编译实验LAB6：循环

这次的实验在前面的基础上实现起来非常简单，1个小时就做完了，所以简单描述一下。

首先修改ANTLR的规则文件，添加stmt节点对while、break、continue语句的识别。

循环

循环本质上也是Block之间的跳转，实现起来和IF语句基本一样，只是跳转方式不同。

每个循环的指令分为三部分，通过下面三个label表示：

1. 条件判断label
2. 内部语句label
3. 结束退出label

进入循环时首先跳转到【条件判断label】，如果条件为真，跳转到【内部语句label】，如果为假，跳转到【结束退出label】。在跳转到【内部语句label】后执行循环体内语句，执行结束后无条件跳转到【条件判断label】。

Continue与Break

本质上只是各自对应一个无条件跳转语句罢了。

- Continue对应跳转到【条件判断label】
- Break对应跳转到【结束退出label】

通过一个栈来保存当前层循环的【条件判断label】和【结束退出label】分别是多少，遇到Continue/Break时直接取栈顶元素即可。这个栈在循环语句分析开始时压栈，结束时弹栈，就可以维护。

```
private Stack<Pair<String,String>>loopLabels = new Stack<>();
```

```
else if(ctx.WHILE() != null) { // While
            String while_reg = "%r" + regId++;
```

```
            String do_reg = "%r" + regId++;
            String out_reg = "%r" + regId++;
            loopLabels.push(new Pair<>(while_reg, out_reg));
            System.out.println("    br label " + while_reg);

            // 条件判断部分
            System.out.println(while_reg.substring(1) + ":");
            String cond_reg = visit(ctx.cond());
            System.out.println("    br i1 " + cond_reg + ",
label " + do_reg + ", label " + out_reg);

            // 循环体部分
            System.out.println(do_reg.substring(1) + ":");
            visit(ctx.stmt(0));
            System.out.println("    br label " + while_reg);

            // 结束部分
            System.out.println(out_reg.substring(1) + ":");
            loopLabels.pop();
        }
        else if(ctx.BREAK() != null) {
            System.out.println("    br label " +
loopLabels.peek().getValue()); // 跳转到退出
        }
        else if(ctx.CONTINUE() != null) {
            System.out.println("    br label " +
loopLabels.peek().getKey()); // 跳转到条件判断
        }
```