AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Systemy operacyjne
Wykład 03N

Wersja 2024

dr inż. Marek Wilkus **http://home.agh.edu.pl/~mwilkus**
Wydział Inżynierii Metali i Informatyki Przemysłowej
AGH Kraków

Na podstawie programu opracowanego przez dr inż. Krzysztofa Wilka

AGH

Zarządzanie pamięcią

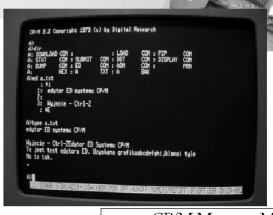
AGH

Zarządzanie pamięcią

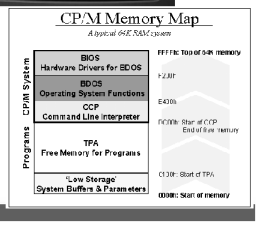
- Przed wykonaniem program musi być pobrany z dysku i załadowany do pamięci.
- Tam działa jako proces.
- Podczas wykonywania, proces pobiera rozkazy i dane z pamięci.
- Większość systemów pozwala procesowi użytkownika przebywać w dowolnej części pamięci fizycznej.
- System musi uwzględniać to, że program „nie wie" pod jakim adresem pamięci będzie umieszczony. System musi zawierać mechanizmy „tłumaczące" adresy w programie na rzeczywiste adresy w pamięci fizycznej.
- System musi też gospodarować wolną pamięcią, racjonalnie ładując kolejne programy w wolne miejsca pamięci.

AGH

Rys historyczny

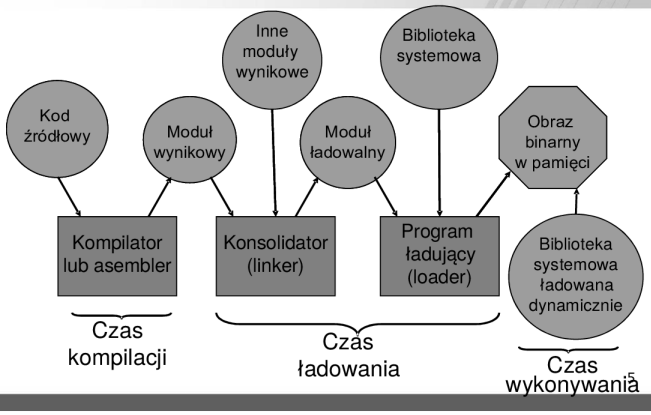


- 198x – CP/M – system z ładowaniem programów do pamięci.
  - TPA – wymienna strefa pamięci dla programów.
  - Ładowanie następnego programu powoduje nadpisanie pozostałości poprzedniego w TPA.
  - Iluzja wieloprogramowości – bankowanie regionów TPA.



AGH

Kod źródłowy → proces



AGH

Powiązanie programu z adresami w pamięci

- Może być dokonane w każdej z trzech faz:
  - Czas **kompilacji** - jeśli podczas kompilacji wiadomo pod jakim adresem pamięci program będzie przebywał, to tworzy się **kod bezwzględny**, np programy .com w DOS-ie, sterowniki w Windows. Aby zmienić położenie takiego programu w pamięci, trzeba go powtórnie przekompilować. Jeśli kod programu ma być **przemieszczalny**, to adresy muszą być określone w sposób względny, np w odległościach od początku modułu.
  - Czas **ładowania** - jeśli podczas kompilacji nie określono rzeczywistych adresów pamięci, to program ładujący wylicza je
    na podstawie adresów względnych.
  - Czas **wykonania** - jeśli proces może ulegać przemieszczeniu w pamięci podczas wykonywania, to muszą istnieć mechanizmy wyliczania w dowolnym momencie rzeczywistych adresów.

Dołączanie dynamiczne

- **Ładowanie dynamiczne** - podprogram nie jest wprowadzany do pamięci dopóty, dopóki nie zostanie wywołany. Do pamięci wprowadza się jedynie program główny, a potem sukcesywnie potrzebne moduły. Dzięki temu oszczędza się miejsce w pamięci nie ładując nie potrzebnie wielkich modułów, np. obsługi błędów.
- **Konsolidacja dynamiczna** - Dynamiczne dołączanie bibliotek zewnętrznych - bez tej właściwości wszystkie programy muszą mieć dołączone kopie bibliotek, w tym systemowych. Powoduje to marnotrawstwo dysku i pamięci. Biblioteki dynamicznie linkowane są sprowadzane do pamięci w momencie ich wywołania i tam mogą służyć nawet kilku programom. Dodatkowa zaleta - aktualizacja biblioteki nie wymaga zazwyczaj przekompilowania programu.

Nakładki

- Potrzebne gdy proces jest większy niż ilość dostępnej pamięci.
- Przykład: Dwuprzebiegowy assembler:
  Mamy do dyspozycji 150kB pamięci, elementy zadań mają rozmiary:
  - Kod przebiegu 1: 70kB
  - Kod przebiegu 2: 80kB
  - Tablice pomocnicze symboli: 20kB
  - Wspólne podprogramy: 30kB
- Razem: 200kB

Nakładki

- Załóżmy, że system nie umożliwia pamięci wirtualnej. Robimy dwie nakładki:
  - Tablica symboli, wspólne podprogramy, kod przebiegu 1, moduł obsługi nakładek (10kB): Razem 130kB.
  - Tablica symboli, wspólne podprogramy, kod przebiegu 2, moduł obsługi: Razem 140kB.
- Obydwie nakładki są poniżej 150kB.
  - Kody nakładek przechowywane są na dysku w postaci obrazów bezwzględnych pamięci i są ładowane przez moduł obsługi w zależności od potrzeb.
  - Program nakładkowy nie potrzebuje wsparcia ze strony systemu operacyjnego – wymaga starannego zaprogramowania programu obsługi.

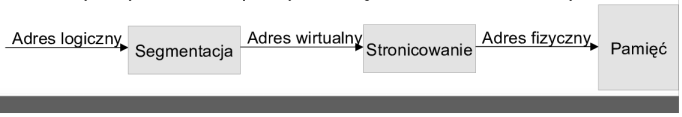
Logiczna i fizyczna przestrzeń adresowa

- **Adres fizyczny** -jest to adres oglądany przez jednostkę pamięci (umieszczony w jej rejestrze adresowym).
- **Adres logiczny** - jest to adres wygenerowany przez procesor.

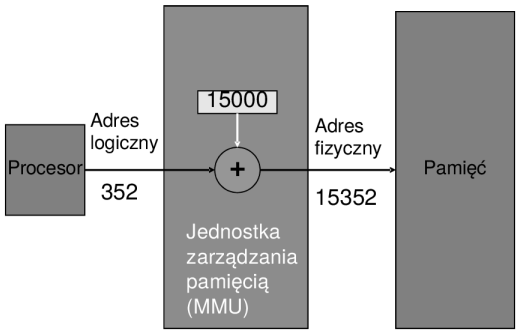
- Odwzorowanie adresów logicznych na fizyczne realizowane jest sprzętowo przez jednostkę zarządzania pamięcią (MMU).
- W MMU przeliczanie adresu odbywa się najczęściej poprzez dodanie do adresu z procesora wartości **rejestru przemieszczenia**.
- Program użytkownika działa wyłącznie na logicznych adresach.

W Intelu...

- Pomiędzy adresem logicznym a fizycznym występuje jeszcze **adres wirtualny.**
- **Adres wirtualny** jest tym, czym intuicyjnie wydaje się być w starszych platformach adres fizyczny - czyli przesunięciem od początku przestrzeni adresowej.
- Ze względu na przełączanie banków pamięci, „rozdymanie" przestrzeni adresowej przełączając całe fragmenty pamięci, PAE, i dodatki w stylu LIM EMS, adres fizyczny może nie pokrywać się z adresem wirtualnym.



Wyliczanie adresu fizycznego



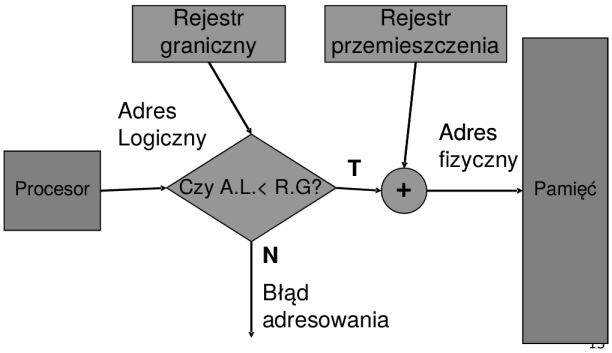
Wymiana

- Jest to tymczasowe odesłanie procesu do pamięci pomocniczej (na dysk) i przepisanie z powrotem w celu kontynuowania działania.
- Zazwyczaj po wymianie proces powraca na to samo miejsce w pamięci, chyba że są zapewnione mechanizmy przeliczania adresów.
- Warunkiem szybkiej wymiany jest używanie dysku o krótkim czasie dostępu i o pojemności zapewniającej pomieszczenie obrazów pamięci wszystkich użytkowników.
- W planowaniu priorytetowym stosuje się czasem wymianę poprzez **wytaczanie** procesu, gdy nadchodzi proces o wyższym priorytecie i **wtaczanie** z powrotem do pamięci, gdy procesy wysoko-priorytetowy skończył działanie.

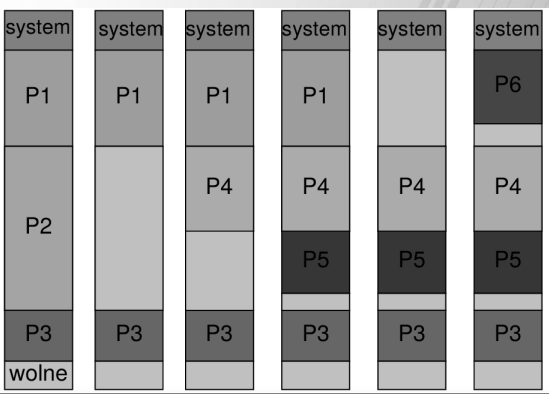
Przydział ciągły

- Pamięć operacyjna zajęta jest przez:
  - System operacyjny - umieszczony zazwyczaj w tej części pamięci, gdzie wektor przerwań - najczęściej pamięć "dolna".
  - Procesy użytkownika, umieszczane zazwyczaj w pamięci "górnej".
- W celu ochrony obszaru pamięci zajętego przez system, a także procesów użytkownika przed wzajemną ingerencją, wykorzystuje się **rejestr przemieszczenia** i **rejestr graniczny:**
  - Rejestr przemieszczenia - wartość najmniejszego adresu fizycznego dla danego procesu (offset),
  - Rejestr graniczny - maksymalny adres logiczny procesu.

Kontrola zakresu adresów



Przydział pamięci przy wielu procesach



Kryteria wyboru wolnego obszaru pamięci

Dziura - ciągły obszar niezajętej pamięci.

- **Pierwsze dopasowanie** - system przydziela pierwszą dziurę o wystarczającej wielkości. Szukanie rozpoczyna się od początku wykazu dziur lub od miejsca w którym zakończono ostatnie szukanie.
- **Najlepsze dopasowanie** - przydziela się najmniejszą z dostatecznie dużych dziur (najmniejsza pozostałość po przydziale).
- **Najgorsze dopasowanie** - przydziela się największą dziurę.
  - Czasami duża pozostałość po takim przydziale jest bardziej przydatna niż malutkie fragmenty po najlepszym dopasowaniu.
  - Ciekawe i oryginalne podejście do zagadnienia, ale praktyka wykazała, że dwie pozostałe metody są lepsze pod względem czasu działania i wykorzystania pamięci.

Fragmentacja

- **Fragmentacja zewnętrzna**- suma wolnych obszarów pamięci wystarcza na spełnienie zamówienia, ale nie tworzą one spójnego obszaru.
- **Fragmentacja wewnętrzna** - jeśli po przydzieleniu pamięci do procesu pozostałby wolny obszar wielkości kilku bajtów, to przydziela się go też do procesu, ale stanowi on „nieużytek"- nie będzie wykorzystany (ale zmniejszy się tablica „dziur").

- Fragmentację zewnętrzną można zmniejszyć poprzez takie upakowanie procesów, aby cała wolna pamięć znalazła się w jednym dużym bloku.
  Jest to możliwe tylko wtedy, gdy ustalanie adresów jest wykonywane dynamicznie podczas działania procesu.
  Przetasowań procesów nie można robić podczas operacji we/wy.

Stronicowanie

Pomaga w racjonalnym wykorzystaniu wolnych miejsc w pamięci.

- Pamięć fizyczną dzieli się na bloki o stałej długości (ramki) o długości 2^n (np. 512 B do 16 MB).
- Pamięć logiczna podzielona jest na strony o tym samym rozmiarze.
- Wolną pamięć obrazuje lista wolnych ramek.
- Proces o wielkości N stron jest ładowany w N ramek (niekoniecznie kolejnych).
- Tablica stron odwzorowuje adresy logiczne na fizyczne.
- Eliminuje się fragmentację zewnętrzną, ale występuje fragmentacja wewnętrzna (zaokrąglenie w górę wielkości procesu do wielokrotności rozmiaru ramki).

Stronicowanie

Każdy adres wygenerowany przez procesor dzieli się na dwie części: numer strony i odległość na stronie. Numer jest używany jako indeks w tablicy stron, która zawiera adresy bazowe wszystkich stron w pamięci operacyjnej.

Łącząc adres bazowy z odległością na stronie uzyskuje się fizyczny adres w pamięci.

Jeżeli rozmiar strony jest potęgą 2 oraz:

- rozmiar przestrzeni adresowej wynosi 2^m,
- rozmiar strony wynosi 2^n,

to m-n bardziej znaczących bitów adresu logicznego wskazuje nr strony ($=2^{(m-n)}$),

- n mniej znaczących bitów wskazuje odległość na stronie.

Tablica stron

- Jest przechowywana w pamięci operacyjnej. Jej położenie wskazuje rejestr bazowy tablicy stron.
- Rozmiar tablicy stron jest przechowywany w rejestrze długości tablicy stron. Określa on na największy dopuszczalny adres.
- Przy korzystaniu z tablicy stron, dostęp do pamięci wymaga **dwukrotnego** dostępu do pamięci - dlaczego?
- W celu przyspieszenia dostępu do pamięci stosuje się rozwiązanie sprzętowe - małą, szybką pamięć podręczną zwaną **rejestrami asocjacyjnymi** lub **buforami translacji adresów stron**. Bufory te zwierają 8 do 2048 pozycji.
  - We współczesnych komputerach zawarte w MMU, wcześniej – część cache.
- Jeśli dany numer strony nie znajduje się w buforach, to przeszukiwana jest cała tablica stron. Przy dobrze skonstruowanym algorytmie, w buforach translacji znajduje się 80 do 98 % potrzebnych numerów stron.

Efektywny czas dostępu

- Jest to średni czas dostępu do każdego adresu pamięci.
- Przykład:
  Przeglądnięcie rejestrów asocjacyjnych trwa 20ns
  Dostęp do pamięci trwa 100 ns
- Współczynnik trafień - procent stron znalezionych w rejestrach asocjacyjnych.
  Dostęp do stron „trafionych”=20+100= 120 ns
  Dostęp do stron nie występujących w rejestrach=20+100+100 =220 ns
- Dla współczynnika trafień = 80%:
  E.C.D.=0,8*120+0,2*220=140 ns
- Dla współczynnika trafień = 98%:
  E.C.D.=0.98*120+0.02*220=122 ns

Ochrona pamięci

- Bity ochrony - przypisane do każdej ramki. Można w ten sposób zaznaczyć strony tylko do czytania, do czytania i pisania i do wykonywania.
- Bit poprawności - jest ustawiony, jeśli dana strona jest w przestrzeni adresowej procesu (strona jest „legalna” dla procesu).
- Jeśli strona jest poza przestrzenią adresową procesu, ma ustawiony znacznik „nielegalna”.

Stronicowanie wielopoziomowe

- Współczesne systemy zezwalają na na stosowanie bardzo wielkich przestrzeni adresów logicznych (2^{32} do $>2^{64}$).
- W takim przypadku tablica stron może zawierać nawet milion wpisów. Jeśli każdy wpis to 4 B, rozmiar tablicy wyniesie 4MB, na każdy proces. Tablice takie mogą być większe niż same procesy!
- Celowy jest więc podział na mniejsze tablice.
- Przy 32-bitowej przestrzeni adresowej:
  20-bitowy adres strony i 12-bitową odległość na stronie
  można zastąpić przez:
  10-bitowy adres strony, 10-bitową odległość na tej zewnętrznej stronie i 12-bitową odległość wewnętrzną.

Stronicowanie wielopoziomowe

- Każdy poziom jest zapisywany jako oddzielna tablica, więc przekształcenie adresu logicznego w fizyczny może wymagać czterech dostępów do pamięci.
  Czas dostępu wynosi wtedy np 520 ns.
- Ale zastosowanie pamięci podręcznej bardzo go skraca.
- Dla współczynnika trafień wynoszącego 98%:
  Efektywny czas dostępu= 0,98*120+0,02*520=128 ns.

Odwrócona tablica stron

- Odwrócona tablica stron ma po jednej pozycji dla każdej ramki w pamięci fizycznej
- Każda pozycja zawiera numer procesu posiadającego ramkę oraz adres wirtualny strony przechowywanej w ramce rzeczywistej pamięci.
- W systemie istnieje tylko jedna tablica stron.
- Ogranicza to zajętość pamięci, ale zwiększa czas przeszukiwania (trzeba przeszukać całą tablicę)
- Stosowanie tablic haszowania ogranicza przeszukiwanie do co najwyżej kilku wpisów.

Strony dzielone

20 użytkowników korzysta równocześnie z edytora tekstu. W pamięci musi się znaleźć 20 bloków danych i 20 kopii kodu edytora.

- Jeśli kod programu nie modyfikuje sam siebie w czasie działania (jest wznawialny) to można zastosować mechanizm stron dzielonych.
- Wznawialny kod programu jest widziany przez wszystkie procesy pod tą samą lokacją.
- Każdy proces dysponuje więc własnym obszarem danych i jednym wspólnym kodem programu.
- Mechanizm ten może być też stosowany przy innych intensywnie używanych programach (kompilatory, systemy okien, bazy danych).
- W systemach z odwróconą tablicą stron mechanizm ten napotyka na trudności - dlaczego?.

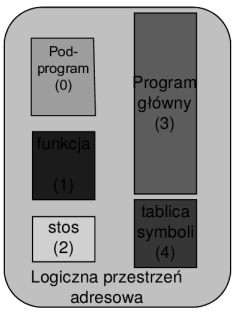
Segmentacja

- Jest to mechanizm naśladujący postrzeganie pamięci tak jak użytkownik.
- Przestrzeń adresów logicznych jest zbiorem segmentów. Każdy segment ma nazwę i długość. Użytkownik określa więc każdy adres poprzez nazwę segmentu i odległość.
- Kompilator automatycznie tworzy segmenty tworzące program wynikowy. Najczęściej oddzielnymi segmentami są:
  - program główny,
  - procedury,
  - funkcje,
  - zmienne lokalne,
  - zmienne globalne,
  - stos wywołań procedur (parametry i adresy powrotu)
  - bloki wspólne (common)...

Segmentacja

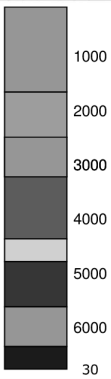
- Adres logiczny składa się z dwóch części:
  numer segmentu, odległość w segmencie
- Tablica segmentów zawiera pary danych:
  - baza (fizyczny adres początku segmentu w pamięci)
  - granica (długość segmentu)
- Rejestr bazowy tablicy segmentów - adres tablicy segmentów w pamięci.
- Ponieważ programy mogą mieć bardzo różniącą się liczbę segmentów, stosuje się też rejestr długości tablicy segmentów, który służy do sprawdzenia czy podany numer segmentu jest poprawny (< RDTS).

Przykład - segmentacja



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

Tablica segmentów

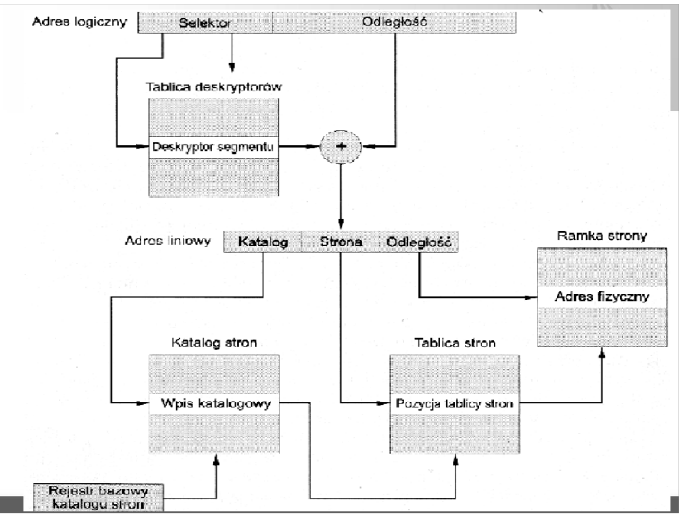


Segmenty dzielone

- Segmentacja ułatwia wspólne użytkowanie kodu programu, podprogramów lub niektórych danych.
- W tablicach segmentów wszystkich współużytkujących procesów obszary (segmenty) współdzielone mają takie same wpisy bazy i granicy (wskazują więc na ten sam obszar pamięci).
- We wszystkich procesach segmenty dzielone muszą mieć ten sam numer - dlaczego?
- Alokacja segmentów w pamięci odbywa się metodą pierwszego lub najlepszego dopasowania (segmenty mają różne długości).
- Ponieważ segmentacja jest algorytmem dynamicznego przemieszczania, można przesuwać segmenty w celu lepszego
  upakowania.

Segmentacja ze stronicowaniem

- W systemie MULTICS - pozycja w tablicy segmentów nie wskazuje adresu bazowego, ale adres tablicy stron dla tego segmentu.
- Systemy oparte na Motoroli 68000 - prosta przestrzeń adresowa
- Systemy oparte na Intel (>=80386) - segmentacja ze stronicowaniem z dwupoziomowym schematem stronicowania
  - maksymalna liczba segmentów w procesie: 16K
  - każdy segment mniejszy niż 4 GB
  - rozmiar strony 4 kB
  - przestrzeń adresowa ma dwie strefy po co najwyżej 8K segmentów:
    a) prywatne segmenty procesów przechowywane są w tablicy lokalnych deskryptorów,
    b) wspólne segmenty procesów przechowywane są w globalnej tablicy deskryptorów



Segmentacja ze stronicowaniem (Intel)

- Selektor jest 16-bitową liczbą:
  - 13 b - numer segmentu
  - 1 b - czy segment jest w lokalnej czy globalnej tablicy deskryptorów,
  - 2 b – ochrona.
- Każdy adres logiczny jest parą: selektor, odległość.
- Procesor ma 6 rejestrów segmentów (do adresowania 6 segmentów) oraz 6 rejestrów mikroprogramowych do przechowywania pozycji z lokalnej i globalnej tablicy deskryptorów,
- Sprawdzanie adresu: rejestr wyboru wskazuje na pozycję w lokalnej lub globalnej tablicy; na podstawie adresu początku segmentu i jego długości tworzy się adres liniowy (sprawdzenie poprawności); jeśli poprawny, to do bazy dodaje się odległość.

Stronicowanie dwupoziomowe (Intel)

- Liniowy adres jest 32-bitową liczbą:
  - 20 najstarszych bitów – numer strony,
  - 10b – wskaźnik do katalogu stron,
  - 10b – wskaźnik do tablicy stron
- 12 najmłodszych bitów: Odległość na stronie.
- W x86_64 adresy są odpowiednio rozszerzone.

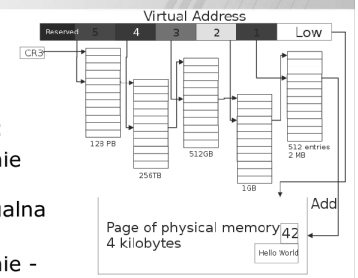
PAE - rozszerzenie adresu fizycznego

- 32-bitowe procesory mogą obsłużyć przestrzeń adresową:

  2^32 = 4 294 967 296 bajtów = 4GB

  Jak obsłużyć więcej?
- Podwajamy rozmiar wpisu w tablicy stron.
- Wskutek tego pamięci może być więcej, ale ze względu na to, że linii adresowych wciąż jest na 32 bity, **każda aplikacja może zaadresować nie więcej niż 4GB.** MMU zajmuje się tłumaczeniem przestrzeni adresowych danego programu na adresy ramek w pamięci.
- Czyli nadal widać maksimum 4GB (nie powiększymy ilości linii adresowych), ale mogą być one inne dla różnych programów.
- Rozwiązanie używane w 32-bitowych systemach Linux/Unix oraz Windows nowszych od XP.

Intel 64-bit



- Architektura przyjęła się w typowych komputerach osobistych.
- Dwa rodzaje adresowania:
  - 4-poziomowe stronicowanie na 64-bitowych adresach. Maksymalna pamięć wirtualna to 256TB.
  - 5-poziomowe stronicowanie - pamięć wirtualna do 128PB, ale kosztem dłuższego czasu dostępu (dłuższe szukanie właściwej strony).

Pamięć wirtualna

Pamięć wirtualna

- Pytanie: Czy proces rezerwuje pamięć i gospodaruje nią w sposób oszczędny?
  - Procesy często zawierają ogromne fragmenty kodu obsługujące sytuacje wyjątkowe.
  - Zadeklarowane tablice lub rozmiary list mają zwykle nadmiar przydzielonej pamięci.
  - Niektóre możliwości programu są niezwykle rzadko wykorzystywane.
- A gdyby tak w pamięci przebywała tylko ta część programu, która jest aktualnie wykonywana?

Pamięć wirtualna: zalety

- Program nie jest ograniczony wielkością pamięci fizycznej - można pisać ogromne programy bez specjalnych „sztuczek" programistycznych.
- Każdy program zajmuje w pamięci mniej miejsca niż program „kompletny". Można więc w tym samym czasie wykonywać więcej zadań, polepszając wykorzystanie procesora.
- Maleje liczba operacji wejścia-wyjścia koniecznych do załadowania programów do pamięci oraz do realizacji wymiany - programy powinny więc wykonywać się szybciej.
- Nie ma konieczności robienia nakładek przy małej pamięci operacyjnej.

Pamięć wirtualna

- Pamięć wirtualna odseparowuje pamięć logiczną od jej fizycznej realizacji. Można ją zaimplementować jako:
  a) stronicowanie na żądanie
  b) segmentację na żądanie
- Procesy przebywają w pamięci pomocniczej (na dysku). Dla wykonania sprowadza się proces do pamięci, ale nie cały, tylko te strony, które są potrzebne (leniwa wymiana). Typowanie (zgadywanie) potrzebnych stron odbywa się podczas poprzedniego pobytu procesu w pamięci.
- Jeśli proces odwołuje się do strony, której nie ma w pamięci, to:
  - System sprawdza, czy odwołanie do pamięci było dozwolone czy nie (bit poprawności)
  - Jeśli było poprawne, sprowadza stroną do pamięci, modyfikuje tablicę stron i wznawia działanie procesu.

Stronicowanie na żądanie a wymiana

- Proces jest ciągiem stron,
- **Wymiana** dotyczy całego procesu (wszystkich jego stron),
- **Procedura stronicująca** dotyczy poszczególnych stron procesu,
- Procedura stronicująca zgaduje jakie strony będą w użyciu i tylko te ładuje do pamięci. Nigdy nie dokonuje się wymiana całego procesu.

Stronicowanie na żądanie a spowolnienie systemu

p - prawdopodobieństwo braku strony

ECD - efektywny czas dostępu:

ECD=(1-p)*cd+p*cos

cd - czas dostępu do pamięci (10 do 200 ns)

cos - czas obsługi strony:

- obsługa przerwania „brak strony" (1 ÷ 100 μs)
- czytanie strony (duuużo μs)
- wznowienie procesu (1 ÷ 100 μs)

- Czynności obsługi braku strony:
  - Przejście do systemu operacyjnego,
  - Przechowanie kontekstu,...

Obsługa braku strony (c.d.)

- Określenie, że przerwanie to „brak strony",
- Sprawdzenie poprawności odwołania do strony i jej położenia\ na dysku,
- Czytanie z dysku do niezajętej ramki:
  - opóźnienie (~8 ms)
  - szukanie sektora (~15 ms)
  - transfer danych (~1 ms)
- Zmiana przydziału procesora do innego procesu
- Przerwanie od dysku po skończonym transferze
- Przełączenie kontekstu
- Skorygowanie tablicy stron
- Czekanie na przydział procesora i wznowienie procesu

Razem: ok. 25 ms

Obsługa braku strony

25 ms = 25 000 μs = 25 000 000 ns

Efektywny czas dostępu:

ECD=(1-p)*cd+p*cos

ECD=(1-p)*100+p*25000000=

100-100*p+25000000*p=

100+24 000 900*p [ns]

- Efektywny czas dostępu jest więc proporcjonalny do prawdopodobieństwa nie znalezienia strony w pamięci.

- Przy prawdopodobieństwie p = 1%
  ECD wyniesie: **240 109** ns

(czyli 2 400 razy więcej niż dostęp bezpośredni do pamięci).

Zastępowanie stron

- Założenie, że tylko część stron każdego procesu jest potrzebna w

  pamięci może doprowadzić do nadprzydziału (nadmiar procesów w pamięci i absolutny brak wolnych ramek).
- Aby nie blokować procesu potrzebującego kolejnej ramki, stosuje

  się **zastępowanie stron**.
  - uruchamia się algorytm typowania ramki-ofiary,
  - stronę-ofiarę **zapisuje** się na dysku,
  - aktualizuje się tablicę wolnych ramek,
  - **wczytuje** się potrzebną stronę do zwolnionej ramki,
  - aktualizuje się tablicę stron
  - wznawia się działanie procesu

- **Dwukrotne korzystanie z dysku bardzo wydłuża czas obsługi braku strony!**

Zastępowanie stron

- **Bit modyfikacji** (dirty bit, zabrudzenia) – jest ustawiany, jeżeli na stronie zapisano choćby jeden bit. Jeśli nic nie zmodyfikowano, nie trzeba takiej strony zrzucać na dysk.

- Zastępowanie stron jest podstawą stronicowania na żądanie.
- Praktycznie każdy proces wykonuje się z użyciem mniejszej ilości ramek niż by wynikało z wielkości procesu.
- Mechanizm działa efektywnie przy dobrze opracowanych algorytmach przydziału ramek i algorytmach zastępowania stron.

Algorytm zastępowania stron

- Algorytmy optymalizowane pod kątem minimalizacji częstości braku stron.
- Algorytmy ocenia się na podstawie ich wykonania dla pewnego ciągu odniesień (odwołań) do pamięci i zsumowanie liczby braku stron w tym ciągu.

- Algorytm FIFO (first in, first out)
  - O każdej ze stron zapamiętuje się informację, kiedy ona została sprowadzona do pamięci.
  - Zastępuje się „najstarszą" stronę.

- Przykład:
- Dla ciągu odniesień do stron: 1,2,3,4,1,2,5,1,2,3,4,5

...1,2,3,4,1,2,5,1,2,3,4,5

- Dla 3 dostępnych ramek dla procesu:

3 ramki

1 4 5
2 1 3 (9 braków stron)
3 2 4

- Dla 4 dostępnych ramek dla procesu:

4 ramki

1 5 4
2 1 5 (10 braków stron)
3 2
4 3

Anomalia Belady'ego ^^^^^^^^^^

Zwiększenie liczby ramek → zwiększenie chybień stronicowania...

Algorytm optymalny

- Zastąp tę stronę, która najdłużej nie będzie używana.

Dla ciągu odniesień:

```
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7 7 7 2   2   2     2     2       7
- 0 0 0   0   4     0     0       0
- - 1 1   3   3     3     1       1
```

9 braków stron, nie ma anomalii Bleady'ego
Algorytm optymalny jest trudny w realizacji, ponieważ wymaga wiedzy o przyszłym ciągu odniesień.
Jest on używany głównie do teoretycznych studiów porównawczych (o ile % dany algorytm jest gorszy od opt.)

Inne algorytmy

- Algorytm LRU (Latest Recently Used) - zastępowanie stron, które były najdawniej używane. Typowanie najstarszych poprzez:
  - **licznik** (w tablicy stron jest rejestr czasu ostatniego używania strony),
  - **stos** (przy każdym odwołaniu do strony, jej numer jest wyjmowany i umieszczany na szczycie stosu.
- Algorytmy bazujące na metodzie LRU
  - **z bitami odniesienia** (po odwołaniu do strony, znacznik przyjmuje wartość 1),
  - **dodatkowe bity odwołań** (co stały czas ustawianie kolejnych bitów rotacyjnie),
  - **druga szansa** (jeśli bit odniesienia=1 to zeruje się go, zmienia czas na bieżący i przegląda kolejne strony -FIFO. Jeśli bit=0 to się stronę wymienia).

Inne algorytmy

- Ulepszony algorytm drugiej szansy
  wykorzystuje się dwa bity: bit odniesienia i bit modyfikacji, jako parę. Powstają cztery klasy stron:
  - (0,0) - nie używana ostatnio i nie zmieniana (najlepsza ofiara)
  - (0,1) - nie używana ostatnio, ale zmieniana (gorsza, bo trzeba ją zapisać)
  - (1,0) - Używana ostatnio, ale nie zmieniana (prawdopodobnie za chwilę będzie znów używana)
  - (1,1) - używana ostatnio i zmieniona (chyba będzie używana niedługo, a poza tym trzeba ją zapisać – najgorsza kandydatka na ofiarę).
- Zastępujemy pierwszą napotkaną stronę z najniższej klasy.

Inne algorytmy

- **Algorytmy zliczające**
  Wprowadzamy licznik odwołań do strony
  - LFU (least frequently used) -wyrzuć najrzadziej używaną
  - MFU (most frequently used) - bo najrzadziej używana jest chyba niedawno wprowadzona do pamięci i będzie niedługo w użyciu
- Obydwa te algorytmy niezbyt dobrze przybliżają optimum.
- **Algorytmy buforowania stron**
- Zanim się usunie ofiarę, wczytuje się potrzebną stronę do wolnej ramki.
- Zaleta: można wcześniej uruchomić proces, zanim strona-ofiara zostanie zapisana na dysku. Zapis robi się w wolnej chwili.
- Po zapisie opróżnioną ramkę dopisuje się do listy wolnych.

Przydział ramek

- Każdemu procesowi system musi przydzielić pulę ramek pamięci potrzebnych do jego pracy.
  Trzy najpopularniejsze algorytmy przydziału:
- **równy** (każdy proces dostaje tyle samo ramek) np 50 ramek i 5 procesów, to każdy dostaje po 10
- **proporcjonalny** (liczba przydzielonych ramek proporcjonalna do wielkości procesu)
- **priorytetowy** (liczba przydzielonych ramek zależy tylko od priorytetu procesu, albo od priorytetu i wielkości).

Globalne i lokalne zastępowanie ramek

- **Zastępowanie lokalne** - proces może zastępować ramki wyłącznie z puli tych, które dostał przy przydziale.
- **Zastępowanie globalne** - Proces może korzystać z puli wszystkich wolnych ramek, nawet jeżeli są wstępnie przydzielone innemu procesowi. Proces może zabrać ramkę drugiemu.

- **Praktyka wykazała, że zastępowanie globalne daje lepszą przepustowość systemu.**

Strona zerowa

- We współczesnych systemach proces może zarezerwować więcej pamięci niż zamawia w momencie jego tworzenia…
  - ...wczytanie pliku do tablicy/zmiennej,
  - ...przechwytywanie danych,
  - ...obliczenia wymagające zmiennej długości buforów.

- Zamówiona pamięć **nie jest przypisywana do procesu od razu.** Wskaźnik na nową stronę wskazuje na stronę zerową – specjalną stronę wypełnioną bajtami \0.
- Dzięki temu do odczytu program może zamówić nawet więcej pamięci niż jest w systemie i w urządzeniu stronicowania. Dopiero zapis wyzwala poszukiwanie nowej ramki w pamięci i przełączenie wskaźnika na stronę odpowiadającą tej ramce.

Szamotanie

- Jeśli proces nie ma wystarczającej liczby ramek, to w pewnym momencie musi wymienić stronę, która będzie potrzebna w niedługim czasie. W konsekwencji, kolejne braki stron będą występowały bardzo często. Taki proces „szamoce się", spędzając więcej czasu na stronicowaniu niż na wykonaniu.
- Zmniejsza się wykorzystanie procesora.

Scenariusz szamotania
- Jeżeli wykorzystanie jednostki centralnej jest za małe, planista przydziału procesora **zwiększa wieloprogramowość**.
- Nowy proces zabiera ramki pozostałym procesom.
- Zaczyna brakować ramek.
- Strony ustawiają się w kolejce do urządzenia stronicującego, a jednocześnie zmniejsza się kolejka procesów gotowych...

Szamotanie

- Wykorzystanie procesora maleje, bo procesy stoją w kolejce,
- System zwiększa wieloprogramowość,
- Sytuacja staje się tragiczna - żaden proces nie pracuje, tylko wszystkie stronicują.

- Jak powstrzymać szamotanie lub do niego nie dopuścić?
  - Stosować metodę lokalnego lub priorytetowego przydziału stron,
  - Stosować mechanizmy zmniejszenia wieloprogramowości przy szamotaniu,
  - Zapewnić dostawę wystarczającej ilości ramek.
- **Mierzenie częstości braków stron** - pomiar szamotania.
- Jeśli proces przekracza górną granicę częstości - dostaje wolną ramkę.
- Jeśli przekracza dolną granicę - odbiera mu się ramkę.

Problem operacji I/O

- Proces zamawia operację we-wy i ustawia się w kolejce do urządzenia,
- Procesor przydziela się innym procesom,
- Występują braki stron,
- W wyniku algorytmu globalnego zastępowania stron zostaje wymieniona strona zawierająca bufor we-wy procesu czekającego na operację we-wy,
- Zaczyna się operacja we-wy i nadpisuje dane innego procesu!

- Zapobieganie:
  - Zakaz wykonywania operacji we-wy wprost do pamięci użytkownika - ale kopiowanie czasochłonne,
  - Blokowanie stron w pamięci - strony czekające lub realizujące operację we-wy nie mogą być zastępowane.

Problemy przy wirtualizacji

- Współczesne systemy dysponują różnymi aspektami wirtualizacji - od uruchamiania programów w osobnych enklawach, do pełnej wirtualizacji kilku kopii systemów.

- Każda z tych opcji spełnia jakąś funkcję i umożliwia osiągnięcie różnych celów:
  - Separowanie programów,
  - Dzielenie mocnych serwerów na słabsze wydajnościowo jednostki,
  - Uruchamianie programów kompatybilnych z innymi systemami lub wersjami systemów,
  - Zapewnienie bezpieczeństwa ...

Problemy przy wirtualizacji

- Czyli podczas wirtualizacji:
  - Strony współdzielone:
    - + mniejsze użycie RAM,
    - - bezpieczeństwo
  - Stronicowanie do wspólnej przestrzeni:
    - + Wydajniejsza praca
    - - Możliwy exploit przy skompromitowaniu systemu.
  - Zabezpieczenie przed szamotaniem na poziomie nadzorcy:
    - + Pewność
    - - Łatwa droga do „przeskoczenia" z jednej maszyny do drugiej
  - Zabezpieczenie przed szamotaniem na poziomie poszczególnych maszyn wirtualnych:
    - + Bezpieczeństwo
    - - Nieoptymalne gospodarowanie zasobami.

Implementacja

- Windows:
  - Plik wymiany/stronicowania,
  - Plik na osobnej partycji.
- Unix:
  - Plik stronicowania,
  - Partycja stronicowania,
  - ZRAM.

Dziękuję za uwagę