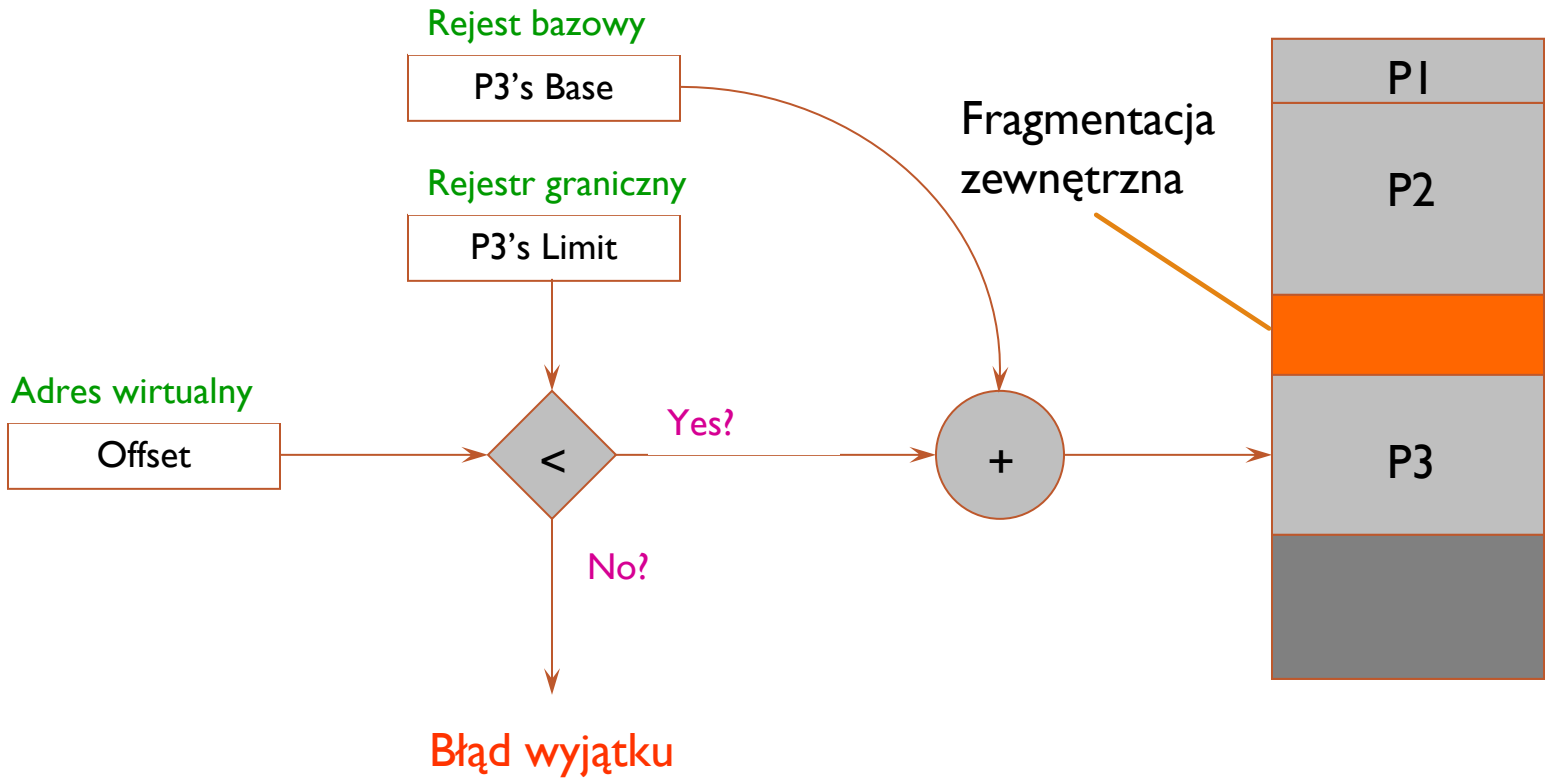
- Naturalne rozszerzenie - pamięć fizyczna jest podzielona na partycje o zmiennej wielkości
  - Wymagania sprzętowe: rejestr bazowy i rejestr graniczny
  - Adres fizyczny = adres wirtualny + rejestr bazowy
  - Dlaczego potrzebujemy rejestr graniczny? Ochrona pomiędzy procesami
  - Jeśli (adres fizyczny = adres bazowy + graniczny) to błąd wyjątku
- Zalety
  - Brak wewnętrznej fragmentacji: przydziela się dokładnie tyle pamięci ile jest potrzebne
- Problemy
  - Fragmentacja zewnętrzna: ładowanie i usuwanie zadań powoduje powstawanie pustych obszarów rozrzuconych w pamięci

OCHRONA SPRZĘTOWA

- Rejestr bazowy: przechowuje najmniejszy dopuszczalny adres fizyczny pamięci.
- Rejestr graniczny: zawiera rozmiar obszaru pamięci potrzebny dla procesu.
- np.. jeżeli rejestr bazowy to : 16384, a rejestr graniczny to 16384 to w programie mogą wystąpić odniesienia do wszystkich adresów z przedziału od (16384 do 32768).
- Każdy wygenerowany przez użytkownika adres jest kontrolowany pod katem zgodności z założonym przedziałem.



FRAGMENTACJA ZEWNĘTRZNA



ZEWNĘTRZNA FRAGMENTACJA

W trakcie ładowania i usuwania procesów z pamięci przestrzeń dzieli się na drobne kawałki.

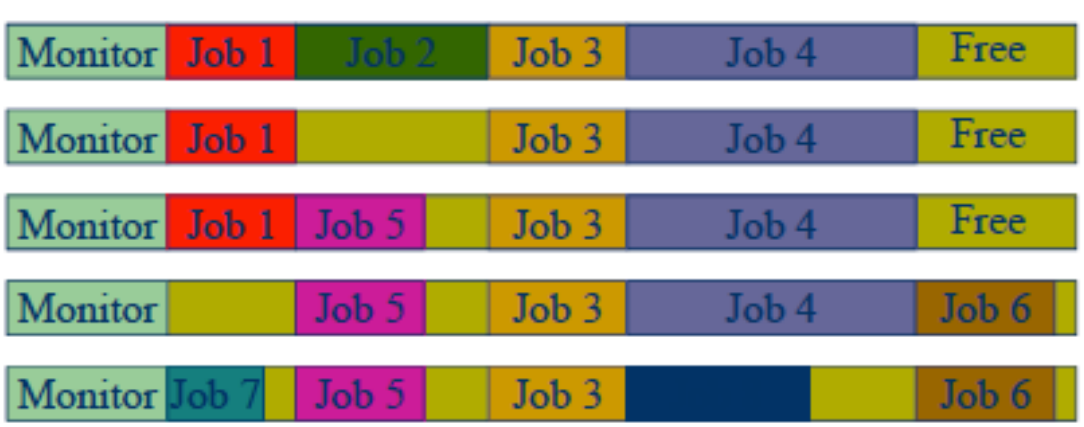
Zewnętrzna fragmentacja objawia się tym, że suma wolnych obszarów wystarcza na załadownie procesu, ale ponieważ nie stanowi spójnego obszaru, nie można jej wykorzystać.

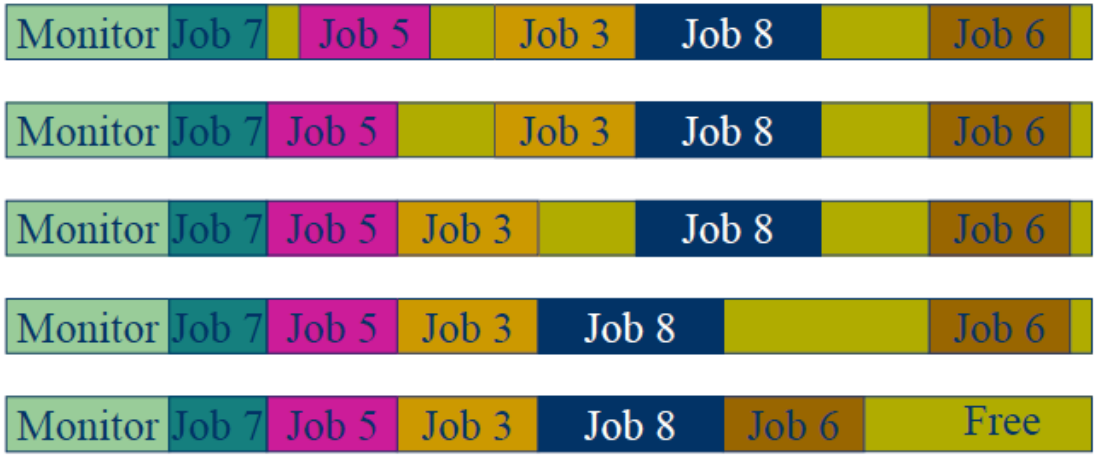
Istnieje ryzyko pozostawienia bardzo małej wolnej przestrzeni, która na pewno będzie za mała na kolejny proces i nawet zapamiętywanie jej w tablicy wolnych obszarów może przekroczyć samą jej wielkość.

UPAKOWANIE JAKO ROZWIĄZANIE PROBLEMU ZEWNĘTRZNEJ FRAGMENTACJI

Przemieszczanie wszystkich wolnych obszarów i tworzenie jednego dużego obszaru wolnej pamięci.

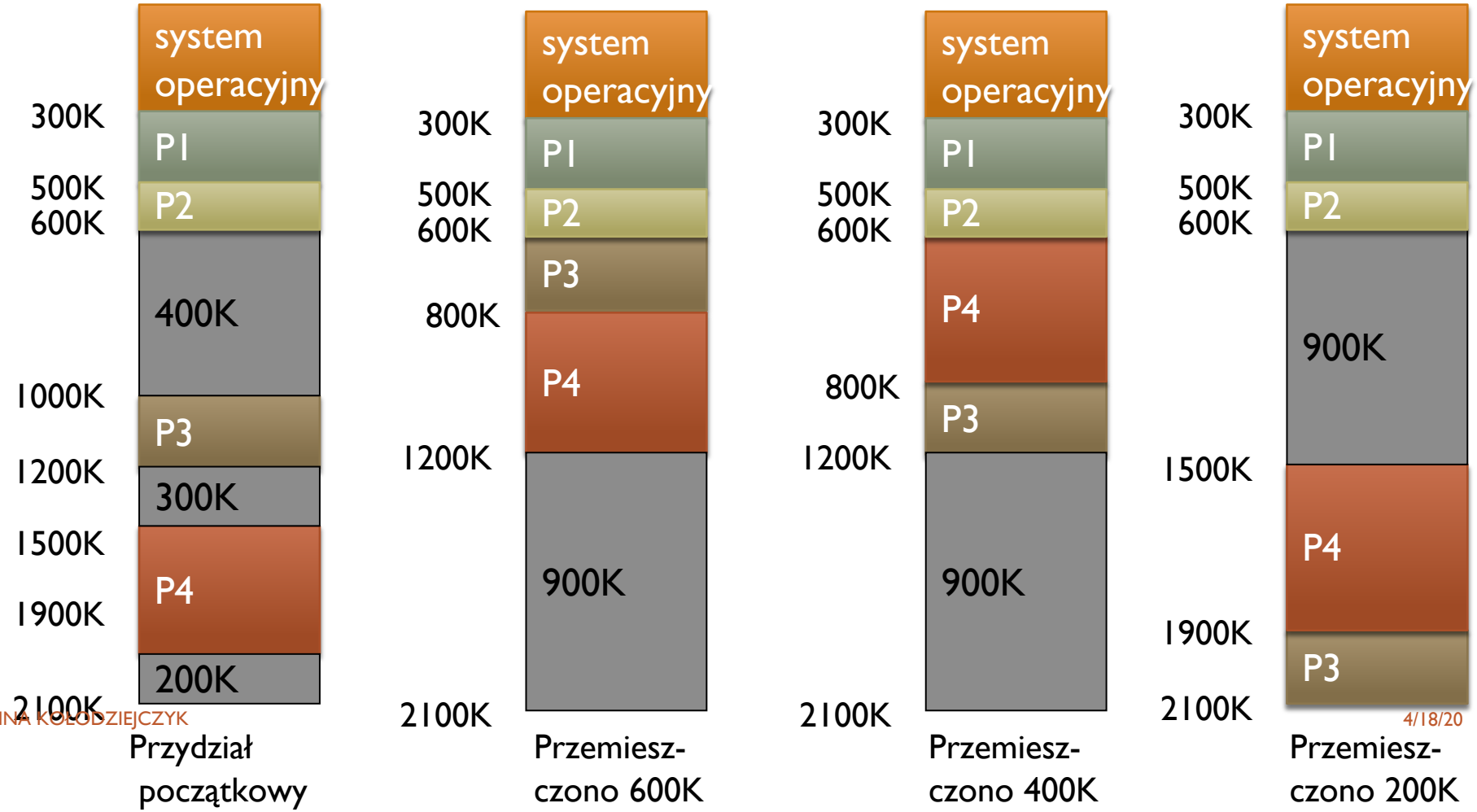
Upakowanie wymusza przesunięcie procesów w pamięci i zmianę ich wewnętrznych adresów; stąd tylko dynamiczne przemieszczanie adresów (podczas wykonania) daje możliwość upakowania.





PRZYKŁADU FRAGMENTACJI I UPAKOWANIE

RÓŻNE SPOSOBY UPAKOWANIA



01

System operacyjny przechowuje w tablicy informacje o zajętych obszarach.

02

Na początku cała pamięć jest wolna.

03

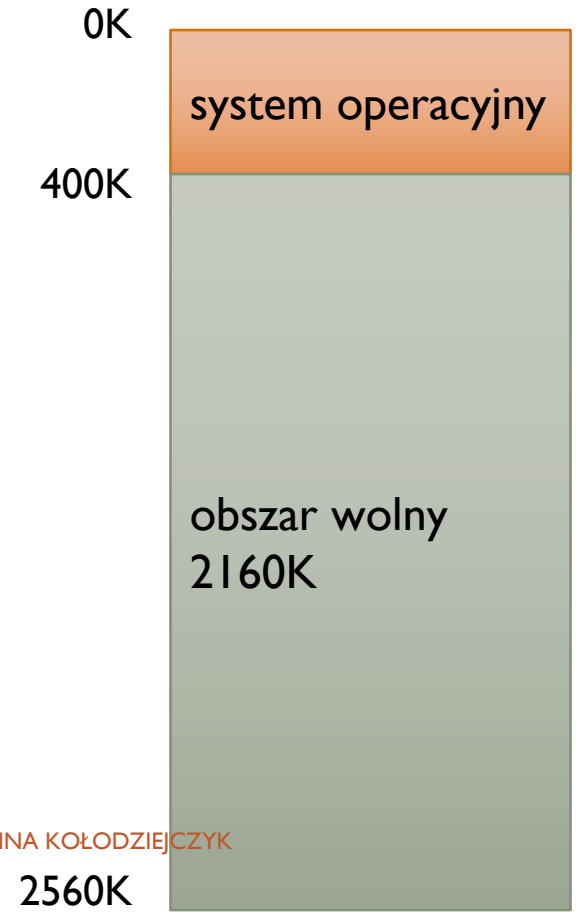
Gdy proces zgłasza zapotrzebowanie, poszukuje się wystarczająco dużego, wolnego obszaru pamięci.

04

Przydziela się dokładnie tyle pamięci ile jest wymagane przez proces.

SCHEMAT PODSTAWOWY PRZYDZIAŁU

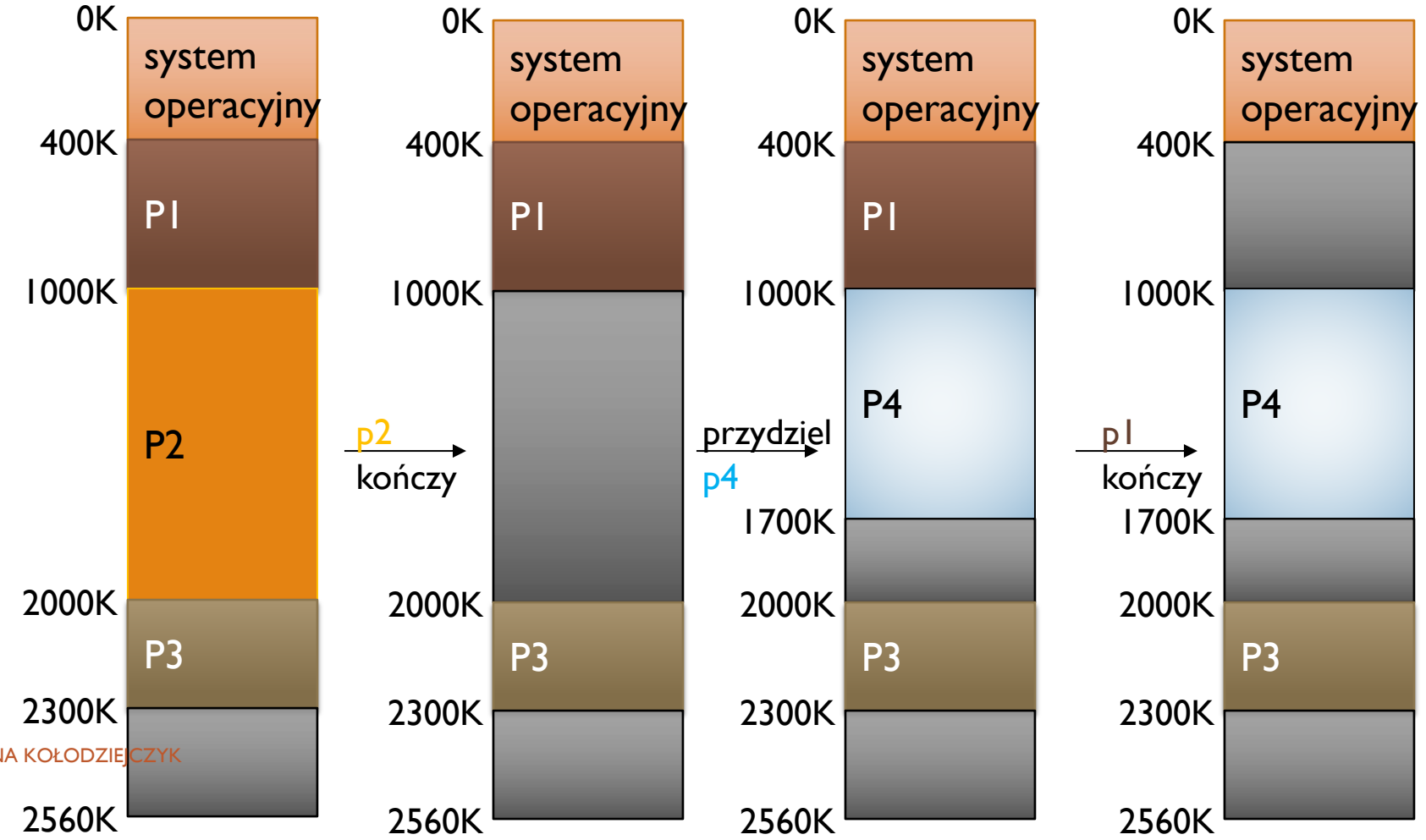
PRZYKŁAD – STAN POCZĄTKOWY



kolejka zadań:

proces	pamięć	czas
p1	600K	10
p2	1000K	5
p3	300K	20
p4	700K	8
p5	500K	15

PRZYKŁAD PRZYDZIAŁU



ALGORYTM

01

W pamięci rozproszonych jest wiele wolnych obszarów o różnych rozmiarach.

02

Gdy proces wymaga przydzielenia pamięci szuka się odpowiednio dużego dla niego obszaru.

03

Jeżeli przestrzeń jest większa niż żądana, pozostała niewykorzystana część przechodzi do listy wolnych obszarów.

04

Obszary wolne, sąsiadujące ze sobą są sumowane.

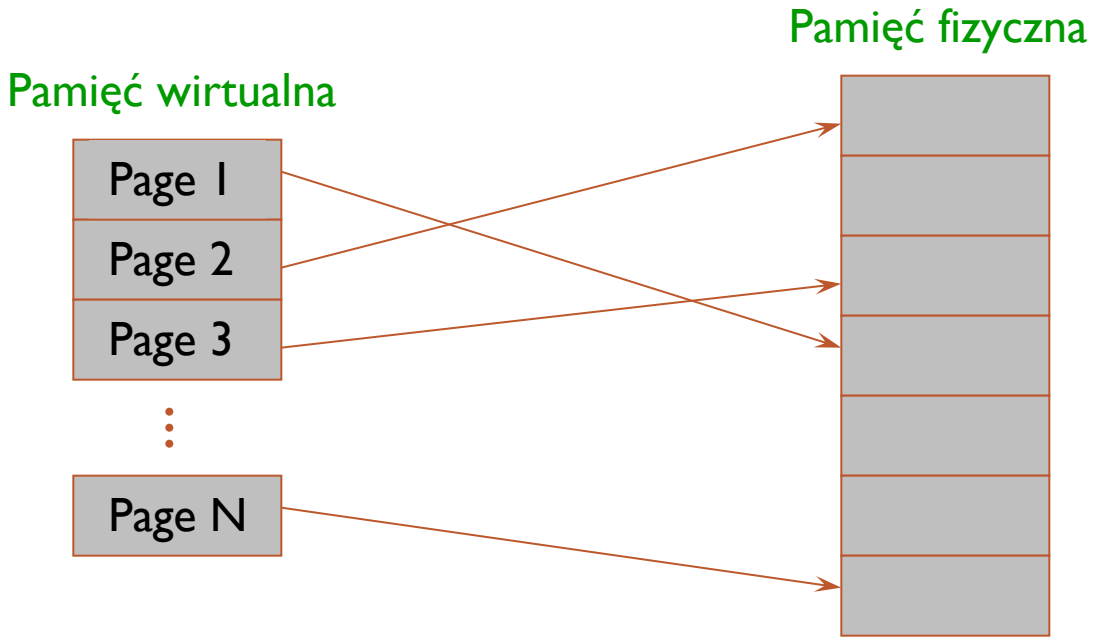
METODY POSZUKIWANIA ODPOWIEDNIEJ WOLNEJ PRZESTRZENI

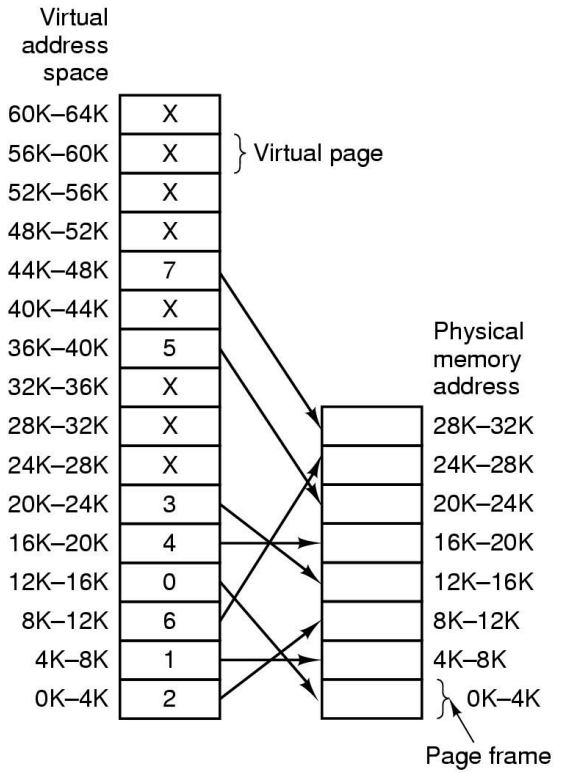
1. Pierwsza pasująca: zakończenie szukania przy napotkaniu pierwszej wolej przestrzeni odpowiedniej wielkości.
2. Najlepiej pasująca: przydział najmniejszego z dostatecznie dużych obszarów. Wykorzystuje się listę obszarów wolnych posortowanych według rozmiarów.
3. Najgorzej pasująca: przydziela się największy wolny obszar.

STRONICOWANIE

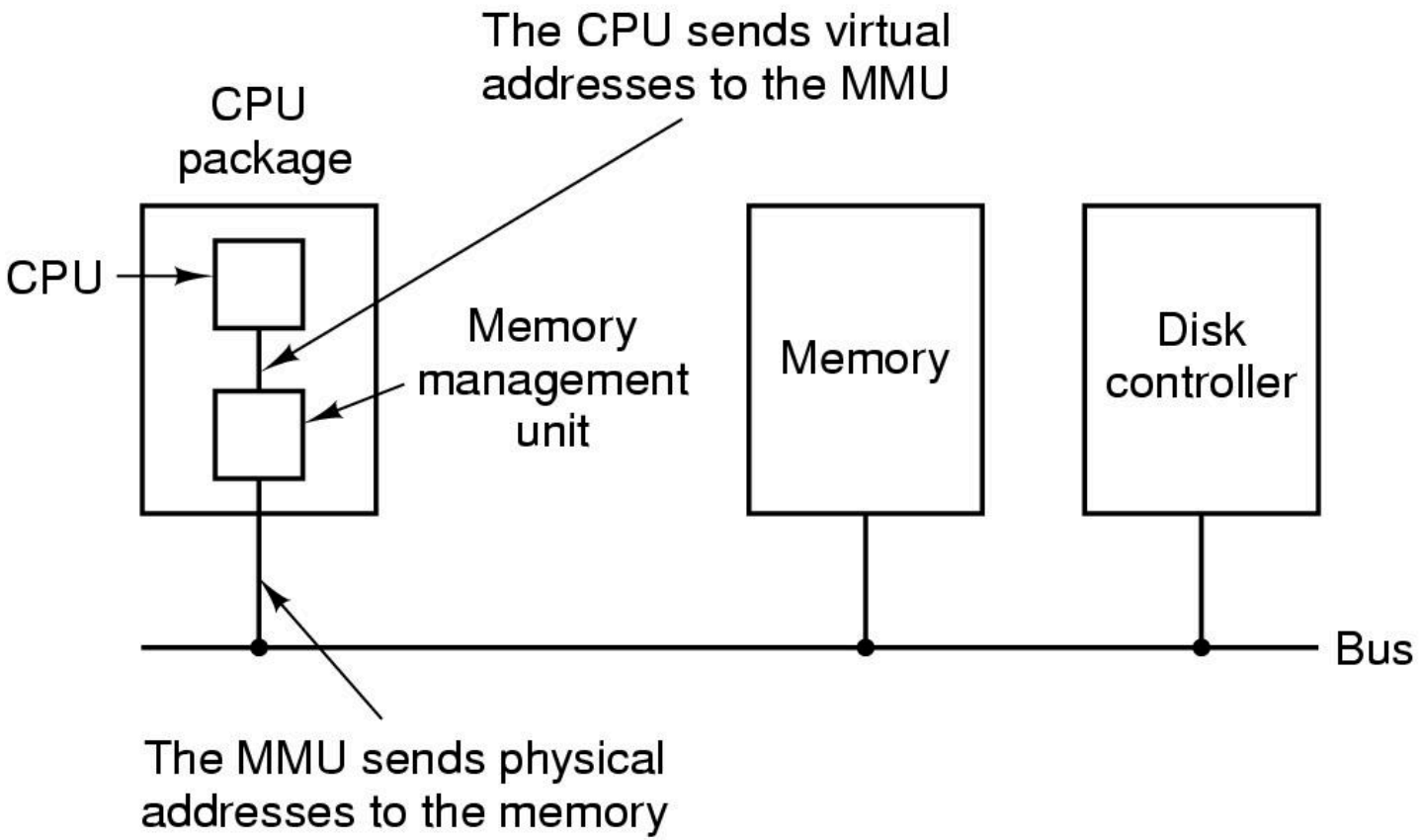
- Pozwala by pamięć procesu była nieciągła.
- Procesowi przydziela się dowolny wolny fragment pamięci.
- Niweluje się problem dopasowania rozmiaru procesu do rozmiaru wolnej przestrzeni w pamięci.
- Założenia:
  - Pamięć fizyczna podzielona na bloki o stałej długości – ramki.
  - Pamięć logiczna podzielona na stałej długości bloki – strony.
  - Podczas wykonania procesu jego strony są wprowadzane do wolnych ramek.
  - Pamięć pomocnicza podzielona na bloki o stałej długości tego samego rozmiaru co ramki.

STRONICOWANIE

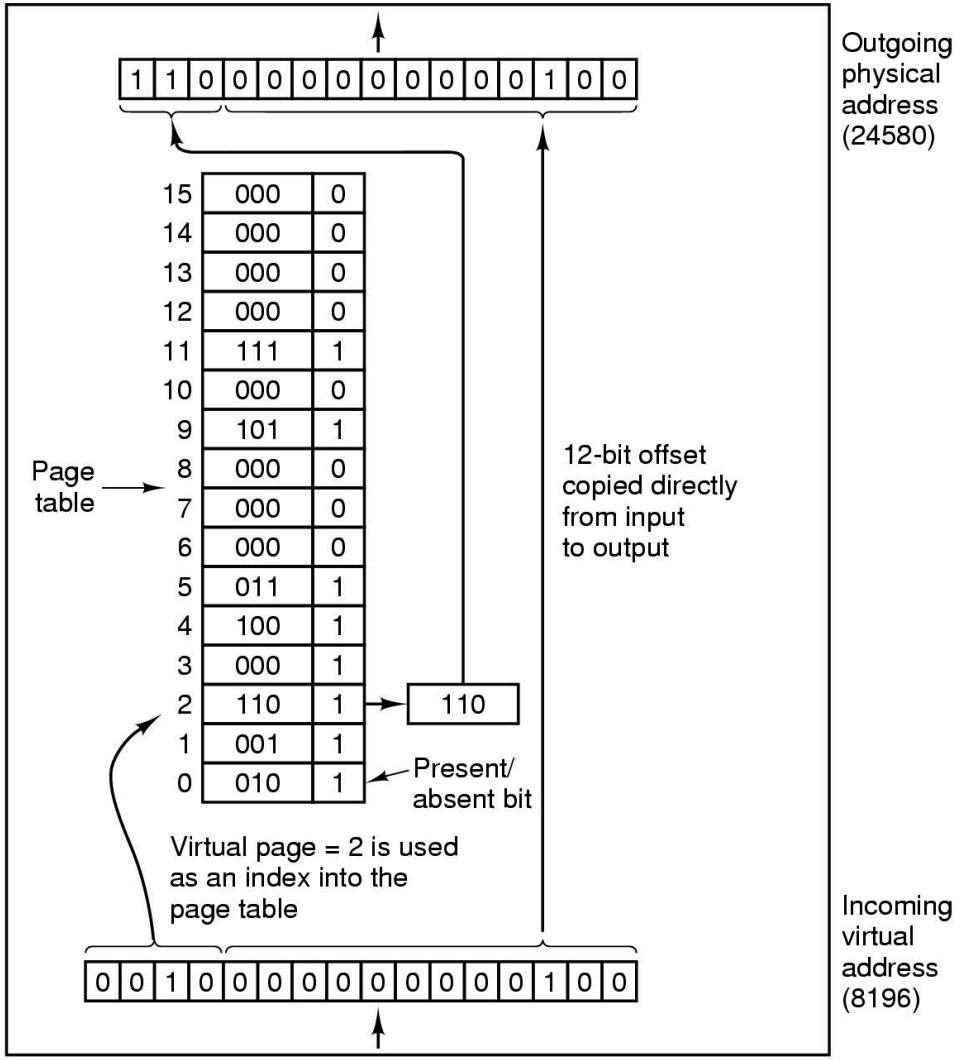




MMU JAKO CZĘŚĆ PROCESORA



WEWNĘTRZNE DZIAŁANIE MMU Z 16 STRONAMI 4 KB.



ROZMIAR STRONY

Rozmiar strony i ramki jest zależny od sprzętu.

Rozmiary są zazwyczaj potęgą liczby 2 (np. 512 czy 2048) co ułatwia tłumaczenie adresu logicznego na numer strony i odległości od niej.

np. Jeżeli rozmiar strony to 2^n słów to *n* najmniej znaczących bitów wskazuje odległość na stronie, pozostałe to numer strony.

Użytkownicy (i procesy) wyświetlają pamięć jako jedną ciągłą przestrzeń adresową od 0 do N - Wirtualna przestrzeń adresowa (VAS)

W rzeczywistości strony są rozproszone w całej pamięci fizycznej

Inna koncepcja niż zmienna partycja, gdzie pamięć fizyczna dla każdego procesu jest przydzielana w sposób ciągły.

Mapowanie jest niewidoczne dla programu

Ochrona jest zapewniona, ponieważ program nie może odwoływać się do pamięci poza swoim VAS

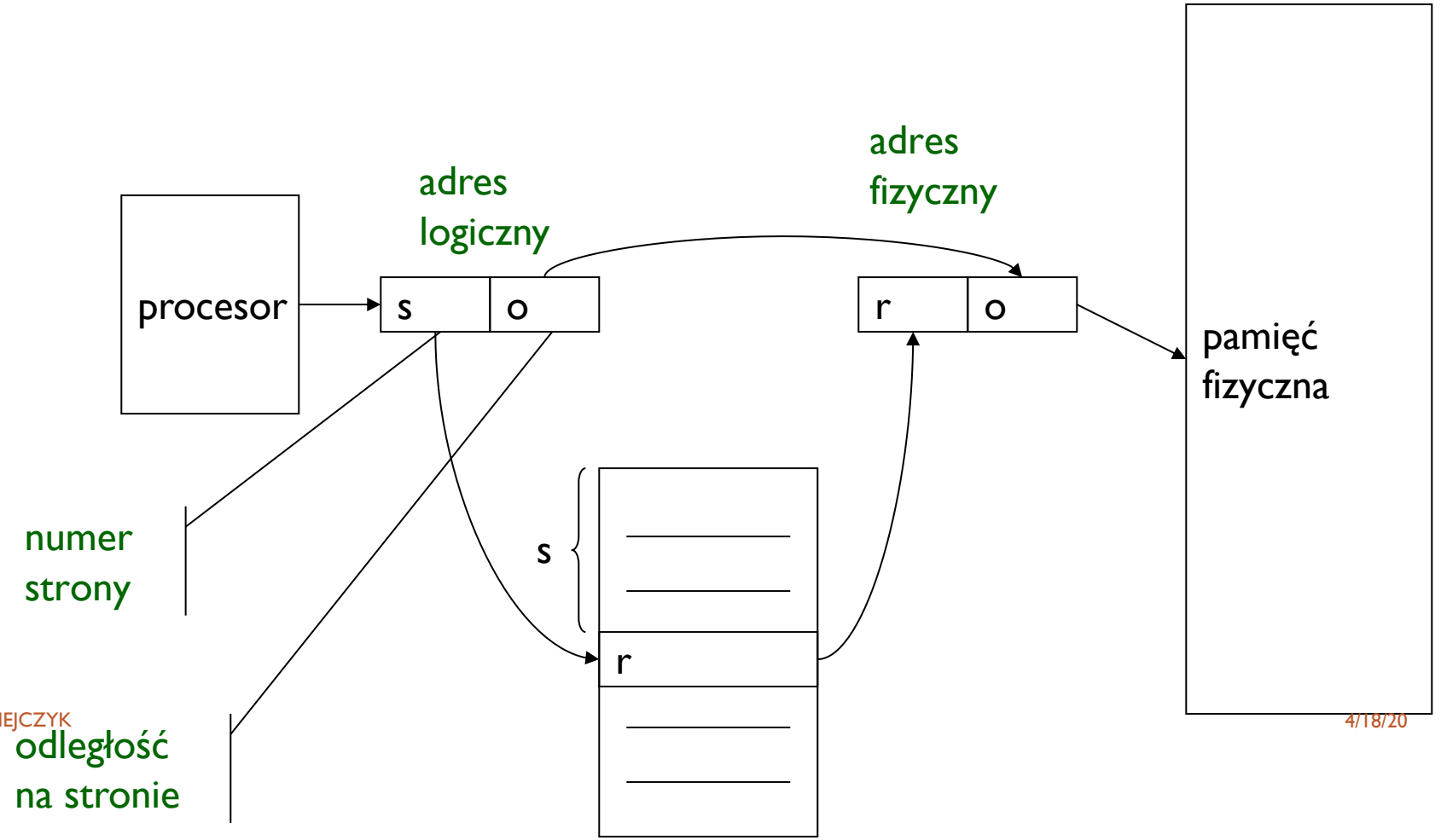
Adres „0x1000” jest mapowany na różne adresy fizyczne w różnych procesach.

WIDZENIE PAMIĘCI Z PERSPEKTYWY PROCESÓW

STRONICOWANIE – TŁUMACZENIE ADRESÓW

- Tłumaczenie adresów
  - Adres wirtualny składa się z dwóch części: numeru strony wirtualnej i przesunięcia
  - Numer strony wirtualnej (VPN - Virtual page number) to indeks do tabeli stron
  - Tabela stron określa numer ramki strony (PFN - page frame number)
  - Adres fizyczny to PFN :: offset
- Tabele stron
  - Mapują numer strony wirtualnej (VPN) na numer ramki strony (PFN)
    - VPN to indeks do tabeli, która określa PFN
  - Jedna strona (PTE - page table entry) na jedną stronę w wirtualnej przestrzeni adresowej (jeden PTE na jeden VPN)

TŁUMACZENIE ADRESÓW



PRZYKŁAD

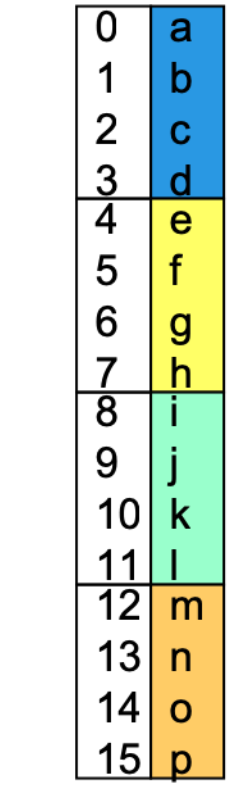
rozmiar strony: 4 słowa
pamięć fizyczna: 32 słowa = 8 stron

logiczny adres 0 = adres fizyczny 20
wynika to z (5*4)+0=20

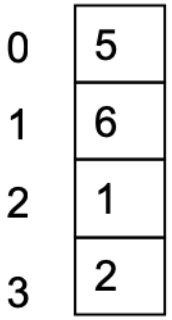
adres logiczny 3 = adres fizyczny 23
z (5*4)+3

adres logiczny 4 = adres fizyczny 24
z (6*4)+0

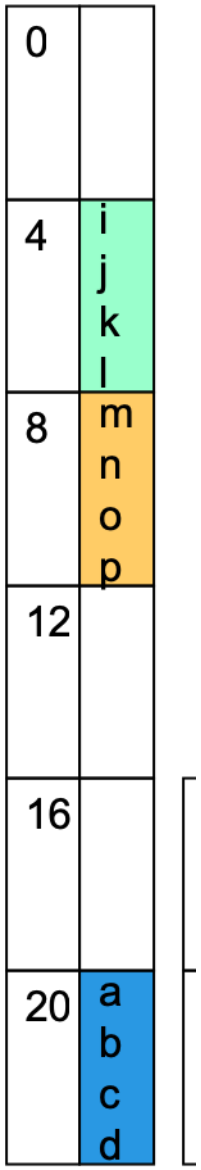
adres logiczny 13 = adres fizyczny 9
z (2*4)+1



pamięć logiczna

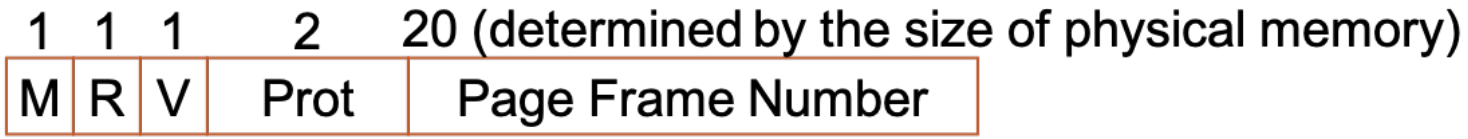


tablica stron



PAGE TABLE ENTRIES

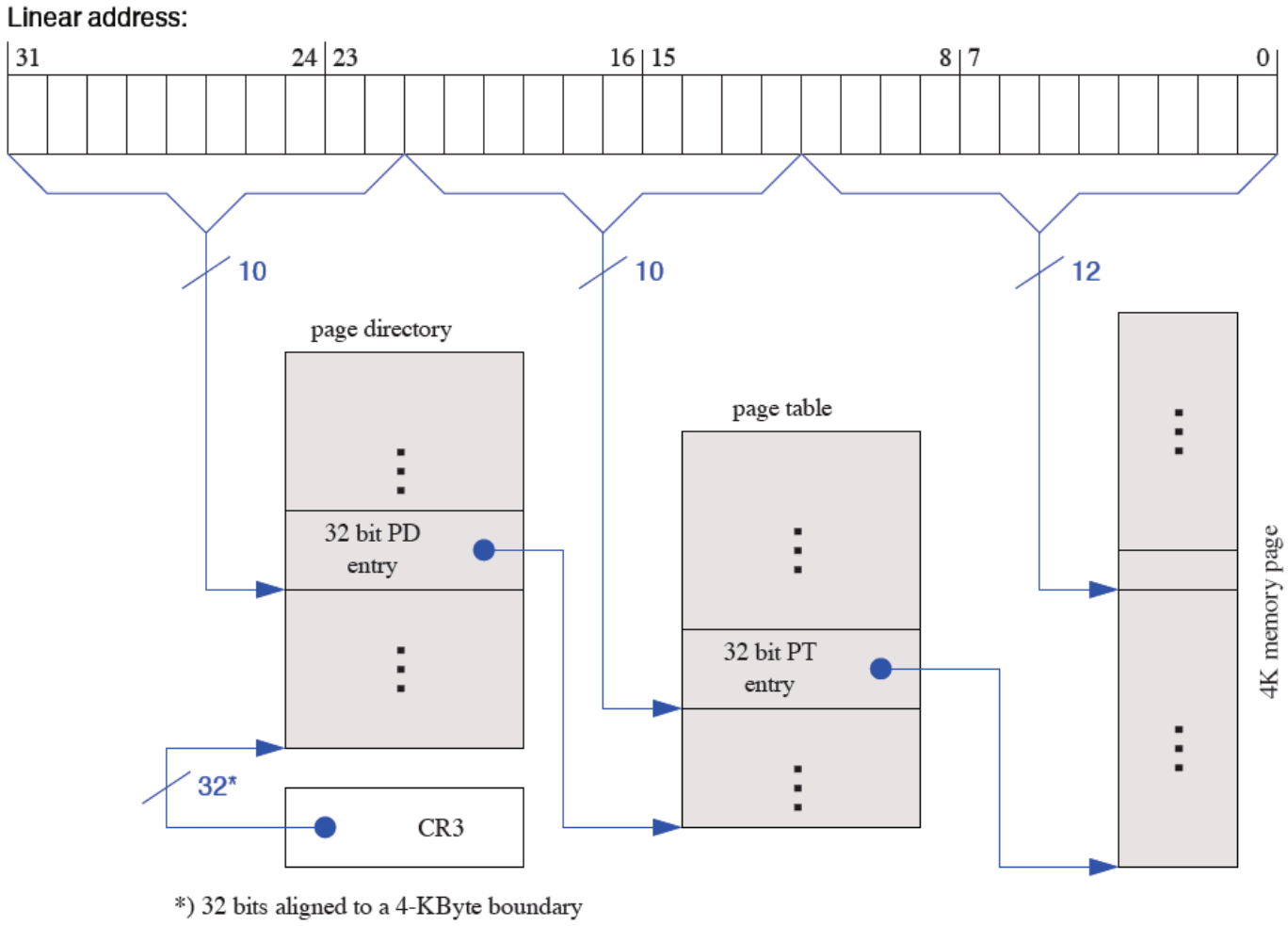
- Wpisy w tablicy stron - struktura
  - Bit Modyfikuj mówi, czy strona została zapisana
    - Jest ustawiany, gdy występuje zapis na stronie
  - Bit odniesienia mówi, czy strona została uzyskana
    - Jest ustawiany, gdy występuje odczyt lub zapis strony
  - Bit Valid mówi, czy PTE może być używany, czy nie
    - Jest sprawdzane za każdym razem, gdy używany jest adres wirtualny
  - Bity ochrony mówią, jakie operacje są dozwolone na stronie
    - Czytaj, pisz, wykonuj
  - Numer ramki strony (PFN) określa stronę fizyczną



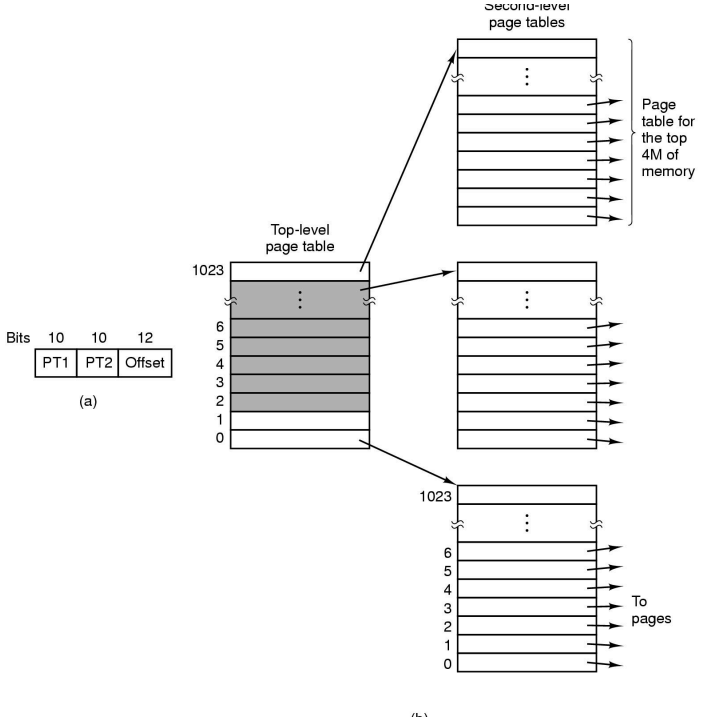
PRZYSPIESZENIE STRONICOWANIA

- Problemy z implementacją stronicowania:
  - Mapowanie z adresu wirtualnego na fizyczny adres musi być szybki.
  - Jeśli wirtualna przestrzeń adresowa jest duża, tablica stron będzie duża.

DWUPOZIOMOWE STRONICOWANIE



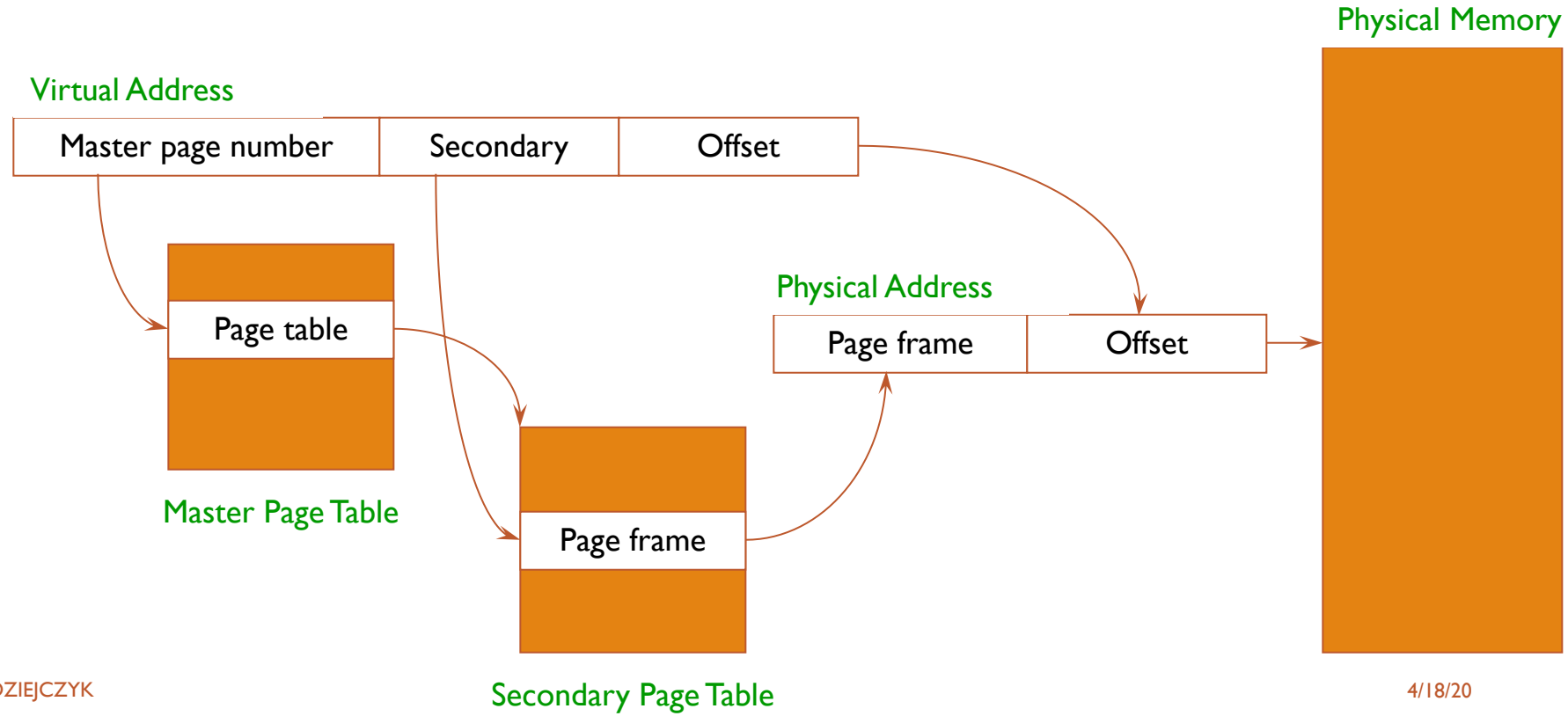
DWUPOZIOMOWE TABELE STRON



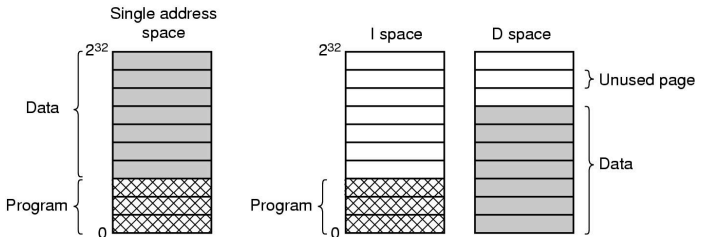
AUTOR: DR INŻ JOANNA KOŁODZIEJCZYK

- Adresy wirtualne (VA) mają trzy części:
  1. Numer strony głównej - odwzorowuje VA na tabelę stron drugiego poziomu
  2. numer strony drugiego poziomu - mapuje numer strony na stronę fizyczną
  3. Offset - wskazuje, gdzie znajduje się adres strony fizycznej
- Przykład
  - 4K stron, 4 bajty / PTE
  - Ile bitów w offsecie? 4K = 12 bitów
  - Chcemy tabelę strony pierwszego rzędu zmieścić na jednej stronie: 4K / 4 bajty = 1K wpisów
  - Stąd, 1K tabel drugiego poziomu. Ile bitów?
  - Master (1K) = 10, offset = 12, drugiego poziomu = 32 - 10 - 12 = 10 bitów

PRZYKŁAD

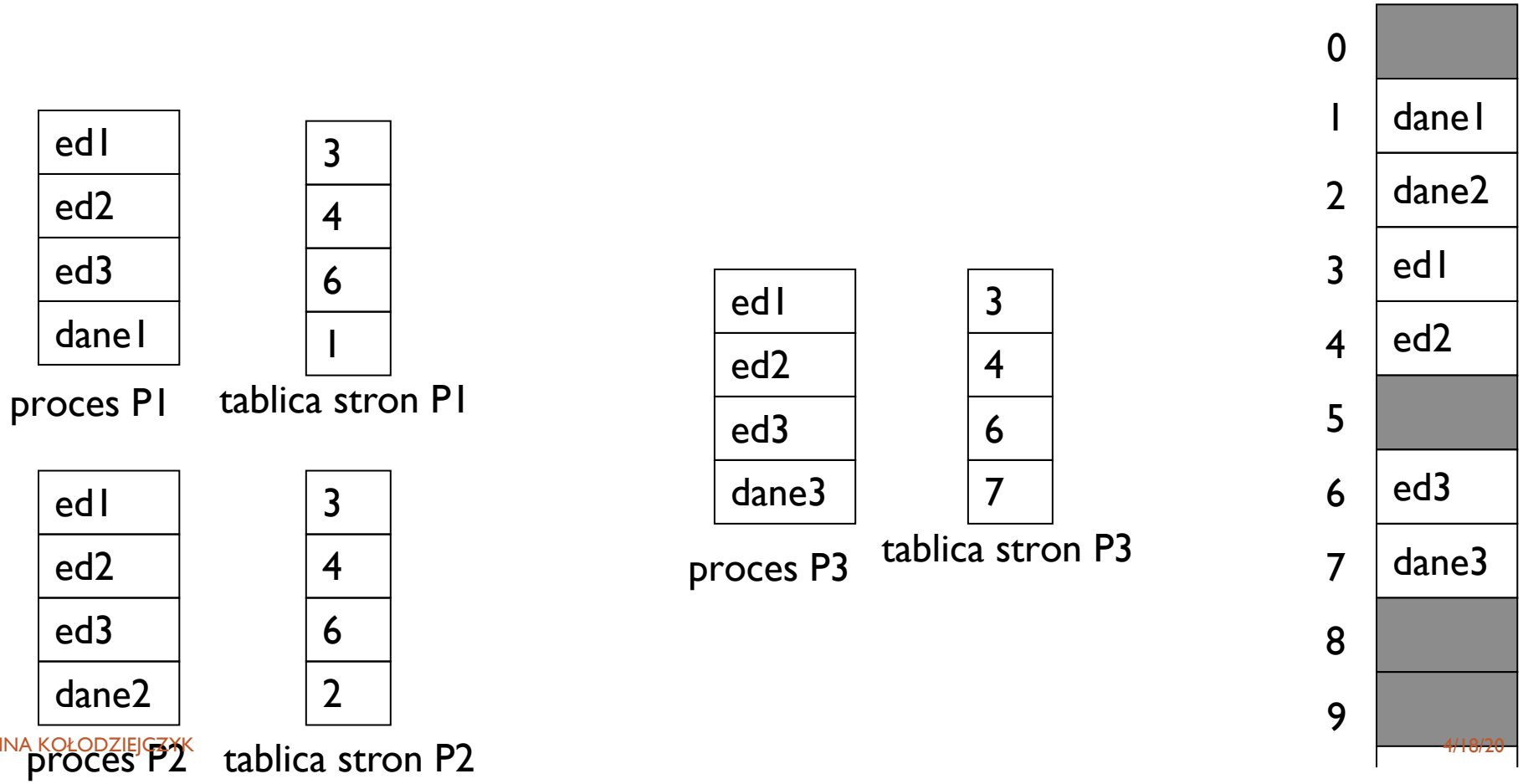

Physical Memory
Virtual Address
Master page number
Secondary
Offset
Page table
Physical Address
Page frame
Offset
Master Page Table
Page frame
Secondary Page Table

STRONY WSPÓŁDZIELONE

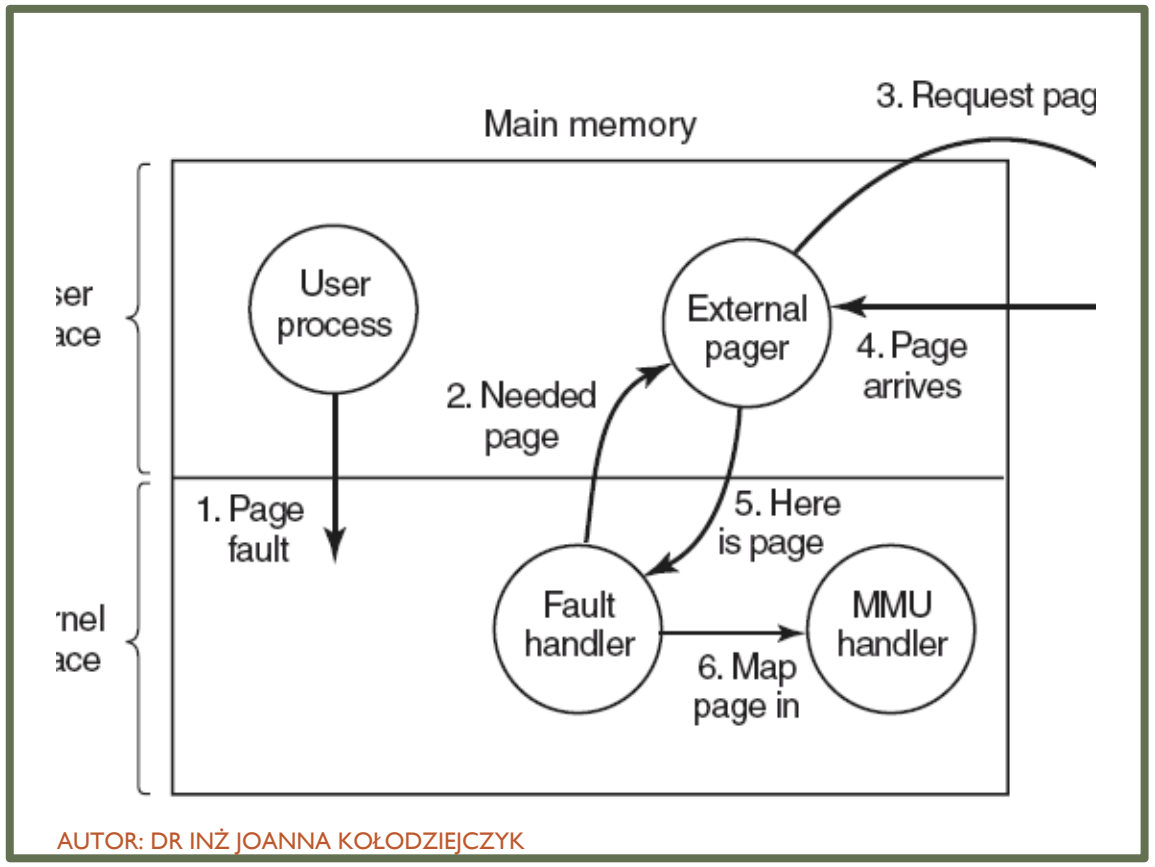


- Stronicowanie pozwala na dzielenie tych samych danych.
- Przykład bez dzielenia wspólnego kodu: 40 użytkowników używa edytora (150KB kodu i 50KB dane) = wymagane 8000KB.
- Jeżeli kod programu jest wznawialny (czysty) innymi słowy nie modyfikuje sam siebie, to kilka procesów może wykonywać ten sam kod w tym samym czasie. W pamięci przechowuje się jedną kopię edytora. np.. dla podanego przykładu 50KB * 40 użytkowników + 150KB=2150KB.

DZIELENIE KODU W ŚRODOWISKU STRONICOWANYM



ODDZIELENIE POLITYKI I MECHANIZMÓW STRONICOWANIA



- System zarządzania pamięcią jest podzielony na trzy części:
  - Niski poziom obsługi MMU.
  - Obsługa błędów stron, która jest częścią jądra.
  - Zewnętrzny pager działający w przestrzeni użytkownika.

Łatwa alokacja pamięci

- Pamięć pochodzi z listy ramek o stałym rozmiarze
- Przydzielanie strony po prostu usuwa ją z listy wolnych ramek
- Fragmentacja zewnętrzna nie stanowi problemu

Łatwa zamiana fragmentów programu

- Wszystkie strony są tego samego rozmiaru
- Użyj bitu poprawności, aby wykryć odniesienia do zamienionych stron
- Strony są wygodną wielokrotnością rozmiaru bloku dysku

ZALETY STRONICOWANIA

WADY STRONICOWANIA

- Nadal może mieć wewnętrzną fragmentację
  - Ostatnia ramka procesu może nie być wypełniona gdy rozmiar całkowity procesu nie jest wielokrotnością rozmiaru ramki. (np. proces=72766B, ramka 2048B- potrzeba: 36 ramek i 1086B). Efekt ten nazywany jest wewnętrzną fragmentacją (2048-1086=962B niewykorzystane).
- Narzut obliczeniowy na odniesienia do pamięci
  - 2 odwołania do wyszukiwania adresu (tabela stron, a następnie pamięć)
  - Jeszcze więcej dla dwupoziomowych tabel stron!
  - Rozwiązanie - użyj sprzętowej pamięci podręcznej
- Pamięć wymagana do przechowywania tabeli stron może być znacząca
  - Potrzebujesz jednego PTE na stronę
  - 32-bitowa przestrzeń adresowa w / 4KB strony = 220 PTE
  - 4 bajty / PTE = 4 MB / strona tabeli
  - 25 procesów = 100 MB tylko dla tabel stron!
  - Pamiętaj: każdy proces ma swoją własną tabelę stron!
  - Rozwiązanie - 2-poziomowe tabele stron

STRONICOWANIE – PODSUMOWANIE

Użytkownik postrzega pamięć swojego procesu jako spójną i jako obszar ciągły.

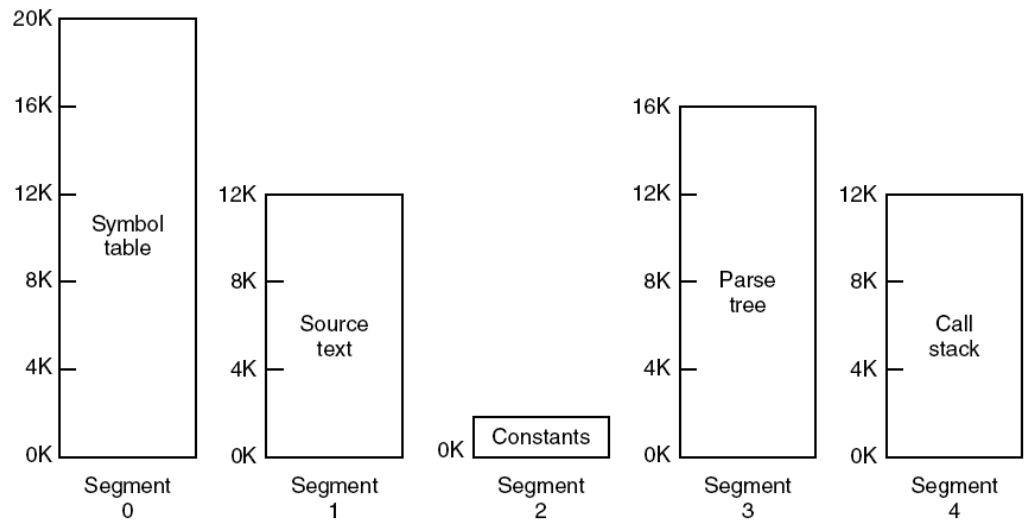
W rzeczywistości strony porozrzucane są po pamięci fizycznej.

Sprzęt tłumaczący adresy logiczne na fizyczne jest nadzorowany przez system operacyjny.

System operacyjny musi znać wszystkie szczegóły dotyczące pamięci fizycznej: które ramki są przydzielone, a które wolne i jaka jest sumaryczna liczba ramek. Dane te przechowuje się w tablicy ramek.

System kontroluje czy procesy użytkowników działają w przestrzeni użytkowników, a adresy logiczne są odwzorowywane w fizyczne.

SEGMENTACJA INNY WARIANT PRZYDZIAŁU



- Segmentacja to technika dzieląca pamięć na logicznie powiązane jednostki danych:
  - Moduł, procedury, stos, dane, pliki itp.
- Adresy wirtualne mają formę <segment #, offset>
- Naturalne rozszerzenie na bloki o zmiennej wielkości
  - Partycje o zmiennej wielkości = 1 segment / proces
  - Segmentacja = wiele segmentów / proces
- Obsługa sprzętowa dla segmentacji
  - Wiele par rejestrów bazowych / granicznych, po jednej na segment (tabela segmentów)
  - Numer segmentów #, używane do indeksowania w tabeli

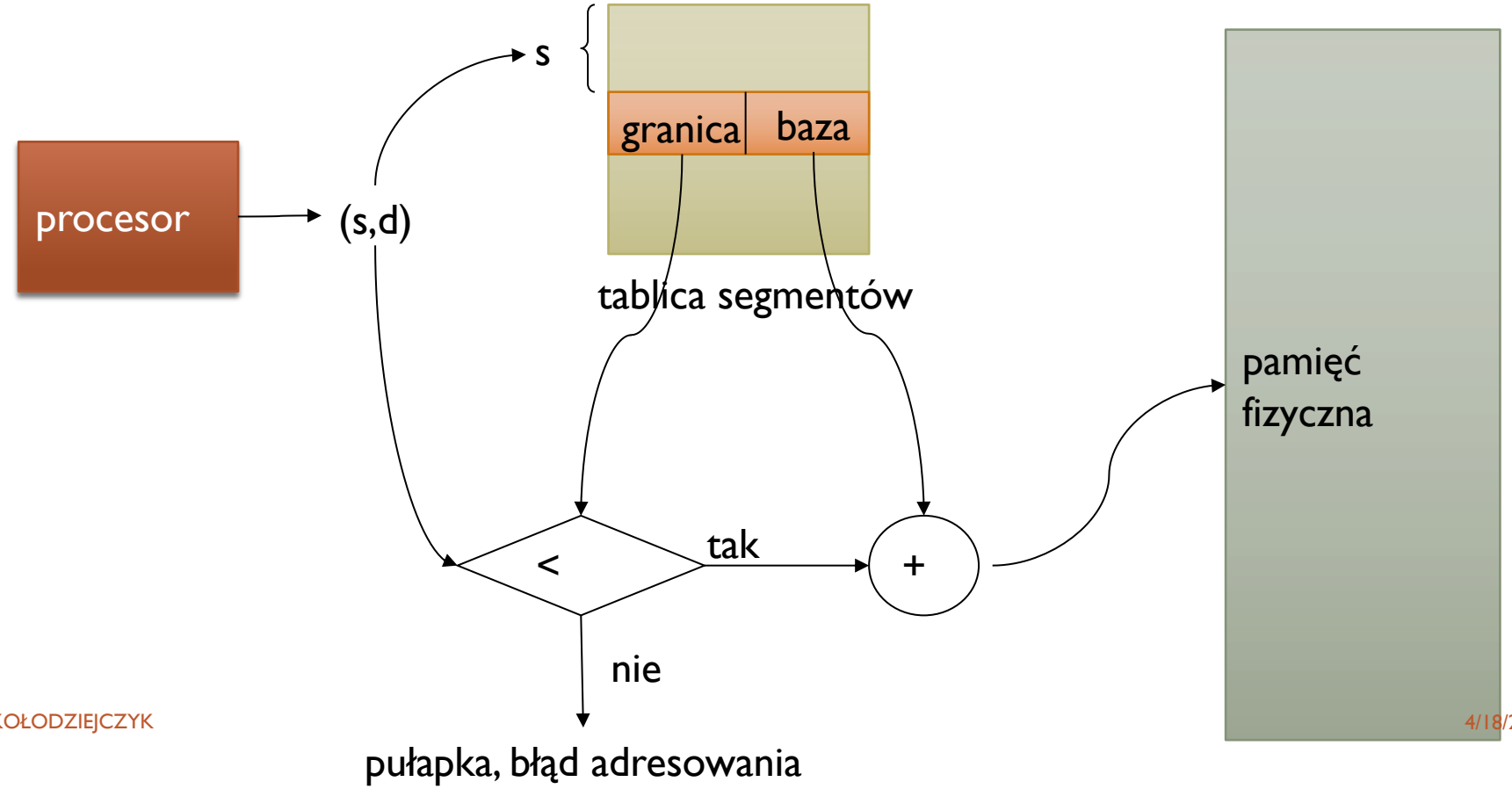
PORÓWNANIE SEGMENTACJI I

Problem	Stronicowanie	Segmentacja
Czy programista musi by świadomy tego, że jest stosowana dana technika?	NIE	TAK
Ile liniowych przestrzeni adresowych występuje?	1	Wiele
Czy całkowita przestrzeń adresowa może przekroczyć rozmiar pamięci fizycznej?	TAK	TAK
Czy procedury można odróżnić od danych i czy można je oddzielnie zabezpieczyć?	NIE	TAK
Czy można w łatwy sposób obsłużyć tabele, których rozmiar się zmienia?	NIE	TAK
Czy istnieje możliwość współdzielenia procedur między użytkownikami?	NIE	TAK
Po co opracowano tą technikę	Aby uzyskać obszerną liniową przestrzeń adresową bez konieczności posiadania dużej fizycznej pamięci	Aby umożliwić podział programów i danych na logiczne niezależne przestrzenie, ułatwienie kontroli dostępu (zabezpiecznia)

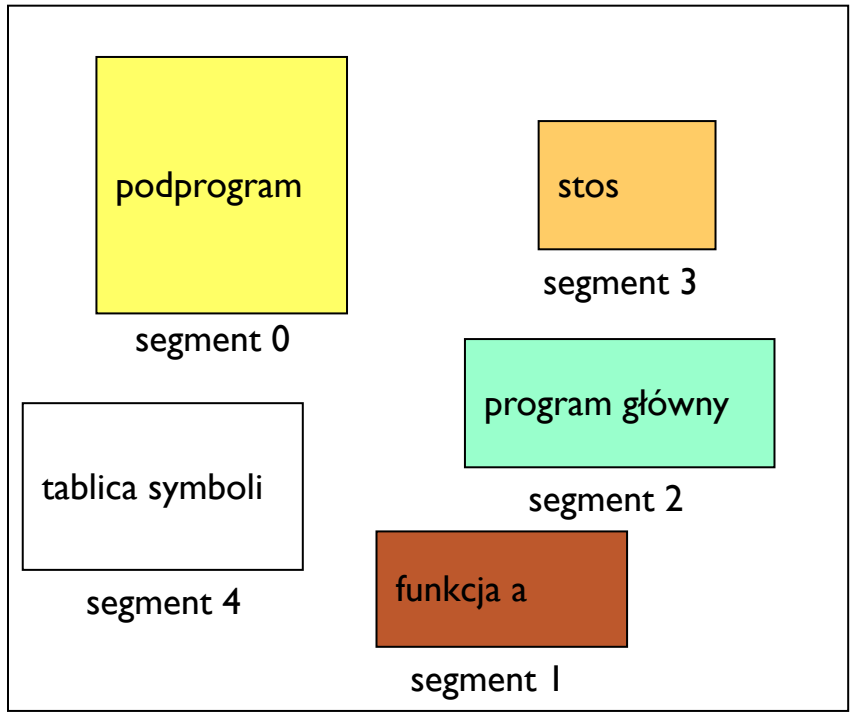
SEGMENTACJA - SZCZEGÓŁY

- Schemat zarządzania pamięcią odwzorowujący segmentację programu z punktu widzenia użytkownika.
- Przestrzeń adresów logicznych jest zbiorem segmentów.
- Każdy segment ma: nazwę i długość.
- Narzędzia do segmentacji
  - Odwzorowanie dwuwymiarowego adresu segmentu na pamięć fizyczną realizuje się za pomocą tablicy segmentów.
  - Adres logiczny: *s* – numer segmentu i *d* –odległość w tym segmencie.
  - *s* jest indeksem w tablicy segmentów
  - Każda pozycja w tablicy segmentów składa się z *bazy* i *granicy* segmentu (para rejestrów).

ILUSTRACJA SEGMENTACJI



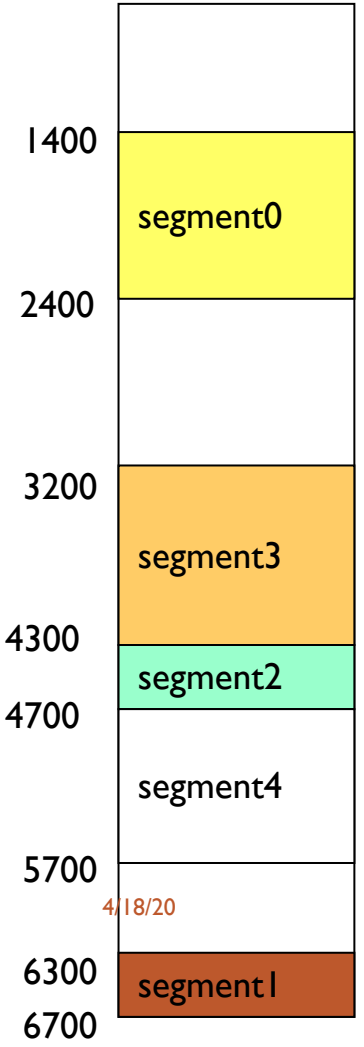
PRZYKŁAD SEGMENTACJI



przestrzeń adresów logicznych

	Granica	baza
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

tablica segmentów



IMPLEMENTACJA TABLICY SEGMENTÓW

Tablica segmentów może być umieszczona w pamięci albo w szybkich rejestrach.

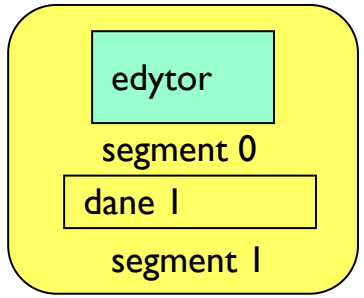
Ze względu na rozmiar tablicy segmentów niemożliwe jest przechowywanie jej w rejestrze. Rejestr przechowuje bazę tablicy segmentów i jej długość.

Można wykorzystać rejestry asocjacyjne pamięci podręcznej, by uniknąć dwukrotnego sięgania do pamięci (analogicznie do tablicy stron).

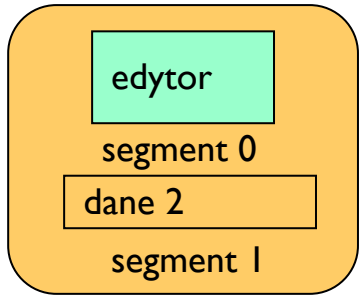
OCHRONA ORAZ DZIELENIE KODU I DANYCH W SEGMENTACJI

- Segmenty składają się z semantycznie spójnych danych. Zatem segment kodu będzie cały dostępny tylko do czytania, a segment danych do czytania i pisania. Kontrolują to bity dostępu.
- Dzielenie segmentów następuje, gdy dwa procesy wskazują na to samo miejsce w pamięci fizycznej.

PRZYKŁAD DZIELENIA SEGMENTÓW DLA DWÓCH PROCESÓW



pamięć logiczna
proces P1



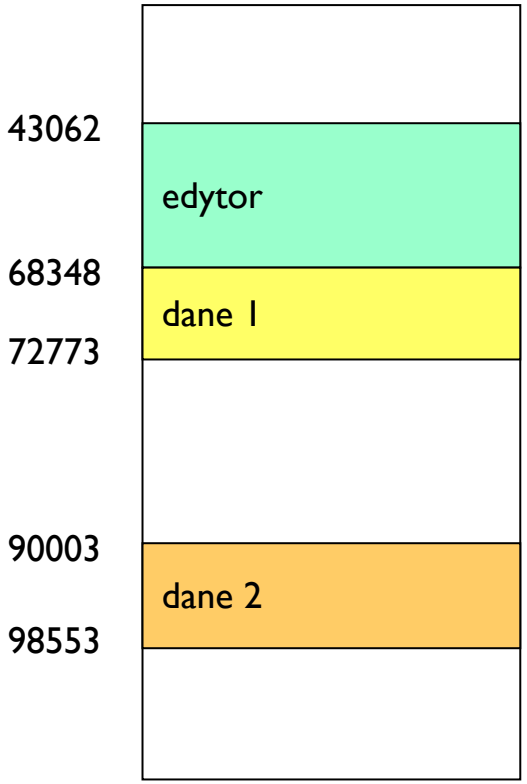
pamięć logiczna
proces P2

	granica	baza
0	25286	43062
1	4425	68348

tablica segmentów
proces P1

	granica	baza
0	25286	43062
1	8550	90003

tablica segmentów
proces P2



FRAGMENTACJA PAMIĘCI W SEGMENTACJI

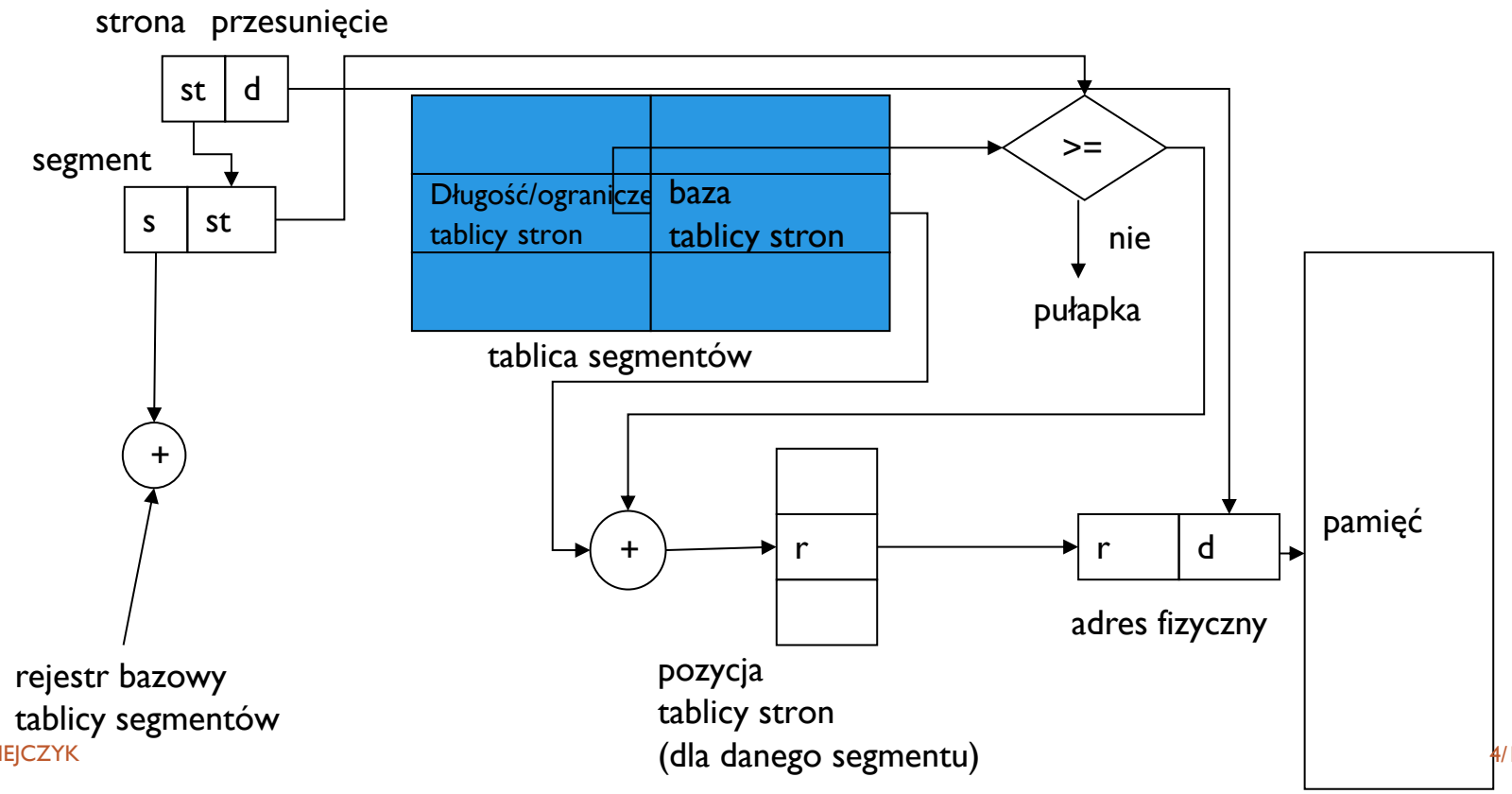
Segmenty w przeciwieństwie do stron mają zmienne długości.

Pojawiający się problem fragmentacji dotyczy fragmentacji zewnętrznej, gdy każdy z wolnych bloków jest za mały by pomieścić cały segment.

O tym jak silny wpływ ma segmentacja decyduje rozmiar segmentu. Dwa skrajne przypadki to:

- segment równy jest wielkości procesu (pierwsza omówiona metoda obszarów o zmiennej długości),
- segment równy pojedynczemu słowu (za dużo rejestrów bazowych) (stronicowanie).

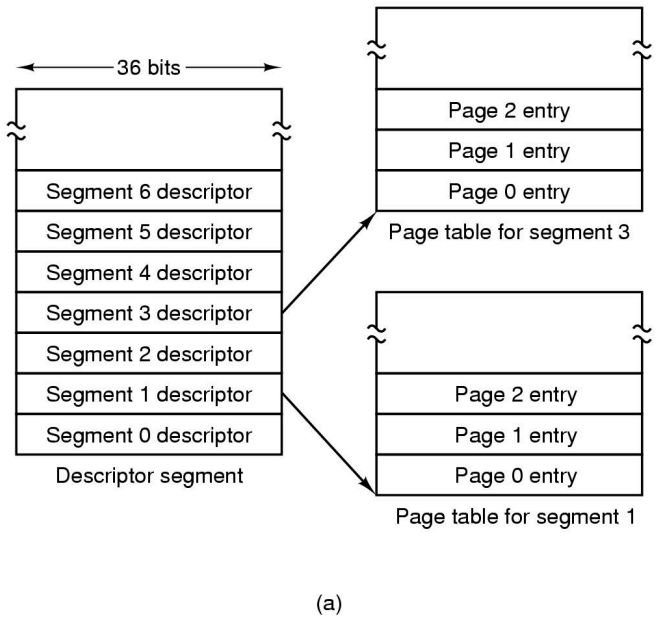
SEGMENTACJA STRONICOWANA – ROZWIĄZANIE PROBLMÓW



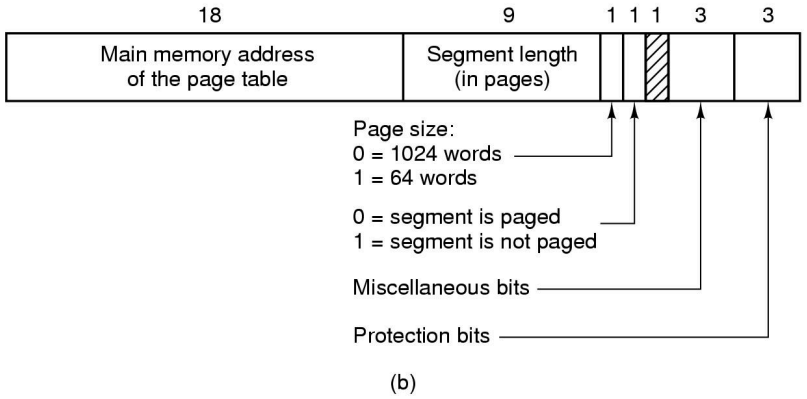
SEGMENTACJA STRONICOWANA

- Może łączyć segmentację i stronicowanie
  - Linux obsługuje tylko segmentację w mikroprocesorach 80x86: stwierdza się, że stronicowanie upraszcza zarządzanie pamięcią poprzez użycie tego samego zestawu adresów liniowych. PENTIUM obsługuje segmenty i stronicowanie
- Używa segmentów do zarządzania logicznie powiązanymi jednostkami: Moduł, procedura, stos, plik, dane itp.
  - Segmenty różnią się rozmiarem, ale zazwyczaj są duże (wiele stron)
- Używa stron do podziału segmentów na porcje o stałym rozmiarze
  - Ułatwia zarządzanie segmentami w pamięci fizycznej
  - Segmenty stają się „stronicowalne” - zamiast przenosić segmenty do i z pamięci, wystarczy przenieść części strony segmentu
  - Należy przydzielić wpisy tabeli stron tylko dla tych segmentów, które same zostały przydzielone
- Jest skomplikowanym rozwiązaniem

SEGMENTACJA ZE STRONICOWANIEM MULTICS



(a)

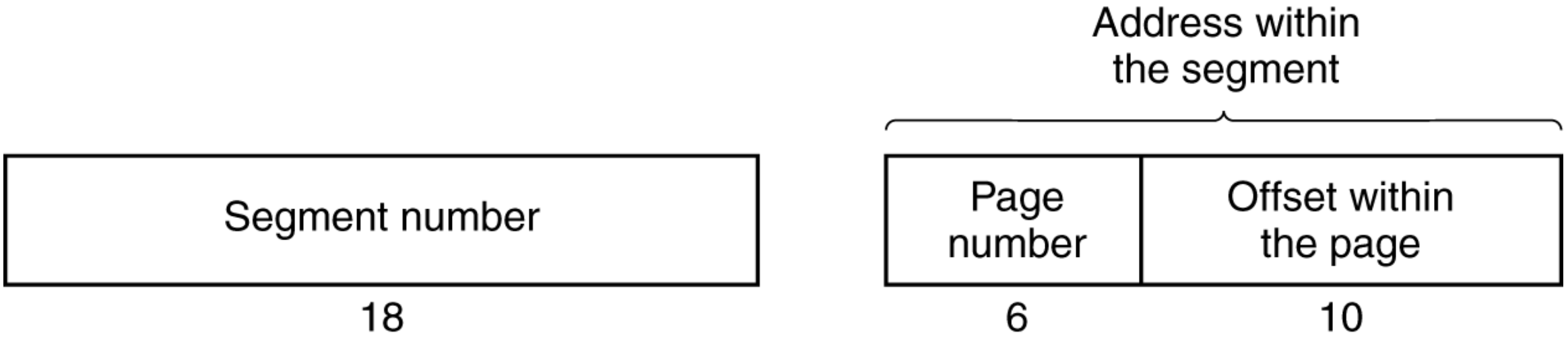


(b)

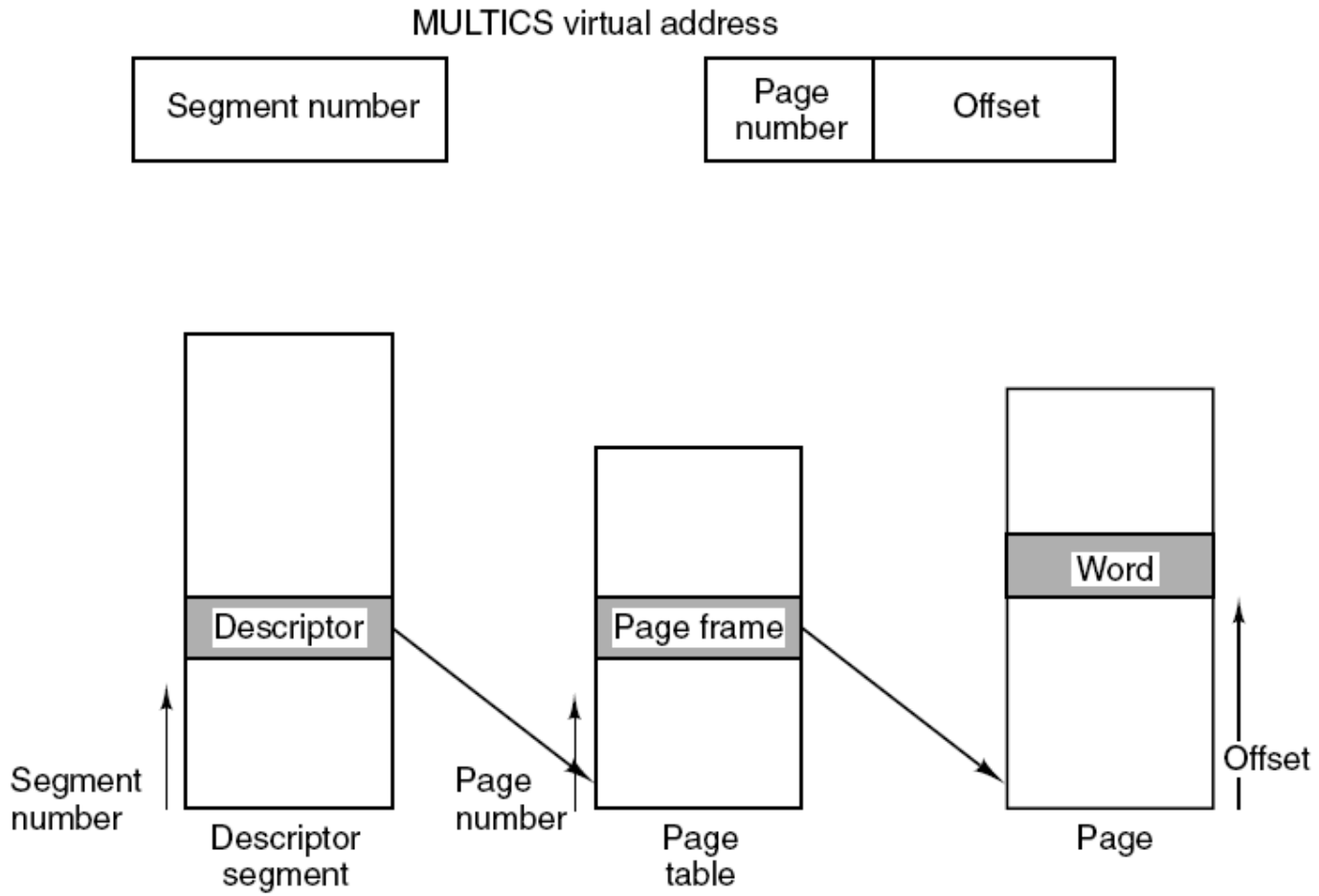
SEGMENTACJA ZE STRONICOWANIEM MULTICS

- Gdy pojawi się odwołanie do pamięci, wykonywany jest następujący algorytm:
  - Numer segmentu jest używany do znalezienia deskryptora segmentu.
  - Sprawdzane jest, czy tablica stron segmentu znajduje się w pamięci.
    - Jeśli nie, wystąpi błąd segmentu.
    - W przypadku naruszenia ochrony występuje błąd (pułapka).
  - Test wpisu tablicy stron dla badanej strony wirtualnej.
    - Jeśli sama strona nie znajduje się w pamięci, wywoływany jest błąd strony.
    - Jeśli jest w pamięci, główny adres pamięci początku strony jest wyodrębniany z wpisu tablicy strony
  - Przesunięcie jest dodawane do początku strony, aby uzyskać główny adres pamięci, w którym znajduje się słowo.
  - Odczytaj lub zachowaj.

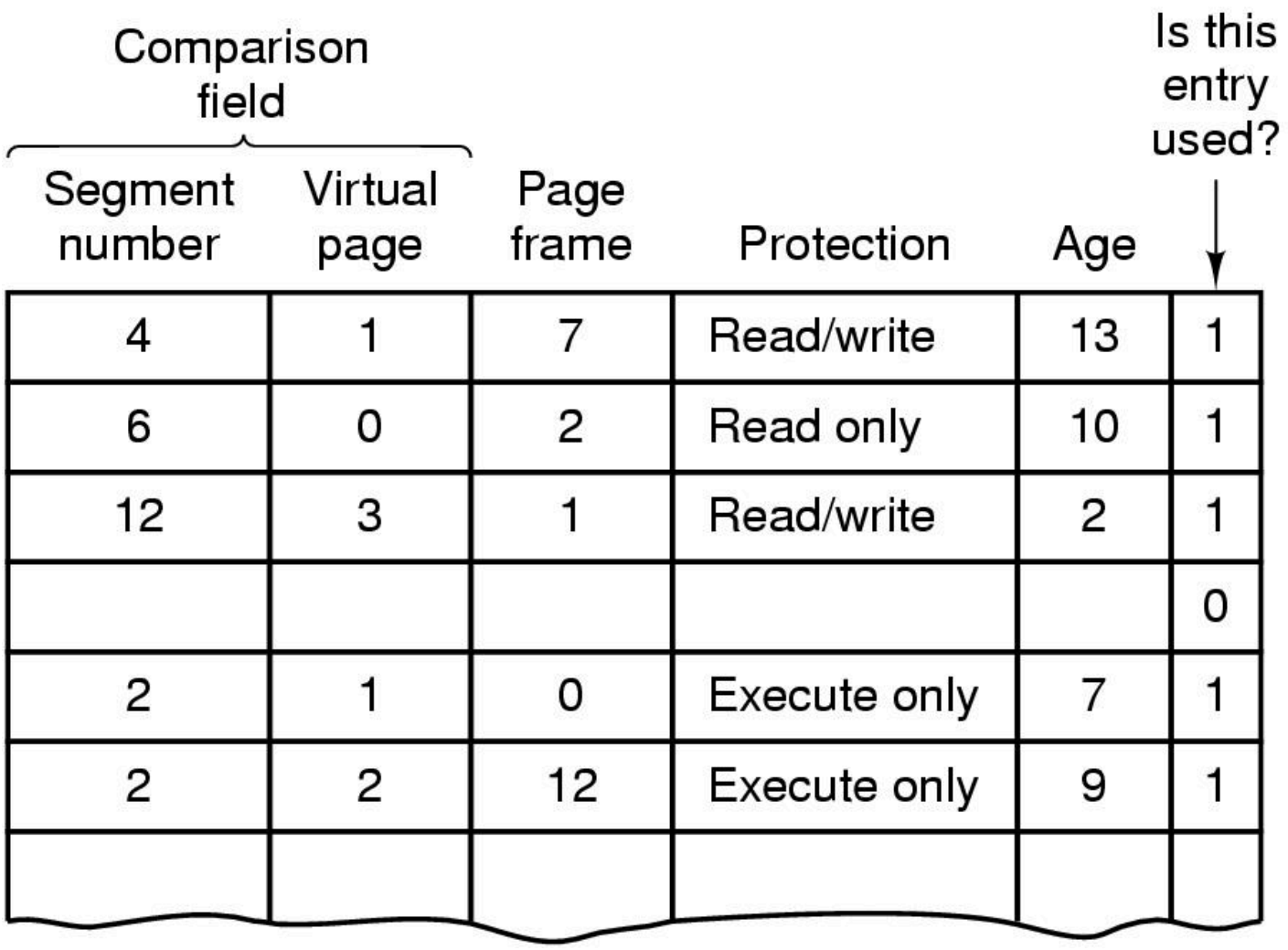
34-BIT MULTICS VIRTUAL ADDRESS



KONWERSJA NA ADRES FIZYCZNY MULTISC



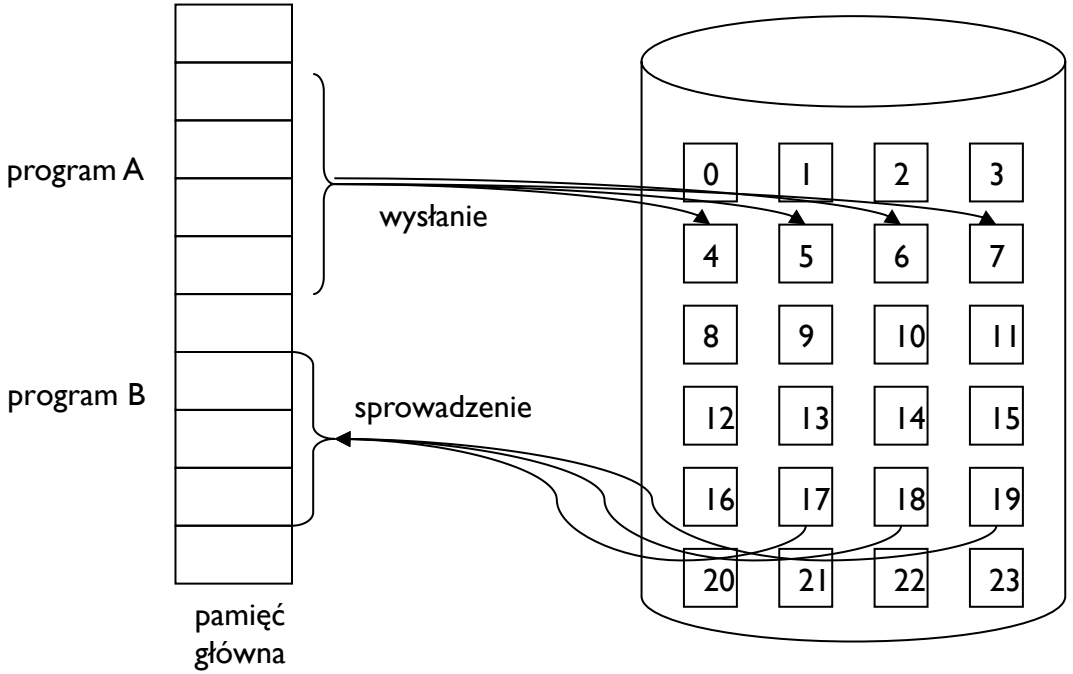
MULTICS TLB



Comparison field: Segment number	Comparison field: Virtual page	Page frame	Protection	Age	Is this entry used?
4	1	7	Read/write	13	1
6	0	2	Read only	10	1
12	3	1	Read/write	2	1
					0
2	1	0	Execute only	7	1
2	2	12	Execute only	9	1

CO JEŚLI PROCES WYMAGA WIĘCEJ PAMIĘCI NIŻ JEST DOSTĘPNEJ W PAMIĘCI FIZYCZNEJ?

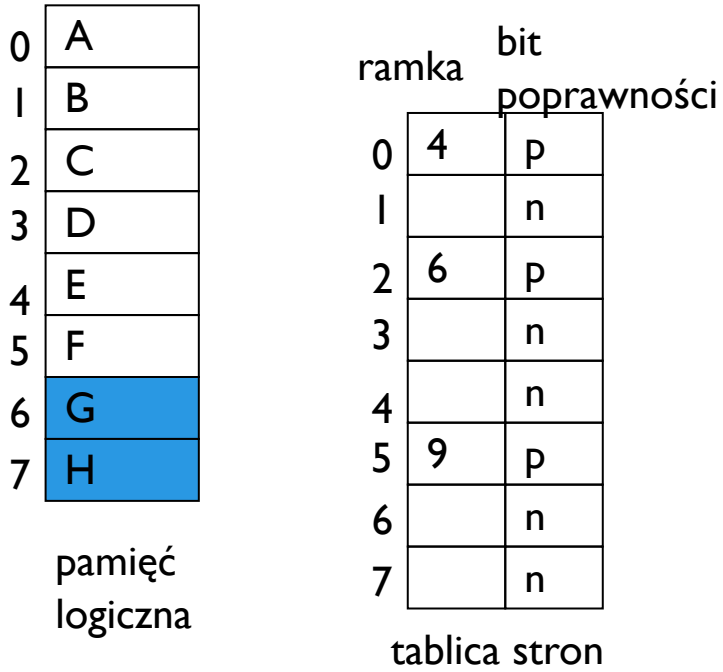
- Zamiana (swapping)
  - Przenieś jedną / kilka / wszystkie strony procesu na dysk
    - Zwolnij pamięć fizyczną
    - „Strona” to jednostka wymiany
  - Uwolniona pamięć fizyczna może być mapowana na inne strony
  - Procesy, które używają dużej pamięci, mogą być wymieniane



STRONICOWANIE NA ŻĄDANIE

- Stronicowanie na żądanie podobne do stronicowania z wymianą.
- Procesy rezydują w pomięci pomocniczej (dysk).
- Proces, który należy wykonać wprowadza się do pamięci głównej, a nadzoruje to program zmieniania stron.
- Zostaje sprawdzone, które strony procesu są niezbędne (w użyciu). Skraca to czas wymiany.
- Ochrona i kontrola:
  - W tablicy stron umieszcza się bit poprawności odniesienia.
    - wartość bitu: poprawny – strona znajduje się w pamięci operacyjnej;
    - wartość bitu: niepoprawny – strona znajduje się na dysku.
  - Jeżeli w trakcie wykonania procesu strona ma znacznik niepoprawna, to znaczy, że nie było potrzebna i nie została wprowadzona do pamięci operacyjnej.

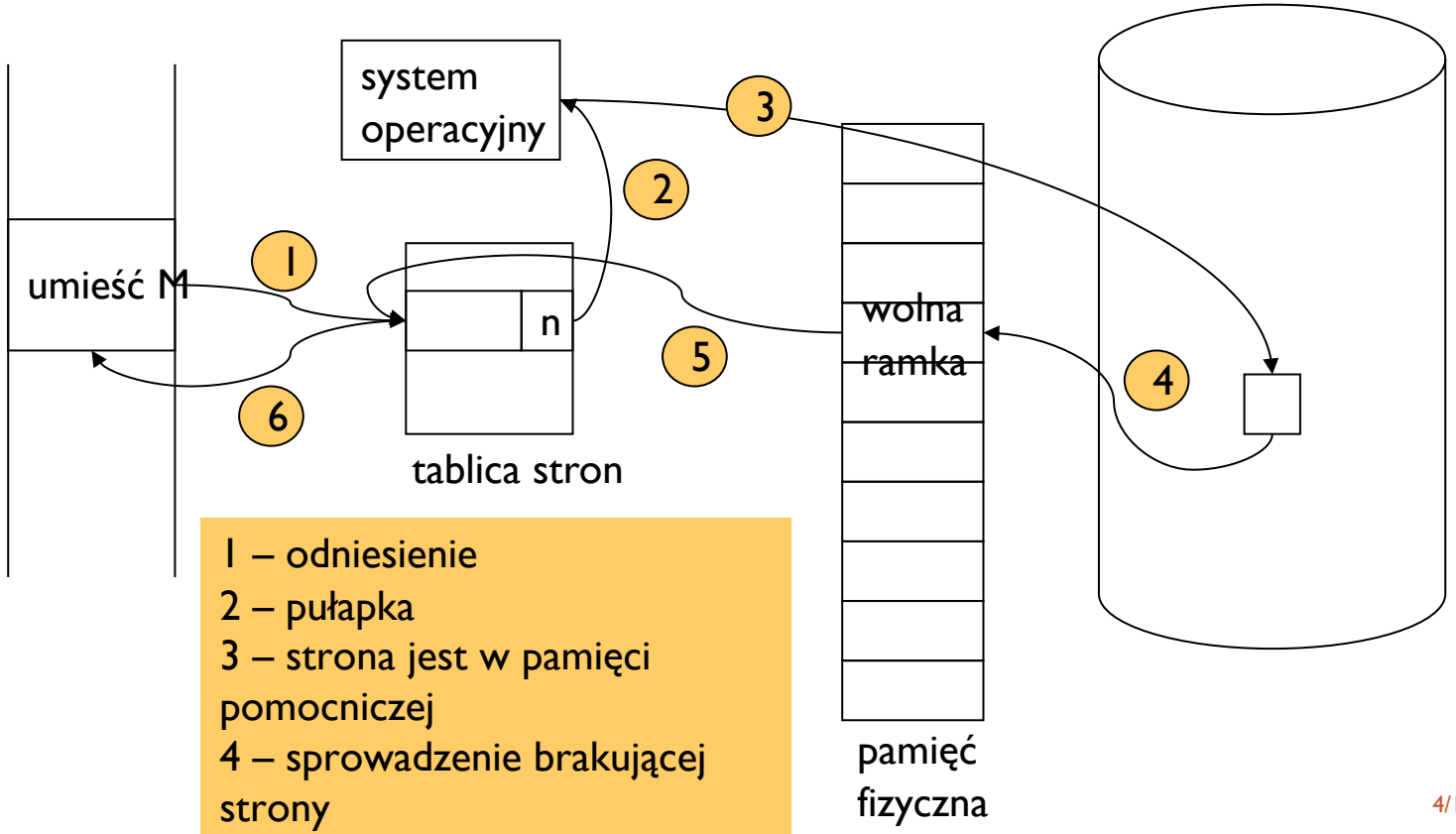
ILUSTRACJA


0 A
1 B
2 C
3 D
4 E
5 F
6 G
7 H
pamięć logiczna
ramka
bit poprawności
0 4 p
1 n
2 6 p
3 n
4 n
5 9 p
6 n
7 n
tablica stron
0
1
2
3
4 A
5
6 C
7
8
9 F
10
11
12
pamięć fizyczna
A B
C D E
F

ALGORYTM SPROWADZANIA STRON DO PAMIĘCI OPERACYJNEJ

1. Sprawdź bit poprawności w tablicy stron.
2. Jeżeli bit poprawny to odczytujemy stronę.
3. Jeżeli bit niepoprawny to powstanie pułapka „błąd strony”.
4. Sprawdzamy wewnętrzną tablicę, aby określić czy odwołanie do pamięci było dozwolone.
5. Jeżeli niedozwolone to kończymy proces, jeżeli zabrakło strony w pamięci to sprowadzamy stronę.
6. Znajdujemy wolną ramkę w liście wolnych ramek.
7. Składamy zamówienie na przeczytanie z dysku potrzebnej strony do nowo przydzielonej ramki.
8. Modyfikujemy procesowi tablicę wewnętrzną i tablicę stron, odnotowując, że strona jest w pamięci
9. Wznawiamy wykonanie instrukcji. Proces może teraz sięgać do strony jakby była cały czas w pamięci.

ILUSTRACJA ALGORYTMU



1 – odniesienie
2 – pułapka
3 – strona jest w pamięci pomocniczej
4 – sprowadzenie brakującej strony
5 – modyfikacja tablicy stron
6 – wznowienie wykonania

SPRZĘT DO STRONICOWANIA NA ŻĄDANIE

Tablica stron: posiada informację o bitach poprawności i numerach ramek.

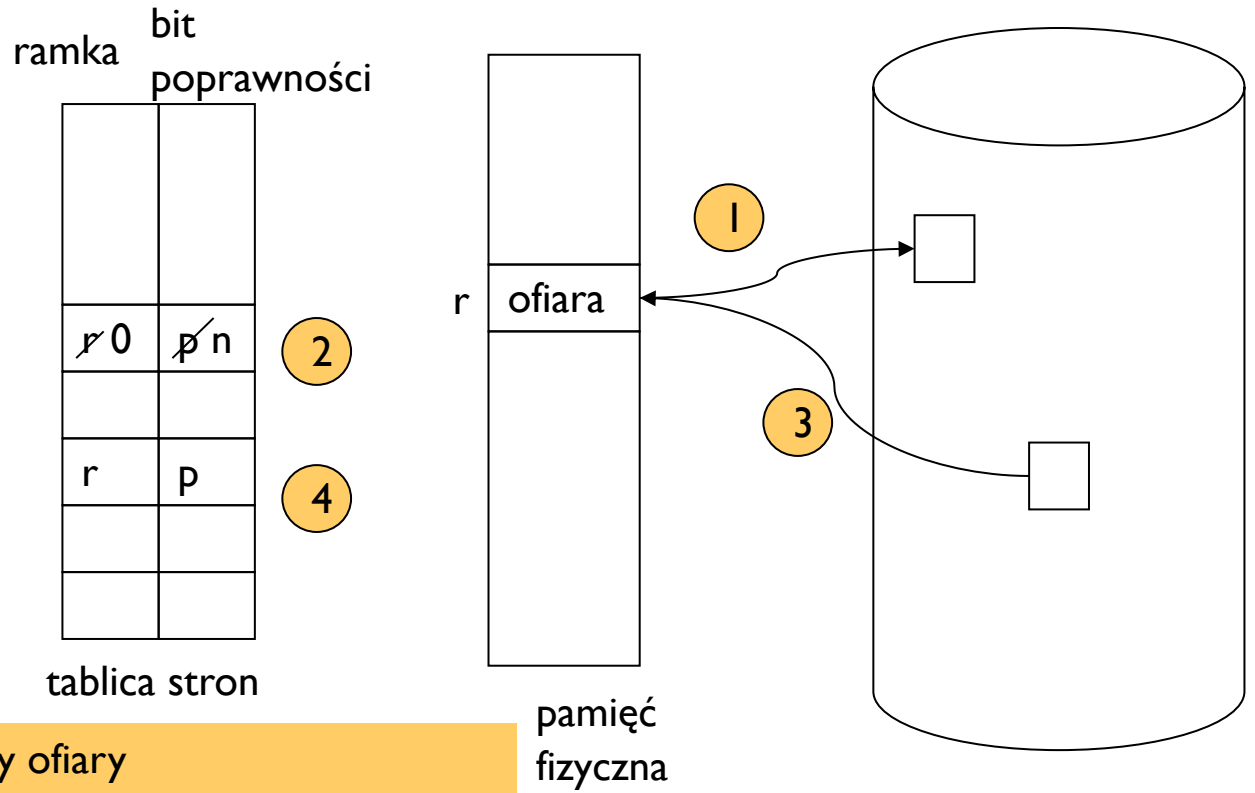
Pamięć pomocnicza: zazwyczaj dysk, część przeznaczona na wymianę nazywana przestrzenią wymiany (ang. swap).

Zapewnienie możliwości wznowienia procesu po błędzie adresu.

ZASTĘPOWANIE STRON

- Im mniej stron zużywa w danej chwili proces, tym więcej możemy ich wprowadzić do pamięci. Zwiększa się wówczas wieloprogramowość.
- Rozszerzanie wieloprogramowości może prowadzić do nadprzydziału pamięci. Każdy z wielu wykonywanych procesów może zażądać sprowadzenia z pamięci pomocniczej pewnej ilości stron, natomiast nie będzie na nie miejsca (brak wolnych ramek).
- W wypadku nadprzydzialu można:
  - zakończyć proces;
  - wymienić proces, tzn. zwolnić wszystkie ramki pewnego procesu (zmniejszyć wieloprogramowość), wykonać pozostałe procesy i wrócić do zwolnionego procesu.

ZASTĘPOWANIE STRON ILUSTRACJA



1 – wysłanie strony ofiary
2 – zmiana: na niepoprawny
3 – sprowadzenie potrzebnej strony
4 – modyfikacja tablicy stron dla nowej strony

OGÓLNY ALGORYTM OBSŁUGI BŁĘDÓW

1. Odnalezienie lokalizacji potrzebnej strony na dysku.
2. Odnalezienie wolnej ramki
    1. jeżeli istnieje wolna ramka to zostanie użyta;
    2. w przeciwnym razie należy uruchomić algorytm zastępowania stron w celu wytypowania ramki- ofiary;
    3. stronę ofiarę zapisujemy na dysku; zamiana tablic stron i ramek.
3. Do zwolnionej ramki wczytuje się potrzebną stronę i zmienia tablice stron i ramek.
4. Wznawia się działanie procesu.

BIT MODYFIKACJI

Specjalne wyposażenie strony lub ramki, by zapamiętać czy strona była modyfikowana od czasu odczytania z dysku.

Zatem jeżeli strona nie jest zmieniona nie trzeba jej wysyłać na dysk, bo już tam istnieje – zmniejsza to liczbę operacji dyskowych.

OCENA ALGORYTMU ZASTĘPOWANIA STRON / WYMIANY NA ŻĄDANIE

Algorytm ocenia się na podstawie wykonania go na pewnym ciągu odniesień do pamięci i zsumowaniu liczby błędów strony.

Aby określić liczbę błędów dla ciągu odniesień należy znać liczbę dostępnych ramek.

Oczekuje się by algorytm spełniał zależność: im więcej ramek tym mniej błędów.

Wszystkie rozważane na kolejnych slajdach przykłady dotyczą ciągu odniesień do stron:

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

do pamięci z trzema ramkami.

ALGORYTMY WYMIANY

MIN — zastępowana jest strona, która najdłużej nie będzie używana (optymalny w tej klasie)

FIFO (ang First In First Out) — zastępowana jest strona najstarsza (najwcześniej sprowadzona)

LIFO (ang Last In First Out) — zastępowana jest strona najmłodsza (najpóźniej sprowadzona)

LRU (ang. Least Recently Used) — zastępowana jest naj dawniej użyta strona (najdłużej nie używana)

LFU (ang. Least Frequently Used) — zastępowana jest najrzadziej używana strona

MFU (ang. Most Frequently Used) — zastępowana jest najczęściej używana strona

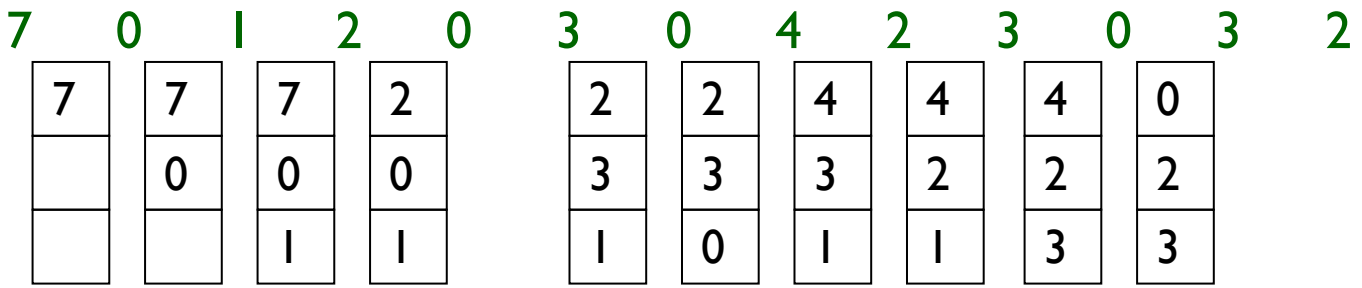
ALGORYTM FIFO

- Algorytm kojarzy z każdą strona jej czas wprowadzenia do pamięci.
- Jeżeli zastępujemy stronę wybieramy najstarszą.
- Można w kolejce FIFO przechowywać wszystkie strony, wówczas wychodzi strona z czoła kolejki. Nowa strona przypisywana jest na koniec kolejki.
- Algorytm łatwy do zrozumienia i zaprogramowania.
- Algorytm może powodować anomalię Belady'ego: tzn. pomimo wzrostu liczby ramek liczba błędów nie koniecznie musi się zmniejszać
- Przykład anomalii Belady'ego
  - W przypadku 3 ramek pierwsze 7 odniesień do stron kończy się błędem, ale kolejna 2 odniesienia przebiegają bez błędów, a później pojawiają się jeszcze 2 błędy. W sumie jest ich 9.
  - W przypadku 4 ramek początek jest bardziej optymistyczny, gdyż po 4 pierwszych odniesieniach z błędem, wypełnione ramki zdają się gwarantować stabilność przetwarzania. Jednak sprowadzenie strony nr 5 powoduje usunięcie strony numer 1, która sprowadzana jest jako następna, usuwając z kolei potrzebną później stronę nr 2 itd. Ostatecznie mamy 10 błędów strony pomimo zwiększenia liczby ramek. Paradoks ten, nazywany anomalią Belady'ego, dotyczy wybranych algorytmów, między innymi algorytmu FIFO. Anomalia nie ujawnia się być może zbyt często, ale może wzbudzać obiekcje co do efektów przydziału dodatkowych zasobów.

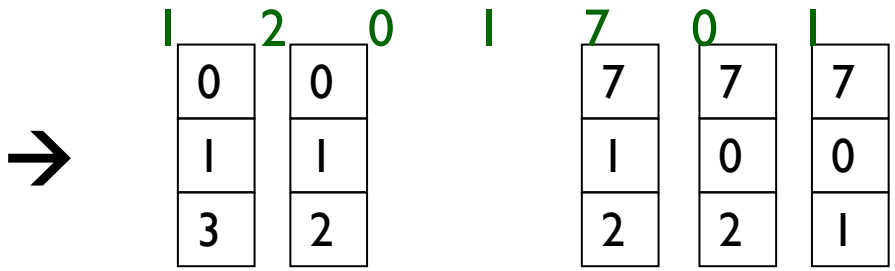
PRZYKŁAD FIFO

ciąg odniesień

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1



ramki stron



Łączna liczba błędów 15

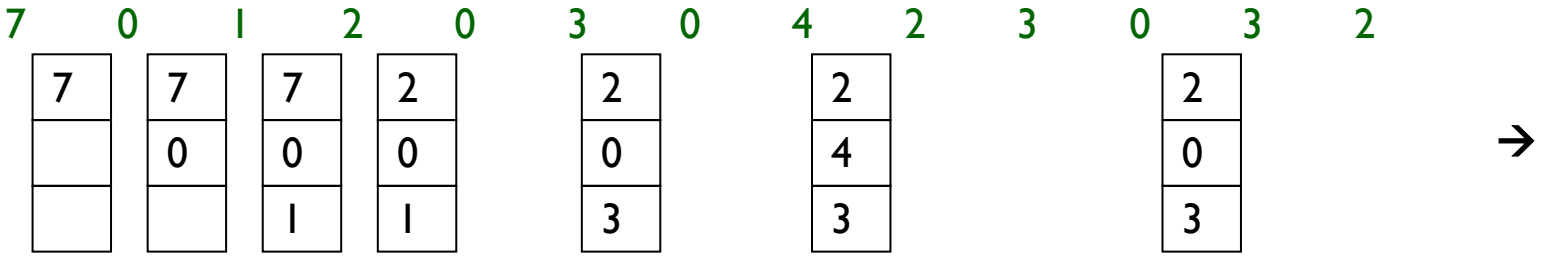
ALGORYTM OPTYMALNY MIN

- Ma najniższy współczynnik błędów.
- Nigdy nie powoduje anomalii Belady'ego.
- Działa według zasady: zastąp tą stronę, która najdłużej nie będzie używana.
- Dla rozpatrywanego przykładu jest dwukrotnie lepszy od FIFO (jeżeli pominiemy pierwsze trzy błędy, które wystąpią w każdym algorytmie).
- Jest trudny do realizacji, ponieważ trudno uzyskać wiedzę na temat przyszłego stanu procesu.

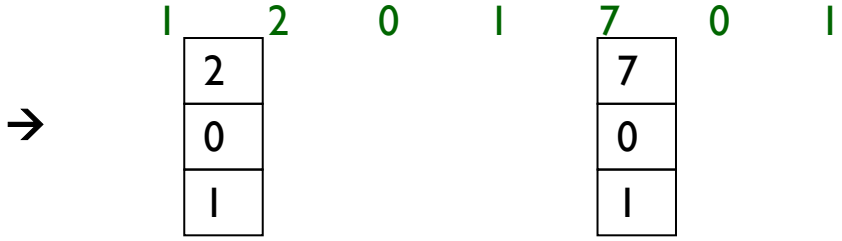
ALGORYTM OPTYMALNY MIN

ciąg odniesień

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1



ramki stron



Łączna liczba błędów: 9

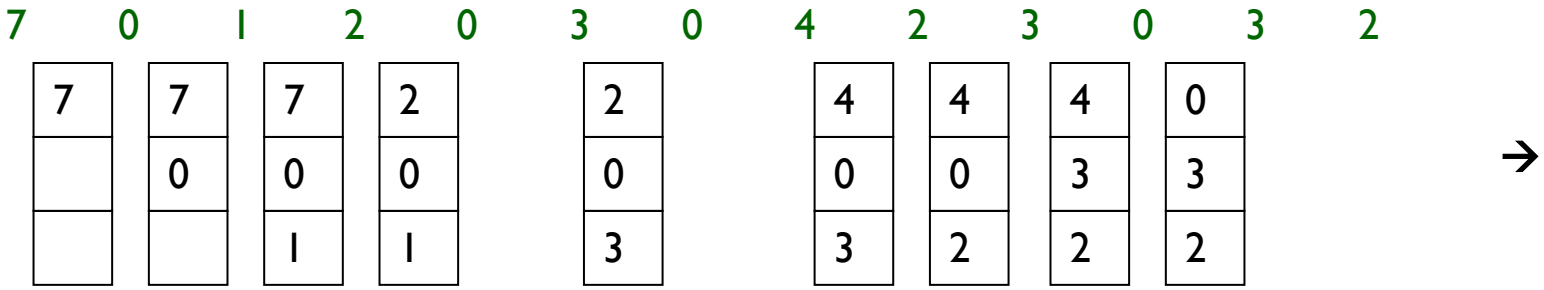
ALGORYTM LRU

- Najdawniej używane strony (ang. least recently used).
- Kojarzy ze stroną czas jej ostatniego użycia.
- Wyrzuca się stronę, która najdłużej nie była używana.
- Uważany za dość dobry.
- Wymaga zastosowania wyposażenia sprzętowego.
- Problem określania porządku ramek według czasu. Stosowane dwie implementacje:
  1. **Liczniki**: do tablicy stron dodaje się rejestr czasu użycia, a do procesora licznik. Licznik zwiększamy przy każdym odniesieniu do pamięci i kopiowany jest do rejestru czasu użycia. Trzeba przeglądać całą tablice by znaleźć element do usunięcia.
  2. **Stos**: przy każdym odniesieniu do strony wyjmuje się jej numer ze stosu i umieszcza na szczycie. Zatem na dnie stosu znajdują się elementy do usunięcia.

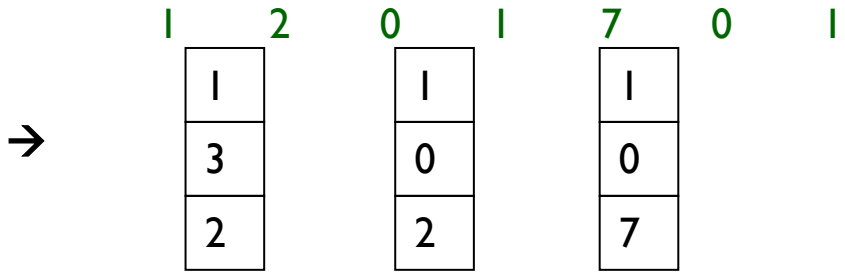
PRZYKŁAD LRU

ciąg odniesień

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1



ramki stron



Łączna liczba błędów 12

PRZYDZIAŁ RAMEK (PAMIĘĆ FIZYCZNA)

- Jest to problem rozdzielania pewnej stałej liczby ramek pomiędzy procesy.
- Proces użytkownika dostaje dowolną liczbę wolnych ramek.
- Na początku wszystkie ramki są wolne.
- W miarę jak proces wymaga kolejnych ramek są one przydzielane, aż pojawia się moment, że zabraknie ramek. Wówczas stosuje się algorytm zastępowania stron.
- Po zakończeniu procesu ramki przez niego zajmowane są zwalniane.

MINIMALNA LICZBA RAMEK

Istnieje minimalna liczba ramek, które mają być przydzielone, ponieważ im mniej ramek tym więcej błędów i spowolnienie wykonania procesu.

Liczba ta określona jest przez zbiór rozkazów architektury komputera i nie może być krótsza niż liczba stron dla pojedynczego rozkazu. (Takie ograniczenie musi być poczynione by rozkaz został wykonany).

ALGORYTMY PRZYDZIAŁU

Przydział równy: m ramek, n procesów, dla każdego procesu a=m/n ramek na proces.

Przydział proporcjonalny: pamięci odpowiednio do rozmiaru według wzoru:

$a=s_i/S*m$,

gdzie S-suma pamięci wirtualnej przydzielanej wszystkim procesom.(np.. 62 ramki dostępne, P1: 10 stron, P2: 127 stron, czyli: P1: 10/137*62=4 ramki, a P2: 127/137*62=57 ramek)

Przydział proporcjonalny według priorytetów procesu.

Przydział wysokopriorytetowy: zabiera się ramki procesom o niskim priorytecie.

SZAMOTANIE

- Proces nie ma wystarczającej liczby ramek.
- Liczba jego ramek zmniejszona jest do minimum.
- Wszystkie ramki są zajęte, a trzeba sprowadzić kolejną stronę, by proces mógł działać poprawnie. Podmienienie stron powoduje brak kolejnej niezbędnej do działania ramki.
- Proces będzie ciągle wykazywał brak strony, wymieniając jakąś stronę, po czym z powodu jej braku będzie ją ściągał ponownie.
- Taką dużą aktywność stronicowania określa się jako szamotanie.