Geração de Código
Rotinas e parâmetros por valor

Portugol

```
procedimento coisa(inteiro x)
inicio
   x <- 1
fim

procedimento principal()
declaracoes
   inteiro a
inicio
   a <- 10
   coisa (a)
fim
```

M. Pilha

```
.data
   COISA_x : 0
   a : 0
.text
   JMP     _PRINCIPAL
_COISA:
   LDI     1
   STO     COISA_x
   RETURN  0
_PRINCIPAL:
   LDI     10
   STO     a
   LD      a
   STO     COISA_x
   CALL    _COISA
   HLT     0
```

getParname(nome_call, contpar)

Consuta a tabela de simbolos e retorna o nome do #contpar parametro da funcao "nome_call"

Expressões

<proc> :: = procedimento id #21 '(' <lista_par> ')' '{'<lcmd> '}' #22
<func> ::= <tipo> id #21 '(' <lista_par> ')' '{' <lcmd> return '('<exp> ')' '}' #22

<cmd> ::= id #23 '(' <lista> ')' #25
<lista> ::= id #24 ',' <lista> | id #24
...
<exp> ...
<exp2> ::= <exp2> '+' <exp3>
| <exp2> '-' <exp3>
| <exp3>
<exp3> ::= ID #24
| NUM_INT #24
| id #23 '(' <lista> ')' #25

```
#21  - Gera o rótulo de início da rotina
nome = token.getlexeme();
gera_cod ("ROT", "_"+nome);
#22  - Gera retorno ao ponto de chamada da rotina
gera_cod ("RETURN", "0");
#23 – Armazena o nome da rotina a ser chamada e inicializa o contador de parâmetros
nome_call = token.getlexeme();
contpar=0;
#24 – Copia os valores passados por parâmetros para as variáveis (parâmetros) da rotina destino
gera_cod ("LD", token.getlexeme()); // ver se é valor ou id
gera_cod ("STO", getParname(nome_call, contpar);
contpar++;
#25  - Faz a chamada da rotina
gera_cod ("CALL", "_"+nome_call);
```

Geração de Código
Rotinas e parâmetros por valor

Portugol

```
inteiro dobro(inteiro x)
inicio
   x <- x + x
   retornar (x)
fim

procedimento principal()
declaracoes
   inteiro a
inicio
   a <- dobro (3)
fim
```

M. Pilha

```
.data
   DOBRO_x : 0
   a : 0
.text
   JMP     _PRINCIPAL
_DOBRO:
   LD      DOBRO_x
   ADD     DOBRO_x
   STO     DOBRO_x
   LD      DOBRO_x
   RETURN  0
_PRINCIPAL:
   LDI     3
   STO     DOBRO_x
   CALL    _DOBRO
   STO     a
   HLT     0
```

getParname(nome_call, contpar)

Consuta a tabela de simbolos e retorna o nome do #contpar parametro da funcao "nome_call"

Expressões

<proc> :: = procedimento id #21 '(' <lista_par> ')' '{'<lcmd> '}' #22
<func> ::= <tipo> id #21 '(' <lista_par> ')' '{' <lcmd> return '('<exp> ')' '}' #22

<cmd> ::= id #23 '(' <lista> ')' #25
<lista> ::= id #24 ',' <lista> | id #24
...
<exp> ...
<exp2> ::= <exp2> '+' <exp3>
| <exp2> '-' <exp3>
| <exp3>
<exp3> ::= ID #24
| NUM_INT #24
| id #23 '(' <lista> ')' #25

```
#21  - Gera o rótulo de início da rotina
nome = token.getlexeme();
gera_cod ("ROT", "_"+nome);
#22  - Gera retorno ao ponto de chamada da rotina
gera_cod ("RETURN", "0");
#23 – Armazena o nome da rotina a ser chamada e inicializa o contador de parâmetros
nome_call = token.getlexeme();
contpar=0;
#24 – Copia os valores passados por parâmetros para as variáveis (parâmetros) da rotina destino
gera_cod ("LD", token.getlexeme()); // ver se é valor ou id
gera_cod ("STO", getParname(nome_call, contpar);
contpar++;
#25  - Faz a chamada da rotina
gera_cod ("CALL", "_"+nome_call);
```

Geração de Código
Rotinas e parâmetros por valor

Verificações

- Procedimento (sem retorno)
  - não pode estar em expressões nem atribuições, é um "comando" da linguagem
  - não pode retornar valor (ex. return a)

- Funções (com retorno)
  - pode ter ou não a diretiva "return ...." (ex. return algo)
  - verificar a compatibilidade de tipos do retorno quando usado em expressões ou atribuições
  - Realizar a cópia do "estado" da expressão antes/após de acordo com a expressão

- Ambos
  - podem ser passados parâmetros ou não – se sim, verificar número de parâmetros e tipos - compatibilidade