Wykład 8 **8.04.2002**

Pamięć wirtualna

A. Motywacje
B. Stronicowanie na żądanie, sprawność
C. Zastępowanie stron
D. Algorytmy zastępowania stron
1) FIFO (anomalia Belady'ego)
2) Optymalny
3) LRU (least recently used; liczniki, stos)
4) Algorytmy przybliżające do LRU
a) Dodatkowe bity odniesienia
b) Algorytm drugiej szansy (na pozostanie w pamięci)
c) Algorytm LFU (least frequently used)
d) Algorytm MFU (most frequently used)
E. Przydział ramek
a) Minimalna liczba ramek
b) Algorytmy przydziału (równy, proporcjonalny)
F. Szamotanie
a) Przyczyny szamotania
b) Model zbioru roboczego
c) Częstość błędów braku strony

A) Pamięć wirtualna jest techniką, która umożliwia wykonywanie programów (procesów), pomimo, że nie są one w całości przechowywane w RAM.
Wynika stąd, że programy mogą być większe niż pamięć fizyczna
Pamięć wirtualna pozwala utworzyć abstrakcyjną pamięć główną w postaci wielkiej tablicy do magazynowania informacji.

- W czasie badania rzeczywistych programów zauważono, że nie jest konieczne by cały program był w pamięci RAM bo na przykład:

- programy często zawierają fragmenty przeznaczone do obsługi sytuacji wyjątkowych; mogą zatem (te fragmenty) nie być nigdy potrzebne
- tablice mają często zadeklarowany znacznie większy rozmiar niż naprawdę potrzebują
- pewne możliwości oferowane przez program mogą być żadko używane.

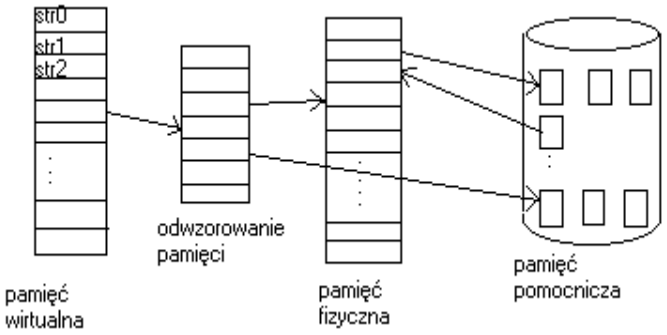
- Możliwości wykonywania programu, który tylko w części znajduje się w pamięci daje wiele korzyści:

- program przestaje być ograniczony wielkością dostępnej pamięci fizycznej
- można w tym samym czasie wykonywać więcej programów
- maleje liczba operacji we-wy związanych z procesem wymiany całych procesów; zatem programy użytkownika wykonują się szybciej

Pamięć wirtualna jest najczęściej implementowana w postaci stronicowania na żądanie.

B) Procesy rezydują w pamięci pomocniczej (zwykle dysk twardy). Proces, który należy wykonać zostaje wprowadzony do pamięci głównej. Stosuje się tu tzw. „leniwy" sposób ładowania procesu. Polega to na tym, że nigdy nie dokonuje się wymiany strony w pamięci, jeśli to nie jest konieczne.
Zamiast terminu wymiana stron będziemy używać terminu zmienianie stron.

Pamięć wirtualna może być znacznie większa niż pamięć fizyczna: (rys.)

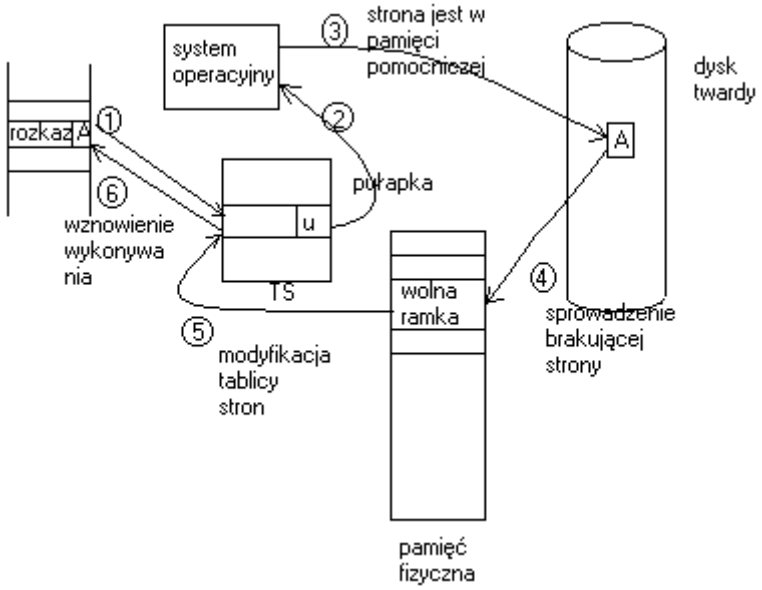


Gdy proces ma zostać wprowadzony do pamięci, wtedy program zmieniania stron zgaduje, które strony będą w użyciu. Zamiast dokonywania wymiany całego procesu, program zmieniania stron sprowadza do pamięci tylko strony niezbędne. Wymaga to wykorzystanie dodatkowych środków sprzętowych. Każda pozycja tablicy stron jest uzupełniona o dodatkowy bit poprawności odniesienia. Gdyby bit ten ma wartość poprawny (1) to dana strona jest w RAM. W przeciwnym razie (0) strona jest na dysku.

Rys1

Oznaczenie strony jako niepoprawnej nie wywołuje żadnych skutków, jeśli proces nigdy się do niej nie odwoła. Zatem dopóki proces działa (odwołuje się) na stronach pozostających w pamięci, dopóty jego wykonanie przebiega normalnie.
Jeśli jednak proces odwoła się do strony nieobecnej w RAM to powstaje błąd strony (odwołanie do strony, której bit poprawności w TS jest niepoprawny).
Następuje wtedy:

1. Sprawdzenie czy odwołanie do pamięci było dozwolone.
2. Jeśli niedozwolone to koniec procesu, jeśli brak strony to dalej.
3. Znajdowanie wolnej ramki.
4. Zamówienie do dysku na przeczytanie strony (może byś kolejka zgłoszeń z innych procesów).
5. Po zakończeniu czytania modyfikacja tablicy stron.
6. Wznowienie wykonanie instrukcji, która spowodowała błąd strony.



Czyste stronicowanie na żądanie- polega na podjęciu realizacji procesu bez żadnej strony w pamięci. Powstaje natychmiast błąd strony i jej sprowadzenie.

rys2

Sprawność stronicowania na żądanie:
Niech p oznacza prawdopodobieństwo błędu strony $p \in [0,1]$
Efektywny czas dostępu **EDC=(1-p)*MA+p*CBS** MA-100ns
gdzie: CBS-rzędu 20ms

MA-czas dostępu do pamięci
CBS-czas wymagany do obsługi błędu strony
Ćwiczenie: Obliczyć ECD jeśli 1 na 1000 odwołań do pamięci powoduje błąd strony.

$ECD=(1-p)*100_{ns} + p*25_{ms} = (1-p)*100 + p*25\,000\,000 = 100-100p+25\,000\,000p =$
$=100+24\,999\,900p$ (w nanosekundach)

jeśli p=0,001, to mamy ECD=100+24 999,9= 24 999,9
+ 100,0
25
099,9 ns

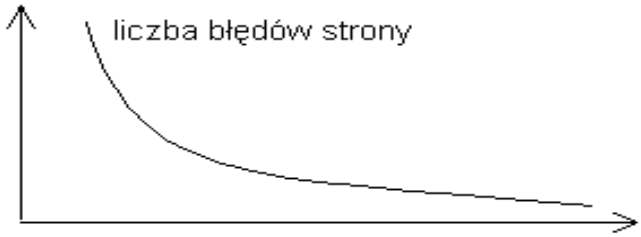
Czyli około 25μs. Jest to 250 razy wolniej!!!
Obliczyć ile co najwyżej może wynosić p aby ECD przekraczało MA co najwyżej o 10%.

C) Załóżmy, że mamy 40 ramek pamięci i realizujemy procesy, z których każdy wymaga 10 ramek. Możemy realizować 4 takie procesy. Jeśli jednak procesy będą faktycznie używały tylko po 5 ramek to moglibyśmy teoretycznie realizować ich 8.
Jeśli teraz, któryś z procesów zażąda następnej ramki, to ze względu na to, że nie ma wolnych ramek, trzeba będzie albo wymienić proces albo zastąpić którąś ze stron pamięci stroną żądaną.
Zasada zastępowania stron jest następująca:
Jeśli wszystkie ramki są zajęte, znajduje się taka, która nie jest bieżąco używana i zwalnia się ją. Ramkę można zwolnić przez zapisanie jej zawartości na dysku i modyfikację tablicy stron (niepoprawność). Zwolnioną ramkę można użyć do wprowadzenia strony żądanej.
Gdy nie ma wolnych ramek, wówczas potrzebne jest dwukrotne przesyłanie stron (jedno na dysk i jedno z dysku).
Można wprowadzić usprawnienie w postaci (sprzętowego) bitu modyfikacji. Każda strona lub ramka może być wyposażona w taki bit. Bit ten jest ustawiany sprzętowo jako równy 1 przy zapisie dowolnego słowa lub bajtu na stronie i wskazuje przy tym samym, że strona uległa modyfikacji. Przy wyborze strony do zastąpienia sprawdza się jej bit modyfikacji. Jeśli bit nie jest ustawiony (0), nie trzeba strony zapisywać na dysku.
Zastępowanie stron jest podstawą stronicowania na żądanie. Aby zrealizować stronicowanie na żądanie należy rozwiązać dwa główne problemy:
- opracować algorytm przydziału ramek (E) (ile ramek przydzielać procesowi)
- opracować algorytm zastępowania stron (D) (jak zastępować strony)

D) Prawie każdy system operacyjny ma własny unikatowy algorytm zastępowania stron. Zależy nam przy tym zawsze na tym by zminimalizować częstość występowania błędów stron.
Algorytmy ocenia się na podstawie wykonania go na pewnym ciągu odniesień do pamięci i zsumowaniu liczby błędów stron. Ów ciąg odniesień można wygenerować lub uzyskać z rzeczywistego działającego systemu operacyjnego.
Na przykład w wyniku śledzenia pewnego procesu zaobserwowano następujący ciąg adresów:
0100, 0432, 0101, 0612, 0102, 0103, 0104, 0101, 0611, 0102, 0103, 0104, 0101, 0610, 0102, 0103, 0104, 0101, 0609, 0102, 0105

który przy 100-bajtowej stronie można zredukować do następującego ciągu odniesień do pamięci:
1, 4, 1, 6, 1, 6, 1, 6, 1, 6, 1 (*)
Aby określić liczbę błędów stron dla każdego ciągu odniesień i algorytmu zastępowania stron, należy także znać liczbę dostępnych ramek. Na przykład gdyby były trzy wolne ramki, to w ciągu (*) wystąpiłyby trzy błędy stron.

Str.4
Na ogół oczekujemy zależności takiej jak na rysunku (rys. 3)



D1) FIFO- do zastąpienia wybieramy stronę najstarszą. Może być zrealizowany przez zastosowanie kolejki FIFO. Do zastąpienia będzie wybierana strona z czoła kolejki (najstarsza).Strona sprowadzana do pamięci trafia na koniec kolejki.
Może wystąpić sytuacja ,że na najstarszej stronie zawarta jest zmienna ,która została wcześnie (dawno) zainicjowana lecz jest ciągle używana.

Ćwiczenie: Rozważyć ciąg odniesień 1,2,3,4,1,2,5,1,2,3,4,5 dla algorytmu FIFO i ilości ramek 1-6.Obliczyć w każdym przypadku ilość błędów strony i narysować wykres.
ANOMALIA BELADE'GO.

D2) Algorytm optymalny- (nazywany OPT lub MIN)-wybiera do zastąpienia tą stronę która najdłużej nie będzie używana. Gwarantuje najniższą liczbę błędów stron.
Trudny do realizacji bo wymaga wiedzy o przyszłym ciągu odniesień!
Służy zatem głównie do porównań .
NIE MA ANOMALII.
D3)Algorytm LRU (najdawniej używana strona)- kojarzy z każdą stroną czas jej użycia .Wybiera stronę ,która nie była używana od najdłuższego czasu (z najmłodszą (najmniejszą)wartością czasu).To jest realizacja licznikowa.
Inny sposób to stos .Przy każdym odniesieniu do strony jej numer wyjmuje się ze stosu i umieszcza na samym szczycie .Na spodzie stosu są strony używane najdawniej .
Realizuje się taki stos jako dwukierunkowa listę ze wskaźnikami do czoła i końca listy).

Algorytm ten wymaga rozwiązań sprzętowych.(Ze względu na stosowanie zegara (licznika)lub stosu wymagającego uaktualniania)Gdyby stosować przerwania i za pomocą oprogramowania uaktualniać te struktury, to spowodowałoby to co najmniej 10-krotne spowolnienie wykonania odniesień do pamięci .
NIE MA ANOMALII.
D4) Algorytmy przybliżające LRU
a)Dla każdej strony 8-bitowy rejestr w RAM. Co pewien czas (np. 100 mili sekund) na najbardziej znaczącą pozycje wstawiamy 1 i przesuwamy w prawo z utratą

bitu najmniej znaczącego .Rejestr zawiera historię użycia strony podczas ośmiu okresów .Strona o zawartości rejestru 11000100 była używana później niż strona o zawartości rejestru 01110111.

b)Jeśli wyżej wspomniany rejestr jest 0-bitowy ,to można wykorzystać tylko bit odniesienia algorytmu „drugiej szansy"

Podstawą jest FIFO .P wybraniu strony do zastąpienia sprawdza się jej bit odniesienia .Jeśli Ø to strona zostaje zastąpiona ,jeśli 1 to strona dostaje drugą szansę ,a jej bit odniesienia jest zerowany a czas jej pobrania ustawia się na bieżący .(Realizacja w postaci kolejki cyklicznej).

c) LFU (Least fragmently used)- prowadzony jest licznik odwołań do każdej strony. Zastąpieniu ulega strona z najmniejszą wartością licznika.

d) MFU (most fragmently used) – zastępowanie strony najczęściej używanej. Oba algorytmy z punktów c) i d) nie przybliżają dobrze algorytmu OPT; ponieważ ich realizacja jest dość kosztowna więc są mało popularne.

E) Przydział ramek – rozstrzyga jak rozdzielić wolną pamięć między różne procesy

a) Minimalna liczba ramek jest zdefiniowana przez architekturę komputera. Błąd strony, który wystąpi przed dokończeniem wykonywania rozkazu powoduje, że taki rozkaz musi być powtórzony. Wynika z tego, że należy mieć wystarczającą liczbę ramek do przechowywania wszystkich stron, do których może się odwołać pojedynczy rozkaz.

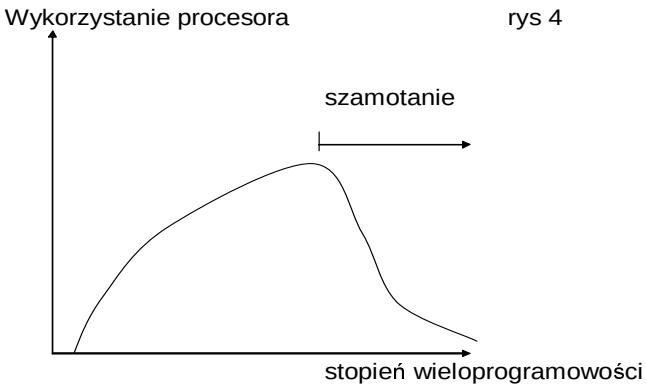
b) Algorytmy przydziału

- przydział równy (5 procesów, 93 ramki $\rightarrow$ 18 ramek/proces + 3 wolne)
- przydział proporcjonalny. Si – wielkość pamięci wirtualnej procesu Pi. Określamy $S=\Sigma Si$. Wówczas gdy m-oznacza ilość ramek to procesowi Pi przydzielamy ai ramek: gdzie $ai=Si/S*m$. Oczywiście ai musi być zaokrąglone w górę do minimalnej liczby ramek, a suma ai nie może przekraczać m ($\Sigma ai \leq m$).

F) Szamotanie występuje wtedy gdy liczba ramek przydzielona do procesu spada poniżej minimalnej liczby ramek określonej przez architekturę komputera. Proces będzie ciągle wykazywał brak strony, będzie zastępował stronę, która za chwilę okaże się potrzebna.

a) Przyczyną szamotania jest zbyt wysoki stopień wieloprogramowości, ustalany przez system operacyjny w celu dociążenia procesora. (rys 4)

W czasie szamotania proces spędza ogromną większość czasu w kolejce do urządzenia stronicującego nie mogąc dokończyć żadnej pracy. Ze względu na bardzo częste błędy stron wzrasta (bardzo) efektywny czas dostępu do pamięci



rys 4

b)W modelu zbioru roboczego zakłada się istnienie stref programu. Definiuje się parametr Δ do określenia okna zbioru roboczego. Polega to na sprawdzaniu ostatnich Δ odniesień do stron. Za zbiór roboczy przyjmuje się zbiór stron do których nastąpiło ostatnich Δ odniesień. Strona, która przestanie być używana wypadnie ze zbioru roboczego po Δ jednostkach czasu. Cechą zbioru roboczego jest jego rozmiar (Work Set Size WSSi).
D=ΣWSSi –ogólne zapotrzebowanie na ramki.
Jeśli D> m to powstaje szamotanie.
c) prostszą strategią zapobiegania szamotaniu jest sterowanie częstością błędów stron. Duża częstość oznacza, że proces potrzebuje więcej ramek. Zarówno w b) jak i w c) istnieje zawsze możliwość wstrzymania procesu. Wolne ramki rozdziela się wtedy między działające procesy z najwyższymi częstościami błędów stron.