Operativni sistemi

- Upravljanje memorijom -

Prof. dr Dragan Stojanović

Katedra za računarstvo
Univerzitet u Nišu, Elektronski fakultet

Literatura

- *Operating Systems: Internals and Design Principles*, edition, W. Stallings, Pearson Education Inc., 7th – 2012, (5th -2005, 6th - 2008, 8th – 2014 , 9th – 2017)
  - http://williamstallings.com/OperatingSystems/
  - http://williamstallings.com/OperatingSystems/OS9e-Student/

- Poglavlje 7: Upravljanje memorijom

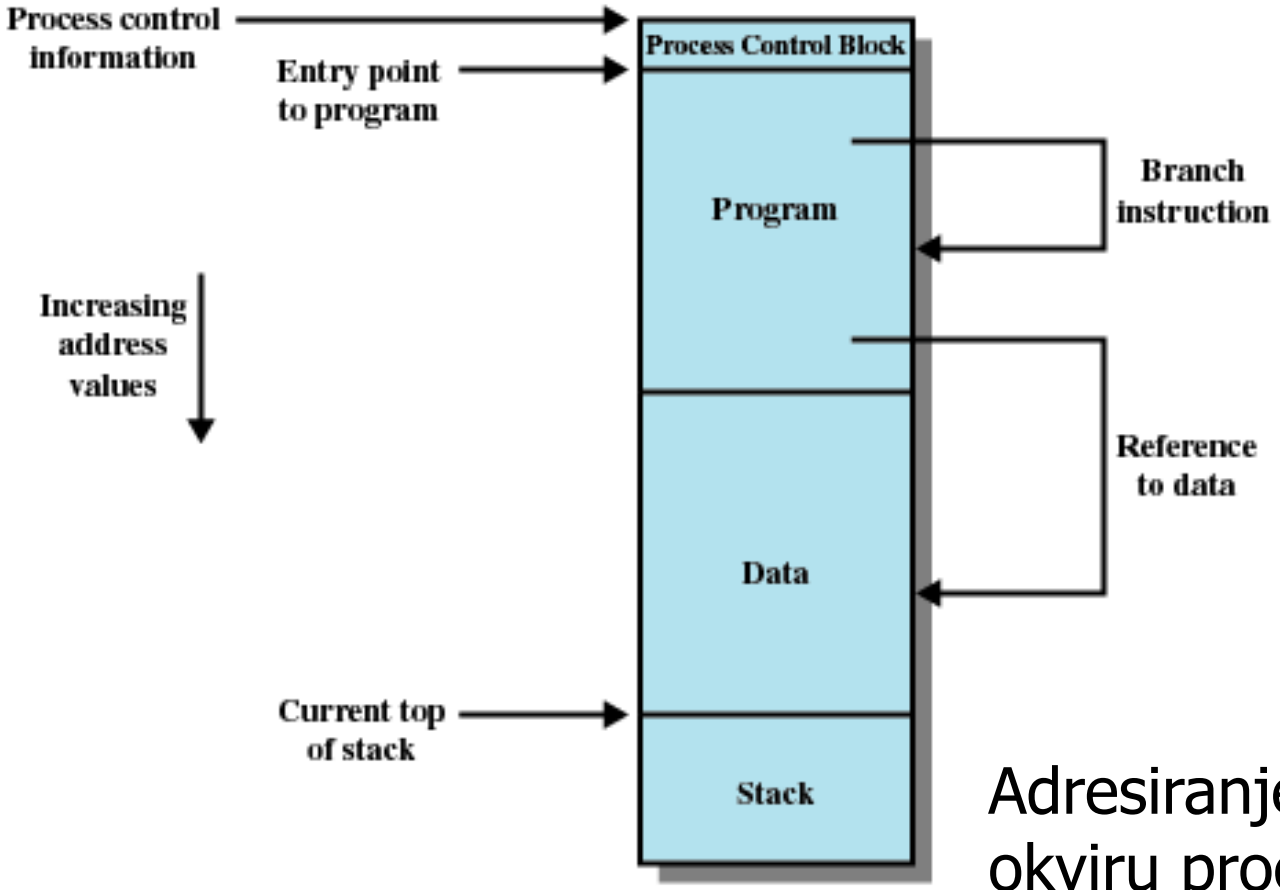
Sistem za upravljanje memorijom

- Osnovne funkcije:
  - Evidencija slobodnih i zauzetih delova memorije u multiprogramskom sistemu
  - Dodela (alociranje) memorije procesima po potrebi
    - Kontinualna dodela memorije
    - Nekontinualna dodela memorije
  - Oslobađanje (dealociranje) memorije kada procesima više nije potrebna
  - Upravljanje zamenom (*swapping*) sadržaja između glavne memorije i diska kada glavna memorija nije dovoljna da smesti sve procese

Zahtevi upravljanja memorijom

- Relokacija
  - Proces se može smestiti počev od bilo koje memorijske adrese, po potrebi premestiti na disk i ponovo vratiti u memoriju počev od nove memorijske adrese. Memorijske reference moraju da se prevedu u stvarne, fizičke memorijske adrese.
- Zaštita memorije
  - Zaštita pristupa memorijskom prostoru jednog procesa od strane drugog procesa, kao i zabrana pristupa memorijskom prostoru OS.
- Zajedničko korišćenje memorije
  - Svaki mehanizam zaštite mora biti fleksibilan da obezbedi zajedničko korišćenje istog dela glavne memorije.
- Logička organizacija
  - Linearni adresni prostor sastavljen od sekvence bajtova ili reči
  - Podela programa u module koji mogu biti kompajlirani nezavisno
- Fizička organizacija
  - Dva nivoa: glavna i sekundarna memorija

Memorijska slika procesa

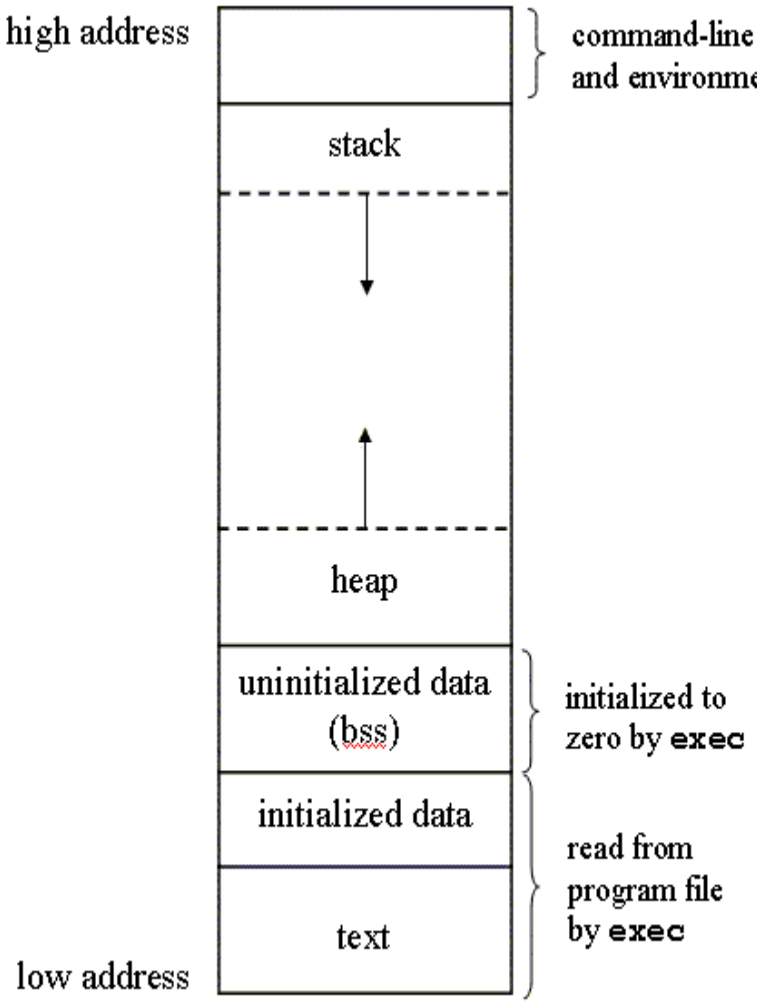


- Programski kod
- Podaci
- Stek
- PCB

Adresiranje memorije u okviru procesa

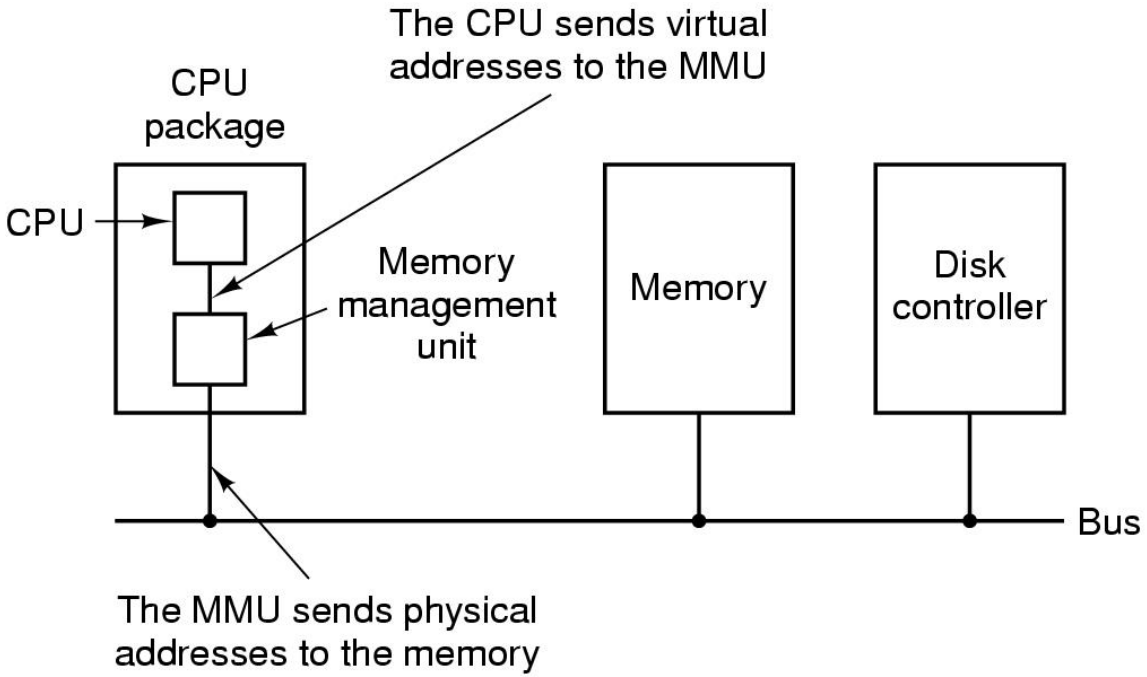
Memorijska slika procesa (detaljnije)

- Izvršna datoteka se u određenom formatu procesa (*exe, obj, elf*,...) nalazi na sekundarnoj memoriji.
- Ova datoteka sadrži programski kôd, podatke koji su neophodni za izvršenje programa i informacije koje su neophodne OS-u da bi smestio taj program u memoriju i da bi ga izvršio.
- Memorijska slika procesa
  - Programski kod (*code, text*)
  - Podaci (inicijalizovani, neinicijalizovani, dinamički (*heap*)
  - Stek



Memory Management Unit (MMU)

- MMU (*Memory Management Unit*) – hardverska komponenta, obično na CPU čipu koja vrši transformisanje logičke (virtuelne) adrese u fizičku adresu
- Korisnički procesi rade sa logičkim (virtuelnim) adresama, oni nikad "ne vide" realne, fizičke adrese

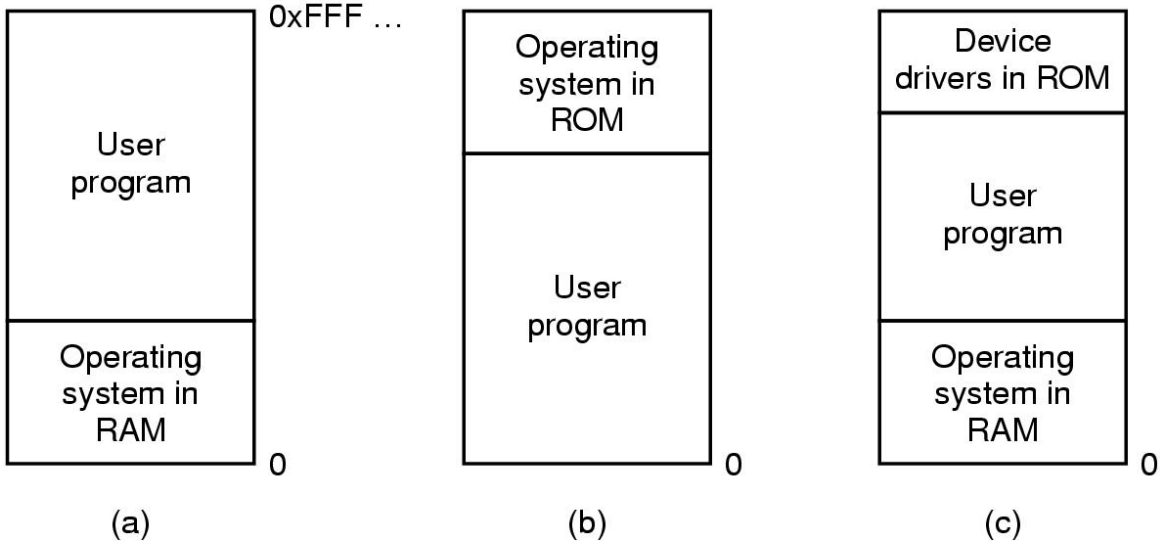


Proširenje memorije prostorom na disku

- *Overlay*
  - Neophodan kada je proces veći od veličine memorije koja mu je dodeljena
  - U memoriji su smešteni samo instrukcije i podaci procesa koji su neophodni u tekućem izvršavanju procesa
  - Nepotrebni delovi procesa (overlay-i) se prebacuju na disk, a potrebni overlay-i se smeštaju u oslobođen deo memorije
  - Aplikacioni programer je određivao delove programa koji će predstavljati overlay-e i redosled njihovog smeštanja u memoriju
  - Primer: dvo-prolazni prevodilac (asembler)
- Swap-ovanje
  - Ceo proces može biti privremeno prebačen na disk i po potrebi opet smešten u memoriju da bi se nastavilo njegovo izvršavanje
  - Mora biti obezbeđen direktan pristup swap području na disku
  - Glavni problem je vreme utrošeno sa transfer memorijskih slika procesa
  - Modifikovane verzije swapovanja mogu se naći na savremenim OS: UNIX, Linux, Windows, itd.

Monoprogramiranje

- Najjednostavnija šema za upravljanje memorijom
- Samo jedan proces (program) može da se izvršava i bude u memoriji i on deli memoriju sa operativnim sistemom
  a) Korišćen na *mainframe* računarima i mini-računarima
  b) Na nekim *palmtop* računarima i ugrađenim računarskim sistemima
  c) Na prvim personalnim računarima (MS DOS) gde je deo OS bio smešten u ROM u obliku BIOS (*Basic Input Output System*)



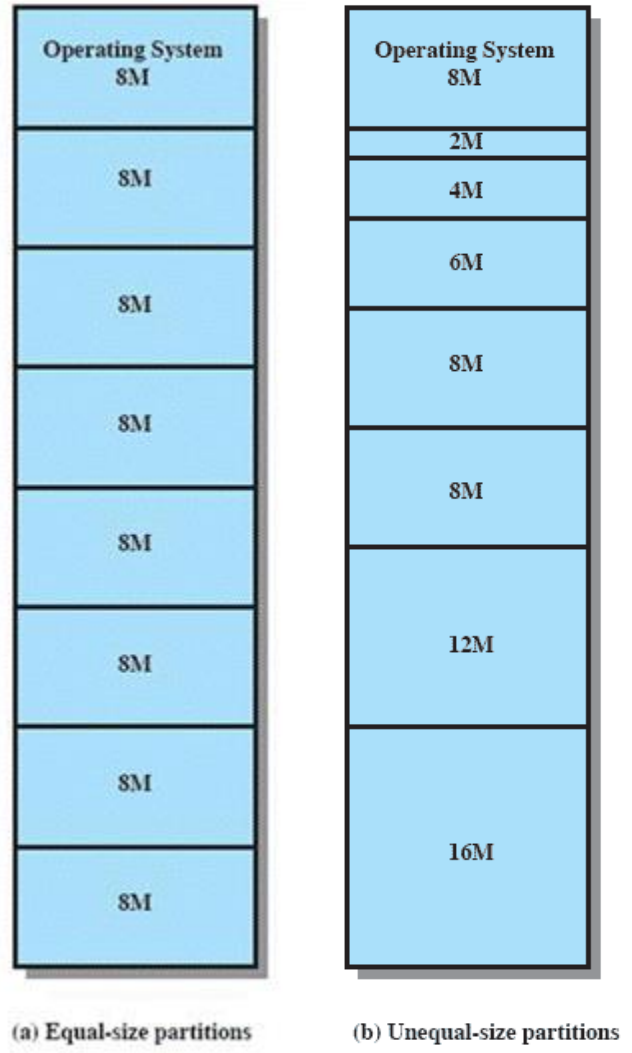
Upravljanje memorijom

Podela (particionisanje) memorije

- Raniji načini upravljanja memorijom
  - Pre razvoja koncepta virtuelne memorije
  - Danas se uglavnom ne koristi osim u ugrađenim/mobilnim računarima
- Dve tehnike
  - Fiksna podela (particionisanje) memorije
  - Dinamička podela (particionisanje) memorije

Fiksna podela memorije (1)

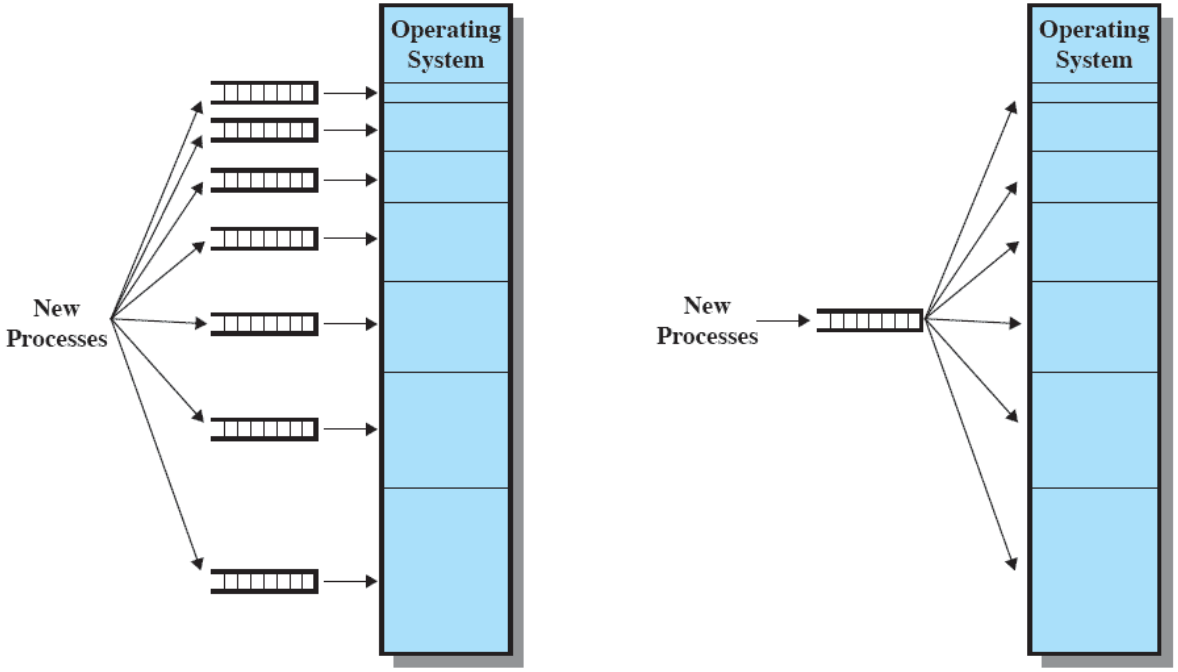
- Memorija je podeljena na fiksne particije pri startovanju sistema
- Particije mogu biti iste ili različitih veličina
- Procesima se dodeljuje memorijska particija veća od zahtevane
  - Proces veći od najveće dostupne particije mora da se deli u module i primenjuje tehnika ***overlay***
  - **Interna fragmentacija** - Preostali deo particije se ne evidentira kao slobodan i ne može se dodeliti nijednom procesu



(a) Equal-size partitions (b) Unequal-size partitions

Fiksna podela memorije (2)

- Algoritam raspoređivanja
  - Proces se smešta u red čekanja na odgovarajuću particiju u skladu sa zahtevom – nepovoljno ako veća particija bude oslobođena (a)
  - Proces se smešta u jedinstveni red čekanja – čim se oslobodi particija biva dodeljena prvom procesu od početka reda kome odgovara (b)

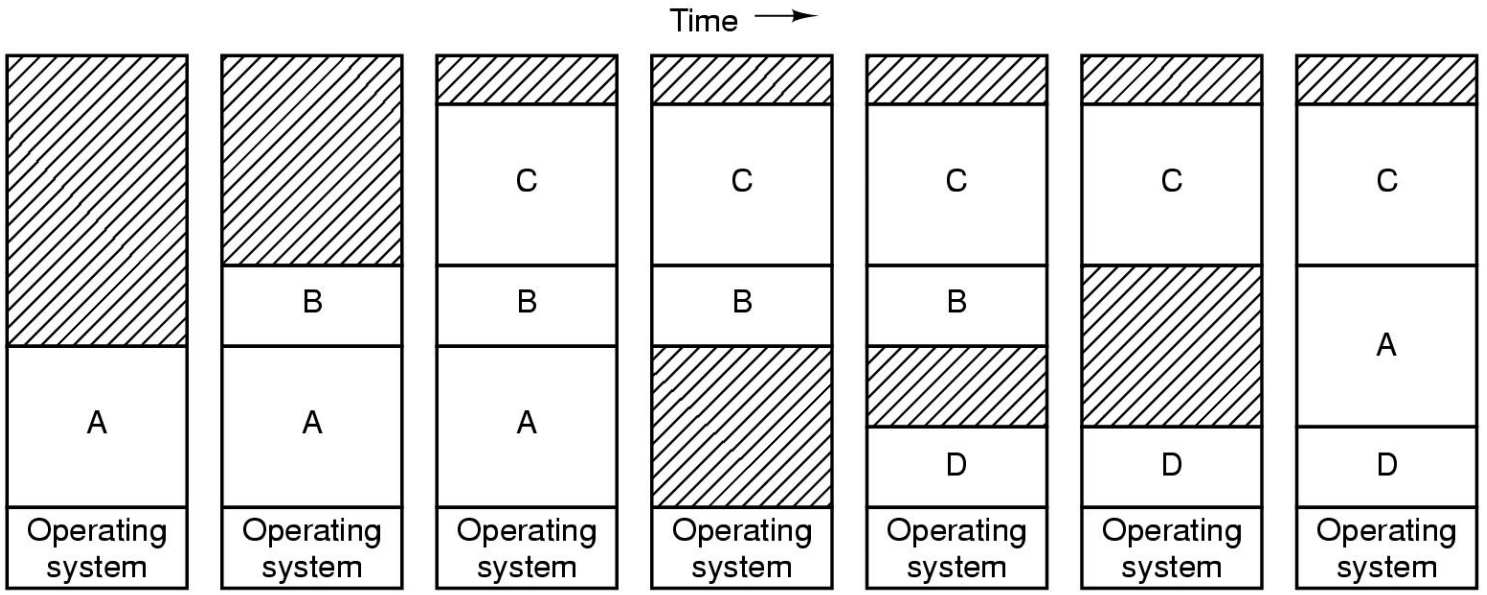


Problemi fiksne podele memorije

- Broj aktivnih procesa je ograničen sistemom
  - Ograničen pre-determinisanim brojem fiksnih memorijskih particija
- Veliki broj veoma malih procesa neće efikasno koristiti memoriju
  - U oba slučaja, sa jednakim i nejednakim particijama

Dinamičko deljenje na particije (1)

- Broj memorijskih particija, njihova veličina i stanje (slobodna, zauzeta) se menjaju sa aktiviranjem i završavanjem procesa – dinamički (IBM OS/MVT)
- Povećava iskorišćenost memorije, ali usložnjava dodelu i oslobađanje memorije, kao i evidenciju slobodnog i zauzetog memorijskog prostora

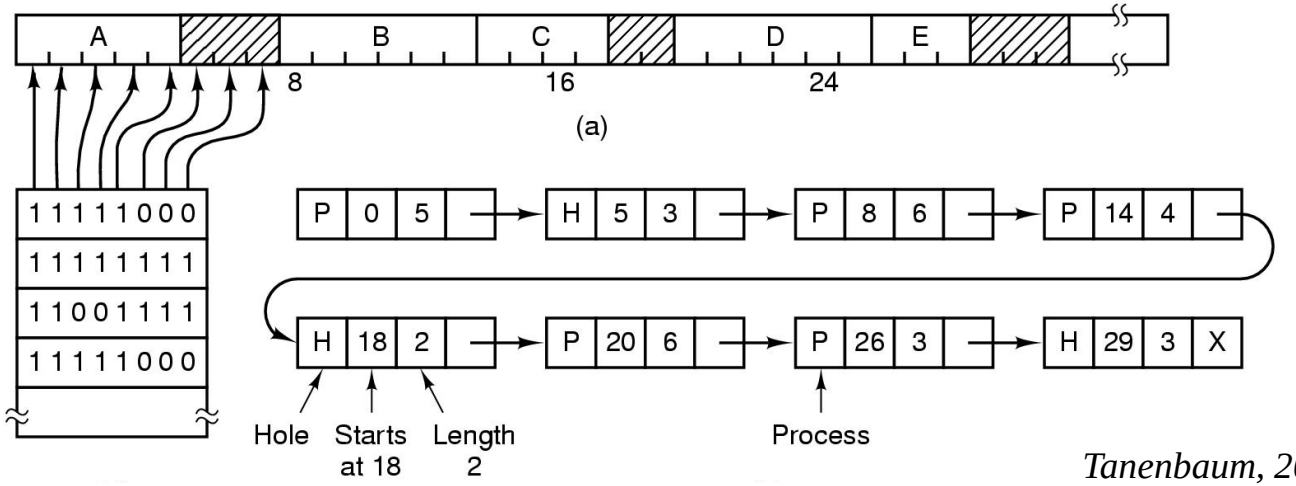


Dinamičko deljenje na particije (2)

- **Eksterna fragmentacija** – Ukupna veličina slobodnih particija je dovoljna za smeštanje određenog procesa, ali je nekontinualna u memorijskom prostoru.
- Rešenje je kompakcija (sažimanje) – prebacivanje memorijskih delova procesa prema jednom kraju adresnog prostora
  - Zahteva mnogo procesorskog vremena

Dinamičko deljenje na particije (3)

- Evidencija slobodnih i zauzetih memorijskih particija
  - Bitmapa – memorija je podeljena na alokacione jedinice od nekoliko reči do nekoliko KB: 0 – slobodna, 1 – zauzeta
    - Za proces koji zahteva k memorijskih jedinica, neophodno je pretražiti bitmapu za k sukcesivnih 0
  - Lančana lista – elementi lančane liste sadrže oznaku da li se radi o procesu ili slobodnoj particiji, početnu adresu, veličinu i link
    - Moguće su odvojene liste za procese i slobodne particije ili ulančavanje samih slobodnih particija



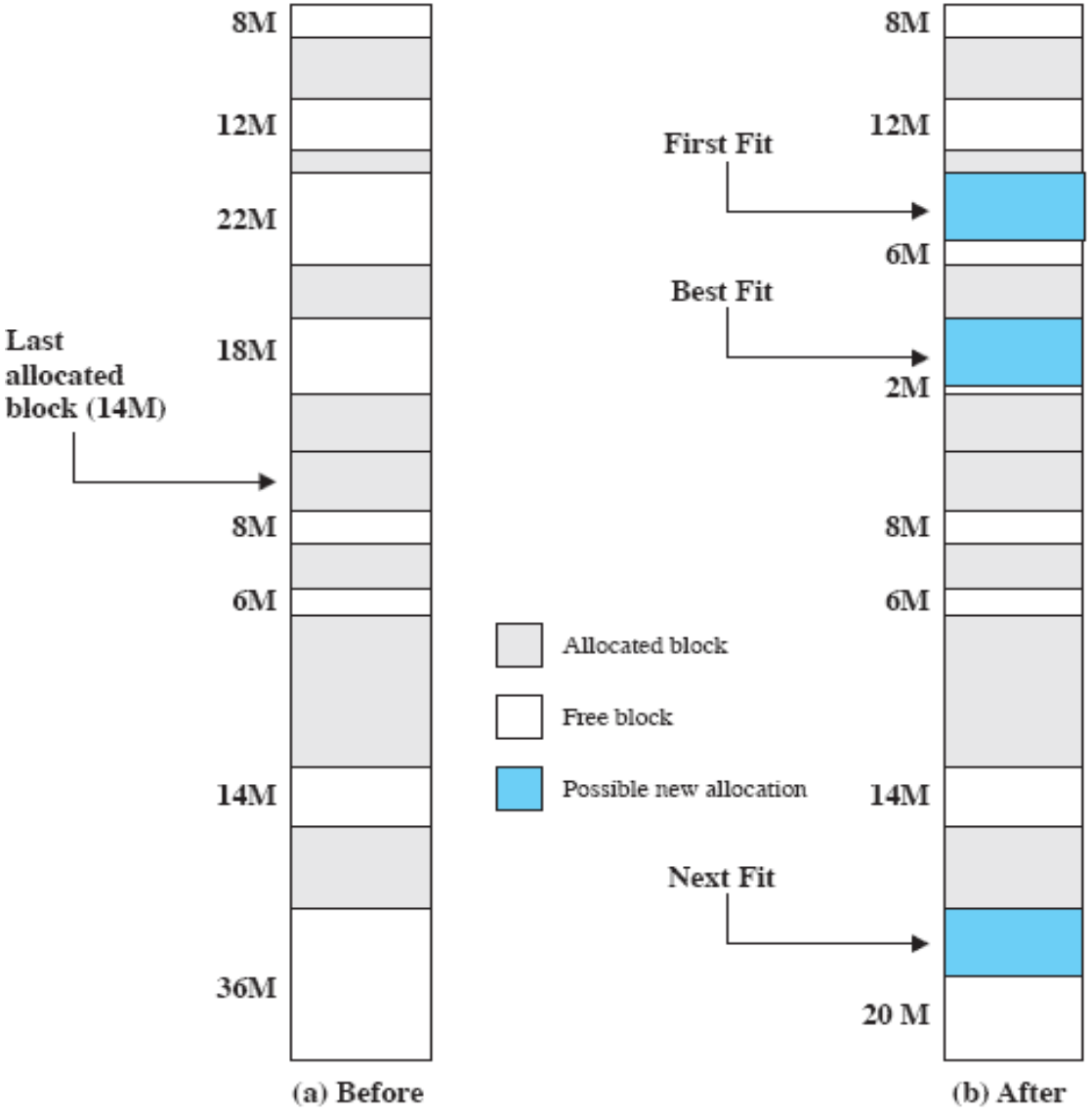
Tanenbaum, 2014

Algoritmi dodele dinamičkih particija

- **First fit** (Prvo poklapanje, prvi odgovarajući) – Pretražuje sve slobodne particije dok ne nađe dovoljno veliku, koja se deli na dva dela, jedan zauzima proces, a drugi deo ostaje kao slobodna particija
  - Jednostavno ukrupnjavanje susednih particija
- **Next fit** (Sledeće poklapanje, sledeći odgovarajući) – Kao i prethodni, ali pretraživanje započinje od adrese gde je nadjena prethodna slobodna particija
- **Best fit** (Najbolje poklapanje, najbolje odgovarajući) – Pretražuje sve particije dok ne nađe najmanju odgovarajuću particiju
  - Sporiji algoritam, jer zahteva pretraživanje cele liste osim ako je sortirana po veličini particija)
  - Generiše puno malih slobodnih particija (eksterna fragmentacija)
- **Worst fit** (Najgore poklapanje, Najgore odgovarajući) – Uzima najveću odgovarajuću particiju sa ciljem da preostali deo bude dovoljno velik da bi bio upotrebljiv
- **Quick fit** (Brzo poklapanje) – Organizuje posebne liste za particije određenih veličina (4KB, 8KB, 12KB, itd.) i eventualno binarno stablo u kome su čvorovi glave lančanih listi elemenata kojima se definišu particije odgovarajuće veličine

Alokacija dinamičkih particija

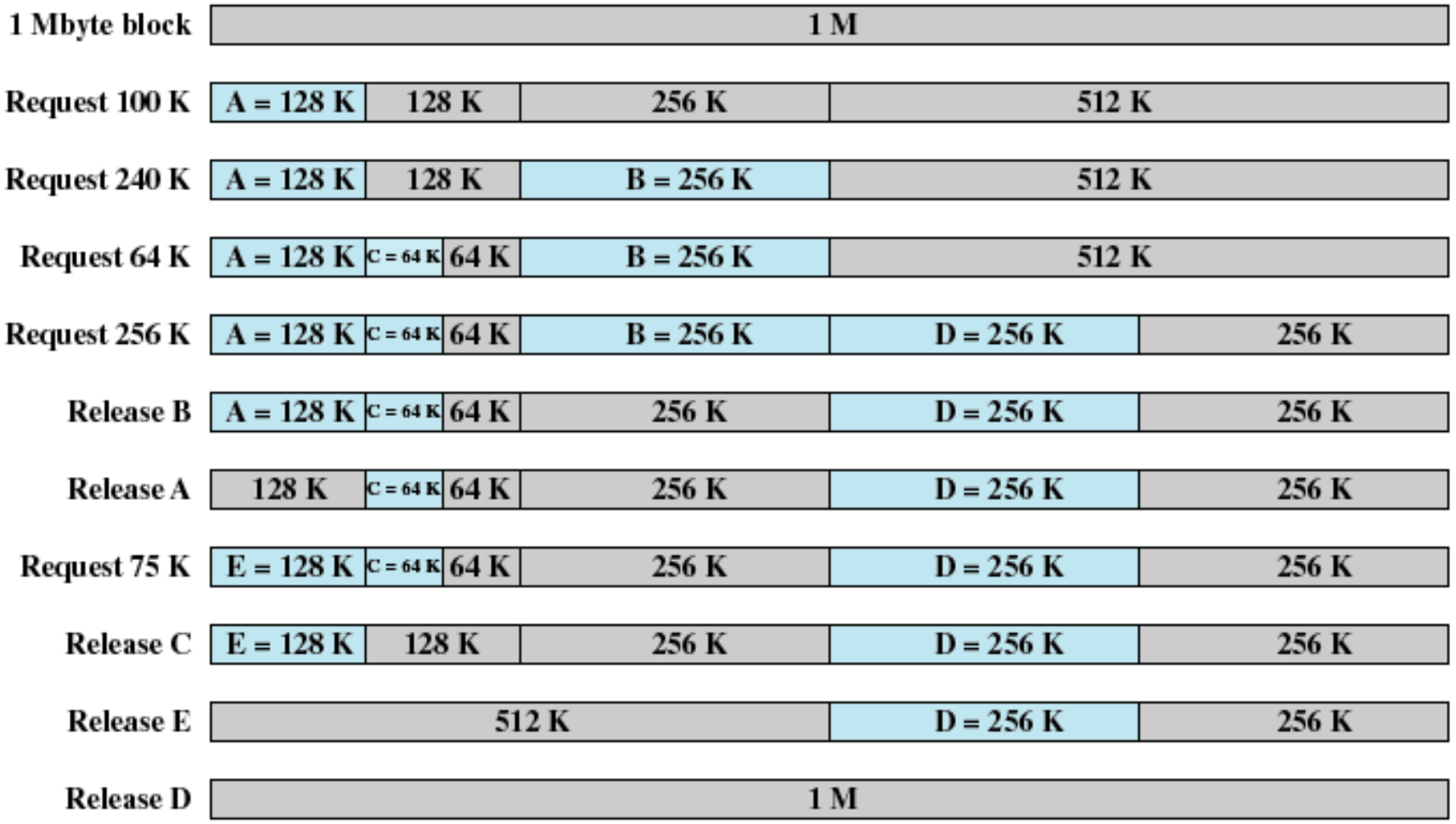
- Primer sadržaja memorije pre i nakon alokacije bloka od 16MB



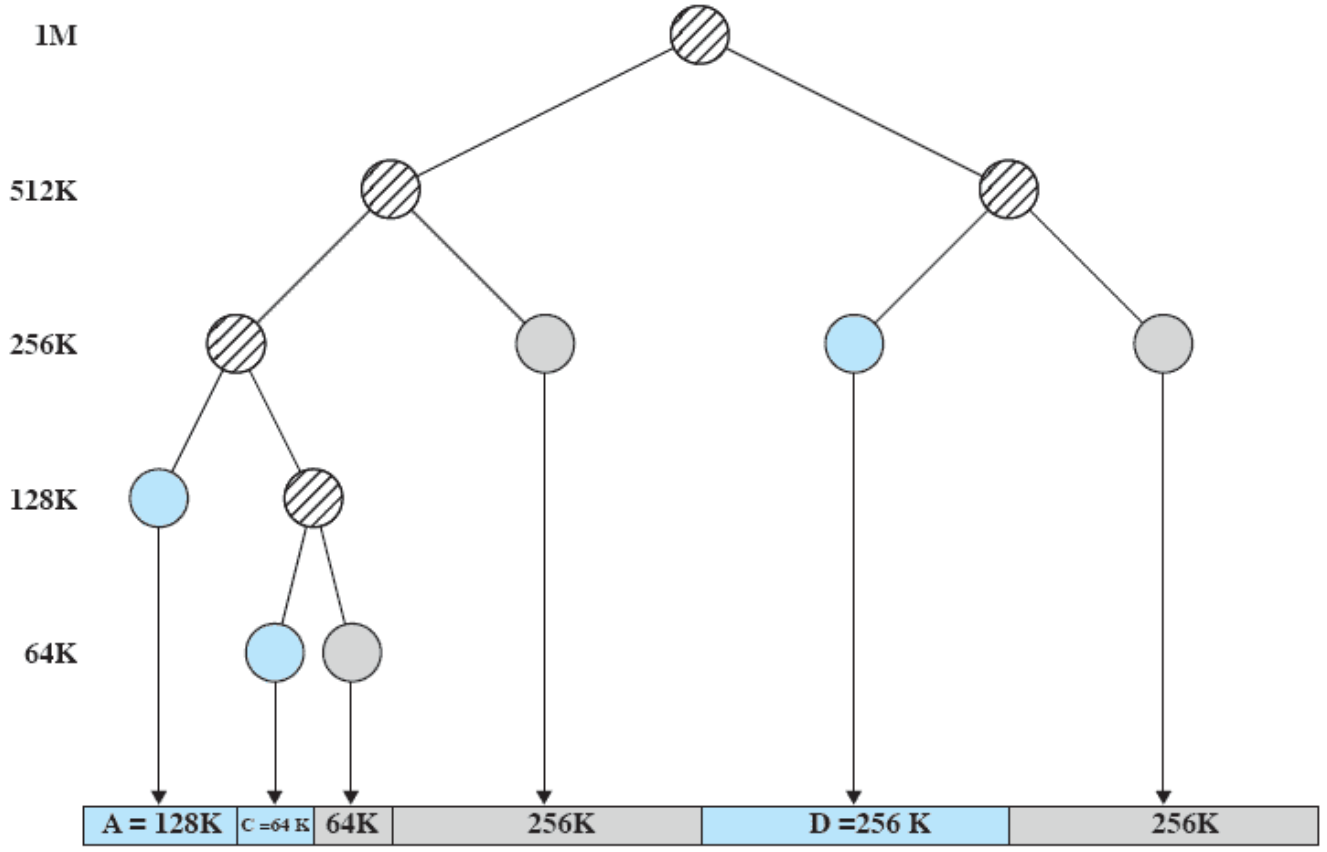
Sistem partnera (*Buddy system*)

- Celokupan raspoloživi memorijski prostor se tretira kao jedinstveni slobodni blok od 2^N
- Ako se pojavi zahtev za memorijom veličine ***s*** takve da je $2^{N-1} < s <= 2^N$, celokupan blok se dodeljuje
  - U suprotnom blok se deli na dva jednaka bloka duplo manje veličine (partnera, *buddies*)
  - Ovaj postupak se nastavlja sve dok se ne dobije najmanji blok veći ili jednak zahtevu s

Sistem partnera - primer



Reprezentacija sistema partnera – struktura stabla

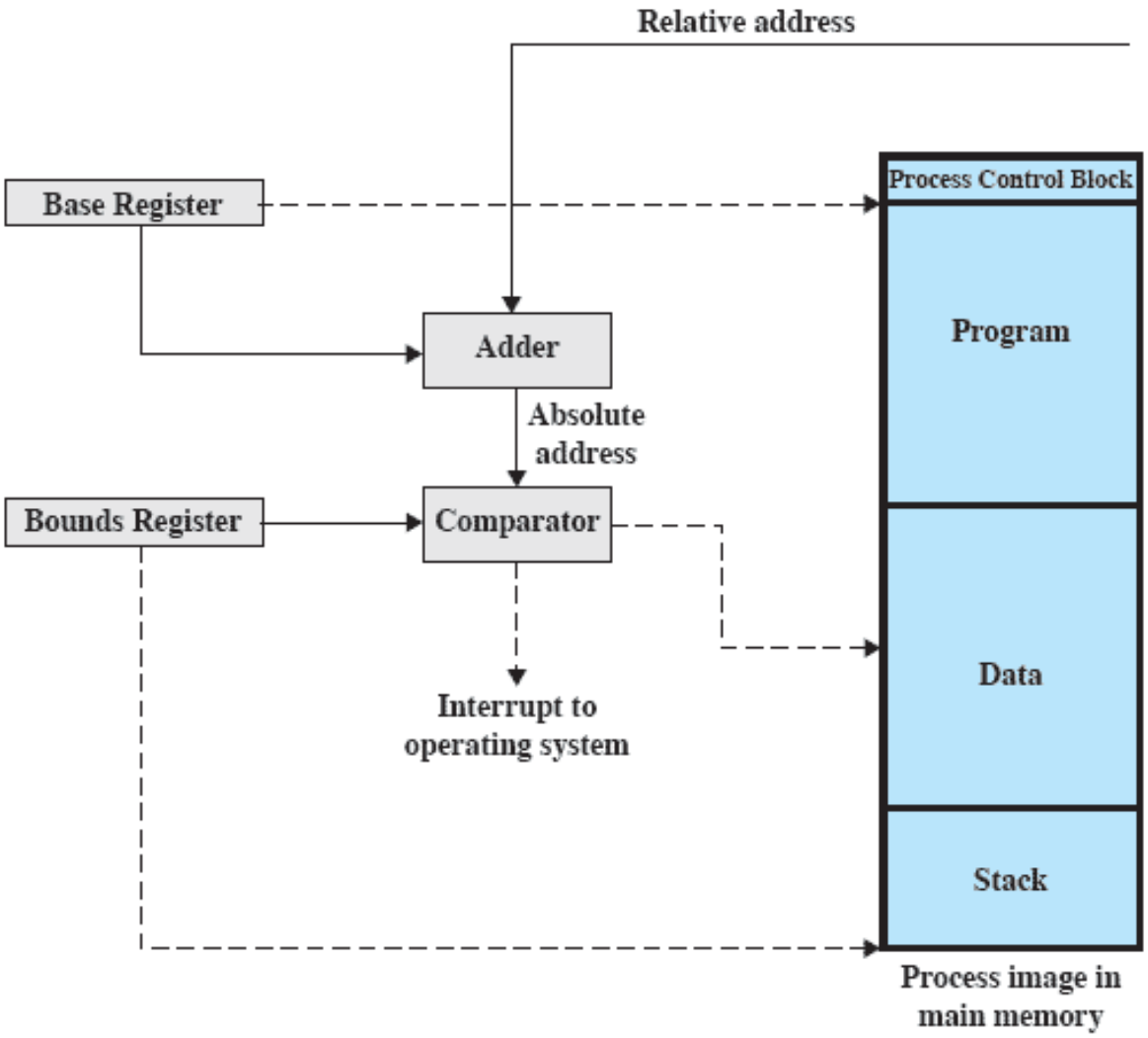


Relokacija

- Relokacija – smeštanje i pomeranje procesa u memoriji
- Kada je program (*main* funkcija, korisničke funkcije, bibliotečke funkcije, itd.) linkovan u link editoru, formira se izvršni kôd (modul) na disku sa logičkim adresama. Relokacija se može obaviti:
  - U vreme punjenja, modifikovanjem svih adresa u skladu sa memorijskom adresom punjenja (IBM OS/360)
  - U vreme izvršenja, korišćenjem ***base*** (relocation) i ***bounds*** registara
- Memorijske adrese
  - Logička adresa – referenca na memorijsku lokaciju nezavisna od stvarnog smeštanja procesa u memoriju
  - Relativna adresa – poseban primer logičke adrese koja se izražava relativno u odnosu na neku poznatu adresu, obično vrednost u CPU registru
  - Fizička adresa – adresa lokacije u glavnoj memoriji

Relokacija

- Hardverska podrška za relokaciju



Nekontinualna dodela memorije

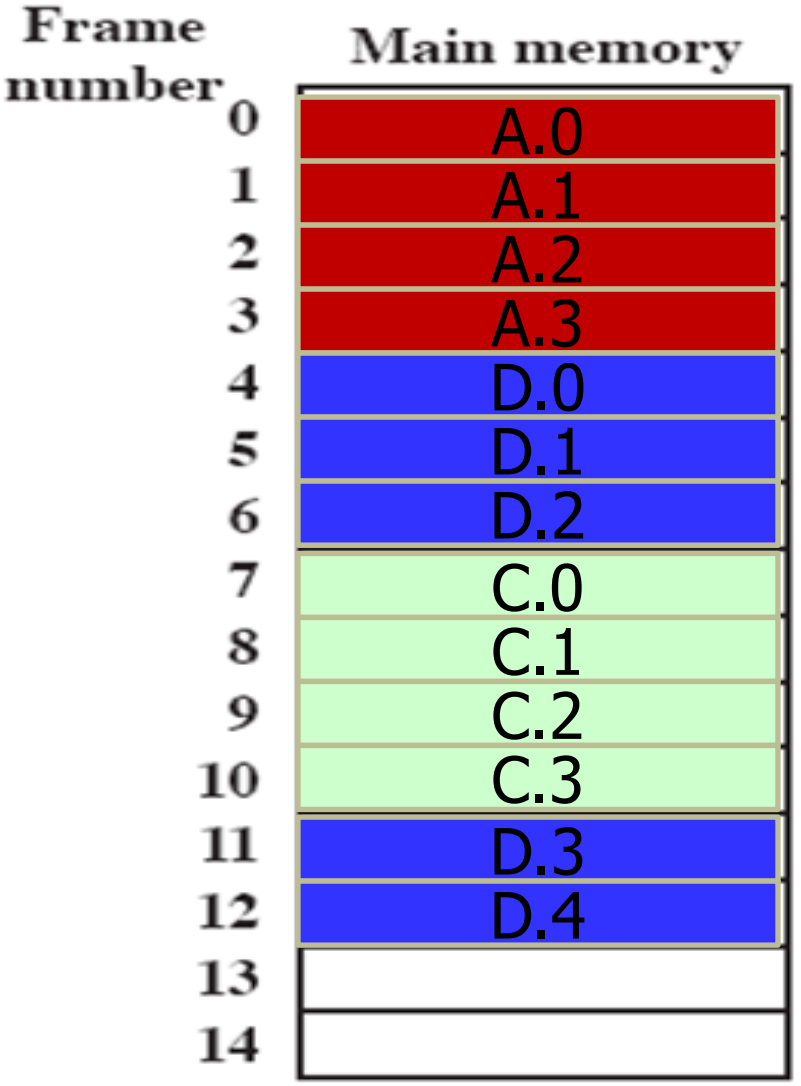
- Nekontinualna dodela memorije – logički adresni prostor procesa ne mora biti smešten kontinualno u glavnoj memoriji
- Metode nekontinualne dodele:
  - Straničenje (*Paging*)
  - Segmentacija (*Segmentation*)
  - Segmentacija sa straničenjem (*Segmentation with paging*)

Straničenje

- Adresni prostor procesa je podeljen na delove određene veličine - **Stranice** (*Page*) – Logičke stranice
- Fizička memorija je (logički) podeljena na delove iste veličine – **Stranični okviri** (*Page Frame*) – Fizičke stranice
- Veličina stranica i straničnih okvira je od 512B – 64KB (tipično 4KB)
- Veličina stranice savremenih računara može biti promenljiva:
  - MIPS (4KB-16MB), UltraSparc (8KB – 4MB), Pentium (4KB – 4MB), PowerPC (4KB), DEC Alpha (8KB), Itanium (4KB – 256MB)
- Transformisanje iz logičke (virtuelne) u fizičku adresu obavlja se korišćenjem **Tabele stranica** (*Page Table*)

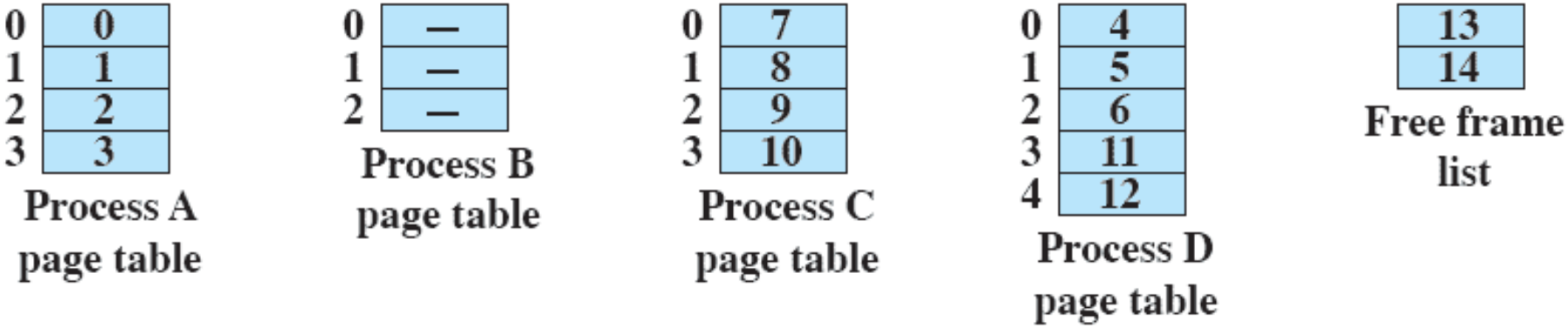
Stranice i okviri

- Da bi se startovao proces veličine **n** stranica neophodno je postojanje **n** slobodnih straničnih okvira u memoriji – nema eksterne fragmentacije
- Interna fragmentacija – u okviru poslednje stranice postoji interni fragment, jer proces (skoro) nikad ne zahteva ceo broj stranica
  - Zahtev 32128 B -> 8 stranica od 4KB = 32768
  - Interni fragment 640 B u poslednjoj stranici

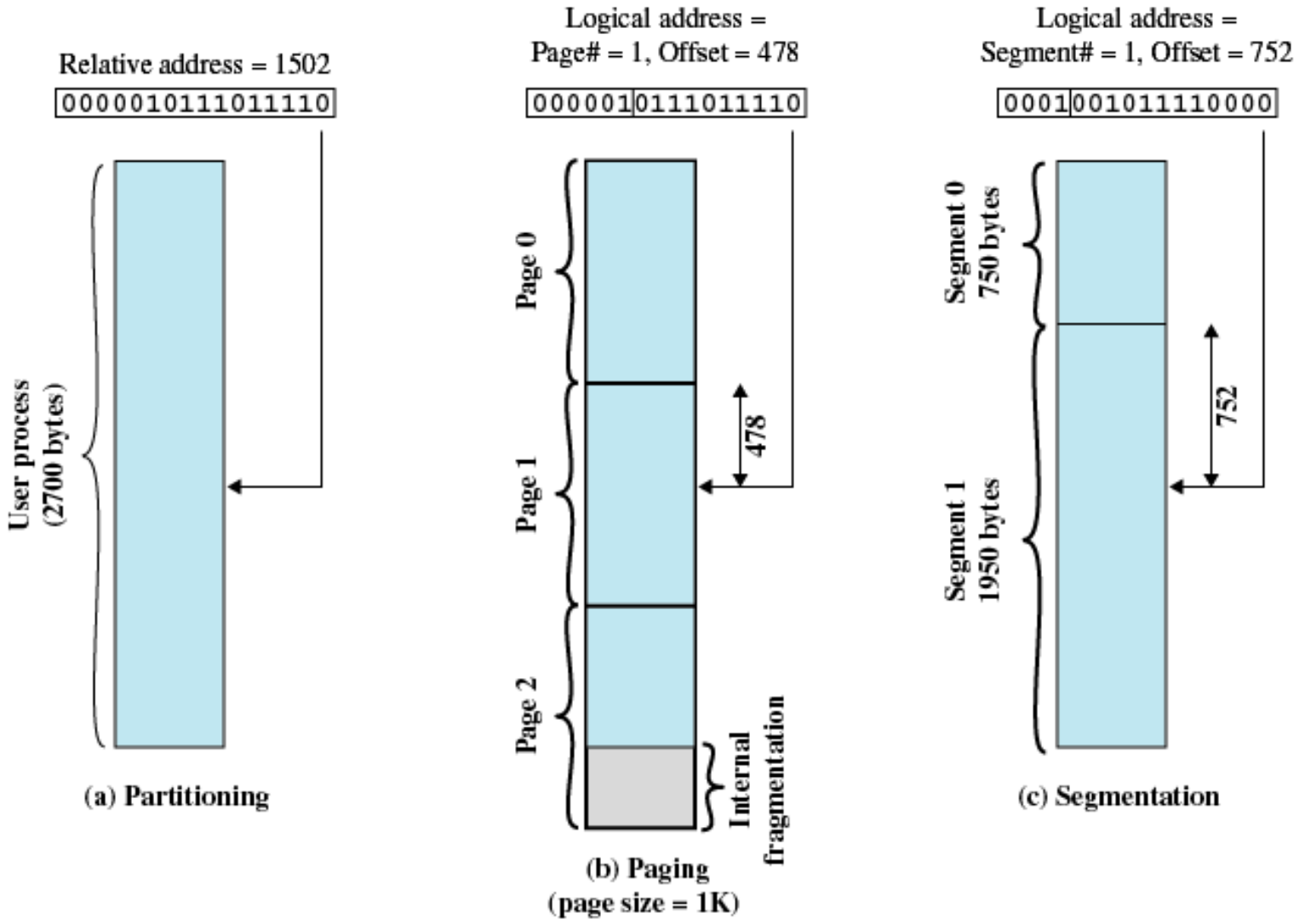


Straničenje – tabele stranica

- Tabele stranica za procese sa prethodne slike u poslednjem vremenskom trenutku

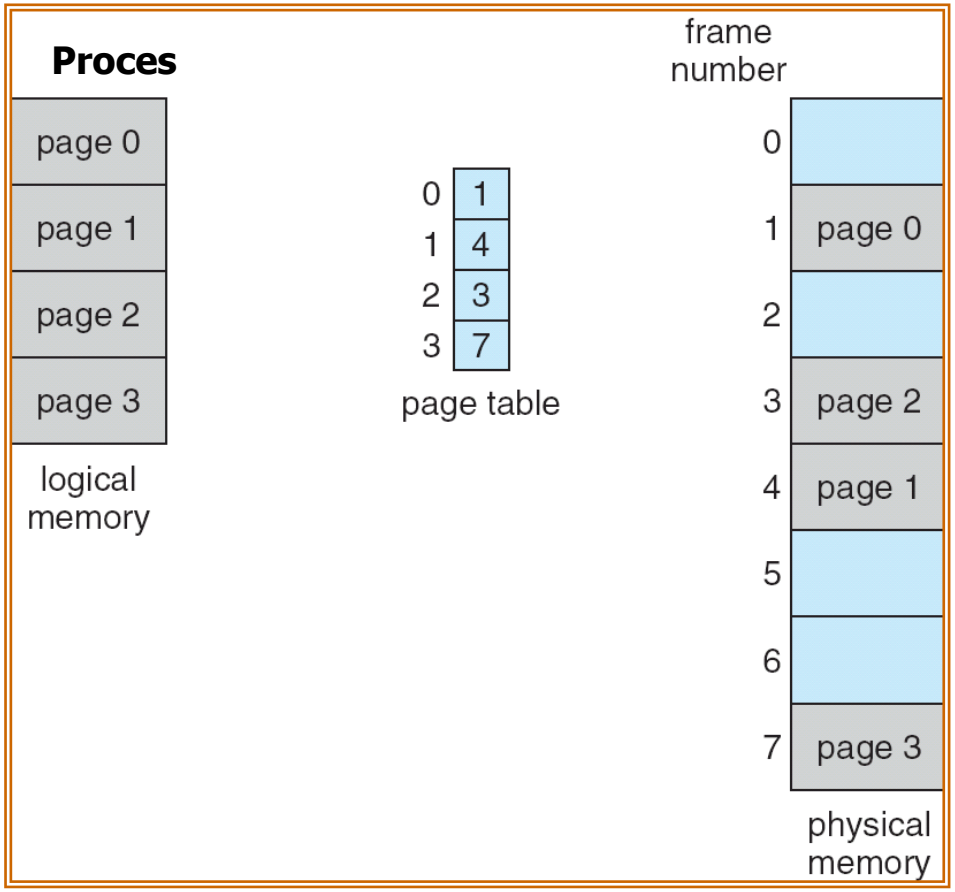


Logičke adrese (primer za adresu 1502)



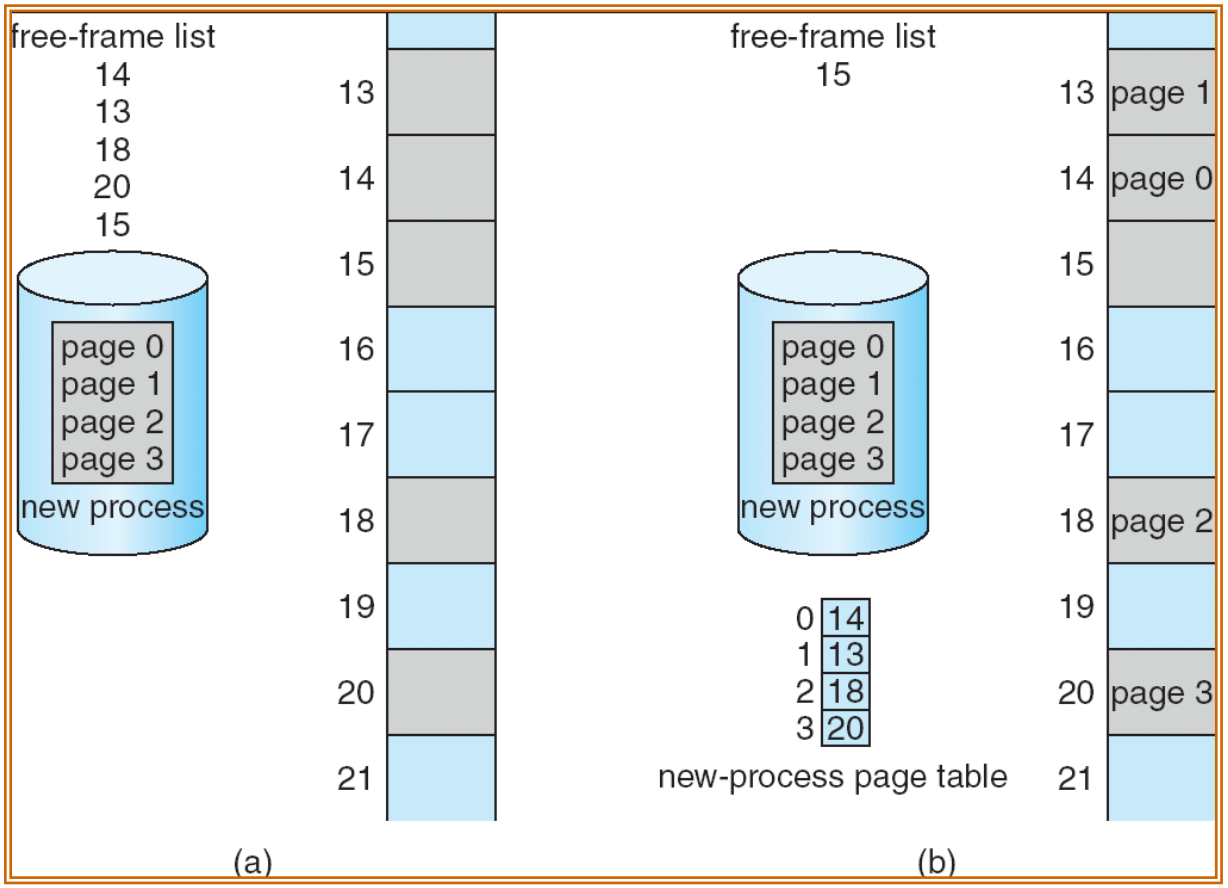
Straničenje – primer

- Adresni prostor procesa sadrži 4 stranice
- Fizička memorija sadrži 8 straničnih okvira



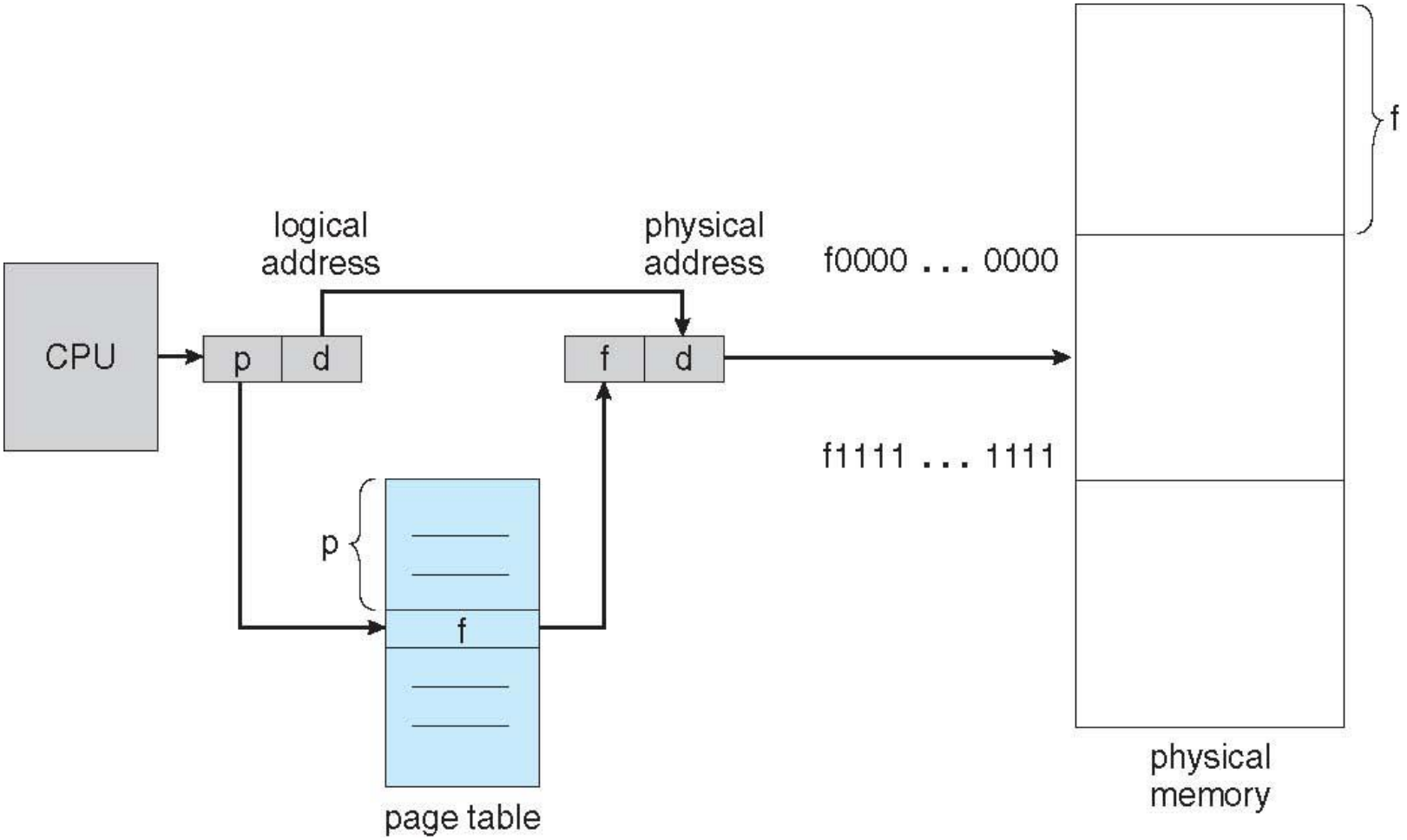
Silberschatz, 2013

Dodela memorije straničenjem - Primer



Silberschatz, 2013

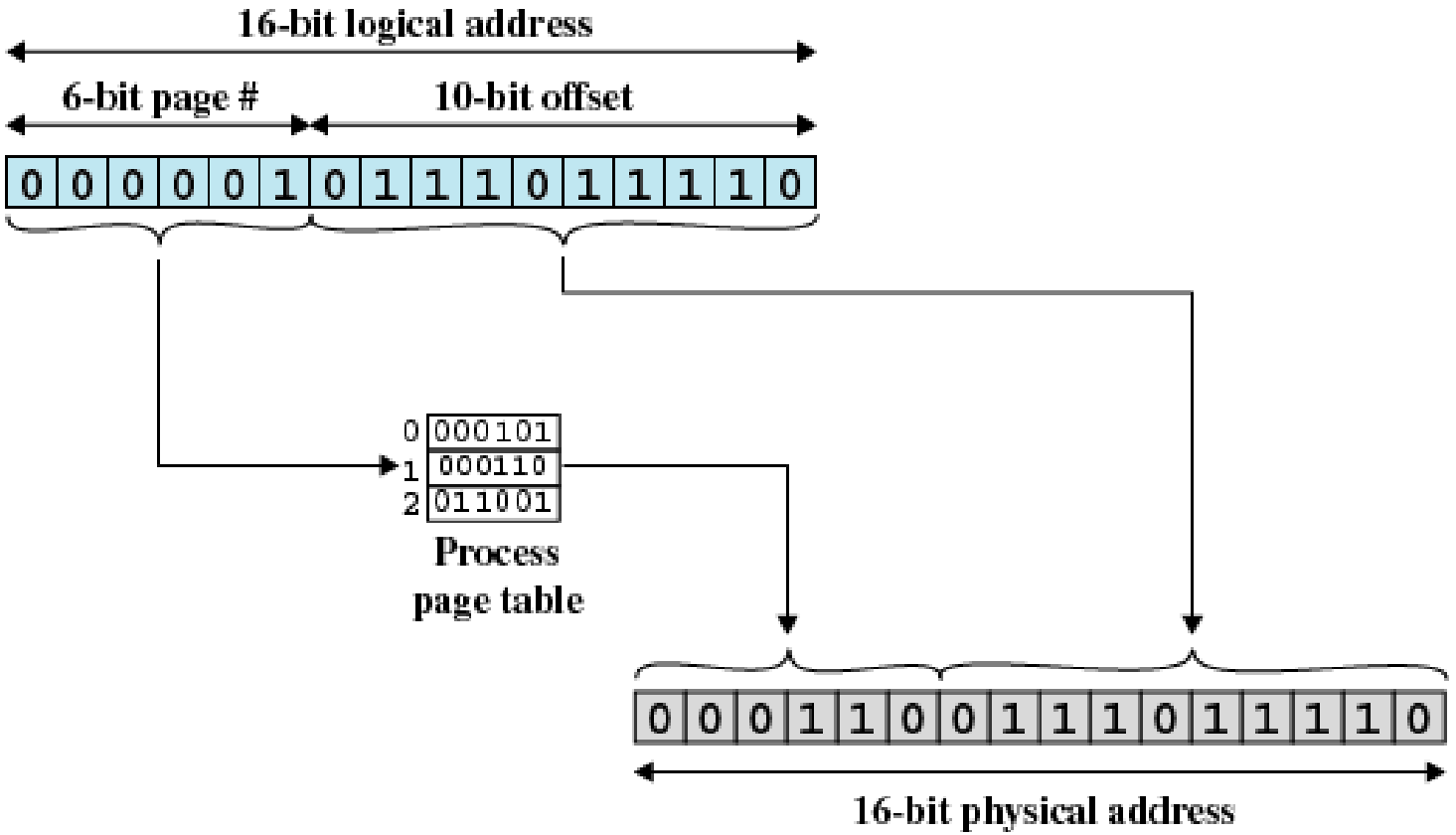
Prevođenje logičke u fizičku adresu – tabela stranica



Silberschatz, 2013

Prevođenje logičke u fizičku adresu – primer

- Stranice su veličine 1KB



Segmentacija

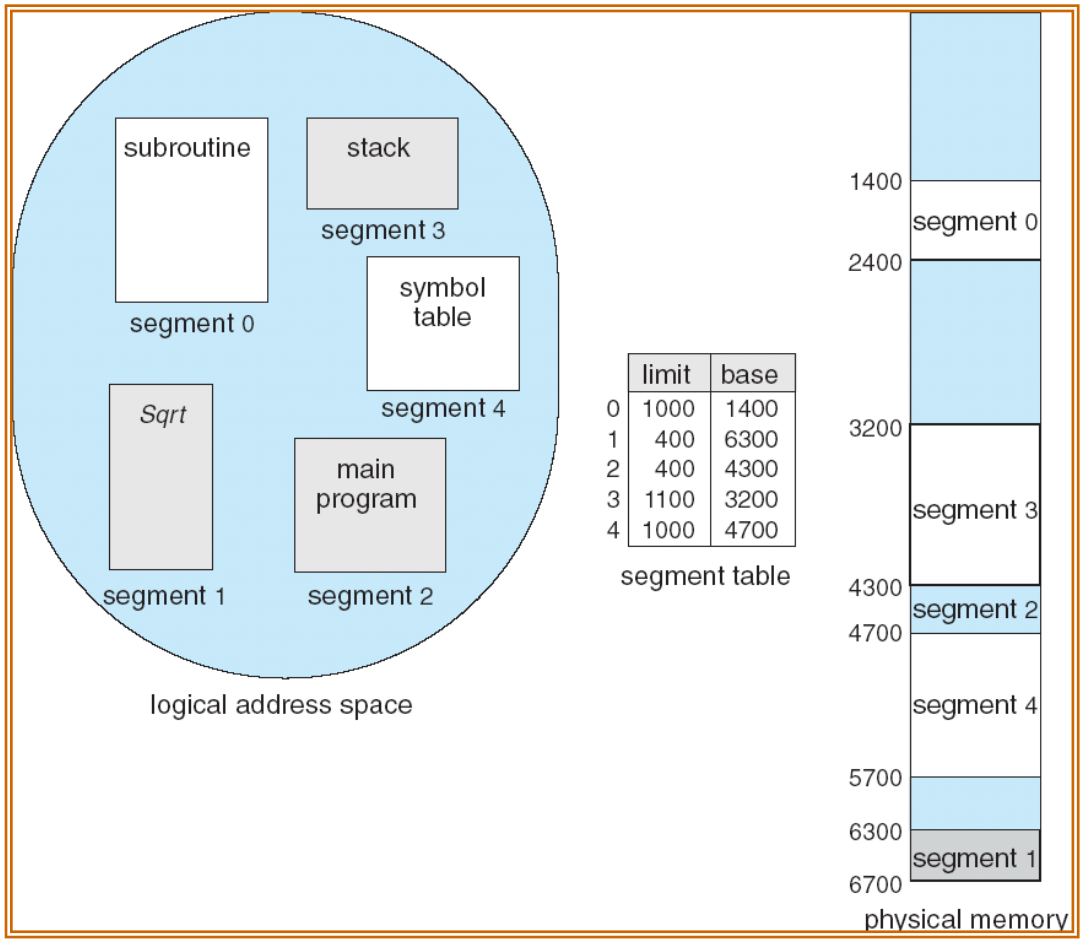
- Šema za upravljanje memorijom koja odgovara korisnikovom pogledu na memoriju u kome je program kolekcija segmenata
- Segment je logička jedinica poput: glavnog programa (*main*), procedura, funkcija, metoda, objekata, lokalnih i globalnih promenljivih, *common* blokova, magacina, tablica simbola, polja, itd.
- Svaki segment predstavlja poseban adresni prostor i sadrži linearnu sekvencu adresa, od 0 do određenog maksimuma.
- Prednosti segmentacije:
  - Segmenti mogu da rastu i smanjuju se nezavisno jedan od drugog, i da budu smešteni u nekontinualne memorijske particije
  - Svaki segment ima poseban tip zaštite u skladu sa tipom segmenta
  - Segmentacija omogućava deljenje segmenata sa procedurama i podacima između različitih procesa (deljene biblioteke – *shared library*)
- Nedostatak:
  - Eksterna fragmentacija

Implementacija segmentacije

- Logička adresa se sastoji od dva dela
  <broj_segmenta, ofset>
- Tabela segmenata – transformiše logičku u fizičku adresu
- Ulaz u tabelu sadrži:
  - Baza (*Base*) – sadrži startnu fizičku adresu segmenta u memoriji
  - Limit (*Length*) – specificira veličinu segmenta
  - Zaštita – specificira prava pristupa sadržaju segmenta
  - Validnost – da li je segment u memoriji ili na disku
- Bazni registar tabele segmenata – sadrži adresu početka tabele segmenata u memoriji
- Granični registar tabele segmenata- sadrži broj segmenata programa ili adresu završetka tabele segmenata

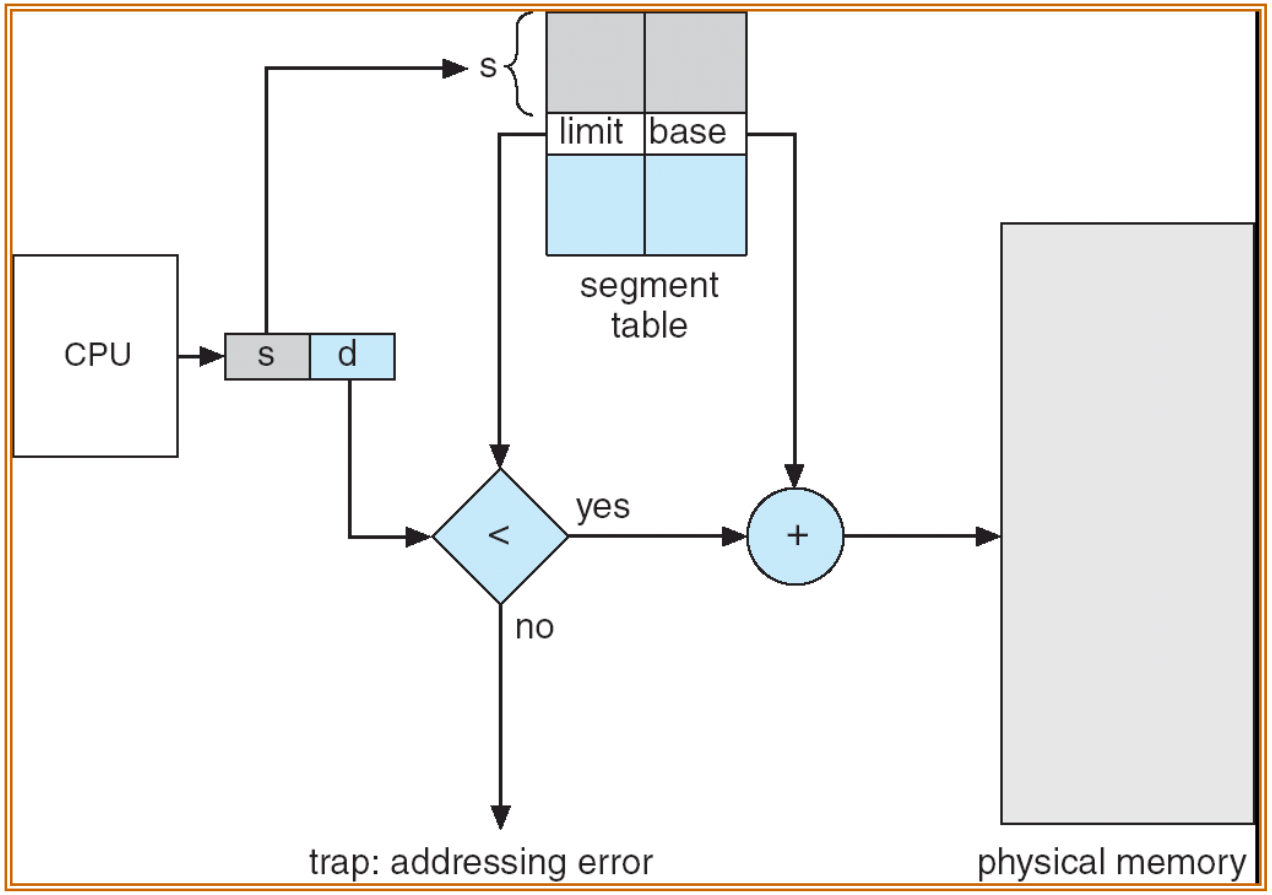
Segmentacija - primer

- Smeštanje u memoriju procesa sa pet segmenata



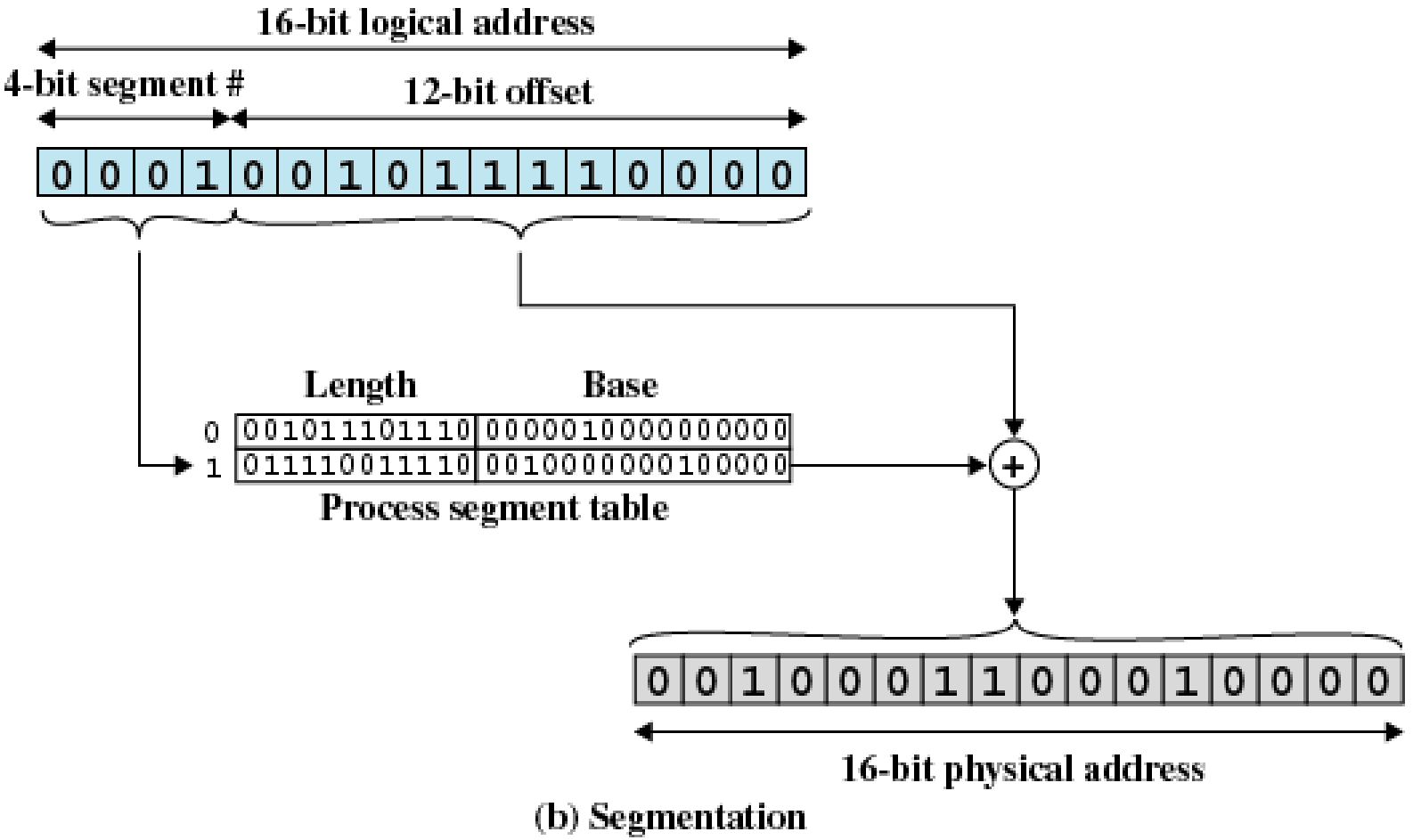
Silberschatz, 2013

Prevođenje logičke u fizičku adresu korišćenjem tabele segmenata



Silberschatz, 2013

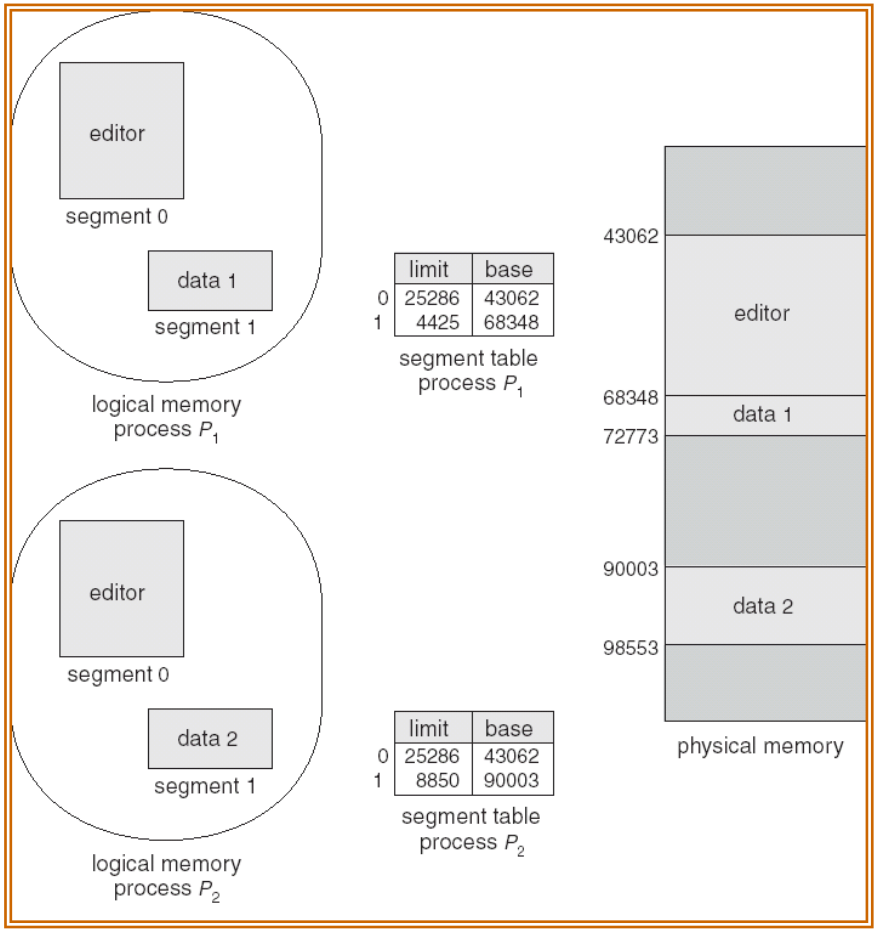
Prevođenje logičke u fizičku adresu - primer



(b) Segmentation

Deljenje segmenata

- Više procesa koristi tekst editor, tako da dele segment sa kôdom editora, dok svaki proces ima sopstveni segment sa podacima



Silberschatz, 2013

Domaći zadatak

- Poglavlje **7 Upravljanje memorijom**
  - 7.8 Ključni pojmovi, kontrolna pitanja i problemi
- Paging & segmentation animations
  - https://apps.uttyler.edu/Rainwater/COSC3355/Animations