Exercise Sheet

2022.11.08

1. 请实现 WhileDB 语言简单编译器的拆分复合表达式的功能。拆分复合表达式需要将语法分析器生成的抽象语法树转化为每个赋值语句至多进行一步计算的 L1 中间语言（参考附件中的 L1.h）。请实现一个函数，该函数接收指向抽象语法树根节点的指针，并生成转化后的 L1_cmd_listbox：

   ```
   struct L1_cmd_listbox * TAC_gen(struct cmd * c);
   ```

   提交时请在文件夹中放置本次编程作业的所有源代码、Makefile 以及必要的说明文件。在 Canvas 系统上提交时，请将所有需要提交的文件夹用 zip 压缩后提交。这个编程任务的具体要求如下：

   - 复合表达式需要被拆分为一系列顺序执行的赋值语句 L1_cmd_listbox 和一个新的简单表达式 L1_expr；
   - 对于 if 语句，拆分复合表达式时需要将原先的条件拆分得到的顺序执行赋值语句加入 L1_cmd 列表中，生成的 if 语句使用新的简单表达式作为条件。例如：

     ```
     x = read_int();
     y = read_int();
     if ((x + y) * 2 > 24)
     then { ... } else { ... }
     ```

     对应的拆分结果为：

     ```
     #0 = read_int()
     #1 = read_int()
     #2 = PLUS(#0, #1)
     #3 = MUL(#2, $(2))
     if (GT(#3, $(24)))
     then { ... } else { ... }
     ```

   - While 语句应当被拆分成计算条件的顺序执行赋值语句（可能为空），新循环条件和循环体。例如：

     ```
     while (x * x + x > 16)do {...}
     ```

     拆分结果为：

     ```
     do
       {
         #1 = MUL(#0, #0)
         #2 = PLUS(#1, #0)
       }
       while (GT(#2, $(16))) do
       {...}
     ```

- 二元操作符 && 和 || 需要短路求值。例如：

```
if (x || y && * y) then { ... } else { ... }
```

拆分结果为：

```
#0 = read_int()
#1 = read_int()
if (#0) then {
  #3 = $(1)
}
else {
  if (#1)
  then { #2 = DEREF(#1) }
  else { #2 = $(0) }
  #3 = #2
}
if (#3)
then { ... } else { ... }
```

- 请注意：拆分后应当尽量避免赋值语句中的表达式为单个辅助变量。例如：

```
x = y + z + 1
```

拆分结果为：

```
#3 = #1 + #2
#0 = #3 + 1
```

而最好不要生成：

```
#3 = #1 + #2
#4 = #3 + 1
#0 = #4
```

更多样例请参考附件中的 samples。

2. 请实现 WhileDB 语言简单编译器的生成基本块的功能。生成的基本块最终存储在数据结构 L1_basic_blocks 中（参考附件中的 L1.h），该结构体的域 l 为一个 L1_basic_block 数组的头指针，order 为基本块编号数组的头指针，n 为基本块个数（数组长度）。附件中的库函数提供了 print_L1_basic_blocks 函数，可以按照 order 域的顺序而不是基本块的编号输出所有基本快。请实现一个函数：

```
struct L1_basic_blocks * BB_gen(struct L1_cmd_listbox * cs);
```

该函数接收指向 L1_cmd_listbox 的指针，并生成转化后的 L1_basic_blocks 的指针。基本块内不能有 if 和 while 语句，每个基本块应当包含跳转信息。请注意：应当避免生成空的无条件跳转基本块，例如：

```
var x;
x = read_int();
if (x > 0)
then {
  while (x > 0) do {
    x = x - 1
  }
}
else {
  x = 0
}
```

生成的基本块为：

```
Block 0:
  #0 = read_int()
  if (GT(#0, $(0))) then jmp 1 else jmp 2
Block 1:
  if (GT(#0, $(0))) then jmp 3 else jmp 4
Block 3:
  #0 = MINUS(#0, $(1))
  jmp 1
Block 2:
  #0 = $(0)
  jmp 4
```

而最好不要生成：

```
Block 0:
  #0 = read_int()
  if (GT(#0, $(0))) then jmp 1 else jmp 2
Block 1:
  jmp 3
Block 3:
  if (GT(#0, $(0))) then jmp 4 else jmp 5
Block 4:
  #0 = MINUS(#0, $(1))
  jmp 3
Block 2:
  #0 = $(0)
  jmp 5
```

因为其中 1 号基本块是空的无条件跳转基本块。