5. POTPORA IZVOĐENJU CILJNOG PROGRAMA

Ciljni program čine naredbe generirane prevođenjem izvornog programa u ciljnji i naredbe potpore izvođenju ciljnog programa.

Naredbe poprore izvoženju ciljnog progama:

* upravljaju dodijelom memorije
* ostvaruju apstraktne tipove podataka izvornog jezika podatkovnim objektima ciljnog jezika
* povezuju imena izvornog programa i podatkovne objekte ciljnog programa. Pomoću relacija okoline (pridružuje imenima izvornog programa podatkovne objekte ciljnog programa tj. dodjeljuje im adresu) i relacija stanja (pridružuje podatkovnom objektu vrijednost).
* upravljaju opisnicima procedura
  + Primjer opisnika aktivirane procedure:

|  |
| --- |
| 1. Aktualne vrijednosti izlaznih parametara procedure |
| 1. Aktualne vrijednosti ulaznih parametara procedure |
| 1. Upravljačka kazaljka (pokazuje na opisnik pozivajuće procedure) |
| 1. Kazaljka nelokalnih imena |
| 1. Stanje računala i sadržaj registara |
| 1. Lokalni podaci |
| 1. Privremeni podaci |

* + Podaci se u opisnik upisuju na način:
    - Pozivajuća procedura upiše u opisnik pozvane procedure aktualne vrijednosti ulaznih parametara (npr. Main zapise u opisnik pozvane funkcije ulazne parametre)
    - Pozivajuća procedura zapiše u svoj opisnik adresu naredbe nastavka izvođenja pozivajuće procedure, a vrijednost kazaljke stoga u opisnik pozvane procedure (npr. Main zapiše u svoj opisnik gdje treba nastaviti kad ova svrši, a kazaljku iduće naredbe promjeni na početak pozvane funkcije)
    - Pozvana procedura zapiše u svoj opisnik vrijednosti registara i ostale podatke koji se koriste poslije u pozivajućoj proceduri (npr. Iz arha, pozvani potprogram spremi na stog stanja registara da ih može koristit u svom izvođenju bez gubljenja podataka)
    - Pozvana procedura zapiše početne vrijednosti lokalnih podataka. Nakon toga kreće izvođenje
  + Nakon završetka izvođenja pozvane procedure događa se:
    - Pozvana procedura zapiše aktualne vrijednosti izlaznih parametara u svoj opisnik (no šit šerlok)
    - Pozvana procedura obnavlja prijašnje vrijednosti registara procesora i ostale vrijednosti stanja računala na temelju vriednosti iz svog opisnika. (npr. Iz arha, potprogram obnovi stanja registara da budu kao prije nego je pozvan)
    - Pozivajuća procedura čita aktualne vrijednosti izlaznih parametara pozvane procedure i zapiše ih u svoj opisnik
* upravljaju dodjelom registara
* omogućuju dinamička svojstva izvornog jezika.

Tijek izvođenja izvornog programa zasnovanog na procedurama:

* Opisuje se stablom aktiviranja.
  + Korijen stabla je aktivirani glavni program, čvorovi su aktivirane procedure.
  + Ako procedura A() aktivira proceduru B() onda je A() roditelj od B()
  + Ako životni vijek (vrijeme dok je procedura aktivna) procedure A() završi prije nego što započne životni vijek procedure B() onda je čvor A() lijevo od čvora B()

Pristup nelokalnim imenima

Ako neki identifikator nije lokalno deklariran u proceduri onda pravila djelokruga određuju njegovu važeću deklaraciju. Pravila djelokruga su:

* Statičko pravilo djelokruga bez ugnježđenih procedura (primjer na slici 5.13. str 235.)
  + Sve procedure su deklarirane isključivo naredbama glavnog programa
  + Na stog se prvo stavlja Statička memorija (slično kao opisnik neki) i u nju globalni podaci definirani u glavnom programu.
  + Kazaljke nelokanih imena svih procedura pokazuju na statičku memoriju
* Statičko pravilo djelokruga ugnježđenih procedura (primjer na slici 5.14. str. 236.)
  + Procedura je deklarirana unutar druge procedure
  + Sprema vrijednosti deklaracije unutar procedura
  + Ako naredbe procedure ne deklariraju ime identifikatora onda je važeća deklaracija zadana naredbama najbliže ugnježđujuće procedure (najbliža u levelu iznad. Taj level Srbljić naziva dubina gnježđenja, main ima dubinu 1, ostalo je logično ☺) i na tu proceduru će pokazivat kazaljka nelokalnih imena
* Dinamički djelokrug
  + Važeće deklaracije nelokalnih imena nasljeđuju se iz pozivajuće procedure tj kazaljka nelokalnih imena pokazuje na isti opisnik kao i upravljačka kazaljka (opisnik pozivajuće procedure)
  + Mogući drugi način izrade: koristiti samo jedan pokazivač, a sva lokalno deklarirana imena spremati u statičku memoriju (prije spremanja provjeriti ako već postoji onda raditi kao s registrima tj spremi kontekst pa odradi sve i na kraju obnovi kontekst)

Razmjena ulazno/izlaznih parametara procedura

* Razmjena vrijednosti (u opisnik pozvane procedure zapišu se vrijednosti aktualnih parametara). Algoritam:
  + U opisniku pozvane procedure se ostavljaju prazna mjesta za zapis vrijednosti formalnih parametara (lokalni podaci pozvane procedure)
  + Pozivajuća procedura odredi vrijednosti aktualnih parametara i zapiše ih u opisnik pozvane prcoedure. Pozvana procedura koristi i mijenja isključivo lokalne vrijednosti parametara zapisane u svom opisniku
  + Klasični način kao u C-u
* Povratna razmjena vrijednosti (u opisnik pozvane procedure zapisuju se vrijednosti aktualnih parametara i njihove adrese). Algoritam:
  + U opisniku pozvane procedure se ostavljaju prazna mjesta za zapis vrijednosti formalnih parametara (lokalni podaci pozvane procedure) i adresa aktualnih parametara
  + Pozivajuća procedura odredi vrijednosti i adrese aktualnih parametara i zapiše ih u opisnik pozvane procedure. Pozvana procedura mijenja i koristi isključivo lokalne parametre ALI prilikom zavrsetka procedure na adrese ulaznih parametara zapise njihove vrijednosti iz SVOG opisnika aktualnih parametara (pomocu adresa koje je dobila na ulazu)
* Razmjena adrese (u opisnik pozvane procedure zapišu se adrese aktualnih parametara) . Algoritam:
  + U opisniku pozvane procedure ostavljaju se prazna mjesta za zapis adresa aktualnih parametara
  + Pozivajuća procedura izračuna adresu aktualnog parametra i zapiše je u opisnik pozvane procedure. Ako je aktualni parametar konstanta onda se njezina vrijednost spremi u memoriju, a adresa memorije u opisnik pozvane procedure. Ako je aktualni parametar izraz onda se taj izraz izračuna i spremi u memoriju a adresa memorije u opisnik pozvane procedure
  + Isko kao u C-u s pointerima. Npr ako u pozvanoj proceduri piše x (x je ulazni parametar) onda zamisliš da piše \*x (to je definirano prilikom poziva procedure)
* Razmjena imena (u opisnik pozvane procedure zapisuju se podaci koji se koriste za računanje adrese aktualnih parametara) . Algoritam:
  + U opisniku pozvane procedure ostavljaju se prazna mjesta za zapis podataka za računanje adresa aktualnih parametara
  + Pozivajuća procedura zapiše u opisnik pozvane procedure podatke koji se koriste za izračun adresa aktualnih parametara. Pozvana procedura nanovo računa adresu za svaki dohvat vrijednosti aktualnog parametra

6. GENERIRANJE MEĐUKODA

Međukod je prijelazni oblik izvornog programa koji se koristi tijekom sinteze ciljnog programa. Omogućuje učinkovito optimiranje te odvojeno i nezavisno programsko ostvarenje faza analize i sinteze.

Strukturu međukoda čine njegova:

* razina - omjer zastupljenosti struktura izvornog i ciljnog jezika. Vrste razina:
  + međukod više razina – početak sinteze ciljnog programa. Ostaju sačuvane strukture petlji i indesa polja izvornog programa. Prevodi se u međukod srednje ili niže razine. Najčešće je grafičkog oblika.
  + međukod srednje razine – pojednostavljene naredbe izvornog jezika (sliče strojnim naredbama). Koristi simbolička imena identifikatora izvornog programa.
  + međukod niže razine – sliči naredbama strojnog jezika. Većina njegovih naredbi se prevodi u jednu naredbu strojnog jezika. Ne koriste se simbolička imena identifikatora već registre (simboličke).
* Oblik. Vrste oblika međukoda:
  + Grafički
    - Sažeto sintaksno stablo – listovi stabla su operandi (npr. x), a unutrašnji čvorovi su operatori i ključne riječi. Nema zagrada jer je sačuvana hijerarhija operacija.
    - Izravni graf bez petlji – dva istovjetna izraza prikazuju se jednim podstablom
  + Postfiksni – nastaje obilaskom čvora sažetog sintaksnog stabla po dubini i zapisuje se primjenom postfiksnog sustava oznaka
  + Linearni – Troadresne naredbe ( npr. X := Y op Z). Y i Z su adrese operanada, a X rezultata, op je oznaka operacije.
    - Sve troadresne operacije: bezuvjetno grananje (goto L), uvjetno grananje (if X relop Y goto L), pridruživanje rezultata binarne operacije (X := Y op Z), pridruživanje rezultata unarne operacije (X :=op Y), preslikavanje (X := Y), poziv potprograma (call P,n ;n je broj ulaznih parametara), povratak iz potprograma ( return Y), definiranje parametara potprograma (param X), indeksirani dohvat podataka ( X := Y[i] i X[i]:=Y), pridruživanje adrese(X := &Y), dohvat podataka adresom (X:=\*Y i \*X:=Y)
    - Kak se one programski ostvaruju pogledaj u knjizi na 258. i 259. stranici, ne da mi se to pisat jer mislim da je nepotrebno ☺
  + Posebni oblici međukoda – prilagoženi su optimiranju. Vrste:
    - Statičko jednostruko pridruživanje – varijabla moraju biti najviše jednom odredišne u naredbama pridruživanja (ne-Srbljićevim rječnikom: samo jednom smije s lijeve strane jednakosti stajati x, ako ima npr 2 puta onda se mijenja x sa x1 i x2 ☺). Sranje je ako se x koristi negdje poslije s desne strane jednakosti, onda se mora dodat x3 i funkcija Φ koja kao ulaz ima x1 i x2. Točnije dodaje se cijela naredba (prije naredbe gdje je x s desne stane) koja izgleda npr. ovako: x3 = Φ (x1, x2) i u mjestu gdje se je koristio x s desne strane staviš x3. Gradi se iz grafa tijeka izvođenja programa.
    - Graf zavisnosti – čvorovi grafa su naredbe programa. Sastoji se od:
      * grafa zavisnosti upravljačkog tijeka – sadrži podatke i zavisnosti izvođenja naredbi. Grane mu se označavaju crtkanim linijama i oznakama „c“.
      * grafa zavisnosti podataka – Grane mu se označavaju punim linijama i oznakama zavisnosti. Postoje 4 vrste zavisnosti podataka:
        + unaprijed zavisnost („f “ na grafu – prva mijenja a druga koristi istu varijablu)
        + unazadna zavisnost („a“ na grafu – prva koristi, a druga mijenja istu varijablu)
        + zavisnost odredišta („o“ na grafu – obje naredbe mijenjaju istu varijablu)
        + zavisnost izvorišta („i“ na grafu – obje naredbe koriste istu varijablu)

Sintaksom upravljano generiranje međukoda – pomoću prijevodne gramatike. Iz prijevodne gramatike je moguće generirati:

* sažeto sintaksno stablo – uporaba mreže, čvorovi sadrže 2 pokazivača (na listove mreže ili druge čvorove) i vrijednost (operaciju koju vrše). Listovi sadrže samo pokazivač na tablicu znakova tj na znak identifikator koji se treba koristiti
* troadresne naredbe – to je trivijalno, samo treba znati prijevodnu gramatiku i troadresne naredbe (objašnjeno iznad)

7. GENERIRANJE CILJNOG PROGRAMA