PA5 report

基本流程

我实现了比较简单的基于图着色的寄存器分配算法。流程为：先给每个变量分配一个节点，再构建干涉图，对干涉图着色，最后把指令中的虚拟寄存器替换为着色得到的物理寄存器。

我的算法中每个变量对应一个干涉图中的节点，连边方法为对于每次定值，把该定值变量和在紧贴该定值点后处的所有活跃变量之间连一条边。

```
class Main {
    static void main() {
        int a0 = 0;
        int a1 = 0;
        int a2 = 0;
        int a3 = 0;
        int a4 = 0;
        int a5 = 0;
        int a6 = 0;
        int a7 = 0;
        int a8 = 0;
        int a9 = 0;
        int a10 = 0;
        int a11 = 0;
        int a12 = 0;
        int a13 = 0;
        int a14 = 0;
        int a15 = 0;
        int a16 = 0;
        int a17 = 0;
        int a18 = 0;
        int a19 = 0;
        int a20 = 0;
        int a21 = 0;
        int a22 = 0;
        int a23 = 0;
        int a24 = 0;
        int a25 = 0;
        int a26 = 0;
        int a27 = 0;
        int a28 = 0;
        int a29 = 0;
        int a30 = 0;
        int a31 = 0;
        int a32 = 0;
        int a33 = 0;
        int a34 = 0;
        int a35 = 0;
        int a36 = 0;
        int a37 = 0;
        int a38 = 0;
        int a39 = 0;
        int a40 = 0;
```

```
        int a41 = 0;
        int a42 = 0;
        int a43 = 0;
        int a44 = 0;
        int a45 = 0;
        int a46 = 0;
        int a47 = 0;
        int a48 = 0;
        int a49 = 0;
        while(true) Print(a0);
        while(true) Print(a1);
        while(true) Print(a2);
        while(true) Print(a3);
        while(true) Print(a4);
        while(true) Print(a5);
        while(true) Print(a6);
        while(true) Print(a7);
        while(true) Print(a8);
        while(true) Print(a9);
        while(true) Print(a10);
        while(true) Print(a11);
        while(true) Print(a12);
        while(true) Print(a13);
        while(true) Print(a14);
        while(true) Print(a15);
        while(true) Print(a16);
        while(true) Print(a17);
        while(true) Print(a18);
        while(true) Print(a19);
        while(true) Print(a20);
        while(true) Print(a21);
        while(true) Print(a22);
        while(true) Print(a23);
        while(true) Print(a24);
        while(true) Print(a25);
        while(true) Print(a26);
        while(true) Print(a27);
        while(true) Print(a28);
        while(true) Print(a29);
        while(true) Print(a30);
        while(true) Print(a31);
        while(true) Print(a32);
        while(true) Print(a33);
        while(true) Print(a34);
        while(true) Print(a35);
        while(true) Print(a36);
        while(true) Print(a37);
        while(true) Print(a38);
        while(true) Print(a39);
        while(true) Print(a40);
        while(true) Print(a41);
        while(true) Print(a42);
        while(true) Print(a43);
        while(true) Print(a44);
        while(true) Print(a45);
        while(true) Print(a46);
        while(true) Print(a47);
        while(true) Print(a48);
```

```
        while(true) Print(a49);
    }
}
```

对于上面这段代码，由于我们的算法没有处理溢出的情况，所以无法正常分配寄存器，但是贪心寄存器分配算法会在无法分配寄存器时把一些变量溢出到栈上，因此原来的贪心寄存器分配算法的效果较好。