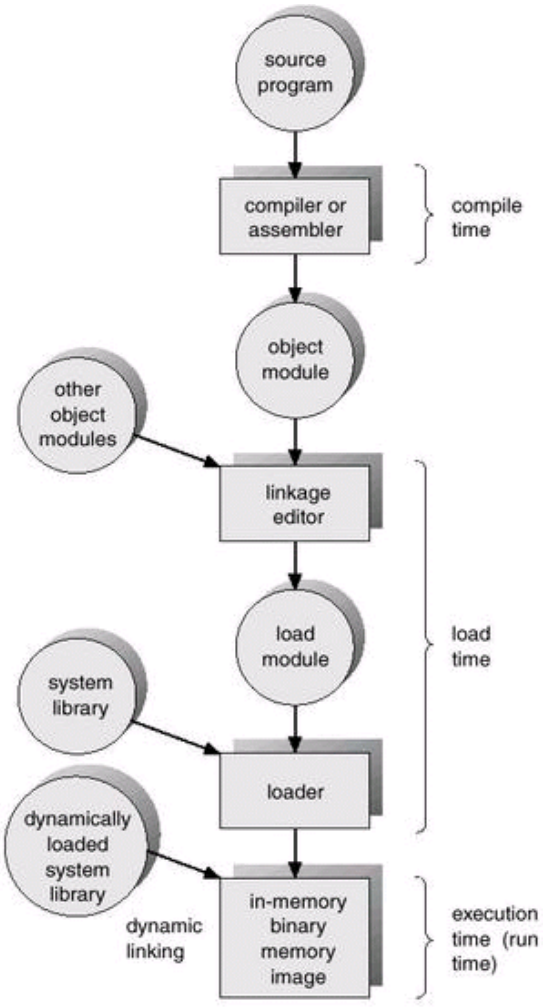
Zarządzanie pamięcią

Wstęp

- Aby program można było wykonać trzeba go umieścić w pamięci w ramach pewnego procesu.
- Kolejka wejściowa — zbiór procesów na dysku czekających na umieszczenie w pamięci.
- Przed rozpoczęciem wykonywania program przechodzi szereg faz.
- Zmiana sposobu reprezentacji adresów: symboliczne → względne → bezwzględne.



Wiązanie rozkazów i danych z adresami

Możliwe jest w każdym z kroków:

- Czas kompilacji: jeśli znamy położenie kodu w pamięci możemy generować kod absolutny. Zmiana tego położenia wymaga rekompilacji kodu.
- Czas ładowania: jeśli nie znamy adresu początkowego w pamięci musimy generować kod relokowalny — ostateczne wiązanie odkłada się do czasu ładowania.
- Czas wykonania: jeśli proces może zmieniać położenie w trakcie wykonywania wiązanie odkłada się do czasu wykonania — konieczny specjalny sprzęt, np. rejestr bazowy i graniczny.

Ładowanie dynamiczne

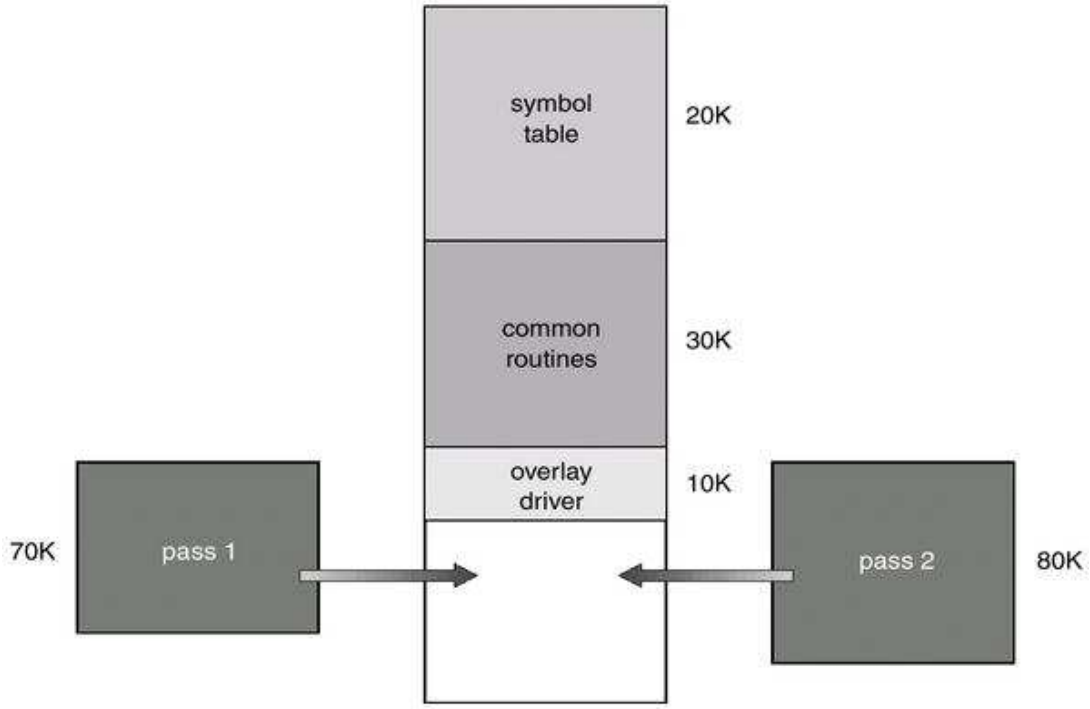
Pozwala na lepsze wykorzystanie pamięci.

- Procedura (w postaci relokowalnej) jest ładowana do pamięci w momencie wywołania.
- Użyteczne w przypadku dużych procedur obsługujących rzadko występujące przypadki.
- Nie wymaga wsparcia ze strony systemu operacyjnego. System może wspierać użytkownika dostarczają procedur odpowiednich bibliotecznych.

Łączenie (linking) dynamiczne

- Konsolidacja jest odłożona do czasu wykonania programu (por. dynamically loaded library).

- Zakładka (stub) w obrazie binarnym w miejscu odwołania się do procedury. Lokalizuje w pamięci albo ładuje odpowiedni kod proceduru oraz realizuje wykonanie.
- Niezbędna pomoc systemu operacyjnego ze względu na mechanizmy ochronne pamięci.
- Umożliwia bezproblemowe zastępowanie bibliotek nowszymi wersjami.



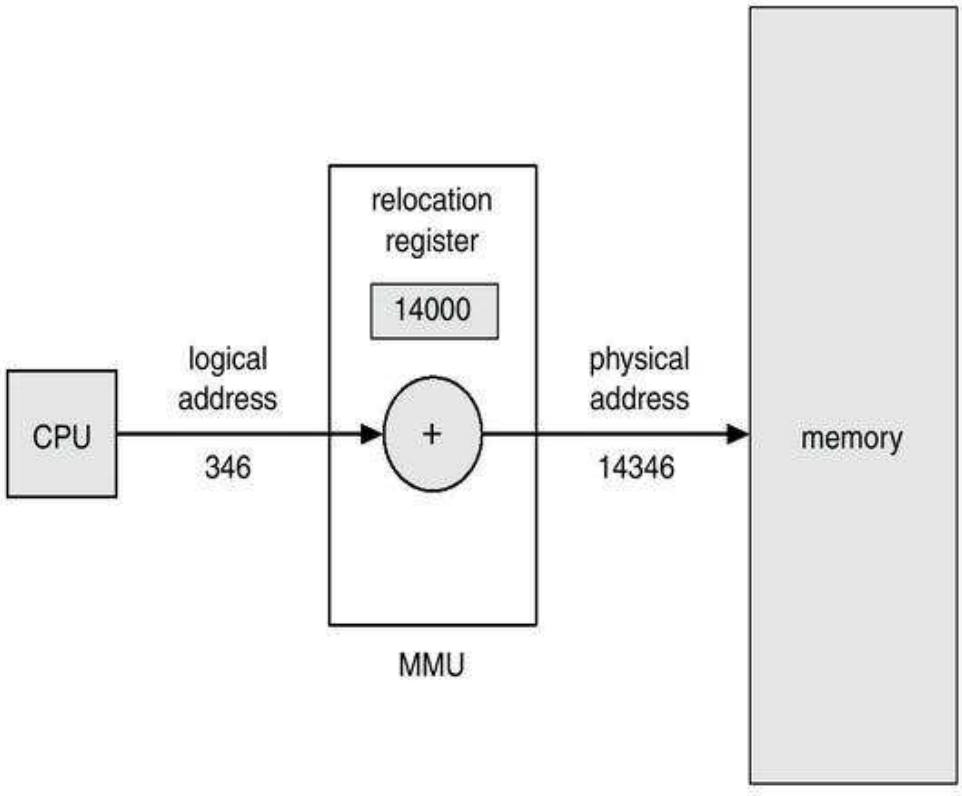
Nakładki (overlay)

- W pamięci znajdują się tylko fragmenty potrzebne w danej chwili.
- Pozwalają na wykonanie programu większego niż dostępny obszar pamięci.
- Nie wymagają wsparcia ze strony systemu — możliwa realizacja środkami programowymi. Implementacja spoczywa na użytkowniku.

Wymaga bardzo dokładnej analizy dużego programu, co nie musi być proste

Logiczna i fizyczna przestrzeń adresowa

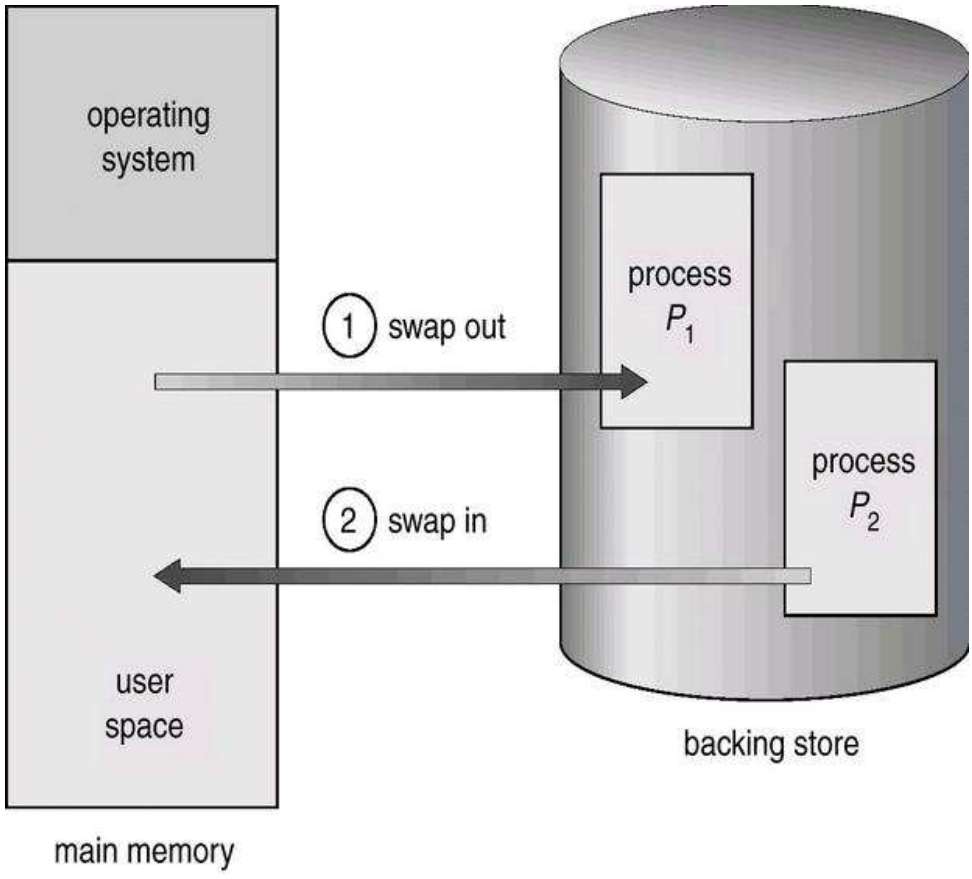
- Adres logiczny — adres z punktu widzenia programu, generowany przez CPU.
- Adres fizyczny — adres widziany przez jednostkę pamięci.
- Wiązanie adresów w trakcie kompilacji i ładowania powoduje, że adresy fizyczne i logiczne są identyczne.
- Wiązanie w trakcie wykonania prowadzi do schematu, w którym adresy te są różne.
- MMU — Memory Menagement Unit.



- Sprzęt mapujący adresy logiczne na fizyczne w trakcie wykonania programu.
- Istnieje kilka sposobów realizacji takiego odwzorowania.
- Jeden z prostszych — wykorzystanie rejestru relokacji.

Wymiana (swapping)

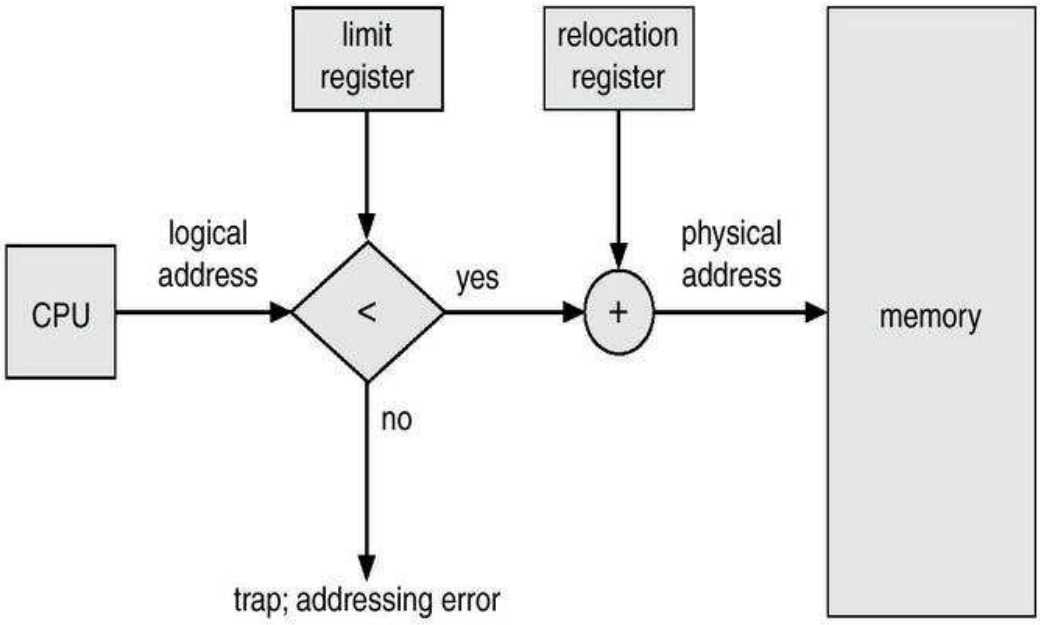
- Proces może zostać czasowo usunięty z pamięci do pamięci pomocniczej a następnie sprowadzony ponownie w celu kontynuowania obliczeń.



- Urządzenie wymiany — szybki dysk, dostatecznie duży aby pomieścić wszystkie obrazy pamięci wszystkich użytkowników i zapewniający dostęp bezpośredni do tych obrazów.

- Roll out, roll in — strategia wymiany w przypadku szeregowania priorytetowego. Procesy o niższych priorytetach są usuwane, aby procesy z wyższymi priorytetami mogły zostać wykonane.
- Podstawowa część czasu wymiany poświęcona jest na transmisje z dysku/ na dysk.
- Wymianie podlegają procesy całkowicie bezczynne. Procesy czekające na zakończenie we/wy nie podlegają wymianie (dostęp do buforów w pamięci użytkownika).
- Warianty wymiany stosowane są w wielu systemach: Unix, Linux, Windows.

Przydział ciągły



Pamięć podzielona zwykle na dwa obszary:

- Rezydentny system operacyjny w obszarze niskich adresów.
- Program użytkownika w pozostałej części.

Przydział pojedynczego obszaru

- Schemat rejestru przemieszczenia stosowany do ochrony programów użytkowników przed sobą i przed naruszaniem kodu systemu.
- Rejestr przemieszczenia zawiera najmniejszy adres fizyczny obszaru użytkownika, rejestr graniczny zawiera zakres adresów logicznych.

Przydział wielu obszarów

- Dziura — spójny obszar wolnej pamięci operacyjnej. Dziury są rozsiane po całej pamięci.
- Pojawiający się proces otrzymuje obszar pamięci dostatecznie duży, aby sie w nim zmieścić.
- System zarządza informacją o przydzielonych obszarach i o wolnych dziurach.
- Jaki obszar spośród wolnych wybrać?
  - first-fit — pierwszy, który się nadaje;
  - best-fit — najmniejszy spośród dostatecznie dużych, wymaga przeszukiwania listy, powoduje powstanie najmniejszej dziury;
  - worst-fit — największy możliwy, powoduje powstanie dużej dziury.
- Dwie pierwsze metody są lepsze (prędkość przetwarzania, wykorzystanie pamięci).

Fragmentacja pamięci

- Fragmentacja zewnętrzna — łączny obszar wolnej pamięci jest wystarczający do zaspokojenia zamówienia, ale nie jest w spójnym kawałku.

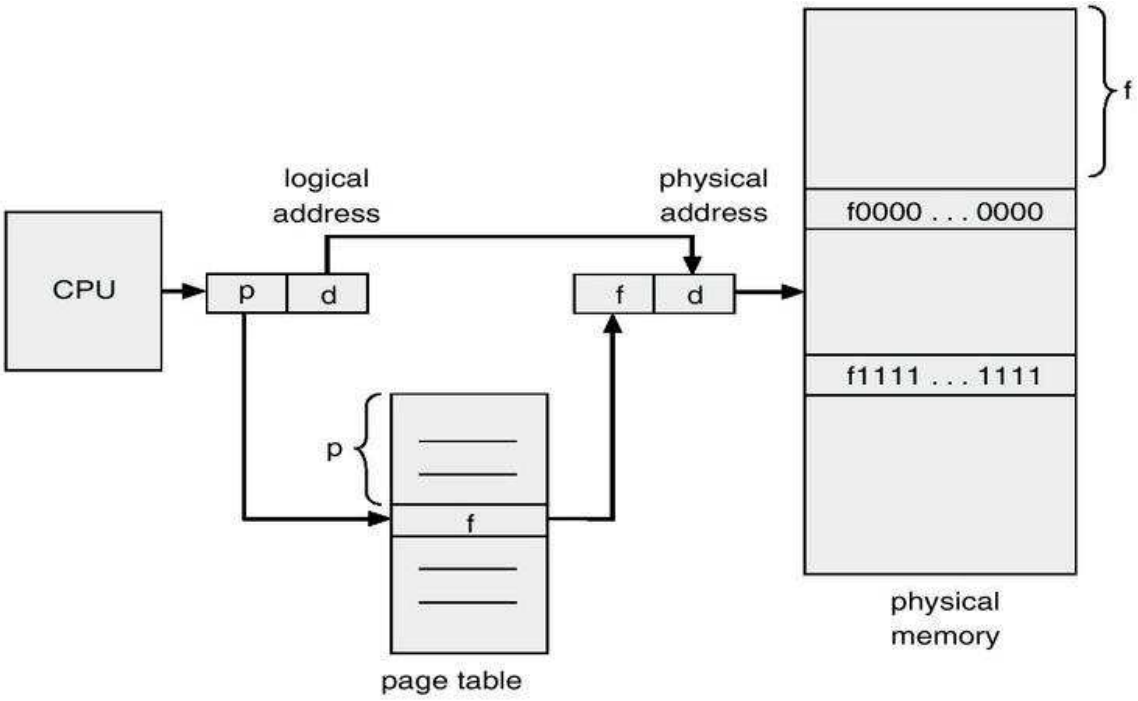
- Fragmentacja wewnętrzna — przydzielona partycja jest nieco większa niż zamówienie (nie opłaca się trzymać informacji o tak małym wolnym obszarze). Należy do przestrzeni procesu mimo, że nie jest wykorzystana.
- Zapobieganie defragmentacji przez upakowanie (kompresję).
  - Przemieszczenie zawartości tak, żeby powstał spójny wolny obszar.
  - Konieczne jest korzystanie z dynamicznego wiązania adresów.
  - Problemy z we/wy.
    * Blokowanie procesów aż do zakończenia we/wy.
    * We/wy wyłącznie przez bufory systemowe.

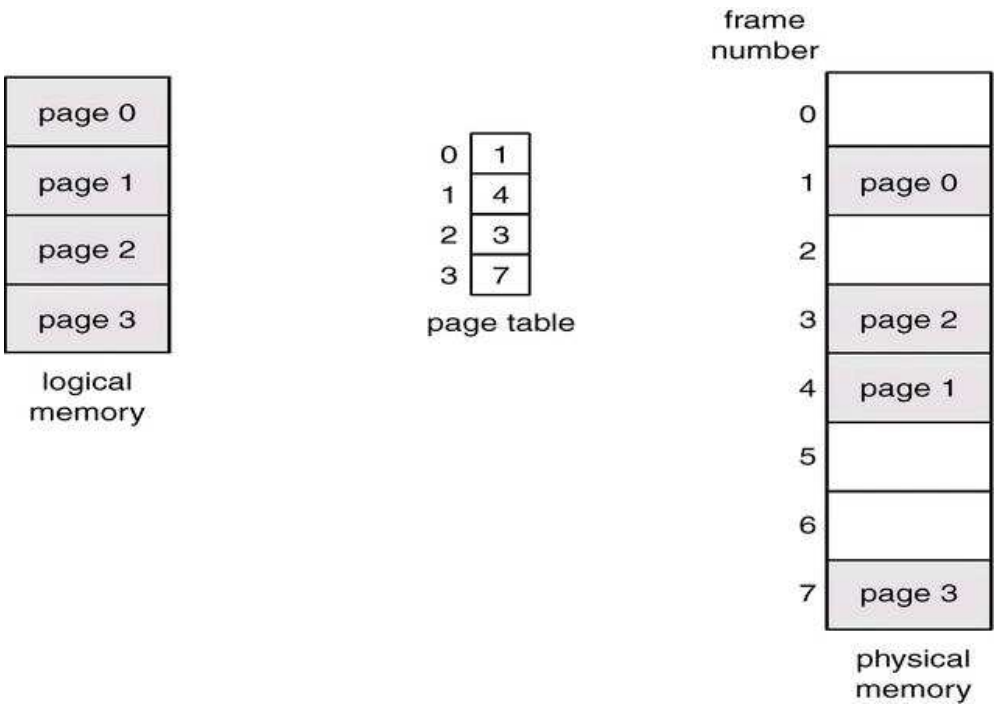
Stronnicowanie pamięci

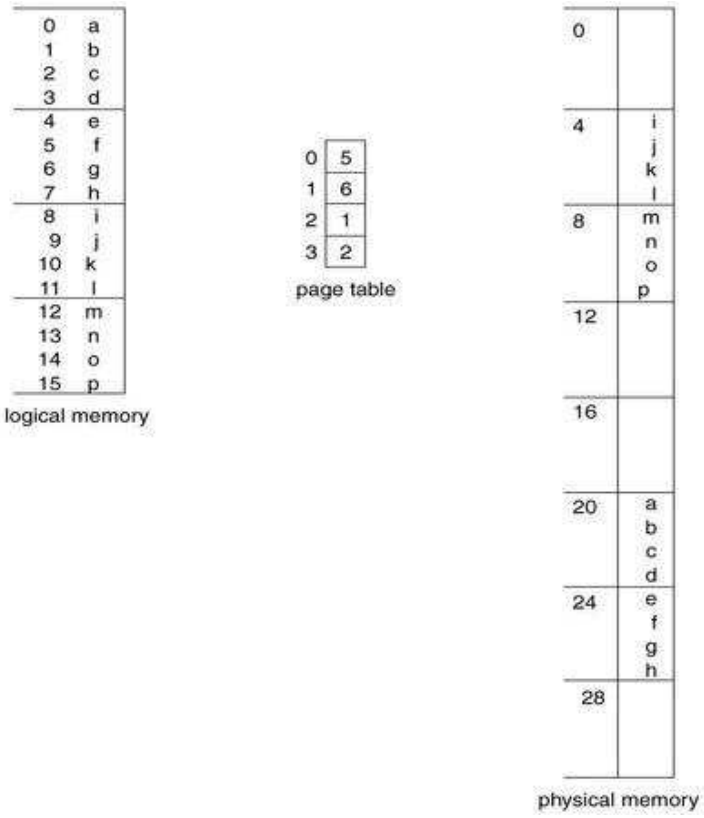
- Dzielimy pamięć fizyczną na jednakowe fragmenty — ramki, wilkości 512 – 8192 bajtów.
- Dzielimy logiczną przestrzeń adresową na takie same fragmenty — strony.
- Utrzymujemy informację o wszystkich wolnych ramkach — tablica ramek.
- Aby wykonać program o n stronach trzeba znaleźć n wolnych ramek i załadować program.
- Tworzymy tablicę stron do tłumaczenia adresów logicznych na fizyczne.
- Fragmentacja wewnętrzna.

Adres logiczny jest dzielony na:

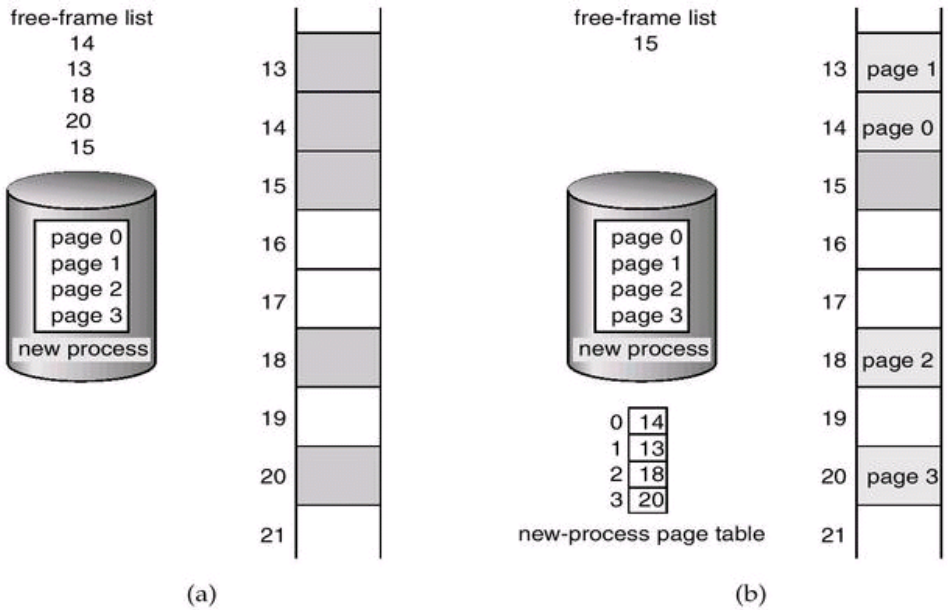
- numer strony będący indeksem do tablicy stron, gdzie znajduje się adres początku tej strony w pamięci fizycznej,
- przesunięcie na stronie, które dodane do adresu podstawowego da adres w pamięci fizycznej.





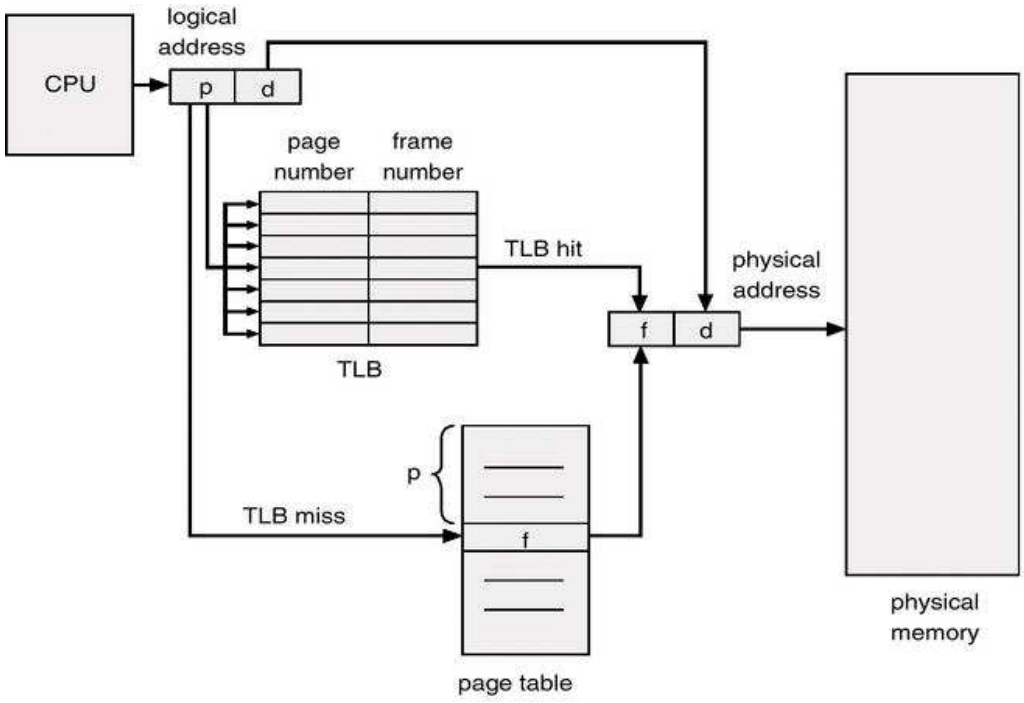

0 a
1 b
2 c
3 d
4 e
5 f
6 g
7 h
8 i
9 j
10 k
11 l
12 m
13 n
14 o
15 p
logical memory
0 5
1 6
2 1
3 2
page table
0
4 i j k l
8 m n o p
12
16
20 a b c d
24 e f g h
28
physical memory

Wolne ramki


free-frame list
14
13
18
20
15
page 0
page 1
page 2
page 3
new process
13
14
15
16
17
18
19
20
21
(a)
free-frame list
15
page 0
page 1
page 2
page 3
new process
0 14
1 13
2 18
3 20
new-process page table
13 page 1
14 page 0
15
16
17
18 page 2
19
20 page 3
21
(b)

Implementacja tablicy stron

- Podstawowy wariant — tablica stron znajduje się w pamięci.
- Rejestr PTBR wskazuje początek tablicy, rejestr PTLR — długość tablicy stron.
- Przy tym schemacie każdy dostęp do danych/instrukcji wymaga dwóch dostępów do pamięci.
- Problem można rozwiązać przez wykorzystanie rejestrów TLB — małej i szybko przeszukiwalnej pamięci asocjacyjnej.



Efektywny czas dostępu dla takiej implementacji:

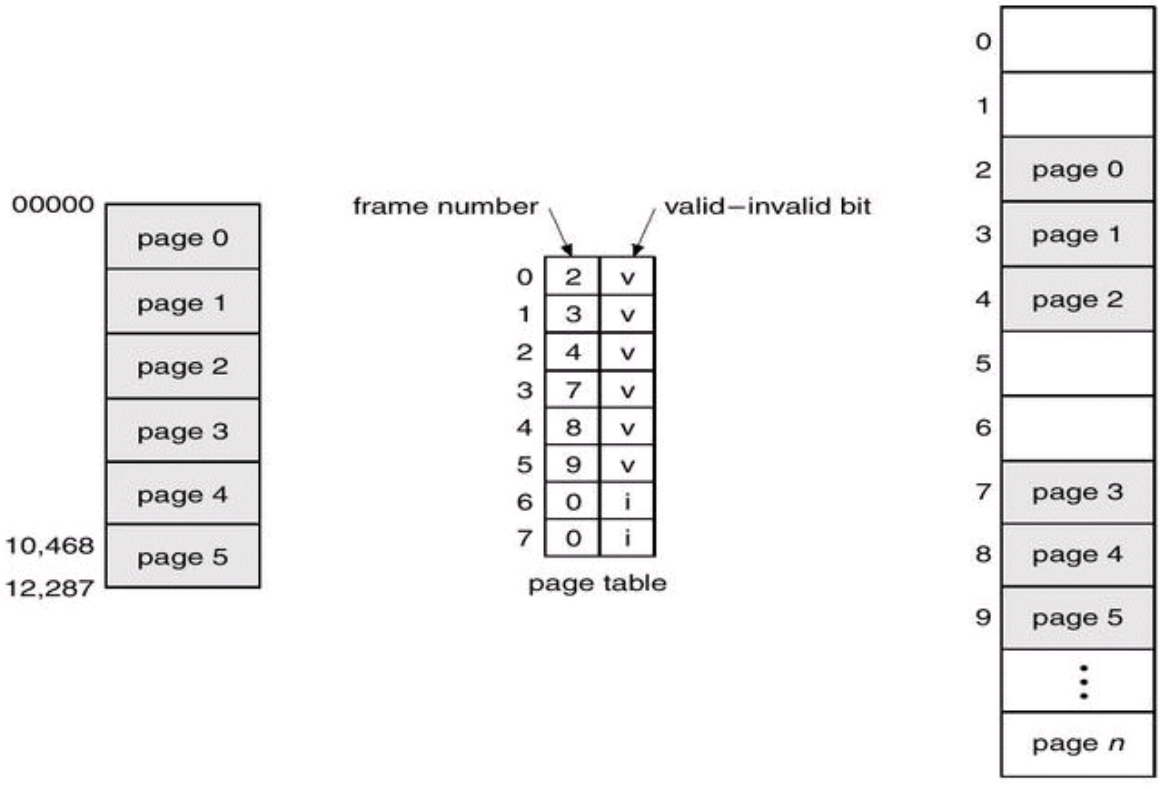
- czas szukania asocjacyjnego — ϵ,
- czas dostępu do pamięci — 1 mikrosekunda,
- współczynnik trafień (prawdopodobieństwo, że numer strony jest rejestrze asocjacyjnym) — α,

- efektywny czas dostępu

$$EAT = (1 + \epsilon)\alpha + (2 + \epsilon)(1 - \alpha)$$

Kontrola poprawności

- Ochrona pamięci może być realizowana przez dołączenie do każdej pozycji tablicy stron dodatkowego bitu poprawności.
- Ustawienie tego bitu na ’v’ oznacza, że strona należy do przestrzeni adresowej procesu — odwołanie się jest legalne.
- Ustawienie na ’i’ oznacza stronę nieważną.
- Fragmentacja wewnętrzna — część adresów ostatniej legalnej strony jest niepoprawna.

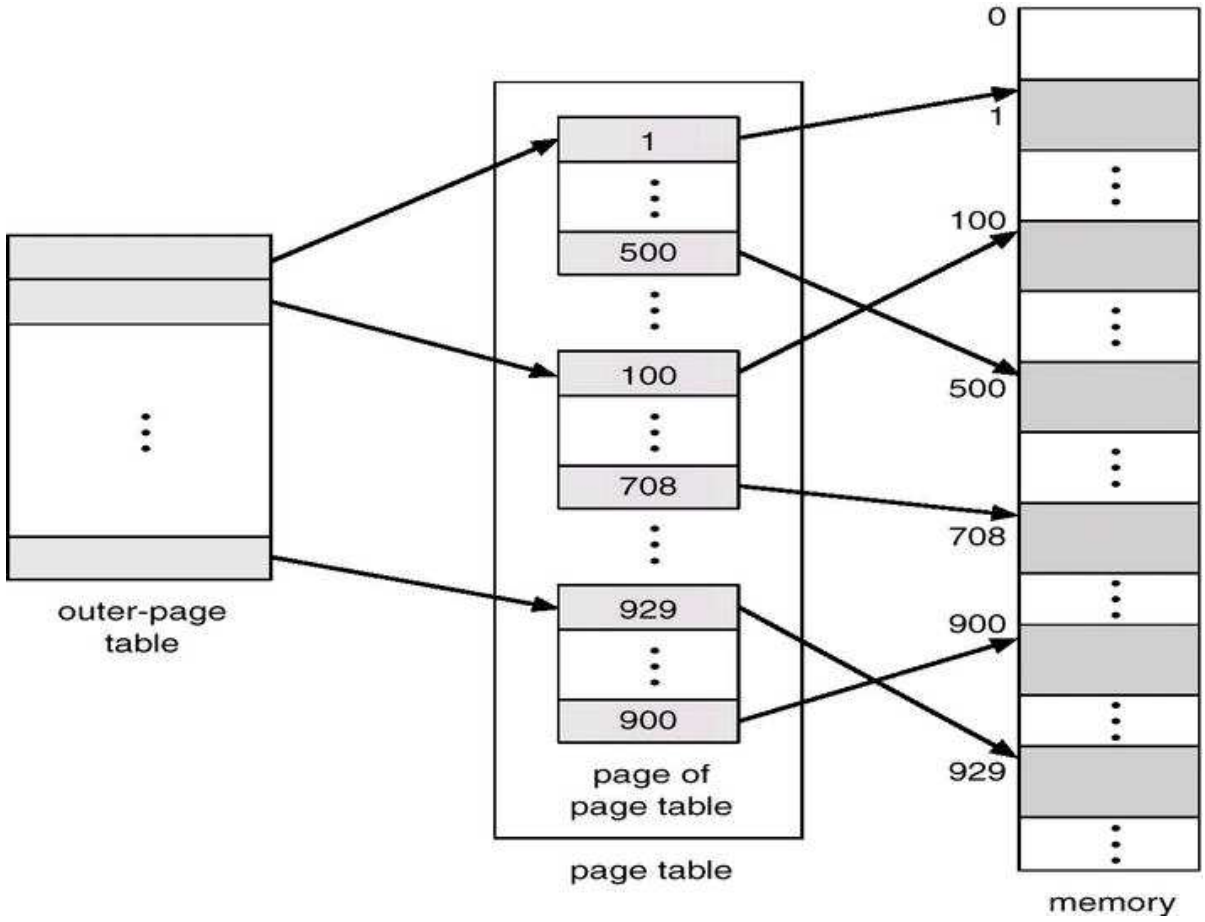


Stronicowanie wielopoziomowe

- Duże przestrzenie adresowe powodują, że tablice stron mogą być bardzo duże (przy 32 bitowym adresie i stronie wielkości 4kB - tablica stron ma 1M pozycji, czyli wielkość 4MB).
- Nie chcemy przydzielać ciągłego obszaru na taka tablicę.
- Rozwiązanie — podzielenie tablicy stron na mniejsze części (można to zrobić na kilka sposobów).

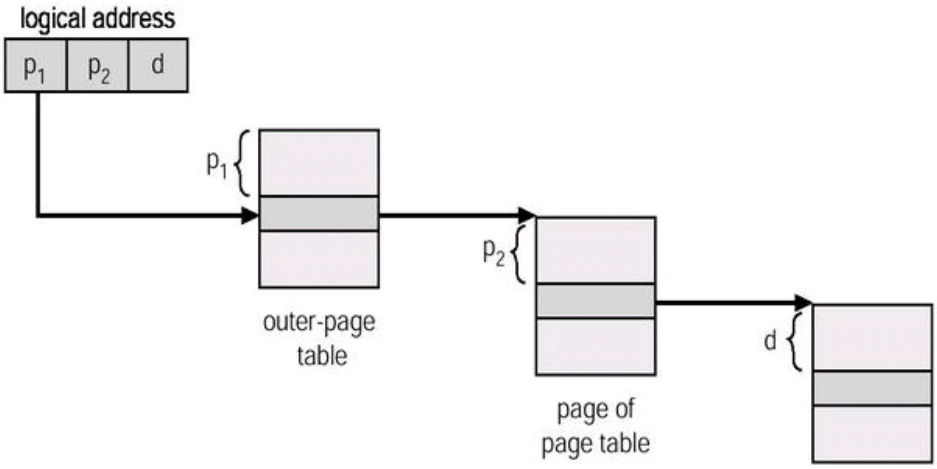
Stronicowanie dwupoziomowe

Tablica stron podzielona jest na strony.



Przykład:

- Adres 32 bitowy, przy stronie wielkości 4kB dzieli się na:
  - 20 bitowy numer strony,
  - 12 bitowy offset na stronie.
- Ponieważ tablica stron ma być stronicowana więc numer strony dzieli się dalej na:
  - 10 bitowy numer strony,
  - 10 bitową odległość na stronie.
- Ostatecznie adres logiczny przyjmuje postać:
  - p_1 — indeks w zewnętrznej tablicy stron,
  - p_2 — przesunięcie na stronie zewnętrznej tablicy stron,
  - d — odległość na stronie.

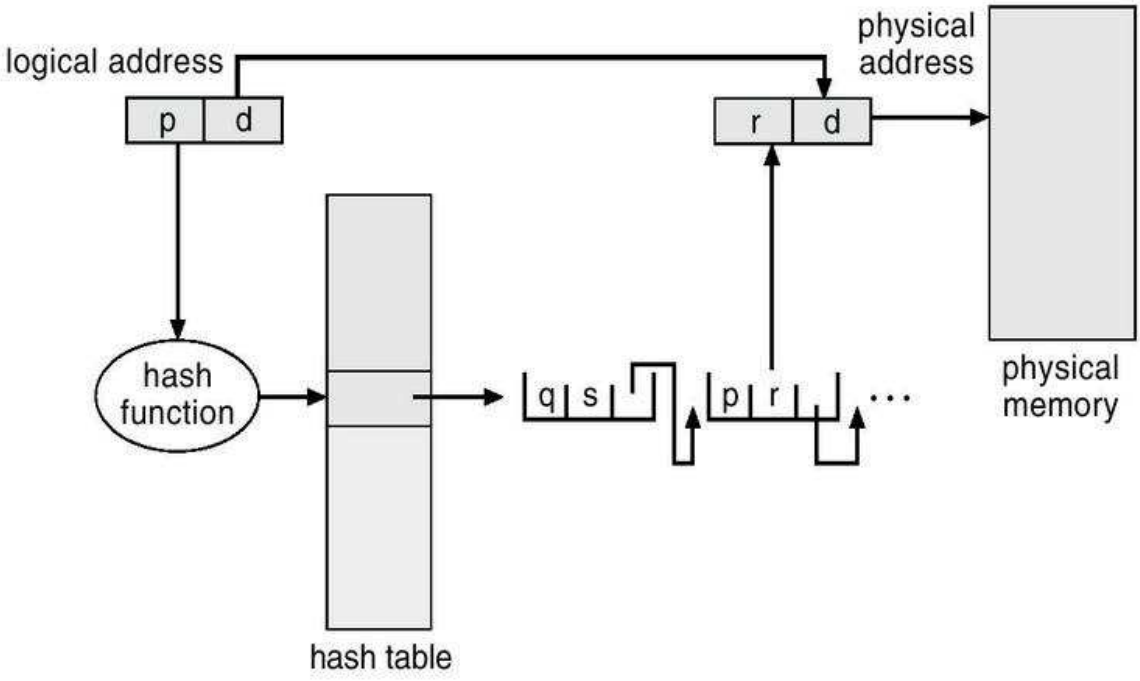


Hashowane tablice stron

Dla większych przestrzeni adresowych schemat dwupoziomowy może być niewystarczający.

Zastosowanie większej ilości poziomów wyraźnie zwiększa efektywny czas dostępu.

Inne rozwiązanie to zastosowanie funkcji hashującej (mieszającej).

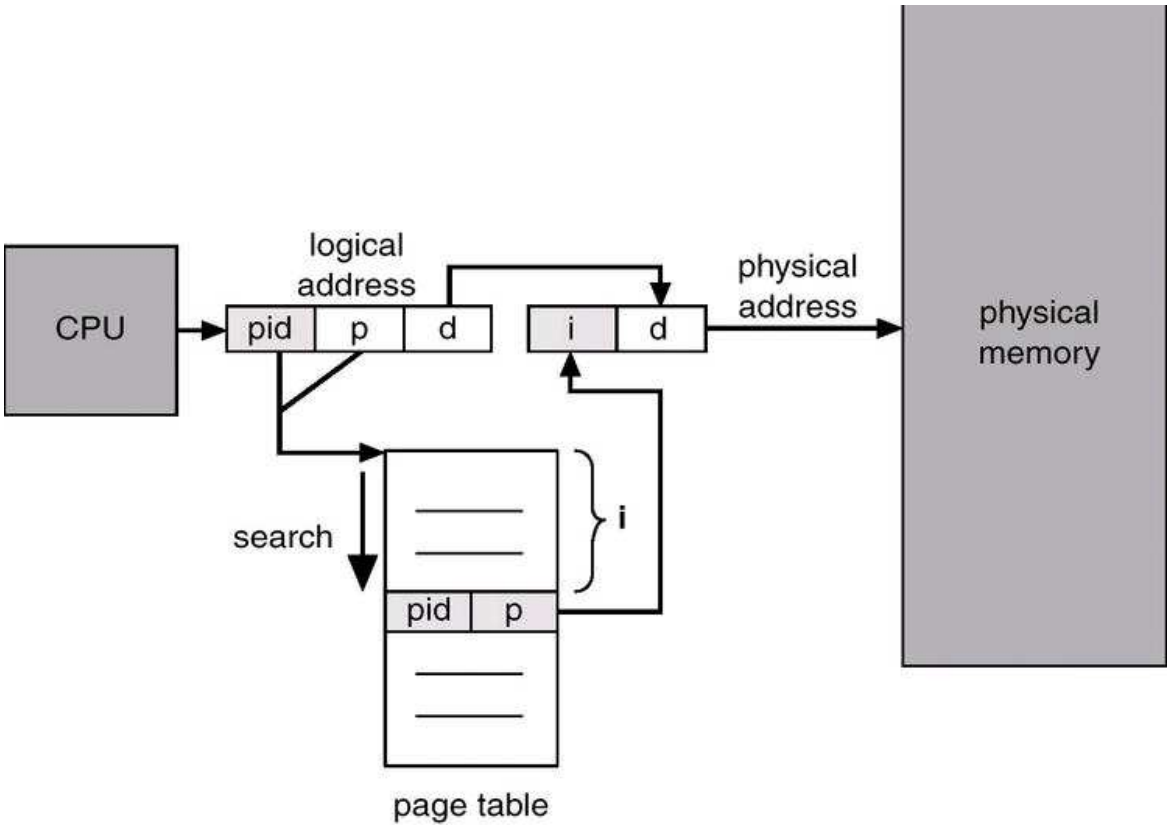


Odwrotna(odwrócona) tablica stron

Tablice stron w systemie w sumie zajmują bardzo dużo miejsca w pamięci.

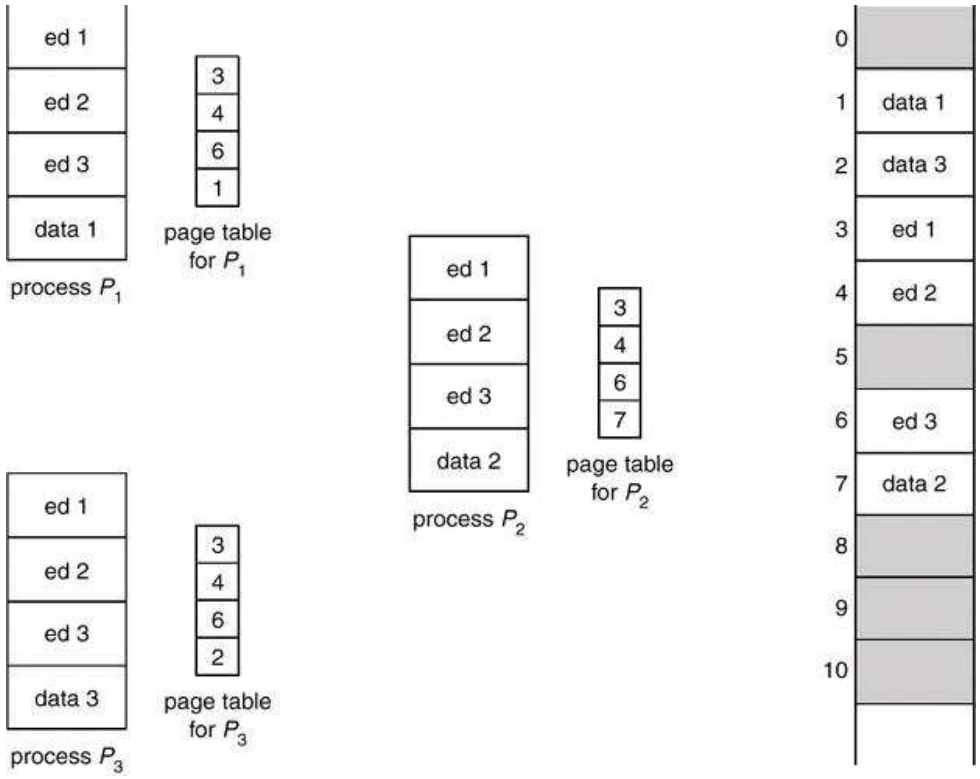
Niekiedy stosuje się inną koncepcję.

- Tablica z jedną pozycją dla każdej ramki pamięci fizycznej.
- Każda pozycja zawiera informację o adresie wirtualnym strony, która jest w tej ramce oraz o właścicielu tej strony.
- Zmniejsza zapotrzebowanie na pamięć potrzebną na tablice stron kosztem wzrostu czasu dostępu do pamięci.
- Zastosowanie tablicy hashującej ogranicza czas przeszukiwania odwrotnej tablicy stron.



Strony dzielone

- Kod wspólny.
  - Jeden egzemplarz kodu (nie modyfikującego sam siebie), np. edytora albo kompilatora.
  - Zwykle Musi występować w logicznych przestrzeniach adresowych kilku różnych procesów w tych samych miejscach (adresacja w obrębie programu).
  - Trudności implementacyjne przy odwrotnej tablicy stron.
- Prywatny kod i dane.
  - Każdy proces utrzymuje prywatną kopię.
  - Może występować w dowolnym miejscu przestrzeni adresowej.



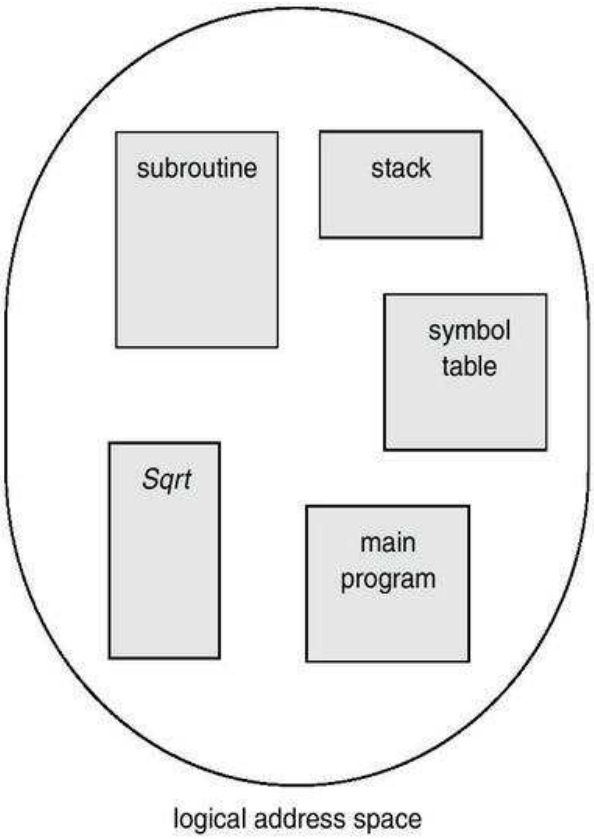
Segmentacja

Schemat zarządzania pamięcią uwzględniający punkt widzenia programisty.

Program jest zbiorem segmentów — elementów logicznych takich jak

- program główny,
- procedura,
- funkcja,
- obiekt,
- metoda,
- zmienne lokalne, zmienne globalne,
- bloki wspólne,
- stos,

- tablice symboli, tablice danych.



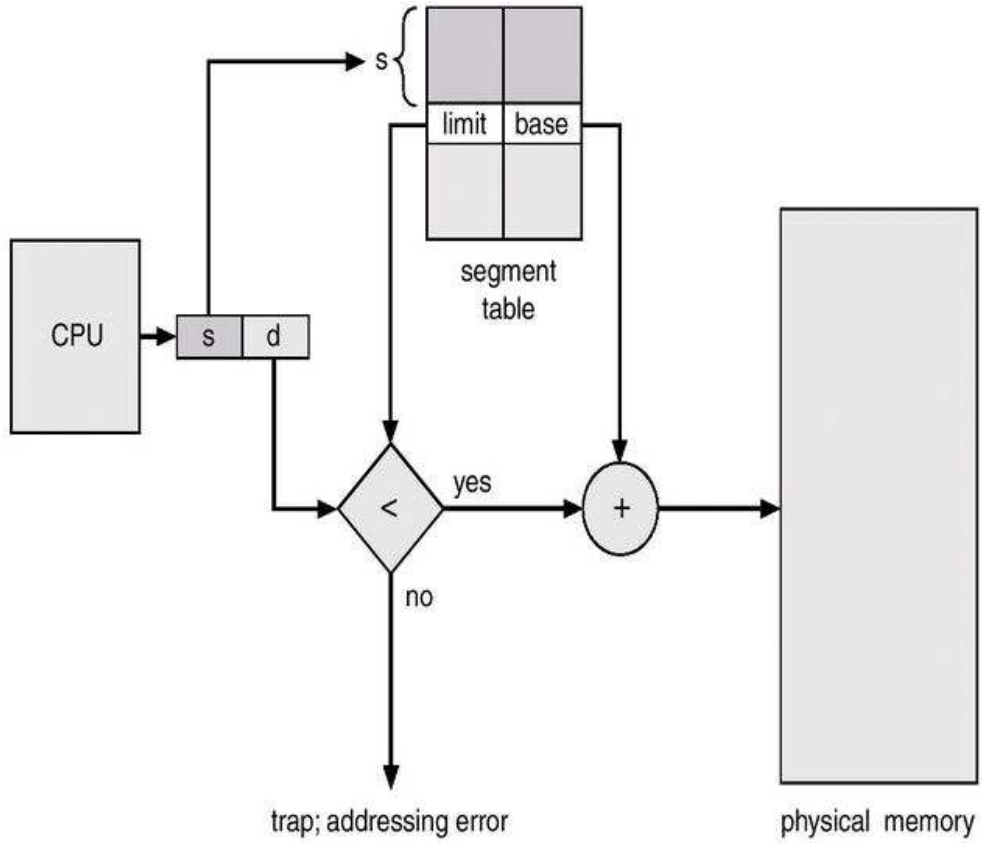
Architektura

- Adres logiczny jest parą:

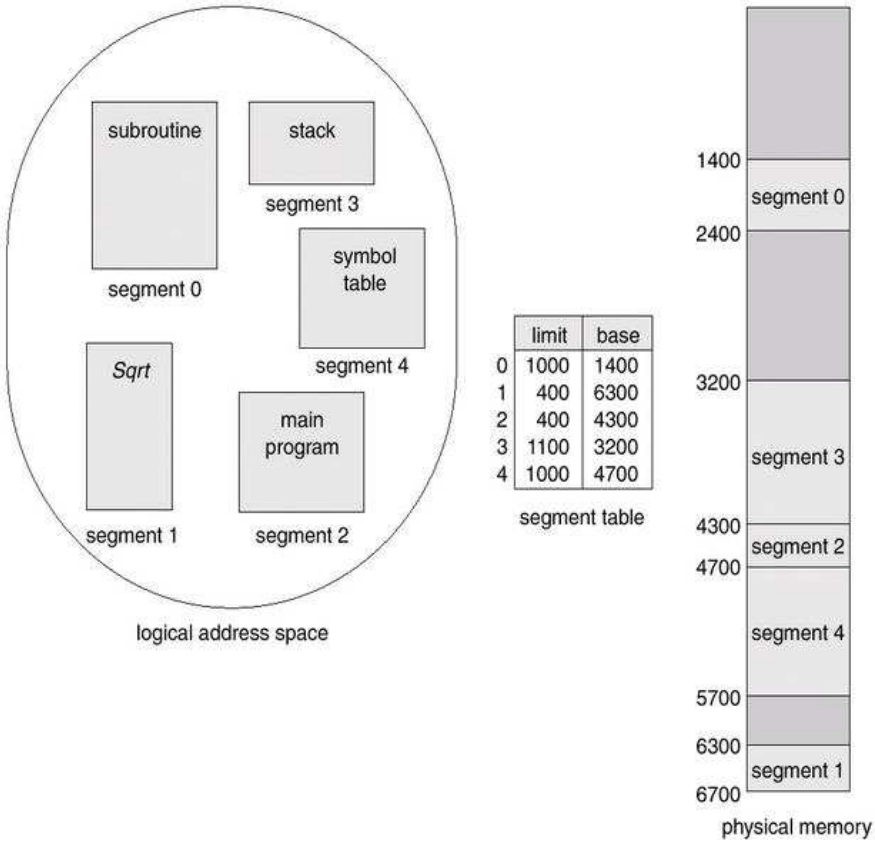
$$< numer\_segmentu, offset >$$

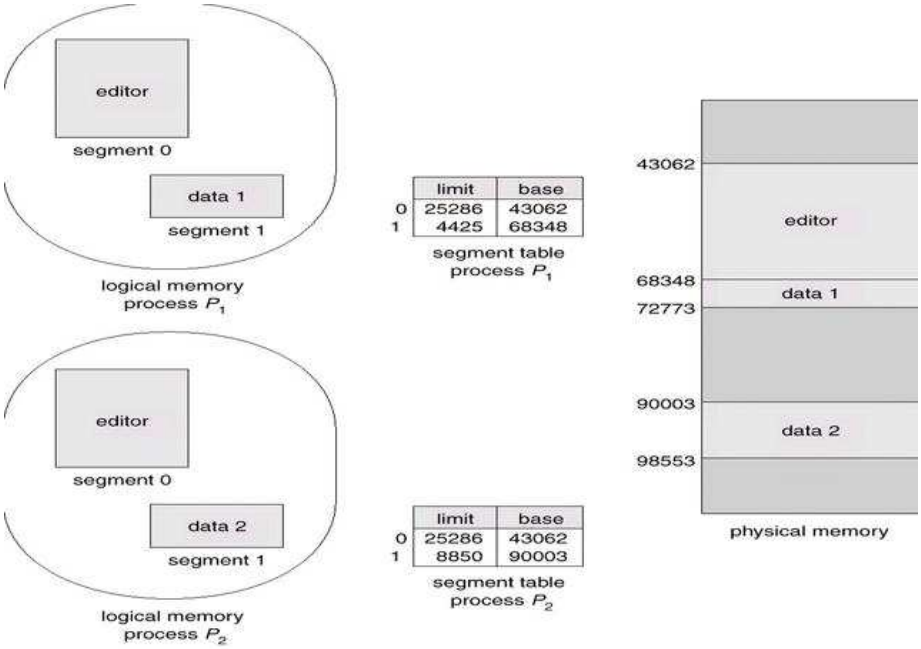
- Tablica segmentów odwzorowuje logiczny adres dwuwymiarowy na jednowymiarową pamięć fizyczną. Każda pozycja zawiera:
  - bazę — adres początku segmentu w pamięci,
  - granicę segmentu — długość segmentu.
- STBR — rejestr początku tablicy segmentów zawierający adres początku tablicy segmentów.

- STLR — rejestr zawierający długość tablicy segmentów. Numer segmentu s jest legalny, jeśli $s < STLR$.



- Ochrona.
  - bit poprawności —jeśli wartość 0, segment nielegalny.
  - bity dostępu — czytanie/pisanie/wykonywanie.
- Dzielenie kodu występuje na poziomie segmentu.
  - numeracja segmentów
- Segmenty różnej długości — dynamiczna alokacja.
  - best fit/first fit,
  - fragmentacja.


subroutine
stack
segment 3
segment 0
symbol table
segment 4
Sqrt
main program
segment 1
segment 2
logical address space
limit base
0 1000 1400
1 400 6300
2 400 4300
3 1100 3200
4 1000 4700
segment table
1400
segment 0
2400
3200
segment 3
4300
segment 2
4700
segment 4
5700
6300
segment 1
6700
physical memory


editor
segment 0
data 1
segment 1
logical memory process P_1
limit base
0 25286 43062
1 4425 68348
segment table process P_1
editor
segment 0
data 2
segment 1
logical memory process P_2
limit base
0 25286 43062
1 8850 90003
segment table process P_2
43062
editor
68348
data 1
72773
90003
data 2
98553
physical memory

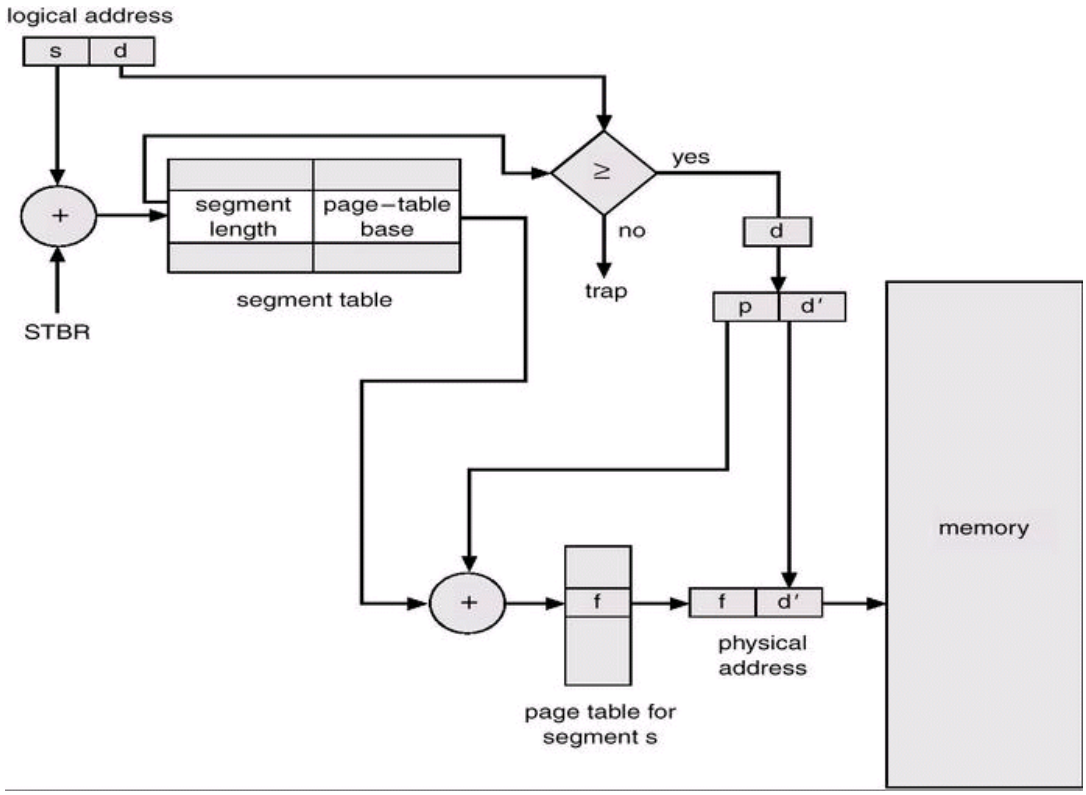
Segmentacja ze stronicowaniem — MULTICS

Problem fragmentacji oraz wyszukiwania w strategii przydziału obszaru rozwiązano dokonując stronicowania segmentów.

Pozycja tablicy segmentów zawiera zamiast adresu początku segmentu adres początkowy tablicy stron segmentu.

Dla każdego segmentu konieczna jest oddzielna tablica stron.

W rzeczywistości schemat ten jest bardziej skomplikowany — adres jest czteroczęściowy.



Schemat adresacji procesora intel 386