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

Trabalho de Compiladores
Vetor 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 vetor. Além disso o compilador deve incluir ações semântica para verificação de tipos nas expressões.

Problema

O vetor é uma estrutura de dados homogênea que corresponde a um conjunto de variáveis. Cada elemento (variável) do conjunto vetor é acessada individualmente através da expressão <nome-vetor> [<posição>]. Na criação do vetor é reservada uma sequência de posições contíguas na memória, do tamanho do vetor, conforme ilustrado na Figura 1. O nome do vetor se refere ao endereço inicial (endereço de base) do vetor na memória. A expressão **vetor[i]**, determina o acesso de uma variável do conjunto, cujo endereço é igual <Endereço de Base do Vetor> + <Deslocamento> de 'i' posições.

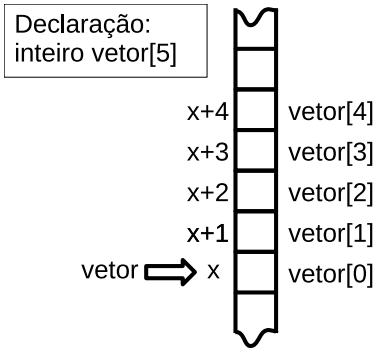

Figura 1: Ilustração do vetor na memória

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 vetores na linguagem simples. Exemplos:

   - Declaração de um vetor de nome A com 10 posições de inteiro e outras declarações:

```
1 inteiro A[10] i j
2 logico B[5]
3 inteiro x y
```

   Para a tradução de vetores, a tabela de símbolos deverá conter os campos:

(a) **id** - nome do identificador

(b) **end** - endereço da variável ou vetor (endereço base)

(c) **tip** - tipo da variável (inteiro ou lógico)

(d) **cat** - categoria da variável (variável ou vetor)

(e) **tam** - tamanho (número de posições ocupadas pela variável na memória)

A declaração anterior deve preencher a seguinte Tabela de Símbolos:

#	id	end	tip	cat	tam
0	A	0	INT	VET	10
1	i	10	INT	VAR	1
2	j	11	INT	VAR	1
3	B	12	LOG	VET	5
4	x	17	INT	VAR	1
5	y	18	INT	VAR	1

- Leitura de um valor para a posição A[x+1]:

```
1 leia A[x+1]
```

- Atribuição de um valor para uma posição do vetor:

```
1 A[i] <- 10
```

- A utilização de um elemento do vetor numa expressão:

```
1 soma <- 10 * A[i+1]
```

2. Considere a existência das instruções **ARZV** e **CRVV** da máquina MVS, representada nos pseudocódigos:

Código 1: Instrução para armazernar no vetor

```
1 ARZV n   (Armazena no Vetor)
2
3 M[n + M[s-1]] <- M[s]
4 s <- s - 2
```

Código 2: Instrução para carregar o valor do vetor

```
1 CRVV n   (Carrega Valor do Vetor)
2
3 M[s] <- M[n + M[s]]
```

- A tradução para **leia** <**nomevetor**>[<**expressão**>] no comando de leitura deverá ser:

Tradução da expressão LEIA ARZV <endereço do vetor>

- A tradução para <**nomevetor**>[<**expressão-1**>] <**- expressão-2** no comando de atribuição deverá ser:

Tradução da expressão-1 Tradução da expressão-2 ARZV <endereço do vetor>

- A tradução para <**nomevetor**>[<**expressão**>] numa expressão deverá ser:

Tradução da expressão CRVV <endereço do vetor>

3. Apresentamos abaixo um programa (Código 3) que usa um vetor e a tradução para MVS (Código 4) correpondente:

Código 3: Programa exemplo em linguagem Simples

```
1  programa testavetor
2    inteiro vetor[4]
3    inteiro i
4  inicio
5    i <- 0
6    enquanto i < 4 faca
7      vetor[i] <- (i+1) * 10
8      i <- i + 1
9    fimenquanto
10   i <- 4
11   enquanto nao (i < 1) faca
12     escreva vetor[4-i]
13     i <- i - 1
14   fimenquanto
15 fimprograma
```

Código 4: Tradução para MVS

```
1          INPP
2          AMEM    5
3          CRCT    0
4          ARZG    4
5  L1      NADA
6          CRVG    4
7          CRCT    4
8          CMME
9          DSVF    L2
10         CRVG    4
11         CRVG    4
12         CRCT    1
13         SOMA
14         CRCT    10
15         MULT
16         ARZV    0
17         CRVG    4
18         CRCT    1
19         SOMA
20         ARZG    4
21         DSVS    L1
22 L2      NADA
23         CRCT    4
24         ARZG    4
25 L3      NADA
26         CRVG    4
27         CRCT    1
28         CMME
29         NEGA
30         DSVF    L4
31         CRCT    4
32         CRVG    4
33         SUBT
```

```
34              CRVV    0
35              ESCR
36              CRVG    4
37              CRCT    1
38              SUBT
39              ARZG    4
40              DSVS    L3
41  L4          NADA
42              DMEM    5
43              FIMP
```

4. Além disso, deverão ser incluídas ações semânticas para **verificação de tipo**, durante a compilação. Alguns exemplos de verificações semânticas de tipo:

   - Na atribuição, o tipo da expressão deverá ser compatível com o tipo da variável no lado esquerdo da atribuição.
   - A expressão, no comando de seleção ou repetição, deverá ser do tipo lógico.
   - O tipo da expressão de acesso a vetor deverá ser inteiro
   - Entre outras.

Entrega

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

```
/*+-------------------------------------------------------------------
  |            UNIFAL - Universidade Federal de Alfenas.
  |              BACHARELADO EM CIENCIA DA COMPUTACAO.
  | Trabalho..: Vetor e verificacao de tipos
  | Disciplina: Teoria de Linguagens e Compiladores
  | Professor.: Luiz Eduardo da Silva
  | Aluno.....: Fulano da Silva
  | Data......: 99/99/9999
  +-------------------------------------------------------------------*/
```

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

```
simples : estrut.c lexico.l sintatico.y;\
        flex -o lexico.c lexico.l;\
        bison -o sintatico.c sintatico.y -v -d;\
        gcc sintatico.c -o simples

limpa   : ;\
        rm -f lexico.c sintatico.c sintatico.output *~ sintatico.h simples\
```

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

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

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