Stage-1 实验报告

沙之洲 2020012408

Step2

实验内容

补全了单元运算符中向 TAC 转换的部分，具体来说，增加了单元运算符 Not 和 BitNot 中向 tacop 的 not 和 seqz 的转换

思考题

`-~2147483647`

Step3

实验内容

补全了双目运算符中向 TAC 转换的部分。具体来说，将双目运算符 Sub, Mul, Div, Mod 分别映射到 TAC 中的 SUB, MUL, DIV, REM 上

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`

RISCV32 结果

`-2147483648`

Step4

实验内容

在之前的双目运算上增加了 <, <=, >, >=, !=, ==, &&, ||

其中只有 小于 和 大于 能够直接对应到 TAC 中对应的指令上

剩下的双目运算符都需要多条汇编指令才能完成，因此，在这一个 step 中，我们还修改了 riscvasmemitter.py 中对应的 visitor 函数，将上述需要多条汇编指令实现的双目运算符顺序对应到汇编指令上。

例如

`a == b`

```
sub c, a, b
seqz c, c
```

`a != b`

```
sub c, a, b
snez c, c
```

`a <= b`

```
sgt c, a, b
snez c, c
```

`a >= b`

```
slt c, a, b
snez c, c
```

思考题

短路特性指的是

- 当 A && B 并且已知 A = False 的情况下，就不需要计算 B，而直接返回 False
- 当 A || B 并且已知 A = True 的情况下，不计算 B，直接返回 True

短路特性可以减少程序的计算量，因为当计算结果已经是肯定的时候，就没必要进行后边冗余的计算了。因为在实际的程序中，可能会出现调用复杂函数来判断逻辑 True 和逻辑 False

短路特性可以减少复杂函数的调用。而在程序员实际工作中，可以利用短路特性，将计算量小的放在前边，将需要复杂计算的部分放在后边。这样在程序实际运行的过程中，可以提高实际运行效率。