Universidade Federal de Alfenas - UNIFAL-MG
Bacharelado em Ciência da Computação
Prof. Luiz Eduardo da Silva

Trabalho de Compiladores
Registro e verificação de tipos

Objetivo

O objetivo desse trabalho é incrementar o projeto do compilador para linguagem simples a fim de permitir a compilação do tipo registro. Além disso o compilador deve incluir ações semântica para verificação de tipos nas expressões que contenham registro.

Problema

O registro é uma estrutura de dados heterogênea, que compõe um conjunto de variáveis que podem ter tipos diferentes. Cada elemento do conjunto registro é acessado individualmente através da **expressão de acesso**: <nome-registro>.<nome-campo>. Os campos só podem ser acessados dessa forma. Os campos dos registros também podem ser registro. Na criação do registro, deve ser reservado espaço para cada campo, em posições contíguas a partir de um endereço base. O nome do registro se refere a esse endereço inicial da estrutura na memória. Então, cada campo corresponde a um deslocamento a partir desse endereço inicial.

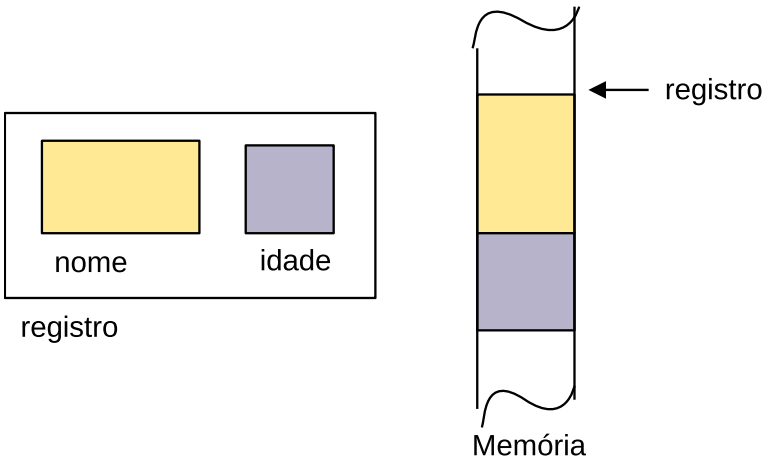

Figura 1: Ilustração do registro na memória

Roteiro

1. Basicamente, deverão ser alteradas as regras para declaração de variáveis, leitura de variáveis, comando de atribuição e expressões para permitir o uso de registros na linguagem simples.

   (a) Modificar as regras de declaração de variáveis para permitir declarações dessa forma:

```
programa registro1
def                  //  Representacao      Memoria
  inteiro a          //   y                   +-------+
  logico b           //  +-------+      0 |       | x
fimdef c             //  | x     |           +-------+
                     //  | +---+ |      1 |       | a  | x  | y
def                  //  | |   |a|           +-------+    |    |
  registro c x       //  | +---+ |      2 |       | b  |    |
  inteiro y          //  | |   |b|           +-------+         |
fimdef d             //  | +---+ |      3 |       | y          |
                     //  | y     |           +-------+
inteiro x            //  | +---+ |      4 |       | z
registro d y         //  | |   | |           +-------+
logico z             //  | +---+ |      5 |       |
                     //  +-------+             (...)
inicio
fimprograma
```

(b) Esta declaração deve ser armazenada na tabela de símbolos para posterior tradução das operação com as variáveis do tipo registro. Uma estrutura sugerida é a lista encadeada de campos:

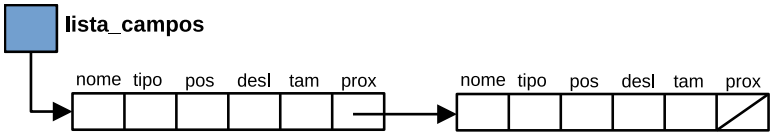

Figura 2: Lista encadeada de campos

Onde os dados de cada campo são:

- nome: é o nome do campo do registro
- tipo: é o tipo de campo (INT, LOG ou REG)
- pos: é a posição do tipo na tabela de símbolos (para simplificar a utilização de tipos registro em outras definições/declarações)
- desl: é o deslocamento a partir da posição inicial do registro para alcançar o campo na memória.
- tam: é o tamanho do campo (o número de posições utilizadas pelo campo na memória)
- prox: é o encadeamento para o próximo campo no registro.

A declaração anterior deve preencher os seguintes valores na tabela de símbolos:

Tabela de Simbolos

ID	END	TIP	TAM	POS	CAMPOS
inteiro	-1	INT	1	0	
logico	-1	LOG	1	1	
c	-1	REG	2	2	(a,INT,0,0,1)=> (b,LOG,1,1,1)
d	-1	REG	3	3	(x,REG,2,0,2)=> (y,INT,0,2,1)
x	0	INT	1	0	
y	1	REG	3	3	(x,REG,2,0,2)=> (y,INT,0,2,1)
z	4	LOG	1	1	

- Onde (x,REG,2,0,2), representa o campo de **nome** x que é do **tipo** registro (REG), a **posição** da definição desse tipo registro na tabela de símbolos é 2, o **deslocamento** para acessar esse campo a partir do endereço inicial do registro é 0 e esse campo ocupa (tem **tamanho** de) 2 posições na memória.
- Observe que para facilitar e generalizar as operações com o tipo registro, sugere-se pré-cadastrar as informações dos tipos pré-definidos (inteiro e lógico). Observe ainda que para os identificadores da tabela de símbolos que são tipos definidos (registros) e pré-definidos (inteiro e lógico) não existe endereço (END = -1). O endereço só existe para as variáveis.
- Observe também que os campos não aparecem como variáveis na tabela de símbolos. Somente como campos na lista de campos. No exemplo, os campos a e b do campo x. E os campos x e y no registro y.

(c) Modificar as regras onde podem ser utilizadas expressões de acesso com registros e campos de registro (leitura, escrita, atribuição e expressões de um modo geral), para traduzir a expressão de acesso para locais na memória, conforme exemplo:

```
/*
                    TABELA DE SIMBOLOS
      ID  |  END  |  TIP  |  TAM  |  POS  |  CAMPOS
-------------------------------------------------------------------------
inteiro   |   -1  |  INT  |    1  |    0  |
 logico   |   -1  |  LOG  |    1  |    1  |
       c  |   -1  |  REG  |    2  |    2  |  (a,INT,0,0,1)=> (b,INT,0,1,1)
       a  |    0  |  INT  |    1  |    0  |
       b  |    1  |  INT  |    1  |    0  |
       x  |    2  |  REG  |    2  |    2  |  (a,INT,0,0,1)=> (b,INT,0,1,1)
       y  |    4  |  REG  |    2  |    2  |  (a,INT,0,0,1)=> (b,INT,0,1,1)
*/
programa registro9
  def
    inteiro a
    inteiro b
  fimdef c
  inteiro a
  inteiro b
  registro c x y
inicio
  leia a
  leia b
  y.a <- a
  y.b <- b
  x <- y
  enquanto x.a < x.b faca
     escreva x.a
     x.a <- x.a + 1
  fimenquanto
fimprograma
```

Para produzir a seguinte tradução:

```
        INPP
        AMEM    6
        LEIA
        ARZG    0
        LEIA
        ARZG    1
        CRVG    0
        ARZG    4
```

```
 9            CRVG    1
10            ARZG    5
11            CRVG    5       // x <- y
12            CRVG    4
13            ARZG    2
14            ARZG    3
15  L1        NADA
16            CRVG    2
17            CRVG    3
18            CMME
19            DSVF    L2
20            CRVG    2
21            ESCR
22            CRVG    2
23            CRCT    1
24            SOMA
25            ARZG    2
26            DSVS    L1
27  L2        NADA
28            DMEM    6
29            FIMP
```

Entrega

1. Incluir um comentário no cabeçalho de cada programa fonte com o seguinte formato:

```
1  /*+-------------------------------------------------------------------
2    |            UNIFAL - Universidade Federal de Alfenas.
3    |              BACHARELADO EM CIENCIA DA COMPUTACAO.
4    | Trabalho..: Registro e verificacao de tipos
5    | Disciplina: Teoria de Linguagens e Compiladores
6    | Professor.: Luiz Eduardo da Silva
7    | Aluno.....: Fulano da Silva
8    | Data......: 99/99/9999
9    +-------------------------------------------------------------------*/
```

2. A pasta com o projeto deverá incluir o seguinte arquivo *makefile*:

```
1  simples : utils.c lexico.l sintatico.y;\
2          flex -o lexico.c lexico.l;\
3          bison -o sintatico.c sintatico.y -v -d;\
4          gcc sintatico.c -o simples
5
6  limpa   : ;\
7          rm -f lexico.c sintatico.c sintatico.output sintatico.h simples\
```

3. O compilador deverá ter o nome "simples"e ser executado através da seguinte chamada:

```
1  ./simples nomeprograma[.simples]
```

4. Enviar num arquivo único (.ZIP), a pasta do projeto com somente os arquivos fontes (lexico.l, sintatico.y, utils.c e makefile), através do Envio de Arquivo do MOODLE.