Construção de Compiladores 1 - 2015.1 - Prof. Daniel Lucrédio
Lista 09 - Geração de código e otimização

1. Cite quais são os 2 tipos de código intermediário apresentados em aula e suas características principais. Quais são as diferenças entre eles?
2. Apresente a sequência de instruções de código de três endereços correspondente a cada uma das expressões aritméticas a seguir. Quais são as árvores sintáticas abstratas que correspondem à geração de código?
   a) 2+3+4+5
   b) 2+(3+(4+5))
   c) a*b+a*b*c
3. Apresente a sequência de instruções de P-código correspondente às expressões aritméticas do exercício anterior
4. Escreva a gramática de atributos para geração de código de três endereços para a gramática de expressões aritméticas de inteiros a seguir. Utilizando a gramática resultante, gere o código de três endereços para todas as expressões da questão 2

```
exp → exp soma termo | termo
soma → + | -
termo → termo mult fator | fator
mult → *
fator → ( exp ) | num | id
```

5. Considerando-se a mesma gramática do exercício anterior, escreva a gramática de atributos para a geração de P-código. Utilizando a gramática resultante, gere o P-código para todas as expressões da questão 2
6. Apresente as instruções de três endereços correspondentes às expressões em C a seguir.
   a) (x=y=2)+3*(x=4)
   b) a[a[i]]=b[i=2]
7. Apresente a sequência de instruções de P-código correspondente às expressões em C do exercício anterior
8. Cite as várias fontes de otimização apresentadas em aula explicando o que vem a ser cada uma delas
9. Quais os tipos de otimização (pequena escala, local, global ou interprocedimento) empregados nos exemplos a seguir:

a)

```
x = y * (-(-1))
```

```
x = y
```

b)

```
t1 = a + 2
t2 = t1
t3 = b + c
t4 = a + 2
t5 = t4
```

```
t5 = a + 2
t3 = b + c
```

c)

```
while(c<n*n-2+3.14) {
   c = c-1;
   cout << "laço "+c;
}
```

```
t1 = n*n-2+3.14;
while(c<t1);
   c = c-1;
   cout << "laço"+c;
}
```

d)

```
int soma(int a,int b) {
```

```
...
```

```
   return a+b;
}
...
x = 2;
y = 3;
c = soma(x,y);
cout << c;
...
```

```
      x = 2;
      y = 3;
      t1 = x;
t2 = y;
t3 = t1+t2;
      c = t3;
      cout << c;
      ...
```

10. As otimizações a seguir são válidas? Justifique.

a)

```
while(c<n+1) {
   c = c-1;
   n = n+5;
   cout << "laço "+c;
}
```

```
t1 = n+1;
while(c<t1);
      c = c-1;
n = n+5;
      cout << "laço"+c;
}
```

b)

```
void funcao(int a) {
   if(a==0)
      print("Bum!");
   else {
      print("Faltam "+a+" secs");
      sleep(1000);
      funcao(a-1);
      print("Passaram-se "+a+" secs");
   }
}
```

```
void funcao(int a) {
   L:
   if(a==0)
      print("Bum!");
   else
      print("Faltam "+a+" secs");
      sleep(1000);
      a = a-1;
      goto L;
           print("Passaram-se  "+a+"
secs");
}
```

c)

```
void f() {
   int i, j;
   for (i=0; i < 10; i++)
      cout << i << endl;
   for (j=10; j < 0; j--)
      cout << j << endl;
   cout << i << j << endl;
}
```

```
void f() {
   int i;
   for (i=0; i < 10; i++)
      cout << i << endl;
   for (i=10; i < 0; i--)
      cout << i << endl;
   cout << i << i << endl;
}
```