

Lab5:Riddle

PB20000248 郑子涵

实验要求

- 用汇编语言重写以下C++代码

```
int judge(int r0) {  
    int i = 2;  
    r1 = 1;  
    while (i * i <= r0) {  
        if (r0 % i == 0) {  
            r1 = 0;  
            break;  
        }  
        i++;  
    }  
    return r1;  
}
```

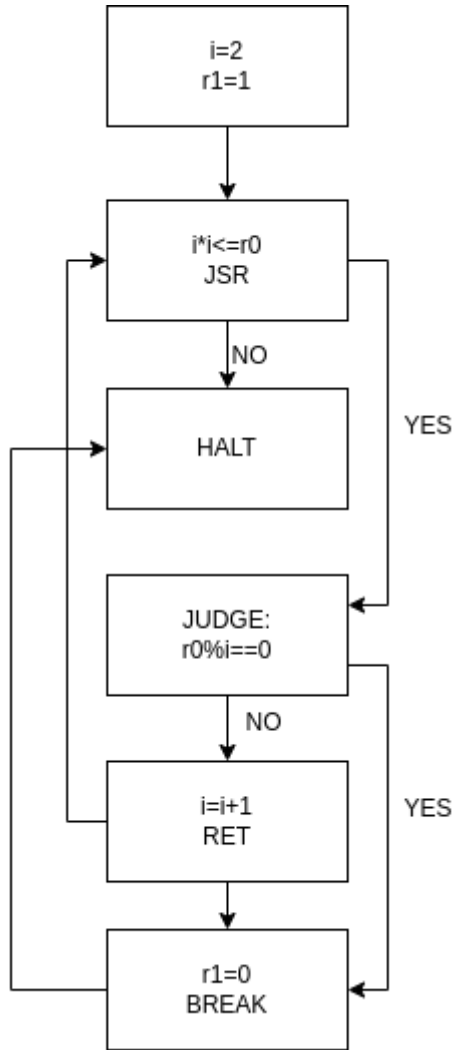
- $r0(int, 0 \leq r0 \leq 10000)$, $r0$ 事先已经给出, 把结果储存在 $r1$ 即可。

实验过程

实验思路

把C++代码while中的语句块看成是一个子程序，每次满足 $i * i \leq r0$ 时都跳转到JUDGE子程序，然后通过JUDGE子程序来判断是Break，还是继续判断是否满足while循环。

具体流程图如下所示：



代码实现

```
.ORIG    x3000
1. NOT   R5, R0
2. ADD   R5, R5, #1
3. AND   R2, R2, #0
4. AND   R3, R3, #0
5. AND   R1, R1, #0
6. ADD   R1, R1, #1
7. ADD   R2, R2, #2
8. POSE3 ADD   R3, R2, #0
9. AND   R4, R4, #0
10. POSE1 ADD   R4, R4, R2
11. ADD  R3, R3, #-1
12. BRnp POSE1
13. ADD  R4, R5, R4
14. BRp  POSE5
15. JSR  JUDGE
16. POSE5 HALT
17. JUDGE ADD   R4, R0, #0
18. NOT  R3, R2
19. ADD  R3, R3, #1
20. POSE2 ADD   R4, R4, R3
21. BRp  POSE2
```

```

22. BRn POSE4
23. AND R1, R1, #0
24. RET
25. POSE4 ADD R2, R2, #1
26. BRnzp POSE3
.END

```

代码分析

- 第1到第7行是对数据的初始化赋值
- 第8到第12行是计算 $i * i$ 并储存到R4
- 第13到第16行是计算 $i * i - r0$ 判断是否要执行子程序，相当于while语句
- 第17到第26行则是子程序部分，判断 $r0 \% i == 0$,分别执行RET或则继续判断while循环

代码优化

将代码的JSR语句舍去，统一换成BR语句会少一行代码，指令的执行条数也会少一些，但是优化不是很明显。

```

.ORIG x3000
1. NOT R5, R0
2. ADD R5, R5, #1
3. AND R2, R2, #0
4. AND R3, R3, #0
5. AND R1, R1, #0
6. ADD R1, R1, #1
7. ADD R2, R2, #2
8. POSE3 ADD R3, R2, #0
9. AND R4, R4, #0
10. POSE1 ADD R4, R4, R2
11. ADD R3, R3, #-1
12. BRnp POSE1
13. ADD R4, R5, R4
14. BRnz JUDGE
15. POSE5 HALT
16. JUDGE ADD R4, R0, #0
17. NOT R3, R2
18. ADD R3, R3, #1
19. POSE2 ADD R4, R4, R3
20. BRp POSE2
21. BRz POSE4
22. ADD R2, R2, #1
23. BRnzp POSE3
24. POSE4 AND R1, R1, #0
25. BRnzp POSE5
.END

```

实验总结

- 该C++程序可以看成是一个无传参的子程序的调用，也可以看成是一个普通的循环条件跳转程序，实现的差别并不是很大。
- 初始程序对于随机测试样本的执行指令数为：10356
- 优化过后的程序对于随机测试样本的执行指令数为：10344

