Compiladores — Folha laboratorial 5

DCC/FCUP

Novembro 2024

Geração de código intermédio

Exercício 1

Assumindo atribuições de variáveis temporárias apropriadas, traduza as seguintes instruções de código C para código de 3 endereços seguindo o esquema de tradução apresentado nas aulas.

(a) `y = 1+x+3*x*x;`

(b) `z = 3+f(x+y,x*y);`

(c) `if(x<0) x = -1*x;`

(d) `y = (x == 1) || (x == 2);`

(e) `if (x<=y && !(x==y || x==1)) x=3; else x=5;`

Exercício 2

Traduza as definições de funções para código intermédio de 3 endereços.

(a)
```
int fact(int n) {
    int i = 1;
    int r = 1;
    while(i <= n) {
        r = r*i;
        i = i+1;
    }
    return r;
}
```

(b)
```
int fact(int n) {
    if (n > 1)
        return n * fact(n-1);
    else
        return 1;
}
```

Exercício 3

As leis de De Morgan dizem que `!(p || q)` é equivalente a `(!p) && (!q)`. Mostre que o esquema de tradução *transCond* apresentado nas aulas gera código equivalente para ambas as condições.

Exercício 4

(a) Considere a seguinte tradução dum ciclo *do/while* (em que a condição é testada no final)

```
i = 0;
do {
  i = i + 1;
} while( i< 10 );
```

para o seguinte código intermédio:

```
i := 0
LABEL loop
i := i + 1
COND i < 10 loop end
LABEL end
```

Escreva uma regra de tradução geral para este tipo de ciclos.

(b) Considere a seguinte alternativa para traduzir um ciclo *while*

```
i = 0;
while (i < 10) {
   i = i + 1;
}
```

para o seguinte código intermédio:

```
i := 0
JUMP cond
LABEL loop
i := i + 1
LABEL cond
COND i < 10 loop end
LABEL end
```

Esta tradução gera código que efetua um salto incondicional fora do ciclo e depois apenas um salto condicional por cada iteração; assim, deverá ser mais eficiente do que a tradução apresentada nas aulas (que gera dois saltos por cada iteração).

Escreva uma regra geral para tradução de ciclos *while* desta forma.