Universidade do Sul de Santa Catarina - UNISUL
Curso de Ciência da Computação **Campus de Tubarão**
Disciplina: Tradução de Linguagens de Programação
Professor: Charbel Szymanski **E-mail:** charbel@evoluma.com

Análise semântica e geração de código intermediário
Máquina Hipotética

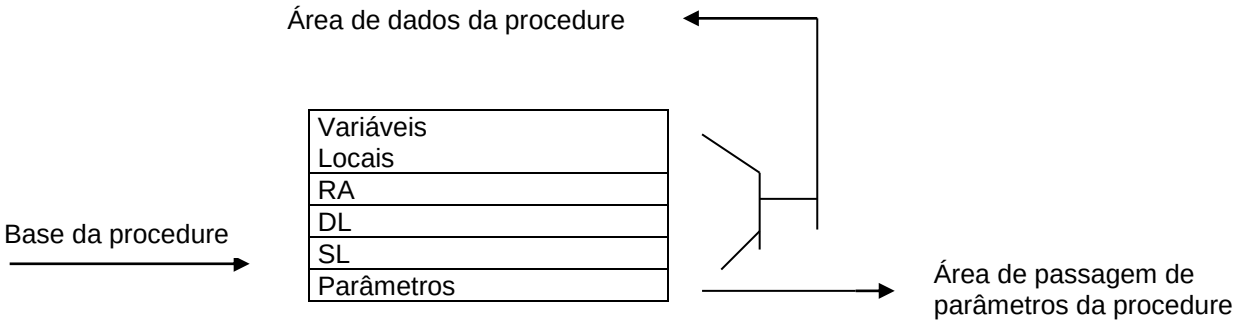
1. DESCRIÇÃO GERAL

FUNÇÃO: Executar o código intermediário, que será gerado pelo compilador desenvolvido. O funcionamento da máquina hipotética será simulado na linguagem hospedeira (Java).

ELEMENTOS:

- ÁREA DE INSTRUÇÕES: Região onde são armazenadas as instruções a serem executadas.
  → pilha_inst [0 .. tam_pilha_inst]

- ÁREA DE DADOS: Região onde serão alocados valores a serem manipulados pelas instruções.
- AREA DE LITERAIS: região para armazenamento de literais da LMS.



Inicialização : B, P :=1
RA, DL, SL :=0
TOPO:=0

Na implementação cada dupla da tabela (área de instruções) corresponde a uma instrução do código intermediário gerado, sendo subdividida no código da instrução a ser executada e em dois operandos que nem sempre são necessários, dependendo da instrução.

SL – ponteiro estático, indica contexto estático da procedure, usado para resolver endereços não locais ao bloco procedure (variáveis não locais referenciadas). Usado para apontar a base do segmento de dados aonde uma variável foi declarada, considerando a diferença de nível aonde está sendo usada.

DL – ponteiro dinâmico, indica histórico de ativação, ou seja, o endereço do segmento de dados da rotina chamadora. Usado quando da desativação da procedure.

RA – endereço de retorno, usado quando da desativação da procedure.

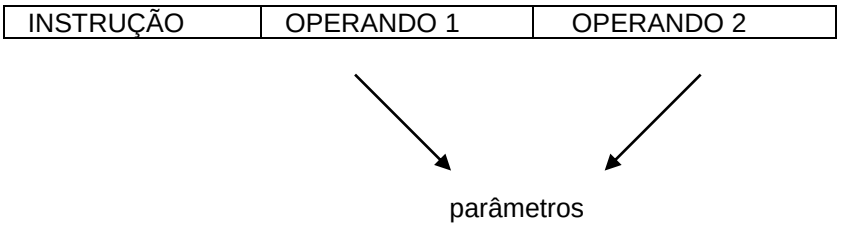
ENDEREÇOS – são obtidos através da fórmula:
BASE + DESLOCAMENTO

VARIÁVEIS:

TOPO - aponta para o topo da pilha (área de dados)
B - indica endereço do segmento de dados (base)
S - é a pilha de execução
P - apontador de instruções
L - primeiro operando
A - segundo operando

INSTRUÇÕES:

Alocadas na área de instruções. São da forma:



Os parâmetros dependem da instrução. Para instruções que referenciam parâmetros na área de dados:

OPERANDO 1: indica a diferença de nível estático (1), ou seja, nível atual – nível onde foi declarado.

OPERANDO 2: indica deslocamento (a).

2. INSTRUÇÕES

A seguir são apresentadas as instruções da máquina hipotética.

1. RETU : retorno de procedure.
2. CRVL : carrega valor na pilha.
3. CRCT : carrega constante na pilha.
4. ARMZ : armazena conteúdo da pilha(topo) no endereço dado.
5. SOMA : operação soma com elementos do topo e sub-topo.
6. SUBT : operação de subtração.
7. MULT : operação de multiplicação.
8. DIVI : operação de divisão.
9. INVR : inverte sinal.
10. NEGA : operação de negação.
11. CONJ : operação AND.
12. DISJ : operação de OR.
13. CMME : compara menor.
14. CMMA : compara maior.
15. CMIG : compara igual.

16. CMDF : compara diferente.
17. CMEI : compara menor igual.
18. CMAI : compara maior igual.
19. DSVS : desviar sempre.
20. DSVF : desviar se falso.
21. LEIT : leitura.
22. IMPR : imprimir topo da pilha.
23. IMPRL : imprimir literal extraído da área de literais.
24. AMEM : alocar espaço na área de dados.
25. CALL : chamada de procedura “a” no nível “l”.
26. PARA : finaliza a execução.
27. NADA : nada faz, continua a execução.
28. COPI : duplica o topo da pilha
29. DSVT : desvia se verdadeiro.

A seguir vamos descrever cada uma das instruções:

1. RETU – Retorno de procedure

```
RETU    --      a   (* a = nº de parâmetros +1*)

P:= S[b+2]
Topo:= b-a
b :=  s  [ b+1]
```

2. CRVL – Carrega valor na pilha

```
CRVL    L       a   (* a = deslocamento; L= diferença de nível*)

topo := topo + 1
s [ topo ]  := s [ BASE ( L ) +  a ]
```

3. CRCT – Carrega constante na pilha

```
CRCT    -   k

topo := topo +1
S [ topo ] := k
```

4. ARMZ – Armazena conteúdo da pilha (topo) no endereço dado

```
ARMZ    L       a   (* a= deslocamento; L= diferença de nível*)

  s [ BASE (L) + a ] := [ topo ]
topo:= topo –1
```

5. SOMA – Operação soma c/ elementos do topo e sub-topo

```
SOMA    -       -

s [ topo –1 ] :=s [ topo –1] + s [ topo ]
topo := topo -1
```

6. SUBT – Operação de subtração

```
SUBT       -       -

 s [ topo  - 1 ]  := s [ topo  - 1 ] – s [ topo ]
 topo := topo - 1
```

7. MULT – Operação de multiplicação

```
 MULT      -         -

 s  [ topo  - 1 ] := s [ topo  - 1 ] * s [ topo ]
 topo := topo  - 1
```

8. DIVI – Operação de divisão inteira

```
DIVI         -          -

se s [ topo ] = 0  então
   escreva ( ´ erro – divisão por zero ´)
fim se
s [ topo  - 1 ]  := s [ topo  - 1 ] div s [ topo ]
topo := topo  - 1
```

9. INVR – Inverte sinal

```
 INVR   -        -

 s [ topo ] := - s [ topo ]
```

10. NEGA – Operação de negação

```
NEGA    -       -

s [ topo ]  := 1  - s [ topo ]
```

11. CONJ – Operação AND

```
CONJ    -       -

se ( s [ topo  - 1 ] = 1 ) e (s [ topo ] = 1) então
    s [ topo – 1 ] := 1
senão
    s [ topo –1 ] := 0
fim se
topo := topo  - 1
```

12. DISJ – Operação de OR

```
 DISJ   -       -

 Se  (s [ topo  - 1 ] =1 ) ou (s [ topo ] =10 então
     s[ topo  - 1] := 1
```

```
    senão
        s [ topo –1 ] :=0
    fim se
    topo := topo  - 1
```

13. CMME – Compara menor

```
CMME   -      -

Se s [ topo  - 1 ] < s [ topo ] então
      s[ topo  - 1] := 1
   senão
      s [ topo –1 ] :=0
   fim se
   topo := topo  - 1
```

14. CMMA – Compara maior

```
CMMA    -      -

Se s [ topo  -1 ] > s [ topo ] então
      s[ topo  - 1] := 1
   senão
      s [ topo –1 ] :=0
   fim se
   topo := topo  - 1
```

15. CMIG – Compara igual

```
CMIG    -       -

Se s [ topo   - 1 ] = s [ topo ] então
      s[ topo  - 1] := 1
   senão
      s [ topo –1 ] :=0
   fim se
   topo := topo  - 1
```

16. CMDF – Compara diferente

```
CMDF   -      -

Se s [ topo –1 ] <>  s  [ topo ] então
      s[ topo  - 1] := 1
   senão
      s [ topo –1 ] :=0
   fim se
   topo := topo  - 1
```

17. CMEI – Compara menor igual

```
CMEI   -      -

Se s [ topo  - 1 ] <= s [ topo ] então
```

```
        s[ topo  - 1] := 1
    senão
        s [ topo –1 ] :=0
    fim se
    topo := topo  - 1
```

18. CMAI – Compara maior igual

```
CMAI   -      -

Se s [ topo –1 ] >= [ topo ] então
        s[ topo  - 1] := 1
    senão
        s [ topo –1 ] :=0
    fim se
    topo := topo  - 1
```

19. DSVS – Desviar sempre

```
DSVS   -     a

p := a
```

20. DSVF – Desviar se falso

```
DSVF  -       a

Se s [ topo ] = 0 então
    p := a
fim se
topo := topo  - 1
```

21. LEIT - Leitura

```
LEIT  -       -

topo := topo + 1
leia s [ topo ]
```

22. IMPR – Imprimir topo da pilha

```
IMPR  -      -

escreva s [ topo ]
topo := topo   - 1
```

23. IMPRL – Imprimir literal extraído da área de literais

```
IMPRL   -    a
```

escreve conteúdo da área da literais utilizando conteúdo do endereço dado *a*

24. AMEM – Alocar espaço na área de dados

    AMEM - a

    t := t + a

25. CALL – Chamada da procedure “a” no nível “L”

    CALL L a (* L= diferença de nível; a= endereço do início da proc. na área de instruções *)

    s [topo + 1] := BASE (L)
    s [topo + 2] := b
    s [topo + 3] := p
    b := t +1
    p := a

26. PARA – Finaliza a execução

    PARA - -

27. NADA – Nada faz, continue execução

    NADA - -

28. COPI – Duplica o topo da pilha

    COPI - -

29. DSVT – Desvia se verdadeiro

    se s [topo] =1 então
    p := a
    fim se
    topo := topo - 1

3. PRINCIPAIS ESTRUTURAS DE GERAÇÃO

A seguir são apresentadas as principais estruturas de geração de código intermediário para a máquina hipotética.

Comando if	**Comando while**
Ex:	**Ex: ……..**
if a> b then c:= a+ b	**While a>f do**
Else c:= a-b;	**Begin**
.....	**soma:= soma +1**;
	Readln(a);
	End; ……
Codigo gerado ➔	**Codigo gerado ➔**
A crvl (a)	**A crvl(a)**
B crvl(b)	**B crvl(f)**
C cmma	**C cmma**
D dsvf (J)	**D dsvf (L)**
E crvl(a)	**E crvl(soma)**
F crvl(b)	**F crct (1)**
G soma	**G soma**
H armz(c)	**H armz (soma)**
I dsvs (N)	**I leit**
J crvl (a)	**J armz (a)**
K crvl (b)	**K dsvs (A)**
L Sub	**L .....**
M Armz (c)	**M .....**
N	

Comando repeat	Comando for
.....	.....
Repeat	For i:= 1 to 10 do
Readln(a);	Begin
Soma:= soma + a;	Soma:= soma + i;
Until a>10;	Writeln(i);
	End; .....
Código gerado →	Código gerado →
A Leit	A crct (1) calculando expressão
B Armz (a)	B armz (i) armaz. Valor inicial
C crvl(soma)	C crct (10) calculando expressão p/ valor limite
D crvl(a)	D copi copiando valor limite
E soma	E crvl (i) carregando valor de i
F armz (soma)	F cmai
G crval(a)	G dsvf (S)
H crct(10)	H crvl(soma)
I cmma	I crvl(i)
J dsvf (A)	J soma
K	K armz(soma)
	L crvl(i)
	M imp
	N crvl(i)
	O crct(1)
	P soma
	Q armz(i)
	R dsvs(D)
	S amem(-1) Limpando área

Comando CASE
……
Case I of
1,2,3 : c:= 30;
4 : c:=20
End;
……..Proxima instrução
Código gerado
A CRVL (I)
B COPI VALOR DE I
C CRCT (1)
D CMIG
E DSVT (N)
F COPI VALOR DE I
G CRCT (2)
H CMIG
I DSVT (N)
J COPI VALOR DE I
K CRCT (3)
I CMIG
M DSVF (Q)
N CRCT (30)
O ARMZ (C)
P DSVS (Y)
Q COPI VALOR DE I
R CRCT (4)
S CMIG
T DSVF (Y)
U CRCT (20)
V ARMZ (C)
X DSVS (Y)
Y AMEM (-1) ARRUMA PILHA DADOS
Z