88. Neovisno poredajte gramatike LL(1), S i Q te gramatike LALR(1), SLR(1), LR(0) i LR(1) uzlazno po općenitosti.

1) S gramatika
2) Q gramatika
3) LL(1) gramatika

---

1) LR(0) gramatika
2) SLR(1) gramatika
3) LALR(1) gramatika
4) LR(1) gramatika

300. Opisati mehanizam povratne razmjene vrijednosti parametara procedura te navesti način ostvarenja.

Glavna fora je da se nakon završetka pozvane procedure kopiraju (prepišu) vrijednosti lokalnih varijabli iz vlastitog opisnika u opisnik pozivajuće procedure.

- Npr pozovemo Funkcija(a,b,c){} s parametrima (x,y,z) … ako prčkamo unutar same funkcije po x,y,z to se neće vidjeti jer će pri izlasku iz funkcije zakeljiti vrijednosti lokalnih (a,b,c) na memorijska mjesta x,y,z pozivajuće funkcije.

- Pozivajuća procedura odredi vrijednosti i adrese aktualnih parametara i zapiše ih u opisnik pozvane procedure. Pozvana procedura koristi i mijenja isključivo lokalne vrijednosti spremljene u svojem opisniku. Nakon što završi njeno izvođenje, pozvana procedura pročita iz svog opisnika vrijednosti aktualnih parametara i njihove adrese. Vrijednosti aktualnih parametara spreme se u memoriju primjenom pročitanih adresa. (ovo je formalna definicija)

301. Opisati graf zavisnosti programa i navesti sve zavisnosti koje se njime opisuju.

Graf zavisnosti programa čine dva grafa: graf zavisnosti tijeka izvođenja i graf zavisnosti podataka.
Čvorovi grafa su naredbe programa, a grane prikazuju zavisnosti upravljačkoh tijeka(iscrtkana linija) i zavisnost podataka. Zavisnost podataka može biti unaprijedna, unazadna, zavisnost odredišta i zavisnost izvorišta.

302. Objasniti kako se dobiva i boji graf zavisnosti simboličkih i stvarnih registara te kako se dodjeljuju stvarni registri

5 koraka

1. Odrede se mrežice

   - Mrežica je skup svih akcija sa nekom varijablom unutar programa, koja počinje sa njenom deklaracijom te svim akcijama u kojima je korištena

     (npr. Definicija varijable ***i*** u naredbi 1, upotreba u naredbama 2, 5 i 11 je mrežica M1)

2. Mrežicama se dodijele simbolički registri kojih ima beskonačno mnogo i mogu se iskoristiti bezbroj puta (npr. varijabla ***i*** u naredbama 1, 2, 5 i 11 zamjenjena je registrom R1)

3. Gradi se graf povezanosti registara, gdje se povezuju oni registri koji su povezani unutar mrežica. Nadalje, grafu se dodaju stvarni registri procesora, te se svi međusobno povežu. Ukoliko se neka vrijednost ne može spremiti u stvarni registar onda se u grafu povezuju simbolički registar koji prikazuje tu vrijednost i stvarni registar.

4. Graf se boji u onoliko boja koliko ima stvarnih registara, sa tim da susjedni registri ne smiju biti iste boje

5. Zamjenjuju se simbolički registri imenima stvarnih registara koji imaju jednake boje kao i simbolički.

303. Objasniti povezivanje imena izvornog programa i objekata ciljnog programa te relaciju okoline i relaciju stanja.

Povezivanje imena izvornog programa i objekata ciljnog programa se vrši pomoću relacija okoline i relacija stanja. Relacija okoline pridružuje imenu izvornog programa memorijsku adresu podatkovnog objekta, a relacija stanja pridružuje podatkovnom objektu vrijednost.

304 . Navesti i kratko objasniti postupke za određivanje djelokruga deklaracije nelokalnih imena.

a) Statičko pravilo djelokruga bez ugnježđenih procedura

   - Varijable glavnog programa spremaju se u statičku memoriju, dok se vrijednosti deklaracija procedura stavljaju u njihove opisnike. Ukoliko je neka varijabla deklarirana istim imenom unutar procedure, kao i ona unutar

glavnog programa, prva ima prednost. Kazaljke varijabli nedeklariranih u proceduri pokazuju na statičku memoriju, a upravljačke kazaljke pojedinih procedura pokazuju na opisnike onih procedura iz kojih su pozvane

b) Statičko pravilo djelokruga sa ugnježđenim procedurama

- Sve isto kao i kod prijašnjeg, samo ako je neka procedura ugnježđena unutar druge onda kazaljka nelokalnih imena pokazuje na opisnik procedure iz koje je pozvana

c) Dinamičko pravilo djelokruga

- Nema statičke memorije, ukoliko je neka varijabla a deklarirana unutar glavnog programa, zatim deklarirana istim imenom unutar neke druge iz glavnog programa pozvane procedure. Pozivom slijedeće procedure unutar glavnog programa, varijabla a će za tu funkciju biti ona poslijednja deklarirana

305. Navesti osnovne razine međukoda i objasniti namjenu svake razine.

- Međukod više, srednje i niže razine.
a) više – služi za analizu tijeka programa, toka i zavisnosti podataka
b) srednje – simbolički registri, ali još i varijable
c) niže – sliči ciljnom, može se prevesti u ciljni na više načina, nema varijabli

1. **Navesti elemente strukture generatora ciljnog programa.**

- ulaz - međukod ;
- izlaz – neki oblik ciljnog jezika;
- izrada adresa podacima – pointeri za pristup vrijednostima
- izrada adresa naredbama – oznake naredbi za jumpove i to
- izbor naredbe – koristiti MUL ili SHIFT ?

2. **Opisati postupak optimiranja petlji kod međukoda niže razine i ciljnog programa.**

- izbacivanje praznih petlji (tijelo prazne petlje nema ni jedne naredbe – mislim da je to npr. nepotrebno uzastopno grananje)

- invertiranje petlje (zamjene dviju naredbi uvjetnog i bezuvjetnog grananja jednom naredbom uvjetnog grananja na kraju petlje – programski je to kad recimo koristiš for petlju pa unutar nje ispituješ uvjet i radiš break; - a to se može zamjeniti korištenjem do while petlje

- isključivanje petlje – (ako unutar petlje imamo uvjetno grananje čiji parametri pri provjeri uvjeta nisu podložni promjeni unutar same petlje, onda to grananje možemo staviti izvan (ispred) čitave petlje)

- izravnavanje petlje – (uklanjanje naredbi grananja i ponavljanje naredbi unutar samih petlji onoliko puta koliko bi se izvodila petlja – skraćuje vrijeme izvođenja jer su naredbe grananja spore ali višestruko povećava veličinu ciljnog programa)

3. **Opisati postupak izrade adresa naredbama.**

Izrađuje ga generator ciljnoga koda. Adrese se računaju primjenom brojača, koji se povećava za veličinu generirane naredbe izražene u oktetima. Postupak koristi dvije liste, listu unaprijednih adresa, koju čine zapisi programskih oznaka i kazaljke koje pokazuju na generirane naredbe, i listu unazadnih adresa, koju čine zapisi simboličkih programskih oznaka i memorijskih adresa naredbi.

4. **Opisati algoritam gradnje lanca kazaljki nelokalnih imena i vektora dubine gniježđenja kod statičkog pravila djelokruga ugniježđenih procedura.**

Vektor dubine gniježđenja služi za izbjegavanje čitanja ulančanih kazaljki nelokalnih imena.
Ako je dubina gniježđenja procedure koja se izvodi N, onda vektor dubine ima N elementa.
Element vektora I je kazaljka koja pokazuje na opisnik posljednje aktivirane procedure dubine gniježđenja I.
Ako se želi dohvatiti deklaracija imena A dubine gniježđenja I, onda se čitanjem kazaljke u vektoru na mjestu I izravno dohvati opisnik procedure koja deklarira zadano ime A.

Algoritam gradnje lanca obrađuje 2 slučaja:
1. Pozvana procedura je deklariranu u pozivajučoj proceduri
Kazaljci nelokalnih imena pozvane procedure odredi se vrijednost tako da pokazuje na opisnik pozivajuće procedure.
Vektoru dubine gniježđenja se doda novi element i=j+1 (j=dubina gniježđenja pozvane procedure) koji pokazuje na opisnik pozvane procedure.

2. Pozvana procedura deklarirana je naredbama procedura koja ugnježđuje pozvanu i pozivajuču proceduru
Kazaljci nelokalnih imena pozvane procedure odredi se vrijednost tako da pokazuje na opisnik procedure koji se nalazi slijeđenjem j – (i – 1) kazaljki počevši od kazaljke pozivajuče procedure.
Vrijednost elementa vektora dubine na mjestu I spremi se u opisnik pozvane

procedure. Element vektora na mjestu I poprimi vrijednost kazaljke na opisnik pozvane procedure.

5. **Navesti i kratko opisati linearne oblike međukoda.**

Linearni oblici međukoda čine troadresne naredbe.
Troadresne naredbe su zapis izravnatog sažetog sintaksnog stabla ili izravnog grafa bez petlji.

Programski se ostvaruju:
1. četvorkama (operator, prvi op, drugi op, rezultat)
2. trojkama (operator i dva operanda)
3. neizravnim trojkama. (koriste vektor izvođenja)

6. **Nabrojati komponente koje čine analizu izvođenja programa.**

Analizu izvođenja programa čine:
1. Analiza tijeka izvođenja programa
2. Analiza toka podataka
3. Analiza zavisnosti podataka
4. Analiza pseudonima

7. **Objasniti generiranje ciljnog programa na temelju postfiksnog sustava oznaka.**

Pomoću potisnog automata izravnava se sintaksno stablo. Međukod je niz znakova operatora i operanada u postfikson sustavu oznaka. Operatori i operandi označeni su rednim brojevima , koji nisu dio međukoda već se za njihovo računanje koristi brojač znakova. Generator čita znakove sa lijeva na desno te radi slijedeće akcije:

-ako je pročitan **operand** akcija – *Stavi pročitani znak međukoda na vrh stoga i pomakni glavu za čitanje na slijedeći znak*

- ako je pročitan **operator** akcija - *Uzmi s vrha stoga zadani broj operanada, generiraj naredbe ciljanog programa i stavi rezultirajući operand na vrh stoga*

Svaka za svaki pročitani operator zna se točno koji broj operanada treba skinuti sa stoga.

13. Definirati atributnu prijevodnu gramatiku.

Atributna prijevodna gramatika je prijevodna gramatika za koju vrijedi:

1. Znakovima gramatike dodjeljuje se konačan skup svojstava koja se označavaju indeksima znakova gramatike, dijele se na nasljedna i izvedena i pridružuju im se vrijednosti.
2. Nasljedna svojstva: vrijednosti nasljednog svojstva desne strane produkcije računaju se na temelju vrijednosti svojstava ostalih znakova koji su na lijevoj ili desnoj strani produkcije, a početna vrijednost nasljednog svojstva početnog nezavršnog znaka gramatike zadaje se s produkcijama.
3. Izvedena svojstva: vrijednost izvedenog svojstva nezavršnog znaka lijeve strane produkcije računa se na temelju vrijednosti svojstava ostalih znakova koji su na lijevoj ili desnoj strani produkcije, a vrijednost izvedenog svojstava izlaznog završnog znaka računa se pomoću vrijednosti ostalih svojstava dodijeljenih tom istom znaku.
4.

15. Objasniti načine ostvarenja dinamičkog pravila djelokruga. (Opisati postupke pretraživanja po dubini i pretraživanja statičke memorije kod dinamičkog pravila djelokruga.)

Kod dinamičkog pravila djelokruga važeće deklaracije nelokalnih imena nasljeđuju se iz pozivajuće procedure. Dinamičko pravilo djelokruga moguće je ostvariti na dva načina: *pretraživanjem po dubini* i *pretraživanjem statičke memorije*.
Kazaljka nelokalnih imena pokazuje na isti opisnik kao i upravljačka kazaljka stoga je moguće koristiti samo jednu kazaljku.
Ako ime nije lokalno deklarirano pretražuju se opisnici procedura korištenjem lanca kazaljki dok se ne pronađe opisnik sa traženom deklaracijom imena. (pretraživanje po dubini)

Ako spremamo vrijednosti svih lokalno deklariranih imena u statičku memoriju (koju prethodno provjerimo) možemo izbjeći pretraživanje po dubini. Ako je neko zadano ime deklarirano, vrijednosti prethodnih deklaracija čuvat će se u opisniku pozvane procedure

a u statičkoj memoriji će se zamijeniti novim vrijednostima. Nakon završetka izvođenja pozvane procedure prethodno sačuvane vrijednosti deklaracija prepišu se iz opisnika u statičku memoriju. (pretraživanje po statičkoj memoriji)

Prednost pretraživanja statičke memorije je u tome što ne troši dodatno vrijeme pretraživanja, no nedostatak je u tome što zahtijeva prijepis podataka iz statičke memorije u opisnik i opisnika u statičku memoriju pa je programsko izvođenje usporeno ako imamo veliku količinu podataka

17. Definirati L-atributnu prijevodnu gramatiku.

Atributno prijevodna gramatika je L-atributna gramatika ako vrijedi:

1. Vrijednost **nasljednog svojstva** znaka **desne strane produkcije** se računa na temelju vrijednosti nasljednih svojstava *nezavršnog znaka lijeve strane produkcije* i na temelju vrijednosti svojstava znakova desne strane produkcije koji su *lijevo* od zadanog znaka. XD

2. Vrijednost **izvedenog svojstva** nezavršnog znaka lijeve strane produkcije računa se na temelju vrijednosti *nasljednih svojstava nezavršnog znaka lijeve strane produkcije* i na temelju vrijednosti svojstava znakova desne strane produkcije.

3. Vrijednost *izvedenog svojstva izlaznog završnog znaka* računa se na temelju nasljednih svojstava istog izlaznog završnog znaka.

18. Opisati opisnik procedure.

Podaci opisnika su razdijeljeni u sedam polja. U prva dva polja smještene su ***aktualne vrijednosti ulaznih i izlaznih parametara*** primjenom kojih pozivajuća i pozvana procedura razmjenjuju podatke.

Slijede ***upravljačka kazaljka*** koja pokazuje na opisnik pozivajuće procedure, te ***kazaljka nelokalnih imena*** koja pokazuje na opisnik one procedure koja se pretražuje u cilju pronalaženja deklaracije nelokalnih imena.
U peto polje se spremaju ***stanja računala i sadržaj registara*** (neposredno prije poziva aktivirane procedure), u šesto ***lokalni podaci*** koje deklariraju i korsite nadrebe aktivirane procedure, te u zadnje polje spremaju se ***privremeni podaci*** koji su međurezultati računanja izraza.

21. Opisati način izrade globalne tablice znakova.

Izrada globalne tablice znakova se može napraviti primjenom dva stoga i raspršenog adresiranja.
Jedan stog, stog znakova sadrži imena identifikatora. Drugi stog, stog lokalnih tablica sadrži kazaljke koje prikazuju na završetke lokalnih tablica znakova. Raspršeno adresiranje koristi se izravni dohvat imena identifikatora.

25. Opisati postupak relociranja za generirani premjestivi kod.

Potpune adrese podataka nije moguce izgraditi tijekom generiranja ciljnog programa jer se pocetne adrese memorijskih prostora unosa podataka i naredbi u radnu memoriju dodjeljuju u trenutku izvodenja ciljnog programa. U zaglavlje zapisa naredbi zadaje se niz kazaljki koje pokazuju na sve naredbe koje nemaju u potpunosti izgradenje adrese.

Te naredbe zadane su relativnim pomacima od pocetne adrese mjesta unosa ciljnog programa u radnu memoriju. Nakon sto program punitelj premjestivog ciljnog programa zatrazi od OS-a dodjelu prostora radne memorije i dobije ju, pocetnu adresu dodijeljenog prostora zapisa podataka dodaje pomacima u naredbama na koje pokazuju kazaljke u zaglavlju zapisa naredbi. Naredbe koje imaju u potpunosti izgradene adrese unose se u radnu memoriju racunala.

26. Opišite postupke oporavka od pogrešaka za parsiranje tehnikom od vrha prema dnu.)

Postupak je jedan (bar sam ja našao samo jedan u knjizi), a to je odbacivanje znakova ulaznog niza sve dok se ne prepozna sinkronizacijski znak, koji je u skupu :
1. SLIJEDI (<A>) – odbacuje se <A> sa vrha stoga i nastavlja dalje
2. ZAPOČINJE (<A>) – ostavlja se <A> na vrhu stoga
3. ili u skupu znakova koji definiraju sintaksne cjeline

Ako je <A> prazan onda dvije mogučnosti:
1. Pristupanje izvođenju oporavka od pogreške čim se ustanovi da pročitani znak nije u skupu SLIJEDI (<A>)
2. Nastavljanje parsiranja primjenom e-produkcije

27. Opišite način prenošenja semantičkih svojstava u tehnici rekurzivnog spusta.

U tehnici rekurzivnog spusta znakovima gramatike pridružuju se potprogrami, a svojstvima gramatike programske varijable. Ako se nekoj varijabli pridružuje nasljedno svojstvo, onda se razmjenjuje njezina vrijednost. Ako se nekoj varijabli pridružuje izvedeno svojsvo, onda se razmjenjuje njezina adresa. 

29. Opisati osnovne postupke u procesu optimiranja ciljnog koda.

Jednostavni postupak zasnovan na prozorčiću – 'prozorčić' obuhvaća nekoliko naredbi te gleda mogu li se naredbe optimirati. Prozorčić se miče od početka do kraja programa. Optimiranje naredbi grananja - predviđanje načina izvođenja grananja, na temelju toga se preuređueje struktura naredbe. Posebice je značajno kod naredbi koje se izvode istodobno, što se postiže procesorskim cjevovodima i višestrukim aritmetičko-logičkim jedinkama.

Optimiranje petlji - izuzimanje praznih petlji, invertiranje, isključivanje i izravnavanje petlji. **Invertiranje** je zamjena uvjetnog i bezuvjetnog grananja samo jednom narebom grananja na kraju petlje. **Isključivanje** petlje je postupak isključivanja naredbi uvjetnog grananja, čiji se uvjet ne može ispuniti unutar petlje, izvan petlje. **Izravnavanje petlji** je postupak u kojem se iz petlje izlučuju naredbe grananja, a ako broj petlji nije velik, slijedno se ponavljaju naredbe tijela petlje, što ubrzava rad.

Optimiranje procedura – umjesto poziva procedure u glavni se program ubacuju same naredbe procedure. Kad se unutar glavnog programa pozivaju više manjih procedura ovakvo optimiranje značajno povećava brzinu, ali povećava veličinu samog programa. Micanjem naredbi koje spremaju i osvježuju registre također se dobiva na brzini.

Optimiranje osnovnih blokova – udruživanje osnovnih blokova i povezivanje zajedničkih završetaka. Ako je prvi blok prije optimiranja u sebi imao naredbu bezuvjetnog grananja, nakon optimiranja se ta grana briše, što povećava brzinu. Sama veličina blokova se povećava, ali veći blokovi omogućavaju lakšu analizu i povećavaju učinkovitost postupka optimiranja.