52. POZIV POTPROGRAMA I POVRATAK U GLAVNI PROGRAM.

za poziv potprograma, kao i povratak u glavni program, koristi se mehanizam steka. glavni problem kod pozivanja potprograma je da se zapamti adresa povratka, odnosno adresa instrukcije glavnog programa koja treba da se izvrši kada se kontrola izvršenja vrati iz potprograma u glavni program. problem se uz pomoć steka rešava lako: kod poziva potprograma adresa povratka se upiše u stek, a kod povratka se ta adresa uzima iz steka. tako se realizuju složeni pozivi potprograma, pozivi drugih potprograma iz pozvanog potprograma, pa i rekurzivni pozivi (mogućnost da potprogram poziva samog sebe). poziv potprograma se izvršava instrukcijom: call adresa potprograma koja se u memoriji smešta u dve (ili tri) memorijske lokacije. povratak iz potprograma u glavni program obavlja se naredbom

return . da bi mikroračunarski sistem pravilno radio potrebno je da se naredbe koje se odnose na rad sa stekom koriste u paru:

push – pop

i

call –return.

53. MEHANIZAM PREKIDA

omogućava da mp, pod dejstvom spoljnjeg događaja, prekine izvršavanje tekućeg progr i pređe na izvršavanje progr za obradu tog događaja. u opštem slučaju, prekid se koristi kada p izvršava dva zadatka: 1) sa nižim prioritetom koji se obavlja kontinualno, i 2) koji ima viši prioritet i obrađuje se povremeno, pod dejstvom nekog spoljnjeg događaja.zato postoje dva progr- tekući ili prekinuti progr, i drugi, tzv. prekidni progr ili progr za obradu prekida , namenjen zadatku sa višim prioritetom. spoljnji događaj izaziva nastanak

signala prekida(interrupt) koji dovodi do prekida izvršavanja progr nižeg prioriteta. kod meh prek se koriste 2 signala: ulazni int javlja da se desio događaj vezan za prekid, i izlazni inta (interrupt acknowledge – prekid prihvaćen). početna adresa prekidnog progr je, ako postoji samo jedan prekidni prog, fiksna i unapred poznata. ako postoji više jca,koristi se tzv. vektorski prekid za određivanje početne adrese prekidnog progr. povratak u tekući progr vrsi instrukcija reti. svaka pj koja učestvuje u lancu prekida poseduje jedan stepen iz tog podsistema. stepen za lančanje prekida poseduje sledeće signale: ulazni - za dozvolu generisanja signala prekida int. ieo - izlazni signal, int-izlazni signal za prekid,

isp

- interni signal prekida.. osnovna ideja u lančanju prekida je da se pj koje generišu prekid povežu po opadajućem redosledu prioriteta tako da jedinica višeg prioriteta može prekinuti prog za obradu prekida jedinice nižeg prioriteta, dok obrnuto nije moguće.

54. ZABRANA PREKIDA

u nekim slučajevima je neophodno da se spreči prelazak p-a u stanje prekida (da bi u tom trenutku sistem radio korektno). zabranu prekida omogućavaju ili kolo i r-s flip-flop sa sledećim ulazno-izlaznim signalima: int – spoljnji ulazni signal prekida,isp – interni signal prekida koji vodi ka upravljačkoj jedinici, m – izlaz iz flip.flopa koji kontroliše prolazak signala

int do upravljačke jedinice, set – ulaz flip-flopa koji prevodi izlaz m u stanje logičke 1, reset – ulaz flip-flopa koji prevodi izlaz m u stanje logičke 0, di – signal za zabranu prekida, ei – signal za dozvolu prekida. prolazak signala prekida int kroz ili kolo zavisi od logičkog stanja signala m: m = 0 , signal int prolazi kroz kolo, odnosno, ako je int na aktivnom nivou, onda je isp aktivan, što znači da je prekid dozvoljen; m = 1, signal

int ne prolazi kroz

ili kolo, odnosno

isp = 1 bez obzira na logičko stanje signala

int, što znači da je prekid zabranjen. stanje flip-flopa se kontroliše softverski, instrukcijama di i ei

koje generišu pozitivne impulse na ulazima flip-flopa koje prevode u stanje logičke 0 ili 1.

56. klasifikacija prekida

prekidi se obično dele na spoljnje, unutrašnje i softverske. spoljnji prekidi se generišu preko spoljnjeg signala prekida koji nastaje na osnovu nekog događaja koji se dogodio izvan □p-a. sličan mu je i unutrašnji prekid – kao posledica nekog događaja unutar □p-a koji ima prioritet u obradi u odnosu na tekući progr. softverski prekid se uvodi u cilju povišenja stepena zaštite u mikroračunarskom sistemu koji treba

korektno da radi i kada korisnik uradi nešto pogrešno. uvode se dva moguća generalisana stanja □p-a: korisničko i privilegovano. u prvom se ne mogu izvršiti sve instrukcije koje □p poseduje i ne može se pristupiti svim resursima mikroračunarskog sistema. u korisničkom stanju se izvršava samo podskup od akcija koje sistem može da obavi, dok akcije koje su korisniku nedostupne obavljaju rutine koje su pod kontrolom operativnog sistema. problem kod ovog pristupa je što su korisniku često potrebne akcije koje su zabranjene u korisničkom stanju

. to se rešava tako što se određena rutina operativnog

sistema aktivira signalom prekida. naime, korisnik može da radi samo u korisničkom stanju, a rutine koje su mu potrebne mogu se izvršiti samo kao odziv na signal prekida. da bi korisnik mogao da samostalno inicira izvršavanje neke rutine u privilegovanom režimu rada, mora da ima mogućnost da simulira prekid i to je osnovni razlog za uvođenje softverskog prekida- kako bi se omogućio prelazak iz korisničkog u privilegovani režim rada. softverski prekid se realizuje kao jedna posebna instrukcija koja se eksplicitno navodi na željenom mestu u programu, a odziv □p-a na ovu instrukciju je identičan odzivu na signal prekida.

57. PODRŠKA TESTIRANJU PROGRAMA

 p-i poseduju mehanizme koji obezbeđuju hardversko- softversku podršku otkrivanju i otklanjanju grešaka a to su: izvršavanje programa korak po korak, tačke prekida i trag programa. izvršavanje programa korak po korak se sastoji u tome da p izvrši jednu instrukciju programa koji se testira, a zatim se aktivira poseban program koji kroz interakciju sa korisnikom omogućava uvid u sadržaje registara p-a i memorije. korisnik može da menja sadržaje registara i memorijskih lokacija. ovaj metod je pogodan kod manjih progr. tačke prekida su pogodne za testiranje i otklanjanje grešaka u programima koji imaju petlje, a u kojima se koraci u petlji ponavljaju veći broj puta. tačka prekida obezbeđuje da p bez zaustavljanja izvrši instrukcije koje se nalaze pre tačke prekida i da se zaustavi u tački prekida, gde se aktivira progr koji omogućava korisniku iste usluge kao za mehanizam »korak po korak«. trag progr je pogodan za testiranje i otklanjanje grešaka u progr koji obezbeđuju rad mikroračunarskog sistema u realnom vremenu. trag progr je hardversko-softverski mehanizam koji omogućava da se neke informacije koje su bitne za testiranje progr smeštaju u poseban fajl u toku izvršavanja progr koji se testira. kada se završi progr, korisnik proverava sadržaj fajla i tako prati kako je program izvršavan. ovako se program nesmetano izvršava, a informacije vezane za izvršavanje progr se čuvaju za naknadnu analizu. navedeni postupci za otkrivanje greški u programu se zasnivaju na dva mehanizma:-unutrašnji prekid za trag programa (engl. trace trap) i -prekidna tačka. unutrašnji prekid se aktivira po završetku instrukcije. ovaj se prekid može softverski maskirati, odnosno dozvoliti ili zabraniti. bit t (od trap) za dozvolu unutrašnjeg prekida za trag progr obično je u sastavu indikatorskog registra. ako je unutrašnji prekid za trag programa dozvoljen, završetak instrukcije generiše unutrašnji prekid, kojim se inicira izvršenje posebnog programa. ovaj program (engl. debugger) u interaktivnom radu omogućava korisniku da na ekranu vidi sadržaje svih registara i memorijskih lokacija i da menja ove sadržaje. prekidna tačka se ne može postaviti ako se program koji se testira nalazi u memoriji tipa rom, jer ova memorija ne dozvoljava izmenu sadržaja memorijskih lokacija na način koji se primenjuje za prekidnu tačku. takođe, prekidna tačka može da se postavi samo na kod operacije neke instrukcije, a ne može da se postavi na lokaciju u kojoj je smešten operand ili neka druga informacija.

58. KONTROLER ZA PARALELNI ULAZ/IZLAZ

kod realizacije mikrorač sistema pogodno je da se koriste kontroleri- specijalizovane jedinice koje se spajaju sa p-om preko međusobno kompatibilnih signala. kontroler za paralelni ulaz/izlaz podržava razmenu memorijskih reči između p-a i neke pj-e.neposredno sprezanje

 p-a i pj-e zahteva da p proverava prenos podataka, spremnost jedinice za prijem ili predavanje podataka i sl. i zbog toga to nije pogodno. a dobro je što kontroler preuzima sve upravljačke funkcije vezane za pj-u. ovaj je kontroler , po pravilu, programabilan, što znači da u njemu postoje upravljački registri u koje p upisuje upravljačke informacije kako bi se izabrali odgovarajući radni parametri koji su specifični za određenu pj-u. upravljačke informacije se, najčešće, odnose na: smer prenosa podataka, način sinhronizacije sa pj-om, mogućnost generisanja prekida i sl. kod izlaza se podaci prenose iz ulazno/izlaznog registra prema pj, a kod ulaza se podaci prenose iz pj-e u ulazno/izlazno kontroler. sinhronizacija kontrolera sa pj-om jedinicom se vrši signalima ready i strobe koji za primer izlaza podataka (prenosa od p-a ka pj-i) imaju sledeća značenja: ready = 1,

na linijama za prenos podataka između kontrolera i pj-e se nalazi stari podatak (koji je pj već primila i obradila); ready =0 , na linijama za prenos podataka između kontrolera i pj-e nalazi novi podatak (koji pj nije obradila); strobe = 0,

pj nije spremna da prihvati novi podatak; strobe = 1,

pj je spremna da prihvati novi podatak. postupak sinhronizacije pomoću navedena dva signala (koji se naziva i rukovanje) odvija se na sledeći način: izlaz podatka preko kontrolera prema pj-i započinje operacijom upisa kojom p upisuje izlazni podatak u ulazno/izlazni registar kontrolera. dekodiranjem adresnih signala generiše se signal cs0 kojim se obavlja to upisivanje. taj podatak je novi podatak koji treba preneti pj-i.

od tog trenutka p obavlja neke druge operacije i više ne učestvuje u sinhronizaciji sa pj-om. podatak koji se do tog trenutka nalazio na linijama podataka između kontrolera i pj-e je stari (već obrađeni) podatak. izlazne linije ulazno/izlaznog registra vezane na izlazne linije za podatke kontrolera tako da će se upisani izlazni podatak odmah naći na izlaznim linijama. kada p upiše novi podatak u ulazno/ izlazni registar, upravljačka jedinica kontrolera prevede signal ready na jedinicu i tako obaveštava pj-u da je na njenim linijama za podatke novi podatak. pj detektuje ovo i prevodi signal strobe na nulu sto znaci da je započela (ali još nije završila) operaciju prijema i obrade novog podatka. završetak ove obrade pj javlja prevođenjem signala strobe na jedinicu i od tog trenutka podatak na linijama se naziva stari podatak. kontroler onda prevodi signal ready na 0 čime se završava ciklus izlaza podatka, svi signali su dovedeni na početno stanje pa treba javiti p-u da upiše sledeći podatak. to se vrsi tako sto kontroler generiše signal prekida int.

59. KONTROLER ZA DIREKTAN PRISTUP MEMORIJI

osnovna ideja mehanizma za direktan pristup memoriji (dma) je da se prenos podataka između kontrolera i mem obavlja bez posredovanja p-a. mehanizam dma zasniva se na primeni posebnog kontrolera koji omogućava neposrednu razmenu podataka između pj i memorije.

signali za spregu sa pj-om imaju značenja: dmarq

je signal kojim pj obaveštava kontroler da je spremna za prenos sledećeg podatka.ako je

dmarq = 1 onda pj ne zahteva dma prenos, a za dmarq

= 0 pj zahteva dma prenos. signal kojim kontroler javlja pocetak ciklusa je dmaa. ako je

dmaa = 1, onda nije u toku dma prenos, dok je za

dmaa = 0 u toku.

dmarw signalom kontroler obaveštava o smeru prenosa podataka.

ako je

dmarw =1 onda se vrši prenos podataka od pj-e u memoriju, a ako je

dmarw = 0 prenos je u

suprotnom smeru.

dma kontroler poseduje registre u koje p upisuje inf kao: količina podataka koje treba preneti, početnu adresu u koju se upisuju podaci (kod ulaza) ili iz koje se čitaju podaci (kod izlaza) i sl. u tipičnom korišćenju mehanizma dma prenosa izdvajaju se tri intervala vremena:1) interval pre početaka 2)interval za vreme dma prenosa. 3)interval posle dma prenosa. p preduzima akcije vezane za završetak dma prenosa koje mogu biti: operacije nad ulaznim podacima, priprema novih izlaznih podataka, obaveštavanje korisnika da je prenos završen, inicijalizacija novog dma prenosa i sl.

60. FUNKCIONISANJE RAČUNARSKIH SISTEMA

da bi se poboljšavale karakteristike rač sistema neophodno je da se rešavaju problemi kao što su: promenljiva struktura, obezbeđenje paralelnog rada pj jedne u odnosu na drugu,i u odnosu na centralni procesor,unifikacija programiranja u/i operacija, obezbeđenja reakcija računarskog sistema na različite situacije. ovi se problemi rešavaju decentralizacijom upravljanja, unifikacijom sprežnih mreža, razvojem sistema prekidanja kao i razvojem sistemskog softvera –skupa progr kojim se upravlja resursima računarskog sistema. promenljivost

strukture raču sistema podrazumeva da repertoar operacija, povezivanje jedinica računarskog sistema kao i logika upravljanja moraju biti takvi da se sistem lako prilagođava zahtevima korisnika. to može biti obezbeđeno ako postoje: standardni formati poruka i upravljačkih signala koje razmenjuju jedinice računarskog sistema, standardizovani format u/i naredbi za sve pj-e. u/i naredba za procesor predstavlja naredbu za predaju poruke i ne vodi računa o fizičkoj prirodi pj-e koje se razlikuju samo po pridruženom broju (logičkom ili fizičkom), zajedničke magistrale za sve (ili samo neke) pj-e. da bi se ostvario paralelni rad procesora i pj-e neophodno je da u/i

naredba samo inicira rad pj-e da bi procesor potom nastavio sa obradom programa. takođe je

neophodno da procesor može da inicira rad sledeće pj-e, ne sačekavši da već inicirana pj-a završi rad.

pj-a, od trenutka iniciranja njenog rada, samostalno nastavlja izvršavanje u/i operacije stupajući u vezu sa procesorom samo na kratko – ako se za to ukaže potreba.

61.MONOPROGR I MULTIPROGRAMSKI RAD RAČUNARA

procesor i kod multiprogr sistema radi strogo serijski, izvršavajući jednu po jednu instrukciju. kada se govori o istovremenom (paralelnom) izvršavanju progr to znači da posle izvršavanja dela jednog progr procesor prelazi na izvršavanje dela drugog progr itd. zadržavajući mogućnost povratka i nastavljanja obrade jednog od prethodnih progr. tako se obezbeđuje povećanje sveukupne efikasnosti procesora. efikasnost multiprogr pristupa sledi iz »istovremene« obrade progr koji imaju različite zahteve za resursima sistema. uprošćeni rad multiprogr sistema, u slučaju obrade tri progr p1, p2 i p3 (aplikacioni progr) od kojih najveći prioritet ima p1, pa p3 i onda p2 teče na sledeći način. prvo se određuje redosled izvršavanja progr (ustanovljava se red čekanja), zatim se u skladu sa redosledom datom progr dodeljuju potrebni resursi i na kraju se upravljanje prenosi na aplikacioni progr. to znači da se posle formiranja reda čekanja aktivira progr p1 koji se izvršava sve dok se ne zatraži izvršavanje u/i operacije. u pravljački progr (tzv. supervizori-deo sistemskog softvera) preuzimaju upravljanje radom računarskog sistema, stavljaju progr p1 u red čekanja za u/i operacije, a upravljanje prenose na progr p3. ako u redu čekanja nema drugih zahteva za u/i operacijama aktivira se izvršavanje u/i operacije za progr p1. tako se p3 i u/i operacije odvijaju paralelno sve dok progr p3 ne zatraži izvršavanje u/i operacije. ako se to desi, onda se progr p3 stavlja u red čekanja za u/i operacije.

62. INTERAKCIJA KORISNIKA I RAČUNARA

interakcije između korisnika i računara mogu se posmatrati kao proces opsluživanja čoveka od strane računara. forme opsluzivanja mogu biti: monoprogramski rezim (individualno opsluzivanje) i multiprogramski rezim (indirektno opsluzivanje-paketna obrada). kod individualnog opsluživanja je računar u popunosti na raspolaganju jednom korisniku. kod paketne obrade se pripremljeni programi predaju operateru na računaru, koji učitava programe u masovnu memoriju odakle se preuzimaju u operativnu memoriju i obrađuju u skladu sa predviđenim redosledom izvršavanja. kako se paketna obrada javlja i kod multiprogramskog režima rada računara to se multipogramski sistemi dele na multiprogramske sisteme sa paketnom obradom i multiprogramske sisteme sa radom u podeli vremena. u podeli vremena

više korisnika istovremeno pristupa računaru,znaci: svaki korisnik poseduje sopstvene uređaje za pristup računaru, pri istovremenom pristupu više korisnika računar se ponaša na isti način kao i pri individualnom opsluživanju. za ovakvo opsluživanje se koriste samo multiprogramski sistemi koji poseduju dodatni hardver koji omogućava interakciju korisnika i računarskog sistema. tri osnovna režima opsluživanja korisnika u podeli vremena su: režim dijaloga, transakcioni režim i univerzalni režim. računarski sistem s radom u režimu dijaloga poseduje i specijalizovanu grupu progr

namenjenih pretraživanju informaci (irs) kako bi se obezbeđivale operativne infor u skladu sa upitima korisnika. kod transakcione obrade

se vrši raspodela računarskih resursa između određenog broja udaljenih korisnika koji računarskom sistemu pristupaju na direktan način. računarski sistem sa radom u univerzalnom režimu poseduje mogućnosti rada u bilo kom od ovih režima.

63. struktura sistemskog softvera

softver rač sistema se može podeliti u dve velike grupe: sistemska programska podrška i specijalna (aplikativna) programska podrška. sist progra podrška sluzi za efikasnu realizaciju algoritama i progr, kao i upravljanje procesima izvršavanja programa na rač sistemu. njenu strukturu cine: 1)operativni sistemi, sist za razvoj progr (1.masinski (simbolicki,asemblerski i makro jezici), 2. algoritamski

i 3.problemski orijentisani jezici (dokumentalisticki jezici, jezici za modeliranje), progr za testiranje i dijagnostiku

i ostali sistemski programi (grupu cine: progr punioci (engl. loader) ,programi za povezivanje

(engl. linker) ,sistemi za upravljanje bazama podataka,editori teksta. operativni sist su namenjeni za upravljanje radom rač sistema. uvodimo pojam posla i procesa,tj. zadatka (engl. task). posao podrazumeva sve aktivnosti od unošenja progr u rač do dobijanja rezultata. kada se koraku posla dodele resursi za njegovo izvršavanje korak posla postaje proces (ili zadatak). operativni sistemi mogu biti

funkcionalni (korisnički) i ulogu upravljanja resursima (administrativni).

progr koji čine op sistem grupisu se u slojeve. prvi sloj ili jezgro, i sledeći sloj –upravljački progr ili supervizori zavise od hardvera,a dalji su nezavisni i njih korisnik vidi prilikom rešavanja problema. progr sistemi za programiranje služe da olaksaju programiranje na mašinskom jez

uvođenjem programskih jez višeg nivoa. ovi sistemi poseduju dve komponente: ulazni jezik na kome programer piše i program za prevođenje sa ulaznog jez na mašinski jez. programi za testiranje i dijagnostiku služe za proveru ispravnosti rada rač sistema.

64. REALIZACIJA U/I OPERACIJA

prilikom rada procesora neophodno je da se obezbedi zaštita memorijskog prostora . zato se operativna memorija deli na blokove određene dužine, i svakom se dodeljuje ključ koji se ne nalazi u op mem pa nije dostupan programeru. postoje posebne operacije koje mogu da izvršavaju samo posebni programi iz op sistema- to su privilegovane operacije. razmena inf između op mem i u/i jedinica vrši se preko dela koji se zove bafer. da bi se izvršila u/i operacija postoji određeni broj makronaredbi koje izvršava op sistem.to su: iniciranje u/i operacije, ispitati u/i operaciju, ispitati rad kanala, prekinuti u/i operaciju, završiti u/i operaciju. upravljačke reči kanala su: čitanje, pisanje – na medijum u/i jedinice se upišu podaci; upravljanje; skok u kanalu- da se naruši normalan redosled izvršavanja u/i operacija; stanje. adresna reč kanala određuje adresu upravljačke reči kanala koja realizuje razmenu podataka između op mem i pj-e. strukturu upravljačke reči kanala cine: kod operacije, adresa podataka, indikatori, rezervisamo polje, brojac. posle završetka razmene proveravaju se uslovi završetka u/i operacije. moze da se zavrsi normalno i nenormalno. ako se radi o nenormalnom završetku potrebno je da se na bazi indikatora stanja preduzmu odgovarajuće akcije pa da se tek onda pristupi izvršavanju naredbe završiti u/i operaciju.

65. FUNKCIONISANJE CENTRALNOG PROCESORA

centralni procesor može da funkcioniše u jednom od 4 nezavisna stanja: stanje normalne obrade korisničkih progr – s1, stanje obrade prekida – s2, stanje analize prekida – s3 i stanje prekida od hardvera – s4. programe koje je pisao korisnik procesor izvršava u stanju s1. u stanju

s2 se izvršavaju pr iz sastava operativnog sistema. po završetku obrade progr u s2 procesor prelazi u stanje

s3 izuzev u slučaju signala prekida koji potiču od hardvera kada procesor prelazi u stanje

s4. u s3 se realizuju progr koji analiziraju situaciju nastalu zbog pojave signala prekida. signali prekida se mogu biti: programski prekidi –reakcija na greške u progr ; spoljni prekidi; prekidi od u/i podsistema; hardverski prekidi – nastaju zbog kvarova u hardveru. informacije neophodne za obradu prekida su u posebnim registrima kao: registar prekida, registar stanja prekida , registar maske prekida i programski brojač. svaki prekid može da bude zabranjen ili dozvoljen u zavisnosti od stanja registra maske prekida. ako je prekid dozvoljen (nije maskiran) pamti se tekuće stanje procesora

a ako je zabranjen (maskiran) nastavlja se normalna obrada tekućeg programa. na pojavu signala prekida tekuće stanje programa pamti se na odgovarajućem mestu u operativnoj memoriji i formira se staro stanje programa (ssp). zatim se formira novo stanje programa

(nsp) koje postaje tekuće stanje. procesi se formiraju kada počinje izvršavanje posla, a uništavaju se kada se posao završi. tokom svog postojanja proces se može naći u jednom od stanja : aktivan, blokiran

i spreman (nije aktivan i nije blokiran).

66. alokacija operativne memorije

iz razloga sto operativna memorija nikada nije dovoljno velika da bi se smestile sve potrebne informacije pojavljuje se problem alokacije procesa u operativnoj memoriji. najjednostavniji način da se to resi je da se celokupni prostor operativne memorije dodeli jednom procesu dok se on izvršava, a da se po prestanku njegovog izvršavanja u operativnoj memoriji zamenjuje sledećim procesom.

navedeni pristup ima svoje nedostatke. neke od tih mana se eliminišu kada se izvrši deoba primarne memorije na relativno velike blokove nazvane particije. procesi se onda uvode u particije i iz particija, tako da se onda u operativnoj memoriji može naći istovremeno više procesa. za rešavanje problema alokacije primarne memorije koristi se i tehnika prekrivača . svi ovi postupci ne zadovoljavaju u potrebnoj meri potrebe za efikasnom alokacijom operativne memorije. zato se pristupilo primenama tehnika koje zahtevaju hardversku modifikaciju načina adresiranja primarne memorije. uvedimo neke nove pojmove : segment , predstavlja skup uzastopnih reči (ili bajta), procedure ili skup podataka. u okviru segmenta svakoj lokaciji pridružuje se simbolički naziv. dužina segmenta je proizvoljna, ali kada se segmentu dodeli memorijski prostor on je nepromenljive dužine. blok predstavlja skup uzastopnih memorijskih lokacija. segment predstavlja logičku jedinicu informacija korisnika, dok je blok fizička jedinica memorijskog prostora. postoje dve osnovne grupe algoritama za alokaciju primarne memorije: 1) algoritmi sa alokacijom segmenata u uzastopne lokacije operativne memorije. 2) algoritmi za alokaciju blokova iste dužine. segment se takođe deli na logičke jedinice iste dužine koje se zovu stranice. stranice su iste dužine kao i blokovi. zbog iste dužine svaka stranica može da se smesti u bilo koji blok. otuda blokovi koji se koriste za smeštanje segmenta ne moraju biti uzastopni, odnosno skup reči koji čini logičku celinu (segment) ne

mora se smestiti u uzastopne memorijske lokacije.

efektivna adresa (efa) reči

se određuje na taj način što se iz tabele adresa iz vrste koja odgovara stranici p uzima adresa

(označimo je sa adresa(p)) i na nju dodaje ostatak koji se dobije deljenjem relativne adrese

reči w sa dužinom bloka b (tj.

w modulo b), odnosno:

efa = adresa(p)

+

w modulo b.

67. virtuelna memorija

virtuelna memorija je pojam koji se često koristi za označavanje klase algoritama za rešavanje problema alokacije memorijskog prostora. prostor simbola čini skup simbola koji se koriste u programu, a prostor adresa obuhvata sve fizičke memorijske lokacije koje se koriste za smeštanje informacija. alokacija mem prostora se sastoji u preslikavanju iz prostora naziva na prostor adresa. preslikavanja obavljaju: progr prevodioci (kompajleri), interpreteri i asembleri, progr za povezivanje (engl. linker) i progr za punjenje primarne memorije (punjač, engl. loader). progr jez sa koga se vrši preslikavanje naziva se izvorni jezik, a jezik na koji se vrši preslikavanje je objektni (ili ciljni). interpretacija progr predstavlja izvršavanje algoritma koga progr predstavlja. progr asembler preslikava algoritam predstavljen u asemblerskom jeziku u algoritam predstavljen u internom mašinskom jeziku. u izvornom obliku nazivi unutar segmenta se nazivaju lokalni simboli,

a nazivi izvan segmenta

se zovu globalni simboli. finalnu konverziju globalnih simbola u apsolutne adrese vrši program za povezivanje (linker). funkcija progr za punjenje primarne mem je da u nju upiše sve procedure koje čine jedan program.

za brze izvrsenju, uvodi se poseban hardver. postoje dve osnovne varijante: hardver za segmentaciju i hardver za straničenje. hardver za segmentaciju koristi tabelu segmenata koja je veoma slična tabeli stranica i koja se čuva u operativnoj memoriji. svakom segmentu u tabeli odgovara jedna vrsta koja se zove deskriptor segmenta. on poseduje najmanje tri polja : jednobitno polje koje određuje prisutnost segmenta u operativnoj memoriji, polje koje definiše dužinu segmenta i adresno polje. hardver za straničenje vrši preslikavanje prostora naziva u prostor adresa koristeći tabelu stranica umesto tabele segmenata. jedina bitna razlika je što stranice koje čine progr ne moraju biti smeštene u uzastopnim memorijskim lokacijama. za rač sistem se kaže da poseduje virtuelnu memoriju kada progr u izvršnom mašinskom obliku poseduje prostor naziva koji nije identičan sa prostorom adresa. prostor naziva je mnogostruko veći od prostora adresa , što znači da progr korisnika mogu biti znatno veći od raspoloživog kapaciteta operativne memorije.

68. opšta šema funkcionisanja operativnih sistem

operativni sistemi predstavljaju sveukupnost programa namenjenih za automatizaciju izvršenja programa korisnika, kao i upravljanje računarskim sistemom u procesu njegovog rada. deo operativnog sistema (jezgro) nalazi se stalno u operativnoj memoriji- to je rezidentni deo. radi obavljanja onih funkcija koje nisu obuhvaćene jezgrom u operativnu memoriju (om) se pozivaju programi koji se zovu tranzitni i oni se smeštaju u jednu istu oblast om-e (tranzitna oblast) . rezervisani deo om-e sadrži rezidentni i tranzitni deo operativnog sistema, dok se u preostali deo om-e smeštaju korisnički programi. planiranje redosleda izvršavanja poslova može biti sekvencijalno ili planiranje sa prioritetima. posle određivanja redosleda izvršavanja poslova poslu se dodeljuju resursi. potom se inicira prvi proces koji pripada poslu i koji treba da bude izvršen – čime taj proces postaje objekat delovanja operativnog sistema. proces se može inicirati samo na dva načina: normalno iniciranje narednog koraka u izvršavanju posla i dinamičkim putem kada program podređen datom zadatku poziva drugi program za čije je izvršavanje neophodno iniciranje novog procesa. u oba slučaja se upravljačke informacije koje su potrebne za izvršavanje procesa smeštaju u red čekanja za izvršavanje datog procesa. ako proces može direktno da koristi centralni procesor za izvršavanje programa koji su mu pridruženi onda se on nalazi u stanju spremnosti. ako proces čeka završetak neke operacije (npr. u/i operacije) on se nalazi u stanju čekanja. prvo se izvršava proces sa najvećim prioritetom. ali ako se taj proces nalazi u stanju čekanja,

pristupa se izvršavanju procesa koji je u stanju spremnosti, a poseduje sledeći nivo prioriteta.

69. sistemi za razvoj programa

za svaki program napisan korišćenjem instrukcija datog procesora kažemo da je napisan na mašinskom jeziku. to su jedini programi koje procesor može direktno da izvršava. zbog tezine pisanja programa na mašinskom jeziku, koriste se viši programski jezici. za taj program kaže se da je dat na izvornom (ili ulaznom) jeziku. neophodno je takav program prevesti na mašinski jezik da bi se on mogao izvršiti, zato se razvijaju posebni programski sistemi nazvani

prevodioci. jezik na koji se prevodi sa izvornog jezika zove se objektni (ili ciljni, ili izlazni) jezik. najjednostavniji viši programski jezici su simbolički jezici. oni su mašinski orijentisani jezici i razlikuju se od jednog do drugog računarskog sistema. smisao korišćenja simboličkog jezika

sastoji se u tome da se instrukcije izražavaju korišćenjem simboličke notacije umesto reći sastavljenih od binarnog alfabeta. dalje su razvijeni makrojezici (često nazvani i autokodovi), koji se sastoje kako od instrukcija tako i od makroinstrukcija (označava datu funkciju ili proceduru i

njoj odgovara više instrukcija). pored mašinski

orijentisanih jezika kao izvorni jezici se koriste i mašinski nezavisni jezici (često se zovu i

algoritamski jezici). ovi jezici u potpunosti oslobađaju programera od potrebe poznavanja unutrašnje strukture računarskog sistema. u pogledu namene razlkuju se: jezici za rešavanje naučno-tehničkih problema (fortran, algol, pascal,...), jezici za poslovne primene (cobol,...), jezici za obradu simboličkih informacija (lisp,...), jezici za modeliranje i simulaciju (simula, simscript,...). pored algoritamskih jezika u upotrebi su i problemski orijentisani jezici koji su posvećeni rešavanju uske klase problema jednog tipa (ecap – za projektovanje elektronskih sklopova, stress – za projektovanje rešetkastih konstrukcija u građevinarstvu, i sl.).

programski sistemi koji omogućavaju programiranje na višim programskim jezicima često se nazivaju i jezički procesori. mogu biti ili hardverski (interpreter mašinskog jezika) ili softverski organizovani (prevodioci za više programske jezike).

70. prevođenje sa izvornog na objektni jezik

progr jez se može definisati kao skup pravila i oznaka za komunikac progr. neophodan je skup sintaksnih i semantičkih pravila. sintaksa definiše skup legalnih progr ovi delovi progr sastoje se iz elemenata i formiraju tzv. sintaksnu klasu. semantika određuje značenje progr pisanog u progr jez. postoje dva osnovna načina prevođenja : kompilacija i interpretacija. kompilacija je transformacija progr pisanog u jednom progr jez u ekvivalentni progr pisan u drugom progr jez. kompilacija ne menja značenje već samo oblik. interpretacija predstavlja izvršavanje algoritma koji koji program predstavlja. ona dovodi do rezultata. proces prevođenja obuhvata faze: redakcija (editovanje) izvornog progr, analiza strukture izvornog progr, uspostavljanje veza između pojedinih elemenata jez i transformacija izraza u izvornom jeziku

na niz jednostavnijih operacija.

71. mašinski orijentisani progr sist za programir

bilo koji progr napisan pomoću instrukcija kodiranih binarnim rečima predstavlja prog u maš jez. asembler je progr koji prevodi sa izvornog, asemblerskog jezika na mašinski jezik. u opštem slučaju u asemb jez postoje sledeći tipovi instr: mnemoničke instr,

pseudoinstrukcije, makroinstrukcije,

kvaziinstrukcije i uslovne instrukcije asemblera. u mnemoničkoj instr umesto mašinskih (binarnih) reči su simboličke (mnemoničke) oznake. pseudoinstrukcije služe za prenos potrebnih inf u progr asembler. one se ne prevode na mašinski jez. makroinstrukcije su sastavljene od skupa mnemoničkih instrukcija i prevode se na više instrukcija maš jez. kvaziinstrukcije generišu u tekućem progr informac potrebne za druge komponente

sistemskog softvera. uslovne instrukcije se koriste za upravljanje procesom prevođenja sa asembler na maš jez. opšti format mnemoničke instrukcije je : polje oznake, polje operac, polje operanda, polje komentara. u polje oznake unosi se oznaka

instrukcije koja omogućava pozivanje na datu instrukciju. u polje operacije

upisuje se simbolička oznaka koda operacije. u polje operanda upisuju se ili simboličke adrese ili identifikatori operanada koje asembler prevodi u mašinske adrese. u polje komentara programer

upisuje svoje primedbe koje se ne prevode na mašinski jez. pri prevođenju izvornog progr pisanog u asembleru na maš jez program asembler razmatra instrukciju jednu po jednu prema redosledu njihovog pojavljivanja.

72. progr za povezivanje i punjenje operativ mem

progr za povezivanje -redaktori ili editori za povezivanje (engl. linker editor) služi za objedinjavanje dva ili više objektnih progr u celinu. veza između takvih progr celina se ostvaruje preko poziva jedne programske celine od strane druge celine ili preko spoljnih naziva. ulaz u redaktor za povezivanje čine odvojeno prevedeni objektni moduli, dok izlaz predstavlja izvršni modul koji je spreman za uvođenje u om-u radi izvršavanja.

osnovna namena progr za punjenje je punjenje izvršnog progr i prenošenje u programski brojač početne adrese za izvršavanje programa. najjednostavniji progr za punjenje su apsolutni punjači. on je jednostavan i efikasan, ali ima mane, jer programer mora unapred da odredi adrese na kojima će program biti smešten. zato su razvijeni relativni punjači

koji omogućavaju da se izvršni programi smeštaju u trenutno slobodan prostor om-e.. često se koriste i tzv. punjači sa upravljačkim parametrima putem kojih se omogućava izmena standardnih procedura povezivanja i punjenja. pri tome se koriste i posebni komandni jezici ili se pak upravljački parametri smeštaju u izvorni program pa ih prevodilac ili asembler prenose u punjač.

74. sist za testiranje i otklanjanje grešaka u pro

jedna od osnovnih funkcija ovih sistema je da se omogući praćenje toka izvršavanja progr. to se omogućuje određivanjem tačaka u kojima se izvršavanje progr zaustavlja da bi se izvršila analiza stanja programa u cilju ustanovljenja eventualnih grešaka.

u procesu testiranja programa realizuju se funkcije: trasiranje (tracing) – putem koje se omogućava praćenje logičkog toka odvijanja programa i inverzno trasiranje (engl. traceback) – koje ukazuje na maršrutu koja je dovela do date naredbe. neophodno je da postoje efikasna sredstva (programi) za proveru ispravnosti njegovog rada

(testiranje) kao i za otkrivanje i lociranje grešaka (dijagnostika). osnovni prilaz kod izrade progr za testiranje i dijagnostiku je u tome da se hardver rač sistema posmatra kao skup elementarnih logičkih i memorijskih elemenata, gde se memorijski element posmatra kao jednobitna ćelija. ponašanje svakog elementa karakteriše se ulaznim signalima i stanjem elementa, dok je

ponašanje elementarnih logičkih kola određeno samo ulaznim signalima. neispravnosti u hardveru se mogu podeliti u dve grupe: permanentne

neispravnosti koje datu kombinacionu ili logičku mrežu prevode u mrežu koja je opisana novim logičkim funkcijama; i

slučajne neispravnosti, koje se pojavljuju s vremena na vreme. radi provere neispravnosti date mreže formiraju se dijagnostičke tablice (ili matrice neispravnosti). dijagnostičke tabele se koriste i za minimalizaciju broja progr za testiranje i dijagnostiku. u toj tabeli se za kompletan skup ulaznih signala upisuju rezultati testiranja tako da se jedan te isti element nepotrebno testira više puta. zbog toga se nameće potreba da se minimizira skup ulaznih signala i to tako da se svaki logički element testira najmanje jedanput. neke karakteristike sistema za testiranje i dijagnostiku: ako se sa

n

označi broj mogućih grešaka datog sklopa onda je

n = n1 + n2 + ... + nr

,

gde je

r

- broj elemenata u sklopu , dok

nk

,

k = 1,2,...,r

označavaju broj mogućih grešaka k-tog elementa. ako je

m

broj grešaka koje otkrivaju programi za testiranje i dijagnostiku, onda se odnos l = m / n

naziva logička gustina obuhvata

testiranja sklopa – a predstavlja (približnu) verovatnoću otkrivanja greške koju izaziva dati element.

parametar

r = s / m , gde je

s – broj fizičkih neispravnosti koje otkriju programi za dijagnostiku, a

m – broj fizičkih neispravnosti k,

predstavlja (približnu) verovatnoću da program za dijagnostiku otkrije elemenat sa fizickom neispravnoscu koji izaziva bar jednu grešku koju je otkrio odgovarajući program za testiranje.

parametar

e = s / n ,

gde je n- broj svih mogućih neispravnosti sklopa,

a

s – broj fizičkih neispravnosti koje otkriju programi za dijagnostiku predstavlja (približnu) verovatnoću otkrivanja svih mogućih neispravnosti sklopa (efektivnost otkrivanja grešaka). ako se sa

t označi vreme izvršavanja osnovnih instrukcija , a sa

t

vreme izvršavanja progr u celini, onda se odnos

k = t / t

zove koeficijent korisnog dejstva

progr za testiranje. isto tako, uvodi se i pouzdanost progr za testiranje

h =

b / n

gde je b broj neispravnosti koje ne utiču na rad pomoćnih instrukcija

u progr za testiranje.

75. prenos podataka i računarske mreže

za potrebe prenosa podataka od terminala do računara i obrnuto koriste se telekomunikacione mreže. računarskie mreže predstavljaju više računarskih sistema povezanih međusobno komunikacionim mrežama. to je omogućeno s razvojem novih h ardverskih i softverskih sredstava. problemi povezivanja računarskih sistema sa linijama za prenos podataka se javljaju zbog razlike u tehničkim parametrima i načinu rada. najvažnije razlike se javljaju u brzinama rada (računari rade mnogostruko brže od brzina prenosa po komunikacionim linijama) i kodu za predstavljanje podataka. takođe: u radu računarskih sistema i komunikacionih linija ne postoji sinhronost u radu, prisutne su slučajnosti u opterećenju linija u procesu prenosa kao i kod opterećenja računara u obradi podataka, postoje razlike u propusnoj moći pojedinih telekomunikacionih linija, zbog neizbežnih grešaka koje se javljaju pri prenosu podataka neophodno je vršiti detekciju, ali i korekciju grešaka.

76. osnovne karakteristike računarske mreže

rač mreža je skup rač i terminala koji su povezani telekomunikac podsistemom. rač mreža predstavlja skup čvorova i linija veza. povezivanje dva čvora u mreži se može ostvariti kao: povezivanje tačka-tačka- gde su dva čvora spojena sa jednim kanalom veze, paralelno povezivanje – jednu liniju veze koristi više čvorova, ali u datom trenutku liniju veze može da koristi samo jedan čvor.. izvor poruke, predajnik, linija veze (kanal sa šumovima), prijemnik i korisnik poruke čine sistem veze. tri su osnovna načina razmene poruka između prijemnog i predajnog uređaja: dupleks (fdx), kada se poruke razmenjuju simultano u oba smera, poludupleks (hdx), kada se naizmenicno razmenjuju u oba smera, i simpleks, kada se salju samo u jednom smeru. kao vodovi u sistemu prenosa koriste se: simetrični kablovi, koaksijalni kablovi, talasovodi i optovodi(optički kablovi). modem je uredjaj cijim se korišćenjem vrši spajanje rač, odnosno terminala. radi poboljšanja iskorišćenja prenosnih kanala koriste se multiplekseri i koncentratori. koriste se dva osnovna načina multipleksiranja: frekventno multipleksiranje (fdx) i vremensko multipleksiranje (tdx).

77. mreže za prenos podataka

kod namenskih mreža za prenos podataka su veze između čvorova kao i između terminalske opreme i čvorova fiksne, pa se vrši komutacija poruka ili paketa. poruka se dovodi do najbližeg čvora koji poseduje opremu koja pregleda adresu destinacije i šalje poruku (paket) do sledećeg odgovarajućeg čvora. postupak se ponavlja sve dok poruka ne stigne do odredišta. kod kom sistema sa komutacijom linija (kanala) uspostavlja se veza između izvora i odredišta na osnovu signala poziva koje šalje izvor. kada je veza između izvora i odredišta u potpunosti uspostavljena i kad je na izvoru primljen signal potvrde koji je poslat od strane odredišta tek onda može da započne razmena poruka. prenosi se poruka u celini od čvora do čvora date komunikacione strukture. kada je kompletna poruka pristigla u čvor počinje ispitivanje sledećih kanala na putu do odredišta. ako slobodnih kanala nema poruka se u čvoru stavlja u red čekanja. zato je neophodno je uvećavati kapacitete komutacionih rač. ovakav način prenosa poruka se zove i »zapamti i prosledi« ili s/f (store-and- forward) sistem. kod sistema

sa komutacijom paketa poruka se razbija na određene manje celine (pakete) i komunikac struktura vrši prenos takvih delova poruke. bitno je da razbijanje poruke na mestu njenog nastajanja mora osigurati da se na odredištu od svih prispelih delova može komponovati prvobitna celovita poruka. komutacija se vrši uz

rač koji su označeni sa imp (interface message processor).

78. detekcija i korekcija grešaka u prenosu podataka

za pouzdaniji prenos podataka koriste se 2 osnovna postupka: detekcija grešaka i detekcija i korekcija grešaka koji se baziraju na tome da se podacima koji se prenose dodaju redundantni kodovi. svakom karakteru se na kraju dodaje 1 bit da u njemu ima paran (ili neparan) broj jedinica. to je bit za kontrolu parnosti. u cilju poboljšanja efikasnosti detekcije grešaka koristi se tehnika dodavanja karaktera za proveru bloka (bcc-block check character). čes
to se za detekciju grešaka koristi i dodavanje cikličkog redundantnog karaktera

na kraju poruke. tako se poruka tretira kao niz bita, a ne kao niz karaktera. posmatrana poruka, posmatrana kao broj u binarnom sistemu brojeva, se deli sa nekim binarnim brojem. rezultat deljenja se odbacuje,a ostatak pri deljenju dodaje na kraj poruke, kao karakter za proveru bloka. na prijemnoj strani se ponovo određuje ostatak.

način korekcije grešaka, kada prijemnik otkrije grešku u pa povratnim kanalom šalje predajniku informaciju o greški ,a predajnik tada ponavlja slanje bloka u kome je detektovana greška naziva se arq (automatic repeat request) metoda. kod kontinualne arq metode nema čekanja na povratne signale od primaoca.

fec (forward error control) metoda se koristi kod satelitskih komunikacija zbog osobina kanala veze .

79. upravljanje linijama za prenos podataka

za linije za prenos, modeme i kola za povezivanje neophodno je postojanje višeg nivoa upravljanja koji se naziva upravljanje linijama za prenos podataka ili dlc (data –link control) ili linijski protokol. osnovne funkcije od dlc su: uspostavljanje veze između dva uređaja za prenos podataka, upravljanje pravilnim tokom podataka, detekcija i otklanjanje grešaka i održavanje sinhronizacije na nivou bajta (ako je to potrebno). dlc se, uopšteno govoreći, mogu klasifikovati u dve grupe:

dlc na osnovnom nivou-orijentisan na karaktere i dlc na višem nivou-orijentisan na ramove. za upravljanje na osnovnom nivou koriste se ascii (ili ebcdic) kodovi. upravljanje na višem nivou je prvenstveno bit orijentisano.
od samog početka je korišćeno više sličnih načina upravljanja kao što su: hdlc, sdlc (synchronous data link control –firma ibm) itd.osnovna jedinica koja se prenosi pri hdlc je ram (okvir). sa razvojem interneta, protokol tcp/ip preuzima primat.

80. upravljanje mrežom za prenos podataka

kom podsistem se organizuje kao niz podstruktura nazvanih slojevi i istovremeno izgrađuju protokoli različitih nivoa. osn princip njegovog rada da hijerarhijski niži sloj obezbeđuje određene usluge višem sloju. pravila koja se koriste u ovom komunic nazivaju se protokol sloja n. između dva susedna sloja nalazi se interfejs (sprežni podsistem) preko koga se obavlja interakcija. skup slojeva i protokola je arhitektura kom podsistema tj. arhitektura mreže. za interfejs je bitno da se formalno definiše koje vrste interakcije se očekuju. protokol definiše interakciju udaljenih podsistema, koja je viruelne prirode, jer se podaci u stvarnoj kom predaju od višeg sloja ka nižem sve do najnižeg sloja preko koga se obavlja fizički prenos podataka prema drugom rač sistemu. protokol predstavlja strogo definisane procedure po kojima se vrši razmena poruka između dva udaljena entiteta. osnovni elementi protokola su: sintaksa- format i značenje poruka i odziva, semantika - definiše vremenski okvir i redosled događaja. protokoli ne predstavljaju ni hardver ni softver, već virtuelno postoje samo u interakciji uparenih procesa.

81. komunikaciona oprema

predaja poruka se vrši sinhrono u blokovima uz dodavanje redundantnih karaktera za otkrivanje grešaka u prenosu i uvođenje arq metode za korekciju grešaka. to je zahtevalo da se sa strane računara i sa strane terminala ugrade dodatne jedinice nazvane komunikacioni kontroleri (ili adapteri). nametnula se ideja da se deo funkcija vezanih za prenos podataka povere mini ili mikro računaru koji će zameniti komunikacioni adapter. takav specijalizovani računar se naziva računar na prednjem kraju, fep (front end processor) ili komunikacioni računar. opšti zadaci koje oni izvršavaju su: povezivanje računara sa linijama veze, prihvatanje poruka (asinhronih, sinhronih, serijskih ili paralelnih) od terminala i drugih rač, identifikacija i adresiranje terminala, prosleđivanje poziva ili automatsko davanje odgovora – pri korišćenju javne telefonske mreže, korišćenje širokog skupa različitih kodova, komutacija poruka, obezbeđenje različitih brzina prenosa, detekcija i korekcija grešaka u prenosu, upravljanje sistemom prenosa podataka, upravljanje sistemom prioriteta poruka, usklađivanje gustine saobrađaja i mogućnosti sistema.

82. lokalne računarske mreže

lokalna rač mreža (lrm ili lan- local area network) je arhitektura sa mogućnošću brzog i direktnog komuniciranja između radnih stanica u lokalnom ambijentu. ranije im je osnovna namena bila da povežu veći br terminala sa rač sist. sada joj je namena povezivanje većeg broja personalnih rač i digitalnih upravljačkih sist. za realizaciju lan razvijene su razlicite tehnologije koje se razlikuju po prenosnim medijumima koje koriste, po topologiji mreže (sa zajedničkom magistralom, zvezdaste, prstenaste strukture, itd.),

po vrstama korisničkih usluga, kao i protokolima u komuniciranju. uređaji za povezivanje dve mreže zovu se gejtveji ili mostovi . komnikacija između dve mreže ostvaruje se kada su kom protokoli kod obe mreže isti pa se jdnostavno vrši razmena podataka preko mosta,cija je uloga da primi poruku (paket) od jedne mreže, izvrši neophodne izmene u adresnim poljima i pošalje poruku (ili paket) u drugu mrežu. osnovne topologije lan su zajednička magistrala i zatvorena petlja (prsten). izbor odgovarajuće lan zavisi od niza faktora kao što su: kapacitet, broj radnih stanica, prioritet prenosa,

oblast koju mreža pokriva, razdvajanje saobraćaja i td.