Operativni sistemi
- Struktura operativnih sistema -

Veljko Stanković

Uvod

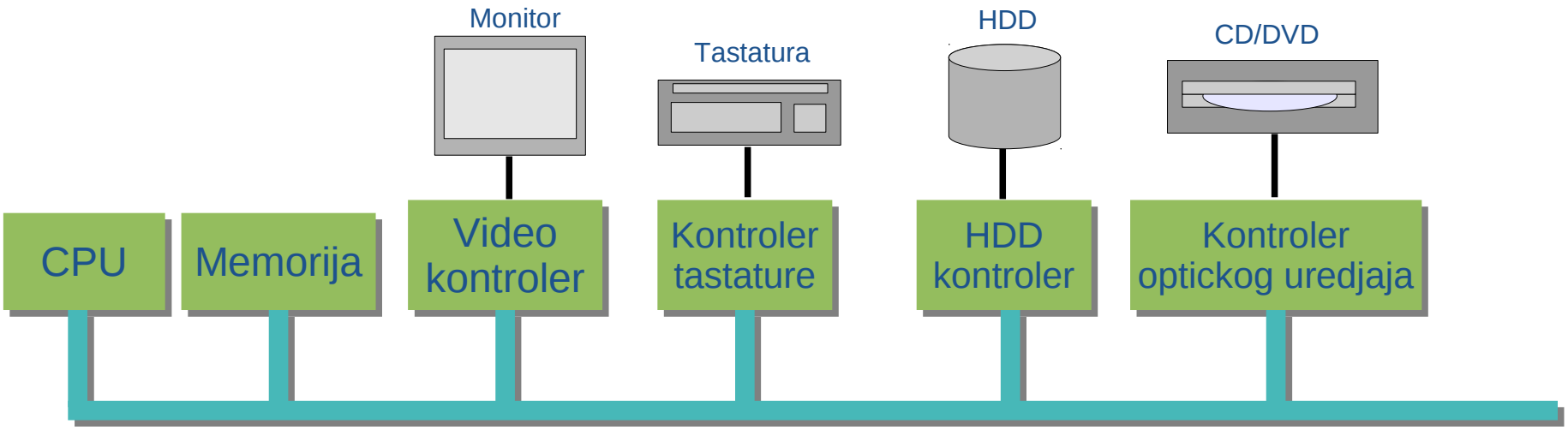
— Arhitektura i organizacija racunara

- Funkcionisanje OS-a je usko povezano sa hardverom racunara na kome radi.
- Neophodno je da bude prilagodjen onome kako programer vidi racunar, njegovim mogucnostima i strukturom.
- OS upravlja i kontrolise I/O uredjaje
- OS mora da obezbedi ispravno funkcionisanje sistema

Organizacija računara

— Moderni racunari se sastoje od CPU i odredjenog broja kontrolera povezanih uz pomoc jedne ili vise magistrala.

- Svaki kontroler je zaduzen da upravlja radom jednom vrstom uredjaja.
  - ☑ HDD, audio uredjaji, monitori...



Organizacija računara

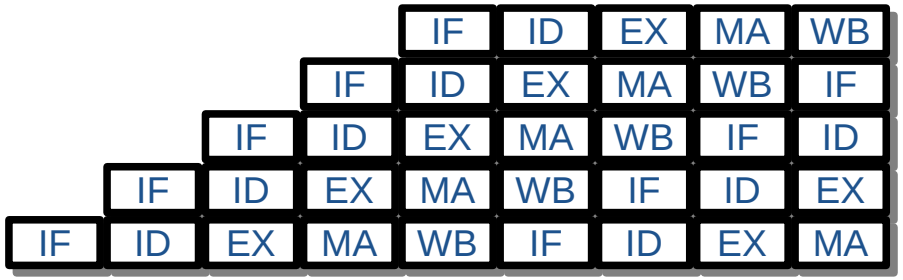
1010011
1110100
1100001
1010110

- CPU predstavlja “mozak” racunara
  - Ciklus izvrsenja instrukcija obuhvata sledeće faze:
    - ☑ CPU pribavlja instrukciju iz glavne memorije računara
    - ☑ Instrukcija se dekodira da bi se utvrdio njen tip i operandi
    - ☑ Ukoliko je potrebno pribavljaju se operandi i izvršava se instrukcija
    - ☑ Rezultat operacije se smešta u glavnu memoriju
  - Svaki CPU ima poseban skup instrukcija koje može da izvršava.
  - Svaki CPU sadrži određeni broj registara u kojima smešta privremene podatke i međurezultate.
  - Uobičajeni skup instrukcija uključuje:
    - ☑ Instrukciju za pribavljanje podataka iz glavne memorije u registre CPU-a.
    - ☑ Instrukciju za smeštanje podataka iz regustra CPU-a u glavnu memoriju.
  - Prilikom izvršenja instrukcija kombinuju se podaci iz registara, memorije ili iz registara i memorije kako bi se dobio rezultat.

— Registri CPU

- CPU obično sadrži određeni broj registara opste namene
  - ☑ Sluze za smestanje podataka i medjurezultata
- Programski brojac (program counter)
  - ☑ Sadrzi adresu sledece instrukcije koju treba izvrsiti
- Pokazivac vrha magacina (stack pointer)
  - ☑ Adresa vrha magacina
  - ☑ Magacina sadrzi deo za svaki program koji se trenutno izvrsava
  - ☑ Deo magacina koji pripada nekom programu sadrzi one ulazne, lokalne i pomocne promenljive koje nisu smestene u registrima CPU-a
- Programska statusna rec (Program Status Word – PSW)
  - ☑ Sadrzi kontrolne bite koje postavljaju rezultat instrukcija za kontrolu toka programa, mod rada CPU, prioritet izvrsavanja itd.
  - ☑ Korisnicki programi mogu da pristupe obicno samo nekim od kontrolnih bita u PSW
  - ☑ PSW veoma znacajan pri izvrsenju I/O instrukcija

- OS mora voditi racuna o sadrzaju svih registara CPU-a.
  - Prilikom prelaska sa izvrsenja jednog programa na drugi, OS mora da memorise sadrzaj svih registara kako bi program kasnije mogao da nastavi sa izvrsenjem.
- Da bi se ubrzao rad procesora vrsi se protocna obrada instrukcija
  - RISC procesori, Pipelining
  - Umesto da ceka da se zavrsi izvrsenje jedne instrukcije da bi krenuo sa izvrsenjem sledece, CPU obradjuje vise instrukcija istovremeno.
  - Protocna obrada instrukcija nosi mnogo problema za programera koji mora da ih predvidi prilikom pisanja programa
  - Superskalarni procesor
  - Superprotocni procesor



- Vecina CPU-a, izuzev najjednostavnijih, ima dva moda rada
  - Korisnicki mod
  - Zasticeni (supervizorski, kernel) mod
- Mod je naznacen bitom u okviru PSW registra
  - U zasticenom modu, CPU moze da izvrsi bilo koju instrukciju i pristupi svom raspolozivom hardveru
    - ☑ OS se izvrsava u zasticenom modu
  - Korisnicki programi se izvrsavaju u korisnickom modu
    - ☑ Imaju pristup samo odredjenim instrukcijama i resursima sistema
    - ☑ Sve I/O instrukcije i instrukcije za upravljanje memorijom su zabranjene
    - ☑ Pristup mod bitu PSW registra je zabranjen

Organizacija racunara

- Da bi dobio usluge OS-a, korisnicki program koristi **sistemske pozive** (system call)
  - Trap je ekvivalent prekida
- Sistemski poziv (trap)
  - Korisnicki program poziva OS
  - OS opsluzuje sistemski poziv u zasticenom modu
  - Nakon opsluzivanja poziva, OS prelazi u korisnicki mod i vraca kontrolu korisnickom programu od instrukcije koja sledi nakon sistemskog poziva
- Memorija
  - U idealnom slucaju memorija bi trebala da bude veoma brza, sa dosta prostora i jeftina.
  - Kako ovo nije moguce, memorija racunara je organizovana hijerarhijski

Organizacija racunara

— Memorija racunara

- Registri
  - ☑ U okviru CPU, brzi, skupa izrada, mali broj
  - ☑ Kapacitet registara je 32×32 (32b CPU) i 64×64 (64b CPU).
- Kes memorija (cache mempry)
  - ☑ Zasnovana na predpostavci vremenskoj i prostornoj lokalnosti izvrsavanja programa
    - Velika verovatnoca da ce program u odredjenom vremenu pristupati odredjenoj grupi podataka ili instrukcija u memoriji
  - ☑ Kada program treba da ucita podatak iz memorije, on prvo proverava da li se on nalazi u kes memoriji.
  - ☑ Ako se podatak nalazi u kes memoriji to se naziva kes pogotkom i ne pristupa se glavnoj memoriji.
  - ☑ Posebnim algoritmima za pribavljanje podataka, moguce je imati verovatnocu pogotka i do 99%
  - ☑ Kes memorija je zbog cene i brzine organizovana u nivoe.

Organizacija racunara

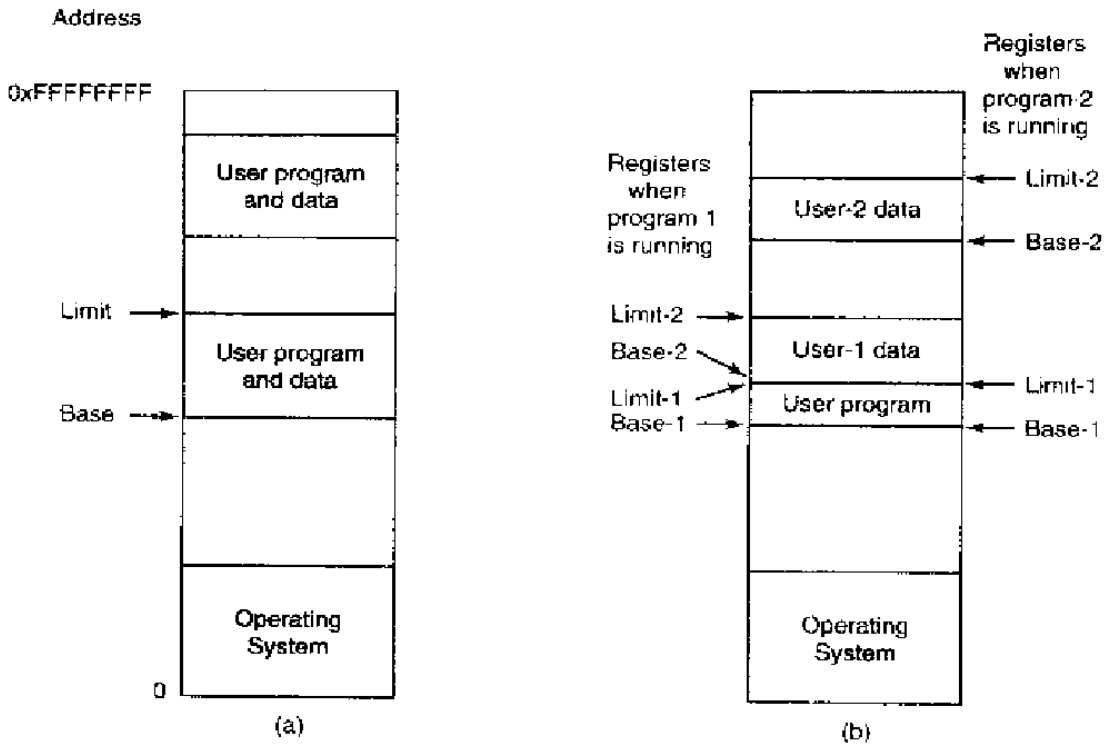
- Memorija racunara
  - Glavna memorija
    - ☑ Memorija sa slucajnim pristupom (RAM – Random Access Memory)
    - ☑ Znatno vecg kapaciteta od kes memorije i sporija
  - Cvrsti disk (HDD – Hard disk drive)
    - ☑ Veoma veliki kapacitet i znatno jeftinija od glavne memorije
    - ☑ Mehanicki uredjaj pa je i brzina pristupa podacima znatno manja
    - ☑ Sastoji se iz jedne ili vise metalnih ploca koje rotiraju brzinom od 5400, 7200 ili 10800 obrtaja u minuti.
    - ☑ Upisi i citanje podataka se vrsi preko magnetne glave lociranoj na mehanickoj polugi
    - ☑ Prostor na HDD je podeljen na staze, sektore i cilindre.
  - Read only memory (ROM)
  - EEPROM (Electrically Erasable Programmable ROM)
    - ☑ Ucitavanje bootstrap loader-a, parametri sistema (CMOS) itd.

- Glavna memorija
  - Pozelno je imati vise programa u glavnoj memoriji koji se konkurentno izvrsavaju
  - Problem
    - ☑ Kako zastiti programe jedne od drugih i kernel od njih
    - ☑ Kako izvrsiti realokaciju programa
- Relokacija programa u glavnoj memoriji
  - Tokom izvrsenja, program se moze vise puta unositi u glavnu memoriju i smestati na HDD
  - Sve adrese su relativne
    - ☑ Bazni registar (Bazno adresiranje, PC – relativno, indeksno adresiranje)
    - ☑ Granica
      - Pokazuje kolika je velicina segmenta memorije (stranice)
  - Resen problem zastite i relokacije programa

Organizacija racunara

— Jedinica za upravljanje memorijom (MMU – Memory Management Unit)

- Vrsi prevodjenje relativnih adresa u programu u fizicke adrese u memoriji
- Nalazi se na CPU cipu, ali je logicki smestena izmedju CPU-a i memorije



- Memorija utice na performanse sistema
  - Performanse sistema zavise od politike smestaja i politike zamene podataka u kes memoriji
    - ☑ Ako se neki program duze izvrsava stopa pogodaka u kes memoriji ce biti velika
    - ☑ Pri cestoj zameni programa koji se izvrsavaju neophodno je unapred pribaviti podatke kako stopa pogodaka ne bila mala
  - Stanje pri izvrsenju programa je odredjeno stanjem registara MMU
    - ☑ Pri promeni programa koji se izvrsavaju (context switch) potrebno je sacuvati sadrzaj ovih registara
    - ☑ U praksi ne tako jednostavan problem

— I/O uredjaji

- I/O uredjaji se obicno sastoje iz kontrolera i samog uredjaja
- Kontroler je cip koji se nalazi na maticnoj ploci i fizicki kontrolise rad I/O uredjaja i prima naredbe od OS-a
  - ☑ Upravljanje I/O uredjajem je veoma komplikovano pa I/O kontroler ima zadatak da za OS uredjaj predstavi preko jednostavnog interfejsa
- Posto se I/O kontroleri razlikuju, za svaki je potreban drugaciji softver da bi se njime upravljalo
- Softver koji se koristi za upravljanjem I/O kontrolerom se naziva drajver (device driver)
  - ☑ Da bi mogao da se koristi, drajver je potrebno smestiti u OS kako bi mogao da se izvrsava u zasticenom modu

- Postoje tri nacina da se drajver smesti u OS
  - Izvrsi se ponovo linkovanje kernela sa novim drajverom a potom se sistem restartuje
    - ☑ Mnogi UNIX sistemi rade na ovaj nacin
  - OS-u se naznacava da je potreban novi drajver, sistem se restartuje i tokom boot-ovanja, OS nalazi novi drajver i ucitava ih.
    - ☑ Windows
  - OS moze da nadje i prihvati nove drajvere bez potrebe da prekida rad i restartuje sistem.
    - ☑ USB ili IEEE 1394 uredjaji uvek zahtevaju dinamicki ucitane drajvere

- Svaki I/O kontroler ima odredjeni (mali) broj registara koji se koriste za komunikaciju sa sistemom
- Adresiranje I/O uredjaja
  - Preslikani I/O memorijski prostor
    - ☑ Glavna memorija i I/O uredjaji dele adresni prostor
    - ☑ Koriste se iste naredbe za pristup memoriji i I/O
  - Izdvojeni I/O memorijski prostor
    - ☑ Razlicit adresni prostor
    - ☑ Razlicite naredbe za pristup memoriji i I/O uredjajima
- Vrste I/O
  - Programirani I/O
  - Prekidacki I/O
  - Direktni pristup memoriji (DMA – Direct Memory Address)

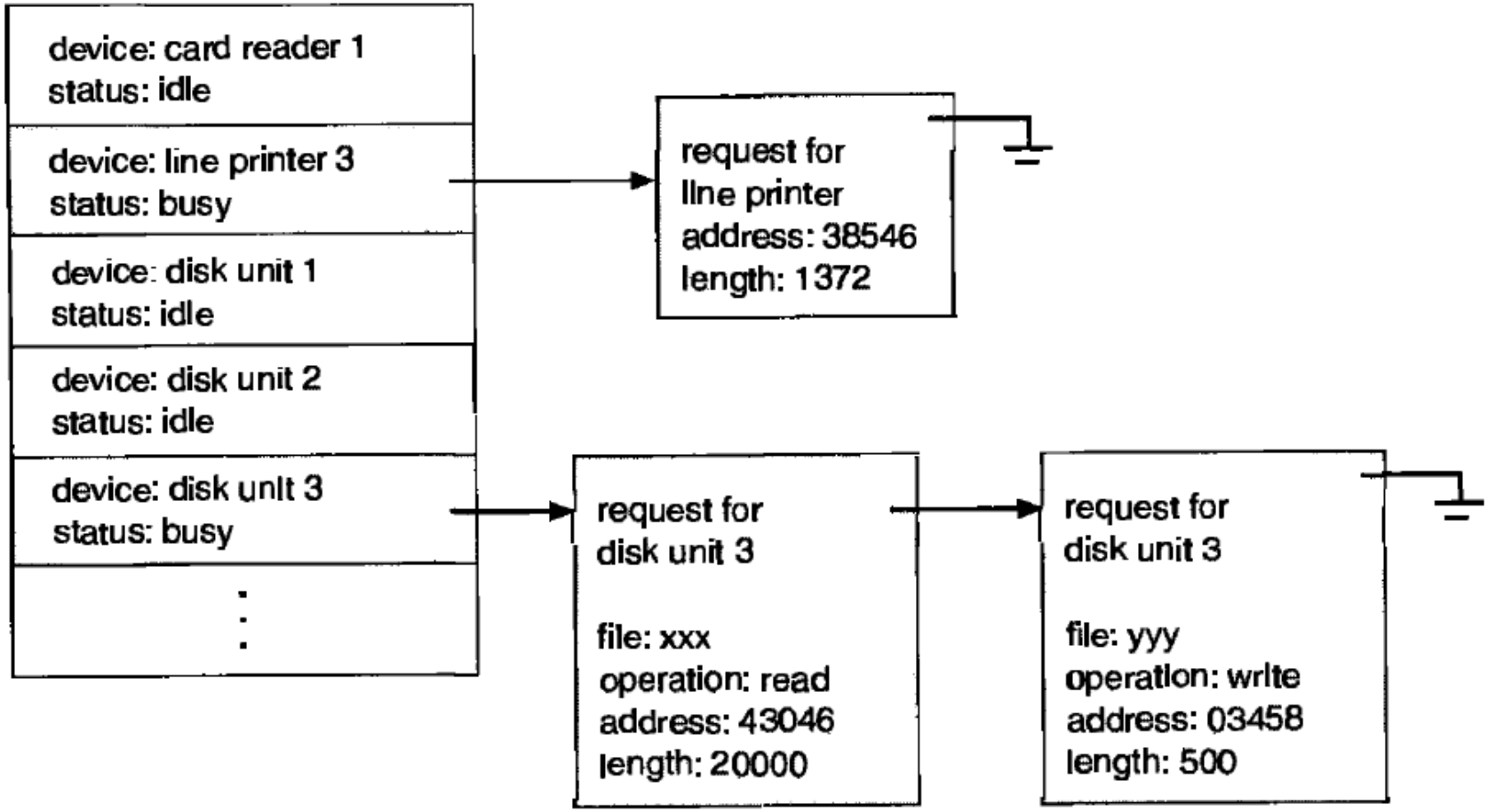
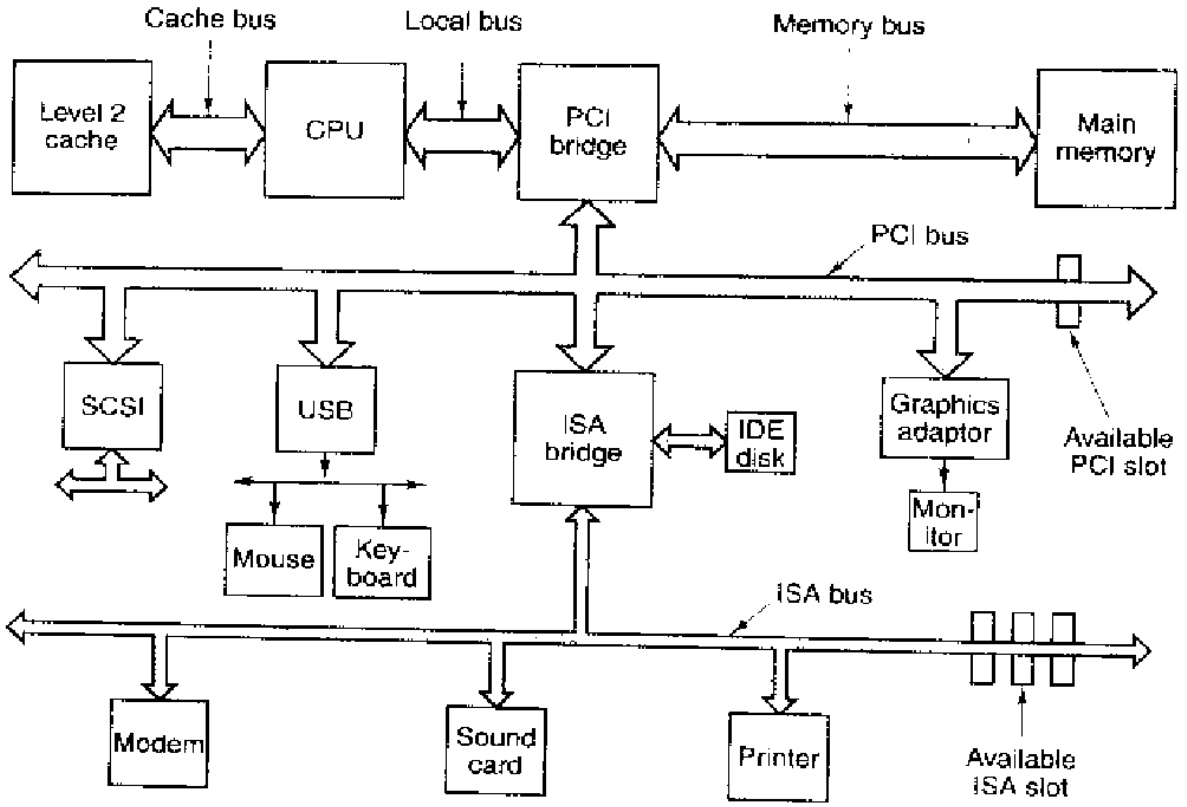


Figure 2.4 Device-status table.

Organizacija racunara

— Sistem magistrala racunarskog sistema predstavlja skup vise razlicitih magistrala koje se koriste za povezivanje razlicitih komponenti u okviru racunarskog sistema.



Organizacija racunara

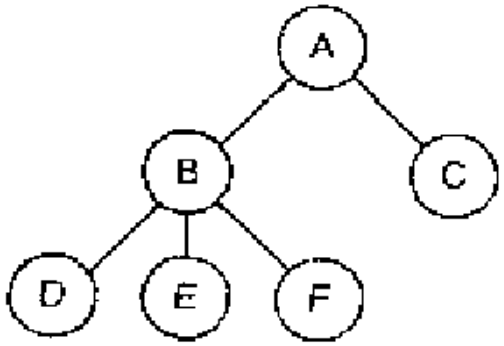
- Magistrale
  - Problem dodele i arbitraze na magistrali
  - Asinhrone i sinhrone magistrale
- Glavne magistrale koje se mogu sresti u racunarskim sistemima
  - ISA – Industry Standard Architecture
  - PCI – Peripheral Component Interconnect
- Specijalne magistrale
  - IDE, SATA
  - USB
  - SCSI

Struktura operativnih sistema

- Svi OS su zasnovani na grupi zajednickih pojmova
- Proces je kljucni koncept OS-a
  - U sustini predstavlja program koji se izvrsava
  - Svakom procesu je pridruzen adresni prostor koji je opisan nekom minimalnom i maksimalnom adresom
    - ☑ Adresni prostor procesa sadrzi instrukcije, podatke i magacin
  - Svakom procesu je pridruzen skup registara (PC, pokazivac magacina...)
  - Prilikom prelaska sa izvrsenja jednog procesa na drugi, potrebno je negde zapamtiti sve informacije o procesu kako bi mogao kasnije da nastavi sa izvrsavanjem.
  - Sve informacije o procesu (izuzev njegovog adresnog prostora) se smestaju u tabelu procesa
    - ☑ Vektor podataka tipa structure

Struktura operativnih sistema

- Suspendovani proces se sastoji od slike njegovog adresnog prostora i procesne tabele
  - Procesna tabela sadrzi izmedju ostalog podatke iz registara tokom izvrsenja programa
- Najvazniji sistemski pozivi za upravljanje procesima su pozivi za kreiranje i obustavljanje procesa
  - Primer: Korisnik unosi komandu u komandni interpreter (shell)
    - ☑ Interpreter kreira proces za izvrsenje komande
    - ☑ Na kraju izvrsenja, proces izvrsava poziv za njegovo uklanjanje
  - Svaki proces moze da kreira drugi proces potomak



Struktura operativnih sistema

- Procesi koji medjusobno kooperiraju pri izvrsavanju nekog zadatka moraju biti u stanju da medjusobno komuniciraju
  - Medju procesna komunikacija
- Sistemski pozivi za dodelu memorije procesima
- Prenos podataka procesu na drugom racunaru
  - Proces salje poruku preko racunarske mreze i zahteva od OS-a da ga obavesti posle odredjenog vremena da je potvrda prijema stigla ili ne.
  - OS nakon odredjenog vremena kontaktira proces
  - Proces prekida sa izvrsavanjem i smesta sadrzaj registara CPU u magacin
  - Proces pokrece rutinu za obradu poruke od strane OS-a (da li je poruka primljena ili ne – retransmisija)
  - Nakon zavrsetka rutine proces nastavlja sa izvrsavanjem

Struktura operativnih sistema

- U slucaju UNIX/LINUX-a
  - Svakom procesu se dodeljuje UID (User ID)
    - ☑ Svaki potomak procesa preuzima UID korisnika
  - Korisnici se mogu grupisati i svakoj grupi se moze dodeliti grupni ID – GID
  - Supreuser moze da izvrsava neke od instrukcija u zasticenom modu
- Problem blokiranja (deadlock)
  - Problem koji se javlja kada je sistem blokiran jer vise procesa koji su medjusobno uslovljeni ceka na izvrsenje ostalih.
    - ☑ Predpostavimo da dva procesa trebaju da prebace podatke sa HDD na CDROM
      - P1 zahteva i dobija odobrenje da pristupi HDD
      - P2 zahteva i dobija odobrenje da pristupi CDROM
      - P1 zatim zahteva CDROM i biva suspendovan dok se CDROM ne oslobodi
      - P2 zahteva HDD i biva takodje suspendovan dok se on ne oslobodi

Struktura operativnih sistema

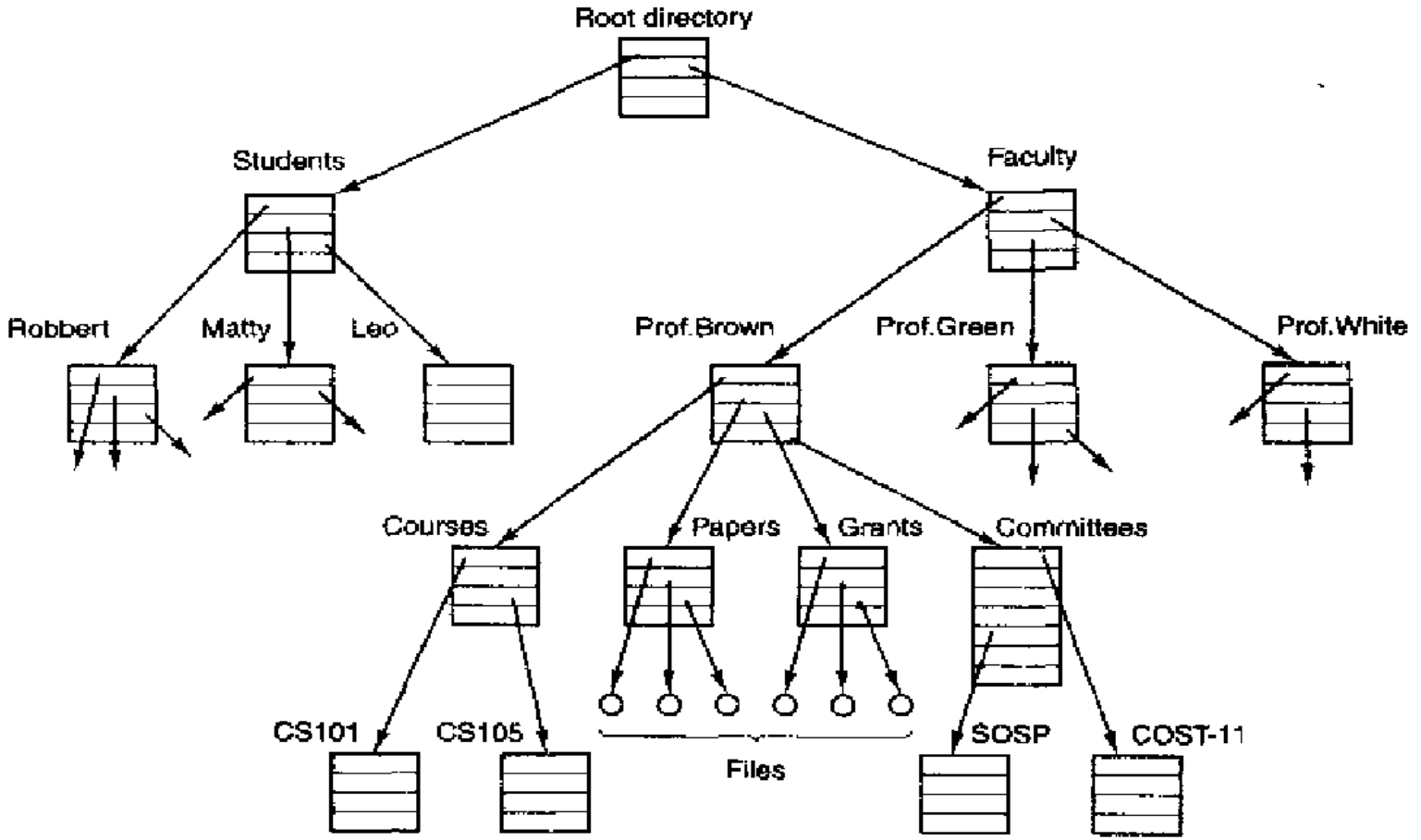
- OS ima funkciju upravljanja memorijom
  - U glavnoj memoriji se nalazi vise programa koji se konkurentno izvrsavaju
  - Da ne bi doslo do mesanja programa potrebno uvesti neki zastitni mehanizam
  - Slican problem se odnosi na adresni prostor procesa
    - ☑ Ako je adresni prostor procesa manji od velicine glavne memorije → nema problema
    - ☑ Ako je velicina adresnog prostora veca od velicine raspolozivog prostora u glavnoj memoriji, ona se prosiruje primenom tehnike poznate pod imenom viertuelna memorija

Struktura operativnih sistema

- Fajl sistem (organizacija datoteka)
  - Veoma vazan koncept OS-a jeste fajl sistem
  - Da bi obezbedili mesto gde se mogu cuvati podaci, vecina OS-a koristi koncept direktorijuma kao nacin grupisanja fajlova (datoteka)
    - ☑ Uvode se sistemski pozivi za kreiranje i brisanje direktorijuma
  - Fajlovi su kao i procesi organizovani hijerarhijski
  - Lokacija svakog fajla je odredjena putanjom od vrha hijerarhije (root direktorijum)
    - ☑ Apsolutna putanja daje put kojim treba proci da bi se doslo do datog fajla
  - U svakom trenutku, svaki proces ima svoj radni direktorijum
  - Pre nogo sto pristupi nekom fajli, OS proverava mogucnost pristupa fajlu
    - ☑ Ako proces sme da pristupi fajlu, OS vraca broj koji se zove fajl deskriptor, u suprotnom vraca poruku da je doslo do greske

Struktura operativnih sistema

1010011
1110100
1100001
1010110



Struktura operativnih sistema

1010011
1110100
1100001
1010110

- Neke od struktura OS-a koje se javljaju najcesce u praksi:
  - Monolitski OS
  - Hijerarhijski
  - Virtuelne masine
  - Egzokernel
  - Klijent-server sistemi
- Ova podela nije potpuna i moguce su i druge vrste OS.

Struktura operativnih sistema

- Monolitski kernel
  - Ceo kernel se izvrsava u zasticenom modu
  - OS je napisan kao kolekcija procedura.
    - ☑ Svaka moze da pozove bilo koju drugu u svakom trenutku
  - Svaka procedura ima jasno definisan interfejs (ulazne i izlazne promenljive)
  - Sistem nije u potpunosti bez ikakve strukture – neke strukture ima
    - ☑ Prilikom poziva procedura OS, parametri se smestaju na dobro definisanu lokaciju (magacin) i potom se aktivira trap
    - ☑ Organizacija sistemskih poziva i usluznih procedura kojie omogucavaju izvrsavanje sistemskih poziva (funkcije zajednicke za sistemske pozive)
  - Posto se sve procedure izvrsavaju u jednom adresnom prostoru, greska u jednoj proceduri moze da narusi ceo sistem
  - Ukoliko su procedure OS-a dobro isprojektovane, jaka integrisanost procedura OS-a omogucava veoma efikasno iskoriscenje resursa sistema.

Struktura operativnih sistema

- Hijerarhijski kernel / Visenivooski OS
  - OS je orgnizovan u vise nivoa
    - ☑ Najnizi (nivo 0) je hardver
  - Svaki nivo koristi usluge nizeg nivoa kako bi opsluzio zahteve viseg nivoa
  - Glavna prednost ovog pristupa je modularnost, tj. svaki nivo koristi usluge samo nizeg nivoa
    - ☑ Visi nivo ne mora da zna kako su implementirane pojedinacne funkcije na nizem nivou, vec samo sta one rade
    - ☑ Lakse debagiranje i verifikacija sistema
  - THE OS (Technische Hogeschool Eindhoven)

Nivo 5: Korisnicki programi
Nivo 4: I/O bafer
Nivo 3: Drajver korisnicke konzole
Nivo 2: Upravljanje memorijom
Nivo 1: Dodela CPU
Nivo 0: Hardver

Struktura operativnih sistema

- Hijerarhijski kernel / Visenivooski OS
  - Najveci izazov vezan za hijerarhijske sistem jeste kako definisati pojedine nivoe, tj kako grupisati pojedina funkcije OS-a u nivoe
  - Problem: manja efikasnost u odnosu na monolitske sisteme
    - ☑ Korisnicki program koji zeli da izvrsi I/O operaciju, izvrsava sistemski poziv koji prihvata I/O nivo koji poziva nivo za upravljanje memorijom, zatim se poziva nivo na kome se vrsi dodela CPU i tek na kraju se poziv izvrsava na hardveru
  - Na svakom nivou se dodaje zaglavlje sistemskom pozivu i kao rezultat, potrebno je vise vremena da se on opsluzi
  - Da bise povecala efikasnost ovih sistema
    - ☑ Broj nivoa treba da bude sto manji
    - ☑ Nivoi se zajednicki projektuju a ne kao pojedinacne celine

Struktura operativnih sistema

1010011
1110100
1100001
1010110

— Virtualne masine

- OS obezbedjuje skup sistemskih poziva koje koriste korisnicki programi
  - ☑ Sistemski programi tretiraju sistemske pozive i hardverske instrukcije na isti nacin i ne prave razliku izmedju njih
    - Sa stanovista sistemskih programa, sistemski pozivi i hardverske instrukcije se nalaze na istom nivou
- VM ostavlja utisak izvrsavanja programa na nezavisnim sistemima primenom CPU multitaskinga i virtuelne memorije
- VM predstavlja interfejs koji je identican hardveru na kome se izvrsava
  - ☑ Imamo kopije OS-a koje se izvrsavaju na svojim VM
  - ☑ Svaki interupt ne ide direktno ka hardveru vec biva obradjen od strane VM koja kombinuje sve zahteve ka resursima sistema i izvrsava ih na hardveru
- Java Virtual Machine (JVM)
  - ☑ slican koncept
  - ☑ obezbedjuje portabilnost softvera preko mreze

Struktura operativnih sistema

- Egzokernel
  - Korak dalje u odnosu na VM
  - Svaki korisnik dobija kopiju sistema ali sa posebnim delom resursa sistema
    - ☑ Primer: HDD se deli na disjunktne oblasti koje se dodeljuju razlicitim korisnicima
  - Cilj je da se uvede sto manje apstrakcija nizih nivoa
  - Veoma su mali
  - Egzokerneli imaju zadatak da dodele resurse sistema korisnicima i zatim obezbede da razliciti korisnici (procesi) ne pokusavaju da pristupe tudjim resursima
    - ☑ Multipleksiranje resursa i zastita (multiplexing and protection)
  - Egzokernel ima zadatak da prati kome je dodelio koje resurse
  - Efikasniji od VM

Struktura operativnih sistema

- Mikrokernel (Klijent-server model)
  - Mikrokernel ne predstavlja OS u smislu klasicne definicije
  - Ne pruza funkcije OS-a vec samo obezbedjuje uslove da se one implementiraju
    - ☑ Upravljanje memorijom
    - ☑ Dodela CPU
    - ☑ Komunikacija izmedju procesa (IPC – Inter-process communication)
  - Mikrokernel je jedini deo sistema koji se izvrsava u zasticenom modu
  - Funkcije OS-a pruzaju razliciti serveri u korisnickom modu
    - ☑ Drajveri uredjaja
    - ☑ Protokoli
    - ☑ Fajl sistem
    - ☑ Korisnicki interfejs
  - Korisnicki proces (klijent) salje zahtev odgovarajucem serveru koji obavlja zadatak i rezultat salje klijentu

Struktura operativnih sistema

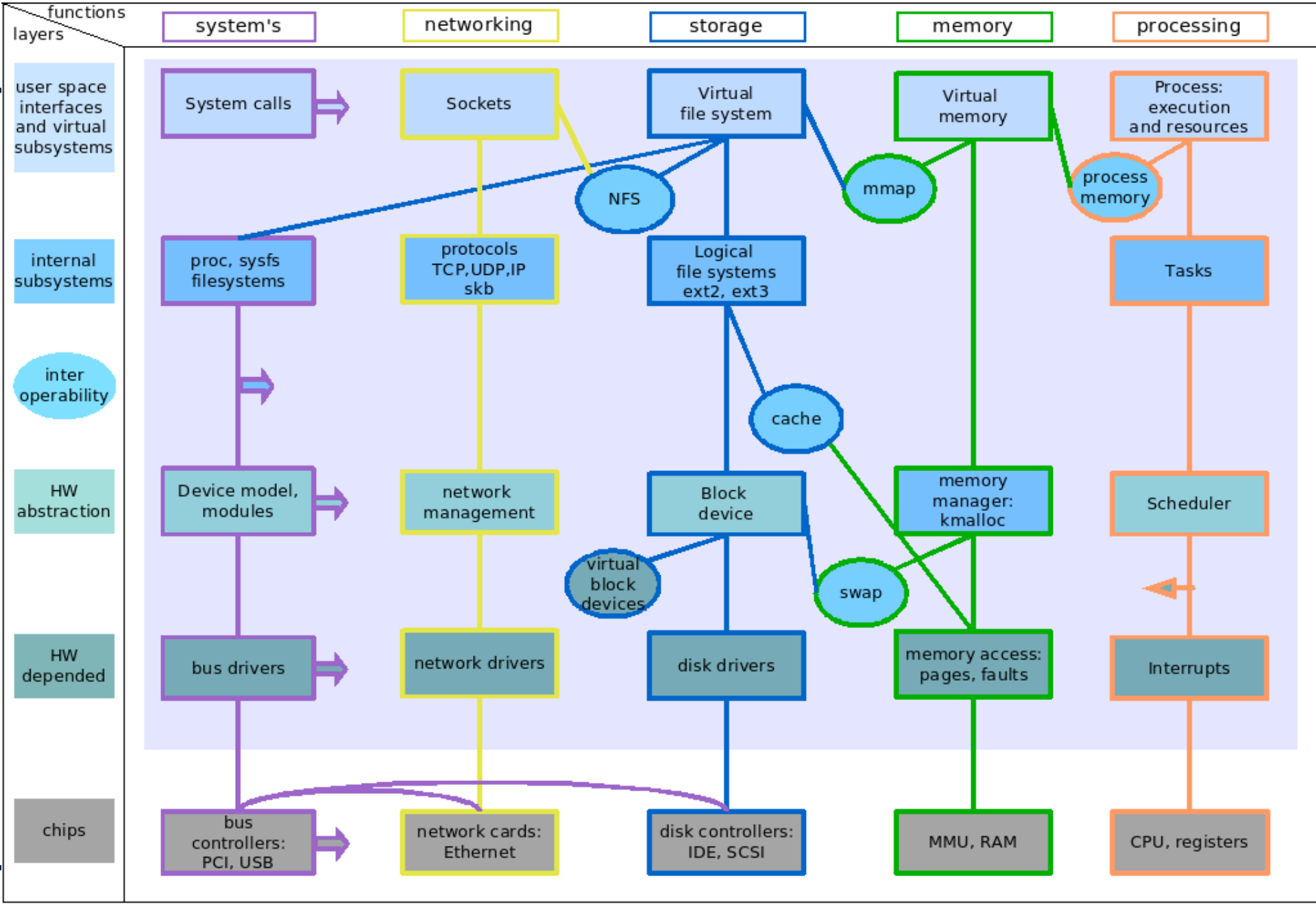
- Mikrokernel (Klijent-server model)
  - Zadatak mikrokernela se svodi na obezbedjivanje komunikacije izmedju klijenta i servera
  - Podelom OS-a na manje delove oni postaju laksi za razvoj, kontrolu i promenu
  - Posto se svi server procesi izvrsavaju u korisnickom modu, oni nemaju direktan pristup hardveru
    - ☑ Greska u serveru moze dovesti do njegovog pada (prestanka rada) ali ne i do prestanka rada celog sistema
  - Klijent i srever komuniciraju preko poruka sto omogucava implementaciju na distribuiranom sistemu (klijent ne zna da li je zahtev obradjen lokalno na istom sistemu ili na nekom drugom)

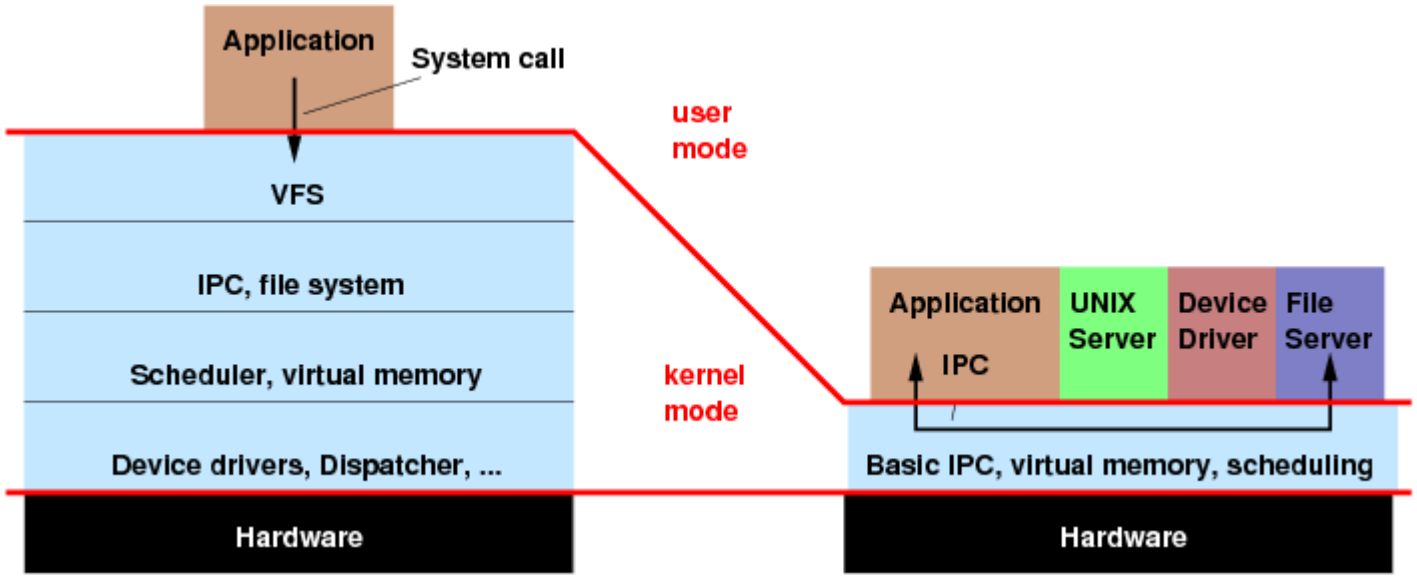
Struktura operativnih sistema

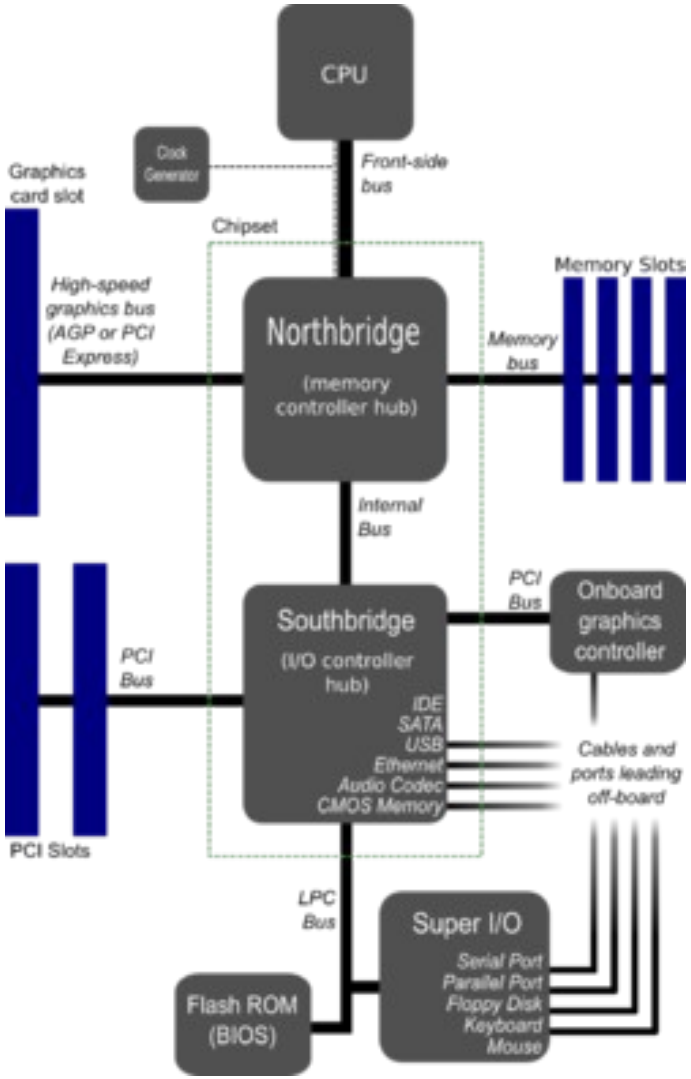
1010011
1110100
1100001
1010110

- Mikrokernel (Klijent-server model)
  - Problem se javlja kada neke zadatke, serveri nemogu da obave u korisnickom modu
    - ☑ Omogucava se da se kriticni server procesi izvrsavaju u zasticenom modu pri cemu i dalje sa ostalim procesima komuniciraju preko poruka
    - ☑ Minimalan broj funkcija se ugradjuje u kernel ali se bitne odluke ostavljaju serverima u korisnickom modu rada
      - Server odredjuje da li neke zahteve klijentskih procesa treba ili ne izvrsiti
      - Stvarno izvrsenje zahteva se obavlja od strane kernela

Simplified Linux kernel diagram in form of a matrix map



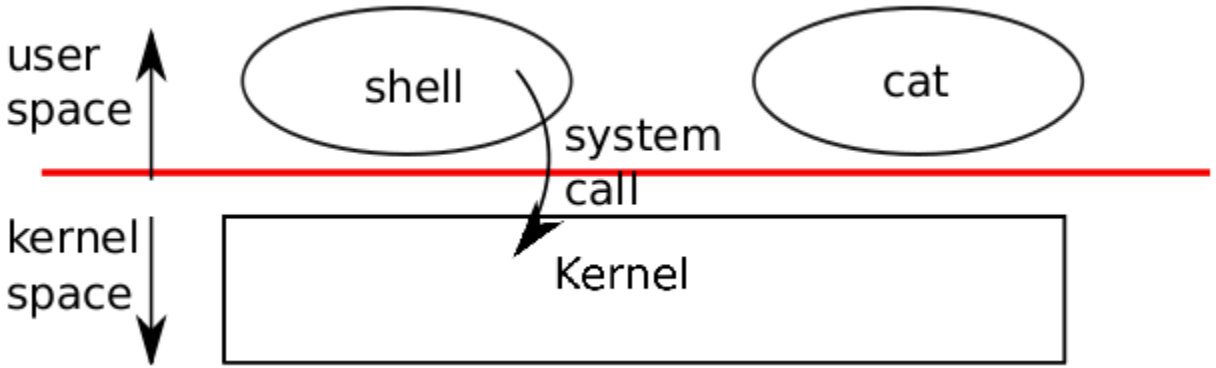

Application
System call
VFS
IPC, file system
Scheduler, virtual memory
Device drivers, Dispatcher, ...
Hardware
user mode
kernel mode
Application
IPC
UNIX Server
Device Driver
File Server
Basic IPC, virtual memory, scheduling
Hardware


CPU
Clock Generator
Front-side bus
Graphics card slot
Chipset
Memory Slots
High-speed graphics bus (AGP or PCI Express)
Northbridge
(memory controller hub)
Memory bus
Internal Bus
PCI Bus
Onboard graphics controller
Southbridge
(I/O controller hub)
PCI Bus
IDE
SATA
USB
Ethernet
Audio Codec
CMOS Memory
Cables and ports leading off-board
PCI Slots
LPC Bus
Super I/O
Serial Port
Parallel Port
Floppy Disk
Keyboard
Mouse
Flash ROM (BIOS)

xv6

- Zasnovan na UNIX v6
- Implementiran na tradicionalan nacin, u formi kernela koji pruza usluge korisnickom procesima.
- Korisnickom procesu se dodeljuje blok memorije za smestanje instrukcija, podataka i magacina.
- Kada korisnicki proces zahteva uslugu kernela, poziva proceduru koja cini interfejs OS-a.
  - Sistemski poziv
  - Sistem prelazi u yasticeni/kernel mod rada, opsluzuje zahtev i vraca kontrolu korisniku
  - Kernel koristi CPU hardverske mehanizme kako bi obezbedio da svaki proces pristupa samo svom adresnom prostoru.
- Skup sistemskih poziva predstavlja interfejs izmedju korisnika i OS.

- Shell predstavlja progam koji prihvata komande korisnika i izvrsava ih.
- Predstavlja korisnicki program i nije deo kernela.
- Svakom korisniku se dodeljuje skup adresa u glavnoj memoriji za smestaj:
  - Instrukcija
  - Podataka
  - Magacina



System call	Description
fork()	Create a process
exit()	Terminate the current process
wait()	Wait for a child process to exit
kill(pid)	Terminate process pid
getpid()	Return the current process's pid
sleep(n)	Sleep for n clock ticks
exec(filename, *argv)	Load a file and execute it
sbrk(n)	Grow process's memory by n bytes
open(filename, flags)	Open a file; the flags indicate read/write
read(fd, buf, n)	Read n bytes from an open file into buf
write(fd, buf, n)	Write n bytes to an open file
close(fd)	Release open file fd
dup(fd)	Duplicate fd
pipe(p)	Create a pipe and return fd's in p
chdir(dirname)	Change the current directory
mkdir(dirname)	Create a new directory
mknod(name, major, minor)	Create a device file
fstat(fd)	Return info about an open file
link(f1, f2)	Create another name (f2) for the file f1
unlink(filename)	Remove a file

— Upravljanje procesima i memorijom

- Za kreiranje procesa koristi se sistemski poziv fork
- Nakon kreiranja novog procesa, moguce je pokrenuti uzvrsavanje novog programa primenom sistemskog poziva exec
- Iako u pocetku i proces roditelj i proces naslednik imaju istisadrzaj u memoriji i registrima CPU, oni zapravo zauzimaju razlicite adresne prostore.
- Sistemski poziv wait omogucava procesu roditelju da obustavi izvrsavanje dok se izvrsava proces naslednik.
- Shell prihvata komandu korisnika primenom sistemskog poziva getsmd, kojipotom poziva fork kako bi izvrsio komandu korisnika i wait kako bi ceka na izvrsenje korisnicke komande. Pomocu exec se ucitava kod za pozvanu komandu. Pozivom exit se zavrsava izvrsavanje komande i vraca upravljanje shell-u.
- xv6 podrzava rad samo jednog korisnika koji odgovara pravima root kod modernih sistema.

— I/O i fajl deskriptori

- Fajl deskriptor je celobrojni broj koji omogucava kernelu da identifikuje objekat u koji proces moze da upisuje podatke i iz koga moze da cita podatke.
- Otvaranjem fajla kreira se fajl deskriptor koji koristimo za komunikaciju sa njim.
- Ovaj mehanizam u UNIXu omogucava apstakciju razlika izmedju fajlova, cevi (pipe) i perifrenih uredjaja koji svi predstavljaju kao sekvenca bajtova.
- Svaki proces ima listu fajl deskriptora, tj fajl deskriptor predstavlja indeks na osnovu koga se pristupa listi objekata sa kojima komunicira proces.
- Opste je prihvacenoda fjla deskriptor 0 odgovara standardni ulaz (std input), 1 standardni izlaz (std output) i 2standardni izlaz za gresku (std error).
- Komande red i write se koriste za citanje i upis podataka.
- Komande opsen i close se koriste za dodelu i oslobadjanje fajl deskriptora.
- Komanda exec menja sadrzaj memorije procesa nalsednika ali ocuvava tabelu otvorenih fajlova.

- Komanda dup() omogucava dupliciranje fajl deskriptora sto omogucava uoptrebu istog fajla za razlicite svrhe.
- Cev (pipe) predstavlja mali kernel bafer koji je dostupan za uoptrebu procesima pomocu dva deskriptora, jedan za citanje i drugi za upis podataka.
- Cevi predstavljaju jedan od nacina komunikacije izmedju procesa (IPC – Inter-process communication)
- Komunikaciju izmedju procesa se moze izvrsiti i pomocu privremenih fajlova (temporary file)
  - Nakon upotrebe pipe oslobadja zauzetu memoriju, primenom tmp fajlova to mroa da uradi shell
  - Primenom pipe semogu preneti podaci proizvoljne velicine, dok je sa tmp kolicina podataka ogranicena velicinom tmp.
  - Pipe omogucava istovremeno citanje i upis podataka.
  - Sinhrona (blokirajuca) komunikacija pomocu pipe je efikasnija od tmp.

```
int p[2];
char *argv[2];

argv[0] = "wc";
argv[1] = 0;

pipe(p);
if(fork() == 0) {
  close(0);
  dup(p[0]);
  close(p[0]);
  close(p[1]);
  exec("/bin/wc", argv);
} else {
  close(p[0]);
  write(p[1], "hello world\n", 12);
  close(p[1]);
}
```

```
echo hello world | wc
```

```
echo hello world >/tmp/xyz; wc </tmp/xyz
```

— Fajl sistem omogucava organizaciju podataka u strukture koje zovemo fajl (datoteka) i direktorijuma koji predtavljaju posebne fajlove koji se koriste za opis strukture fajlova i podatke o fajlovima.

- Fajlovi formiraju strukturu stabla. Koren stabla se zove root i oznacava sa /
- Komanda mknod kreira fajl bez sadrazaja. Metadata fajla pokazuje da se radi o uredjaju. major i minor brojevi na jedinstveni nacin identifikuju uredjaj u krenelu.
- Komandom fstat se mogu pribaviti podaci o fajlu koji se cuvaju u strukturi stat.

```
chdir("/a");
chdir("b");
open("c", O_RDONLY);

open("/a/b/c", O_RDONLY);
```

```
mkdir("/dir");
fd = open("/dir/file", O_CREATE|O_WRONLY);
close(fd);
mknod("/console", 1, 1);
```

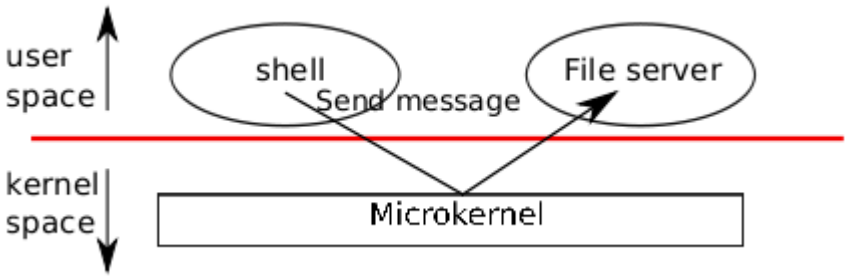
```
#define T_DIR  1   // Directory
#define T_FILE 2   // File
#define T_DEV  3   // Device

struct stat {
  short type;  // Type of file
  int dev;     // File system's disk device
  uint ino;    // Inode number
  short nlink; // Number of links to file
  uint size;   // Size of file in bytes
};
```

— Organizacija operativnog sistema

- Zadatak OSa je da omoguci konkurentno izvrsavanje razlicitih procesa.
- Treba da omoguci izvestan stepen izolacija izmedju procesa u pogledu pristupa resursima tokom izvrsvanja (CPU, fajlovi, GM, perif. uredjaji).
- Treba naci dobar odnos stepena multiprocesiranja, izolaciji i slobode komunikacije procesa.
- Da bi omogucili izolaciju, vrsi se apstrakcija resursa i njima se omogucava pristup preko posebnih usluga tj sistemskih poziva.
- Pristup resursima je transparentan korisnicima tj procesima, odnosno vise procesa ima konkurentan pristup resursima na osnovu nekih kriterijuma ravnopravnosti.
- Jaka izolacija zahteva jasnu granicu izmedju kernela i korisnickih procesa kako oni ne bi uticali na ceo sistem.
- CPU arhitektura obicno ima razlicite nivoe privilegija koje omgoucavaju razlicit pristup resursima sistema, t privilegovanim instrukcijama CPU.

— Organizacija kernela

- Pitanje lokacija pojedinih funkcija OS.
- Kod monolitskih OS sve funkcije OSa su jedna celina koja se izvrsava u zasticenom/kernel modu.
  - ☑ Efikasniji rad ali moguc slozen interfejs izmedju delova kernela.
- Mikrokernel OS, delovi funkcija OSa se nalaze u korisnickom modu rada cime se izbegava prelazak u zasticeni moda rada i komunikacija je tipa server-client.
  - ☑ SAmo se funkcije na snikom nivou (upravljanje CPU, GM, komunikacija) nalaze u kernelu.



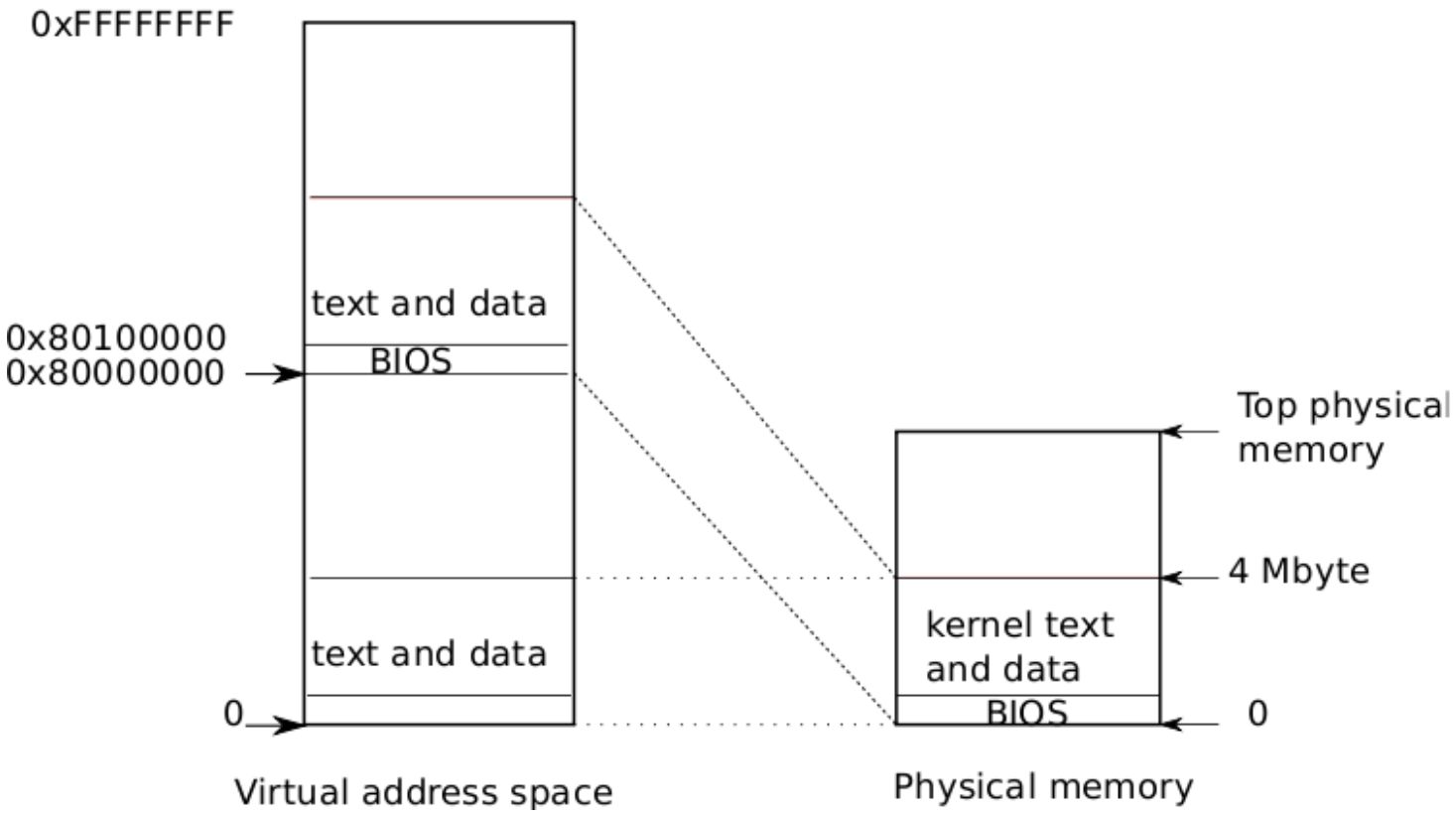
- Zahvaljujuci postojanju zasticenog/korisnickog moda, izfvojenog adresnog procesa i vremensko multipleksiranje niti.
- Stvara se iluzija da se svaki proces izvrsava na svojoj apstraktnoj masini.
  - Svakom procesu se dodeljuju tabele stranicenja, kojim se omogucava preslikavanje virtuelnog u fizicki adresni prostor.
  - Prilikom sistemskog poziva, kernel se izvrsava u adresnom prostoru procesa kako bi imao direktan pristup adresnom prostoru procesa.
  - Sve podatke o procesu cuva u strukturi **proc**.
  - Svai proces ima **nit** izvrsavanja.
  - Svaki proces ima korisnicki i kernel magacin.
  - Kernel magacin (p->kstack) se koristi samo u kernel modu.
  - Stanja procesa: p->state
  - Tabela stranicenja: p->pgdir

xv6 – Inicijalna dodela adresnog prostora

- Po pokretanju racunara, vrsi se inicijalizacija hardvera i pokretanje boot loadera.
- Hardver stranicenja nije aktiviran, takoda se virtuelne adresedirektno preslikavaju na fzicke.
- BIOS se smesta u adresni prostor: 0xa0000:0x100000
- Kernel se ucitava pocev od adrese 0x100000
  - Logicka adresa 0x80000000 (u KERNBASE) se preslikava u fizicku adresu 0x0.
- Logicki prostor KERNBASE: KERNBASE+0x400000, se preslikava na prostor: 0x400000
- Fizicka adresa entrypgdir seucitiava u kontrolni registar %cg3.
- Hardver stranicenja se aktivira postavljanjem CR0_PGu kontrolnom registru %cr0

— U %esp se smesta adresa magacina

— Konacno se skace na prvu prvu instrukciju main



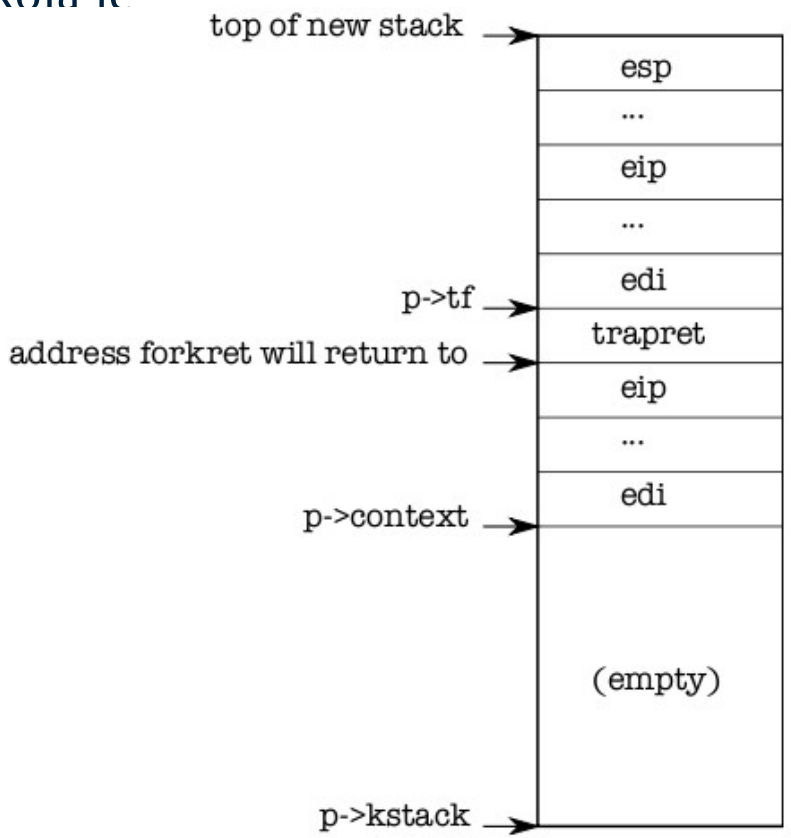
xv6 – Pokretanje prvog procesa

- Nakon sto main inicijalizuje nekoliko uredjaja i podsistema, pokrece prvi proces komandom: userinit
  - userinit se poziva samo za prvi procesa, a allocproc pri svakom pokretanju novog procesa
  - allocproc: Kreira strukturu proc u tabeli procesa i inicijalizuje proces kako bi njegova kernel nit mogla da se izvrsava
  - allocproc pretrazuje tabeli procesa dok ne nadje slot sa markerom UNUSED
  - Prosec se postavlja u stanje EMBRYO, i dodeljuje mu se pid
  - Pokusava da dodeli kernel magacin. Ako ne uspe, proces postavlja u stanje UNUSED i vraca signal za neupseh operacije kreiranja procesa

xv6 – Pokretanje prvog procesa

1010011
1110100
1100001
1010110

- Kernel nit se izvrsava kopiranjem sadrzaja magacina pocev od p->context
- Postavljanjem p->context->eip pokrece se forkret
- Ova funkcija se vraca na bilo koju adresu koja je dnu magacina
- Kod za promenu konteksta postavlja poka posle p->context
- allocproc postavlja p->context u magacin, postavlja pointer ka trapret odmah iznad n
- trapret obnavlja sadrzaj registara na osnov vrednosti sa vrha magacina skace na kod p



xv6 – Pokretanje prvog procesa

- Prvi proces izvrsava proces initcode.S
- userinit poziva setupkvm kako bi dodelio tabelu stranicenja procesa
  - U pocetku sadrzi samo stranice sa kernel magacinom
- userinit poziva inituvm koji bira jednu stranicu, postavlja logicku adresu 0 ka toj stranici i ucitava kod initcode.S u tu stranicu
- userinit postavlja trap ram
- Pokazivac magacina %esp se postavlja na najvecu vaecu logicku adresu procesa, p->sz
- Vrednost IP se postavlja na prvu instrukciju initcode, a to je 0
- za potrebe debagiranja userinit postavlja p->name to initcode
- Radni direktorijum procesa se postavlja u p->cwd
- Konacno se stanje procesa p->state postavlja u RUNNABLE

xv6 – Pokretanje prvog procesa

1010011
1110100
1100001
1010110