227. Opišite algoritam gradnje lanca kazaljki nelokalnih imena i vektora dubine gniježđenja kod statičkog pravila djelokruga ugniježđenih procedura.

Način određivanja vrijednosti kazaljke nelokalnih imena pozvane procedure i gradnja vektora ovise o dubini gniježđenja pozivajuće i pozvane procedure. Algoritam zasebno obrađuje dva različita slučaja:

1. Pozvana procedura deklarirana je naredbama pozivajuće procedure
    - Kazaljci nelokalnih imena pozvane procedure odredi se vrijednost tako da pokazuje na opisnik pozivajuće procedure.
    - Vektoru dubine gniježđenja doda se novi element i=j+1 koji pokazuje na opisnik odabrane procedure. Ostali elementi vektora se ne mijenjaju.
2. Pozvana procedura deklarirana je naredbama procedure koja ugnježđuje pozvanu i pozivajuću proceduru
    - Kazaljci nelokalnih imena pozvane procedure odredi se vrijednost tako da pokazuje na opisnik procedure koji se dohvati slijeđenjem j-(i-1) kazaljki nelokalnih imena počev od kazaljke pozivajuće procedure.
    - Vrijednost elementa vektora dubine gniježđenja na mjestu i spremi se u opisnik pozvane procedure. Element vektora na mjestu i poprimi vrijednost kazaljke na opisnik pozvane procedure.

228. Objasnite povezivanje imena izvornog programa i objekata ciljnog programa te relaciju okoline i relaciju stanja.

Način povezivanja imena izvornog programa, podatkovnih objekata ciljnog programa i vrijednosti podatkovnih objekata određen je vrijednostima dviju relacija: relacija okoline i relacija stanja. **Relacija okoline** pridružuje imenima izvornog programa podatkovne objekte ciljnog programa. **Relacija stanja** pridružuje podatkovnom objektu vrijednost.

229. Ukratko definirajte relaciju okoline i relaciju stanja.

Relacija okoline i relacija stanja su relacije čije vrijednosti određuju način povezivanja imena izvornog programa, podatkovnih objekata ciljnog programa i vrijednosti podatkovnih objekata. **Relacija okoline** pridružuje imenima izvornog programa podatkovne objekte ciljnog programa. **Relacija stanja** pridružuje podatkovnom objektu vrijednost.

230. Objasnite načine ostvarenja dinamičkog pravila djelokruga.

Ako se primjenjuje dinamičko pravilo djelokruga, onda se važeće deklaracije nelokalnih imena nasljeđuju iz pozivajuće procedure. Dinamičko pravilo djelokruga moguće je ostvariti na dva načina: pretraživanjem po dubini i pretraživanjem statičke memorije.

1. **Pretraživanje po dubini** – Ako ime nije lokalno definirano, onda se pretražuju opisnici procedura primjenom lanca kazaljki sve dok se ne pronađe opisnik koji sadrži zadanu deklaraciju imena. Dubinu pretraživanja nije moguće odrediti tijekom prevođenja izvornog programa u ciljni program te dubina pretraživanja ovisi o načinu poziva procedura tijekom izvođenja ciljnog programa.
2. **Pretraživanje statičke memorije** – Pretraživanje po dubini moguće je izbjeći spremanjem vrijednosti svih lokalno deklariranih imena u statičku memoriju. Prije spremanja vrijednosti lokalnih deklaracija se provjerava sadržaj statičke memorije. Ako je zadano ime deklarirano, onda se vrijednosti prethodnih deklaracija sačuvaju u opisniku pozvane procedure. U statičkoj memoriji prethodne se vrijednosti deklaracija zamijene novim. Nakon završetka izvođenja pozvane procedure prethodno sačuvane vrijednosti deklaracija prepišu se iz njezinog opisnika u statičku memoriju.

231. Opišite mehanizam povratne razmjene vrijednosti parametara procedura te navedite način ostvarenja.

Ako se koristi mehanizam povratne razmjene vrijednosti, onda se u opisnik pozvane procedure zapisuje vrijednost aktualnih parametara i njihove adrese:

1. **Formalni parametri** su lokalni podatci pozvane procedure. U opisniku pozvane procedure ostavljaju se prazna mjesta za zapis vrijednosti i adresa aktualnih parametara.
2. **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 svom opisniku. Nakon izvođenja vrijednosti aktualnih parametara spreme se u memoriju primjenom pročitanih adresa.

232. Navedite i kratko objasnite postupke za određivanje djelokruga deklaracije nelokalnih imena.

Djelokrug deklaracije nelokalnih imena identifikatora određuje se na 1 od sljedećih načina: statički djelokrug bez ugniježđenih procedura, statički djelokrug ugniježđenih procedura i dinamički djelokrug. Ako se koristi **statičko pravilo djelokruga bez ugniježđenih procedura**, onda su sve procedure deklarirane isključivo naredbama glavnog programa te se djelokrug deklaracije određuje statičkom analizom tijekom prevođenja izvornog u ciljni program. **Statičko pravilo ugniježđenih** procedura zasniva se na pravilu najbliže ugniježđene procedura. **Dinamičko pravilo djelokruga** određuje se na temelju izvođenja programa. Ako naredbe pozvane procedure ne deklariraju ime identifikatora, onda su važeće deklaracije pozivajuće procedure. Postupak određivanja važeće deklaracije nastavlja pretraživanje ostalih pozivajućih procedura.

233. Objasnite način ostvarenja statičkog pravila djelokruga nelokalnih imena ugniježđenih procedura.

Statičko pravilo djelokruga ugniježđenih procedura zasniva se na pravilu najbliže ugnježđujuće procedure:

1. Djelokrug deklaracije koja je zadana naredbom procedure uključuje sve naredbe te procedure.
2. Ako ime x nije deklarirano naredbama procedure q(), onda je važeća deklaracija zadana naredbama ugnježđujuće procedure p() za koju vrijedi:
    a. Ime x deklarirano je naredbama procedure p().
    b. Ne postoji nijedna druga procedura r() za koju vrijedi: ime x je deklarirano naredbama procedure r(), procedure p() ugnježđuje proceduru r() i procedura r() ugnježđuje proceduru q().

234. Navedite osnovne načina razmjene uzlazno-izlaznih parametara procedura i što se zapisuje u opisnik pozvane procedure prilikom pojedinog načina razmjene.

Četiri su osnovna mehanizma razmjene uzlazno-izlaznih parametara:

1. **Razmjena vrijednosti**
    - U opisnik pozvane procedure se zapisuju vrijednosti aktualnih parametara
2. **Povratna razmjena vrijednosti**
    - U opisnik pozvane procedure se zapisuju vrijednosti aktualnih parametara i njihove adrese
3. **Razmjena adrese**
    - U opisnik pozvane procedure se zapisuju adrese aktualnih parametara
4. **Razmjena imena**
    - U opisnik pozvane procedure se zapisuju podatci koji se koriste za računanje adrese aktualnih parametara

235. Objasnite pojmove djelokrug deklaracije i životni vijek pridruživanja imena. Što se događa sa životnim vijekom pridruživanja imena prilikom rekurzivnih poziva potprograma?

Dio izvornog programa u kojem je važeća deklaracija naziva se djelokrug deklaracije. Životni vijek procedure započinje izvođenjem njezine početne naredbe, a završava izvođenjem njezine završne naredbe. Ako imamo rekurzivni poziv potprograma, onda se za svako aktiviranje procedure stvara zasebni opisnik te se time u svaki opisnik spremaju podatci.

236. Objasnite djelokrug deklaracije i navedite moguća pravila definiranja djelokruga deklaracija.

Ako varijabla, procedura, polja ili neki drugi identifikator nije lokalno deklariran naredbama procedure, onda pravila djelokruga određuju njegovu važeću deklaraciju. Djelokrug deklaracije nelokalnih imena identifikatora određuje se na jedan od sljedećih načina: statički djelokrug bez ugniježđenih procedura, statički djelokrug ugniježđenih procedura i dinamički djelokrug.

236. Objasnite vektor dubine gniježđenja i algoritam njegove izgradnje.

Čitanje vrijednosti ulančanih kazaljki nelokalnih imena moguće je izbjeći uporabom vektora dubine gniježđenja. Ako je dubina gniježđenja procedure koja se izvodi n, onda vektor ima n elemenata. Dubini gniježđenja i, gdje je $1 \leq i \leq n$, dodjeljuje se element vektora i. 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.

245. Nabrojite i kratko opišite linearne oblike međukoda.

Linearni oblici međukoda su troadresne naredbe jer koriste tri adrese te zato što je izvršeno izravnavanje sintaksnog stabla. Postoje tri vrste niza troadresnih naredbi:

1. **Niz troadresnih naredbi ostvaren četvorkama** – koriste 4 polja: polje operatora, dva polja operanada i polje rezultata
2. **Niz troadresnih naredbi ostvaren trojkama** – koriste 3 polja: polje operatora i dva polja operanada
3. **Neizravne trojke** – koriste se za učinkovitu promjenu redoslijeda, određuje se dodatnim vektorom izvođenja

246. Navedite osnovne razine međukoda i objasnite namjenu svake razine.

Međukod više razine – započinje sinteza ciljnog programa, služi za analizu tijeka izvođenja programa, toka podataka, zavisnosti podataka i analizu pseudonima

Međukod srednje razine – čine pojednostavljene naredbe izvornog jezika koje sliče strojnim naredbama, koristi simbolička imena identifikatora izvornog programa, imena privremenih varijabli i registara

Međukod niže razine – sliči naredbama strojnog jezika računala, generiranje ciljnog programa uz njegovo generiranje i optimiranje na najpovoljniji mogući način

247. Objasnite graf zavisnosti.

Graf zavisnosti drugi je oblik međukoda prilagođen postupcima optimiranja. Najviše se koristi tijekom optimiranja redoslijeda izvođenja naredbi i tijekom pripreme istodobnog izvođenja primjenom više procesora. Graf zavisnosti čine 2 grafa: **graf zavisnosti upravljačkog tijeka i graf zavisnosti podataka**. Graf zavisnosti upravljačkog tijela sadrži podatke o zavisnosti izvođenja naredbi te se označava crtkanim linijama koje povezuju čvorove. Postoje 4 vrste zavisnosti podataka: unaprijedna, unazadna zavisnost, zavisnost odredišta i zavisnost izvorišta.

249. Navedite tri oblika međukoda te objasnite općeniti izgled međukoda svakog oblika.

Po svom obliku međukod se dijeli na grafički, postfiksni i linearni.

Grafički oblici međukoda su sažeto sintaksno stablo i izravni graf bez petlji. Operatori i ključne riječi su isključivo unutrašnji čvorovi sažetog sintaksnog stabla te je sačuvana hijerarhija operacija (nema zagrada). Kod izravnog grafa bez petlji se provjerava je li čvor prethodno definiran.

Postfiksni sustav oznaka nastaje obilaskom čvorova sažetog sintaksnog stabla po dubini. Pojedini čvorovi upisuju se u međukod neposredno nakon obilaska svih svojih nasljednika. Potpuno izravnavanje sintaksnog stabla postiže se primjenom potisnog stoga.

Međukod linearnog oblika čine troadresne naredbe te se tako nazivaju zbog korištenja tri adrese te zato što je izvršeno poravnavanje sintaksnog stabla.

253. Opišite graf zavisnosti programa i navedite sve zavisnosti koje se njime opisuju.

Graf zavisnosti programa drugi je oblik međukoda prilagođen postupcima optimiranja. Čine ga 2 grafa: graf zavisnosti upravljačkog tijeka i graf zavisnosti podataka. Graf zavisnosti upravljačkog tijela sadrži podatke o zavisnosti izvođenja naredbi te se označava crtkanim linijama koje povezuju čvorove. Postoje 4 vrste zavisnosti podataka koje se njime opisuju: unaprijedna, unazadna zavisnost, zavisnost odredišta i zavisnost izvorišta.

257. Navedite tri oblika međukoda.

Po svom obliku međukod se dijeli na grafički, postfiksni i linearni.

263. Objasnite generiranje ciljnog programa na temelju postfiksnog sustava oznaka.

Ako je međukod zapisan primjenom postfiksnog sustava oznaka, onda se osim generiranja naredbi ciljnog programa izravnava sintaksno stablo primjenom potisnog stoga. Na temelju naredbe izvornog programa generira se niz znakova postfiksnog sustava oznaka. Međukod je niz znakova operatora i operanada zapisanih primjenom postfiksnog sustava oznaka. **Generator ciljnog programa** čita znakove slijedno slijeva nadesno i primjenjuje jednu od akcija:

1. Stavi pročitani znak međukoda na vrh stoga i pomakni glavu za čitanje na sljedeći znak
2. Uzmi s vrha stoga zadani broj operanada, generiraj naredbe ciljnog programa i stavi rezultirajući operand na vrh stoga

Ako se u međukodu pročita operand, onda generator primijeni akciju 1. Pročita li se operator, generator ciljnog programa primijeni akciju 2.

264. Opišite postupak izrade adresa naredbama.

Generator ciljnog programa izrađuje memorijske adrese naredbama ciljnog programa. Adrese naredbi računaju se primjenom brojača. Veličina brojača povećava se za veličinu generirane naredbe izražene u oktetima. Postupak izrade adrese naredbi koje upravljaju tijekom izvođenja programa koristi dvije liste: lista unazadnih adresa i lista unaprijednih adresa.

Listu unazadnih adresa čine zapisi simboličkih programskih oznaka i memorijskih adresa naredbi kojima su te oznake dodijeljene, dok listu unaprijednih adresa čine zapisi programskih oznaka i kazaljke koje pokazuju na generirane naredbe što koriste te oznake.

265. Općenito definirajte (ne na primjeru) ulaze i izlaze iz programa semantičkog analizatora i generatora ciljnog programa ako su oba programa ostvarena kao zasebni prolazi jezičnog procesora.

- **Generator ciljnog programa**: ULAZ: sintaksno stablo, IZLAZ: generirani program

- **Semantički analizator**: ULAZ: sintaksno stablo, IZLAZ: podatci potrebni za generiranje atributnog sintaksnog stabla / niza naredbi stogovnog stroja / niza slijednih naredbi

266. Opišite Chaitinov heuristički postupak za bojanje grafa zavisnosti simboličkih i stvarnih registara.

U grafu zavisnosti odredi se čvor koji ima manji broj susjeda o broja boja R. Izabrani čvor i sve njegove grane izuzmu se iz grafa zavisnosti. Na stog se spremi izuzeti čvor zajedno s listom svojih susjeda. Postupak izuzimanja čvorova nastavlja se s izmijenjenim grafom koji ima 1 čvor manje te se ponavlja izuzimanje čvorova. Ako se izuzmu svi čvorovi, moguće je obojati graf s R boja, inače ako se svi čvorovi ne uspiju izuzeti, onda u grafu zavisnosti ostaju čvorovi s R ili više susjeda. Na temelju izabrane funkcije odlučivanja izabere se 1 od simboličkih registara, a njegova vrijednost spremi se u memoriju. Promijeni se ciljni program tako da se varijabli koja je spremljena u izabranom registru pristupa izravno u memoriji uz njezino izuzimanje.

267. Opišite algoritam generiranja ciljnog programa na osnovi troadresnih naredbi.

1. Odredi mjesto M zapisa rezultata R. Prednost se daje registrima. Nije li rezultat moguće spremiti u registar, on se sprema u memoriju.
2. U opisniku podataka pronađi sva mjesta p gdje je spremljena vrijednost varijable P. Prednost se daje registrima. Ako je mjesto p različito od mjesta zapisa rezultata M izabranog u 1. koraku, onda se generira naredba: MOVE p, M.
3. U opisniku podataka pronađi sva mjesta d gdje je spremljena vrijednost varijable D. Prednost se daje registrima. Generira se naredba: OP d, M. U opisnik podataka zapiše se da je varijabla R spremljena na mjestu M. Ako je M registar, onda se u opisnik registara zapiše da registar M sadrži vrijednost varijable R. U opisnike se zapiše da ostala mjesta ne sadrže vrijednost varijable R jer je naredba R := P op D promijenila njenu vrijednost.
4. Ako se nakon izvedene naredbe R := P op D vrijednosti varijabli P i D ne koriste u nastavku programa i ako su njihove vrijednosti u registrima, onda se u zapisnike zapiše da registri više ne sadrže vrijednosti tih varijabli.

268. Navedite elemente strukture generatora ciljnog programa.

Generator ciljnog programa definira se ulazom i izlazom, a čine ga postupci izrade adrese, izbora naredbe, izbora redoslijeda izvođenja naredbi i dodjele registara podacima.

269. Objasnite kako se dobiva i boji graf zavisnosti simboličkih i stvarnih registara te kako se dodjeljuju stvarni registri.

1. Odrede se mrežice. Mrežica je unija definicija/uporaba lanaca iste varijable koji imaju zajedničkih naredbi.
2. Mrežicama se dodijele simbolički registri
3. Gradi se graf zavisnosti simboličkih i stvarnih registara.
4. Ako generator raspolaže s R stvarnih registara procesora, onda se čvorovi grafa zavisnosti boje s R različitih boja tako da 2 susjedna čvora nisu obojana istom bojom.
5. Stvarni registar dodijeli se onim simboličkih registrima koji su obojani istom bojom.

279. Navedite razradbu jezičnih procesora s obzirom na stupanj pripremljenosti ciljnog programa za izvođenje.

Ovisno o stupnju pripremljenosti ciljnog programa za izvođenje, jezični procesori dijele se na spremi-i-pokreni jezične procesore, generatore izvodivog ciljnog programa, generatore premjestivog ciljnog programa i generatore zasebnih dijelova programa.

280. Opišite što se nastoji utvrditi analizom toka podataka.

Analiza toka podataka izravni je nastavak analize tijeka izvođenja programa. Na temelju grafa tijeka izvođenja programa analizira se način uporabe podataka u programu primjenom iterativnog ili eliminacijskog postupka.

281. Objasnite razlike između strojno nezavisnog i strojno zavisnog optimiranja.

Strojno nezavisni postupci optimiraju programske strukture bliske strukturama višeg programskog jezika, dok strojno zavisni postupci optimiraju programske strukture bliske strukturama strojnog jezika.

282. Nabrojite komponente koje čine analizu izvođenja programa.

Analizu izvođenja programa čine analiza tijeka izvođenja programa, analiza toka podataka, analiza zavisnosti podataka i analiza pseudonima.

283. Opišite postupak optimiranja petlji kod međukoda niže razine i ciljnog programa.

Optimiranje naredbi petlji obuhvaća više različitih preinaka: izuzimanje praznih petlji te invertiranje, isključivanje i izravnavanje petlji. Tijelo prazne petlje nema ni jedne naredbe. Ako se ne koristi popratni rezultat računanja uvjeta prazne petlje, moguće ju je izuzeti iz programa. Invertiranje petlje jest postupak zamjene naredbe uvjetnog i bezuvjetnog grananja samo 1 naredbom grananja na kraju petlje. Isključivanje petlje je postupak izlučivanja te naredbe uvjetnog grananja izvan petlje. Petlja se izravnava izuzimanjem naredbi grananja i ponavljanjem naredbi njezinog tijela onoliko puta koliko se izvodi petlja.

284. Opišite analizu tijeka izvođenja programa.

Analiza tijela izvođenja programa ispituje upravljačku strukturu međukoda. Dva su osnovna pristupa: analiza dominacije i analiza strukture. Obje analize koriste graf tijeka izvođenja programa. **Analiza dominacije** određuje strukturu petlji i ona prethodi analizi toka podataka zasnovanoj na iterativnim postupcima. **Analiza strukture** određuje više različitih podgrafova u tijeku izvođenja programa te prethodi analizi toka podataka zasnovanoj na eliminacijskim postupcima.

285. Nabrojite i kratko opišite postupke optimiranja međukoda srednje razine.

Optimiranje procedura – Postupcima optimiranja nastoji se smanjiti utjecaj izvođenja naredbi poziva i povratka iz procedure te utjecaj naredbi koje spremaju i osvježavaju stanja izvođenja programa.

Optimiranje petlji – Postupcima preuređenja petlji optimira se provjera granica indeksa polja i uporaba spregnutih varijabli.

Optimiranje izraza – Širok je spektar postupaka optimiranja izraza koji skraćuju vrijeme izvođenja ciljnog programa i njegovu veličinu, a neki od njih su: unaprijedno računanje izraza konstantne vrijednosti, izlučivanje izraza izvan petlje, itd.

Prenošenje jediničnih naredbi – Naredba oblika Lijevo := Desno naziva se jediničnom gdje je Lijevo ime varijable, a Desno vrijednost konstante ili ime isključivo 1 varijable.