LABORATOR NR. 9

GENERAREA ŞI OPTIMIZAREA CODULUI OBIECT

Ultima fază a compilării are ca scop sinteza programului obiect, program executabil sau aproape executabil. Pentru generarea codului obiect sunt necesare cunoştinţe aprofundate despre maşina cu care se lucrează. Forma luată de programul obiect este, de obicei, una din următoarele:\

-program executabil
-program obiect
-program în limbaj de asamblare
-program într-un alt limbaj

Generarea codului pentru calculatoare cu un singur registru acumulator.
Toate calculele se efectuează într-un singur registru, numit adesea acumulator. Printre instrucţiunile existente în limbajul de asamblare sunt următoarele:

LOD X - încarcă valoarea X în acumulator
STO X - memorează conținutul acumulatorului în cuvântul de memorie notat cu X
ADD X – adună valoarea lui X la valoarea acumulatorului
SUB X – scade valoarea lui X din valoarea acumulatorului
MUL X- se multiplică valoarea X cu valoarea acumulatorului
DIV X – se împarte valoarea acumulatorului la valoarea variabilei X

Algoritmul asamblare cod
Intrare: o expresie aritmetică în forma poloneză, având n simboluri depuse în vectorul Pol
Ieşire : limbajul de asamblare corespunzător acestei expresii, pentru un calculator cu un registru acumulator
Metoda: algoritmul utilizează stiva Stiva cu indexul ind. Curent conține caracterul cutent de la intrare, variabila I este asociată cu o locație temporară de memorie. Variabila opcod conține codul operației curente ce trebuie executată. Subalgoritmii PUSH şi POP sunt folosiți pentru a depune, respectiv a extrage un element din stivă.

$i \leftarrow 0$
$ind \leftarrow 0$
$for\ j = 1,2,...,n\ do$
Begin
$Curent \leftarrow Pol(j)$
$if\ 'A' \le Curent \le 'Z'$

```
Then   call   PUSH(Stiva,ind,Curent)
Else   if   Curent ='←'
     Then   if  ind ≠ 2
        Then
           Begin
             Write (‘instrucţiune invalidă’)
            Exit
           End
        Else
        Begin
          Write(‘LOD’,POP(Stiva, ind))
         Write(‘STO’,POP(Stiva, ind))
        End
Else
Begin
 Case Curent of
‘+’:opcod←’ADD’
‘-‘:opcod←’SUB’
‘*’:opcod←’MUL’
‘+’:opcod←’DIV’
If  ind<2
        Then
        Begin
          Write(‘operaţie invalidă’)
        Exit
        End
        Else
        Begin
          Drept← POP(Stiva, ind)
          Stâng← POP(Stiva, ind)
          Write(‘LOD’, Stâng)
         Write(opcod, Drept)
        i←i+1
        Temp←’T’°i
        Write(‘Sto’, Temp)
        Call(Push(Stiva,ind,Temp)
        End
End
```

If ind≠0
Then
Begin
Write('instrucțiune invalidă')
Exit
end

Algoritmul îmbunătățire asamblare cod

Intrare: o expresie în forma poloneză având n simboluri depuse în vectorul Pol

Ieşire: limbajul de asamblare corespunzător unei instrucțiuni de atribuire

Metoda: Se utilizează aceleaşi variabile ca şi în algoritmul Asamblare cod. Comut este o variabilă nouă care ia valoarea True dacă operația curentă este comutativă. Simbolul special # este plasat în stivă în loc de a genera o instrucțiune de memorare.

$i \leftarrow 0$
$ind \leftarrow 0$
for $j = 1,2,...,n$ *do*
Begin
$Curent \leftarrow Pol(j)$
if $'A' \le Curent \le 'Z'$
Then
Begin
call $PUSH(Stiva, ind, Curent)$
if $ind > 2$
Then *if* $Stiva[ind-2] = '\#'$
Then
Begin
$i \leftarrow i+1$
$Temp \leftarrow 'T' \circ i$
$Stiva[ind-2] \leftarrow Temp$
$write('STO', Temp)$
End
End
Else
Begin
If Curent='←'
Then
If ind≠2

```
                Then if ind=1
                 Then
                 Begin
                  ind←ind-1
                 write('STO',POP(Stiva,ind))
                end
                else
                begin
                    Write ('instrucțiune invalidă')
                    Exit
                  End
                Else
                Begin
                  Rezultat←POP(Stiva, ind)
                 If Rezultat≠'#'
                        Then
                        Begin
                        Write('LOD',Rezultat)
                        Write('STO',POP(Stiva,ind))
                       End
                End
        Else
        If Curent='+' sau Curent='*'
                Then
                Begin
                Comut←True
                If Curent='+'
                  Then opcod←'ADD'
                  Else opcod←'MUL'
                End
                Else if Curent='-' sau Curent='/'
                  Then
                    Begin
                         Comut←False
                        If Curent='-'
                           Then  opcod←'SUB'
                           Else  opcod←'DIV'
                   End

If ind<2
```

```
                    Then
                     Begin
                          Write('instrucțiune de atribuire invalidă')
                         Exit
                     End
                     Else
                        Begin
                          Drept← POP(Stiva, ind)
                          Stâng← POP(Stiva, ind)
                         If Drept='#'
                         Then if Comut
                           Then write(opcod,Stâng)
                          Else
                         Begin
                          i←i+1
                          Temp←'T'°i
                          Write('Sto', Temp)
                          Write('LOD', Stâng)
                          Write(opcod,Temp)
                         i←i-1
                         end
                    Else if Stâng≠'#'
                     Then
                      Begin
                          write('LOD', Stâng)
                          write(opcod,Drept)
                      End
                    Else write('ADD',Drept)
                    If Drept=variabilă temporară
                          Then
                           Begin
                             i←i-1
                             Call(Push(Stiva,ind,'#')
                          End
                    End
               End
  End
If ind≠0
       Then
        Begin
```

Write('instrucţiune invalidă')
Exit
end

Generarea codului pentru calculatoare cu regiştri generali.
Algoritmul GENCOD1

Intare: O instrucţiune cu trei adrese de tipul $A \leftarrow B \quad op \quad C$
Ieşire: Codul obiect corespunzător acestei instrucţiuni
Metoda:

P1. Cheamă funcţia *GESTREG* pentru a determina locaţia *L* unde se va executa operaţia *B op C*. În mod obişnuit L este un registru, dar poate fi şi o locaţie de memorie.
P2. Consultă mulţimea Loc(B) pentru a determina locaţia curentă B' a lui B. Preferăm un registru pentru B' dacă valoarea lui B se află atât în memorie cât şi într-un registru. Dacă valoarea lui B nu se află deja în L, generează instrucţiunea:

MOV B',L

P3. Generează instrucţiunea op C',L unde C' este locaţia curentă a lui C. Din nou preferăm un registru dacă valoarea lui C se află şi în memorie şi într-unul sau mai mulţi regiştri.

Utilizarea GOFC (grafuri orientate fără cicluri) în generarea codului obiect

Avantajul utilizării grafurilor orientate fără cicluri constă în posibilitatea alegerii unei secvenţe de instrucţiuni cu trei adrese într-o ordine mai convenabilă pentru obţinerea unui cod eficient: program mai scurt şi variabile temporare mai puţine. Pentru obţinerea unui cod optimizat se folosesc algoritmii de listare a nodurilor interioare şi de etichetare a nodurilor.

Algoritmul de listare a nodurilor(LISTNOD)

Intrare: GOFC
Ieşire: secvenţa optimizată a instrucţiunilor în ordine inversă
Metoda:

while mai există noduri interioare nelistate do
begin
selectează un nod interior nelistat *n*, ai cărui părinţi au fost listaţi
listează *n*
while cel mai din stânga descendent m al lui n are toţi părinţii

```
    listați şi m nu este nod frunză do
    begin
        listează m
        n←m
    end
end
```

Algoritmul de etichetare a nodurilor

Intrare: nodul *n* împreună cu descendenții săi etichetați

Ieşire: eticheta lui *n*

Metoda:

If n este nod terminal

Then if *n* este descendentul cel mai din stânga al părintelui său

Then

Eticheta(*n*)←1

Else

Eticheta(*n*)←0

Else

Begin

Fie $n_1, n_2, ..., n_k$ descendenții lui *n* în ordinea dată de etichete: $et(n_1) \geq et(n_2) \geq ... \geq et(n_k)$

$et(n) \leftarrow \max_{1 \leq i \leq n}(et(n_i) + i - 1)$

End

În cazul când *n* este un nod binar cu descendenții având etichetele n_1 , n_2, formula de etichetare a sa devine

$$et(n) = \begin{cases} \max(n_1, n_2) & pentru \quad n_1 \neq n_2 \\ n_1 + 1 & pentru \quad n_1 = n_2 \end{cases}$$

Eticheta rădăcinii reprezintă numărul minim de regiştri necesari generării codului obiect. După etichetare se generează codul ulilizând algoritmul GENCOD2.

Algoritmul GENCOD 2

Intrare: un arbore de derivare etichetat

Ieşire: codul obiect asociat arborelui

Metoda: Algoritmul este o procedură recursivă care aplicată unui nod generează codul arborelui dominat de acel nod

Se utilizează o stivă care reține toți regiştri disponibili la un moment dat, din cei r ai maşinii; inițial stiva conține toți cei r regiştri. Pentru a lucra cu o

Stivă s se folosesc: procedura PUSH(s,x) care depune în stivă valoarea x, funcția POP(s) care descarcă stiva, funcția VARF(s) care întoarce valoarea din vârful stivei, procedura PERM(s) care permută două valori din vârful stivei, o stivă Temp care menține lista variabilelor temporare T0,T1,… pe care le poate folosi în continuare. Când cei *r* regiştri sunt cupați, algoritmul ia o celulă temporară din vârful stivei.

Procedure GENCOD2(*n*)

```
Begin
    If n este un nod frunză reprezentând operandul nume şi este cel mai
       din stânga descendent al tatălui său
    Then
          Write('MOV', nume, VARF(Stiva))
    Else if n este un nod interior cu operatorul op, descendentul stâng n1 şi
    cel drept n2
    then
     if et(n2)=0
        then
           begin
              fie nume operandul reprezentat de n2
              call GENCOD(n1)
              write(op,nume,VARF(Stiva))
           end
           else if 1 ≤ et(n1) < et(n2)  şi  et(n1) < r
                   then
                    begin
                      call PERM(Stiva)
                      call GENCOD2(n2)
                      reg←POP(Stiva)
                      call GENCOD2(n1)
                      write(op,reg,VARF(Stiva))
                      call PUSH(Stiva,reg)
                      call PERM(Stiva)
                   end
           else
             if 1 ≤ et(n2) < et(n1)  şi  et(n2) < r
                   then
                     Begin
                          Call GENCOD2(n1)
```

```
                    reg←POP(Stiva)
                    Call GENCOD2(n2)
                    Write(op,VARF(Stiva),reg)
                    Call PUSH(STIVA,reg)
                End
            Else
                Begin
                    Call GENCOD2(n2)
                    t←POP(Temp)
                    write('Mov',VARF(Stiva),t)
                    Call GENCOD2(n1)
                    Call PUSH(Temp,t)
                    Write(op,t,VARF(Stiva))
                End
End
```

Teme propuse

1. Implementați algoritmul de generare a codului obiect pentru calculatoare cu un singur registru acumulator.
2. Implementați algoritmul îmbunătățit de generare a codului obiect pentru calculatoare cu un singur registru acumulator
3. Implementați algoritmul de listare a nodurilor pentru un GOFC dat.
4. Implementați algoritmul de etichetare a nodurilor pentru un GOFC dat.
5. Implementați algoritmul GENCOD2 pentru generarea codului obiect asociat unui arbore de derivare etichetat.