Construção de Compiladores
Aula 10 - Chamadas de Procedimento
Sem Parâmetros, Sem Variáveis Locais

Bruno Müller Junior

Departamento de Informática
UFPR

17 de Setembro de 2014

1 Sintaxe

2 Escopo e visibilidade
- Exemplo
- Registro de Ativação
  - Esquema de R.A na MEPA
  - Esquema
- Instruções Novas
- Funcionamento
  - Esquema de Tradução
- Exemplo sem Parâmetros, Sem V.L.

3 Projeto

Sintaxe

- A regra de declaração de um procedimento em Pascal ocorre a partir da regra <bloco> (regra 2).
- Nas regras do Pascal Simplificado, corresponde à regra 11: <parte de declaração de subrotinas>.
- Esta aula concentra-se na regra 12: <declaração de procedimento> .

```
2.  <bloco> ::=  ...                                          |
                 <parte de declaração de subrotinas>          |
                 ...                                          |
11.  <parte de declaração de subrotinas> ::=
                 <declaração de procedimento> |
                 <declaração de função>
12. <declaração de procedimento> ::= PROCEDURE <ident> [<parâmetros formais>]
                                     ; <bloco>
```

Escopo e visibilidade

- Chamadas de procedimento estão presentes em praticamente todas as linguagens de programação.
- Elas consistem em:
  1. desviar o fluxo de execução para um escopo nomeado (o nome do procedimento);
  2. ao final do procedimento, retornar o fluxo para o comando seguinte daquele onde foi feita a chamada.
- Na linguagem Pascal, onde os procedimentos podem ser de vários níveis léxicos, existe a questão da visibilidade.
- Um procedimento de nível léxico k "vê" todos os procedimentos que:
  1. tem nível léxico menor ou igual a k nos quais ele está encaixado;
  2. tem nível léxico $k + 1$ encaixados nele.

Escopo e visibilidade

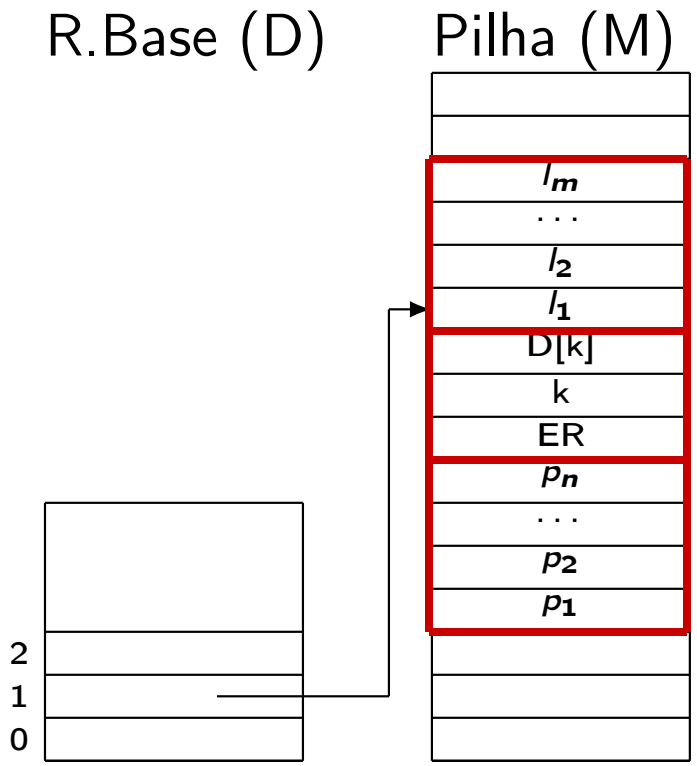
```
program escopoProc (input, output);
|   procedure p;
|   |  procedure q;
|   |  | procedure r;
|   |  | |-- r --- (vê p, q, r)
|   |  | procedure s;
|   |  | |-- s --- (vê p, q, r, s)
|   |  |-- q --- (vê p, q, r, s)
|   |-- p --- (vê p, q)
|   procedure t;
|   |  procedure u;
|   |  |-- u --- (vê p, t, u)
|   |  procedure v;
|   |  |-- v --- (vê p, t, u, v)
|   |-- t --- (vê p, t, u, v)
|-- principal --- (vê p, t)
```

Registro de Ativação

- Em tempo de execução, cada chamada de procedimento é associado a uma estrutura chamada "Registro de Ativação", que contém informações como:
  - parâmetros e variáveis locais;
  - ponto do código para onde voltar (endereço de retorno);
  - endereço de base;
- Existem várias formas de implementar um R.A.
- Em Pascal, um procedimento de nível léxico k precisa ter acesso a todas as variáveis de nível léxico menor que k (preservados os "encaixamentos" dos procedimentos).

Esquema de R.A na MEPA

- O esquema ao lado corresponde a um R.A. genérico para um procedimento de nível léxico 1.



Esquema

- Um R.A. é dividido em três partes: parâmetros, informações gerenciais e variáveis locais.
- Cada D[k] aponta para a primeira variável local do R.A. de nível k.
- Assim o endereço léxico das variáveis locais "funciona" (por exemplo, `CRLV k,0`.
- Os parâmetros estão presentes a partir de $k - 4$ (`CRLV k,-4`).
- As informações gerenciais armazenam:
  - Endereço de Retorno (ER) em $k - 3$
  - Nível léxico do chamador (k)
  - conteúdo anterior de D[k]

Instruções Novas

- Este esquema é implementado no uso de três instruções novas.

Instrução	Ação	Significado

CHPR p,k	`M[s+1]:=i+1`
	`M[s+2]:=nivel lex. atual`
	`s:=s+2`
	`i:=p`
		Chama Procedimento

RTPR k,n	`D[k]:=M[s]`
	`i:=M[s-2]`
	`s:=s-(n+3)`
		Retorna de Procedimento

ENPR k	`s:=s+1`
	`M[s]:=D[k]`
	`D[k]:=s+1`
		Entra em Procedimento

Funcionamento

- Na chamada de procedimento:
  1. empilha parâmetros
  2. Chama procedimento (CHPR), que empilha ER e k.
- Na entrada do procedimento
  1. Substitui D[k] (salva o anterior)
  2. Aloca variáveis locais (AMEM)
- Na saída do procedimento
  1. Desaloca variáveis locais (DMEM)
  2. Retira infos gerenciais, atualiza D[k] e retira parâmetros (RTPR).

Esquema de Tradução

PROCEDURE p	`DSVS R00` `R01:ENPR k`
VAR ... ... BEGIN	*Traduz bloco*
END	*RTPR k, n*
... p; ...	*Empilha Parâmetros* `CHPR P01, k`

Exemplo sem Parâmetros, Sem V.L.

```
1  program proc1 (input, output);
2        var   x, y: integer;
3        procedure p;
4              var z:integer;
5              begin
6                    z:=x;
7                    x:=x-1;
8                    if (z>1)
9                          then p(1)
10                         else y:=1;
11                   y:-y*z
12             end
13       begin
14             read(x);
15             p(2);
16             write (x,y)
17       end.
```

Projeto

- Problema: quando o bison encontrar o identificador, ele não tem informação suficiente para saber se vai para atribuição ou se vai para procedimento.
- Isto ocorre porque o bison só olha um símbolo à frente, e precisamos de dois.
- Uma solução é fatorar a gramática como no modelo abaixo.

```
A ::= <ident> ‘‘:=’’  ... | # Atribuição
      <ident> ‘‘(’’   ... | # Chama Proc (com param.)
      <ident> ‘‘;’’   ...   # Chama Proc (sem param.)

A ::= <ident> <A-Continua>
A-Continua ::=  ‘‘:=’’  ... | # Atribuição
                ‘‘(’’   ... | # Chama Proc (com param.)
                ‘‘;’’   ...   # Chama Proc (sem param.)
```