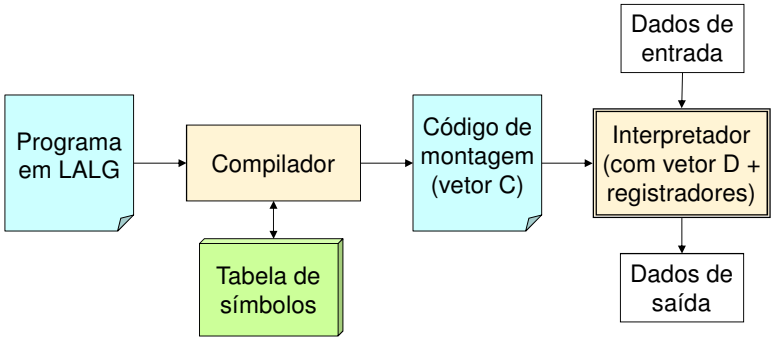
Geração de Código para LALG **(continuação)**

Ambiente de execução para LALG
Máquina hipotética
Repertório de instruções

Prof. Thiago A. S. Pardo

Geração de código para LALG

- Interpretador



Exercício - relembrando

- Gere código para o programa ao lado e interprete o código gerado

```
program exemplo3;
var x, y: integer;
begin
read(x);
y:=x*x;
if (x<y) then
while (x<y) do
begin
y=y-2;
write(y);
end;
else write(x);
write(x*y);
end.
```

Instruções vistas até agora
ALME, ARMZ, CDES, CMAI, CONJ, CPIG, CPMA, CPME, CPMI, CRCT, CRVL, DISJ, DIVI, DSVF, DSVI, IMPR, INPP, INVE, LEIT, MULT, NEGA, PARA, SOMA, SUBT

3

Exercício - relembrando

```
0)  INPP
1)  ALME 1
2)  ALME 1
3)  LEIT
4)  ARMZ 0
5)  CRVL 0
6)  CRVL 0
7)  MULT
8)  ARMZ 1
9)  CRVL 0
10) CRVL 1
11) CPME
12) DSVF 25
13) CRVL 0
14) CRVL 1
15) CPME
16) DSVF 24
17) CRVL 1
18) CRCT 2
19) SUBT
20) ARMZ 1
21) CRVL 1
22) IMPR
23) DSVI 13
24) DSVI 27
25) CRVL 0
26) IMPR
27) CRVL 0
28) CRVL 1
29) MULT
30) IMPR
31) PARA
```

```
program exemplo3;
var x, y: integer;
begin
read(x);
y:=x*x;
if (x<y) then
while (x<y) do
begin
y=y-2;
write(y);
end;
else write(x);
write(x*y);
end.
```

4

Instruções para procedimentos

- Características da LALG
  - Passagem de parâmetros por valor
  - Somente procedimentos globais

5

Instruções para procedimentos

- Ao chamar procedimento
  - Empilha-se endereço de retorno (ainda não definido, pois depende do número de parâmetros)
  - Empilham-se valores de parâmetros
  - Salta-se para 1ª instrução de procedimento
- No início do procedimento
  - Desvia-se para programa principal
  - Copia valores dos parâmetros passados
- No fim do procedimento
  - Libera memória (variáveis locais e parâmetros)
  - Retorna do procedimento

6

Instruções para procedimentos

- PUSHER e

  {empilha endereço de retorno}

  s:=s+1

  D[s]:=e

- CHPR p

  {desvia para instrução de índice p no vetor C, obtido na tabela de símbolos}

  i:=p

7

Instruções para procedimentos

- DESM m

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

  deve retirar (ou tornar inacessível) da tabela de símbolos as posições desalocadas (em tempo de compilação)

  s:=s-m

- RTPR

  {retorna do procedimento}

  i:=D[s]

  s:=s-1

8

Instruções para procedimentos

- COPVL

  sem efeito na execução

  coloca na tabela de símbolos o endereço do parâmetro, associando parâmetros atuais com formais (em tempo de compilação)

- PARAM n

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

  s:=s+1

  D[s]:=D[n]

9

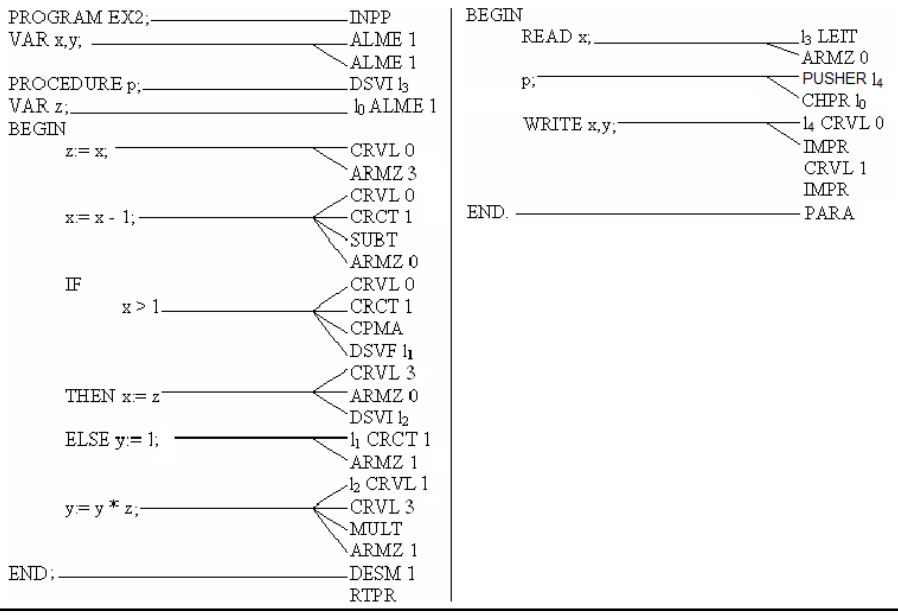
Instruções para procedimentos

- Ao chamar procedimento
  - PUSHER e
  - {PARAM n}
  - {.....}
  - CHPR p
- No início do procedimento
  - {DSVI k}
  - {COPVL}
  - {.....}
- No fim do procedimento
  - DESM n
  - RTPR

Deve levar em conta a posição usada pelo endereço de retorno na pilha D, isto é, soma-se 1 ao endereço obtido para o primeiro parâmetro ou variável local

10

Exemplo: sem parâmetros



Exemplo: com parâmetros

```
Program ex3;
var a,b: integer;
procedure proc(x,y:integer);
        var l: integer;
        begin
                l:= x + y;
                x:= l;
        end;
begin
        read(a,b);
        proc(a,b);
end.
```

1.INPP
2. ALME 1
3. ALME 1
4. DSVI 16
5. COPVL
6. COPVL
7. ALME1
8.CRVL 3
9. CRVL 4
10. SOMA
11. ARMZ 5
12. CRVL 5
13. ARMZ 3
14. DESM 3
15. RTPR
16. LEIT
17. ARMZ 0
18. LEIT
19. ARMZ 1
20. PUSHER 24
21. PARAM 0
22. PARAM 1
23. CHPR 5
24. PARA

Instruções para procedimentos

- Ao se gerar código para um procedimento, coloca-se o endereço de sua primeira instrução na tabela de símbolos
- Ao se retornar do procedimento, a pilha deve estar exatamente como estava antes da chamada do procedimento

13

Exercício

- Gere código para o programa ao lado e interprete o código gerado

```
program exemplo2;
var a: real;
var b: integer;
procedure nomep(x: real);
var a, c: integer;
begin
read(c,a);
if a<x+c then
begin
a:=c+x;
write(a);
end
else c:=a+x;
end;
begin {programa principal}
read(b);
nomep(b);
end.
```

14

Exercício

- Gere código para o programa ao lado e interprete o código gerado

```
program p;
var x: integer;
procedure nomep(a:real);
var y: integer;
begin
y:=1;
end;
procedure teste(b,c:real);
var d: integer;
begin
d:=1;
if (b>c) then
begin
b:=b-c;
c:=2;
end;
end;
begin
read(x);
nomep(x);
teste(x,5);
write(x);
end.
```

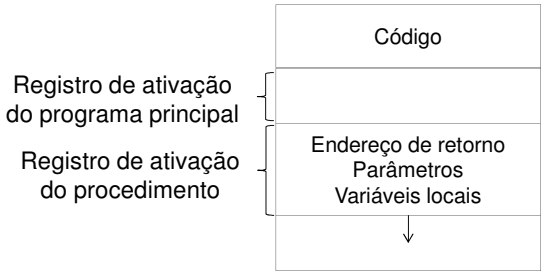
Sequência de ativação para LALG

- ???

16

Sequência de ativação para LALG

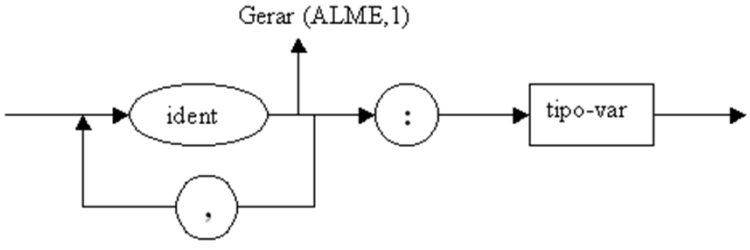
- Versão simplificada do ambiente baseado em pilhas real
  1. Empilhar endereço de retorno do procedimento
  2. Empilhar parâmetros, se houver
  3. Salta-se para o início do código do procedimento
  4. Empilhar variáveis locais, se houver



17

Geração de código para LALG

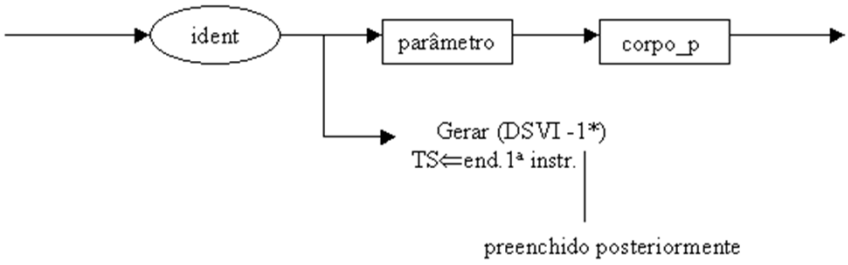
- Alguns exemplos de onde se gerar código
  - Declaração de variáveis



18

Geração de código para LALG

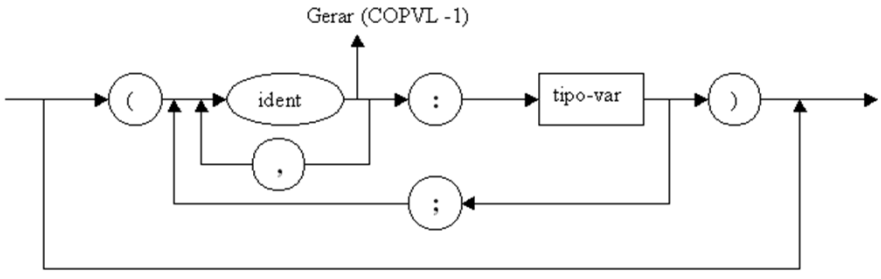
- Alguns exemplos de onde se gerar código
  - Declaração de procedimentos



19

Geração de código para LALG

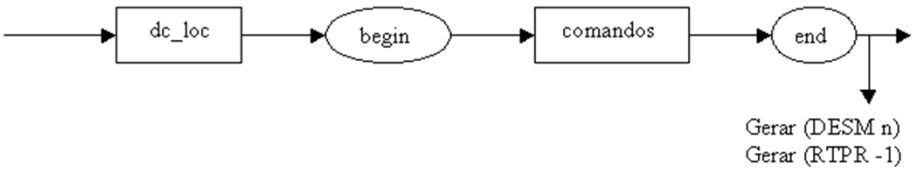
- Alguns exemplos de onde se gerar código
  - Parâmetros de procedimentos



20

Geração de código para LALG

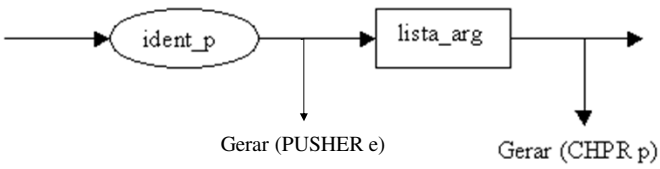
- Alguns exemplos de onde se gerar código
  - Corpo do procedimento



21

Geração de código para LALG

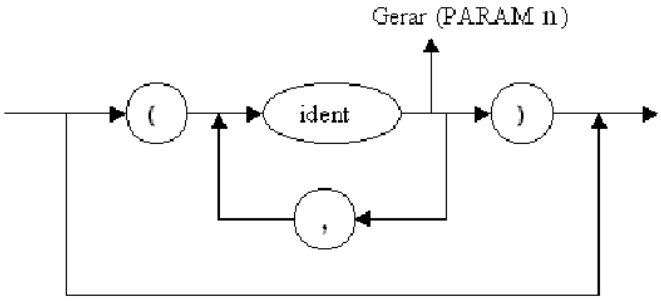
- Alguns exemplos de onde se gerar código
  - Chamada de procedimento



22

Geração de código para LALG

- Alguns exemplos de onde se gerar código
  - Lista de argumentos na chamada de procedimento



23

Exercício

- Escreva o procedimento sintático completo para a declaração de variáveis na LALG
  - Interação com análise léxica
  - Tratamento de erros sintáticos pelo modo pânico
  - Análise semântica e tratamento de erros semânticos
  - Geração de código

<dc_v> ::= var <variaveis> : <tipo_var> ;
<tipo_var> ::= real | integer
<variaveis> ::= ident <mais_var>
<mais_var> ::= , <variaveis> | λ

24

Exercício – possível solução

```
procedimento dc_v(S) {
        se (simb=var)
                então obter_símbolo()
                senão {
                        imprimir("Erro: var esperado");
                        ERRO(S+{id});
                }
        se (simb=id)
                então {
                        se busca_TS(cadeia,token=id,cat=var,escopo=0)=TRUE
                            então imprimir("Erro: identificador declarado novamente")
                            senão inserir_id_TS(cadeia,token=id,cat=var,escopo=0,end=s++);
                        se erro_léxico=FALSE e erro_sintático=FALSE e erro_semântico=FALSE
                            então gera_codigo(contador_linha++ || "ALME 1");
                        obter_símbolo();
                }
                senão {
                        imprimir("Erro: id esperado");
                        ERRO(S+{:}+{,});
                }
...
```

Rosa=interface com léxico, azul=semântica, verde=geração de código

Exercício – possível solução

```
        enquanto (simb=simb_virgula) faça {
                obter_símbolo();
                se (simb=id)
                    então {
                        se busca_TS(cadeia,token=id,cat=var,escopo=0)=TRUE
                            então imprimir("Erro: identificador declarado novamente")
                            senão inserir_id_TS(cadeia,token=id,cat=var,escopo=0,end=s++);
                        se erro_léxico=FALSE e erro_sintático=FALSE e erro_semântico=FALSE
                            então gera_codigo(contador_linha++ || "ALME 1");
                        obter_símbolo();
                    }
                    senão {
                        imprimir("Erro: id esperado");
                        ERRO(S+{:}+{,});
                    }
        }
        se (simb=simb_dp)
                então obter_símbolo()
                senão {
                        imprimir("Erro: ':' esperado");
                        ERRO(S+{real,integer});
                }
...
```

Rosa=interface com léxico, azul=semântica, verde=geração de código

Exercício – possível solução

```
        se (simb=real) ou (simb=integer)
                então {
                        inserir_tipo_ids_declarados_TS(cadeia,cat=var,escopo=0);
                        obter_símbolo();
                }
                senão {
                        imprimir("Erro: 'real' ou 'integer' esperado");
                        ERRO(S+{;});
                }
        se (simb=simb_pv)
                então obter_símbolo()
                senão {
                        imprimir("Erro: ';' esperado");
                        ERRO(S+{var});
                }
}
```

Rosa=interface com léxico, azul=semântica, verde=geração de código

27

Exercício para casa (entregar na próxima aula)

- Escreva o procedimento sintático completo para os comandos da LALG
  - Interação com análise léxica
  - Tratamento de erros sintáticos pelo modo pânico
  - Análise semântica e tratamento de erros semânticos
  - Geração de código

```
<cmd> ::= readln ( <variaveis> ) |
          writeln ( <variaveis> ) |
          while <condicao> do <cmd> |
          if <condicao> then <cmd> <pfalsa> |
          ...
<variaveis> ::= ident <mais_var>
<mais_var> ::= , <variaveis> | λ
...
```

28

Para pensar

- E se procedimento recursivo?
  - Gere o código e interprete
    - Funciona? Justifique.
    - E procedimentos que chamam outros procedimentos?

```
program p;
var x: integer;
procedure nomep(a:real);
var y: integer;
begin
y:=a+2;
if y<10 then
nomep(y);
end;
begin
read(x);
nomep(x);
end.
```