Osnovni koncepti operacijskih sustava

Uvodna razmatranja

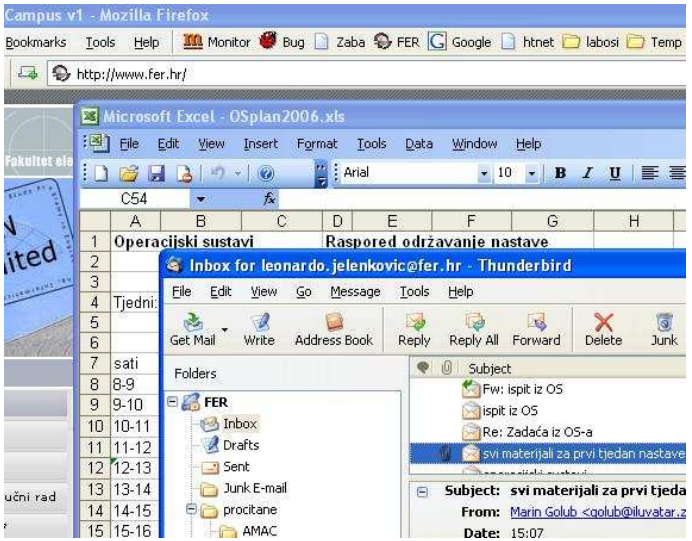
Uvod

- Što je to: operacijski sustav?
  - podrška izvođenju raznim *primjenskim programima*
  - *skup programa* koji omogućuju provođenje radnih zahvata na računalu: **izvođenje *operacija računala***
    - po tome je dobio i ime *operacijski sustav*
- Operacijski sustav je dio ***računalnog sustava***:

Sklopovlje

Programi

Operacijski sustav



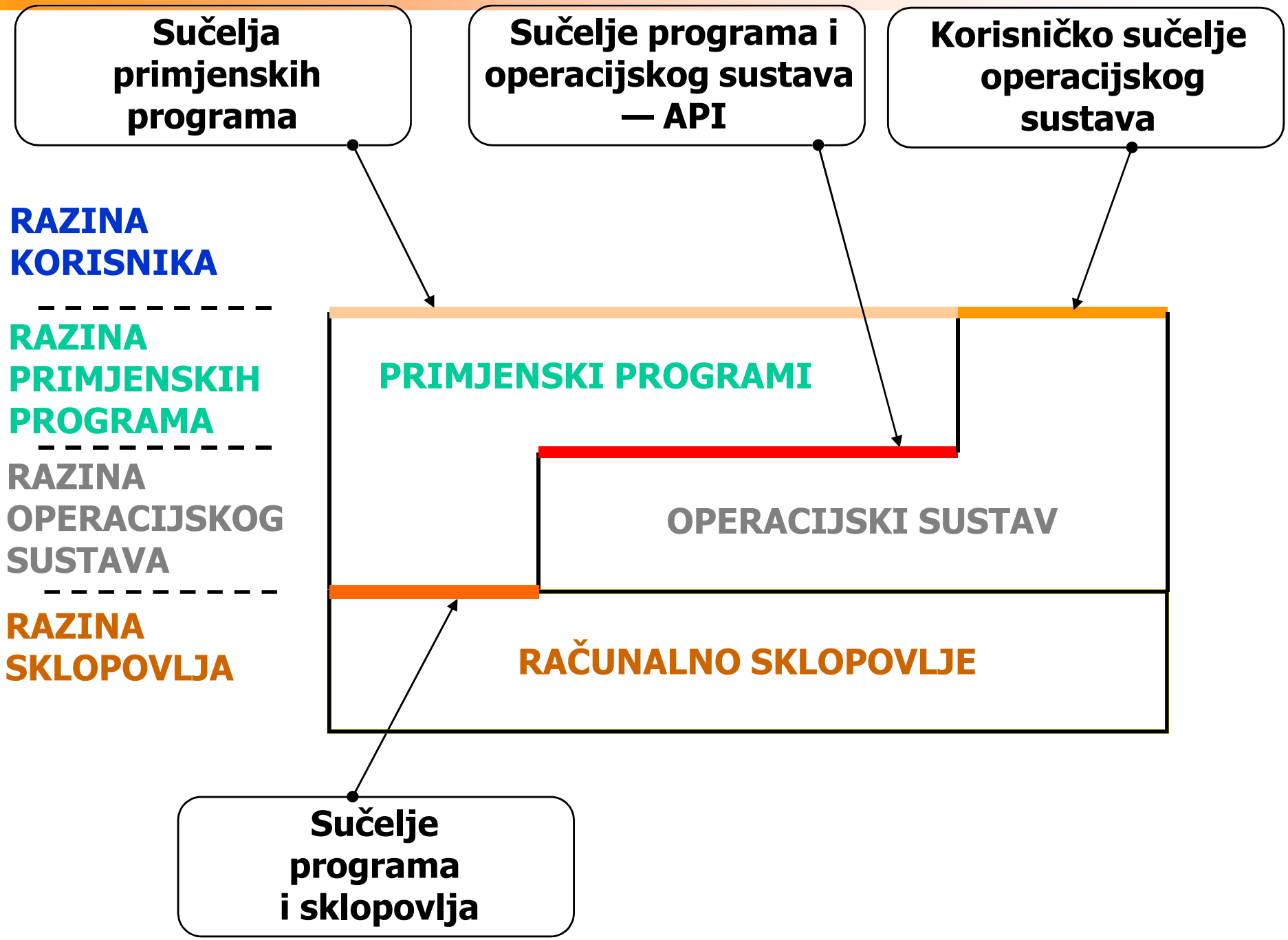
+ Korisnik

Povezivanje elemenata sustava: korištenjem sučelja

- sve operacije OS-a moraju biti dohvatljive preko sučelja
- sučelja definiraju
  - kako se operacije pozivaju (parametri koji se šalju)
  - oblik i interpretacija povratne vrijednosti (ili statusa)

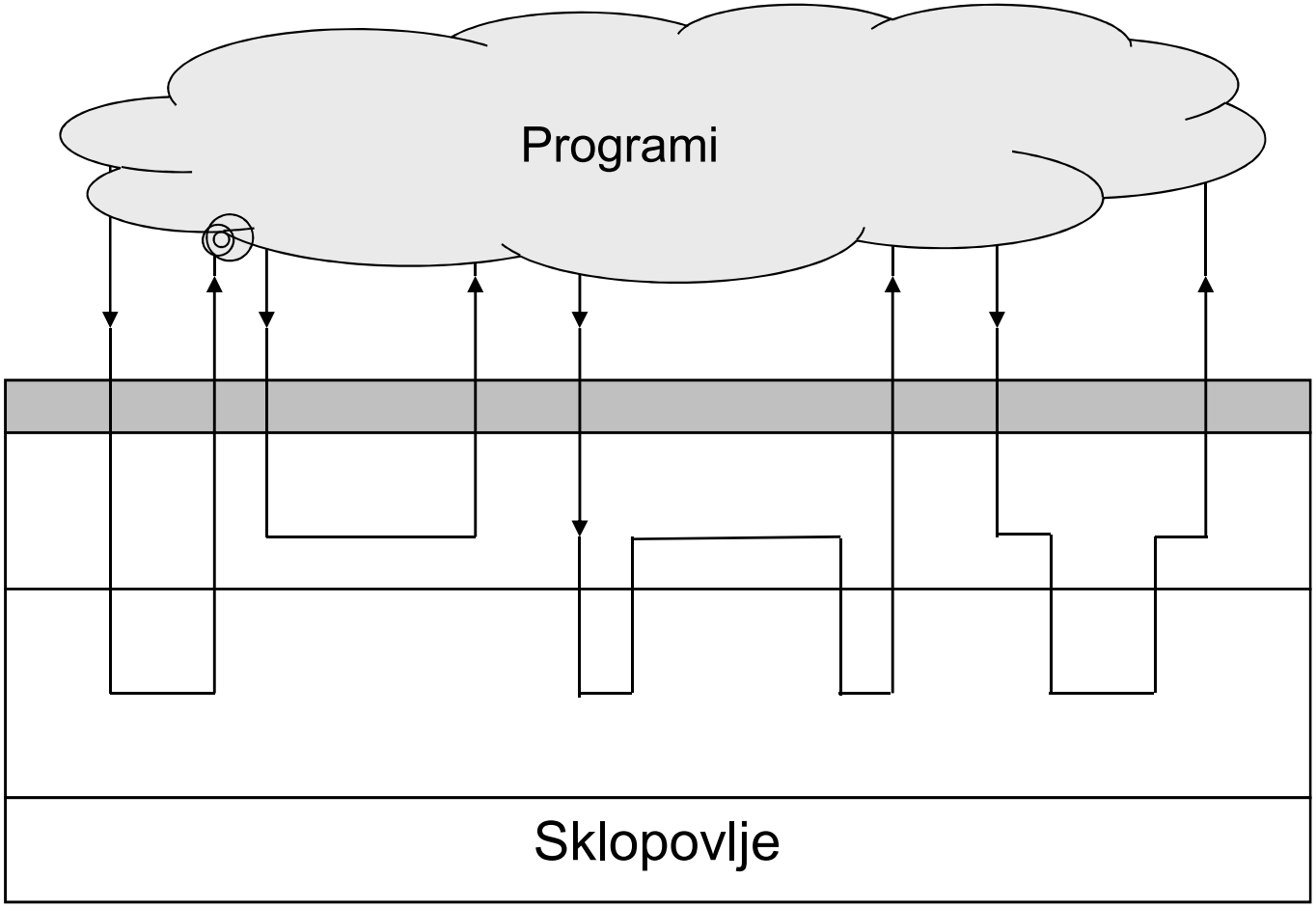
- OS ima sučelja koja koriste:
  - *programi* (API)
  - *korisnik*, najčešće grafičko (GUI)

- I *Programi* imaju sučelja za korisnika


Sučelja primjenskih programa
Sučelje programa i operacijskog sustava — API
Korisničko sučelje operacijskog sustava
RAZINA KORISNIKA
RAZINA PRIMJENSKIH PROGRAMA
RAZINA OPERACIJSKOG SUSTAVA
RAZINA SKLOPOVLJA
PRIMJENSKI PROGRAMI
OPERACIJSKI SUSTAV
RAČUNALNO SKLOPOVLJE
Sučelje programa i sklopovlja

Slojeviti prikaz operacijskog sustava

(s aspekta poziva njegovih operacija)



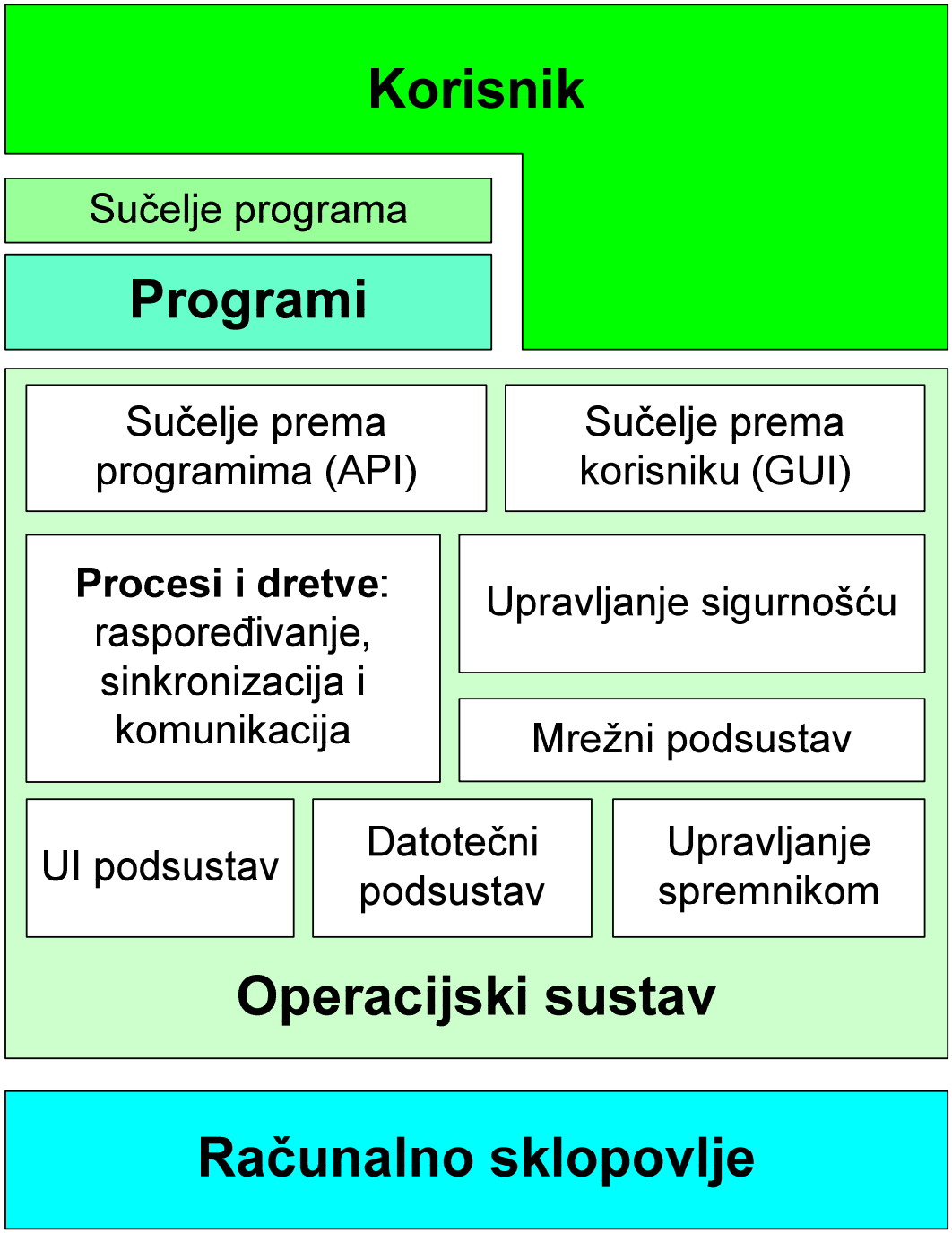
API (sučelja)

funkcije sustava (biblioteke)

jezgra OS-a

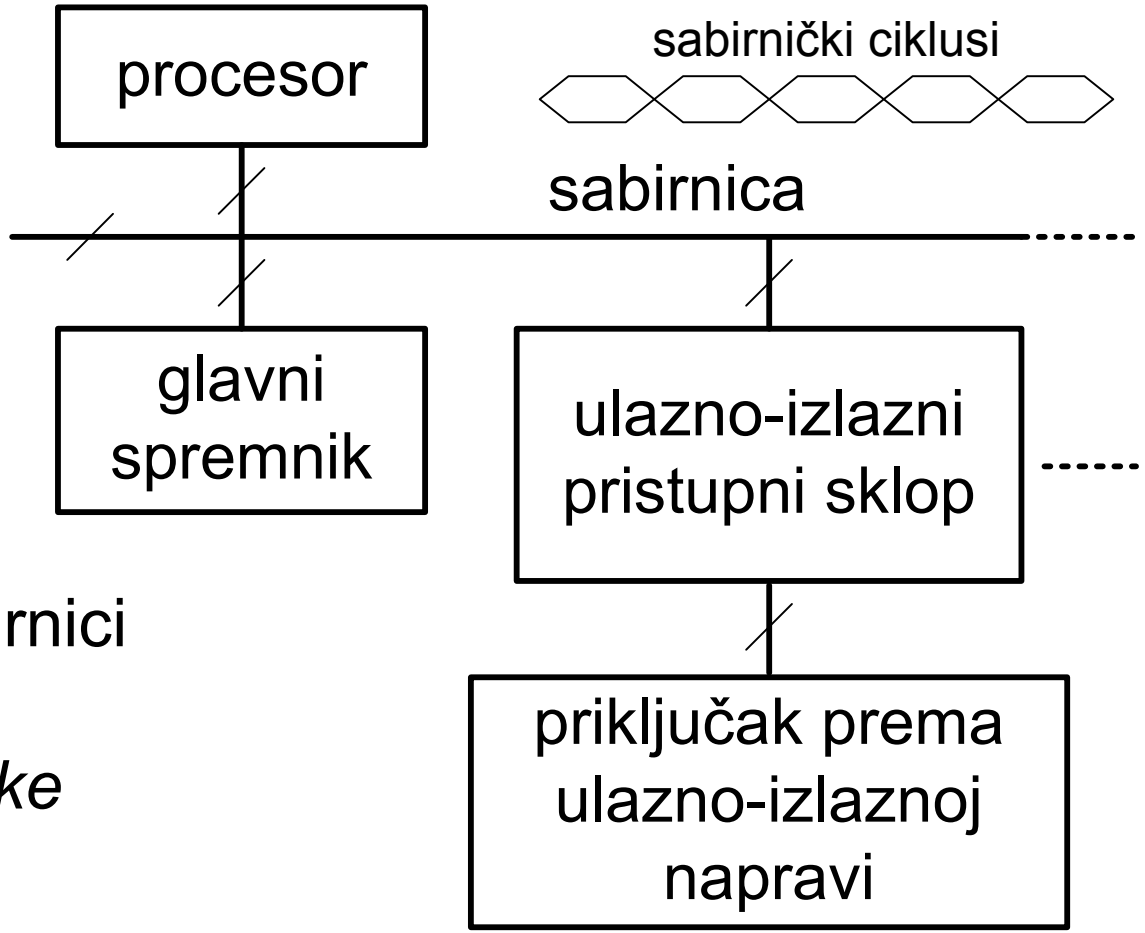
- **Jezgra OS-a**
  - sadrži samo osnovne podatke i instrukcije (što se smatra “osnovnim”?)
  - provodi osnovne operacije (npr. upravlja UI napravama (prekidima), dretvama/procesima, upravlja spremnikom…)
- **Funkcije sustava**
  - koriste jezgru da bi ostvarile potrebne funkcionalnosti operacijskog sustava (npr. ispis na zaslon, čitanje datoteke, korištenje pisača, …)
  - funkcije sustava nude se programeru kroz API sučelje, a korisniku kroz GUI
- *Kako programer koristi (treba koristiti) te funkcije?*
- *Što se zbiva prilikom poziva (jezgrinih) funkcija?*
  - nastavak predavanja pokušava dati odgovor na ovakva pitanja
  - započinje se s kratkim pregledom osnovnih operacija računalnih sustava na niskoj razini (sklopovlje)

Računalni sustav



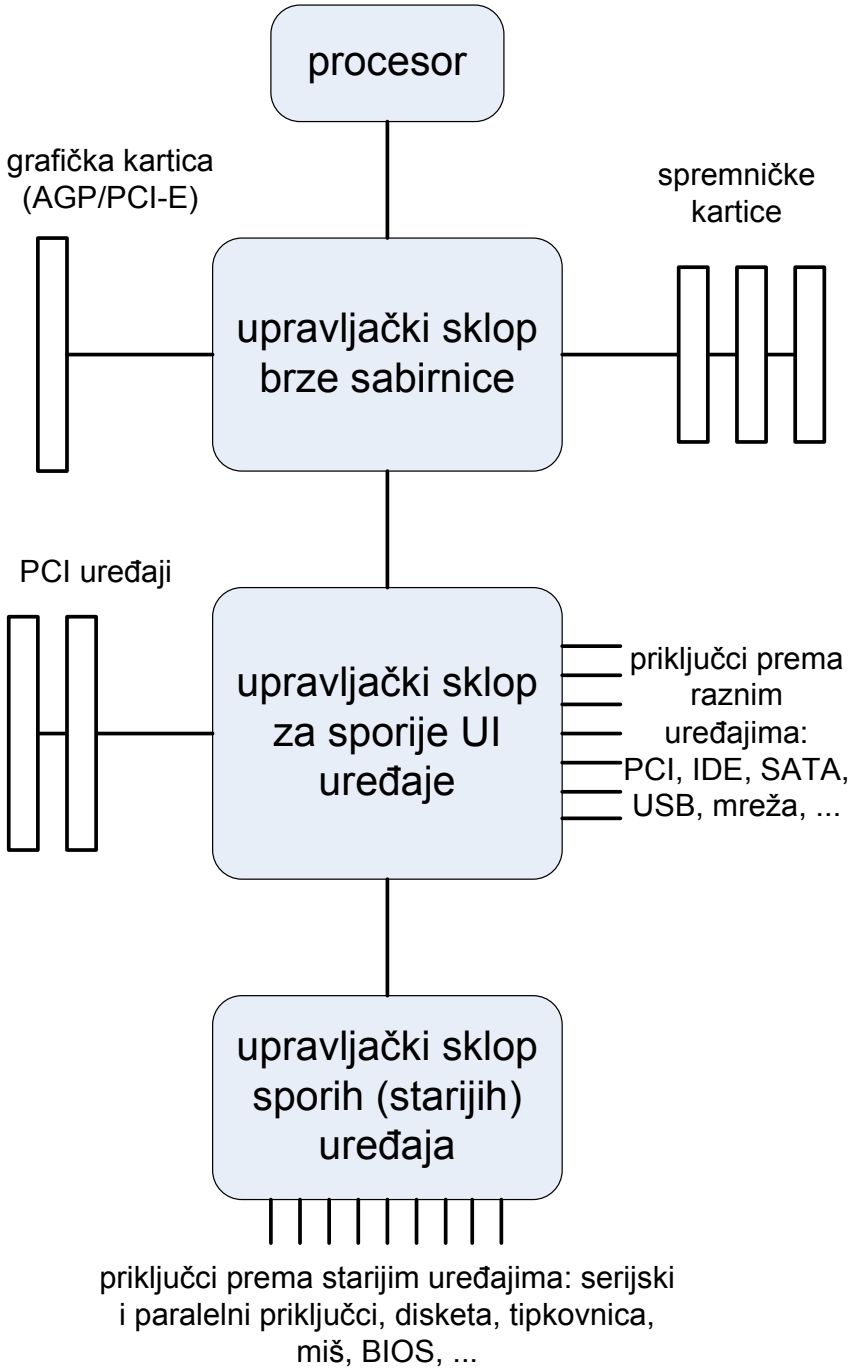
Sabirnička građa računala (pojednostavljeno)

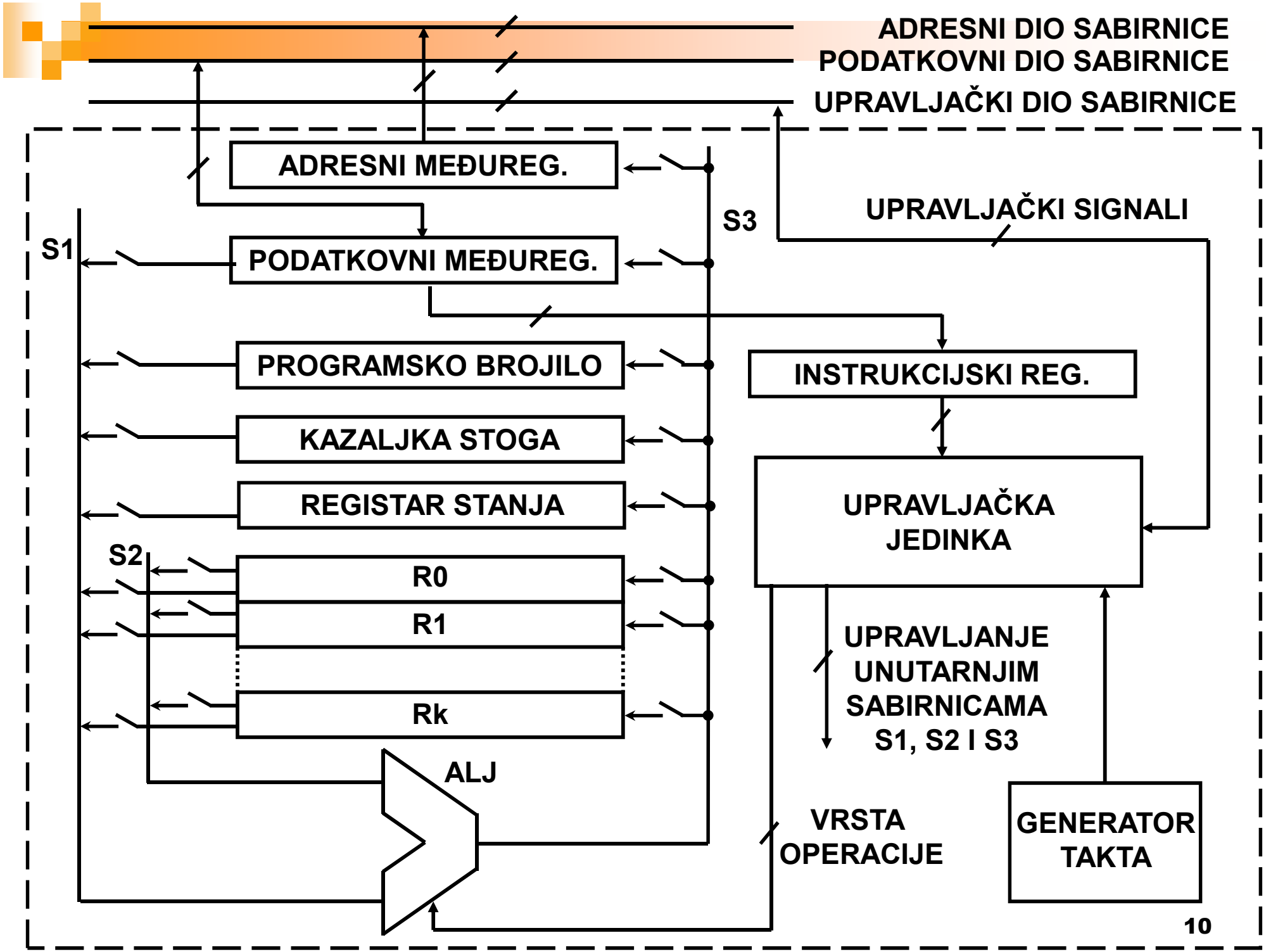
Sve potrebne razmjene podataka, instrukcija i upravljačkih signala obavljaju se preko zajedničkih vodiča sabirnice.



Vrijeme se na sabirnici dijeli na sabirničke periode ili *sabirničke cikluse*.

Primjer arhitekture računala danas (hijerarhija sabirnica)




ADRESNI DIO SABIRNICE
PODATKOVNI DIO SABIRNICE
UPRAVLJAČKI DIO SABIRNICE
ADRESNI MEĐUREG.
PODATKOVNI MEĐUREG.
S1
S2
S3
PROGRAMSKO BROJILO
KAZALJKA STOGA
REGISTAR STANJA
R0
R1
Rk
ALJ
UPRAVLJAČKI SIGNALI
INSTRUKCIJSKI REG.
UPRAVLJAČKA JEDINKA
UPRAVLJANJE UNUTARNJIM SABIRNICAMA S1, S2 I S3
VRSTA OPERACIJE
GENERATOR TAKTA
10

Procesor se može promatrati kao automat koji, nakon uključivanja, trajno izvodi instrukciju za instrukcijom strojnog programa.

```
ponavljati {
    dohvatiti iz spremnika instrukciju na koju pokazuje
    programsko brojilo;
    dekodirati instrukciju, odrediti operaciju koju
    treba izvesti;
    povećati sadržaj programskog brojila tako da
    pokazuje na sljedeću instrukciju;
    odrediti odakle dolaze operandi i kuda se
    pohranjuje rezultat;
    operande dovesti na aritmetičko-logičku jedinku,
    izvesti zadanu operaciju;
    pohraniti rezultat u odredište;
}
dok je (procesor uključen);
```

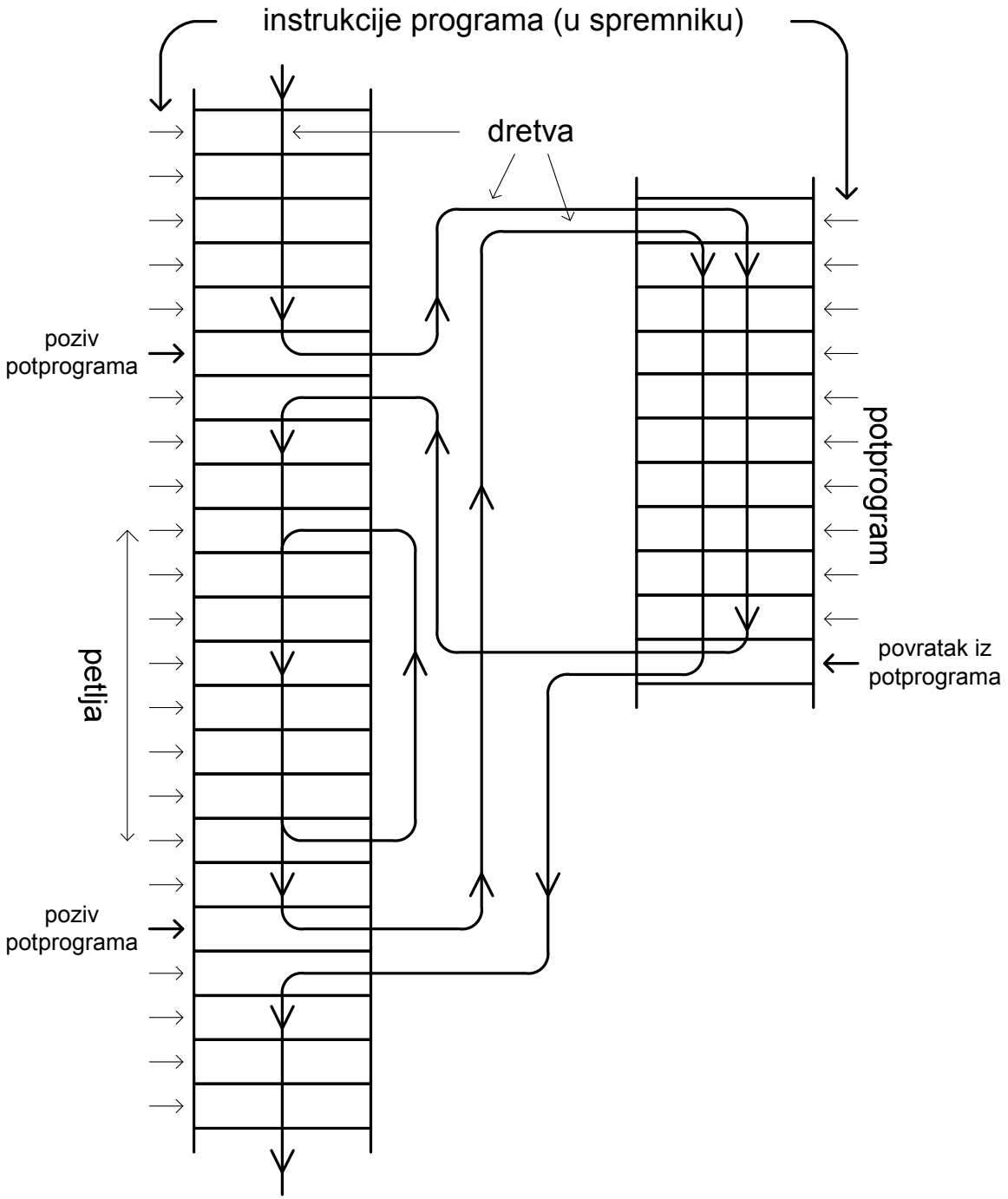
Instrukcijska dretva

- Sadržaj spremnika možemo podijeliti na:
  - dio u kojem je smješten program (niz strojnih instrukcija)
  - dio rezerviran za stog
  - dio za smještanje podataka.

- Pri izvođenju nekog programa procesor izvodi niz instrukcija – *instrukcijsku dretvu* – kraće: *dretvu*

- osnovna razlika između *niza instrukcija* i *dretve*:
  - niz instrukcija u spremniku je povezan redoslijedom spremničkih lokacija (jedna iza druge u susjednim spremničkim lokacijama)

  - niz instrukcija koje čine *dretvu* povezan je vremenskim slijedom (izvođenja)


instrukcije programa (u spremniku)
dretva
poziv potprograma
potprogram
petlja
povratak iz potprograma
poziv potprograma

Računalni proces

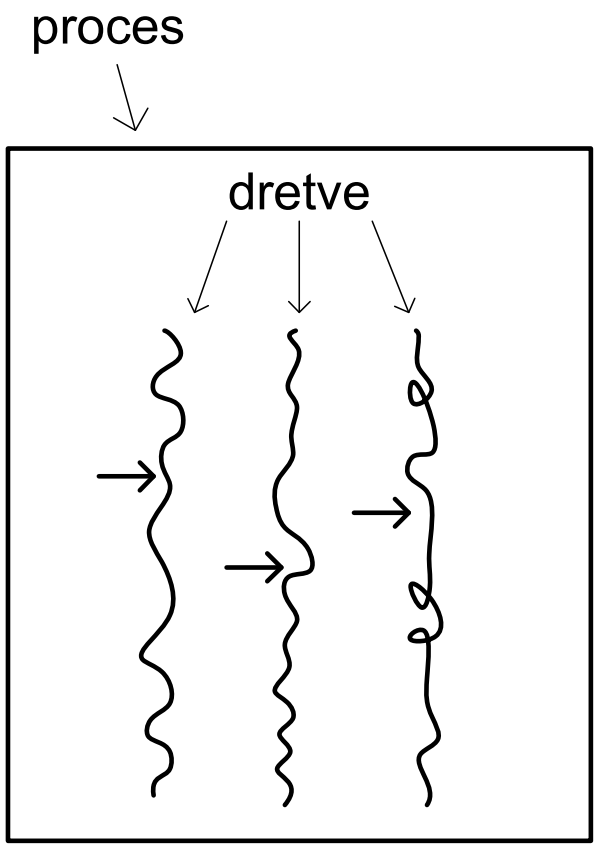
- *program*: statična tvorevina (na nekom mediju, npr. CD)
- učitavanjem u spremnik, program se “transformira”, dobiva određena sredstva sustava, počinje s izvođenjem: program dobiva dinamička obilježja (njegovo se stanje mijenja s vremenom!)

- *proces*: okolina u kojoj se program izvodi

- operacijski sustav osigurava sva potrebna sredstva za njegovo izvođenje (spremnički prostor, procesorsko vrijeme, pristup UI napravama, …)

- svaki proces se izvodi s barem jednom instrukcijskom dretvom

- proces može imati i više dretvi koje se mogu izvoditi:
  - *paralelno* (istodobno), ako se u sustavu nalazi više procesorskih jedinki
  - *naizmjence (prividno paralelno)*, ako je samo jedna procesorska jedinka raspoloživa



- korištenje više dretvi za obavljanje posla (jednog ili više):
  - ***višedretvenost***

- upravljanje dretvama u višedretvenom sustavu?
  - obilježja dretvi?

Kontekst dretve

- *kontekst* čine *svi registri procesora* koje dretva koristi

- prebacivanje izvođenja s jedne dretve na drugu zahtijeva spremanje konteksta jedne dretve te obnovu konteksta druge, koja onda nastavlja s radom

- postupak zamjene dretvi:
  1. prekid izvođenja trenutne dretve
  2. spremanje konteksta prekinute dretve
  3. odabir iduće dretve za rad (nove dretve)
  4. obnavljanje konteksta nove dretve
  5. nastavak rada u novoj dretvi

- navedeni mehanizam promjene konteksta stvara privid paralelnog rada dretvi (iako je on možda zapravo naizmjenični)

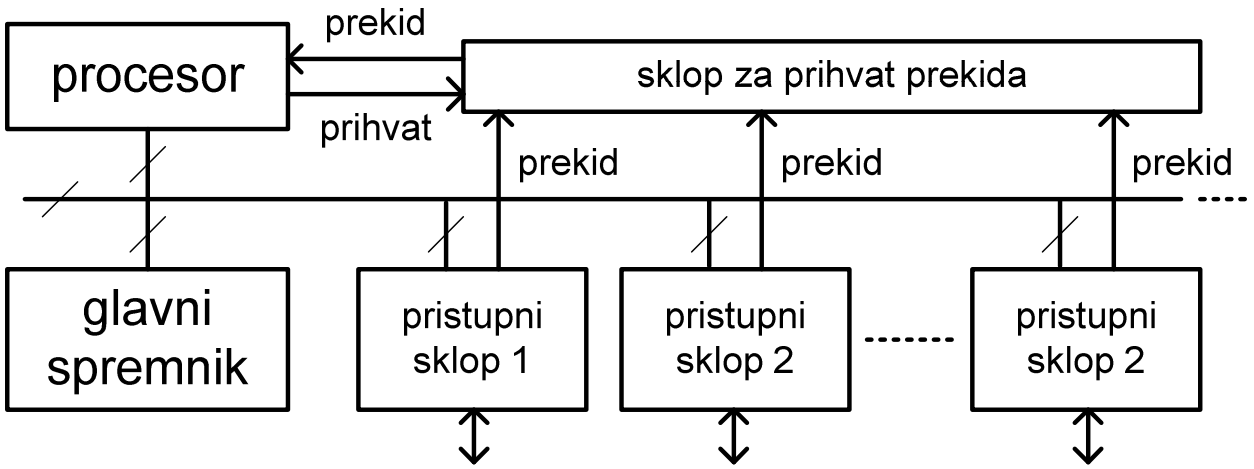
- *zamjena konteksta* je osnovni mehanizam na kojem je izgrađen operacijski sustav!

- kada se promjena zbiva?
  - kako se zamjena dretvi obavlja?

- korištenjem mehanizma *prekida*!

Prekidi

- prekidni podsustav – potpomaže upravljanju UI naprava
  - umjesto da procesor sam provjerava stanje naprava (treba li ih “poslužiti”)
  - naprava sama signalizira spremnost korištenjem *prekida*
- procesor provjerava prisutnost zahtjeva za prekid na kraju svake instrukcije
  - prihvat i obrada prekida uključuju zamjenu konteksta!
  - trenutna se dretva “sprema”
  - za prihvat i obradu prekida se koristi “prekidna dretva”



Primjeri zahtjeva za prekid:

- naprava postaje spremna → posluži ju
- vremenski interval je istekao → zamijeni dretvu drugom
- pokušaj dekodiranja nepostojećeg instrukcijskog koda → prekini trenutnu dretvu (program) – dogodila se nepopravljiva greška

- …
- svi navedeni (i ostali) primjeri traže posebno postupanje – obradu, koja se može ostvariti samo kroz OS

Potrebna podrška upravljanju prekidima:

- OS se mora izvršavati s najvećim privilegijama
  - npr. mora imati dozvole za rad sa svim sklopovljem

- običnoj dretvi treba uskratiti takve dozvole!

Načini rada procesora

- svi netrivijalni procesori imaju više načina rada:
  - korisnički način za obične dretve
  - prekidni način rada
  - sustavski način rada (može se koristiti i prekidni)

- u različitim načinima rada procesor ima različite mogućnosti (instrukcije, registre, spremnički prostor, pristup UI napravama, …)

Po prihvatu prekida, procesor (automatski, u postupku prihvata prekida):

- zabranjuje daljnje prekidanje (drugih naprava)
- prebacuje se u prekidni način rada
- sprema programsko brojilo (PC) na stog sustava (često i još neke registre, npr. registar stanja, kazaljku stoga)
- u programsko brojilo stavlja se adresa procedure za obradu prekida, koja se dobiva tako što je:
  - ugrađena u procesor (za svaki različiti prekid generira se različita unaprijed određena početna adresa za obradu)
  - spremljena u nekom registru procesora (ili se pomoću njega dobiva – izračunava)
- iduća instrukcija je prva instrukcija prekidne procedure!

- u različitim arhitekturama postoje
  - razni načini prihvata prekida
  - različite mogućnosti generiranja zahtjeva za prekid
  - različiti broj prekida (koji se dovode izvan procesora)

- obrada prekida UI naprava se obično obavlja korištenjem *upravljačkih programa* posebno pripremljenih za zadane naprave (engl. *device driver*)

- upravljački programi se zato “registriraju” za određene prekide – operacijski sustav mora omogućiti takav mehanizam

- obrada prekida je “skupa operacija”
  - prekinuta se dretva mora pohraniti – njen kontekst pospremiti (te kasnije obnoviti)
  - promjena konteksta može uključivati i druge operacije
    - npr. rad s priručnim spremnikom – njegovo pražnjenje i punjenje

- nakon zamjene konteksta, obrada prekida može započeti
  - obrada prekida je specifična operacija koja ovisi o napravi
  - teško je unaprijed predvidjeti njeno trajanje!

- po završetku obrade prekida, treba
  - obnoviti kontekst i nastaviti s prekinutom dretvom, ili
  - obnoviti kontekst i nastaviti s drugom dretvom

- Za sustave za rad u stvarnom vremenu (RT sustavi), prekidi predstavljaju poseban problem, jer:
  - obrada prekida nižeg prioriteta može znatno odgoditi obradu značajnijeg prekida
  - zato se u takvim sustavima:
    - obrade prekida dizajniraju da traju kratko
    - svakom prekidu dodjeljuje prioritet te se prekidi obrađuju prema prioritetu
      - nakon detekcije zahtjeva za prekid te njegovim prihvatom prije same obrade, a nakon pohrane konteksta, dozvoljava se prekidanje
        - obrada se obavlja s dozvoljenim prekidanjem!
        - prioritetniji prekidi tako mogu prekinuti obradu onih manjega prioriteta
    - obrada prekida dijeli se na dva dijela: *kratki* i *duži*

- Uobičajeno rješenje u modernim operacijskim sustavima jest da se obrada prekida dijeli na dva dijela:
  - ***kratki*** (uz zabranjeno prekidanje)
    - koji se obavlja u samoj prekidnoj proceduri pri prihvatu prekida
    - najčešće samo sprema parametre za obradu i postavlja zahtjev u red (s prioritetom) ili aktivira dretvu koja će prekid obraditi
  - ***duži*** (uz dozvoljeno prekidanje)
    - obavlja se naknadno, s pridijeljenim prioritetom
      - koristi se zasebna dretva za obradu
      - ili jedna ili više dretvi obrađuju zahtjeve prema prioritetima (ili redu prispijeća)
  - npr.
    - u *Linux*-u: *top half / bottom half (tasklet, softirq, work queue)*
    - u *Win32*: *interrupt servicing routine (ISR) / interrupt servicing thread (IST)*

Programski prekidi

- Kako će obična (korisnička/primjenska) dretva:
  - koristiti UI napravu
  - rezervirati sredstvo za korištenje (npr. objekt u spremniku)
  - zaustaviti svoje izvođenje dok neka druga dretva ne završi
  - izmjenjivati poruke s drugim dretvama
  - …
- Navedene operacije nisu izravno dostupne dretvama jer:
  - nekontrolirani (istovremeni) pristup može izazvati grešku u podacima
  - dretva s greškom može ugroziti cijeli sustav
  - dretva koja treba na nešto čekati ne bi trebala neproduktivno trošiti procesorsko vrijeme
- Očito postoji potreba za mehanizmom kojim će se dretvi omogućiti obavljanje navedenih i sličnih operacija na “siguran” način

- Potrebno je omogućiti dretvi da *pozove* ***funkciju jezgre*** koja će moći napraviti tražene operacije

- Pozivi jezgri se ostvaruju korištenjem
  - **programskog prekida**

- U svakoj arhitekturi postoji instrukcija za izazivanje programskog prekida

- Parametri se u ***jezgrinu funkciju*** prenose ili korištenjem registara ili korištenjem stoga dretve ili korištenjem određenih spremničkih lokacija

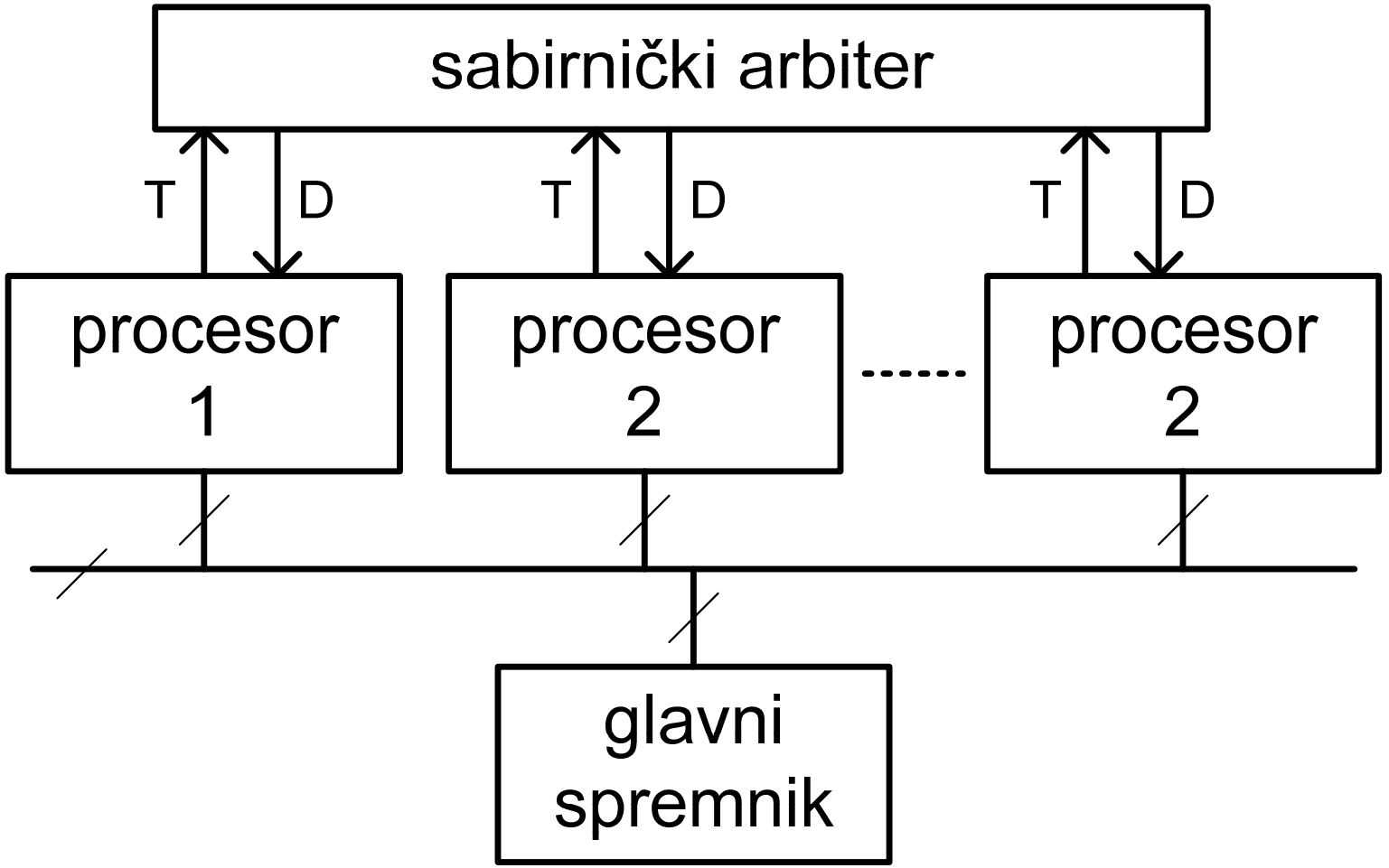
Višeprocesorski sustavi

- OS treba na jednaki način pružati svoje usluge bez obzira je li u sustavu samo jedan procesor ili ih ima više

- Postoje razne izvedbe višeprocesorskih sustava!
- U nastavku se razmatraju samo oni koji koriste (imaju na raspolaganju) zajednički spremnik u kojem se mogu pohraniti podaci OS-a:
  - podaci o korisničkim podacima (procesi)
  - podaci za upravljanje dretvama (opisnici)
  - podaci za upravljanje ostalima elementima sustava (upravljanje spremnikom, UI napravama, …)

- Što je to višeprocesorski sustav (ili višestruki procesor – procesor s više procesorskih jedinki; engl. *multi/many core*)?

SABIRNIČKI POVEZANI VIŠEPROCESORSKI SUSTAV



- Svaki procesor može pristupiti do *dijeljenog spremnika* (engl. *shared memory*) i to preko zajedničke sabirnice.

- U jednom sabirničkom ciklusu spremniku može pristupiti samo jedan od procesora.

- *Dodjeljivač sabirnice* (engl. *bus arbiter – sabirnički arbitar*) odlučuje kojoj će dretvi i kada biti dodijeljena sabirnica

- Ako su svi procesori jednaki - *homogeni* sustav
  - dretve mogu se izvoditi na bilo kojem procesoru.

Višezadaćnost

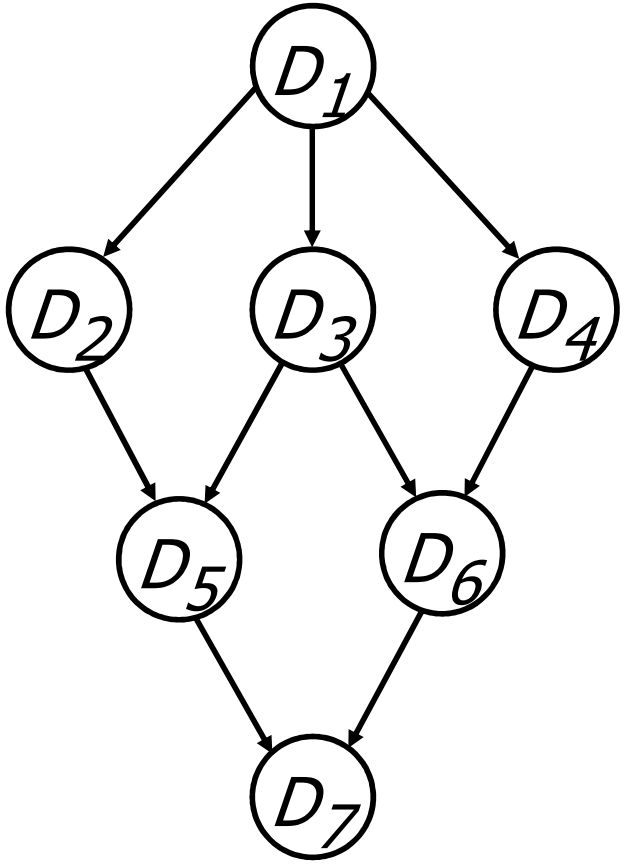
- ciljevi višezadaćnog (ili paralelnog) rada:
  - bolje iskoristiti dostupne procesne jedinke
  - smanjiti složenost programske potpore odjeljivanjem nezavisnih elemenata
- višezadaćnost je svojstvo OS-a; OS ga omogućuje
- tehnike programskog iskorištenja višezadaćnosti:
  - višedretvenost (dretve unutar istog procesa)
    - dretve se izvode unutar istog adresnog prostora
    - dijele sve podatke procesa (lakše i brže komuniciraju)
  - višeprocesni rad
    - izvode se nezavisno (adresni su im prostori odvojeni)
    - ne dijele podatke ili sredstva
    - za komunikaciju moraju koristiti usluge OS-a
  - kombinacija prethodna dva načina

Model višedretvenog programa

- skup međuovisnih dretvi
  - ovisnost može biti izravno zadana (desni graf) ili složenija

- dretve koriste *sinkronizacijske primitive* (funkcije) da bi uskladili svoje izvođenje (u definiranom redoslijedu)
  - koriste se pozivi jezgre

- OS mora osigurati mehanizme za:
  - stvaranje dretvi
  - zaustavljanje dretvi
  - sinkronizaciju
  - komunikaciju



Međusobno isključivanje

- Osnovni oblik sinkronizacije dretvi
- *primjeri*: korištenje zajedničke naprave ili drugog objekta
- dio koda dretve koji (kontrolirano) koristi zajedničko sredstvo: ***kritični odsječak***
  - kritični se odsječak mora izvoditi međusobno isključivo:
    - najviše jedna dretva može biti u kritičnom odsječku u bilo kojem trenutku
- ***nekritični odsječak***: dio koda u kojem se sredstva ne koriste
- algoritam međusobnog isključivanje pruža isključujuće korištenje sredstva (pojedinačno)

- Ostvarenje algoritama za međusobno isključivanje
  - korisnička dretva ne ostvaruje algoritam
    - postoje algoritmi, ali oni ne koriste efikasno sustav!
  - za međusobno isključivanje koriste se ***jezgrine funkcije***

- Pozivom *jezgrine funkcije sinkronizacije*
  - ili se dretvi dodjeljuje sredstvo i omogućuje nastavak rada

  - ili se dretva blokira – kontekst joj se pohranjuje, a druga se dretva propušta na procesor

Jezgra operacijskog sustava

- Skup podataka i jezgrinih funkcija
- Osnovna zadaća jezgre:
  - upravljanje dretvama (višedretveno izvođenje)
  - pružanje mehanizama za sinkronizaciju i komunikaciju među dretvama
  - upravljanje UI napravama i ostalim sredstvima sustava (procesorsko vrijeme, spremnik)