



Uvod u računarstvo u oblaku

**Tehnologije i sistemi neophodni za razumevanje koncepta
računarstva u oblaku**

Odabrana poglavlja iz operativnih sistema

Nemanja Maček

- Šta je operativni sistem i koje su mu funkcije?
- Šta je jezgro, šta su procesi i kako se njima upravlja?
 - Problem sinhronizacije
 - Problem zastoja
- Radna i virtuelna memorija
- Sekundarne memorije i sistemi datoteka

Šta je operativni sistem?

- Operativni sistem (OS):
 - Upravlja programima, podacima i delovima računara (procesorom, kontrolerima, memorijom), s ciljem da oni budu što celishodnije upotrebljeni.
 - Kreira pristupačno radno okruženje za krajnjeg korisnika računara.
 - Sakriva od korisnika detalje funkcionisanja ovih delova koji nisu bitni za korišćenje računara.
- Drugim rečima, OS je skup programa koji:
 - **upravljaju resursima** računarskog sistema i
 - **obezbeđuju interfejs** prema korisniku.

Šta su resursi računara?

- Pod pojmom resurs podrazumevamo sve što je programu potrebno za rad.
- Resursi mogu biti:
 - **Hardverski** (procesor, memorija, ulazno-izlazni uređaji)
 - **Softverski** (programi, podaci, odnosno datoteke svih vrsta).
- Zadatak Osa je da vodi računa o resursima računara, odnosno da zadovolji potrebe programa, da prati koji program koristi koje resurse itd.
- Primer 1.
 - Dva korisnika višekorisničkog sistema istovremeno žele nešto da štampaju.
 - OS je dužan da obezbedi dostupnost štampača programima tih korisnika i da spreči da se podaci poslati na štampu mešaju.
- Primer 2.
 - Dva korisnika istovremeno pregledaju različite Web stranice preko *Remote Desktop*-a.
 - OS je dužan da spreči mešanje sadržaja preuzetih sa Mozart-ove kladionice i *Cannibal Corpse* kanala na Youtube serisu.

Koje grupe programa upravljaju resursima?

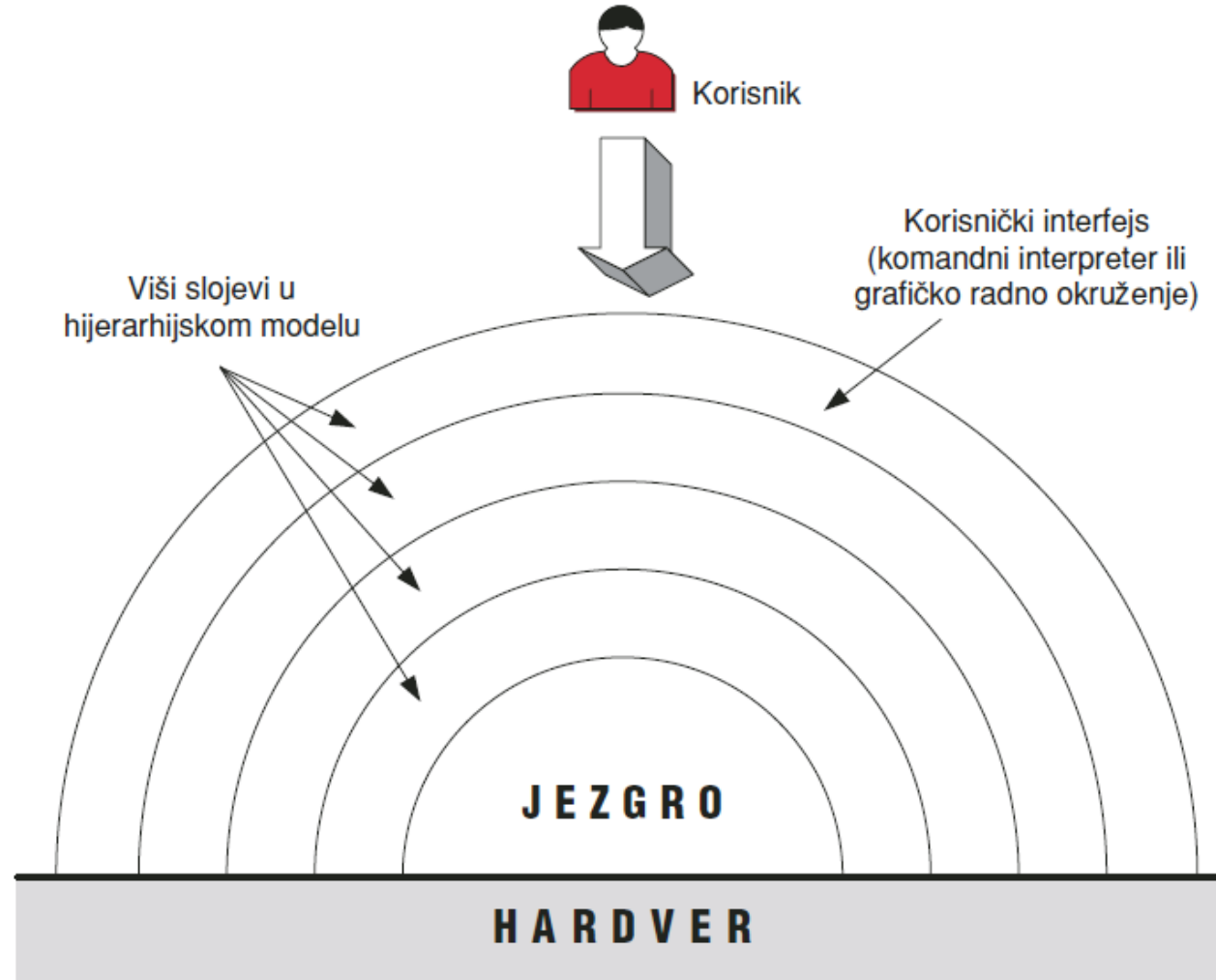
- Upravljanje osnovnim resursima računarskog sistema obezbeđuje više funkcionalnih grupa programa:
 - Upravljanje procesorom
 - Upravljanje memorijom
 - Upravljanje ulazom i izlazom
 - Upravljanje podacima
 - Upravljanje sekundarnom memorijom
 - Umrežavanje
 - Zaštita
- Ove grupe su **hijerarhijski organizovane**.
- Ponekad je teško precizno reći (usaglasiti) šta se podrazumeva pod delovima OS, odnosno da li kontrolni programi pripadaju ili ne pripadaju OS.

Šta je to interfejs prema korisniku?

- Da li zaista želite da sami određujete stazu i sektor prilikom čitanja sa diska ili Vam je lakše da takav posao prepustite operativnom sistemu?



Hijerarhijski model operativnog sistema



Hijerarhijski model operativnog sistema

- Jedan sloj (čak ni najniži) ne nameće konkretan ostatak OS-a!
- Više OS-a mogu da koriste isto ili slično jezgro.
 - Primer: jedno (vrlo slično) jezgro koriste dva operativna sistema – Linux i Android.
- Slično, više OS-a mogu da koriste isti sistem datoteka.
 - Primer: i Windows i Linux koriste FAT i NTFS.



Šta karakteriše operativne sisteme?

- **Konkurentnost.**
 - Postojanje više simultanih, paralelnih aktivnosti.
- **Deoba resursa** (softverskih i hardverskih).
 - Konkurentne aktivnosti mogu da zahtevaju deljenje resursa ili informacija.
- **Efikasnost.**
 - Efikasnost je važna.
 - *Housekeeping* poslovi su takođe važni, ali na njih ne sme da otpadne veći deo vremena!
- **Visok nivo pouzdanosti.**
- **Prihvatljiva veličina.**

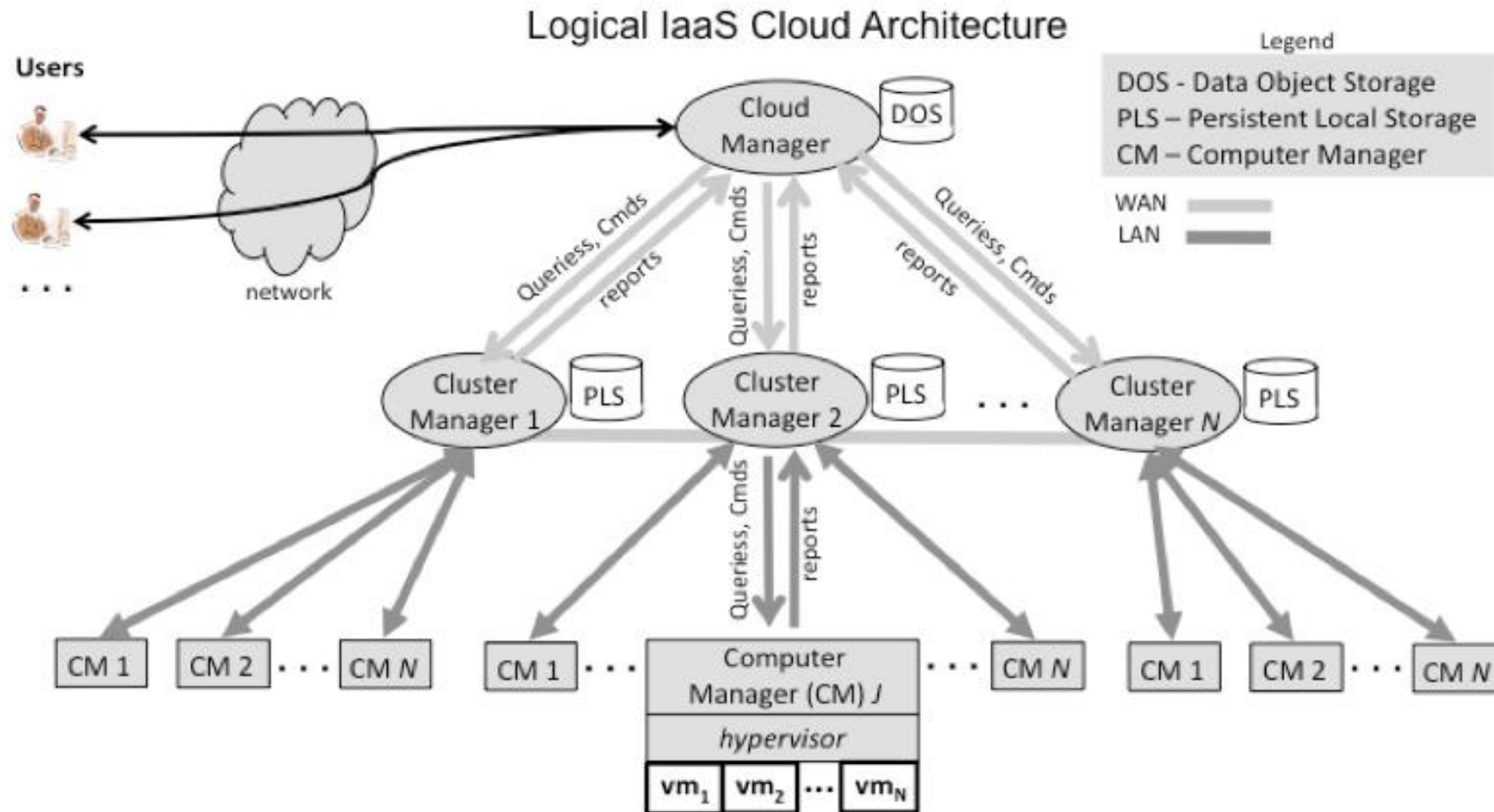
Šta su distribuirani sistemi?

- Kolekcija računara koji **ne dele** zajedničku memoriju i sistemski časovnik.
 - Svaki procesor, odnosno računar, ima sopstvenu lokalnu memoriju, a međusobna komunikacija se ostvaruje putem mreže realizovane kao LAN ili WAN.
 - Osim podataka, datoteka i štampača, distribuiraju se i procesi.
- Distribuirani sistemi zahtevaju mrežnu infrastrukturu i mogu biti realizovani ili kao:
 - **Klijent-server arhitektura.**
 - **Ravnopravni računarski sistemi** koji po mreži dele resurse.
- U klijent-server arhitekturi, postoje računari koje predstavljaju servere i računari koji koriste njihove usluge - klijenti.
- Postoje dve vrste servera:
 - **Serveri za izračunavanje**, kojima klijenti šalju zahteve na obradu.
 - **Serveri podataka** koji služe za smeštanje datoteka.

Šta su udruženi sistemi?

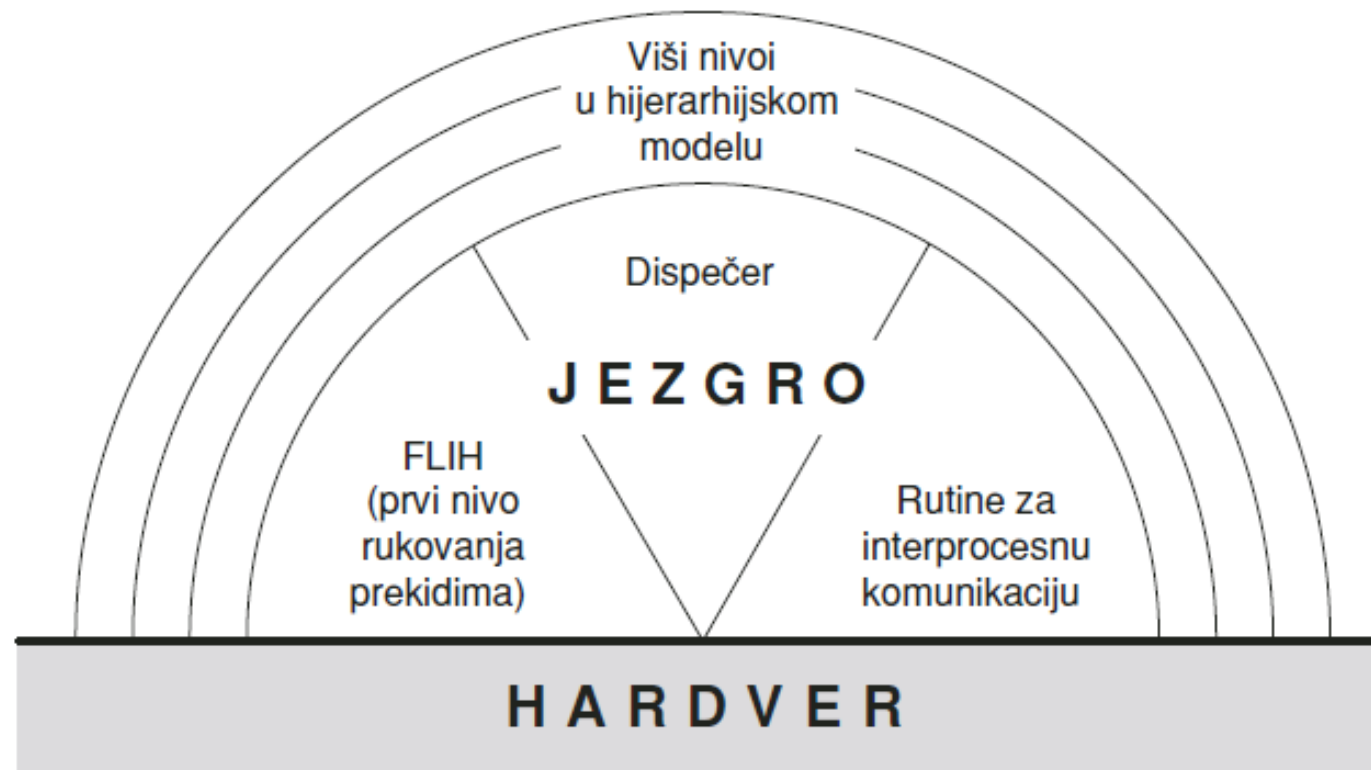
- Udruženi sistemi (*clustered system*) sastoje se od udruženih računara, odnosno od dva ili više nezavisnih računara koji dele diskove i čvrsto su povezani LAN mrežom.
- Svaki računar u udruženom sistemu naziva se čvor (*node*).
- Na svakom čvoru se izvršava jedan nivo softvera za udruživanje.
- Postoje dve vrste udruživanja:
- **Asimetrično udruživanje.**
 - Jedan server (čvor) izvršava aplikaciju, dok ostali (prateći) serveri prate rad glavnog servera u budnom ali neaktivnom stanju.
 - U slučaju otkaza glavnog servera jedan od pratećih servera preuzeće njegovu ulogu.
- **Simetrično udruživanje.**
 - Svi serveri su aktivni i izvršavaju aplikaciju.
 - Time se znatno poboljšavaju performase, ali sistem mora da izdrži otkaz jednog ili više servera kao i u perthodnom slučaju.

Da li se sećate ove slike?



- **Kernel** (jezgro) je osnovni je deo svakog OS.
- U hijerarhijskom (slojevitom) modelu, kernel je najbliži hardveru.
 - Kernel je veza, odnosno interfejs između hardvera i ostalih slojeva OS.
 - U slojevitom modelu NT arhitekture ispod kernela se nalazi sloj apstrakcije hardvera (HAL) koji omogućava OS da vidi različit hardver na isti način.
- Osnovna funkcija kernela je upravljanje procesima, odnosno:
 - Stvaranje okoline u kojoj mogu postojati procesi
 - Dodeljivanje procesora procesima
 - Obezbeđivanje mehanizama za interprocesnu komunikaciju.
- Procesor je **nedeljivi resurs!**
 - Na jednom procesoru sa jednim jezgrom u jednom trenutku može se izvršavati samo jedan proces.
 - Kernel određuje kada i na koje vreme će proces dobiti procesor.
 - Ova pojava je poznata pod imenom multipleksiranje i predstavlja osnovu kvaziparalelnosti.

- Jezgro takozvani model OS-a „na papiru“.



- **Prvi nivo obrade prekida** (*FLIH, First Level Interrupt Handler*).
 - Rutine za određivanje izvora prekida i iniciranje servisa (opsluživanje nekih vrsta prekida).
 - *FLIH* odgovara na spoljašnje prekide i sistemske pozive.
 - Posle izvršavanja koda koji čini ovaj deo Osa, prekid se smatra opsluženim, a dispečer dalje odlučuje kom će procesu dalje predati procesor na korišćenje.
- **Dispečer sistema** (planer poslova niskog nivoa, *low-level scheduler*).
 - Deo kernela koji dodeljuje procesor procesima.
 - Procesor se uvek dodeljuje na osnovu nekog algoritma (na primer, *Shortest Job First*).
- **Rutine za ostvarivanje interprocesne komunikacije.**
 - Deo jezgra Osa koji obezbedjuje različite načine komunikacije među procesima, kao što su:
 - Slanje poruka (*send message, post message*)
 - Semaforske tehnike
 - Korišćenje imenovanih cevi (*named pipes*, mehanizam karakterističan za UNIX sisteme)
 - Korišćenje deljive memorije.

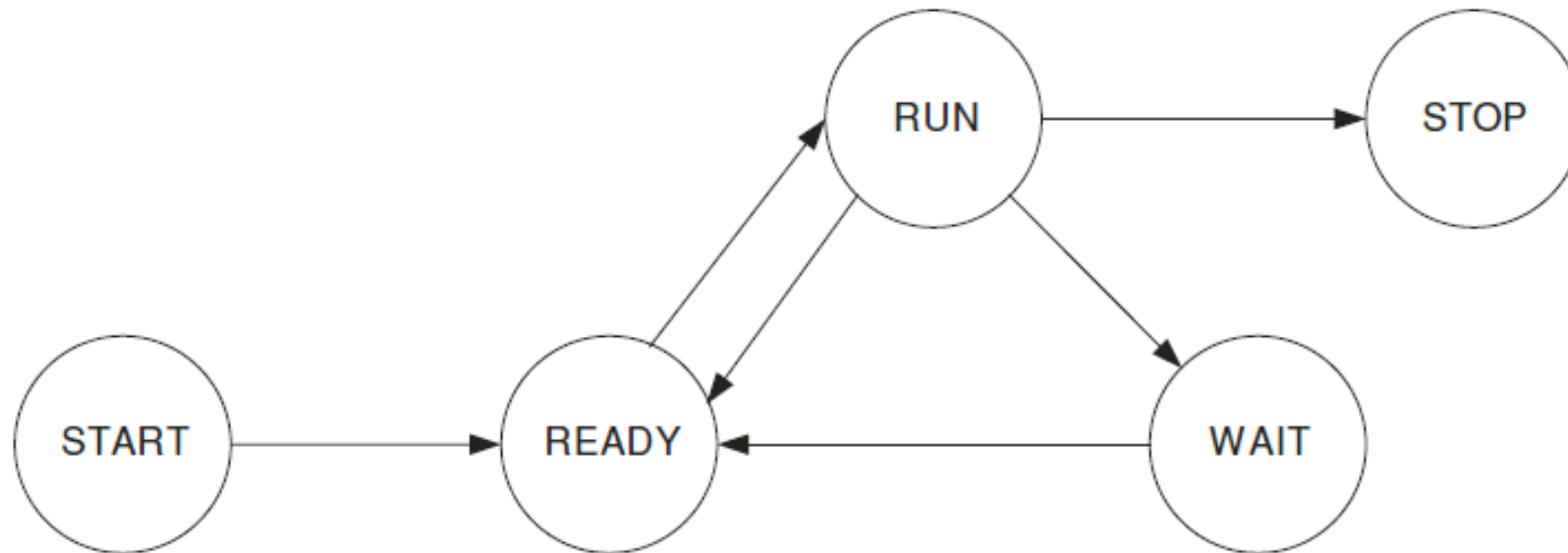
- **Proces** je program ili deo programa u stanju izvršavanja, zajedno sa svim resursima koji su potrebni za rad. Drugim rečima:
 - Program je **pasivan objekat**, odnosno datoteka na disku.
 - Kada se ta datoteka učitava u memoriju, ona postaje proces, odnosno **aktivan objekat** koji ima svoje resurse, poput registara i memorije.
- To znači sledeće:
 - Tri korisnika koji obavljaju neku aktivnost biće predstavljena sa tri razna procesa.
 - Jedan program, koji je sam sebe razdelio na dva dela radi istovremenog (simultanog) izvođenja, biti predstavljen sa dva razna procesa.
- Sam operativni sistem je takođe sastavljen od niza procesa.
- Svaki proces ima tri fundamentalne memorijska dela, odnosno sekcije:
 - **Programska sekcija** koja se ne menja i koja sadrži programski kod
 - **Stek sekcija** (*stack section*) koja sadrži privremene podatke (parametre za procedure, povratne adrese, itd ...)
 - **Sekcija podataka** (*data section*).

Šta je kontrolni blok, a šta kontekst procesa?

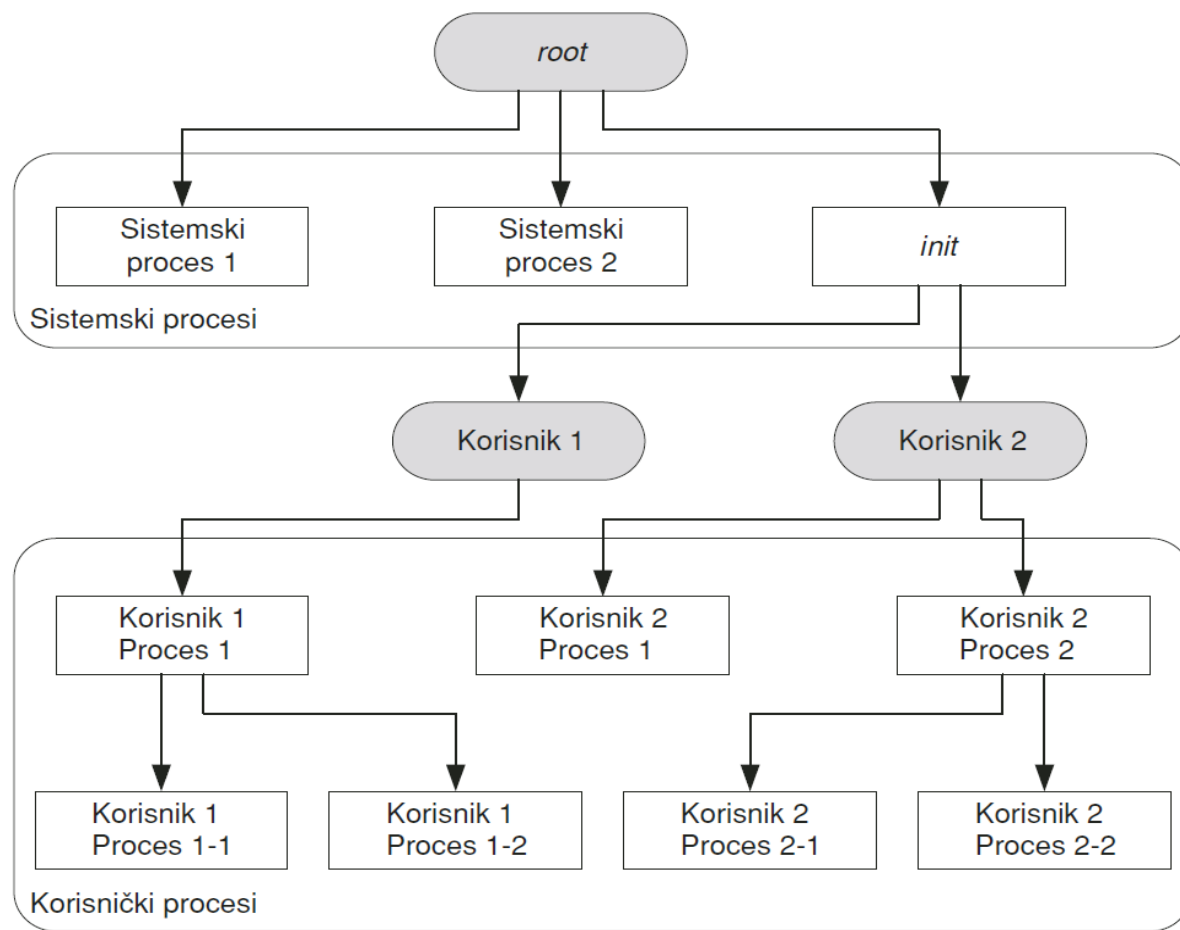
- **Kontrolni blok procesa** (*Process Control Block*, PCB) je memorijska struktura koju generiše OS.
- PCB sadrži osnovne informacije o procesu koje OS koristi za upravljanje tim procesom.
- Zahvaljujući PCB izvršenje programa se može prekidati i nastavljati više puta.
- U informacije iz kontrolnog bloka spadaju:
 - Ime ili jedinstveni identifikator procesa (PID)
 - Kontekst (okolina) procesa
 - Prioritet procesa
 - Informacije vezane za memoriju procesa
 - Lista otvorenih datoteka
 - Status zauzetih ulazno-izlaznih resursa
 - Trenutno stanje procesa.
- **Kontekst procesa** čine podaci koji se čuvaju prilikom oduzimanja procesora, a koje generiše sam hardver: programski brojač, vrednosti registara i pokazivači na deo memorije koji proces koristi.
- Deo PCB u kome se čuva kontekst se još naziva i **hardverski kontrolni blok** procesa.
- PCB se u realnim OS nikad ne nalazi u istom memorijskom području kao i proces.

Životni ciklus jednog procesa

- Konačni automat sa pet stanja.
- Pitanje: da li je tranzicija RUN -> WAIT moguća u jednoprocensnim sistemima?



Hijerarhijska procesna struktura (Linux)



Šta je raspoređivanje procesa?

- Svaki proces ostaje u stanju izvršenja dok mu:
 - Ne istekne vremenski kvantum ili
 - Dok ne dođe u situaciju da mora da čeka na neki događaj (npr. završetak U/I) komande.
- Na prostim sistemima procesor ne radi ništa dok proces čeka.
- Na višeprocesnim sistemima, memorija se puni većim brojem procesa.
 - Kad aktivni proces mora da čeka, OS mu oduzima procesor i dodeljuje ga drugom procesu.
 - Dodela procesora po nekom algoritmu je jedna od najbitnijih funkcija OS-a!
- Tipovi raspoređivanja:
 - Raspoređivanje **bez predpražnjenja**.
 - Raspoređivanje bez prekidanja izvršenja tekućeg procesa.
 - Procesor se može oduzeti samo od procesa koji je završio svoje aktivnosti ili čeka na resurs.
 - Raspoređivanje **sa predpražnjenjem**.
 - Procesor se može oduzeti procesu koji nije završio svoje aktivnosti i nije blokiran!

Koji se algoritmi koriste za raspoređivanje procesa?

- First Come, First Served
- Shortest Job First
- Shortest Remaining Time First (SJF sa pretpražnjenjem)
- Raspoređivanje na osnovu prioriteta
- Round Robin (FCFS sa pretpražnjenjem)

Šta je sinhronizacija procesa?

- **Kooperativni proces** može da utiče na druge procese ili da trpi uticaj drugih procesa.
 - Do toga dolazi usled deljenja podataka koji su smešteni u memoriji ili u datotekama.
- Ukoliko nema sinhronizacije, moguća je **nekonzistentnost** tih podataka!
- Primer:
 - Dva procesa P1 i P2 žele da privremeno sačuvaju neku vrednost na memorijskoj lokaciji A.
 - Proces P1 proverava da li je memorijska lokacija A slobodna.
 - Lokacija A je slobodna → Proces P1 je obavešten da je lokacija A slobodna.
 - Proces P2 proverava da li je memorijska lokacija A slobodna.
 - Lokacija A je slobodna → Proces P2 je obavešten da je lokacija A slobodna.
 - Proces P1 upisuje podatak na memorijsku lokaciju A.
 - Proces P2 upisuje drugi podatak na memorijsku lokaciju A.
 - Proces P1 čita **pogrešan** podatak sa memorijske lokacije A!
- Da bi se izbegle slične situacije, OS mora da obezbedi mehanizme za očuvanje konzistentnosti podataka, odnosno mehanizme **sinhronizacije procesa**.

Primer sinhronizacije: problem proizvođač – potrošač

- **Proivodač** je proces koji proizvodi informacije.
- **Potrošač** je proces koji ih troši (koristi).
- Ta dva procesa mogu da rade konkurentno ukoliko postoji **deljivi bafer** koji će puniti proizvođač, a prazniti potrošač.
- **Pravila sinhronizacije:**
 - Potrošač ne može uzeti ništa iz bafera ako proizvođač to prethodno nije stavio u bafer.
 - Ukoliko je bafer pun, proizvođač ne može ubaciti informaciju dok potrošač najpre ne uzme nešto iz bafera.

Primer sinhronizacije: problem proizvođač – potrošač

- **Deljivi bafer** se može realizovan kao cirkularna struktura koja se sastoji od N elemenata $[0, N-1]$.
 - Pokazivači **in** i **out** određuju prvo slobodno i prvo puno mesto u baferu.
 - Bafer je prazan kada je **in=out**.
 - Bafer je pun kada **(in+1) mod N = out**.

```
# define N 10
typedef struct {
    /* Promenljive koje čine jedan element bafera, odnosno jednu informaciju */
} item;
item buffer[N] // Bafer veličine N
int in = 0; // Prvo slobodno mesto
int out = 0; // Prvo zauzeto mesto
```

Primer sinhronizacije: problem proizvođač – potrošač

- Proizvođač:

```
item next_produced;
while (1) {
    /* Proizvođač proizvodi informaciju */
    while (((in+1)%N) == out);
    /* Sinhronizacija (bafer je pun).
       Proizvođač čeka da potrošač nešto
       uzme iz bafera */
    buffer[in] = next_produced;
    in = (in+1) % N;
}
```

- Potrošač:

```
item next_consumed;
while (1) {
    while (in == out);
    /* Sinhronizacija (bafer je prazan).
       Potrošač čeka se da proizvođač
       nešto stavi u bafer */
    next_consumed = buffer[out];
    out = (out+1) % N;
    /* Potrošač konzumira informaciju */
}
```

- Ovo rešenje je korektno samo u slučaju da se u baferu mogu naći najviše $N-1$ informacija.
- Za slučaj bafera u kome se mogu naći svih N informacija mora se obezbediti dodatna sinhronizacija.

- Sistem se sastoji od **konačnog broja resursa** koje koriste konkurentni procesi.
- Svaki resurs se može sastojati od jedne ili više identičnih **instanci**.
 - Primer: server ima četiri procesora.
 - Procesor je resurs sa četiri instance.
- Ako resurs ima više instanci a proces zahteva jednu, alokacija bilo koje slobodne instance će zadovoljiti potrebe procesa.
- U normalnom režimu rada proces može da koristi resurs samo na sledeći način:
 - **Zahtev** (*request*). U ovoj fazi proces zahteva resurs.
 - Ako zahtev za dodelom resursa ne može da se ispuni trenutno, proces mora da čeka dok se resurs ne oslobodi.
 - **Korišćenje** (*use*). U ovoj fazi proces je dobio resurs i može ga slobodno koristiti.
 - **Oslobađanje** (*release*). Nakon korišćenja resursa proces mora da oslobodi resurs.
- Drugim rečima: proces mora zahtevati resurs pre korišćenja i otpustiti resurs nakon korišćenja.

- U višeprocennoj okolini više procesa se mogu međusobno takmičiti za konačan broj resursa.
- Proces **P1** koji zahteva **neraspoloživ resurs R1** ulazi u stanje WAIT i postaje blokiran.
- Pitanje: da li blokirani proces **P1** može zauvek ostati u tom stanju?
 - Ova pojava je moguća.
 - Jedan proces može zahtevati više različitih resursa.
 - Scenario:
 - Pre ulaska u stanje WAIT procesu **P1** je dodeljen drugi resurs **R2**.
 - Resurs **R2** ostaje neraspoloživ za druge procese.
 - Resurs **R1** je dodeljen na korišćenje drugom procesu **P2**.
 - Proces **P2** u toku vremena prelazi u stanje čekanja na neraspoloživ resurs **R2**.
- U ovoj situaciji niko ne oslobađa svoje resurse a traži nove – procesi ostaju zaglavljani.
- Takva situacija naziva se **zastoj** (deadlock).
 - Zastoj treba izbeći.
 - Sistem doveden u stanje zastoja se mora oporaviti.

- Primeri zastoja koji nisu vezani za računarski sistem:
 - Dve osobe, od kojih se jedna penje uz merdevine a druga spušta niz merdevine.
 - Dva voza koja se kreću jedan prema drugom istom šinom.
 - Zvezdasta raskrsnica u kojoj lako dolazi do zastoja.
 - ... i varijacije na temu.

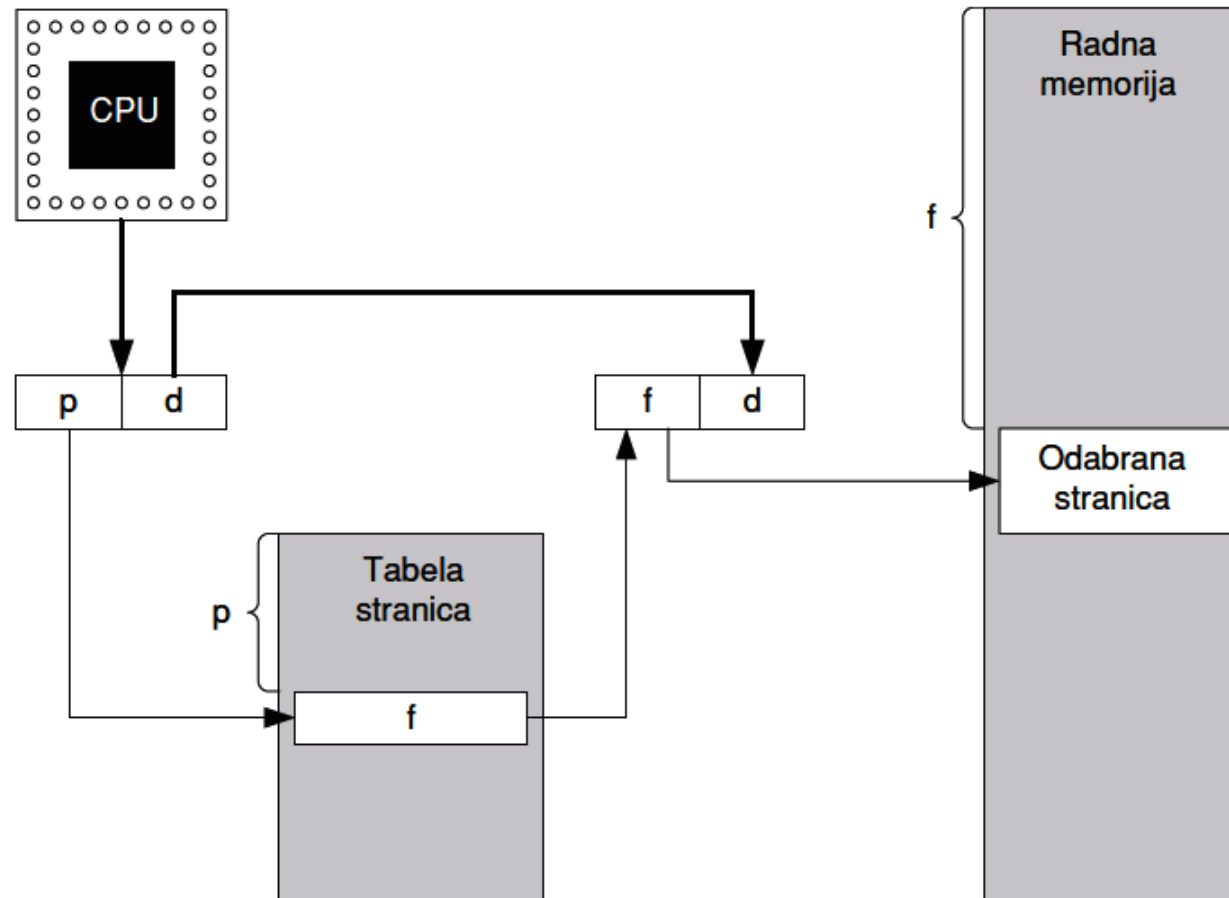
Šta je neophodno da do zastoja dođe?

- **Međusobno isključenje.**
 - Samo jedan proces u jednom trenutku može koristiti resurs ili jednu njegovu instancu.
 - Drugi proces koji zahteva taj resurs (instancu) mora da čeka dok se resurs ne oslobodi.
- **Nema pretpražnjenja** (otimačine resursa).
 - Resurs se ne može nasilno oduzeti i predati drugom procesu.
 - Proces koji ga koristi mora da završi posao i oslobodi resurs.
- Uslov **zadržavanja resursa i čekanja na drugi** (*hold and wait*).
 - Proces drži jedan resurs.
 - Proces istovremeno čeka na dobijanje resursa koga koristi neki drugi proces.
- **Kružno čekanje** (*circular wait*).
 - Postoji skup procesa $\{P_0, P_1, \dots, P_n\}$ koji čekaju na resurse u kružnom poretaku.
 - Proces P_0 čeka na resurs koga drži proces P_1 .
 - Proces P_1 čeka na resurs koga drži proces P_2 ...
 - Proces P_{n-1} čeka na resurs koga drži proces P_n .
 - Proces P_n čeka na resurs koga drži proces P_0 .

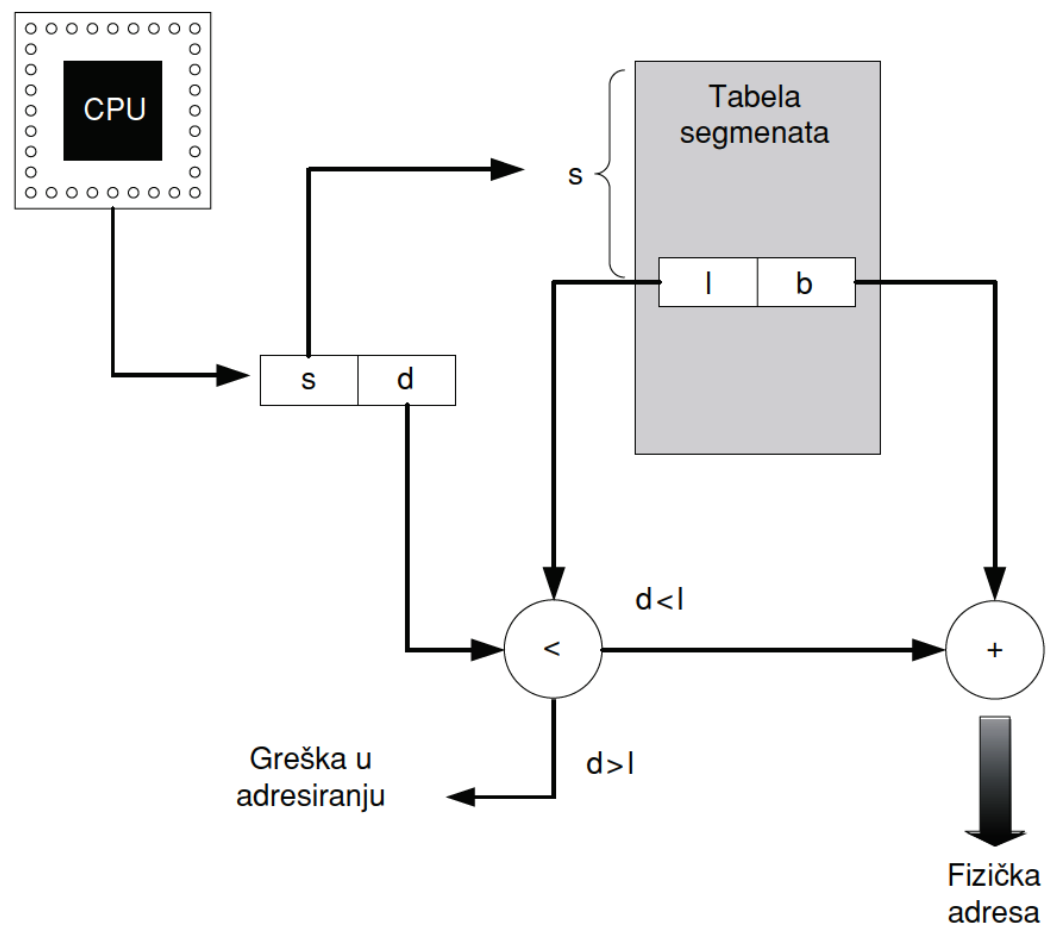
- Memorija je niz memorijskih reči od kojih svaka ima jedinstvenu adresu.
- Prilikom izvršavanja procesa:
 - Procesor na osnovu vrednosti programskog brojača čita instrukcije iz memorije.
 - Instrukcije u toku izvršenja mogu zahtevati:
 - Čitanje podataka sa drugih lokacija.
 - Upis podataka na druge memorijske lokacije.
- Pitanja:
 - Da li proces bez memorije može nešto da uradi?
 - Da li može da se izvrši?
 - Da li ima logike izvršavati proces sa diska?
- Odgovori na ova pitanja kažu da je fizička memorija pored procesora fundamentalni deo računarskog sistema!

- Sloja za upravljanje memorijom (*memory manager-a*):
 - Vodi računa o korišćenju memorije.
 - Dodeljuje memoriju procesima kad je zatraže.
 - Oslobađa memoriju od procesa kad završe svoje aktivnosti.
 - Vršiti razmenu između memorije i diska (kada glavna memorija nije dovoljno velika da sadrži sve procese).
- *Memory manager* takođe:
 - Razdvaja fizički i logički adresni prostor.
 - Prevodi relativne (relokatibilne) adrese u fiksne (vezivanje adresa).
 - Obavlja relokaciju (kompakciju, odnosno defragmentaciju operativne memorije).
 - ...

Šta je straničenje?

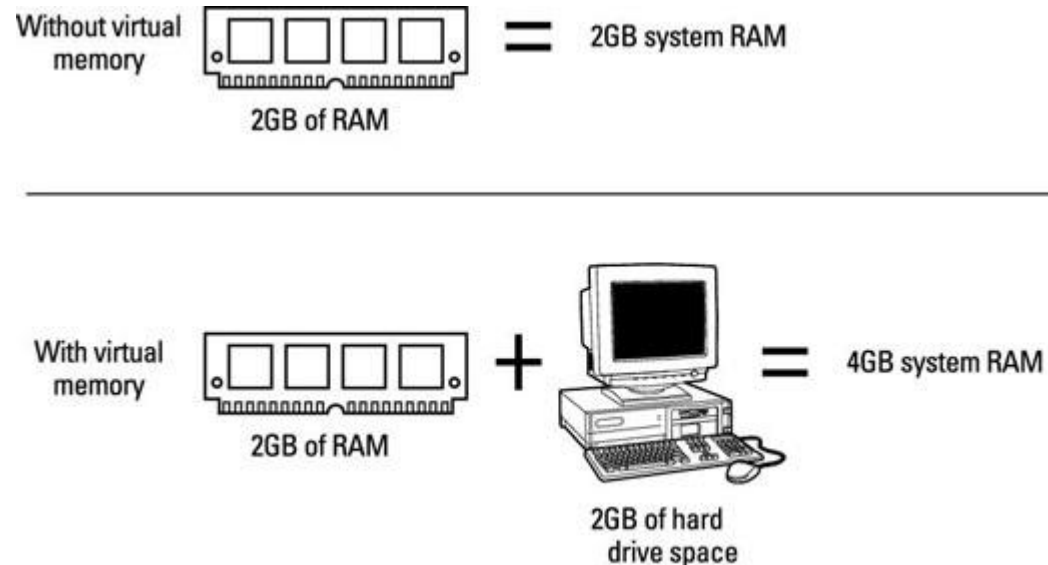


Šta je segmentacija?

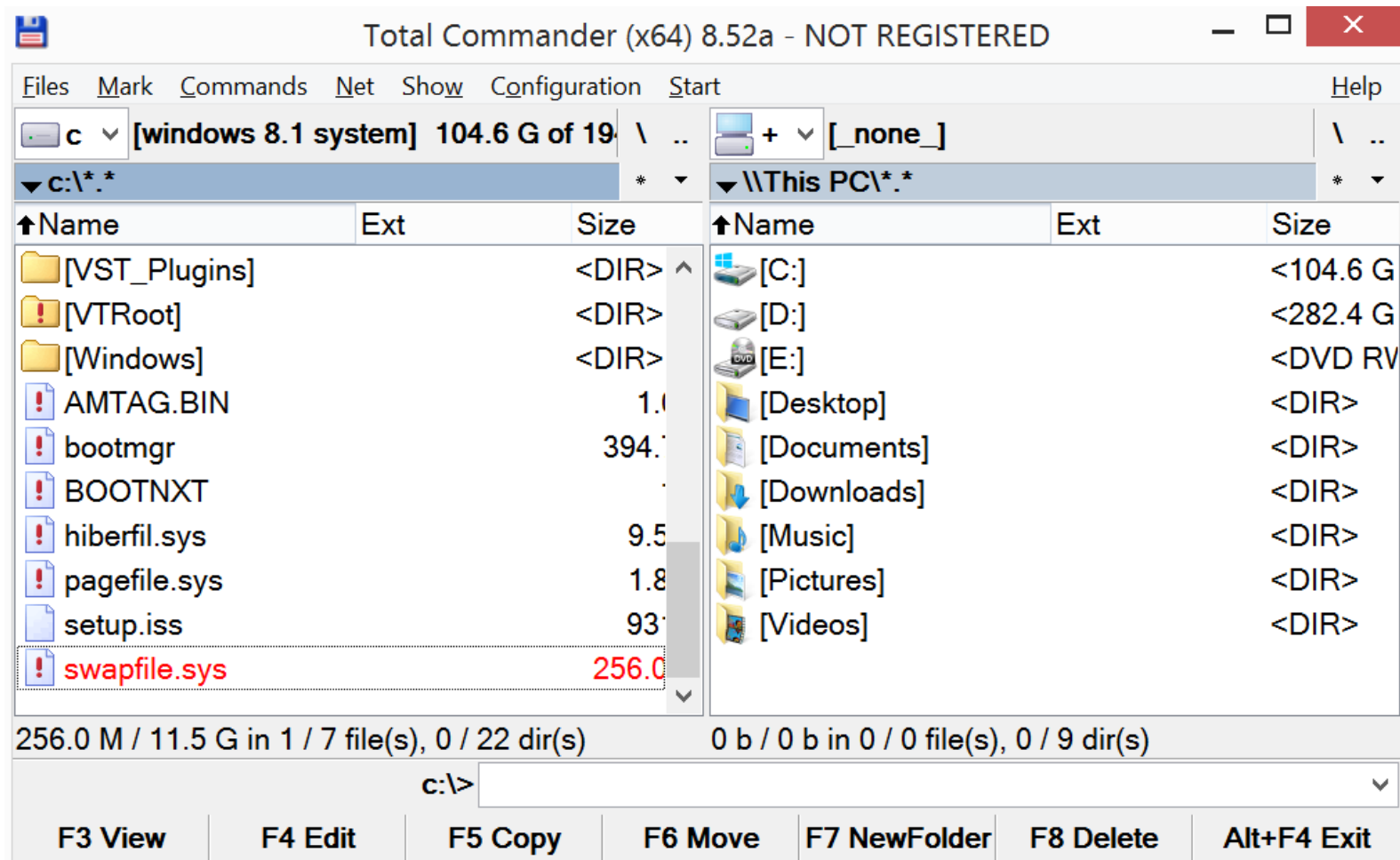


Šta je virtuelna memorija?

- Tehnika koja dozvoljava izvršavanje procesa čiji delovi mogu biti smešteni na diskovima.
- Virtuelna memorija:
 - Formira apstrakciju (logička memorija = operativna memorija + sekundarna memorija).
 - Omogućava izvršenje programa većih od same fizičke memorije.
 - Dozvoljava smeštaj znatno većeg broja procesa u memoriju (konkretno, delova procesa).



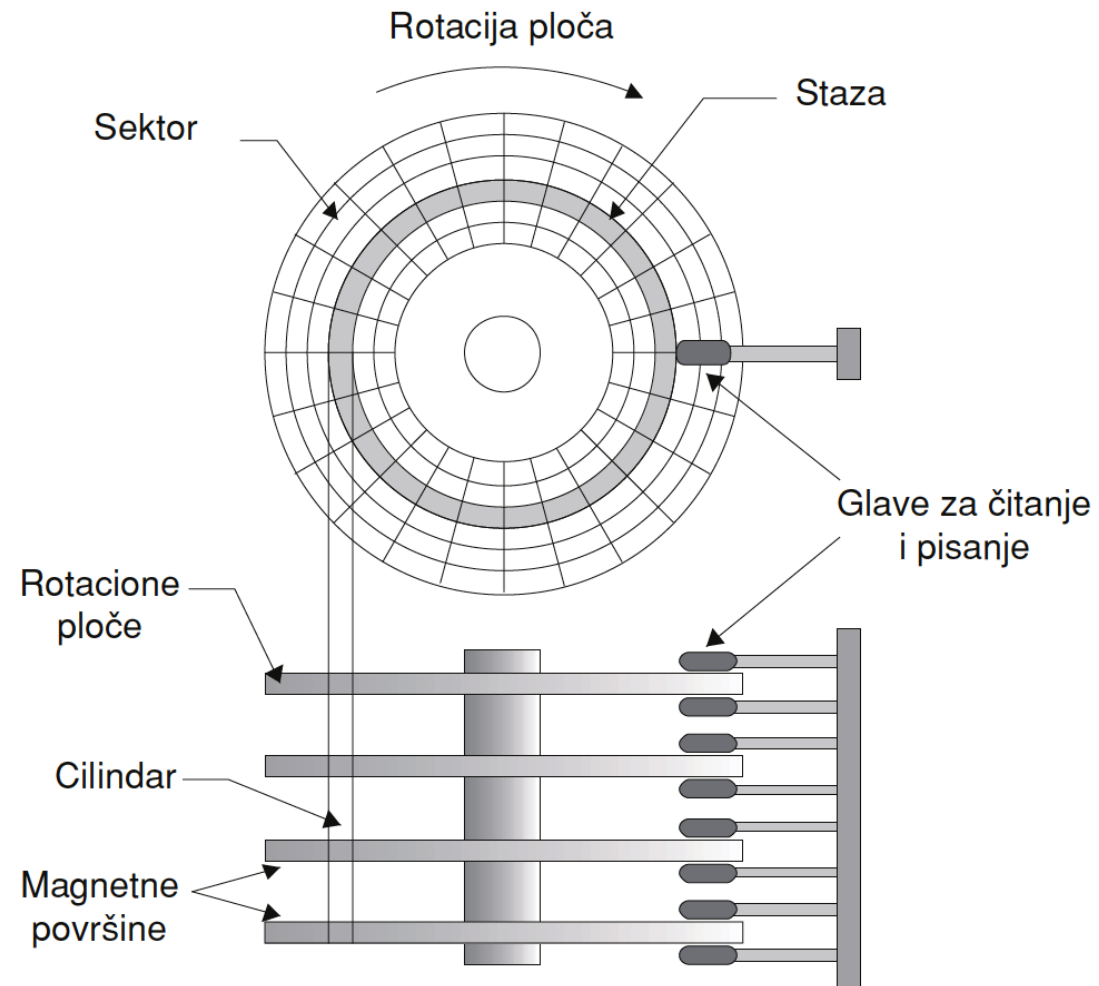
Gde je swap smešten na Windows-u?



Šta je hard disk a šta disk kontroler?

- Skup **rotacionih kružnih ploča** koje rotiraju oko zajedničke ose.
- Površine ploča su presvučene **magnetnim materijalom**.
- Svaka površina ima pridruženu **glavu za čitanje i pisanje**.
 - Čitaju ili upisuju podatke sa magnetnih ploča.
 - Linearno se pokreću pomoću sopstvenog servo-sistema.
 - Na taj način im je uz rotaciju ploča omogućen pristup svim delovima magnetne površine.
- Računar i disk komuniciraju putem **disk kontrolera** (*disk controller*).
 - Disk kontroleri pružaju **interfejs ka ostatku računara**.
 - Računar ne mora da zna način rada niti da kontoliše elektro-mehaniku diska.
 - Dodatne funkcije kontrolera:
 - Baferovanje podataka koje treba upisati na disk.
 - Keširanje diskova.
 - Automatsko obeležavanje neispravnih sektora diska.

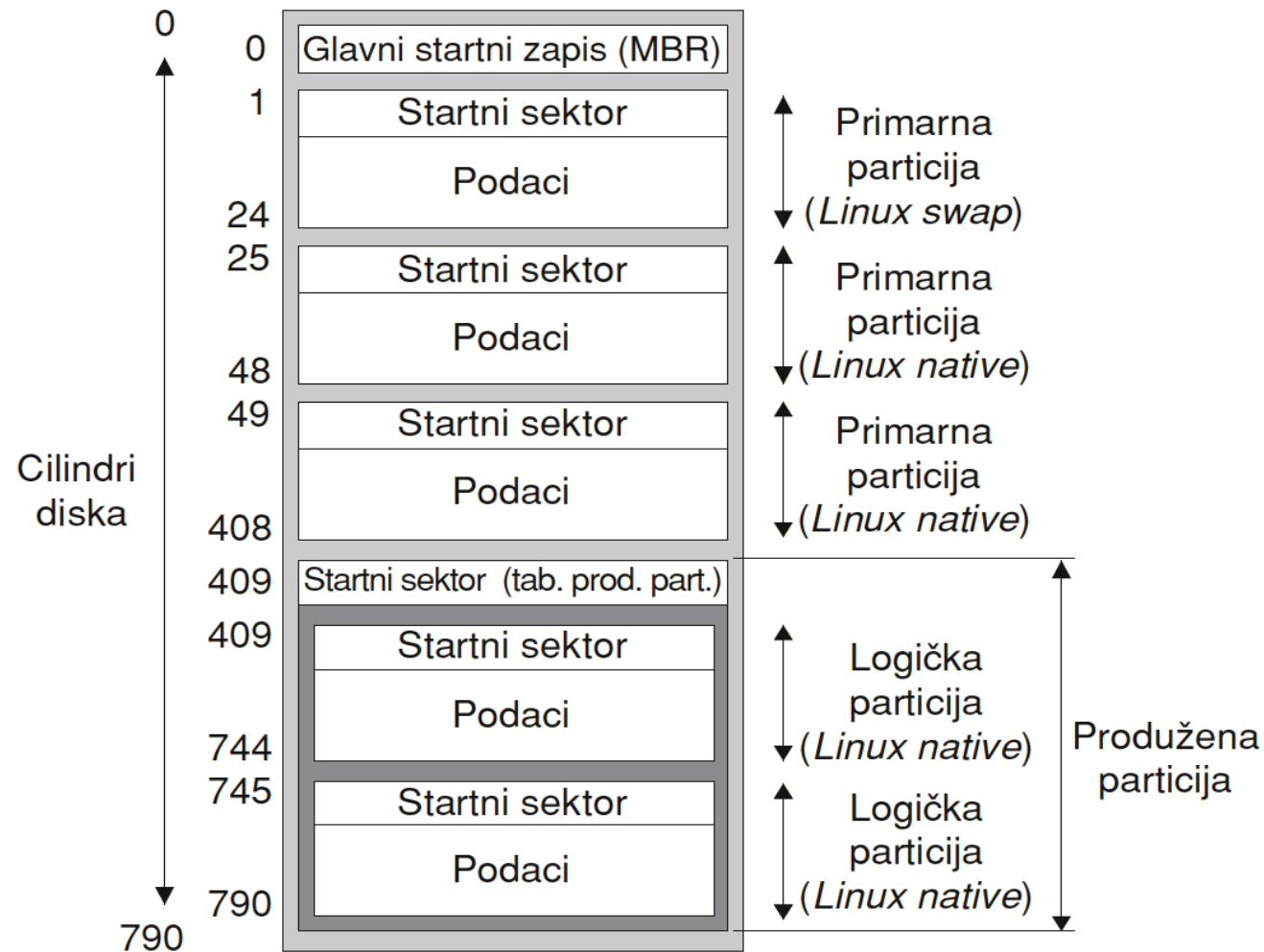
Šta je hard disk a šta disk kontroler?



Šta je hard disk a šta disk kontroler?

- **IDE** (*Integrated Drive Electronics*) tj. **ATA** (*Advanced Technology Attachment*)
 - Kontroler integrisan na matičnoj ploči.
 - /dev/hda, /dev/hdb, ...
 - Dva kanala: *primary* i *secondary*.
 - Na svaki se mogu vezati do dva uređaja u odnosu *master-slave*.
- **SATA** (*Serial ATA*)
 - Kontroler integrisan na matičnoj ploči.
 - /dev/sda, /dev/sdb, ...
- **SCSI**
 - Kontroler NIJE integrisan na matičnoj ploči.
 - Na kontroler je moguće vezati od 7 do 15 uređaja.
 - SCSI uređaji se ne nalaze u master-slave odnosu već se vezuju prema prioritetima.
 - Prioritet uređaja određen je njegovim ID koji se postavlja preko džampera.
 - ID=0 (najviši prioritet), ID=15 (najniži prioriter), ID=7 (rezervisan za SCSI kontroler).

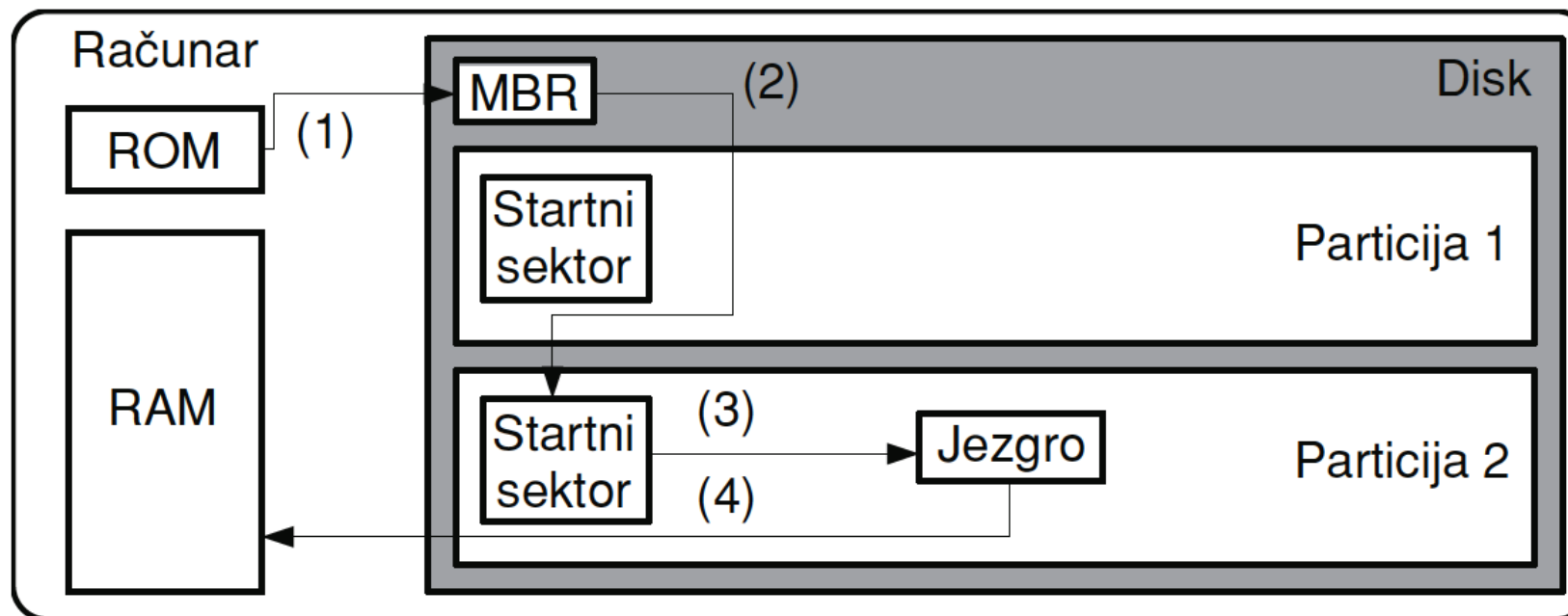
Šta je particija diska (na primeru MBR)?



Šta je bootstrap rutina (na primeru MBR)?

- Kada se računar uključi BIOS izvršava **POST rutinu** (*Power On Self Test*).
- POST = serija testova hardvera.
- **Podizanje sistema** (*boot*) je procedura koja se izvršava u cilju dovođenja sistema u operativno stanje.
- Primer (MBR):
 - Kod upisan u prvom MBR najpre identifikuje **aktivnu particiju** u particionoj tabeli.
 - Zatim se izvršava **kod upisan u boot sektoru** aktivne particije.
 - Program u boot sektoru je zadužen da **pokrene punjenje RAM memorije OS-om**.
 - Napomena:
 - Delovi koda u toj fazi nalaze se na fiksnim područjima diska, a ne u sistemima datoteka.
 - Zašto?
 - U toj fazi nema kernela, pa nemamo podršku za sistem datoteka.
 - Rana faza podizanja operativnog sistema se završava učitavanjem jezgra.

Šta je bootstrap rutina (na primeru MBR)?



Šta je sistem datoteka?

- Diskovi se mogu podeliti na više delova.
 - Delovi diska se nazivaju **particije** (partitions) ili **volumeni** (volumes).
 - Da bi deo diska mogao da se iskoristi za skladištenje podataka, potrebno je na njemu kreirati sistem datoteka.
 - Svaki deo diska predstavlja granice u čijim se okvirima može kreirati sistem datoteka.
- **Sistem datoteka** je skup metoda i struktura podataka koje operativni sistem koristi za čuvanje datoteka.
- Sistem datoteka čine:
 - **Zaglavlje** (podaci neophodni za funkcionisanje sistema datoteka)
 - **Meta podaci** (strukture za organizaciju podataka na medijumu)
 - **Podaci** (datoteke i direktorijumi).
- Zaglavlje i meta-podaci čine premašenje sistema, ali bez njih sistem datoteka ne može da funkcioniše.

1. B. Đorđević, D. Pleskonjić, N. Maček (2005): Operativni sistemi: teorija, praksa i rešeni zadaci. Mikro knjiga, Beograd.
2. R. Popović, I. Branović, M. Šarac (2011): Operativni sistemi. Univerzitet Singidunum, Beograd. *
3. B. Đorđević (2017): radni materijali iz predmeta “tehnike virtuelizacije i računarstvo u oblaku”, Visoka škola elektrotehnike i računarstva strukovnih studija, Beograd.

* Može se besplatno preuzeti sa portala: www.singipedia.com

Pitanja su dobrodošla.