Intermediate Code Generation

May 7, 2022

1 Exercises

1. Use the translation functions given in the course slides (Intermediate Code Generation) to generate three address code for the expression

   $$2 + g(x + y, x * y)$$

   . The result should be put in the intermediate code variable r.

2. (a) Add to the command compilation function of the course slides a new case to compile a *repeat until* command.

   (b) Using this new compilation function generate three address code for the command:

   ```
   x := 2+y;
   if x<y then x:=x+y;
   repeat
     y:=y*2;
     while x>10 do x:=x/2
   until x<y
   ```

3. Translate to three-address code the following statement:

   ```
   if x<=y && !(x==y || x==1)
   then x:=3
   else x:=5
   ```

4. De Morgan‘s law tell us that

   ```
   !(p || q)
   ```

   is equivalent to

```
(!p) && (!q)
```

Show that these may generate identical three address code when compiled.

5. Make patterns for the same subset of the intermediate language covered in the theoretical lecture about Code Generation but, instead on MIPS, use RISC V as the target instruction set (a description of RISC-V instruction set can be found in the theoretical lecture notes about Register Allocation-part 1). Use this to translate the intermediate-language instruction sequence:

```
t2 := t1 + 116
t3 := M[t2]
```