Upravljanje U/I uređajima

Vladimir Filipović

Upravljanje U/I uređajima

- Jedan od osnovnih zadataka OS je da upravlja U/I uređajima i da pri tome pruži što jednostavniji, ali i koliko je moguće univerzalniji interfejs između uređaja i sistema.
- Sa druge strane, treba od korisnika prikriti detalje upravljanja uređajima i obezbediti što pogodniji interfejs za njihovo efikasno korišćenje.
- Ispunjavanje ovih zahteva predstavlja izazov, uzimajući u obzir činjenicu da postoji čitav spektar različitih uređaja i da se njihov broj i raznolikost svakim danom uvećava.

Upravljanje U/I uređajima (2)

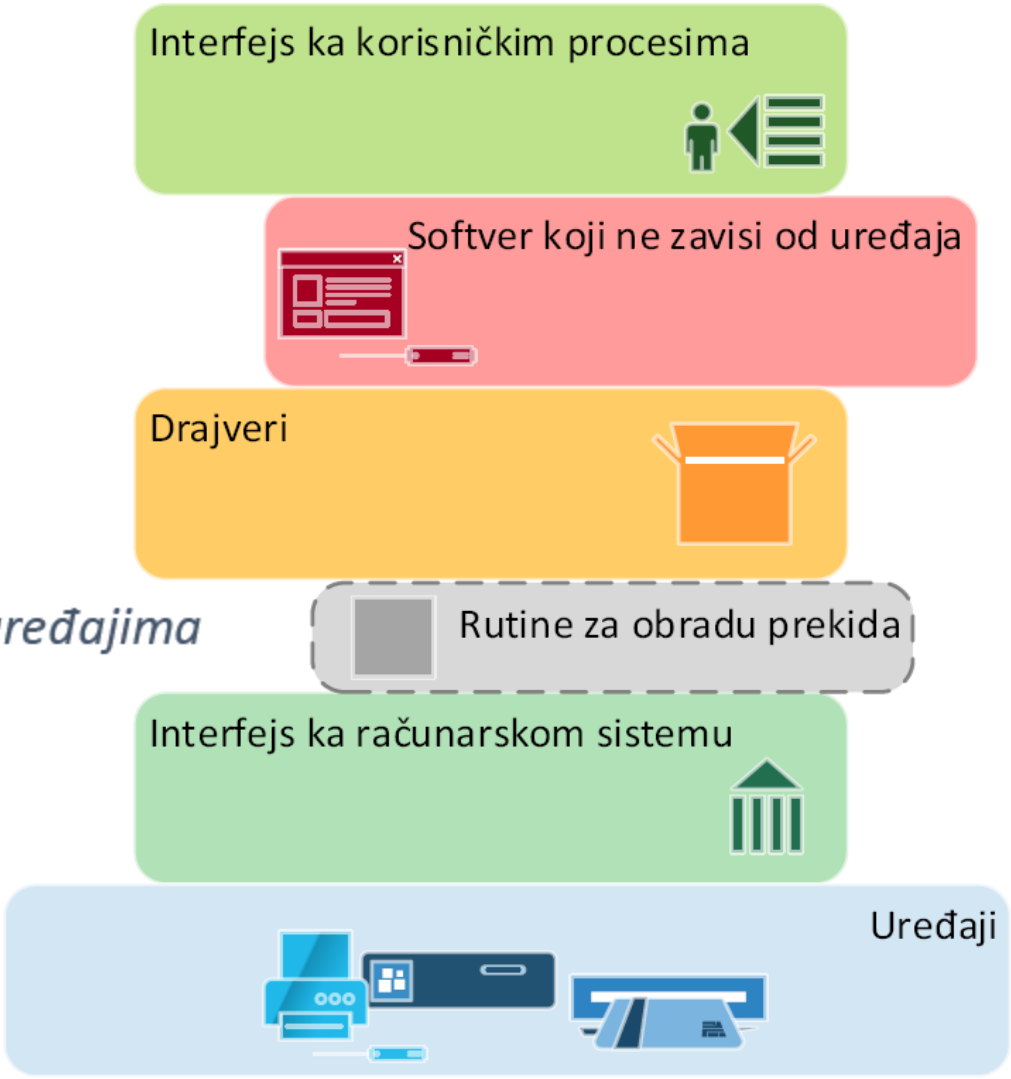
- Ulazno-izlazni uređaji predstavljaju usko grlo kada su performanse u pitanju.
- Sva komunikacija sa ulazno-izlaznim uređajima odvija se u posebnom režimu operativnog sistema za koji je odgovorno jezgro (kernel).
- Sistemski pozivi su jedini način kojim korisnik ima mogućnost da pristupi hardveru. Iz tog razloga, često se događa da se veliki deo softvera u jezgru operativnog sistema odnosi na upravljanje uređajima.

Upravljanje U/I uređajima (3)

- Upravljanje uređajima ima složenu hijerarhiju kako bi bilo što efikasnije i udobnije.
  - na najnižem nivou nalaze se uređaji, odnosno hardver;
  - zatim sledi interfejs kojim se on vezuje za računarski sistem;
  - u nekim slučajevima se koriste rutine za obradu prekida koje se nalaze na sledećem nivou;
  - onda slede drajveri koji omogućavaju korišćenje uređaja;
  - na sledećem (logičkom) nivou se nalazi softver koji ne zavisi od uređaja;
  - na kraju je interfejs ka korisničkim procesima uz pomoć kog se obezbeđuju usluge.

Upravljanje U/I uređajima (4)

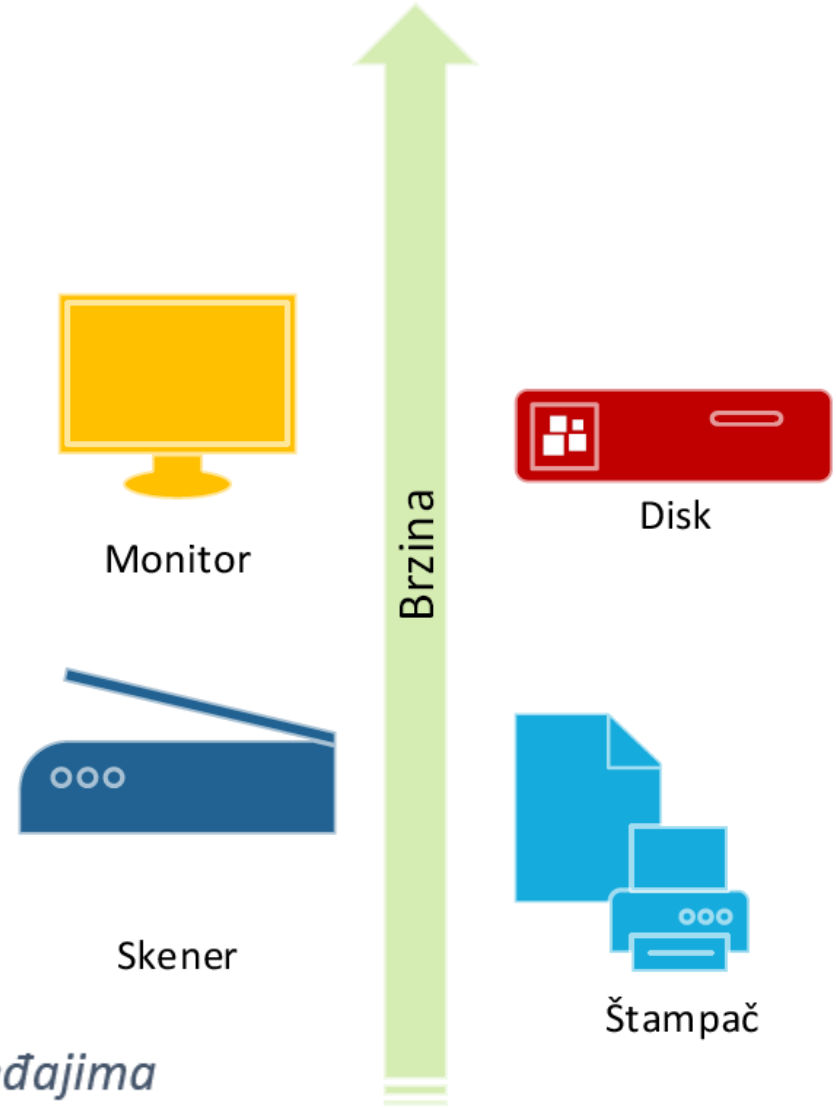
Nivoi upravljanja ulazno-izlaznim uređajima



Hardverske komponente

- Postoje razni tipovi uređaja koji se mogu koristiti u okviru računarskih sistema.
  - Diskovi i optički uređaji su jedan primer. Oni služe kao sekundarna memorija, odnosno omogućavaju skladištenje i trajno čuvanje podataka, a često se koriste i za smeštanje podataka koji su potrebni aktivnim procesima prilikom multiprogramiranja.
  - Tastature, miševi, monitori, itd. su uređaji koji korisnicima omogućavaju interakciju sa procesima.
  - Tu su i video kamere, mikrofoni, komunikacioni kablovi koji omogućavaju da se uređaji priključe na sistem.
- Prisutna je i razlika u brzinama pristupa i rada ovih uređaja.

Hardverske komponente (2)



Brzina pristupa ulazno-izlaznim uređajima

Hardverske komponente (3)

- Na osnovu načina prenosa podataka U/I uređaji se dele na:
  - Blok uređaje - obrađuju podatke u prethodno definisanom formatu, odnosno blokovima fiksne dužine. Dužina blokova može varirati od uređaja do uređaja, a blokovi se mogu i adresirati. Svi transferi se obavljaju prebacivanjem odgovarajućeg broja međusobno nezavisnih blokova. Svakom bloku se može direktno pristupiti.
    Primeri: diskovi, DVD-ROM, USB memorije.
  - Znakovne uređaje - manipulišu podacima u formi niza (toka) znakova. Uređaj ovog tipa nemaju mogućnost direktnog pristupanja memoriji i nisu adresibilni.
    Primeri: štampači, mrežni interfejsi, miševi, itd.

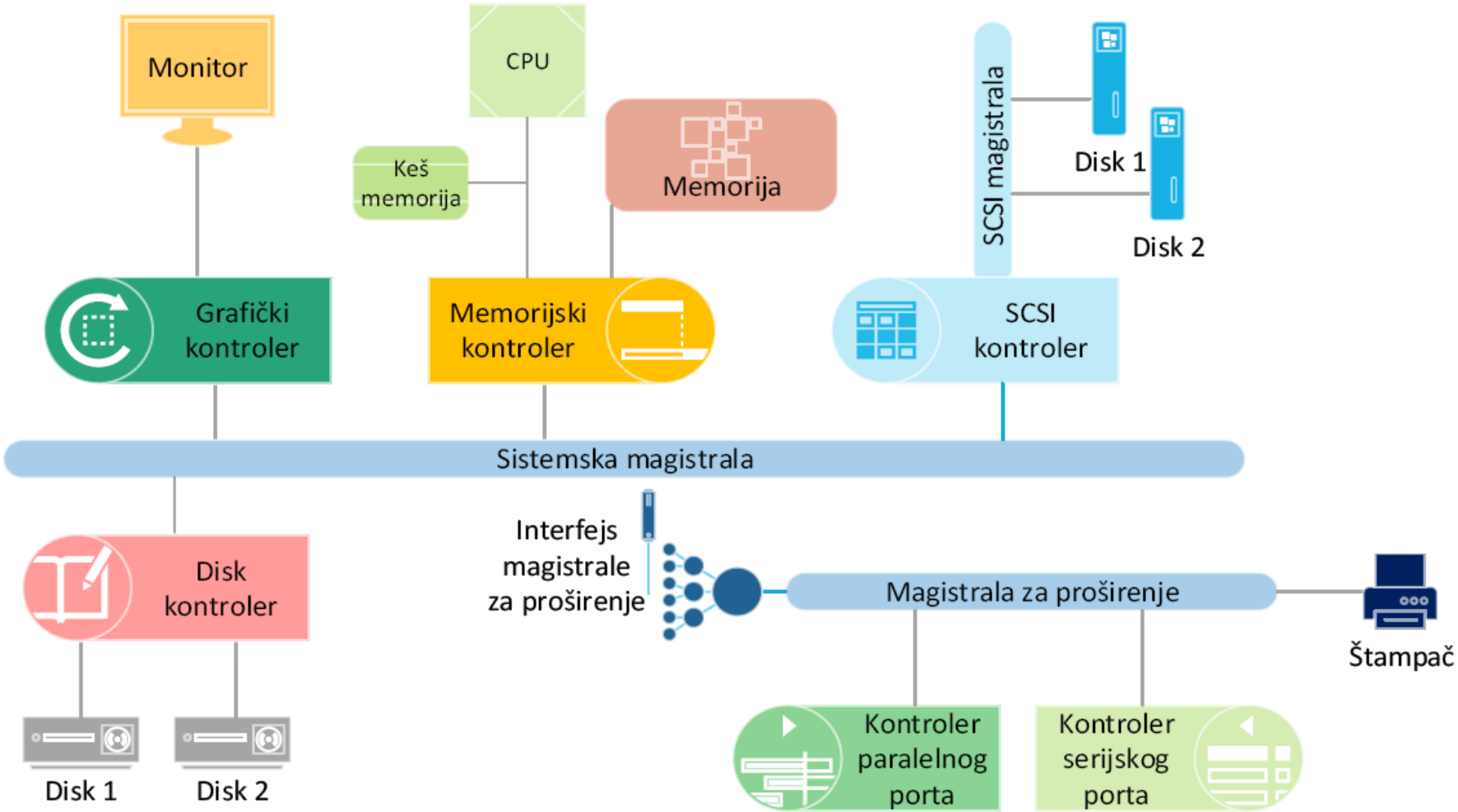
Interfejs uređaja

- Uređaji se na različite načine mogu povezati na računarski sistem.
- Način povezivanja često određuje brzina uređaja, tako da oni sporiji obično imaju jednostavnije vezivanje dok brži zahtevaju složeniji pristup.
- Uređaji se povezuju na sistem uz pomoć priključaka. Ova veza ne mora da bude isključivo izvedena uz pomoć kabla već može biti i bežična.

Interfejs uređaja (2)

- Komunikacija se dalje odvija preko magistrale koja povezuje priključke sa ostalim delovima sistema.
- Magistrala je skup linija prenosa koje povezuju uređaje i pri tome podržavaju jasno definisana pravila, tj. protokole za komunikaciju između njih.
- U okviru računarskog sistema obično postoji više magistrala, od kojih su najpoznatije:
  - sistemska magistrala, koja vezuje procesor sa brzim ulazno-izlaznim uređajima i
  - magistrala za proširenje koja sadrži serijske i paralelne priključke namenjene za sporije uređaje poput tastature, miša, štampača, itd.

Interfejs uređaja (3)



Sistemska magistrala i magistrala za proširenje

Interfejs uređaja (4)

- Kontroleri su deo hardvera koji mogu opsluživati portove, magistrale ili uređaje.
  - Na primer, kontroler serijskog porta je čip koji se nalazi u računarskom sistemu i ima zadatak da kontroliše signale na žicama serijskog porta.
  - Sa druge strane, SCSI (Small Computer System Interface) kontroler je adapter koji sadrži procesor i memoriju a funkcija mu je da formira novu magistralu na koju se mogu povezivati uređaji.
- Kontroleri obično imaju registre koji su najčešće memorijski čipovi i koriste se za prenos podataka i za komunikaciju sa ostatkom sistema.

Interfejs uređaja (5)

- Uobičajeno da kontroleri imaju sledeće registre:
  1. Registar primljenih podataka, u koji se smeštaju podaci preuzeti sa uređaja.
  2. Registar poslatih podataka, u koji se smeštaju podaci koje treba uskladištiti na uređaj.
  3. Statusni registar, u koji se smeštaju podaci u vidu kodova koji govore o trenutnom statusu uređaja. Ovi podaci su namenjeni procesoru, i može se signalizirati: da li su podaci učitani, da li je izvršena zadata komanda, da li je došlo do greške, itd.
  4. Kontrolni registar, u koji procesor upisuje komande koje treba izvršiti ili informacije vezane za način rada uređaja.

Interfejs uređaja (6)

- Procesor komunicira sa uređajem korišćenjem registara.
  - Jedan od načina za komunikaciju je uvođenje posebnih instrukcija za upis ili čitanje iz registara.
  - Takvim instrukcijama aktiviraju se posebne linije magistrale, da bi se pravila razlika između pisanja u memoriju i pisanja u registar kontrolera.
  - Podaci potom putuju magistralom i smeštaju se u registre, ili se prenos vrši u obrnutom smeru.
  - U ovom slučaju registri se mogu koristi za prenos podataka od procesora ka uređaju i obrnuto, za prenos instrukcija od procesora ka uređaju i za prenos informacija o statusu uređaja procesoru.

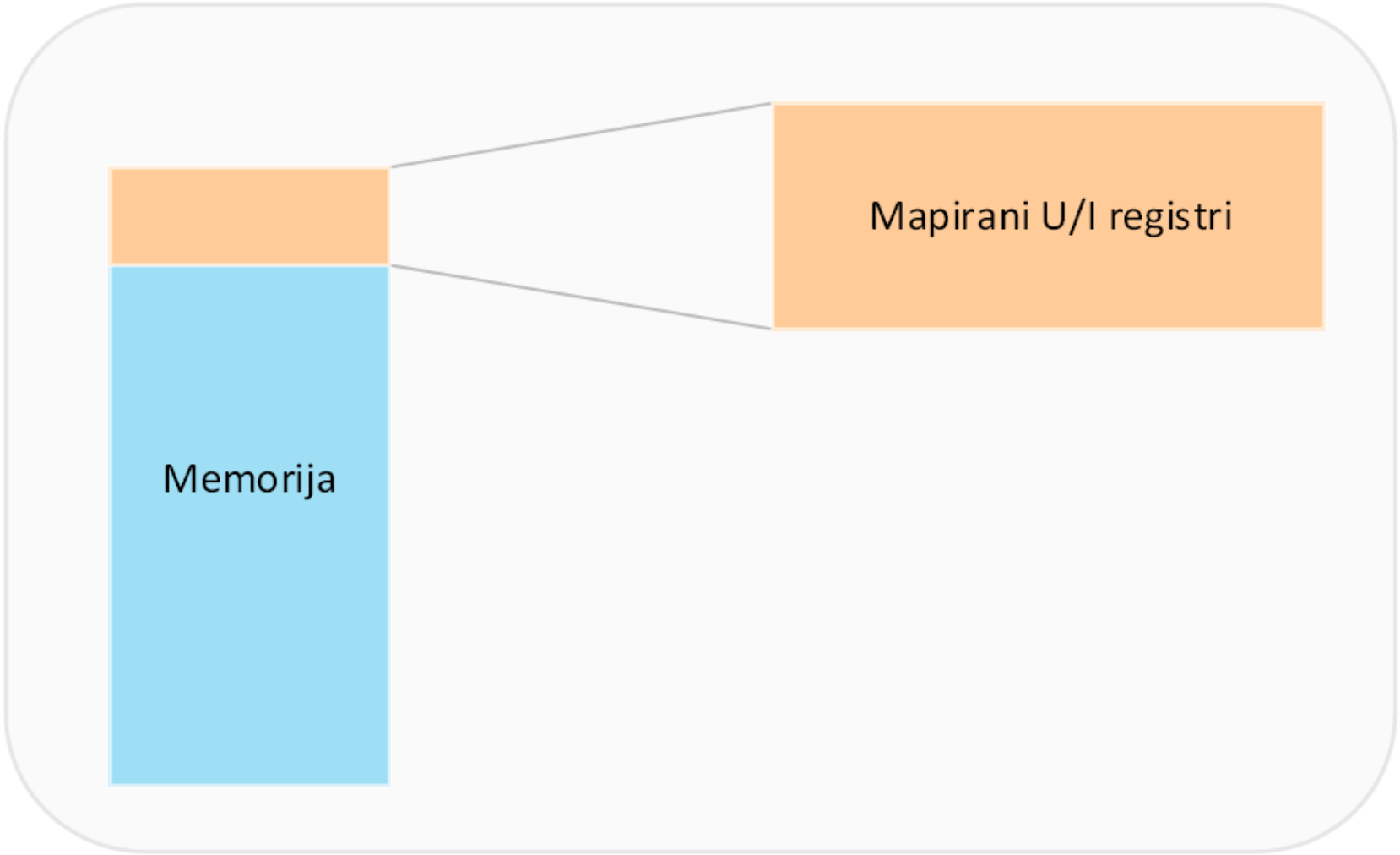
Interfejs uređaja (7)

- Drugi način za komunikaciju podrazumeva da kontroleri podržavaju memorijski mapirane U/I operacije.
  - U tom slučaju je registrima moguće pristupati direktno putem memorijske adrese, kao da su sastavni deo glavne memorije, tako da nema potrebe za uvođenjem posebnih instrukcija.
  - Ovakav pristup zahteva da procesor ima mogućnost da adresira registre kao što adresira ostale delove memorije.
  - Postoje sistemi koji podržavaju oba pristupa, gde je moguće adresirati registre ali i koristiti posebne instrukcije (npr. grafički procesor GPU).

Interfejs uređaja (8)

- Memorijsko mapiranje registara ima svojih prednosti, ali i mana:
  - Na ovaj način izbegava se upotreba posebnih instrukcija asemblera, inače obaveznih pri radu sa registrima. U slučaju memorijskog mapiranja OS vidi registre kao obične promenljive pa ih je moguće koristiti u programskim jezicima navođenjem njihovih adresa.
  - Međutim, ova mogućnost može predstavljati problem. Naime, jedna od čestih grešaka pri programiranju jeste upis podataka na slučajni, greškom izabrani deo memorije. Pošto se registru uređaja pristupa kao i svakom drugom delu memorije, njegov sadržaj se lako može greškom promeniti.

Interfejs uređaja (9)



Memorijsko mapiranje registra U/I uređaja

Tehnika prozivanja

- Tehnika prozivanja (polling) predstavlja najjednostavniji pristup za upravljanje U/I uređajima koji podrazumeva da procesor odradi sav posao.
- Ideja je da se statusni registar uređaja, proverava u regularnim vremenskim intervalima.
- Pri tome uređaj postavljanjem određene vrednosti u registar signalizira sistemu da je pročitao ili pripremio podatak.

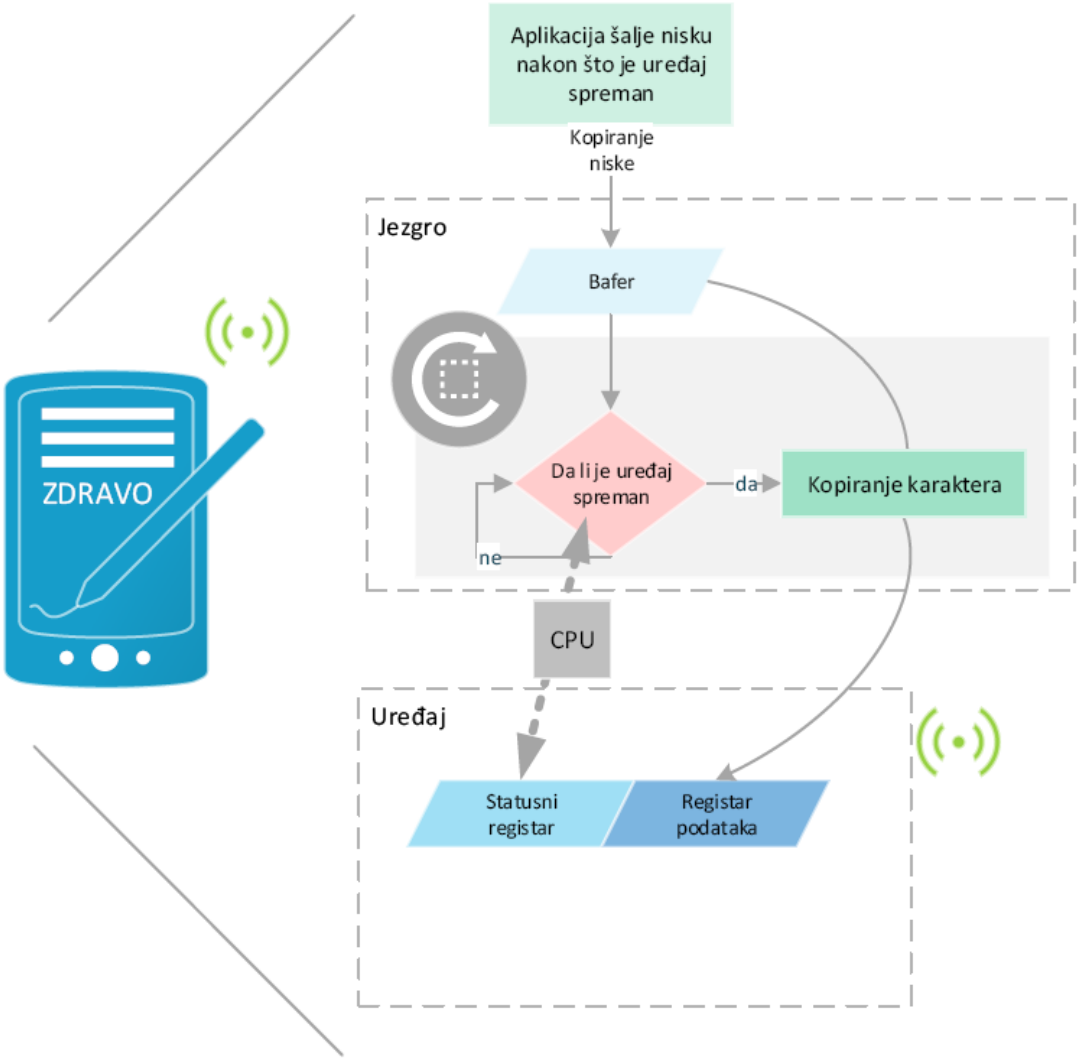
Tehnika prozivanja (2)

- Ova ideja se može prikazati na jednostavnom primeru:
  - Neka je dat uređaj za bežično slanje podataka (npr. Bluetooth modul) i aplikacija koja želi da pošalje niz znakova „ZDRAVO“.
    1. Aplikacija prvo koristi sistemski poziv kako bi jezgru operativnog sistema prosledila zahtev za uređajem.
    2. Ako uređaj nije spreman, poziv će vratiti grešku ili će se blokirati sve dok uređaj ne postane spreman.
    3. Kada se uređaj dodeli aplikaciji, upućuje se sistemski poziv sa zahtevom da mu se prosledi data niska znakova.
    4. Operativni sistem prvo kopira nisku u bafer jezgra, radi lakšeg daljeg pristupa, i proverava da li je uređaj spreman da primi podatke.

Tehnika prozivanja (2)

5. Ako uređaj nije spreman, čeka se.
6. Kada uređaj postane spreman, operativni sistem kopira prvi znak u registar podataka uređaja.
7. Čim se prvi znak iskopira, operativni sistem proverava da li je uređaj spreman za novi. Ova provera se vrši tako što se čita vrednost statusnog registra, u koji uređaj upisuje dogovorene statusne kodove.
8. Sve dok uređaj ne pošalje prvi znak (u ovom primeru se podrazumeva da se prenos vrši znak po znak), u statusnom registru će se nalaziti kod koji označava zauzeće.
9. Operativni sistem, odnosno procesor, sve vreme će proveravati sadržaj statusnog registra i čim se iz njega pročita da je uređaj spreman, poslaće mu se drugi znak i tako u petlji, sve dok se ne prosledi cela niska.

Tehnika prozivanja (3)



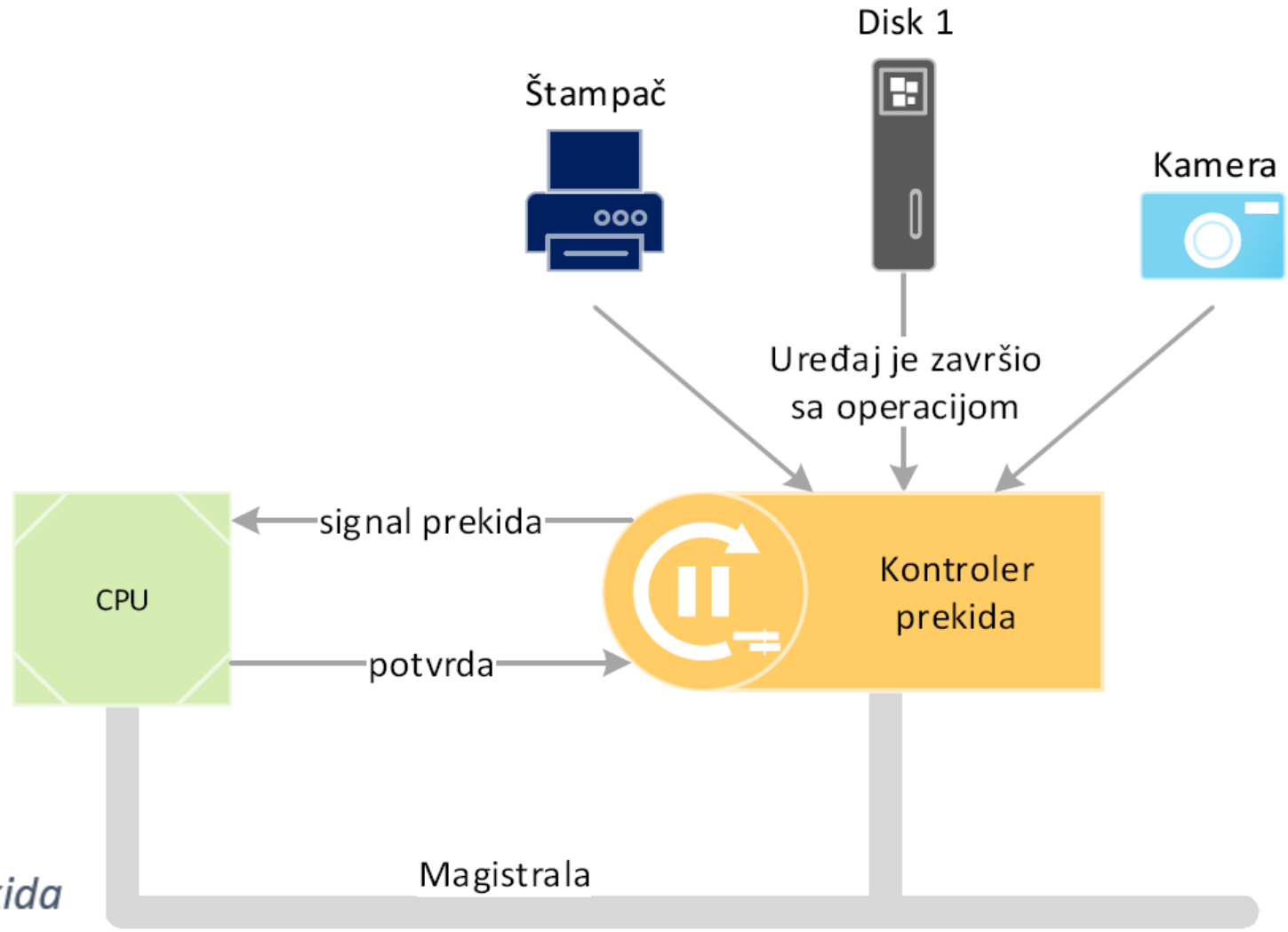
Tehnika prozivanja (4)

- Dakle, procesor neprekidno „proziva“ uređaj kako bi proverio da li je spreman za novi znak.
- Ovakav način izvršavanja se još naziva i aktivno čekanje.
- Ova tehnika je jednostavna, ali je neefikasna, jer se procesorsko vreme troši na učestalo proveravanje statusnog registra.
  - U prethodnom primeru, procesor bi prosledio znak, a onda bi većinu vremena provodio proveravajući statusni registar u iščekivanju da će iz njega pročitati kod koji označava da je moguće proslediti sledeći znak.

Prekidi

- Upravljanje U/I operacijama korišćenjem prekida se zasniva na hardverskom mehanizmu koji daje mogućnost uređajima da signaliziraju stanje, dajući tako vremena procesoru da obavlja drugi posao sve dok uređaj ne pošalje signal da je spreman.
- Za realizaciju ovakvog pristupa potrebno je da hardverski deo računarskog sistema sadrži posebnu liniju za prekide.

Prekidi (2)

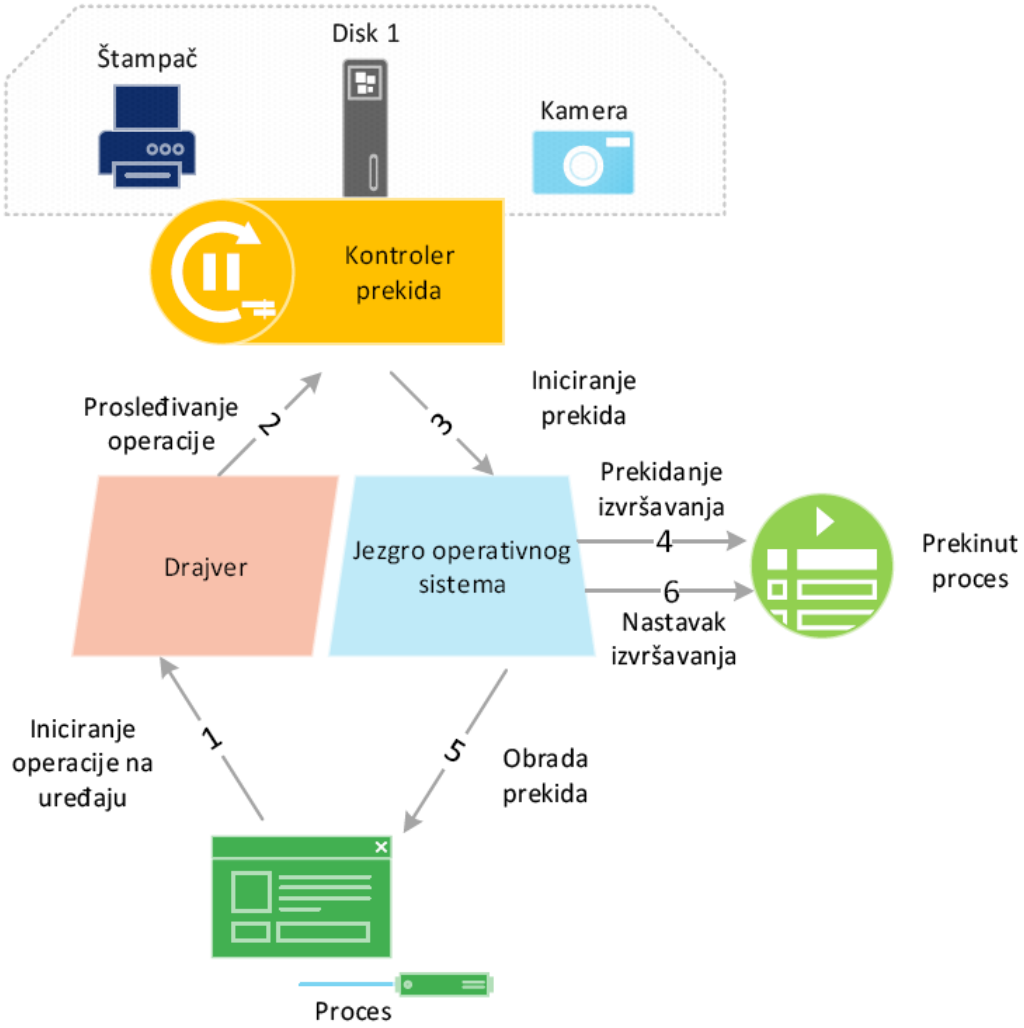


Kontroler prekida

Prekidi (3)

- Princip funkcionisanja sistema zasnovanog na prekidima:
  1. Proces inicira neku operaciju na uređaju obraćajući se odgovarajućem drajveru.
  2. Drajver preko kontrolera prosleđuje operaciju uređaju.
  3. Kada uređaj završi sa operacijom njegov kontroler procesoru šalje signal preko linije za slanje prekida.
  4. Procesor prekida izvršavanje procesa koji se trenutno obrađuje a dispečer prebacuje njegov kontekst, kako bi se obradio prekid.
  5. Prekid se obrađuje uz pomoć rutine za obradu prekida.
  6. Nakon obrade prekida prekinuti proces može da nastavi izvršavanje ukoliko mu to planer dozvoli.

Prekidi (4)



Princip funkcionisanja sistema zasnovanog na prekidima

Prekidi (5)

- Kada je prethodni primer (sa slanjem niske „ZDRAVO“) u pitanju, to bi značilo sledeće:
  1. Procesor šalje znak uređaju i prelazi na izvršavanje nekog drugog zadatka.
  2. Kada uređaj prosledi znak i bude spreman da primi novi, on izaziva prekid.
  3. Tada procesor prekida posao koji trenutno radi, prosleđuje novi znak uređaju i nastavlja prekinuti posao i tako u petlji sve dok se ne pošalju svi znaci niske.

Prekidi (6)

- U praksi, opisanom mehanizmu se obično dodaje još nekoliko funkcija:
  - Potrebno je da OS bude u mogućnosti da odloži prekid ako se trenutno izvršava neki veoma bitan proces.
  - Potrebno je efikasno detektovati koji uređaj je prosledio prekid, umesto da se ispituju svi.
  - Potrebno je prekidima dodeliti prioritete, kako bi sistem mogao da odreaguje sa različitim vremenima odziva u zavisnosti od hitnosti obrade signaliziranog prekida.

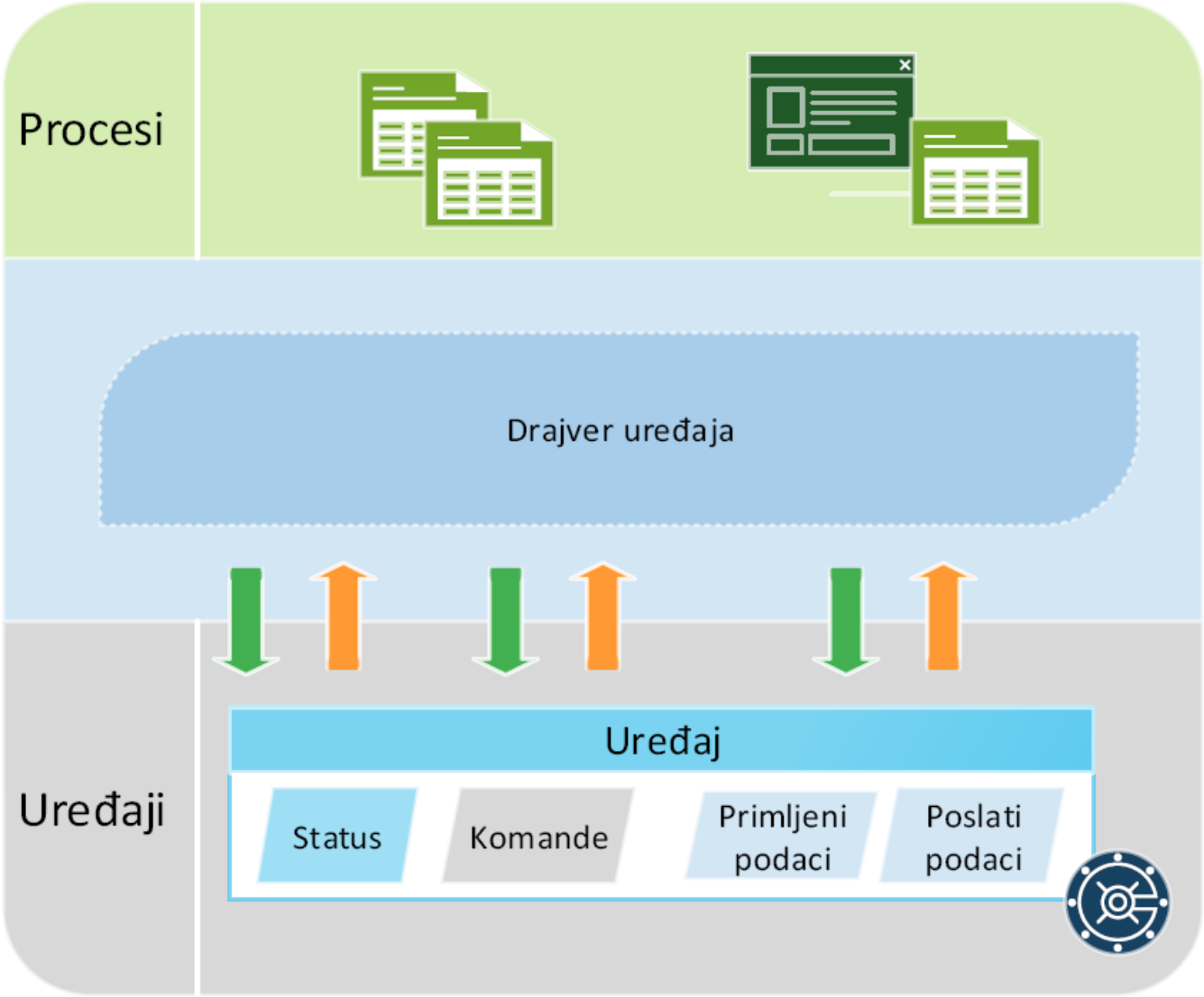
Prekidi (7)

- Sve ove funkcije realizuje hardverski uređaj koji se naziva prioritetni prekidni kontroler PIC (Priority interrupt controller).
- Većina procesora umesto jedne ima dve posebne linije za signale prekida:
  - Nemaskirajuća (non-masking) linija i
  - Maskirajuća (masking) linija.

Prekidi (8)

- Signal prekida koji je poslat po nemaskirajućoj liniji uvek može da prekine izvršenje tekućeg procesa.
  Nemaskirajuća linija se koristi za slanje signala prekida kao što su kritične hardverske greške, npr. greške u memoriji.
- Maskirajuća linija se može isključiti (zamaskirati) tokom izvršavanja važnih delova procesa koji ne smeju biti prekinuti.
  Ona se koristi za slanje signala prekida koji potiču od uobičajenih operacija, kao što je zahtev za nekom ulazno-izlaznom operacijom.

Prekidi (9)


Procesi
Drajver uređaja
Uređaji
Uređaj
Status
Komande
Primljeni podaci
Poslati podaci

Prekidi (10)

- Mehanizam prekida se oslanja na sledeće elemente:
  - Relativno mali broj uređaja može da pošalje signal prekida, i za svaki od njih postoji posebna rutina za obradu, koja se nalazi u memoriji.
  - Mehanizam prekida prihvata adresu, broj koji predstavlja poziciju u tabeli prekida (interrupt vector).
  - Tabela prekida koja čuva adrese svih rutina za obradu prekida u memoriji.
  - Glavna funkcija tabele je da se, po nastanku prekida, ne moraju pretraživati svi mogući izvori kako bi se utvrdilo koji od njih zahteva operaciju, već se iz tabele odmah pročita adresa rutine koju treba izvršiti.

Prekidi (11)

- U praksi, sistemi obično imaju više uređaja koji mogu izazvati prekid nego što postoji mesta u tabeli prekida tako da se taj problem rešava ulančavanjem.
  Svi izvori se dele u skupove, i za svaki skup se pravi lista sa adresama odgovarajućih rutina za obradu prekida.
  Kada se prekid izazove, rutine iz liste se pretražuju jedna po jedna dok se ne nađe odgovarajuća.
- Sistem prekida obično primenjuje prekidne prioritetne nivoe, a to je mehanizam koji omogućava da se odloži opsluživanje prekida niskih prioriteta, dok se opslužuju prekidi visokih prioriteta (bez maskiranja).
- Kada se desi prekid višeg prioriteta, prekida se opsluživanje prekida nižeg prioriteta, i opslužuje se prekid višeg prioriteta (pretpražnjenje - preemption).

Prekidi (12)

- Tokom podizanja sistema, ispitivanjem magistrale se određuje da li su uređaji prisutni i ako jesu postavlja se odgovarajuća prekidna rutina u memoriju i njena adresa se upisuje u tabelu prekida.
- Za vreme U/I ciklusa, uređaj postavlja signal prekida kad god zahteva servisiranje, tj. kada je komanda gotova ili kad se javila greška.
- Mehanizam prekida se koristi i za rukovanje izuzecima, deljenje nulom ili pristup zaštićenim adresama.

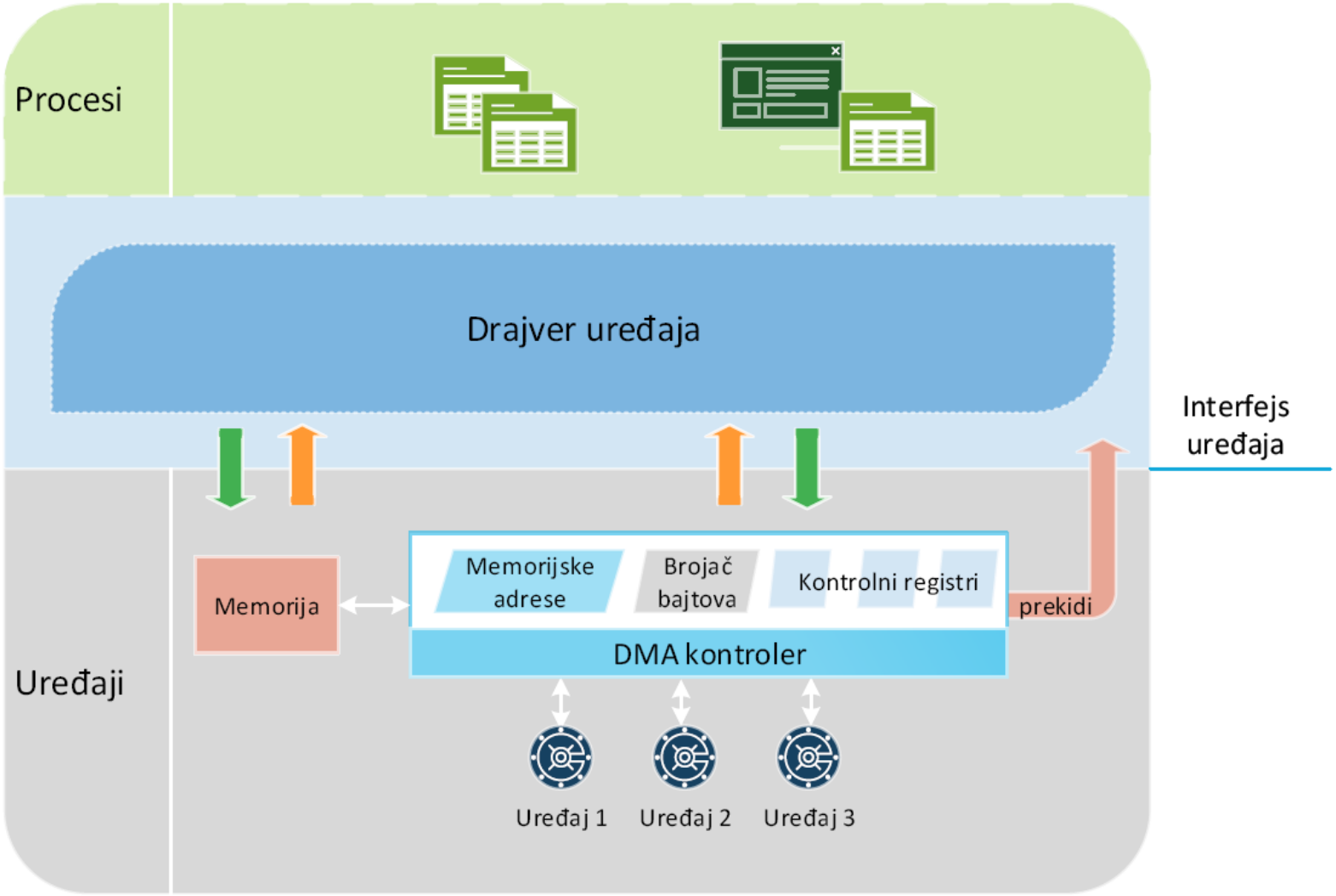
Direktan pristup memoriji

- Prekidi su veoma efikasan pristup ako se ne dešavaju previše često.
  - U prethodnom primeru, sistem sa prekidima bi izvršavao prekid za svaki znak, što je prilično neefikasno, jer se za svaki prekid vrši prebacivanje konteksta na šta se troši određeno vreme.
- Direktan pristup memoriji (DMA) je jedan od za rešavanje ovakvih problema.
  - To je hibridni pristup tehnike prozivanja i prekida.
  - Za njegovu implementaciju potreban je hardverski uređaj koji se naziva DMA kontroler.

Direktan pristup memoriji (2)

- DMA kontroleri imaju pristup sistemskoj magistrali nezavisno od procesora.
  - Sadrže nekoliko registara: registar memorijskih adresa, registar brojača bajtova i kontrolne registre.
  - U kontrolne registre smeštaju se informacije o U/I uređaju koji se koristi, smeru transfera podataka, jedinici prenosa, broju jedinica koje treba preneti, itd.
  - Suština DMA kontrolera da se on koristi kako bi prosleđivao uređaju karakter po karakter, bez opterećivanja glavnog procesora.
  - DMA u stvari vrši aktivno čekanje (polling) na uređaju, dok glavni procesor ostaje slobodan za druge poslove.
  - Prekidi ne dešavaju kod svakog znaka već DMA pristupa memoriji samo kada se bafer isprazni.

Direktan pristup memoriji (3)



DMA kontroler

Direktan pristup memoriji (4)

- Treba primetiti da u trenucima kada DMA pristupa memoriji, glavni procesor je blokiran da uradi isto, što usporava sistem.
- Međutim, prebacivanje posla na DMA kontroler generalno poboljšava ukupne performanse sistema.

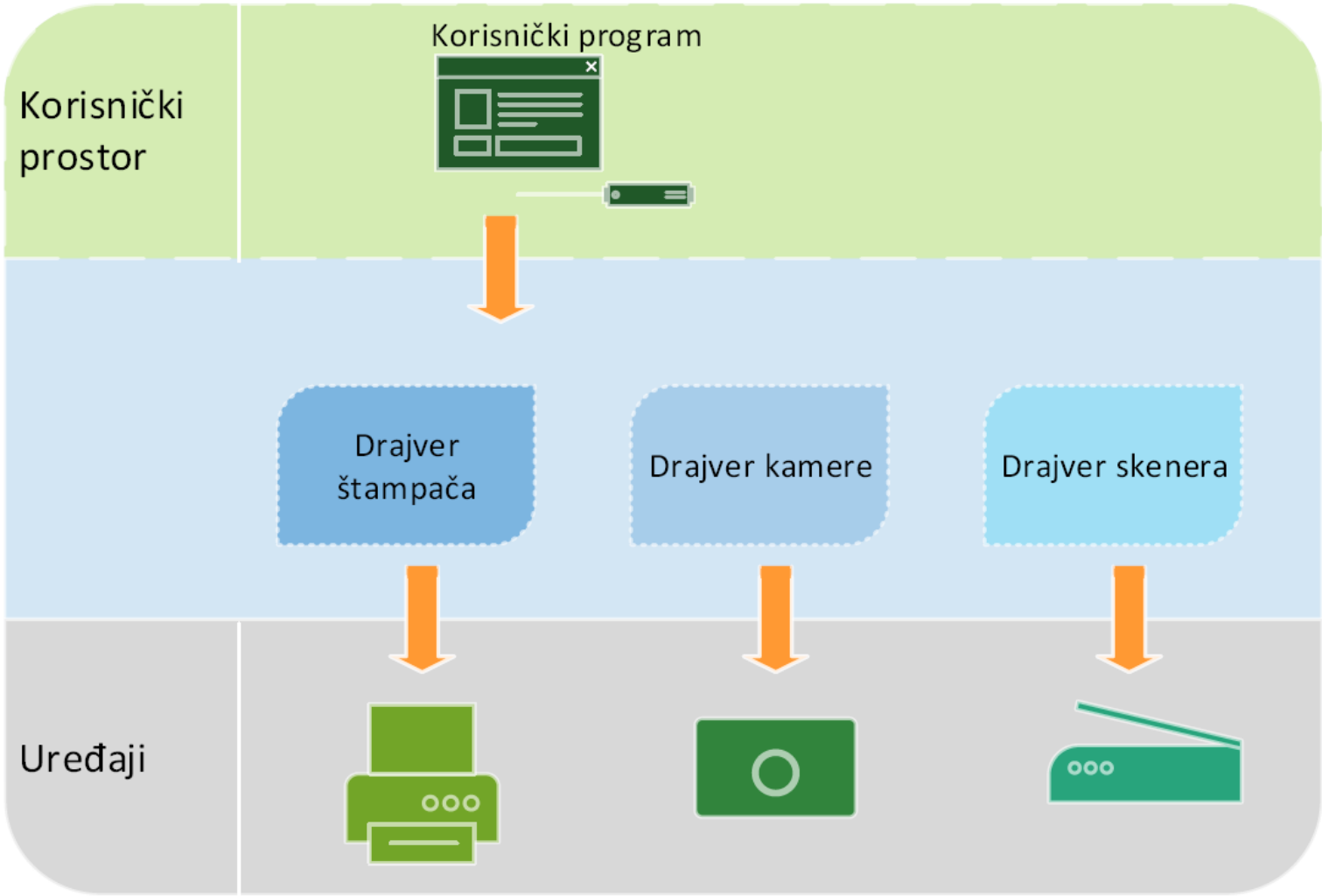
Drajveri

- Veoma je važno da interfejs U/I podsistema prema aplikacijama, odnosno korisnicima, bude što uniformniji.
  - Na primer, ako aplikacija ima potrebu da odštampa podatke, ona ne mora da ima informaciju o tome kom tipu štampača se obraća, niti kako on tačno radi.
- Uređaji se grupišu u klase i za svaku klasu definiše se skup funkcija kojima se upravlja konkretnim uređajem.
- Taj skup čini interfejs čija je glavna uloga da razlike i detalje o pojedinačnim uređajima sakrije od aplikacije.

Drajveri (2)

- Svaka aplikacija koja želi mogućnost da koristi štampač koristi isti standardizovani interfejs koji se ne menja, bez obzira na model, generaciju štampača, itd.
- Detalji upravljanja uređajima su enkapsulirani u posebne module koji se nazivaju drajveri i koji se pišu za svaki uređaj posebno.
- Drajveri sakrivaju razlike između uređaja koji pripadaju istoj klasi i da pružaju što standardnije interfejse.

Drajveri (3)



Drajveri

Drajveri (4)

- Drajveri se mogu pisati tako da funkcionišu u privilegovanom režimu rada, gde direktno pristupaju potrebnom hardveru, ali i u korisničkom režimu.
  - Prvi pristup je dobar zbog direktnog pristupa i veće slobode koja se pruža drajveru, ali nosi i veću odgovornost, jer loše napisan drajver može prouzrokovati pad sistema.
  - Drajver koji radi u korisničkom režimu mora koristiti sistemske pozive koji su mu pruženi, što degradira performanse (nekad je i nemoguće napisati drajver kao korisničku aplikaciju), ali zato ne postoji mogućnost pada sistema usled greške u drajveru.

Drajveri (5)

- U savremenim računarskim sistemima postoji velika šarenolikost kada su uređaji u pitanju.
  - Uređaji se mogu razlikovati po smeru prenosa
    - isključivo ulazni (miš, tastatura)
    - isključivo izlazni (monitor, štampač) i
    - ulazno-izlazni (disk, mrežna karta).
  - Ako se posmatra i veličina jedinice sa kojom se barata
    - blok uređaji i
    - znakovni uređaji

Drajveri (6)

- Ako se posmatra način pristupa:
  - uređaji sa sekvencijalnim pristupom (modem) i
  - uređaji sa direktnim pristupom (fleš memorija).
- Ako se posmatra da li mogu da budu deljeni između više procesa:
  - deljivi uređaji - može ih koristiti veći broj procesa u istom trenutku
  - nedeljivi uređaji – njih može koristiti tačno jedan proces u jednom trenutku

Drajveri (7)

- Jedna od najvažnijih podela U/I uređaja se može izvesti na osnovu načina upotrebe:
  - uređaji za čuvanje podataka (disk, fleš memorija);
  - uređaji za prenos podataka (mrežna karta, modem);
  - uređaji za interakciju sa korisnicima (miš, tastatura, monitor).
- Uređaji se na osnovu ovakvih zajedničkih karakteristika grupišu u klase koje imaju isti interfejs prema aplikacijama.
- Bilo bi poželjno da i interfejsi za različite uređaje budu što sličniji.

Upravljački softver nezavistan od uređaja

- Kada drajveri premoste različite osobine uređaja koji čine računarski sistem i stvore udoban interfejs, na scenu stupa softver koji treba da omogući što efikasnije korišćenje tih uređaja.
- Pri tome, ovaj sloj softvera se projektuje tako da ne zavisi od uređaja koji se nalaze u sistemu i da prema svima ima što uniformniji pristup.
- Na ovom nivou se teži da se što manje razlikuje pristup kada su različite klase uređaja u pitanju.

Planiranje U/I operacija

- Vremensko planiranje ulazno-izlaznih operacija podrazumeva određivanje redosleda kojim će se ove operacije izvršavati tako da sistem funkcioniše što efikasnije.
- Izvršavati operacije onim redosledom kojim ih aplikacije zatraže obično nije efikasno za sistem na globalnom nivou.
- Dobro raspoređivanje operacija može u velikoj meri doprineti boljim performansama sistema.
- Najčešće se za svaki uređaj implementira red čekanja.

Planiranje U/I operacija (2)

- Kada aplikacija pošalje zahtev, on se smešta u red.
- Operativni sistem organizuje red menjajući redosled zahtevima tako da se uređaji efikasno koriste i da vreme odziva (prosečno vreme koje prođe od slanja zahteva do dobijanja usluge) bude što manje.
- Operativni sistemi bi trebalo da se brinu i o pravednoj dodeli resursa, odnosno da nijedna aplikacija ne bi trebalo da okupira uređaj dok ostale čekaju, osim ako tako nešto nije definisano prioritetom.

Baferovanje

- Pod baferom se podrazumeva deo memorije u koji se privremeno smeštaju podaci koje treba preneti od uređaja ka aplikaciji (ili drugom uređaju) ili u obrnutom smeru.
- Baferovanje je tehnika koja podrazumeva korišćenje bafera i može se koristiti iz više razloga.
- Jedna od situacija u kojima se efikasno primenjuje baferovanje je kada uređaji koji „sarađuju“ imaju različite brzine.

Baferovanje (2)

- Na primer, neka su data dva uređaja, Bluetooth modul za primanje podataka bežičnim putem i disk računara, koji je daleko brži. Neka je zadatak da se podaci koji se primaju uz pomoć uređaja upišu na disk.

  Bez baferovanja, za svaki znak koji se dobije događao bi se prekid kojim bi modul javljao da znak treba upisati - što ne bi bilo efikasno.

  Korišćenjem bafera omogućava se privremeno upisivanje podataka u memoriju sve do određene granice, tj. do popunjavanja bafera.

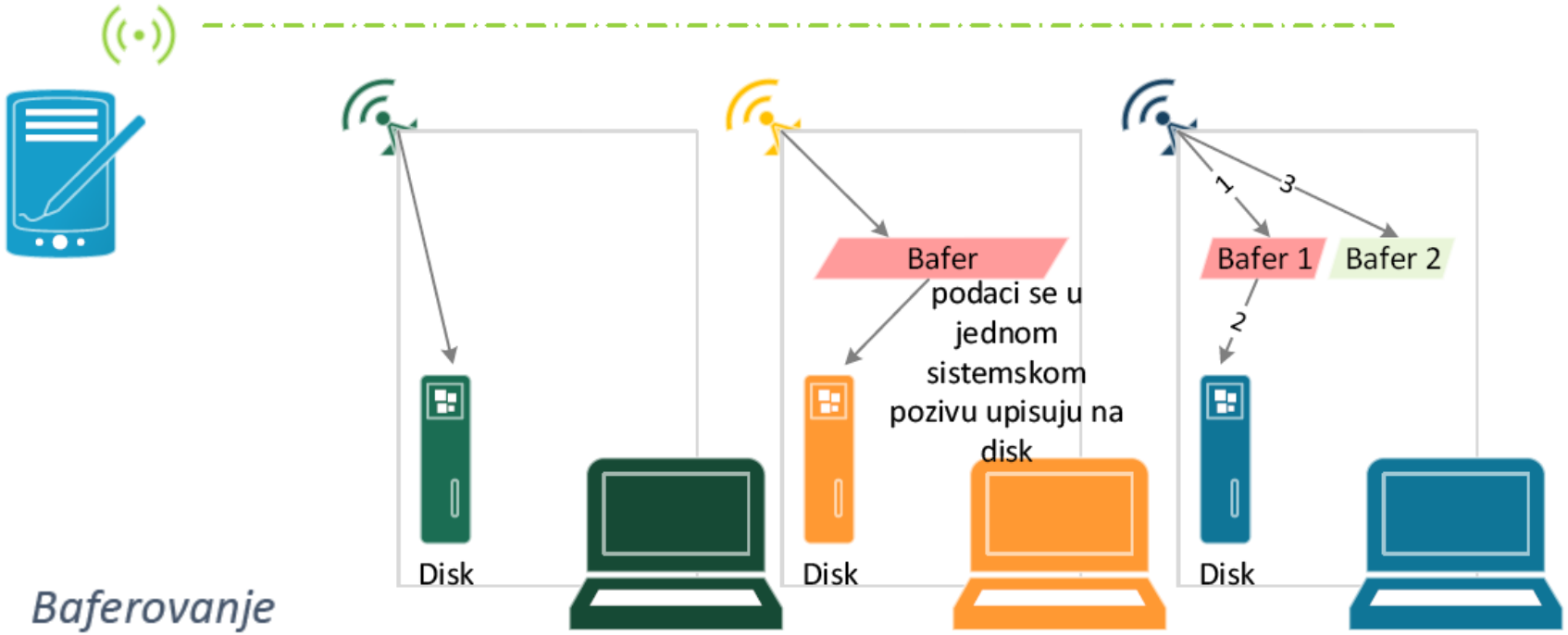
  Jednom kada se bafer napuni, izvršava se prekid, podaci se jednim sistemskim pozivom upisuju na disk i bafer se puni dalje.

Baferovanje (3)

- Unapređenje bi bilo korišćenje dva bafera, takozvano dvostruko baferovanje.
- Za upis podataka na disk potrebno je neko vreme, a nove podatke koji pristižu treba za to vreme negde smeštati, i za to se koristi drugi bafer.
- Kada se on napuni, podaci se smeštaju ponovo u prvi i tako naizmenično.
  - Dvostruko baferovanje je jedna od tehnika koja se koristi i pri keširanju diskova, jedan bafer na relaciji memorija-disk, za podatke koji se upisuju, a drugi na relaciji disk-memorija, za čitanje, čime se premošćava razlika u brzini između memorije i diska.

Baferovanje (4)

- Ideja dvostukog baferovanja se generalizuje kružnim baferovanjem, gde se određeni broj bafera se redom pune i po istom redosledu prazne.



Baferovanje (5)

- Baferi su vrlo korisni u mrežnim okruženjima, gde se velike poruke pre slanja dele na manje pakete koji se pojedinačno šalju preko mreže.
  - Paketi dolaze i primaju se u bafer po redosledu koji nije nužno isti kao pri slanju. Neki paketi mogu doći drugim mrežnim putem, a neki se mogu izgubiti i poslati ponovo, što izaziva kašnjenje a time i poremećen redosled kod primaoca.
  - Paketi su, međutim, obeleženi rednim brojevima, tako da se, nakon što se svi privremeno prime u bafer, od njih može rekonstruisati početna poruka.

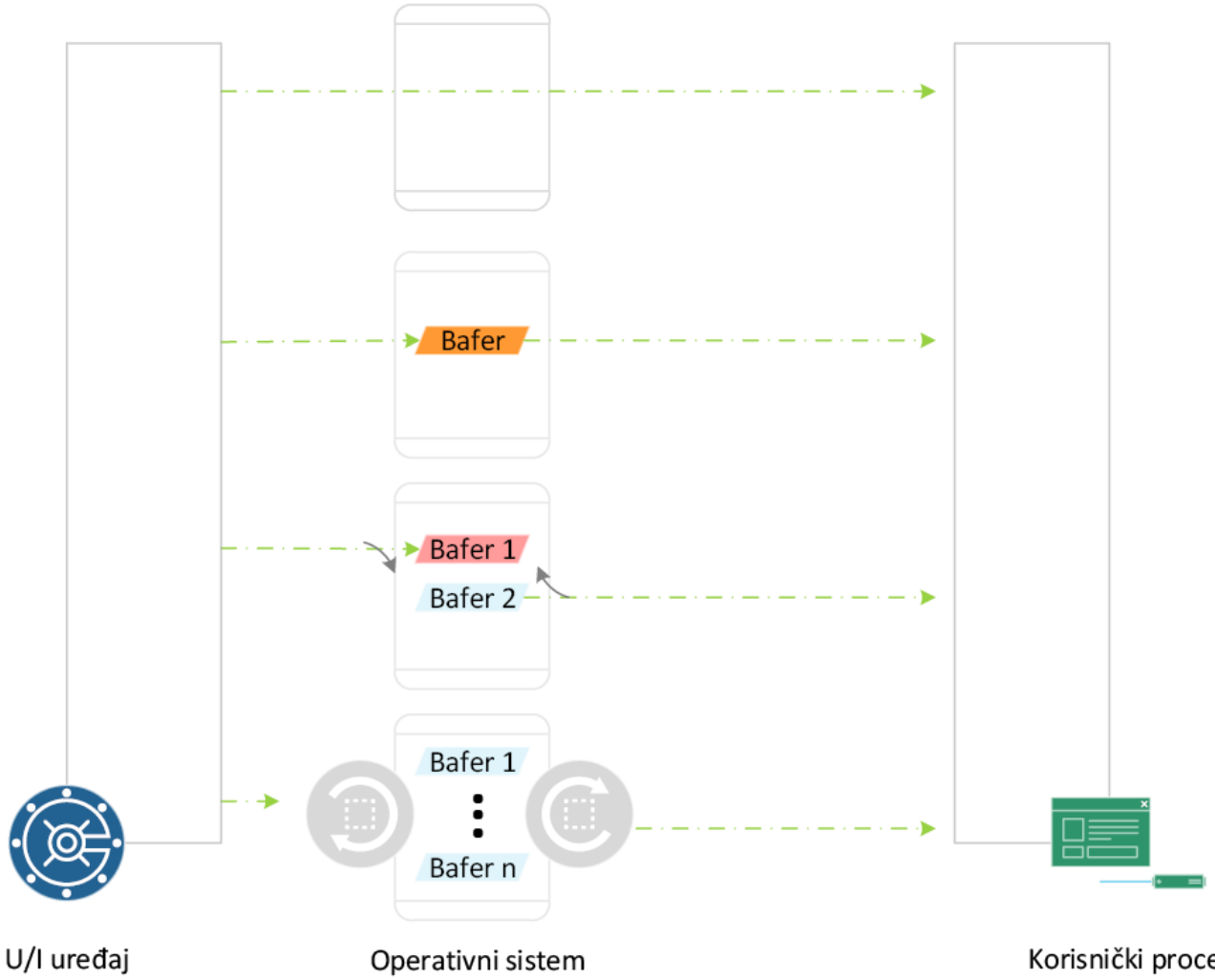
Baferovanje (6)

- Održavanje semantike kopiranja je još jedno polje gde je vrlo pogodno koristiti bafer.
  - Ako neka aplikacija u memoriji čuva određene podatke koje želi da upiše na disk, ona izdaje sistemski poziv za upis podataka.
  - Međutim, do nekonzistentnosti podataka može doći ako pre nego što se svi podaci uspešno upišu na disk, aplikacija izvrši promene u memoriji, odnosno promeni izvor sa kog se podaci čitaju.
  - Tako nešto je moguće kod asinhronih sistemskih poziva, gde bi aplikacija zadala komandu da se podaci upišu, i dobila nazad kontrolu pre kompletnog upisa.

Baferovanje (7)

- U takvom slučaju, moglo bi se očekivati da podaci na disku ne bi bili identični onim u memoriji u trenutku poziva funkcije za upis na disk, i kaže se da je semantika kopiranja narušena.
- Da bi se to izbeglo, operativni sistem u svom jezgru definiše bafer u koji, pri sistemskom pozivu za upis, kopira sve podatke koje mu aplikacija prosleđuje, pa tek onda predaje kontrolu nazad.
- Time poziv i dalje ostaje asinhron, jer se ne čeka da se podaci upišu na disk, ali su isti sigurni od promene u baferu jezgra.

Baferovanje (8)



Bafer, dvostruki bafer i kružni bafer

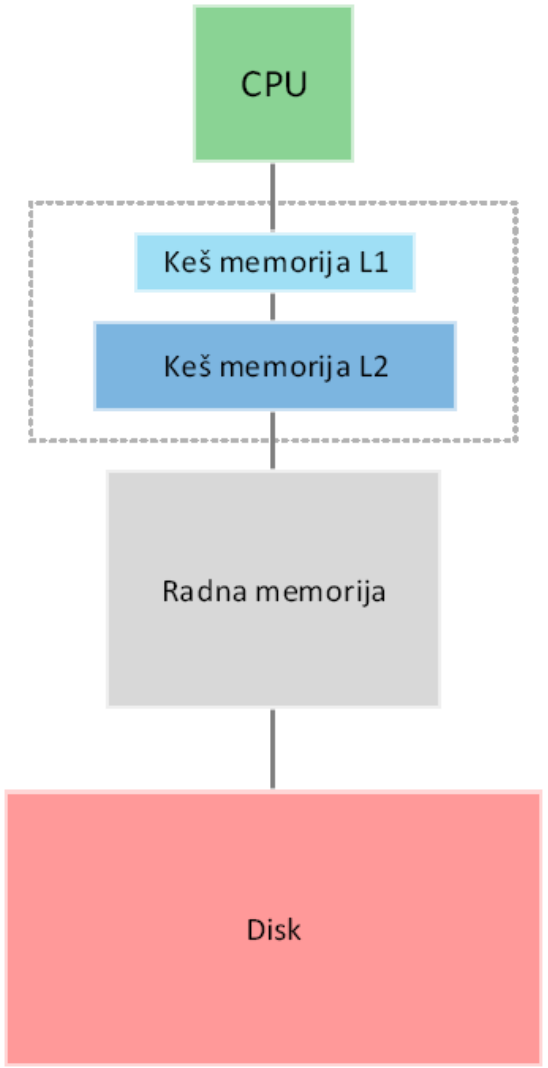
Keširanje

- Keš memorija je brza memorija koja čuva kopiju podataka. Ona je obično malog kapaciteta, ali veće brzine od radne memorije, što može biti memorija na U/I uređaju ili neka druga.
- Keširanje je tehnika pri kojoj se deo podataka kopira u keš memoriju, i kada se zatraže podaci, prvo se proverava da li se oni nalaze u kešu.
  - Ako se nalaze, štedi se vreme koje bi se trošilo na pristup glavnoj, sporijoj memoriji, i podaci se čitaju iz keš memorije.
  - Ako se traženi podaci ne nalaze u keš memoriji, onda se oni ipak moraju pročitati iz glavne memorije.

Keširanje (2)

- Na primer, moderni procesori sadrže nekoliko nivoa keš memorije - svaka je brža od prethodne, ali i manjeg kapaciteta.
- Prvo se podatak traži u najmanjoj, najbržoj memoriji. Ako nije tamo, prelazi se na sledeći nivo i tako redom.
- Keširanje je bitna tehnika koja značajno doprinosi performansama sistema.

Keširanje (3)



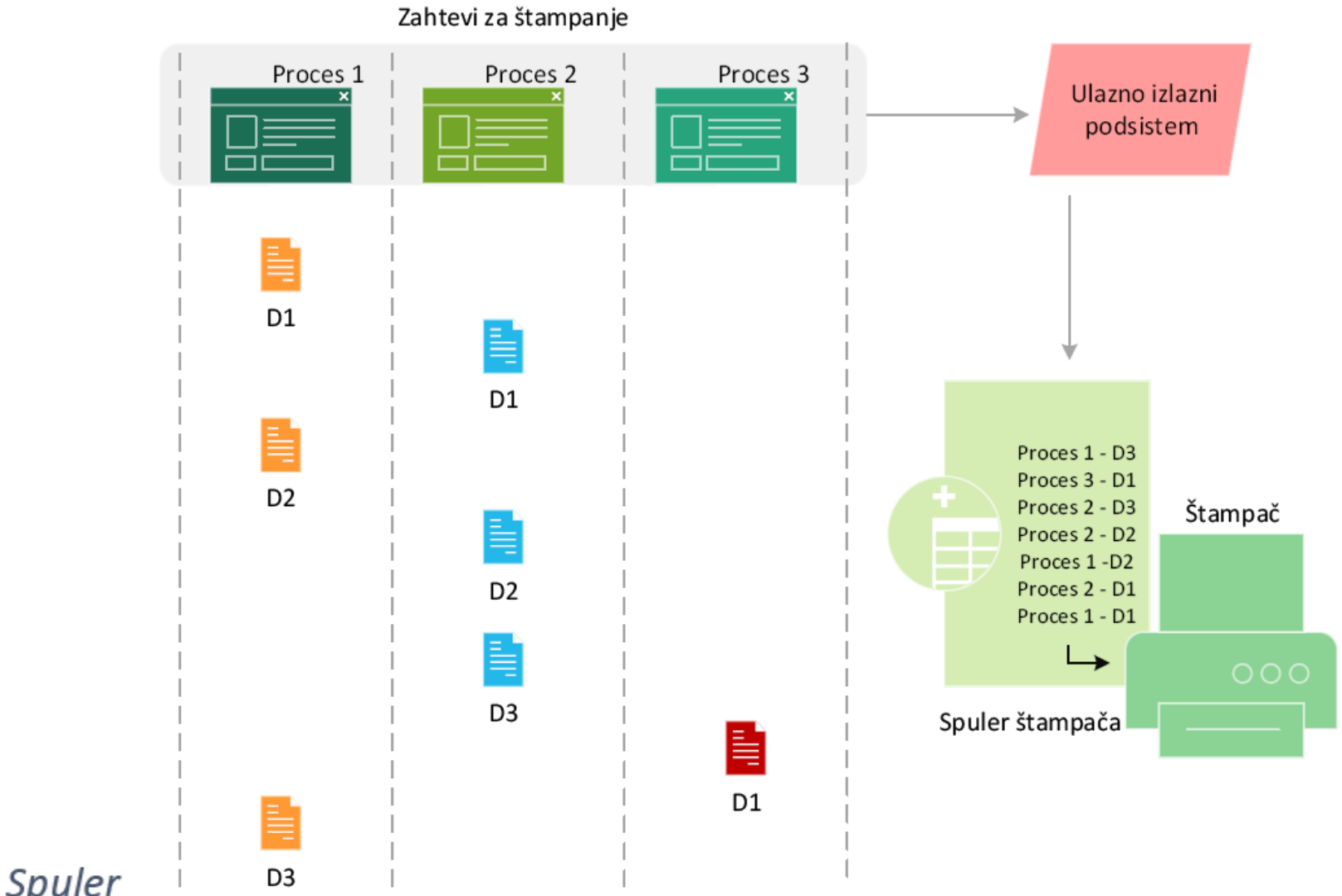
Keširanje (4)

- Važno je napomenuti da su keširanje i baferovanje dve različite tehnike, ali se u pojedinim slučajevima ista memorija može koristiti u obe svrhe.
  - Ako se podaci pročitaju sa diska i kopiraju u bafer u memoriji, radi se o baferovanju.
  - Međutim, aplikacija može nadalje iste podatke čitati direktno iz memorije, bez trošenja vremena na pristup daleko sporijem disku, pa je time efektivno izvršeno i keširanje.

Spuler

- Spuler (SPOOL - simultaneous peripheral operations on-line) je u stvari bafer u kojem se čuvaju podaci koje treba proslediti uređaju.
  - Zgodan primer za demonstraciju spulera je štampač. Operativni sistem obezbeđuje da više aplikacija može zadati zahtev štampaču i proslediti podatke za štampanje u isto vreme, ali da se ti podaci štampaju redom.
    Podaci iz svake aplikacije se kopiraju u posebne datoteke.
    Kada su podaci jedne aplikacije odštampani, spuler preusmerava ulazni tok podataka tako da on postaje sledeća datoteka iz reda čekanja.

Spuler (2)

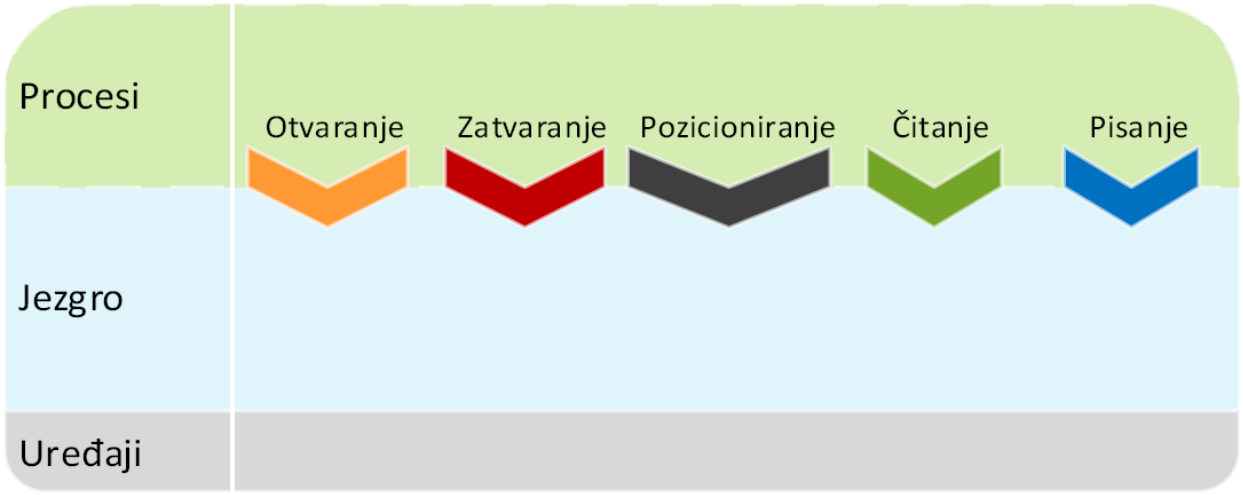

Zahtevi za štampanje
Proces 1
Proces 2
Proces 3
Ulazno izlazni podsistem
D1
D1
D2
D2
D3
D3
D1
Proces 1 - D3
Proces 3 - D1
Proces 2 - D3
Proces 2 - D2
Proces 1 -D2
Proces 2 - D1
Proces 1 - D1
Štampač
Spuler štampača
Spuler

Spuler (3)

- Korišćenje spulera doprinosi poboljšanju efikasnosti sistema.
- Na ovaj način se omogućava da proces podnese svoj zahtev, kopira podatke u spuler i nastavi sa daljim radom, bez čekanja da se operacija izvrši.

Interfejs ka korisničkim procesima

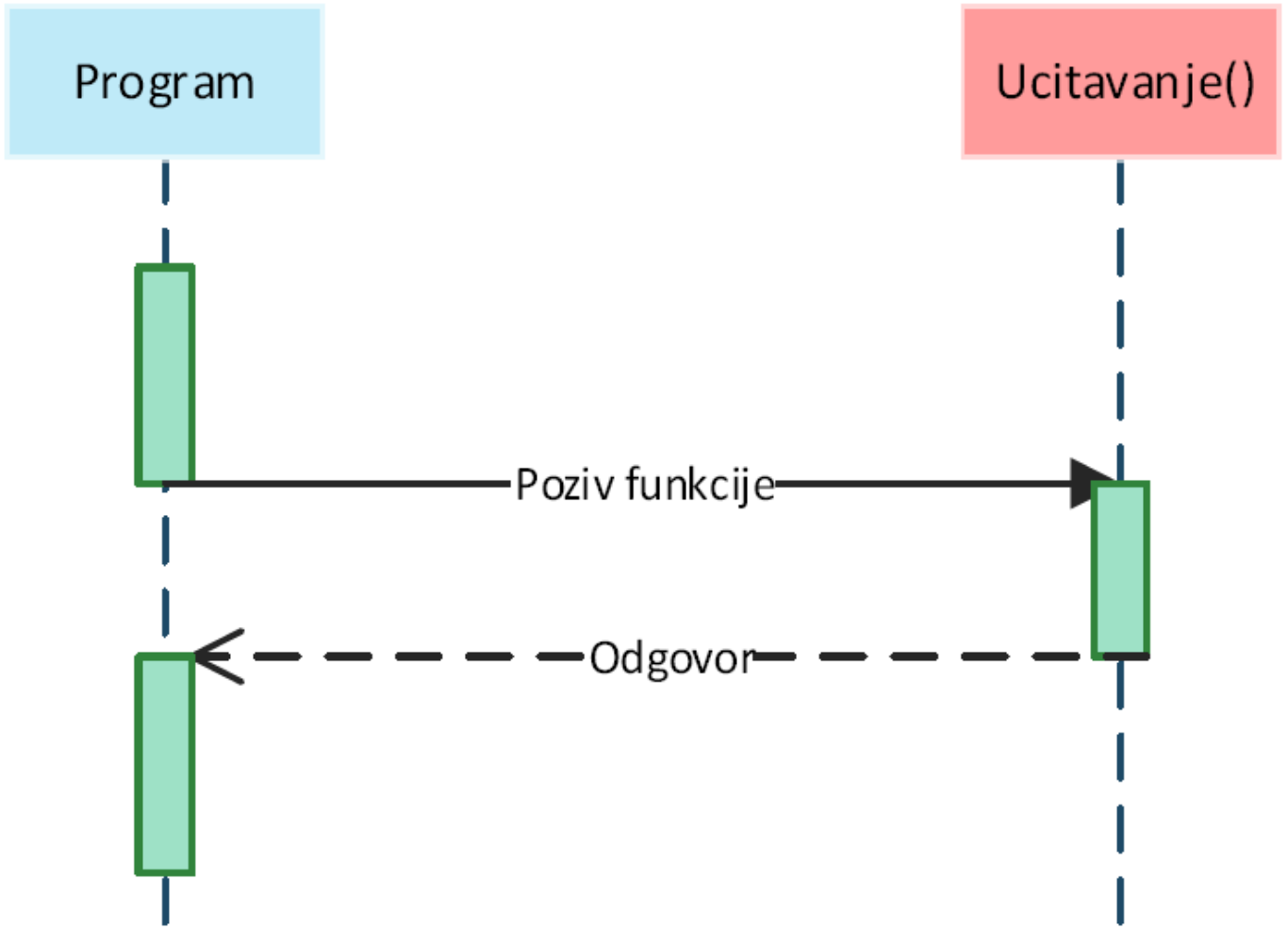
- Na vrhu hijerarhije upravljanja U/I uređajima se nalazi sloj koji predstavlja interfejs prema aplikativnom softveru odnosno ka procesima koji koriste uređaje.
- Sistemski pozivi su mehanizam kojim se korisničkim procesima obezbeđuje korišćenje uređaja.



Interfejs ka korisničkim procesima (2)

- Sistemski pozivi mogu biti blokirajući i neblokirajući.
  1. Kada aplikacija pozove blokirajuću funkciju, dalje izvršavanje aplikacije je suspendovano i tek kada sistemski poziv završi, aplikacija nastavlja dalje sa izvršavanjem, primajući povratnu vrednost sistemskog poziva.
     - Primer bi bio poziv funkcije koja učitava znak sa ulaza, i pritom čeka, blokirajući aplikaciju, sve dok se na ulazu ne pojavi znak.

Interfejs ka korisničkim procesima (3)

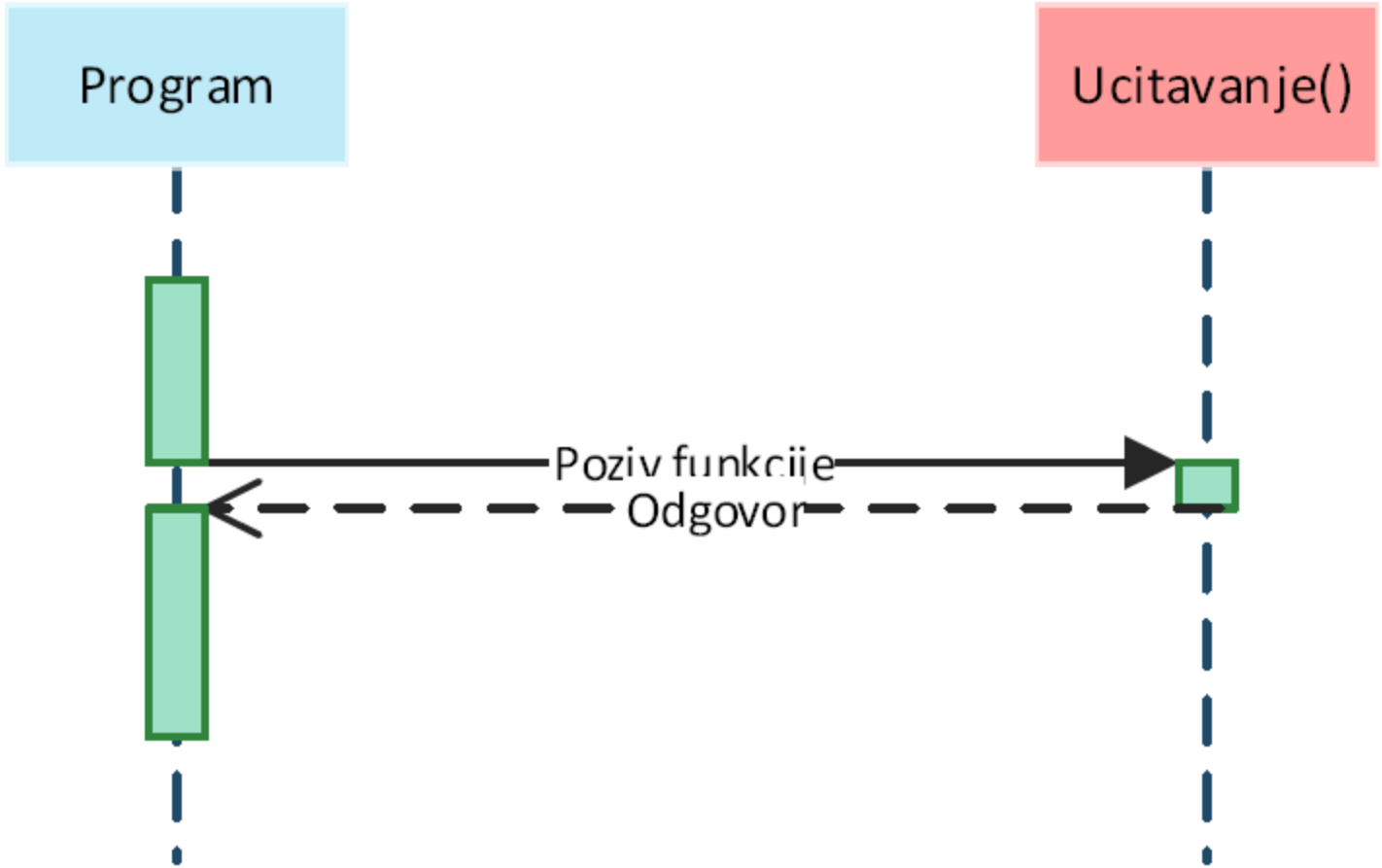


Blokirajući sistemski poziv

Interfejs ka korisničkim procesima (4)

2. Kada aplikacija pozove neblokirajuću funkciju, dalje izvršavanje aplikacije se nastavlja.
   - Neblokirajući sistemski poziv bi se mogao opisati na primeru funkcije koja učitava znak sa ulaza, kao u prethodnom primeru, ali ako na ulazu ne postoji jedan, funkcija se svejedno završava (vraća kod greške ili slično), i ne čeka da se na ulazu pojavi karakter da bi vratila vrednost.

Interfejs ka korisničkim procesima (5)

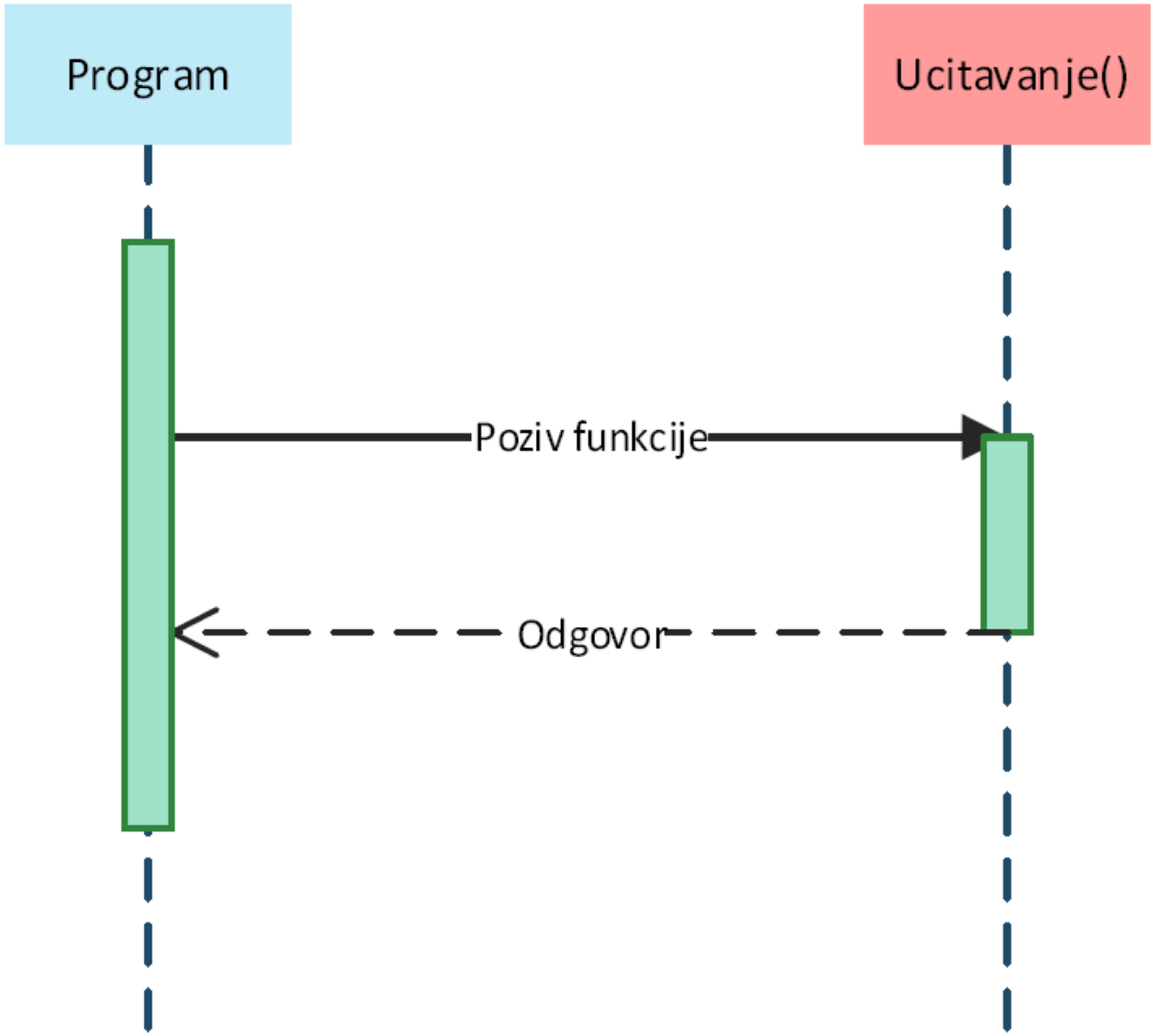


Neblokirajući sistemski poziv

Interfejs ka korisničkim procesima (6)

3. Alternativa neblokirajućim sistemskim pozivima jesu asinhroni pozivi.
   Asinhroni pozivi su neblokirajući u smislu da dozvoljavaju dalje izvršavanje aplikacije nakon što se pozovu, ali pritom ne vraćaju nužno vrednost odmah nakon poziva, već se izvršavaju paralelno sa aplikacijom.
   - Primer bi bila funkcija koja učitava znak sa ulaza i čeka dok se jedan ne pojavi, ali dopušta aplikaciji da nastavi dalji rad paralelno. Kada se znak pojavi, funkcija vraća vrednost aplikaciji.

Interfejs ka korisničkim procesima (7)



Asinhroni sistemski poziv

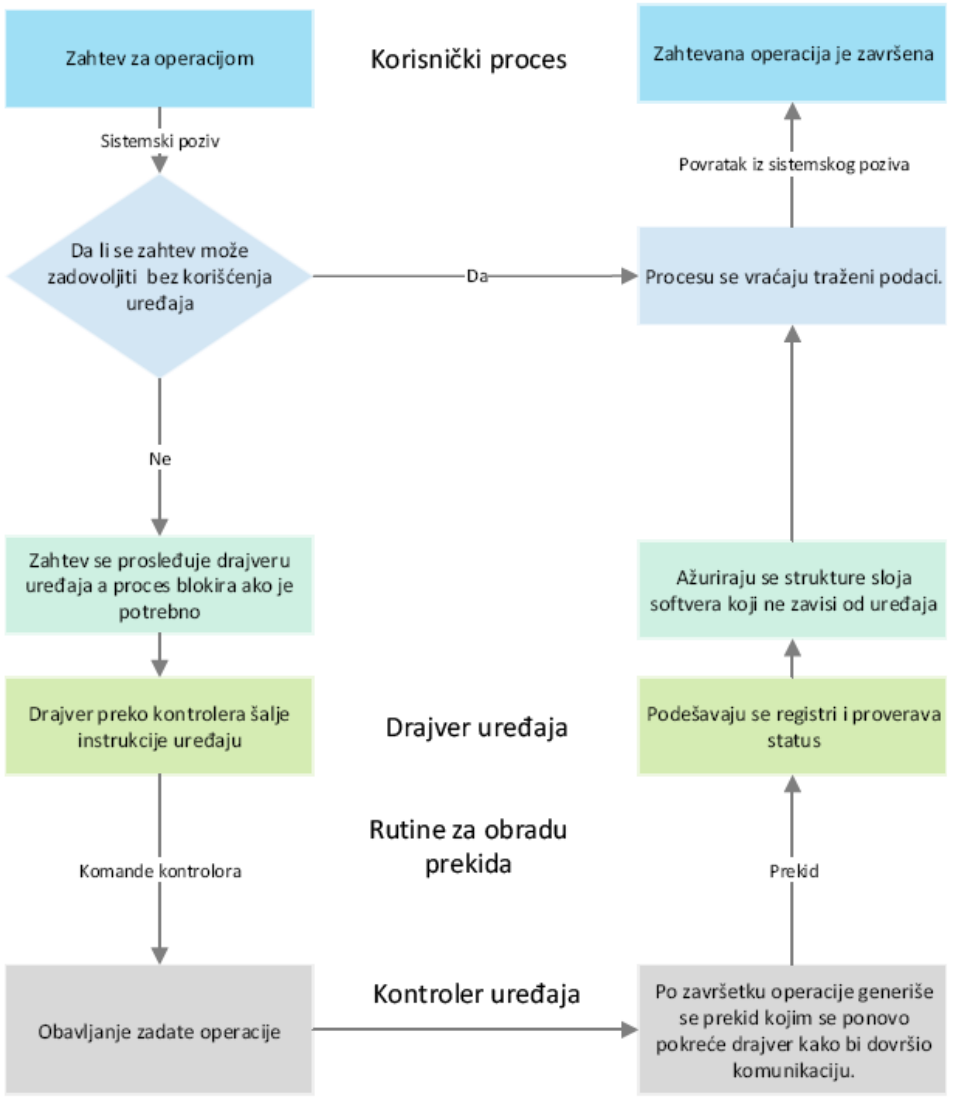
Interfejs ka korisničkim procesima (8)

- Primer komunikacije korisničkog procesa sa hardverom:
  1. Proces sistemskim pozivom saopštava svoj zahtev za operacijom na određenom uređaju.
  2. Sloj nezavisan od hardvera prima ovaj zahtev i proverava da li može da zadovolji zahtev bez korišćenja uređaja (npr. traženi podatak je već u keš memoriji) pa ako je to ispunjeno vraća rezultat procesu.
  3. U suprotnom, zahtev se prosleđuje drajveru uređaja a proces blokira ako od tražene operacije zavisi nastavak njegovog izvršavanja.
  4. Drajver preko kontrolera šalje instrukcije uređaju i obavlja zadatu operaciju.

Interfejs ka korisničkim procesima (9)

5. Po završetku operacije generiše se prekid kojim se ponovo pokreće drajver kako bi dovršio komunikaciju.
6. Podešavaju se registri i proverava status.
7. Ažuriraju se strukture sloja softvera koji ne zavisi od uređaja.
8. Procesu se vraćaju traženi podaci.

Interfejs ka korisničkim procesima (10)



Zahvalnica

Najveći deo materijala iz ove prezentacije je preuzet iz knjige „Operativni sistemi“ autora prof. dr Miroslava Marića i iz slajdova sa predavanja koje je držao prof. dr Marić.

Hvala prof. dr Mariću na datoj saglasnosti za korišćenje tih materijala.