Compiladors (CL)
GEI (2021-22)

Pràctica 3: Compilador de MiniJulia

Objectiu

Fer les parts frontal i dorsal d'un compilador per al llenguatge MiniJulia, ampliació del de la Pràctica 2.

Aprenentatges

Generació de codi intermedi de tres adreces mitjançant back-patching.

Introducció

A la Pràctica 2 s'ha creat un compilador pel llenguatge MiniJulia capaç de generar codi de tres adreces (C3A) per a expressions aritmètiques, procediments i funcions. En aquesta Pràctica 3 s'afegeixen les expressions booleanes i les estructures de flux de control. Per fer-ho, cal utilitzar la tècnica del back-patching, que permet generar C3A per a estructures de control en una única passada.

Diferència principal respecte la Pràctica 2

- En la Pràctica 2, el C3A corresponent a les expressions aritmètiques i subprogrames era complet, en el sentit que cada proposició de C3A generada era definitiva, i es podia escriure directament a arxiu sense cap pèrdua d'informació.
- La situació canvia quan hi ha estructures de flux de control, ja que sovint es generen instruccions de salt (`GOTO`) cap endavant, on el destí no es coneix inicialment. Per exemple, en una sentència `if-then-else`, quan es compila l'expressió booleana, apareixen `GOTO`s que han d'anar dirigits cap als inicis de les parts del `then` i de l'`else`. Les destinacions d'aquests `GOTO`s no es coneixen fins que finalitza la compilació de tota l'expressió booleana (la condició de l'`if`) i de la llista de sentències que formen el `then`. Per tant, cal guardar-se la llista de `GOTO`s incomplets, i completar-los quan es conegui les seves destinacions; aquest procediment és l'anomenat back-patching.
- Per fer-ho, cal guardar el C3A que s'està generant en memòria, de manera que es puguin substituir els `GOTO`s incomplets per `GOTO`s amb destinació. Una opció senzilla seria utilitzar una taula de cadenes, `codi_c3a`, de manera que la proposició de C3A amb número de línia `n` estigui a la posició `n`, és dir, a `codi_c3a[n]`. Evidentment, això s'ha de fer bé, definint-ho com un tipus abstracte de dades, que proporciona tota la funcionalitat necessària per a crear, guardar, afegir i completar.
- A més de l'estructura de dades per emmagatzemar i manipular el C3A, també fan falta llistes de números de línia on hi ha `GOTO`s indeterminats. Cal poder-les crear, fusionar, i utilitzar per a completar els `GOTO`s indeterminats. Aquestes llistes existiran com a atributs de la gramàtica, i seran gestionades en les accions semàntiques corresponents.

Llenguatge font

A continuació es detallen les diferències en el llenguatge font respecte la Pràctica 2.

Literals i comentaris

- Es recuperen els literals booleans `true` i `false`.

Identificadors, variables i tipus

- El tractament de variables booleanes es recupera, però de forma optativa.

Expressions aritmètiques

- Sense canvis.

Expressions booleanes

- Es recuperen les expressions booleanes. L'avaluació d'expressions booleanes s'ha de fer en curtcircuit, utilitzant la tècnica del back-patching per a generar el C3A corresponent. La sintaxi és la mateixa que a la Pràctica 1, amb els operadors relacionals (>, >=, <, <=, ==, `!=`) aplicats a expressions aritmètiques, i mitjançant els operadors booleans (`!`, `&&`, `||`) aplicats a expressions booleanes.
- L'ordre de precedència dels operadors booleans és també el mateix que a la Pràctica 1: `!` major precedència que `&&`, i `&&` major que `||`.

Sentències

- Les sentències poden ser: expressions aritmètiques, assignacions, crides a procediments, sentències de retorn, condicionals i iteratives. Les condicionals i iteratives són noves, la resta mantenen el funcionament respecte la Pràctica 2.
- La utilització d'expressions booleanes com a sentències és optativa, i està relacionada amb l'optativitat de les variables booleanes. La idea és que es necessiten expressions booleanes per a les sentències de flux de control, però no és necessari poder emmagatzemar el seu valor per a fer-les servir. Quan una expressió booleana apareix com a sentència, aleshores el que s'està demanant implícitament és mostrar el seu valor, operació equivalent a guardar-lo.
- Hi ha tres tipus de sentències condicionals: sense alternativa, amb alternativa, i amb múltiples alternatives.
- La sintaxi de les sentències condicionals sense alternativa és:

```
if expressió_booleana
  llista_de_sentències
end
```

- La sintaxi de les sentències condicionals amb alternativa és:

```
if expressió_booleana
  llista_de_sentències
else
  llista_de_sentències
end
```

- La sintaxi de les sentències condicionals amb alternatives múltiples admet dues versions:

  `if` *expressió_booleana*
  llista_de_sentències
  `elseif` *expressió_booleana*
  llista_de_sentències
  ...
  `elseif` *expressió_booleana*
  llista_de_sentències
  `end`

  o també

  `if` *expressió_booleana*
  llista_de_sentències
  `elseif` *expressió_booleana*
  llista_de_sentències
  ...
  `elseif` *expressió_booleana*
  llista_de_sentències
  `else`
  llista_de_sentències
  `end`
- Hi ha dos tipus de sentències iteratives: condicionals i indexades.
- La sintaxi de les sentències iteratives condicionals és:

  `while` *expressió_booleana*
  llista_de_sentències
  `end`
- La sintaxi de les sentències iteratives indexades és:

  `for` *id* **`in`** *rang*
  llista_de_sentències
  `end`
- La variable d'indexació *id* queda declarada com de tipus enter.
- Els rangs de les iteratives indexades són de la forma

  inici`:`*final*

  o alternativament

  inici`:`*increment*`:`*final*

  on *inici*, *final* i *increment* són expressions aritmètiques que prenen valors enters. Com a cassos particulars habituals, poden ser també literals enters o variables amb valor enter..
- *Exemples*: el rang `1:5` equival a la seqüència de valors 1, 2, 3, 4, 5, mentre els rangs `1:2:7` i `1:2:8` equivalen a la seqüència de valors 1, 3, 5, 7. També podem utilitzar rangs com `I:(J + 3 * K)`.

Procediments i funcions

- Sense canvis.

Funcions predeterminades

- Sense canvis.

Programa

- Sense canvis.

Codi de tres adreces

- L'única diferència és la utilització de proposicions de C3A de salt, tant condicionals com incondicionals, que no feien falta anteriorment.

Entrades i sortides

Entrada

- Sense canvis.

Sortida principal

- Sense canvis.

Altres sortides

- Sense canvis.

Opcions

- Enters i reals de diferents mides.
- Nous tipus bàsics: cadenes, caràcters., registres, llistes, apuntadors, tuples, etc.
- Operadors d'actualització com `+=`, `*=`, `-=`, `/=`, `%=`.
- Operador ternari: `? :`
- Operacions aritmètiques de vectors i matrius.
- Variables booleanes, i sentències que són expressions booleanes.
- Noves sentències de flux de control condicionals, com el **`switch`**.
- Noves sentències de flux de control iteratives, com el **`repeat-until`** (iterativa amb condició de sortida al final de la sentència), o el bucle infinit **`loop`** (iterativa incondicional).
- A les iteratives, afegir noves sentències de sortida immediata **`break`**, de retorn a la condició d'iteració **`continue`**, i/o de sortida condicional **`break-if`**.
- Procediments i funcions encaixades, amb o sense accés a variables no locals.

Exemple

La sortida podria ser lleugerament diferent, segons com s'implementi el compilador.

Entrada

```
i = 1
total = (1.0 + 2 * 3) - i
while total < 1000.0
  if i % 2 == 1
    total = total * 2
  else
    total = total + 1
  end
  j = -i
  if total > 666.6 || total == 500.0
    i = i + 1
  end
end
total
```

Sortida

```
 1:  START main
 2:  i := 1
 3:  $t01 := I2D i
 4:  $t02 := 7.0 SUBF $t01
 5:  total := $t02

 6:  IF total LTF 1000.0 GOTO 8
 7:  GOTO 23

 8:  $t03 := i MODI 2
 9:  IF $t03 EQ 1 GOTO 11
10:  GOTO 14
11:  $t04 := total MULD 2.0
12:  total := $t04
13:  GOTO 16
14:  $t05 := total ADDD 1.0
15:  total := $t05

16:  $t06 := CHSI i
17:  j := $t06

18:  IF total GTF 666.6 GOTO 22
19:  GOTO 20
20:  IF total EQ 500.0 GOTO 22
21:  GOTO 6
22:  $t07 := i ADDI 1
23:  i := $t07
24:  GOTO 6

25:  PARAM total
26:  CALL PUTD,1
27:  HALT
28:  END
```

Execució del compilador

- Els noms dels arxius d'entrada i sortida s'han de poder passar per línia de comandes, en algun dels següents formats:
  - `$ ./nom_compilador nom.jl nom-c3a.txt`
  - `$ ./nom_compilador nom`
  - `$ ./nom_compilador -i nom.jl -o nom-c3a.txt`

Lliurament

- Aquesta pràctica és fa en parelles, encara que s'admet fer-la de forma individual.
- El lliurament es farà via Moodle, en les dates indicades.
- Cal lliurar un únic arxiu comprimit (zip, rar, gz, bz, etc.) que contingui:
  1. Tot el codi font (sense arxius generats en el procés de compilació)
  2. Exemples i la seva sortida corresponent
  3. Scripts o Makefile (`make`, `make clean`, `make examples`)
  4. Documentació: preferiblement en format pdf. Ha de contenir:
     - Instruccions per la compilació i execució
     - Llistat de les funcionalitats obligatòries implementades
     - Llistat de les funcionalitats opcionals implementades
     - Llistat de les limitacions de funcionament
     - Descripció extensa de tot allò que vulgueu destacar de la vostra pràctica: detalls de les funcionalitats opcionals, decisions de disseny, etc.
     - Contribució de cadascun dels membres a la pràctica
- El nom de l'arxiu lliurat ha de ser del tipus:
  - `PR3-Nom1Cognom1-Nom2Cognom2.zip`.