

# Upravljanje memorijom

- Background
- Razmena (Swapping)
- Kontinualna alokacija (Contiguous Allocation)
- Diskontinualna alokacija (Discontinuous Allocation)
- Straničenje (Paging)
- Segmentcija
- Segmentacija sa straničenjem
- Pregled

# Background

- **Memorija je jedan od fundamentalnih delova OS**
- **Glavne funkcije memorijskog menadžera:**
  - ☞ **1. Vodi računa o korišćenju memorije**
  - ☞ **2. Dodeljuje memoriju procesima kad je zatraže**
  - ☞ **3. Oslobađa memoriju od procesa kad završe svoje aktivnosti**
  - ☞ **4. Vršiti razmenu između memorije i diska**
    - 📄 **kada glavna memorija nije dovoljno velika**
    - 📄 **da sadrži sve procese**

# Background

- Program se mora učitati
- **u memoriju**
- unutar adresnog prostora novostvorenog procesa
  
- **Ulazni red (Input queue):**
  - ☞ kolekcija procesa na disku
  - ☞ koja čeka povratak u memoriju
  - ☞ i nastavak izvršenja
  
- **Korisnički program**
  - ☞ prolazi **preko više faza**
  - ☞ dok ne dođe do izvršavanja

# Vezivanje adresa

Korisnički program prolazi **kroz više faza** dok ne dođe do izvršavanja  
**Adrese** mogu biti predstavljene  
na **različite načine**  
**za vreme ovih faza**

## ■ 1. Adrese u izvornom programu

☞ su **simboličke**

☞ (npr. X)

## ■ 2. Prevodilac (Compiler)

☞ **vezuje ove simboličke adrese**

☞ **za relokabilne adrese**

☞ (npr. promenljiva count se vezuje na lokaciju na adresi 14 u odnosu na početak modula)

## ■ 3. Punilac (Loader)

☞ **pretvara ove relokabilne adrese**

☞ **u apsolutne adrese**

☞ (npr. 74014)

■ **Vezivanje (Binding)** = mapiranje iz jednog adresnog prostora u drugi

# Vezivanje adresa

## ■ Vezivanje

- ☞ instrukcija i podataka na memorijske adrese
- ☞ može se izvršiti **na tri različita načina:**

## ■ 1. Vreme prevođenja (Compile time):

- ☞ Kako nije poznato gde će proces koji će izvršavati,
- ☞ generisani kod biti smešten u memoriji;
- ☞ prevodilac generiše relativne, a ne apsolutne adrese

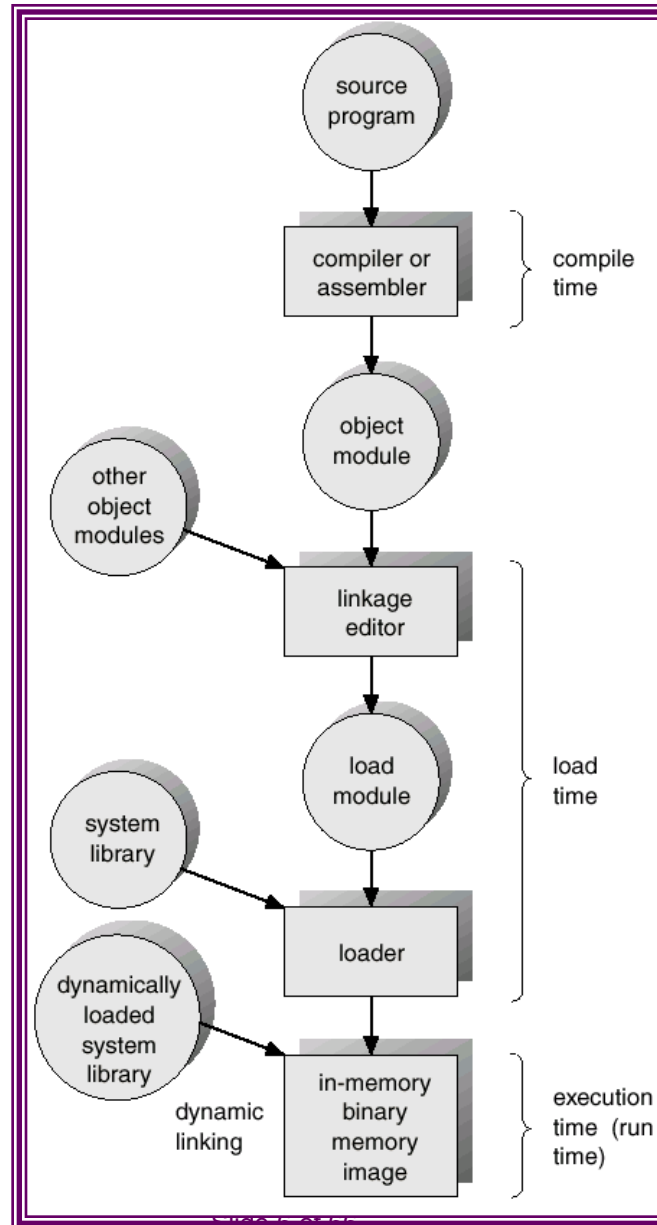
## ■ 2. Vreme učitavanja u memoriju (Load time):

- ☞ Povezivač i punilac na bazi relokabilnog koda
- ☞ generišu apsolutne adrese i pune memoriju programom

## ■ 3. Vreme izvršavanja (Execution time):

- ☞ Za vreme izvršavanja,
- ☞ proces se može pomerati
- ☞ s jednog memorijskog segmenta na drugi.
- ☞ Zahteva hardversku podršku za adresno mapiranje

# Višefazno procesiranje korisničkog programa



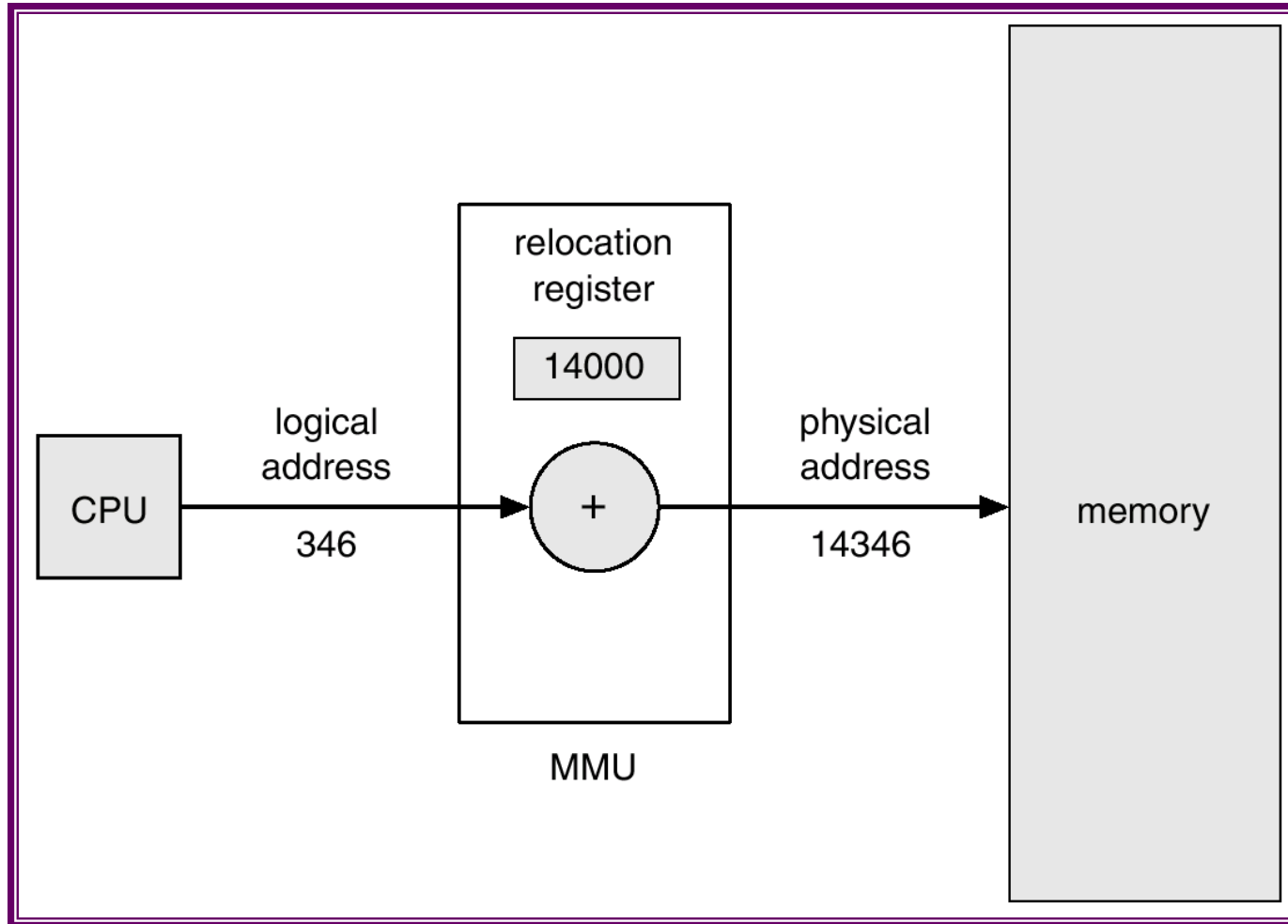
# Logički i fizički adresni prostor

- **Koncept logičkog adresnog prostora**
  - ☞ bazirano na razdvojenim fizičkim adresni prostorima
  - ☞ ima centralnu ulogu u upravljanju memorijom.
- **Logička adresa:**
  - ☞ generisana od strane CPU;
  - ☞ u fazi izvršavanja naziva se i virtuelna adresa
- **Fizička adresa:**
  - ☞ adresa same memorijske jedinice
- **Logičke i fizičke adrese su potpuno iste**
  - ☞ u fazi prevođenja
  - ☞ u fazi učitavanja
- **Logičke i fizičke adrese**
  - ☞ se razlikuju u fazi izvršavanja

# Jedinica za upravljanje memorijom (MMU-Memory Management Unit)

- **MMU: Hardverski uređaj koji mapira**
  - ☞ **virtuelni adresni prostor u fizički**
- U naj-prostijoj MMU šemi,
  - ☞ vrednost u **relokacionom registru**
  - ☞ se sabira sa logičkom adresom koju generiše program
  - ☞ i tako se dobija fizička adresa
- **Korisnički program**
  - ☞ **uvek počinje od nulte adrese;**
  - ☞ i ne treba da vodi računa o svom fizičkom prostoru.

# Mapiranje pomoću relokacionog registra



# Razmena (Swapping)

## ■ Proces se može

- ☞ **privremeno prebaciti iz memorije na disk,**
- ☞ i onda
- ☞ **ponovo vratiti u memoriju da bi nastavio izvršavanje**

## ■ Vraćanje u memoriju:

- ☞ **brzi disk, dovoljno veliki**
- ☞ **da prihvati kopije za sve memorijske slike za sve korisnike;**
- ☞ **potrebno je obezbediti direktan pristup za ove memorijske slike**

## ■ **swap out, swap in:**

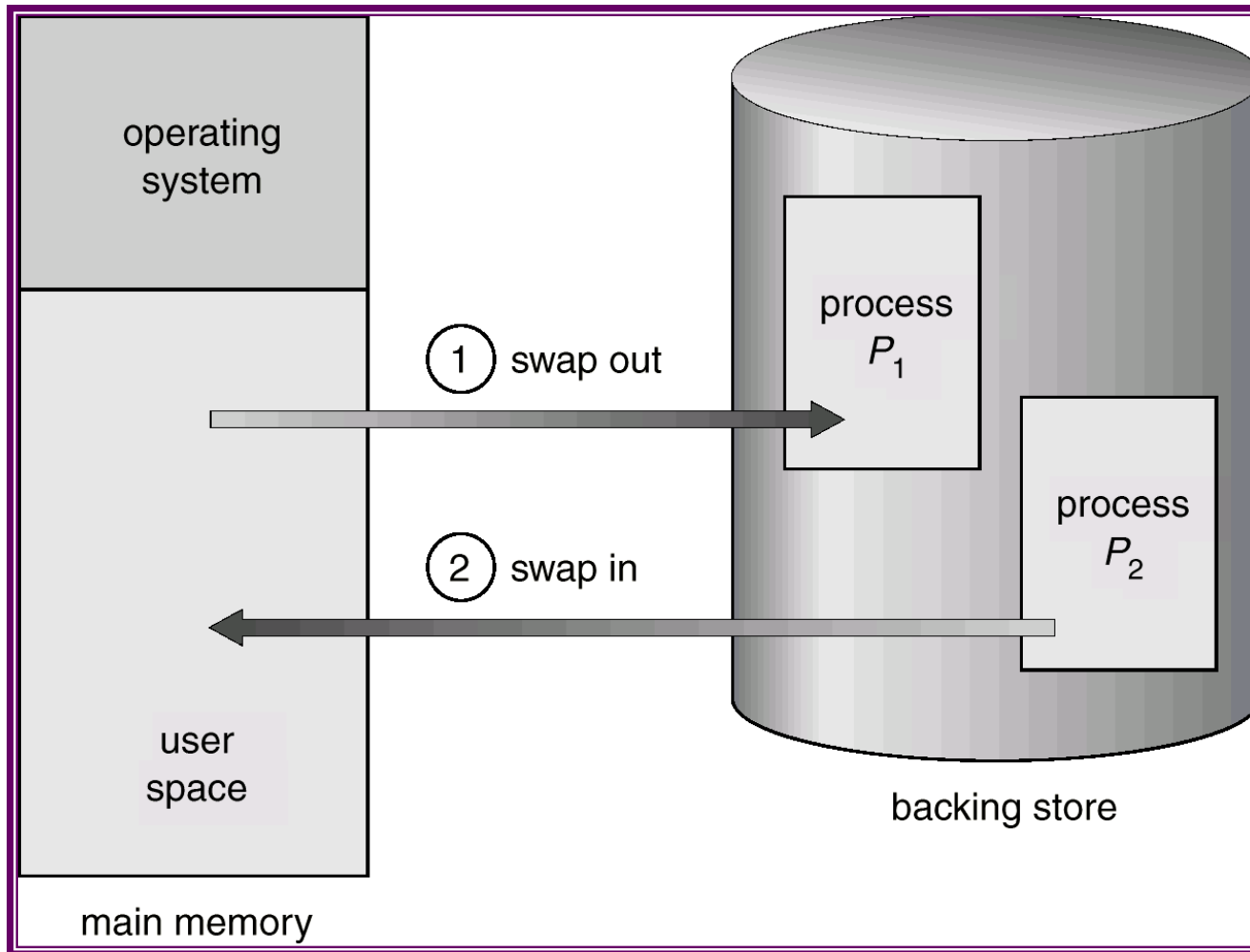
- ☞ razmenjivanje se koristi **u prioritetnim šemama** za raspoređivanje procesa;
- ☞ **proces niskog prioriteta** se upisuju na disk dok se
- ☞ **proces visokog prioriteta** učitavaju u memoriju i izvršavaju

## ■ **Najveći deo vremena** u ciklusima razmene otpada na **prenos podataka;**

- ☞ **trajanje jedne razmene**
- ☞ **zavisi od količine podataka za prenos, karakteristika diskova i hardvera**

## ■ **Modifikovana verzija razmenjivanja postoji na svim OS,** npr., UNIX, Linux, i Windows

# Šematski prikaz razmene



# Dinamičko učitavanje (Dynamic Loading)

- **Rutina se puni u memoriju samo kada je program pozove**
- **Bolja iskorišćenost memorijskog prostora;**
  - ☞ nepotrebne rutine se ne učitavaju u memoriju
- **Korisno je kod:**
  - ☞ velikih programa
  - ☞ jer se u memoriju smeštaju
  - ☞ samo potrebni delovi programa.
- **Ne zahteva specijalnu podršku**
  - ☞ od **operativnog sistema**
  - ☞ **programer** projektuje programe da koriste dinamičko punjenje

# Preklapanje (Overlays)

## ■ Overlay čuva u memoriji

- ☞ samo one delove programa
- ☞ koji su potrebni u tom trenutku

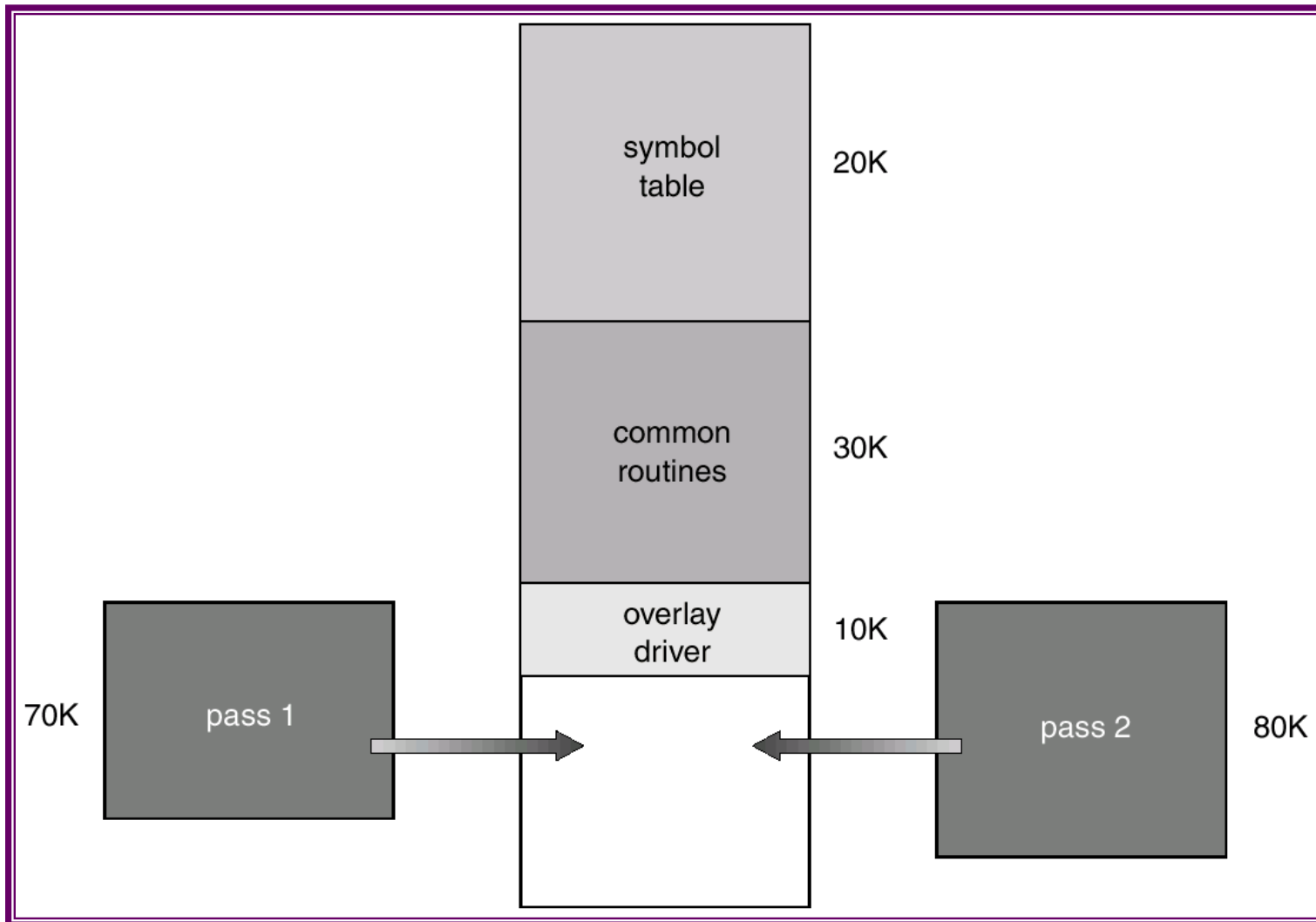
## ■ Koristi se kada je proces veći

- ☞ od memorije koja mu je dodeljena

## ■ Implementira je programer,

- ☞ ne postoji specijalna podrška operativnog sistema
- ☞ konkretnu strukturu **overlay delova** definiše **programer**

# Preklapanje kod dvoprolaznog asemblera



# Dinamičko povezivanje (Dynamic Linking)

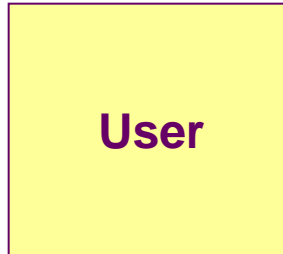
- Linkovanje je odloženo za kasnije, u toku izvršavanja
- Malo parče koda za svaki poziv sistemske biblioteke,
  - ☞ stub,
  - ☞ pokazuje kako da se locira odgovarajuća sistemska rutina
- Stub
  - ☞ puni rutinu u memoriju
  - ☞ i
  - ☞ izvršava je
- Operativni sistem obezbeđuje da
  - ☞ da istu sistemsku rutinu koja je u memoriji
  - ☞ mogu koristiti više procesa.
- Dinamičko povezivanje je naročito korisno za sistemske biblioteke.

# Tipovi memorijskih alokacija

- Razlikujemo različite tipove memorijske alokacije na osnovu:
  - ☞ Efektivnosti
  - ☞ Složenosti
  - ☞ Hardverske podrške
- **Kontinualna alokacija:**
  - ☞ **Single-programming**
    - ☞ **Bare**
    - ☞ **Resident Monitor**
  - ☞ **Multiprogramming**
    - ☞ **MFT**
    - ☞ **MVT**
- **Diskontinualna alokacija:**
  - ☞ **Paging**
  - ☞ **Segmentation**

# Kontinualna alokacija za jedan proces

- **Bare machine**



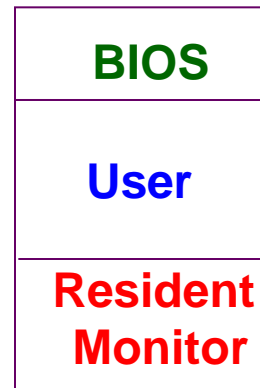
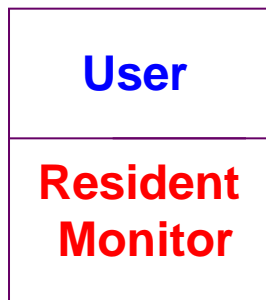
- **Dve particije: Resident Monitor + User area**

- ☞ **Rezydentni deo operativnog sistema**

- 📄 **se obično čuva u nižoj memoriji**

- 📄 **sa tabelom prekidnih rutina**

- ☞ **Korisnički procesi se drže u višoj memoriji**



# Kontinualna alokacija

## ■ Multiprogramming:

- ☞ više korisničkih programa se čuvaju u memoriju,
- ☞ istovremeno
- ☞ bez mešanja

## ■ Alokacija uz pomoć particija

## ■ relokacioni registar

- ☞ se koristi da zaštiti
  - 📄 korisničke procese između sebe,
  - 📄 i
  - 📄 operativnog sistema od korisničkih procesa.

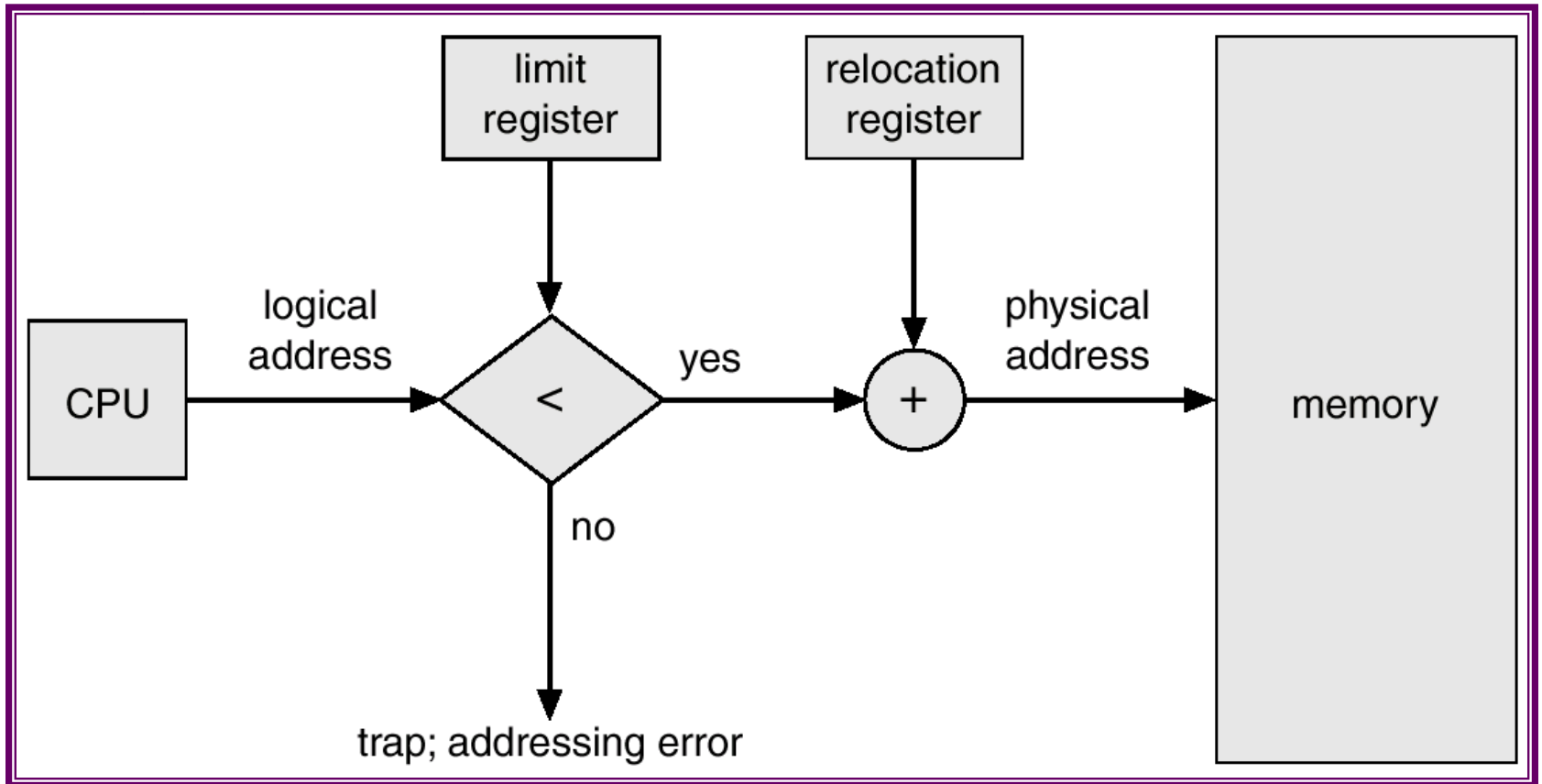
## ■ relokacioni registar

- ☞ sadrži najnižu adresu procesa;

## ■ limit registar

- ☞ sadrži maksimalan opseg logičkih adresa procesa
- ☞ svaka logička adresa **treba biti manja** od limit registra.

# Hardverska podrška za relokacione i limit registre



# Kontinualna alokacija (MFT)

- **MFT = Multiprogramming with Fixed Partitions**
  - ☞ (fiksni broj zadataka)
  - ☞ Idealna za multiprogramiranje sistema sa grupnom obradom
  
- **Memorija se deli na N particija fiksne veličine**
  - ☞ {multiply input Q, single input Q}
  
- **Novi proces**
  - ☞ se smešta u najmanju moguću particiju
  - ☞ uvek postoji neiskorišćenost memorije (interna fragmentacija)

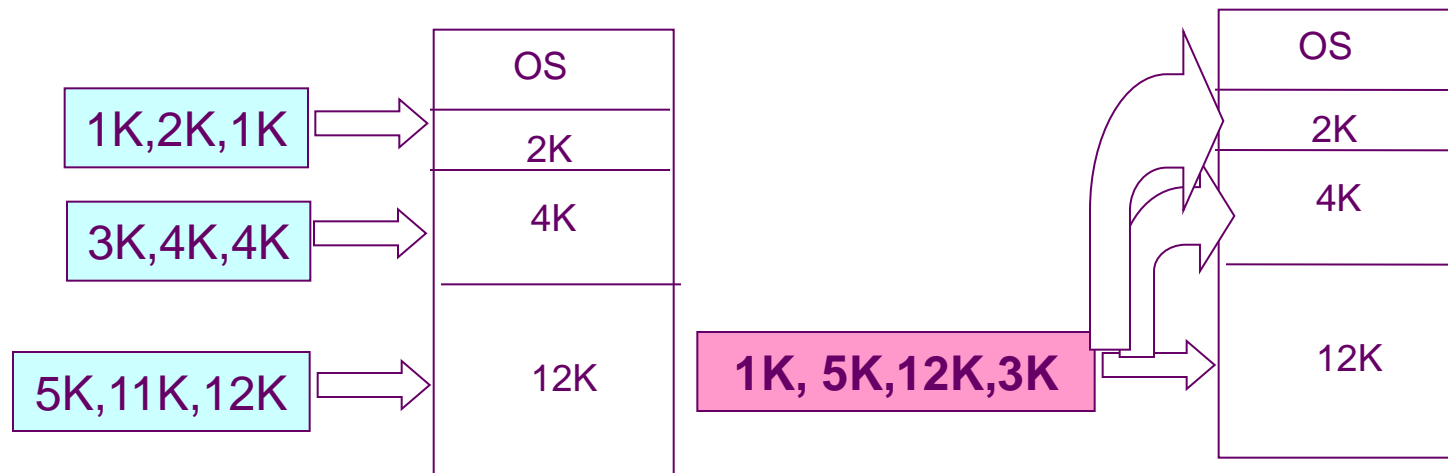
# Kontinualna alokacija (MFT)

## ■ Višestruki redovi (*Multiple Queues*)

- ☞ mali neiskorišćen prostor
- ☞ procesi mogu čekati, dok su velike particije neiskorištene

## ■ Jedinствeni red (*Single Queue*)

- ☞ bolja iskorišćenost particija
- ☞ **veliki neiskorišćen prostor** (mali procesi u velikim particijama)



# Kontinualna alokacija (MFT)

- **U sistemima sa deljenjem vremena** (time-sharing)
  - ☞ **više korisnika**
  - ☞ **više procesa**
  - ☞ **zahtevaju više od raspoložive memorije** (swapping)
- **U sistemima sa deljenjem vremena dolazni procesi su:**
  - ☞ **previše mali ili previše veliki**
  - ☞ **menjaju se u vremenu** (promenljive veličine)
  - ☞ **procesi mogu da rastu u vremenu** (podaci i stek)
- **Obe osobine su nepovoljna za MFT=>**
  - ☞ **mного neiskorišćenog prostora u particijama => novi MVT**
- **MFT je neupotrebljiv;**
  - ☞ **=> novi MVT**

# Kontinualna alokacija (MVT)

## ■ Multiple-partition allocation

☞ šupljina (hole): slobodni kontinualni deo memorije

☞ šupljine raznih veličina su **razbacane po memoriji**

☞ Kad **proces** naiđe,

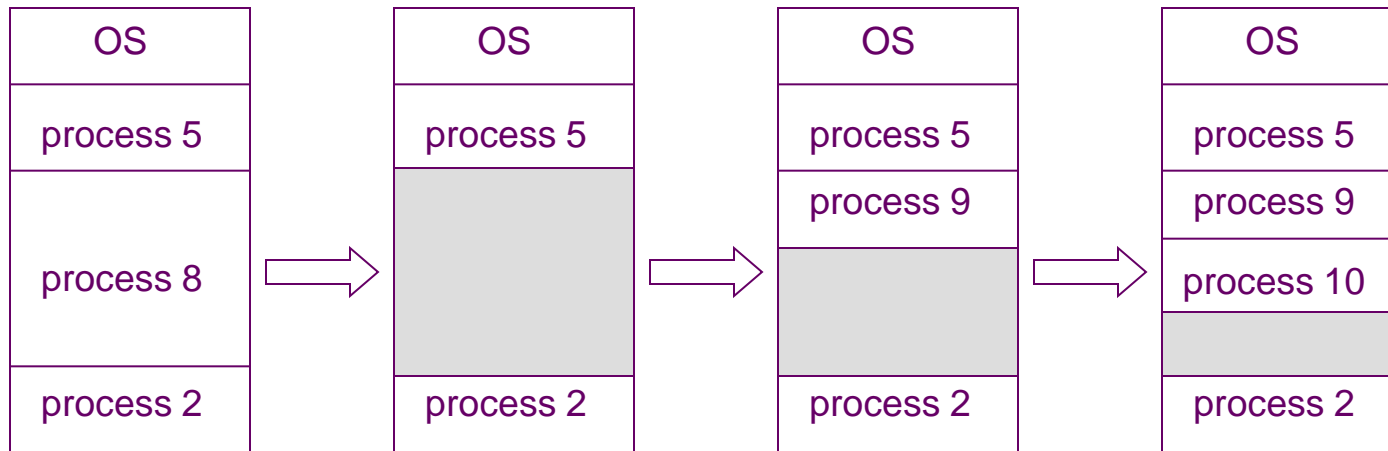
📄 traži se **šupljina** dovoljno velika

📄 da preuzme proces

☞ **Operativni sistem vodi evidenciju o:**

**a) alociranim particijama**

**b) slobodnim particijama (šupljine)**



# Upravljanje memorijom (slobodna lista)

## ■ Bit mape (*Bit Maps*)

- ☞ Alocirani delovi memorije + bit mapa (0=free, 1=allocated)
- ☞ Ako je alocirani deo manji => bit mapa je veća

## ■ Povezane liste (*Linked Lists*)

- ☞ Povezana lista slogova = Proces (Šupljina) { start, dužina}

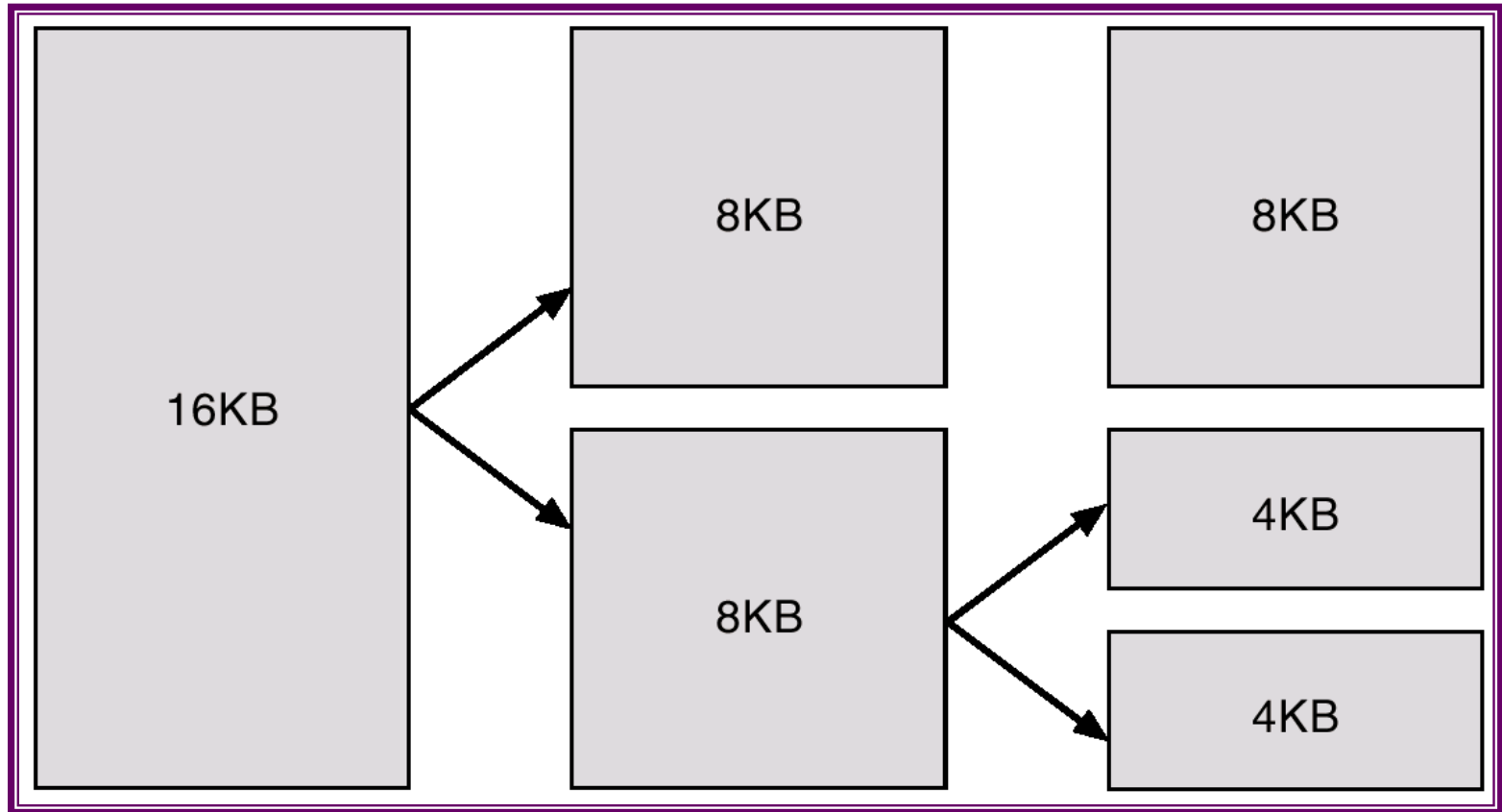


- ☞ Zasebne liste za šupljine sa istim veličinama => tabela sa N ulaza:
- ☞ traženje šupljine je brzo, ali je spajanje šupljina veoma teško

## ■ Sistem udruženih parova (*Buddy Systems*)

- ☞ MM koristi listu slobodnih blokova veličine 1, 2, 4, 8, 16, 32 ....
  - 📄 (za 1MB = 21 liste)
- ☞ Cilj= $2^k$ , svi procesi trebaju imati veličinu približno  $2^k$
- ☞ brzo pretraživanje i spajanje šupljina
- ☞ ali je dominantna neiskorišćenost prostora

# Sistem udruženih parova (*Buddy Systems*)



# Algoritmi za izbor prazne particije

Postoji više **algoritama za dodeljivanje slobodnih šupljina**

## ■ 1. First-fit:

☞ Procesu se alocira **prva šupljina** koja je dovoljno velika

## ■ 2. Best-fit:

☞ Alocira se **najmanja šupljina** koja je dovoljno velika za proces;

☞ **pretražuju se cela lista**, osim ako nije **sređene po veličini**

☞ **u ostatku ostavlja najmanju moguću šupljinu**

☞ => ideja za worst fit

## ■ 3. Worst-fit:

☞ Procesu se dodeljuje **najveća moguća šupljina**;

☞ **opet se pretražuje cela lista**

☞ **u ostatku ostavlja najveću moguću šupljinu**

**First-fit i best-fit su bolji i po performansama i po iskorišćenju memorije od worst-fit.**

# Fragmentacija

- **Eksterna fragmentacija** – ukupan memorijski prostor može da zadovolji zahtev procesa **ali nije kontinualan**
- **Interna fragmentacija** – alocirana memorija može biti veća od zahtevane memorije procesa, ono što je preostalo od alocirane memorije **se ne koristi za ništa**.
- **Eksterna fragmentacija se može smanjiti sažimanjem**
  - ☞ Cilj sažimanja je da se ispremešta sadržaj memorije kako bi se dobile veće šupljine
  - ☞ Sažimanje je moguće *samo* ako je relokacija programa dinamička i radi se u vreme izvršavanja
  - ☞ **I/O problem**
    - 📄 opadaju performanse sistema, jer se procesi prekidaju,
    - 📄 i privremeno prebacuju na disk

# Diskontinualna alokacija

- **Straničenje (*Paging*)**
- **Segmentacija (*Segmentation*)**
- **Kombinacija:**
  - ☞ **Straničenje** sa **segmentacijom**
  - ☞ **Segmentacija** sa **straničenjem**

# Straničenje (*Paging*)

- **Logički** adresni prostor procesa **mora biti kontinualan**;
- ali
- **fizički adresni prostor procesa ne mora biti kontinualan**;
- fizička memorija je dodeljena procesu kad god je raspoloživa
- **Fizička memorija se podeli na blokove fiksne veličine** koji se nazivaju **okviri (*frames*)**
  - ☞ (veličina je stepen broja 2, između 512 B i 8192 B)
- **Logički adresni prostor se podeli na blokove iste veličine** koje se nazivaju **stranice (*pages*)**
- **Za pokretanje programa veličine *n* stranica:**
  - ☞ potrebno je pronaći *n* slobodnih okvira
  - ☞ učitati program
- Tabela stranica (*page table*) prevodi logičke u fizičke adrese
- **Interna fragmentacija**

# Straničenje (2)

## ■ Adresa koju generiše CPU podeljena je na:

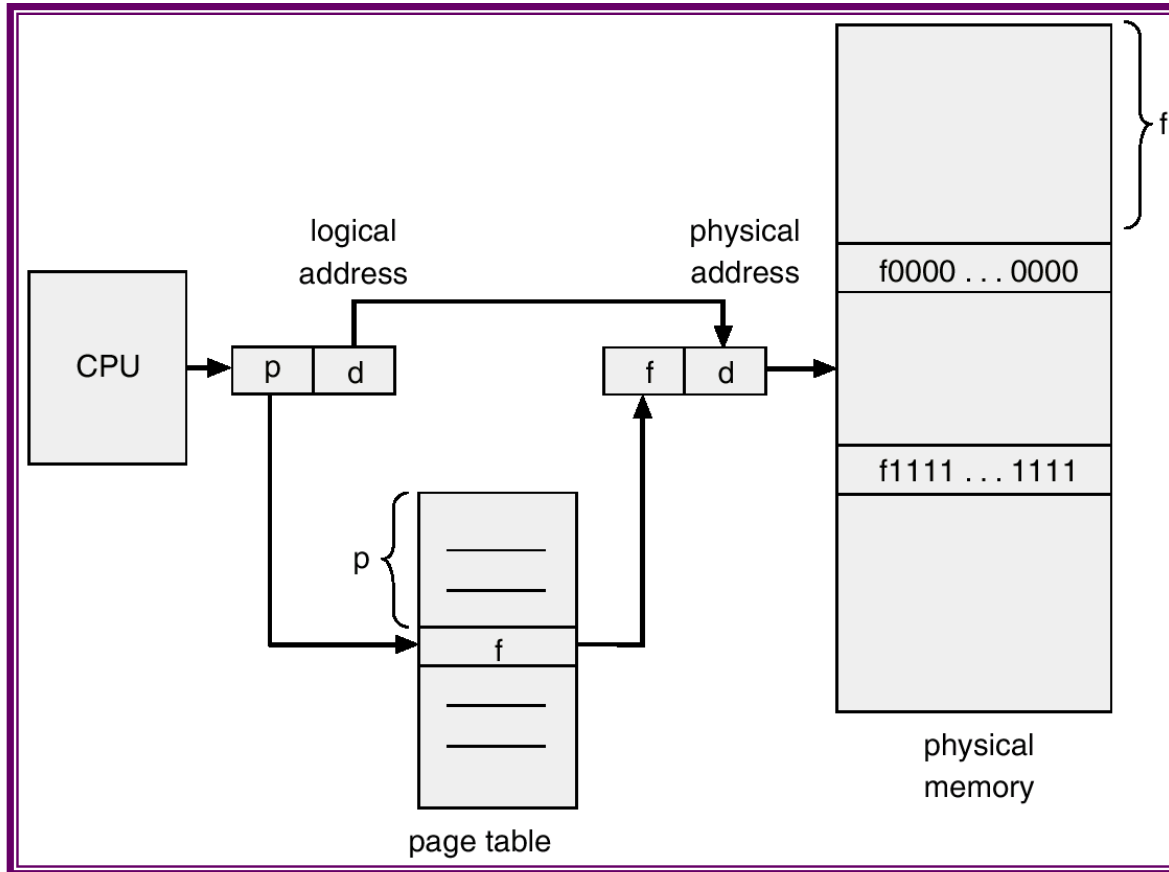
### ☞ Broj stranice (*Page number*) ( $p$ ):

- ☞ koristi se kao indeks u tabeli stranica
- ☞ koja sadrži
- ☞ baznu adresu fizičke stranice, okvira

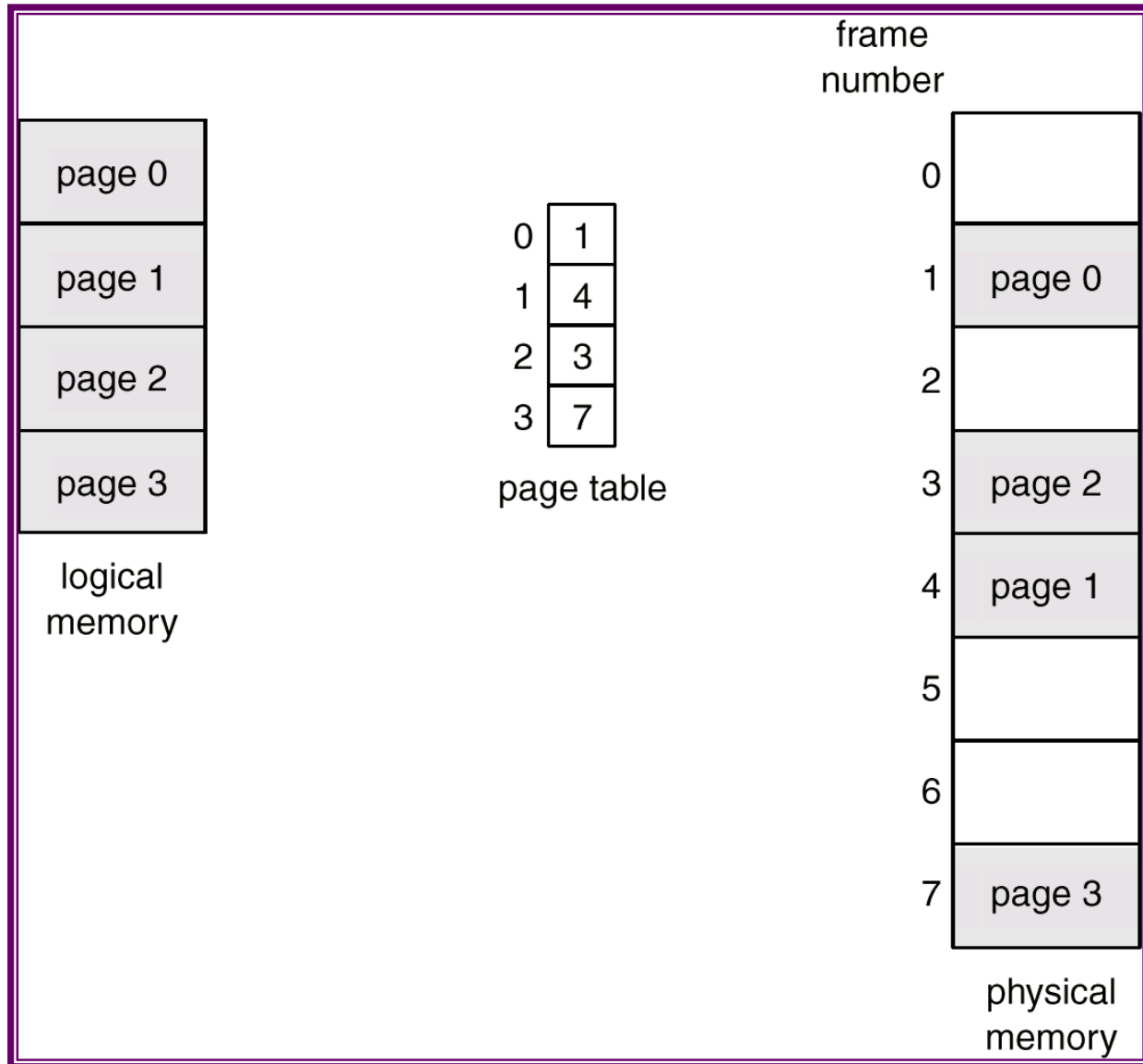
### ☞ Pomeraj unutar stranice (*Page offset*) ( $d$ ):

- ☞ u kombinaciji sa baznom adresom
- ☞ definiše
- ☞ punu fizičku adresu
- ☞ koja se šalje memorijskoj jedinici

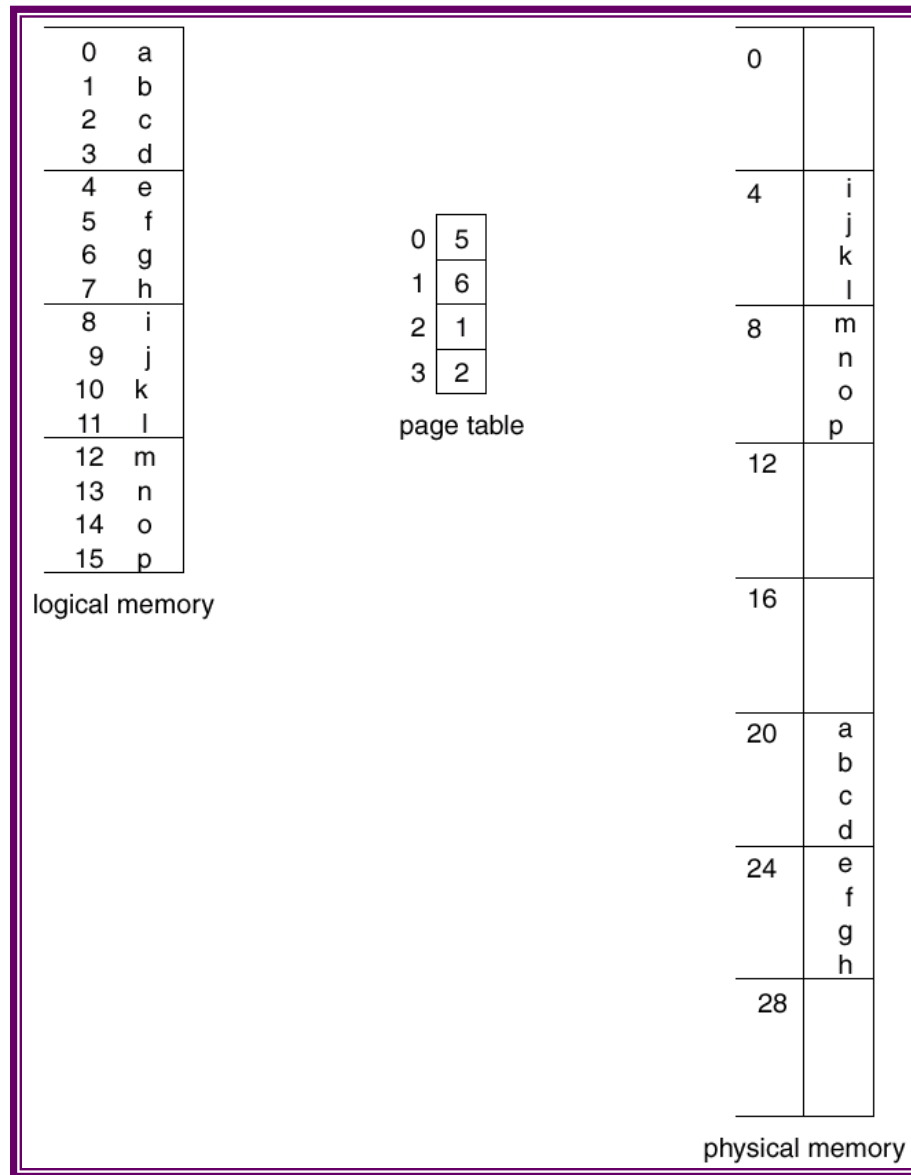
# Metoda straničenja



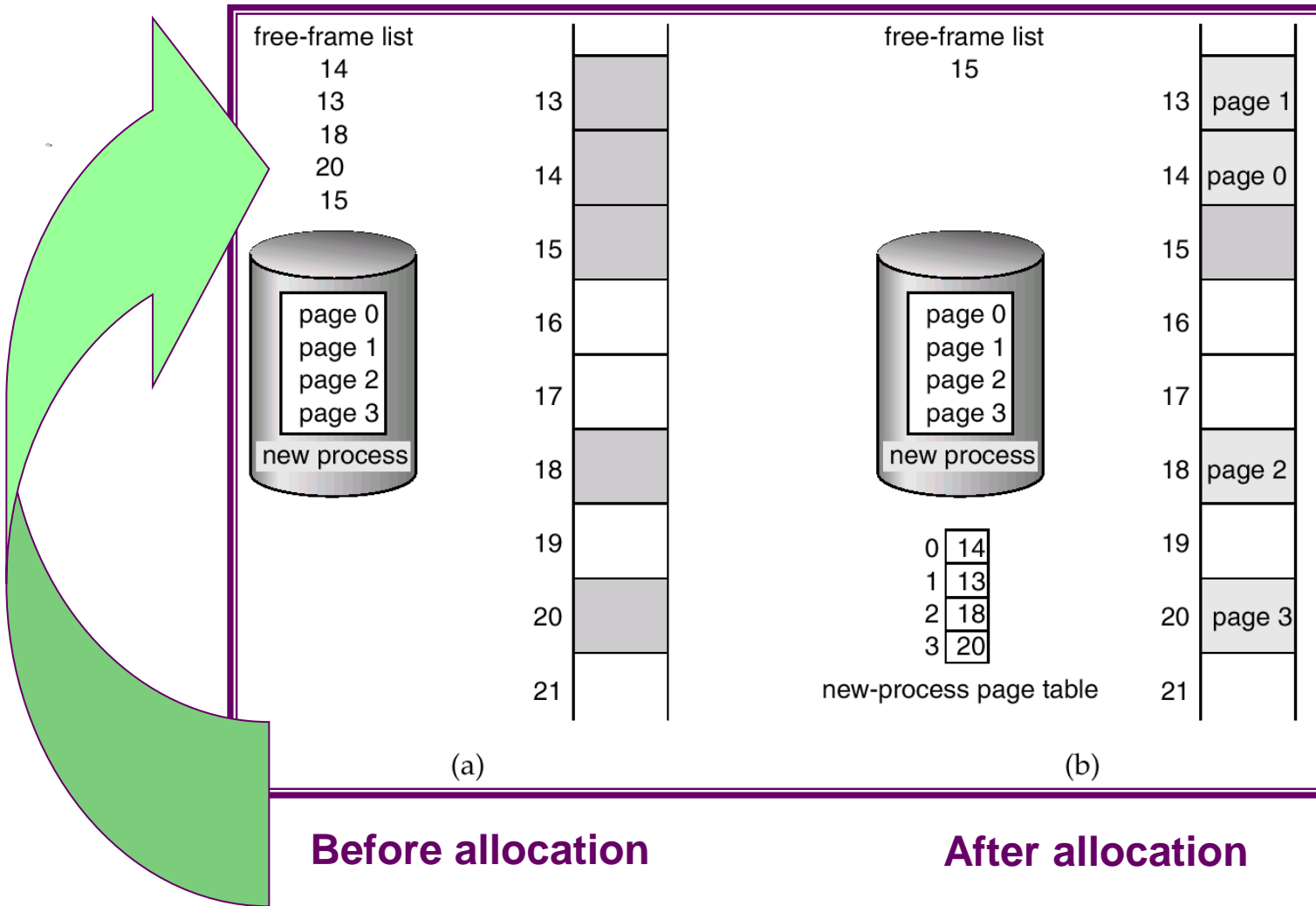
# Primer straničenja



# Primer straničenja



# Slobodni okviri (*Free Frames*)



# Implementacija tabele stranica

- **Tabela stranica** se čuva u glavnoj memoriji
- **Page-table base register (PTBR)**
  - ☞ **pokazuje** na tabelu stranica
- **Page-table length register (PRLR)**
  - ☞ ukazuje na **veličinu** tabele stranica
- U ovakvoj šemi
  - ☞ **svaka korisnička memorijska referenca**
  - ☞ **se odvija preko dve memorijske reference**
  - ☞ **jedna za tabelu stranica** i **druga za pristup željenoj referenci**
- **Problem dvostrukog pristupanja** memoriji se rešava
  - ☞ **korišćenjem specijalne keš strukture**
    - 📄 koja se naziva
  - ☞ **translation look-aside buffers (TLB)**

# Asocijativna memorija

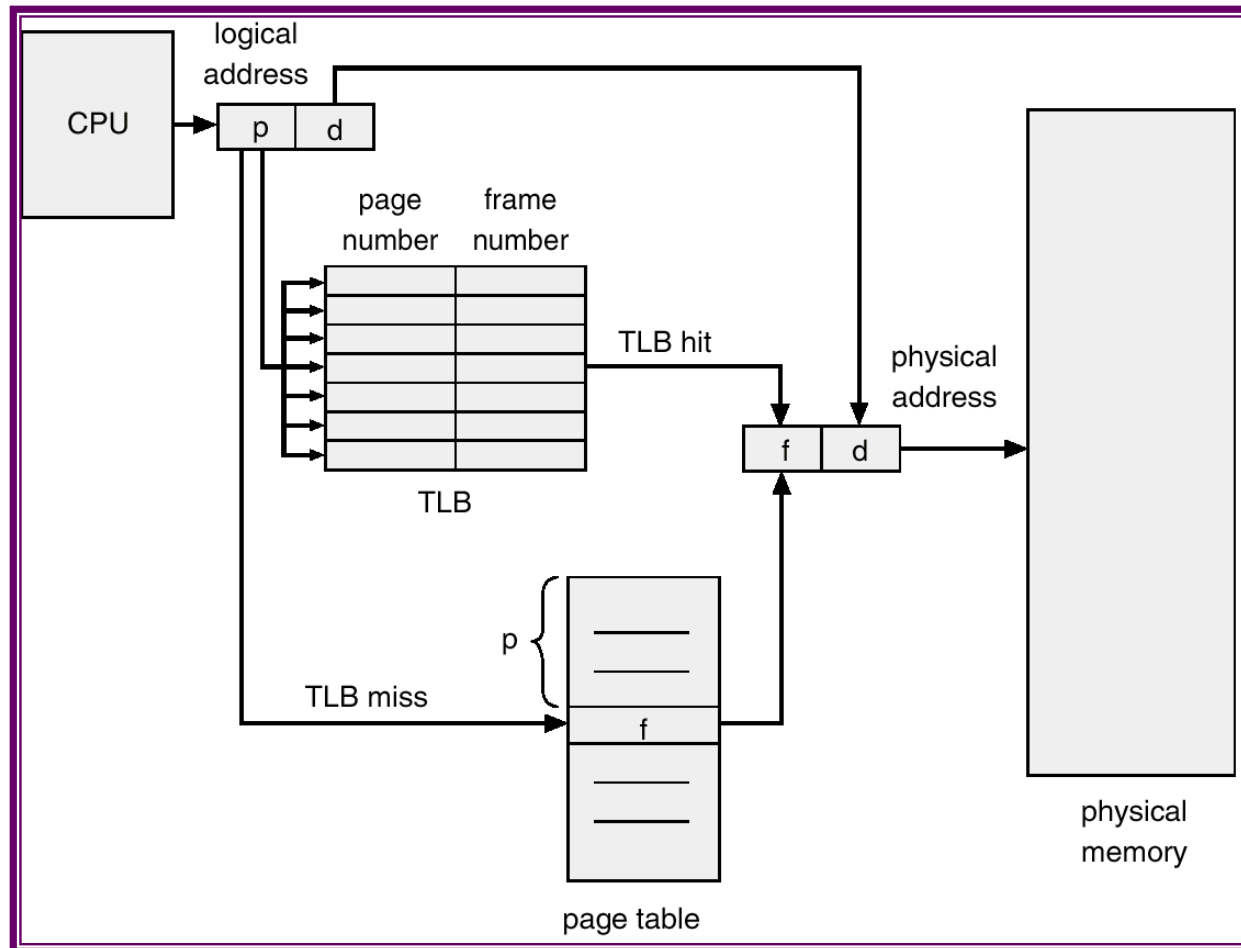
- Asocijativna memorija – paralelno pretraživanje

Page #	Frame #

## Prevođenje adresa ( $A'$ , $A''$ )

- ☞ Ako je  $A'$  u asocijativnom registru, (hit)
  - 📄 uzmi okvir # (tj  $A''$ )
- ☞ Ako nije (miss)
  - 📄 uzmi okvir #
  - 📄 iz tabele stranica u memoriji

# Hardversko straničenje sa TLB-om



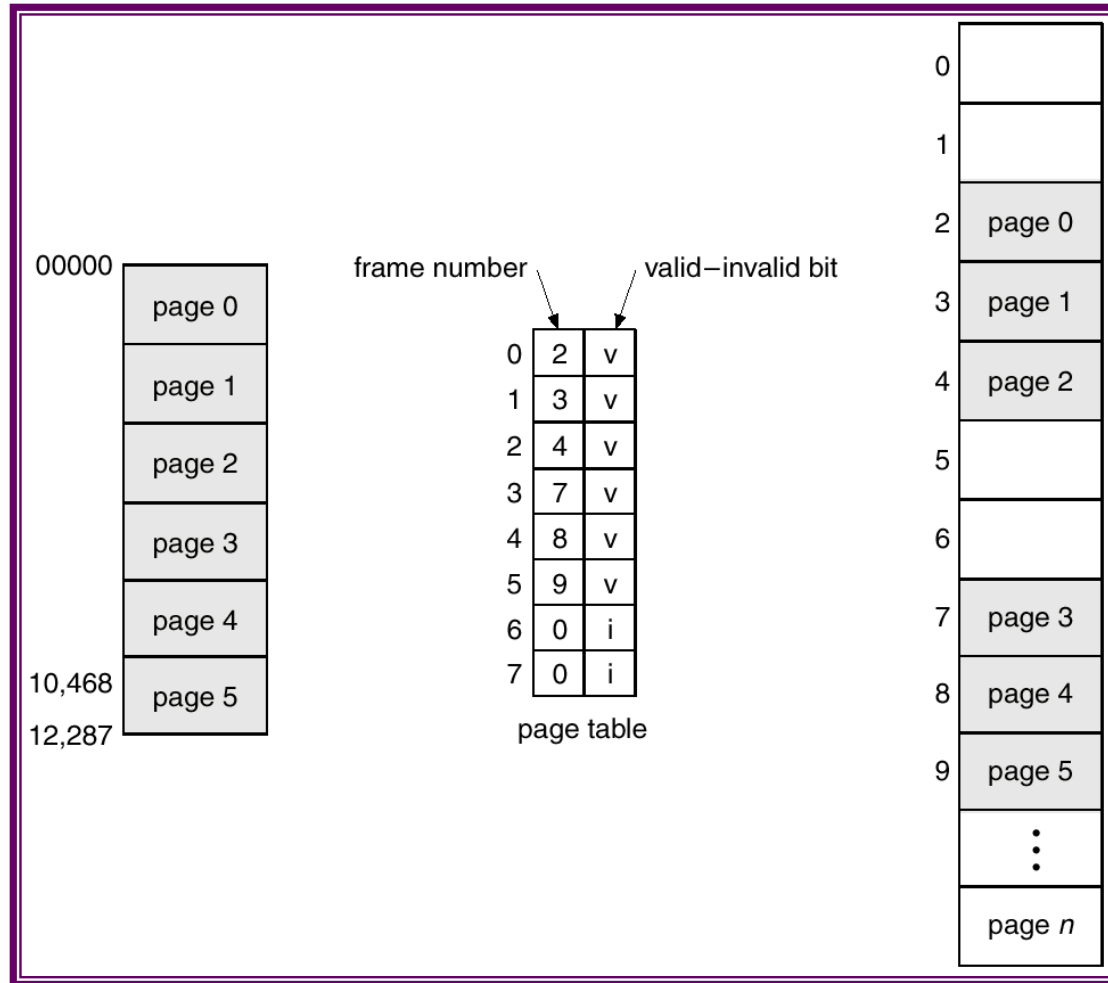
# Efektivno vreme pristupa

- **Asocijativno pretraživanje** =  $\varepsilon$  vremenskih jedinica
- Pretpostavimo da je **vreme trajanja jednog memorijskog ciklusa MC mikrosekundi**
- **Hit ratio:**
  - ☞ procenat vremena
    - 📄 vreme koje je potrebno da se pronađe broj stranice u asocijativnim registrima;
  - ☞ povezan sa brojem asocijativnih registara.
- **Hit ratio** =  $\alpha$
- **Effective Access Time (EAT)** =  $\alpha T_{hit} + (1 - \alpha) T_{miss}$ 
$$EAT = \alpha (MC + \varepsilon) + (1 - \alpha)(2MC + \varepsilon)$$
$$= 2MC + \varepsilon - \alpha MC$$

# Zaštita memorije

- **Zaštita memorije** postiže se
  - ☞ **uvođenjem bitova**
  - ☞ **sa specijalnim značenjem**
  - ☞ **uz svaki okvir**
- **Valid-invalid bit** se čuvaju u tabeli stranica:
  - ☞ **“valid”** ukazuje da je stranica
    - 📄 **u logičkom adresnom prostoru procesa,**
    - 📄 **pa je tako to važeća stranica**
  - ☞ **“invalid”** ukazuje da stranica
    - 📄 **nije u logičkom adresnom prostoru procesa**

# Valid (v) or Invalid (i) Bit In A Page Table



# Struktura tabele stranica

- Hijerarhijsko straničenje (*Hierarchical Paging*)
- Heš bazirane tabele stranica (*Hashed Page Tables*)
- Invertovane tabele stranica (*Inverted Page Tables*)

# Hijerarhijsko straničenje

- **Podeliti logički adresni prostor**

- ☞ u

- **više tabela stranica**

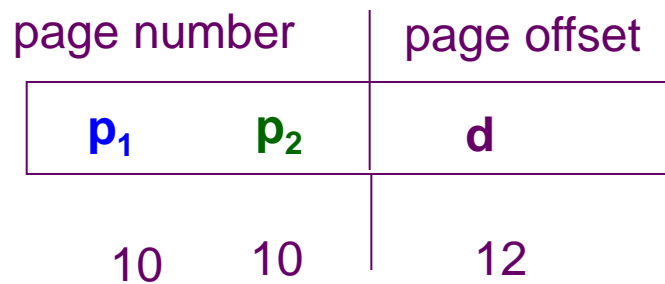
- **Jednostavna tehnika**

- ☞ je straničenje u **dva nivoa**

- ☞ (***two-level page table***)

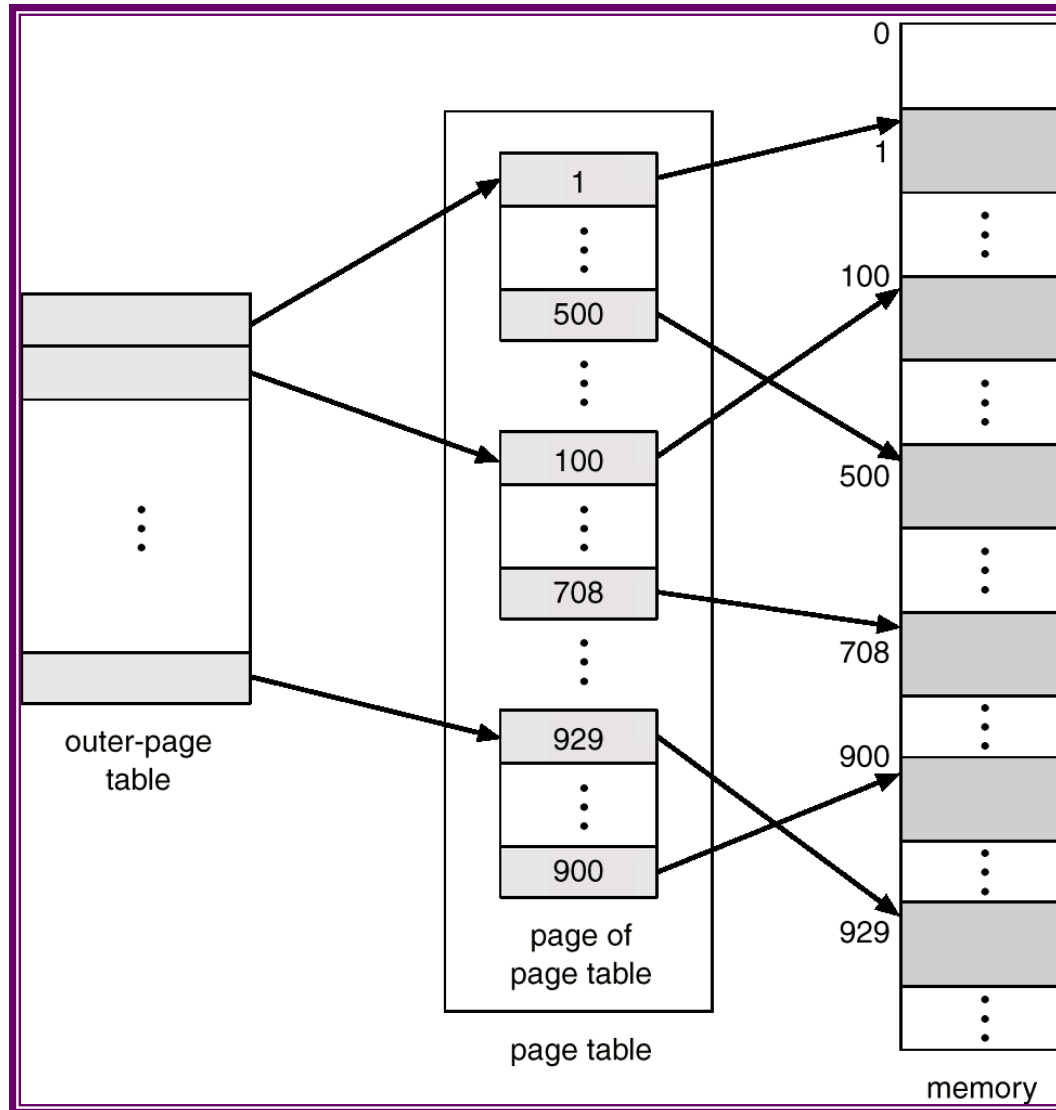
# Straničenje u dva nivoa

- **Logička adresa** (na **32-bitnoj mašini** sa **veličinom stranice 4K**) je **podeljena** na:
  - ☞ **broj stranice koji se sastoji od 20 bitova**
  - ☞ **offset stranice koji se sastoji od 12 bitova**
- **Kada se primeni tabela stranica, broj stranice se dalje deli na:**
  - ☞ **10-bitova za broj stranice**
  - ☞ **10-bitova za offset stranice**
- Iz toga sledi, **logička adresa:**



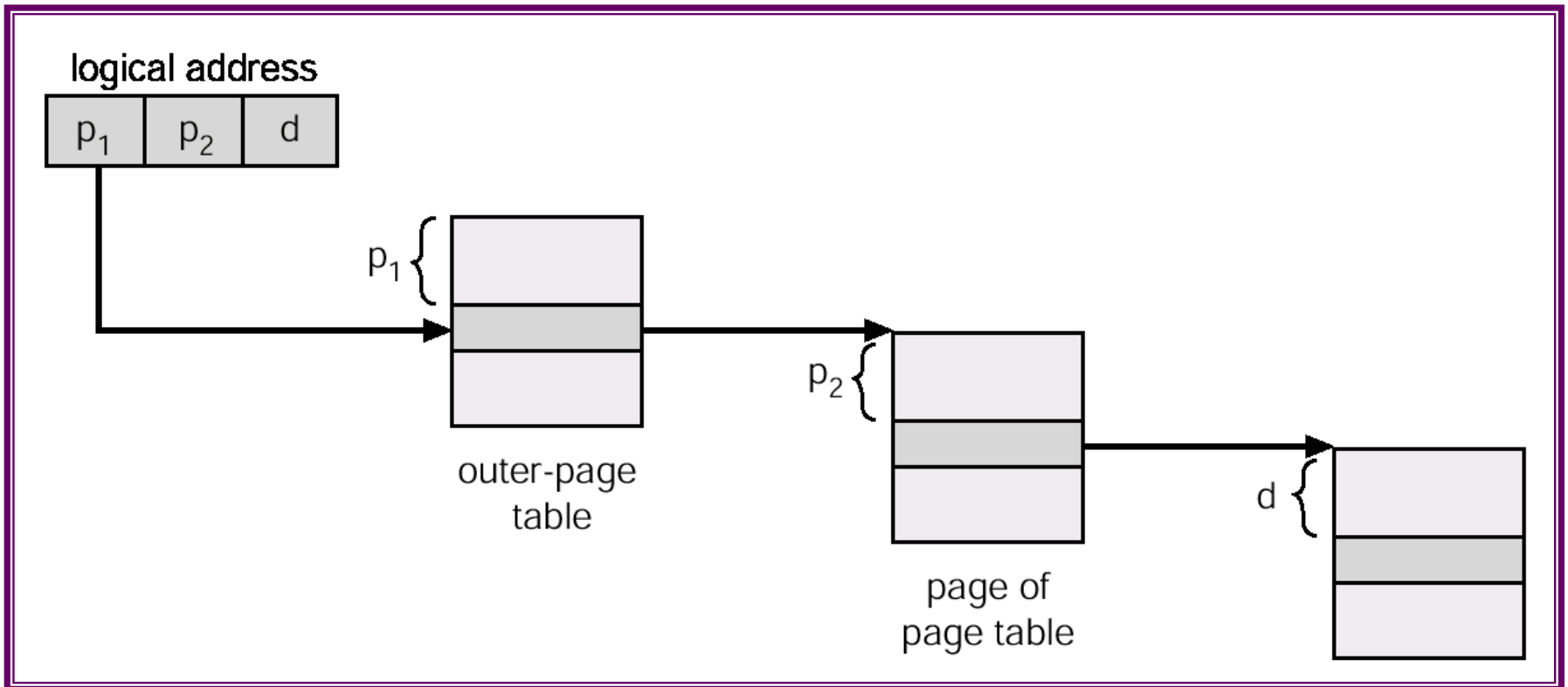
- gde je:
  - ☞  $p_1$  **indeks u spoljnoj tabeli stranica**
  - ☞  $p_2$  **je pozicija u selektovanoj untrašnjoj stranici**

# Prikaz straničenja u dva nivoa



# Šema prevođenja adresa

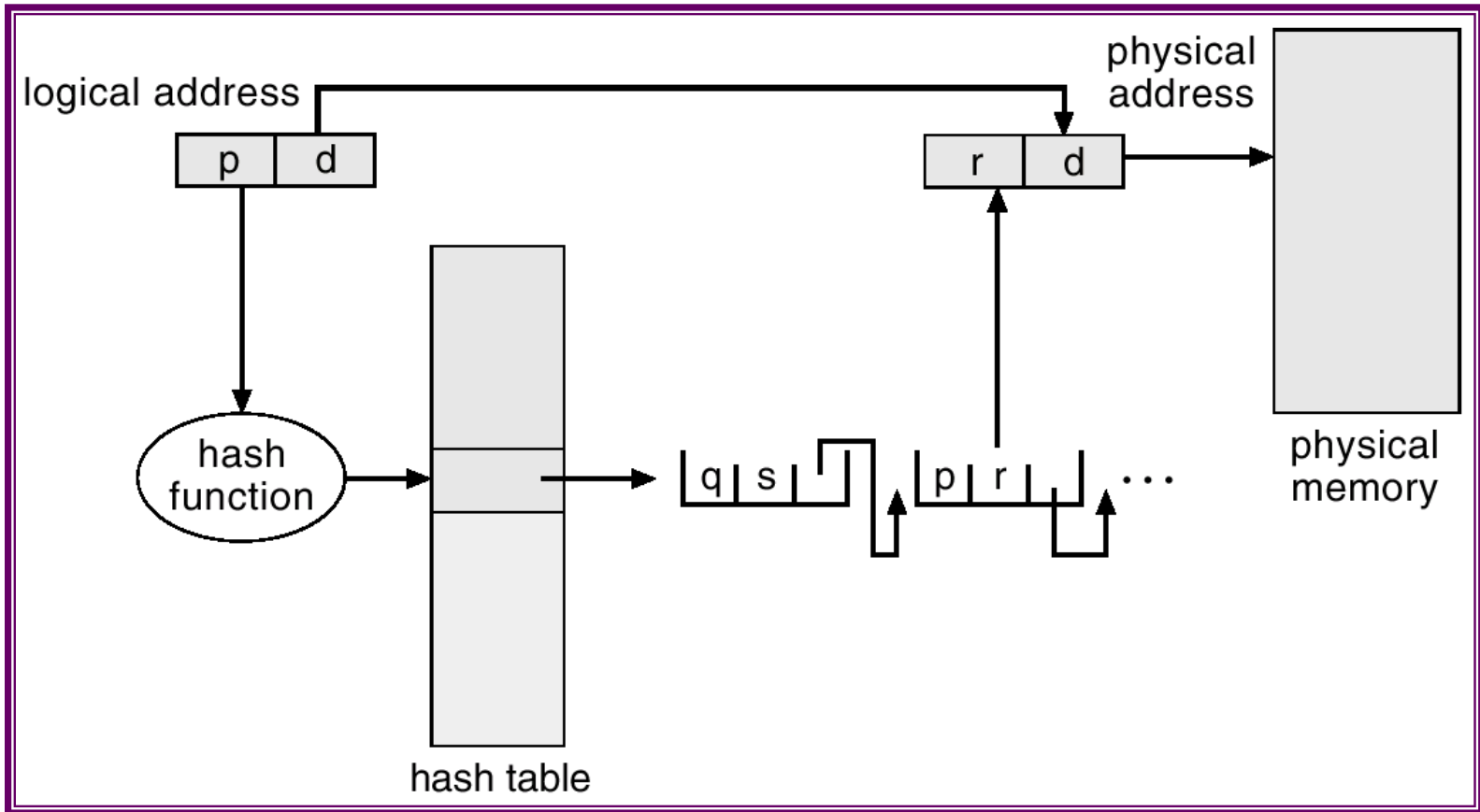
- Šema prevođenja adresa za
- straničenje u **dva nivoa** na 32-bitnoj arhitekturi



# Heš bazirane tabele stranica

- Koristi se kod **adresnih prostora** koji su **> 32 bita**
- **Virtuelni broj stranice** se **dobija iz heširane tabele stranica**
- **Ova tabela stranica sadrži:**
  - ☞ povezanu listu elemenata
  - ☞ **koji imaju istu vrednost heš funkcije.**
- **Virtuelni brojevi stranica se upoređuju**
  - ☞ **u ovom nizu**
  - ☞ **i traži se odgovarajući broj**
- **Kada se pronađe odgovarajući virtuelni broj,**
  - ☞ **dobijamo vrednost fizičke stranice**

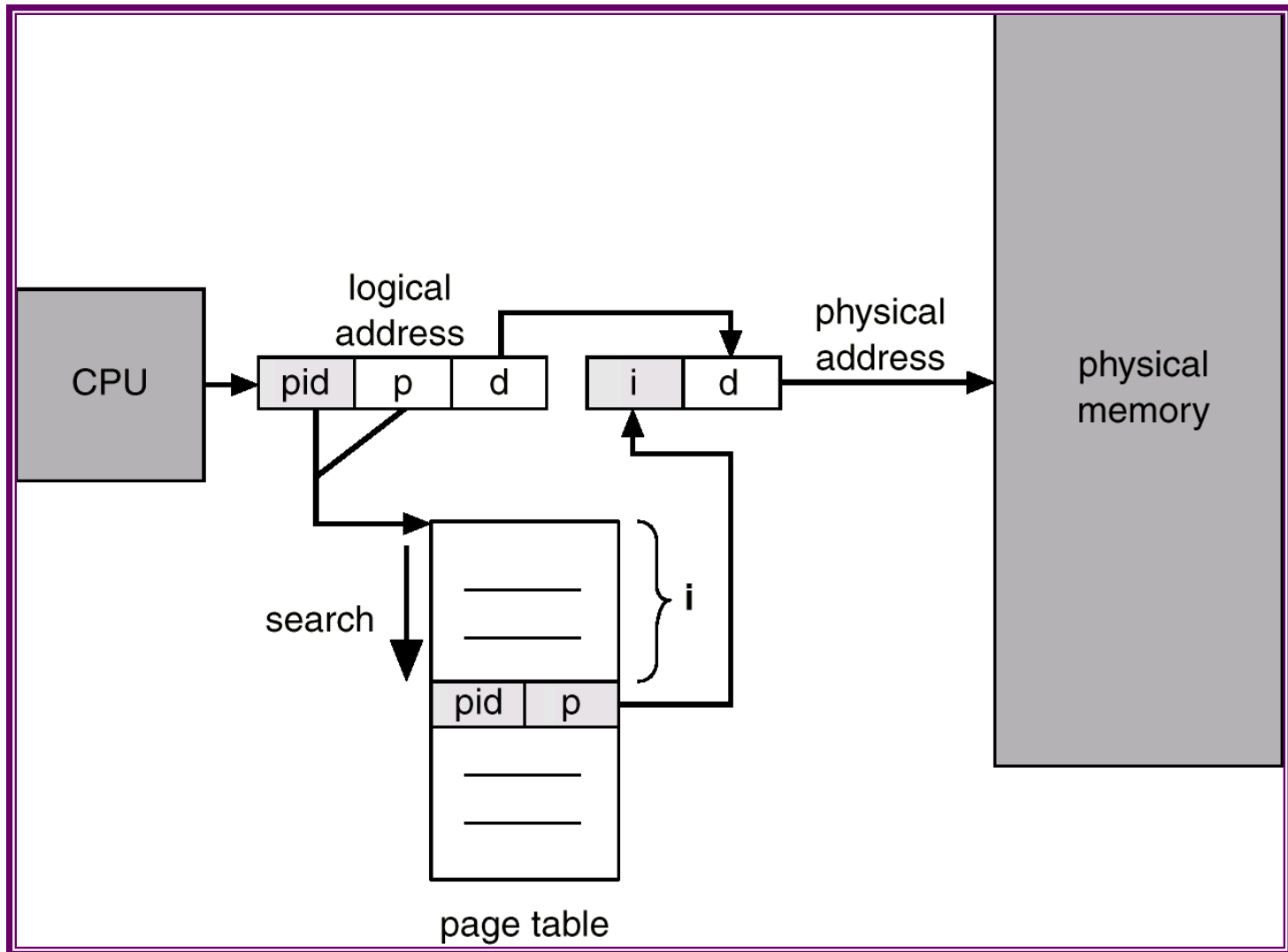
# Heš bazirane tabele stranica



# Invertovane tabele stranica

- **Jedan ulaz** za **svaki okvir ili fizičku stranicu memorije**
- **Ulaz se sastoji od:**
  - ☞ **virtuelne adrese stranice**
  - ☞ **koja je pridružena ovoj memorijskoj lokaciji,**
  - ☞ **sa informacijom o procesu koji je dobio tu stranicu PID**
- **Smanjuje se memorija potrebna da uskladišti svaku tabelu stranica,**
- **ali se**
- **povećava vreme potrebno za pretraživanje tabele**
  - ☞ **jer se pretražuje cela tabela.**
- **Zbog toga se kotisti **heš tabela** da ograniči pretraživanje**

# Invertovane tabele stranica



# Deljive stranice (*Shared Pages*)

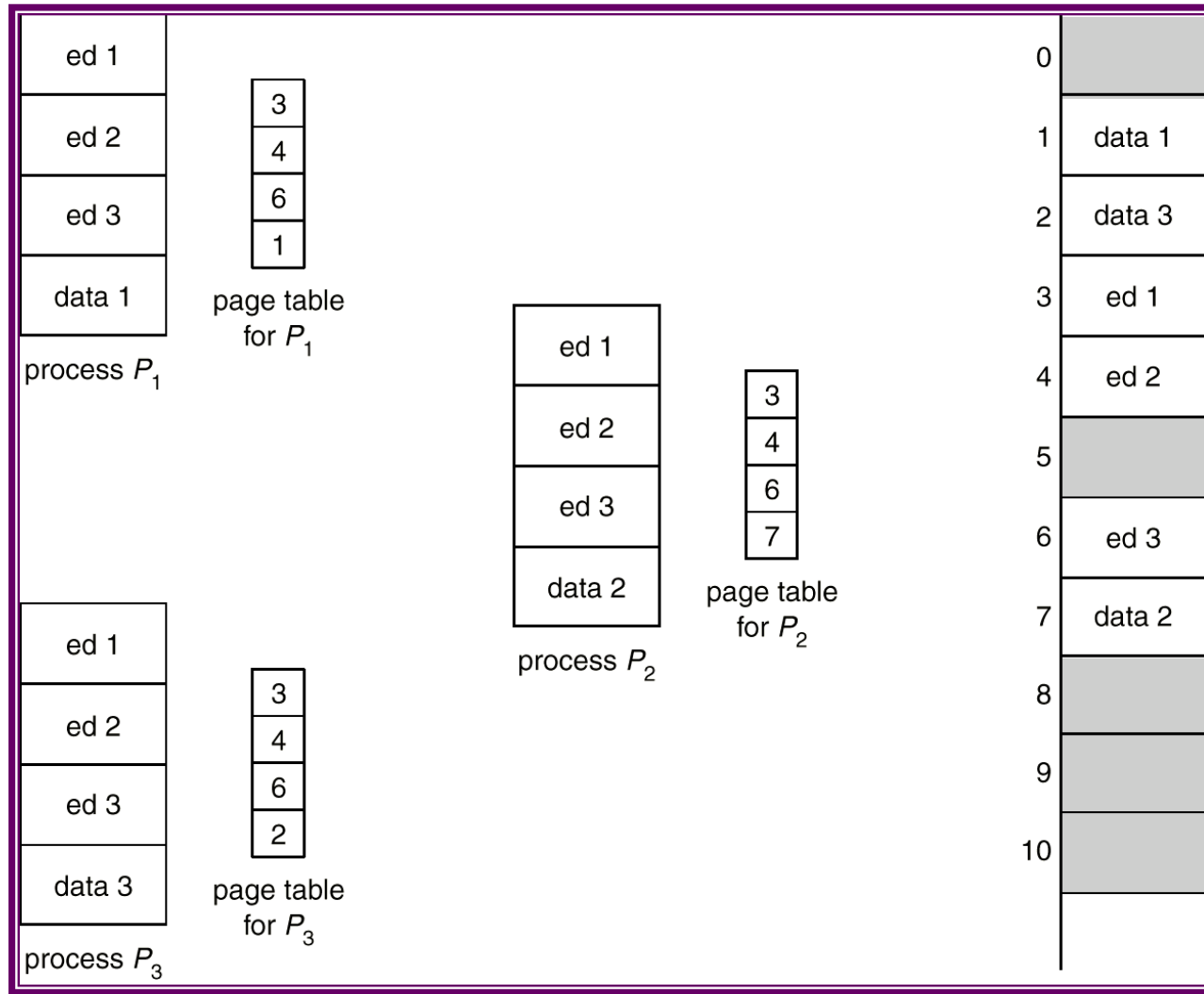
## ■ Deljivi kod

- ☞ **Jedna kopija** read-only (reentrant) koda
- ☞ **deli se između procesa** (npr., tekst editori, kompajleri, grafički sistemi)
- ☞ **deljivi kod treba biti na istoj fizičkoj lokaciji**
  - 📄 u logičkom adresnom prostoru svih procesa

## ■ Privatni kod i podaci

- ☞ **svaki proces čuva odvojenu kopiju koda i podataka**
- ☞ **stranice za privatni kod i podaci mogu biti bilo gde**
  - 📄 u logičkom adresnom prostoru.

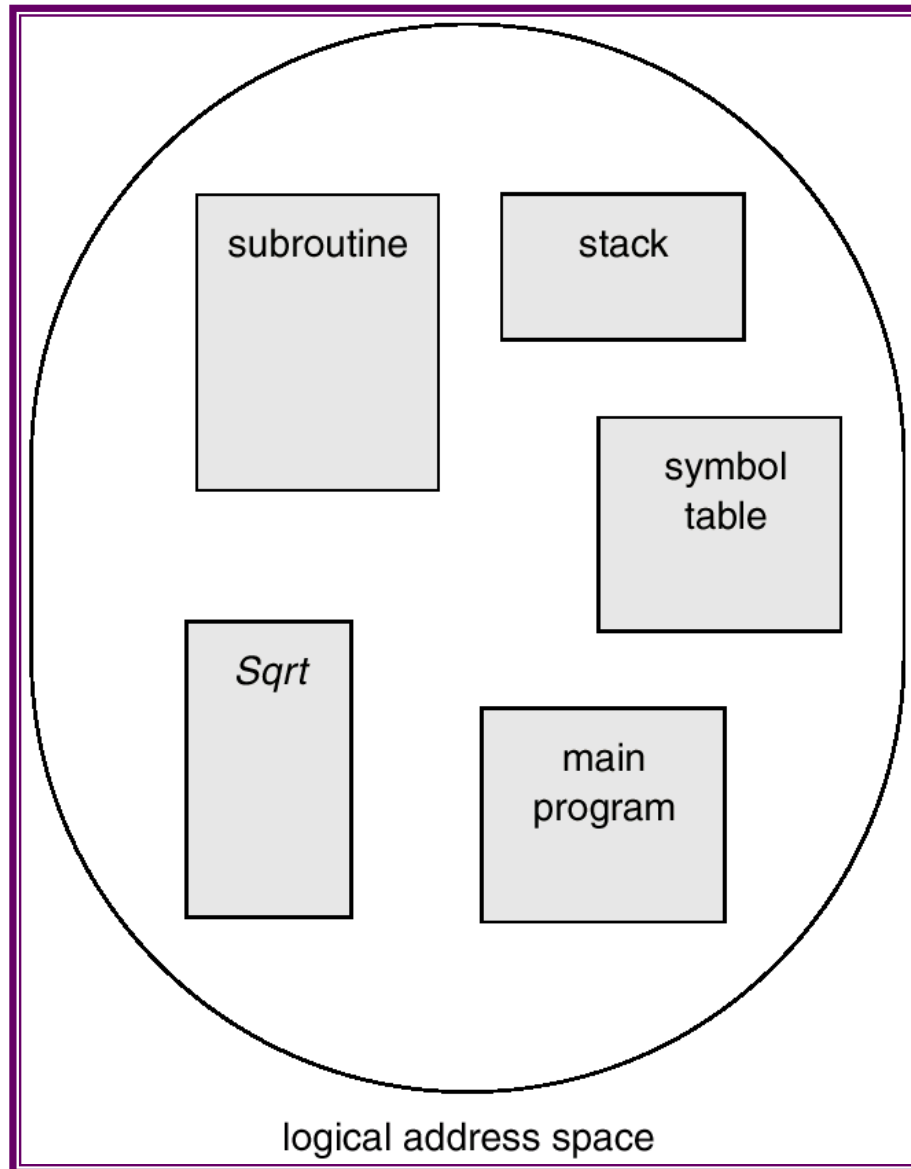
# Deljive stranice



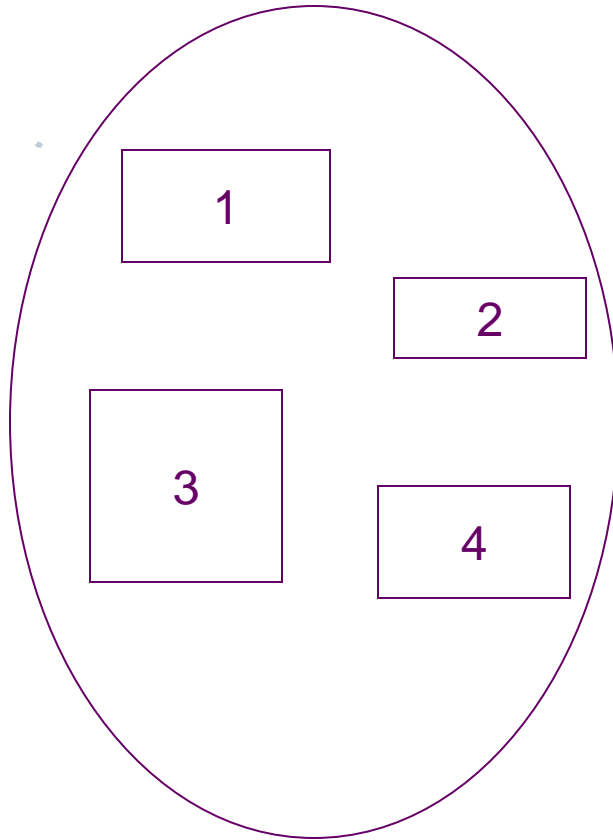
# Segmentacija

- Šema za upravljanje memorijom
  - ☞ koja podržava
  - ☞ korisnički pogled na memoriju
- Program je kolekcija segmenata
- Segment je logička jedinica kao što je:
  - glavni program
  - procedura
  - funkcija
  - metod
  - objekat
  - localne promenljive, globalne promenljive,
  - blok
  - stek
  - tabela simbola, nizovi

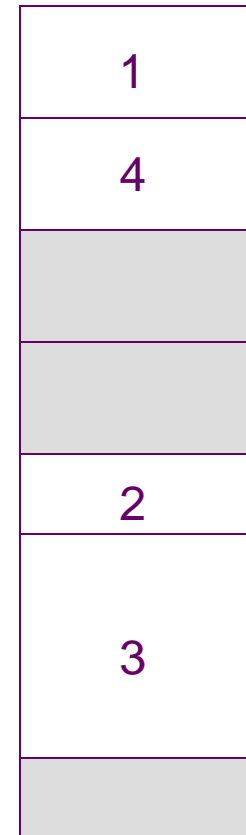
# Logički izgled programa



# Logički izgled segmentacije



user space



physical memory space

# Arhitektura segmentacije

- **Logička adresa** se sastoji od **dva dela**:  
    <**segment-number**, **offset**>
- **Tabela segmenata** – mapiranje dvodimenzionalnih fizičkih adresa;
- **Svaki ulaz u tabeli** sadrži:
  - ☞ **bazna adresa segmenta**:
    - 📄 definiše **početnu fizičku adresu segmenta** u memoriji
  - ☞ **ograničenje segmenata**:
    - 📄 definiše **dužinu segmenta**
- **Segment-table base register (STBR)**
  - ☞ pokazuje na **lokaciju u memoriji** gde je **smeštena tabela segmenata**
- **Segment-table length register (STLR)**
  - ☞ predstavlja **broj segmenata koje koristi program**;
  - ☞ broj segmenata je ispravan ako je  $s < \text{STLR}$ .

# Arhitektura segmentacije (2)

## ■ Relokacija:

- ☞ dinamičko, preko tabele segmenata

## ■ Deljenje:

- ☞ deljivi segmenti
- ☞ isti broj segmenata

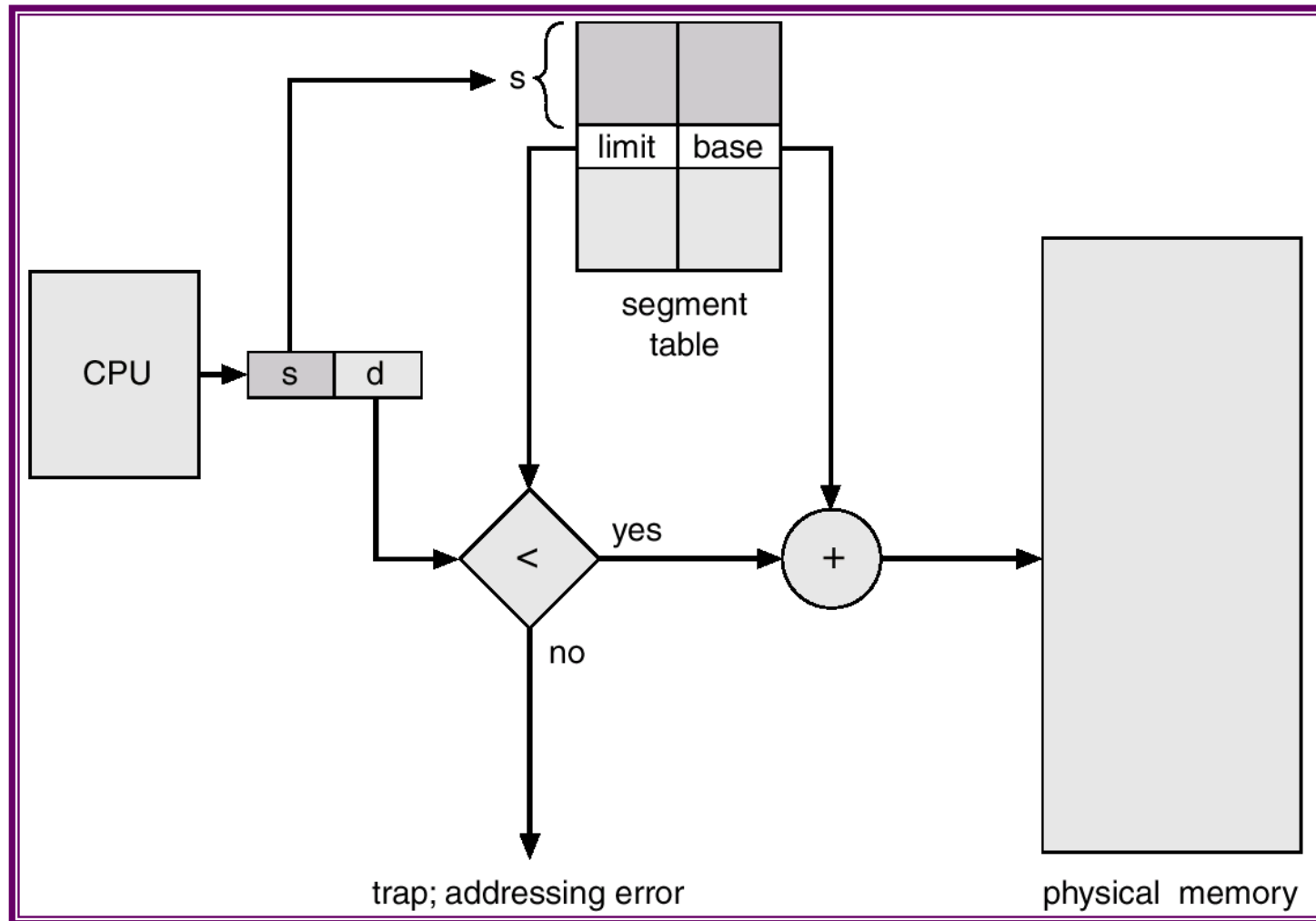
## ■ Alokacija:

- ☞ first fit/best fit
- ☞ eksterna fragmentacija

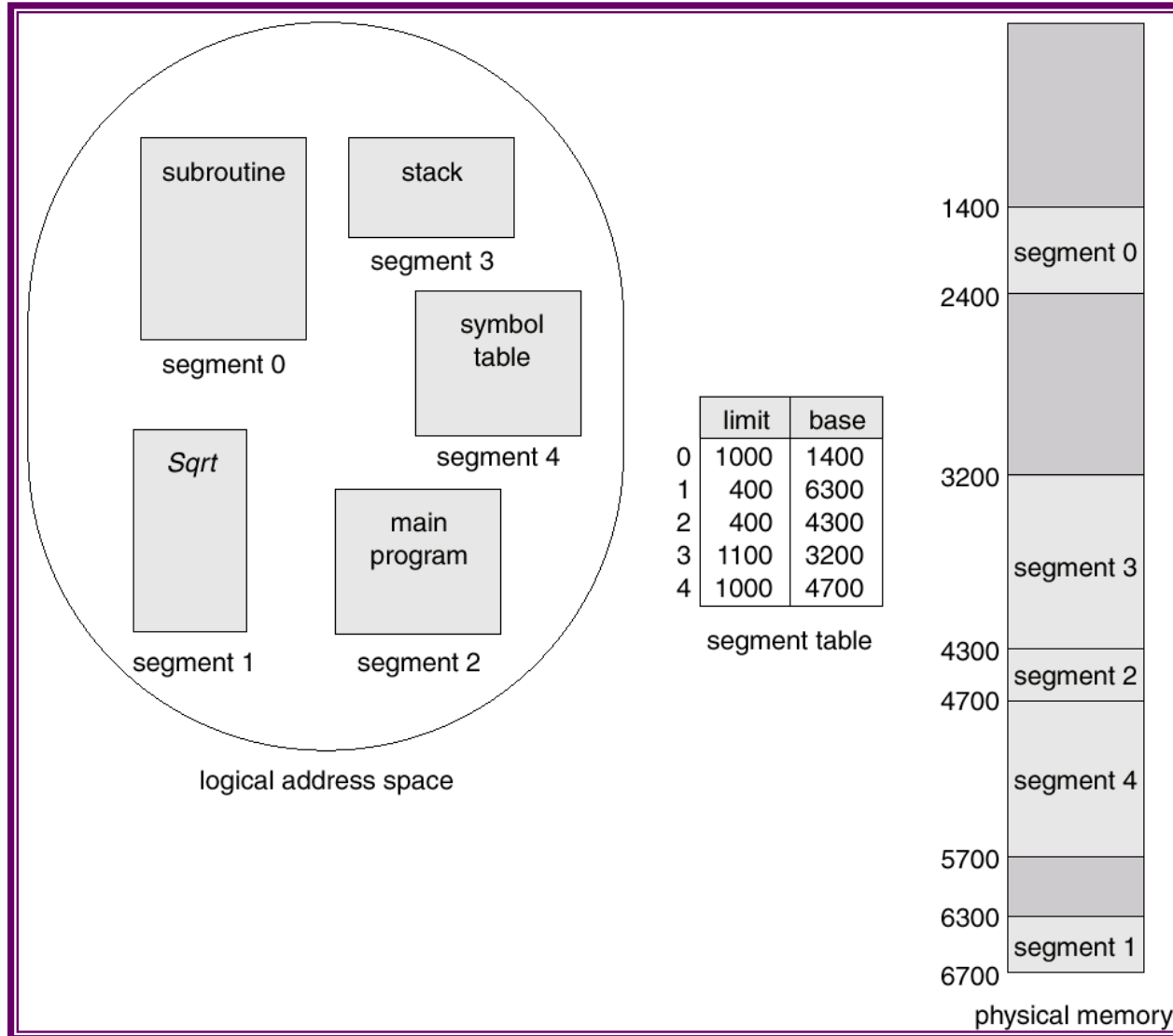
# Arhitektura segmentacije (3)

- **Zaštita.**
- Za svaki ulaz u tabelu segmenata **asocira se:**
  - ☞ **validation bit = 0** ⇒ nepravilan segment
  - ☞ **read/write/execute** privilegije
- Zaštita je povezana sa segmentima;
  - ☞ deljenje koda se odvija u nivoima segmenata.
- Pošto segmenti variraju u vremenu,
  - ☞ alokacija memorije ima problem dinamičke alokacije.
- Primer segmentacije prikazan je u sledećem dijagramu

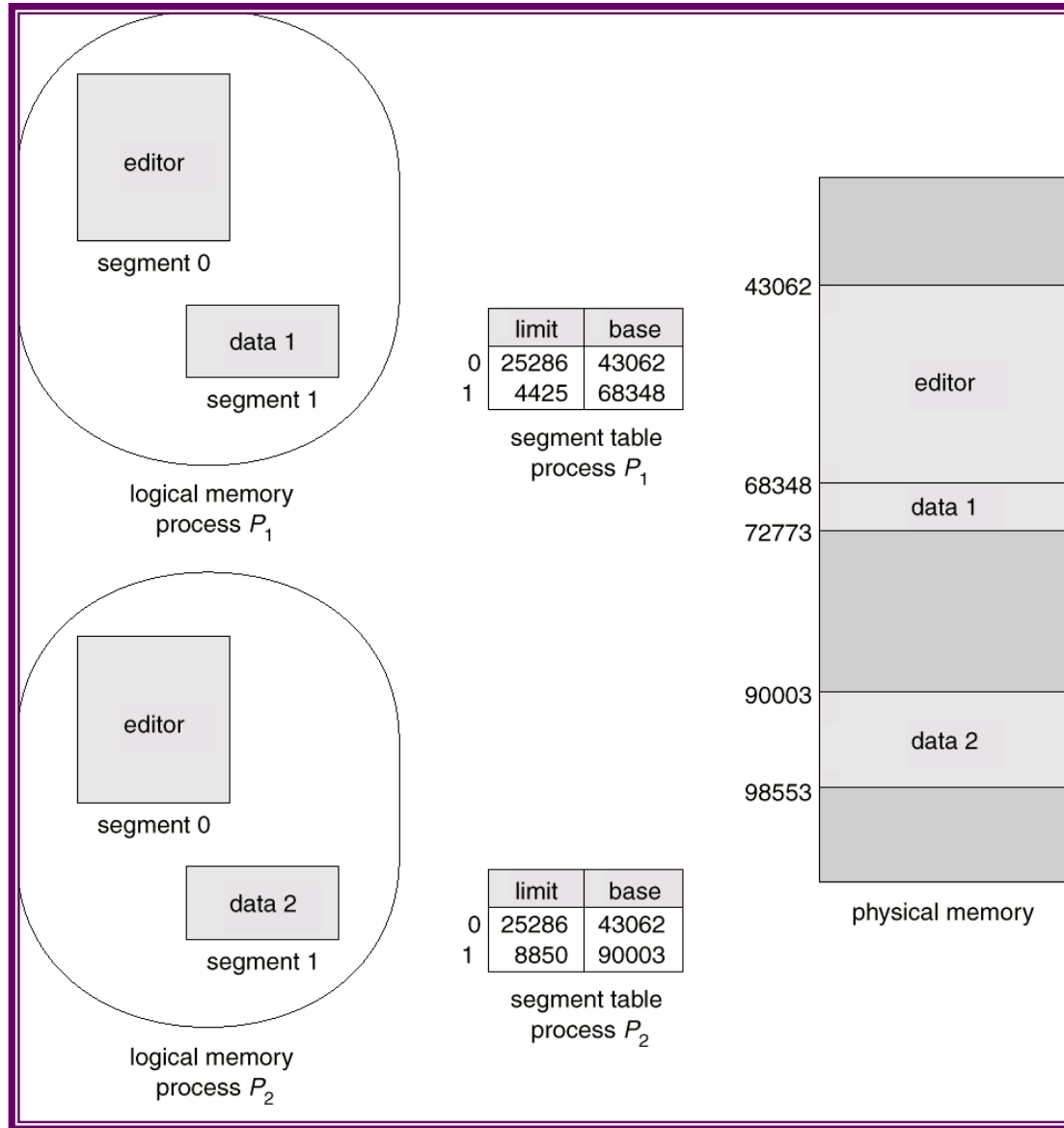
# Hardverska segmentacija



# Primer segmentacije



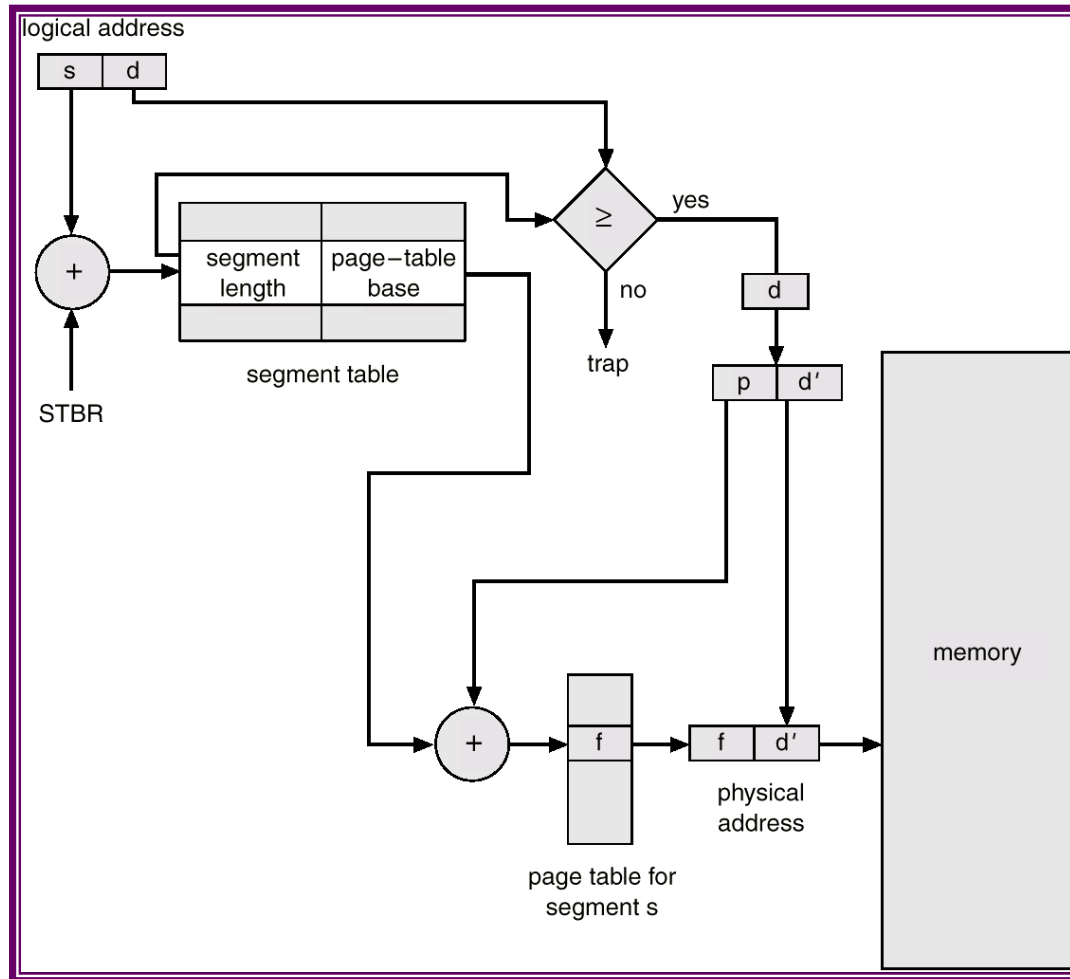
# Deljivi segmenti



# Segmentacija sa straničenjem – MULTICS

- MULTICS sistem **rešava probleme**
  - ☞ **eksterne fragmentacije i**
  - ☞ **velikog vremena za pretraživanje**
- uz pomoć **straničenja segmenata**
- **Ovo rešenje se razlikuje od čiste segmentacije**
  - ☞ u tome što se **na ulazu u tabelu segmenata**
  - ☞ **ne nalazi bazna adresa segmenta,**
  - ☞ ali
  - ☞ **se nalazi bazna adresa tabele stranica za taj segment.**

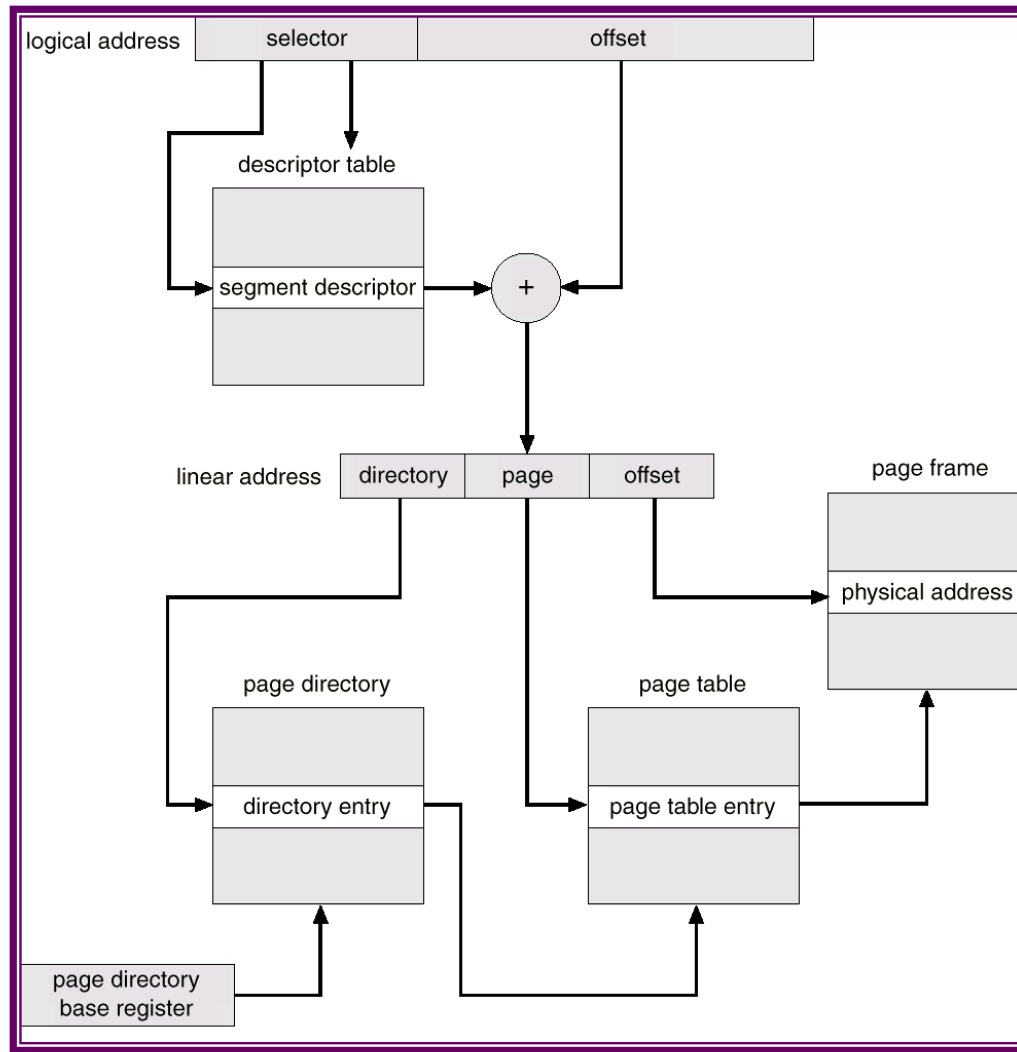
# Prevođenje adresa - MULTICS



# Segmentacija sa straničenjem – Intel 386

- Kao što je prikazano na sledećem diagramu,
- Intel 386
- koristi
- segmentaciju sa straničenjem
- sa šemom za straničenje u dva nivoa

# Prevođenje adresa - Intel 30386



# Zaključak

## ■ MM šeme

- ☞ {Bare, Resident Monitor, MFT, MVT, Straničenje, Segmentacija}

## ■ Hardverska podrška

- ☞ RM, MFT=MVT registar ili par ograničenih registara
- ☞ P&S zahteva tabele mapiranja

## ■ Performanse

- ☞ Prostije šeme su brže
- ☞ P&S potreban je hardverski akcelerator, registri, CPU-TLB

## ■ Fragmentacija

- ☞ MFT i straničenje pate od interne fragmentacije (fiksna veličina prostora)
- ☞ MVT i segmentacija pate od eksterne fragmentacije

# Zaključak

## ■ Pomeranje

- ☞ Logička adresa se može pomerati dinamično,
  - 📄 za vreme izvršavanja.
  - 📄 To je potrebno za kompakciju memorije

## ■ Swapping

- ☞ Svaki algoritam može imati ugrađen swapping

## ■ Deljenje

- ☞ deljenje obično zahteva da bude korišćemo ili straničenje ili segmentacija,
- ☞ da obezbedi male pakete informacija (stranice ili segmente)
  - 📄 koji mogu biti deljeni.

## ■ Zaštita

- ☞ valid/invalid bit
- ☞ read only, execute only, r/w