COMPILADORES

Prova 2 - 05/04/2018 – Prof. Marcus Ramos

Questão 1 (1,5 ponto): Considere a declaração Pascal abaixo. Se o endereço real de `m` é 1000 e size[integer]=2, qual é a origem virtual de `m`?

```
var m: array [3..5,-2..4,-8..-3] of integer;
```

```
address[m[i,j,k]]=1000+(i-3)*(4+2+1)*(-3+8+1)*2+(j+2)*(-
3+8+1)*2+(k+8)*2
address[m[i,j,k]]=1000+(i-3)*7*6*2+(j+2)*6*2+(k+8)*2
address[m[i,j,k]]=1000+(i-3)*84+(j+2)*12+(k+8)*2
address[m[i,j,k]]=1000+84*i-252+12*j+24+2*k+16
address[m[i,j,k]]=1000-252+24+16+84*i+12*j+2*k
address[m[i,j,k]]=788+84*i+12*j+2*k
Resposta: 788.
```

Considere o seguinte programa Pascal:

```
program p;
      var a,b: integer;
      procedure q (a,c: integer);
            var d,e: integer;
            procedure r (f,g: integer);
                  var h,i: integer;
                  begin (* r *)
                  b:=c+e+g+i;
                  end (* r *)
            begin (* q *)
            b:=a+c+d+e;
            end (* q *)
      begin (* p *)
      a:=a+b;
      end (* p *).
```

Questão 2 (1,5 ponto): Qual o escopo de cada nome definido no programa acima?

```
p:    p
aP:   p - q
b:    p
q:    p
aq:   q
c:    q
d:    q
e:    q
r:    q
f:    r
g:    r
h:    r
i:    r
```

Questão 3 (2,0 ponto): Mostre o código gerado para os três comandos de atribuição, destacando o endereço completo (deslocamento/registrador) de cada variável e parâmetro.

```
b:=c+e+g+i;
LOAD        -1[L1]
LOAD        4[L1]
CALL        ADD
LOAD        -1[LB]
CALL        ADD
LOAD        4[LB]
CALL        ADD
STORE       1[SB]

b:=a+c+d+e;
LOAD        -2[LB]
LOAD        -1[LB]
CALL        ADD
LOAD        3[LB]
CALL        ADD
LOAD        4[LB]
CALL        ADD
STORE       1[SB]

a:=a+b;
LOAD        0[SB]
LOAD        1[SB]
CALL        ADD
STORE       0[SB]
```

Questão 4 (2,0 ponto): Considere o fluxo de execução p→q→r→q→r e mostre a situação da pilha de execução, com todos os frames, LB, links estáticos e dinâmicos, variáveis e parâmetros alocados. Considere que o tipo integer e os endereços ocupam uma única posição de memória.

```
SB →        a
            b
            a
            c
(1) →       LE (SB)
            LD (SB)
            ER
            d
            e
            f
            g
(2) →       LE (1)
            LD (1)
            ER
            h
            i
            a
            c
(3) →       LE (SB)
            LD (2)
            ER
```

```
            d
            e
            f
            g
LB →        LE (3)
            LD (3)
            ER
            h
ST →        i
```

Questão 5 (1,5 ponto): O operador “expressão condicional” da linguagem C (“?:”) possui três operandos (caso geral: “<exp1>?<exp2>:<exp3>”; exemplo: “i>j?i+1:j*2”) e a seguinte semântica: (i) a primeira expressão (<exp1>), do tipo booleano, é avaliada; (ii) se o valor de <exp1> for verdadeiro, então o valor da expressão condicional corresponde ao valor de <exp2>; caso contrário, corresponde ao valor de <exp3>. Obtenha um template de código para o operador expressão condicional da linguagem C, considerando as funções de códigos estudadas em sala de aula e a linguagem-objeto TAM.

```
evaluate [<exp1>?<exp2>:<exp3>] =
                evaluate [<exp1>]
                JUMPIF (0) L_001
                evaluate [<exp2>]
                JUMP L_002
     L_001      evaluate [<exp3>]
     L_002
```

Questão 6 (1,5 ponto): Justifique o fato de a subfase de identificação da análise de contexto ser o caminho crítico no tempo de execução de um compilador, possuindo tempo $O(n^2)$ quando é feita uma busca linear.

Se n é uma medida do comprimento do programa-fonte, então podemos considerar que n corresponde ao número de identificadores declarados no mesmo. Numa busca linear, os mesmos são armazenados numa tabela ou vetor, sendo que para cada identificador o tempo médio de busca é de n/2. Assim, para localizar n identificadores são necessárias n*n/2 buscas e o tempo de execução é $O(n^2)$. No entanto, é possível reduzir este tempo para algo próximo de O(n), o que contribui diretamente para a redução do tempo total de execução do compilador.