Geração de Código Objeto (Hipo) – Continuação

Fonte: http://www.icmc.sc.usp.br/~gracan/download/sce126/

Instruções

- Instruções para procedimentos
  - No caso de procedimentos com parâmetros, na chamada (CHPR) e no término (RTPR), deve-se estabelecer a associação entre os parâmetros formais e os atuais, na pilha D
  - Na LALG, só passagem por VALOR

Exemplo 1

Endereços pilha de dados:
x(0), y(1), *ret**(2), z(3)

program ex1;	0. INPP	*else y := 1;*	18. CRCT 1
var x, y : integer;	1. ALME 1		19. ARMZ 1
	2. ALME 1	*y := y * z*	20. CRVL 1
procedure p;	**3. DSVI 26**		21. CRVL 3
var z: integer;	4. ALME 1		22. MULT
begin			23. ARMZ 1
z := x;	5. CRVL 0	*end; {procedure}*	**24. DESM 1**
	6. ARMZ 3		**25. RTPR**
x := x - 1;	7. CRVL 0	*begin {principal}*	
	8. CRCT 1	*read(x);*	26. LEIT
	9. SUBT		27. ARMZ 0
	10. ARMZ 0	*p;*	**28. PUSHER 30**
If x > 1	11. CRVL 0		**29. CHPR 4**
	12. CRCT 1	*write(x,y);*	30. CRVL 0
	13. CPMA		31. IMPR
	14. DSVF **18**		32. CRVL1
then x := z	15. CRVL 3		33. IMPR
	16. ARMZ 0	*end. {principal}*	34. PARA
	17. DSVI **20**		

Exemplo 2

Endereços pilha de dados:
a(0), b(1), *ret**(2), x(3), y(4), k(5)

program ex2;	0. INPP	*p(a, b);*	**17. PUSHER 21**
var a, b: integer;	1. ALME 1		**18. PARAM 0**
	2. ALME 1		**19. PARAM 1**
procedure p(x, y: integer);	**3. DSVI 13**		**20. CHPR 4**
var k: integer;	4. ALME 1	*end. {principal}*	21. PARA
begin			
k := x + y;	5. CRVL 3		
	6. CRVL 4		
	7. SOMA		
	8. ARMZ 5		
x := k;	9. CRVL 5		
	10. ARMZ 3		
end;	**11. DESM 3**		
	12. RTPR		
begin {principal}			
read(a, b);	13. LEIT		
	14. ARMZ 0		
	15. LEIT		
	16. ARMZ 1		

Instruções

(26) PARAM n

{aloca memória e copia valor da posição n para o topo de D}

s:= s+1;

D[s]:= D[n]

(27) PUSHER e

{empilha o índice *e* da instrução seguinte à chamada do procedimento, como endereço de retorno, no array C}

s := s + 1;

D[s] := e (depende do número n de parâmetros: e=i+n+1)

Instruções

(28) CHPR p

{desvia para instrução de índice p no array C, obtido na TS}

i:= p

(29) DESM m

{desaloca m posições de memória, a partir do topo s de D}

s:=s - m

(30) RTPR

{retorna do procedimento – endereço de retorno estará no topo de D – e desempilha o endereço}

i:= D[s];

s:= s-1

Instruções

- Seqüência de instruções na:
  - chamada de procedimento:

    PUSHER e

    {PARAM n}

    {.....}

    CHPR p
  - no início do procedimento:

    {DSVI k}

    {.....}
  - no término do procedimento:

    DESM n

    RTPR

Instruções

- **Observações:**
  - Ao terminar a execução de um procedimento, deveremos ter na pilha de dados as variáveis globais e locais. Assim, se o procedimento tiver n variáveis locais, s deve apontar para a n-ésima variável quando o procedimento tiver sido executado uma única vez
  - Todo retorno de procedimento deve vir conjugado a uma desalocação de memória. Assim, ao desalocar memória, s estará justamente apontando o endereço que guarda o endereço de retorno à instrução seguinte
  - Repare que ao encontrar a declaração de um procedimento, desviamos o controle para a 1ª instrução que segue as instruções daquele procedimento. O Programa Principal, ao encontrar a referência ao procedimento, desvia para sua 1ª instrução

Detalhes da Implementação

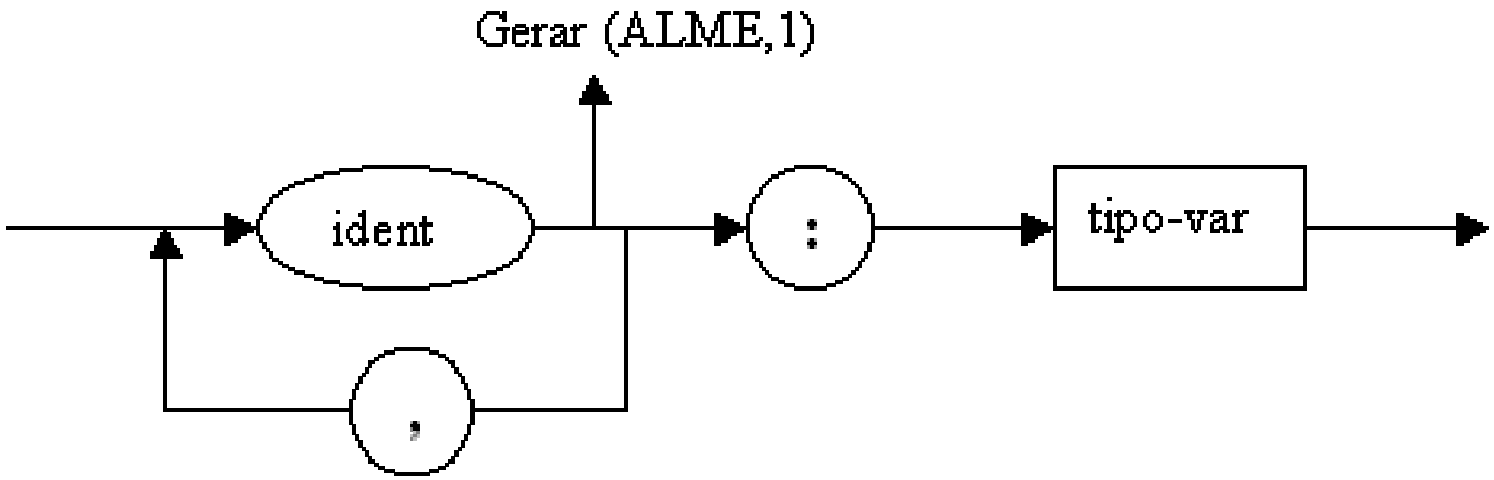
- Uma instrução é:
  - Um nome seguido de 0 ou 1 argumento, que é um índice de D ou C
- **Exemplo Pascal:**

```
Type Funções = (SOMA,SUBT,ARMZ,....);

var C: array [0..MaxC] of

        record

                F: Funções;

                P: -1..MaxInt

        end;
```

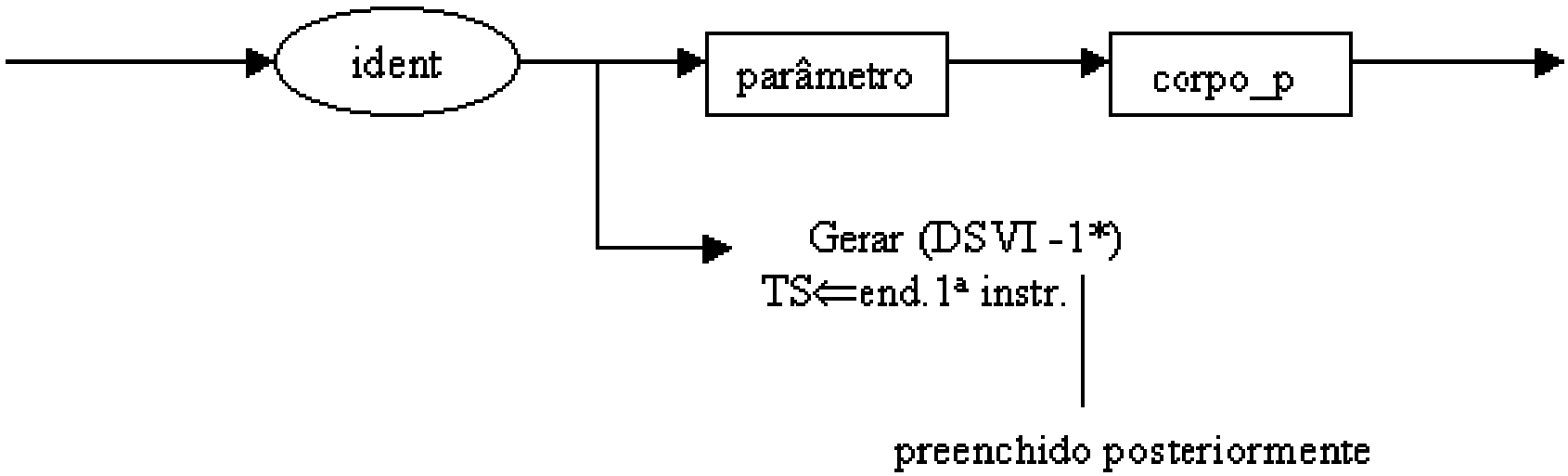
Detalhes da Implementação

```
procedure Gerar (X: Funções; Y: integer);
        begin
                i:= i+1;
                with C[i] do begin
                                F:= X; {código}
                                P:= y {argumento}
                        end
        end;
```

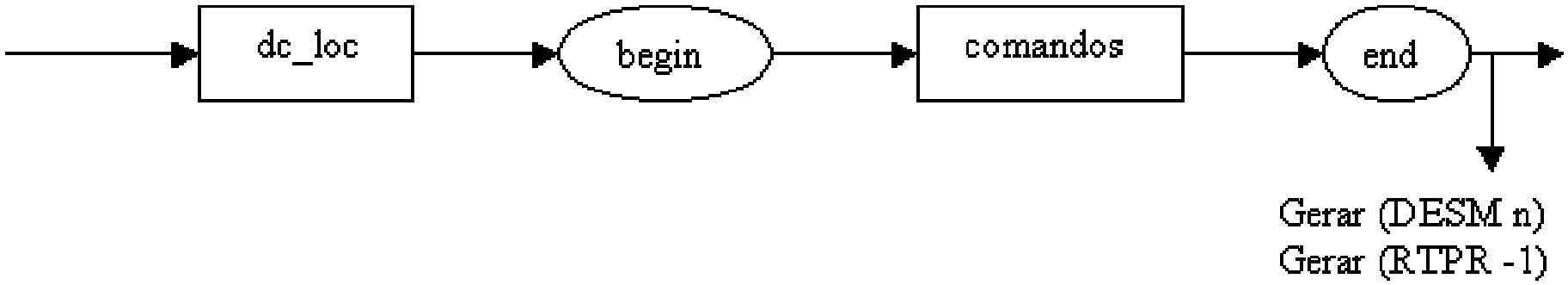
<dc_v> <variáveis> <mais_var>



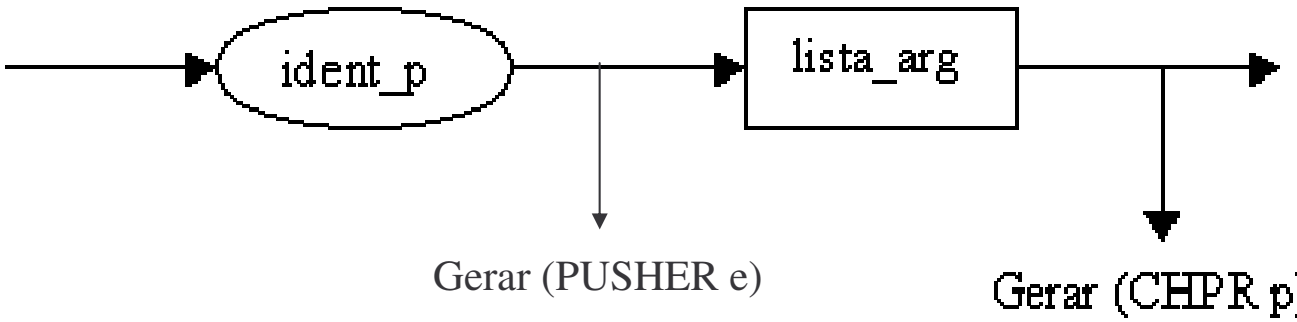
<dc_p>


ident
parâmetro
corpo_p
Gerar (DSVI -1*)
TS⇐end.1ª instr.
preenchido posteriormente

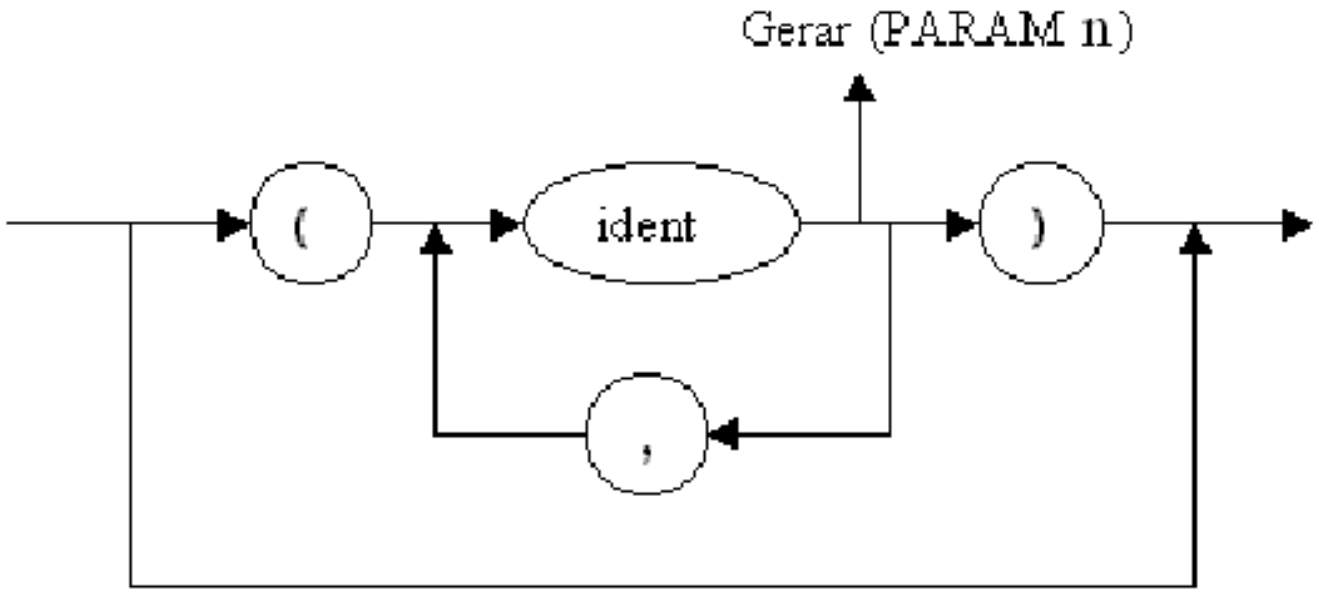
<corpo_p>



<comando>



<lista_arg>



O interpretador

- A entrada para o Interpretador é o array de códigos C. A pilha D será sua área de trabalho

- O procedimento principal seria aquele que percorreria o array C a partir da posição 0, interpretando cada instrução, conforme já vimos anteriormente. Terminaria ao encontrar a instrução PARA

- **Não implementaremos o interpretador!**
  - A saída do compilador será o **array C preenchido**
    - A pilha de Dados (D) e os ponteiros de manipulação são parte do interpretador HIPO e não do compilador