COMPILADORES

2ª Prova - 06/12/2010 - Prof. Marcus Ramos

1. (2 pontos) Descreva em detalhes o que acontece em cada uma das subfases (i) identificação e (ii) verificação de tipos durante a fase de análise de contexto. Exemplifique.

2. (2 pontos) Descreva e exemplifique:
   a. Em que consiste a alocação de diâmica de variáveis e no que ela difere das alocações estática e automática;
   b. Que tipo de problemas a alocação dinâmica impõe para o sistema de execução;
   c. Que tipo de estratégias são usadas para resolver esses problemas?

3. (2 pontos) Obtenha um padrão de geração de código para a operação de indexação:

   fetch (I[e_1,e_2, ...,e_n]), correspondente à declaração de uma matriz de n dimensões I:

   I: array [min_1..max_1] of array [min_2..max_2] of ... array [min_n..max_n] of t_n;

   Considere dados *size* (t_1), *size* (t_2), ..., *size* (t_n) e também que *address* (I) é representado pelo par d[r]. Construa o seu padrão usando as instruções da linguagem TAM e também outras funções de código que você considere necessárias.

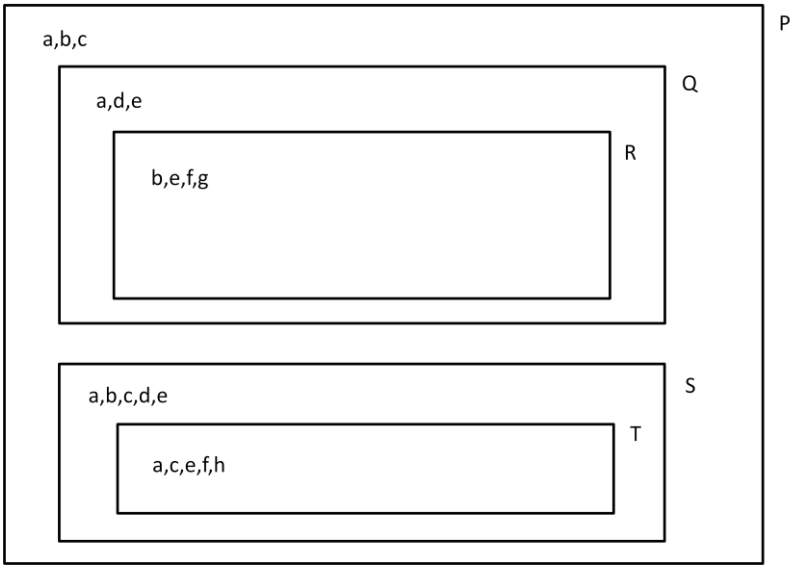
   O padrão deve gerar as instruções que correspondam à avaliação da expressão:

   address (I[e_1,e_2, ...,e_n])= *base* (I)+(e_1-min_1)**size* (t_1)+ ... + (e_n-min_n)**size* (t_n)

   fetch (I[e_1,e_2, ...,e_n])=

```
LOADA d[r]              // d[r] corresponde o endereço da matriz I
evaluate (e1)
LOADL min1
SUB
LOADL size (t1)
MUL
ADD
evaluate (e2)
LOADL min2
SUB
LOADL size (t2)
MUL
ADD
...
evaluate (en)
LOADL minn
SUB
LOADL size (tn)
MUL
ADD
LOADI
```

4. (2 pontos) Considere o programa cuja estrutura de blocos é apresentada a seguir:



Mostre:

- O escopo de cada um dos nomes declarados nesse programa;
- A conteúdo da tabela de símbolos durante a compilação do bloco R;
- A conteúdo da tabela de símbolos durante a compilação do bloco T;
- A situação da pilha de execução considerando o fluxo P→S→T→T→Q→R;
- A situação da pilha de execução considerando o fluxo P→Q→R→Q→R→S→T;

Nos dois últimos casos, identifique e posicione claramente:

- A alocação das variáveis;
- A situação do links estáticos;
- A situação dos links dinâmicos;
- Todos os demais registradores e informações relacionadas.

a_P: P-Q-S
b_P: P-R-S
c_P: P-S
a_Q: Q
d_Q: Q
e_Q: Q-R
b_R: R
e_R: R
f_R: R
g_R: R
a_S: S-T
b_S: S
c_S: S-T
d_S: S
e_S: S-T
a_T: T
c_T: T
e_T: T
f_T: T
h_T: T

a	0	

b	0	
c	0	
a	1	
d	1	
e	1	
b	2	
e	2	
f	2	
g	2	

a	0	
b	0	
c	0	
a	1	
b	1	
c	1	
d	1	
e	1	
a	2	
c	2	
e	2	
f	2	
h	2	

5. (2 pontos) Considere o seguinte programa escrito em Pascal:

```
program P;
     var a,b: integer;
     procedure Q (c: real);
          var d: boolean;
          procedure R (e: integer);
               var f,g: real;
               begin
               while ((f*e+b)<(g+c-a)) or d do
                    if a=f then b:=0 else c:= 1;
               end;
          begin
          …
          end;
     begin
     …
     end;
```

Suponha que P→Q→R. Mostre o código gerado para o corpo do procedimento R (na linguagem TAM) considerando os endereços de cada uma das variáveis envolvidas (par deslocamento / registrador) e os seguintes tamanhos: *size* (endereço)=4, *size* (integer)=2, *size* (real)=4, *size* (boolean)=1.

```
L_001     LOAD       12[LB]     // f
          LOAD       -2[LB]     // e
          MUL
          LOAD       2[SB]      // b
          ADD
          LOAD       16[LB]     // g
          LOAD       -4[L1]     // c
```

```
          ADD
          LOAD       0[SB]      // a
          SUB
          LOAD       12[L1]     // d
          OR
          JUMPIF     (0)   L_002
          LOAD       0[SB]      // a
          LOAD       12[LB]     // f
          EQUAL
          JUMPIF     (0)   L_003
          LOADL 0
          STORE      2[SB]      // b
          JUMP       L_004
L_003     LOADL 1
          STORE      -4 [L1]]   // c
L_004
          JUMP  L_001
L_002
```