王建楠 2019013296

step2

实验内容

词法语法分析，引入了 NotExpr 和 BitNotExpr。

中间代码生成，补全 Visitor 模式中其对应的操作。

目标代码生成，增加了 not 和 seqz 两个汇编指令。

思考题

`-~2147483647`

step3

实验内容

词法语法分析，引入了减、乘、除、取模三种运算表达式的分析。

中间代码生成，补全 Visitor 模式中其对应的操作。

目标代码生成，增加了 sub、mul、div、rem 四个汇编指令。

思考题

```c
#include <stdio.h>

int main() {
  int a = -2147483648;
  int b = -1;
  printf("%d\n", a / b);
  return 0;
}
```

x86_64

`floating point exception (core dumped)`

RISCV-32

`-2147483648`

step4

实验内容

词法语法分析，引入了 !=、==、<=、<、>=、>、&&、|| 八种运算表达式的分析。

由于大于等于和小于等于、大于和小于，可以分别归成一类，所以新增了 EquExpr、NeqExpr、AndExpr、AndExpr、OrExpr、GeqExpr（大于等于）、GrtExpr（大于）六个语法树结点

中间代码生成，依照 Visitor 模式，补全了其对应的操作。

目标代码生成，增加了 snez、and、or、slt、sltu、sgt 四个汇编指令。

a==b

```
c=a-b
d=isZero(c)
```

故而可以用 sub 和 seqz。

a!=b

```
c=a-b
d=notZero(c)
```

故而可以用 sub 和 snez。

a>=b

```
c=a<b
d=isZero(c)
```

故而可以用 slt 和 seqz

思考题

因为短路求值可以帮助我们节省不必要的代码/程序运行时间。

`A && B`：当 A 为假时，直接返回假，不对 B 进行判断。

`A || B`：当 A 为真时，直接返回真，不对 B 进行判断。

这样在我们代码运行时，相较于 `&`，`|` 这类逻辑运算符，可以为我们省去某些时候对 B 的不必要判断。