lab1实验报告

PB17030889

吉志远

代码说明

程序代码

在群里讨论~~助教透露~~了仅用移位算符来求gcd的方法之后，弃掉了原先~~只用了10行~~的更相减损法，Google得stein算法：

先用很接近汇编语言的c++描述为：

```
//stein_gcd.c
#include<stdio.h>
int main(){
    int a=10;
    int b=45;
    int acc=0;
    while((a&1)==0&&(b&1)==0){
        acc++;
        a=a>>1;
        b=b>>1;

    }
    while((a&1)==0)a=a>>1;
    while((b&1)==0)b=b>>1;
    if(a<b){
        int t=a;
        a=b;
        b=t;

    }
    while((a!=b)){
        a-=b;
```

```
        if(a<b){int t=a;a=b;b=t;}

    }
    printf("%d",b<<acc);
    return 0;
}
```

然后由于lc3并没有提供基本的移位运算符，但是可以通过打表的方法来生成

由于全部都打(2^15)超出了内存总数，所以使用高八位和低八位的掩码方法 LOWMASK: x01ff

HIGHMASK: xff00

让待右移的数字和LMASK求与，得出结果，再通过这个结果作为lut的地址，查找提前打好表的右移结果，

同理，求出高八位的右移结果，相加，为右移结果

用lc3的asm实现为：

```
.ORIG x3000
    AND R5,R5,#0    ;clear r5 as acc
    AND R3,R3,#0    ;flag
;part0
;check for vaild input
    AND R0,R0,R0
    BRnz END
    AND R1,R1,R1
    BRnz END
;part1 把r1化简至不能整除2
TESTRb  AND R2,R1,#1
    BRnp TESTRa
    LD  R6,LMASK
    AND R2,R1,R6    ;R2为R1低9位的值
    LD  R6,ADRES
    ADD R2,R6,R2    ;R2为lut中R1低8位右移动对应值地址
    LDR R4,R2,#0    ;根据结果读取右移结果
    LD  R6,HMASK
    AND R2,R1,R6
    LD  R6,ADRES
    ADD R2,R6,R2    ;R2为lut中R1低8位右移动对应值地址
    LDR R2,R2,#0
    BRz THRZERO     ;三个0相加还是0
    ADD R2,R2,R2
    ADD R2,R2,R2
    ADD R2,R2,R2    ;提高8位
    ADD R4,R4,R2    ;得到右移结果
THRZERO ADD R1,R4,#0    ;R1=R4
    NOT R3,R3   ;flag=1
;part2 把r0化简至不能整除2
;由于是顺序执行，若r0，r1均被右移一位则递增c
TESTRa  AND R2,R0,#1
    BRnp ENDLOOP
    LD R6,LMASK
    AND R2,R0,R6
```

```
    LD R6,ADRES
    ADD R2,R6,R2
    LDR R4,R2,#0
    LD R6,HMASK
    AND R2,R0,R6
    LD R6,ADRES
    ADD R2,R6,R2
    LDR R2,R2,#0
    BRz ZERO
    ADD R2,R2,R2
    ADD R2,R2,R2
    ADD R2,R2,R2
    ADD R4,R4,R2
ZERO    ADD R0,R4,#0    ;R0=R4
    AND R3,R3,R3    ;根据R3判断是否需要递增acc
    BRz SKIP    ;R3为0,skip,R3为1111(负数)递增
    ADD R5,R5,#1
    AND R3,R3,#0    ;递增之后clear acc
    ;如果R1奇数,此时R3为0不会递增
SKIP
    BRnzp TESTRb    ;继续右移动R0,R1
ENDLOOP
LOOP
    ;part3
    NOT R6,R1;
    ADD R6,R6,#1;
    ADD R0,R0,R6    ;r0=r0-r1
    BRp LOOP
    BRz DONE
    ADD R2,R0,R1    ;temp=a+b
    ADD R0,R1,#0
    ADD R1,R2,#0    ;change a,b
    BRnzp LOOP

DONE    ADD R0,R1,#0
LSHFT   ADD R5,R5,#-1
    BRn END
    ADD R0,R0,R0
    BRnzp LSHFT
END HALT

ADRES .FILL x3100
LMASK .FILL x01ff
HMASK .FILL xff00

.END
```

查找表(lut)生成

```
#include<iostream>
int main (){
        for(int i = 0;i<512;i++){
                printf("%x\n",i>>1);
        }
        return 0;
}
```

利用lc3convert转化为obj

```
./a.out > lut
./lc3convert -b16 ./lut
```

复杂度分析

时间复杂度：由于stein算法为依次除2所以为logn型

空间复杂度：使用了512个内存单元+7个寄存器

无效输入处理

让R0，R1分别对自己求与，若结果为非正数则直接HALT

代码：

```
;part0
;check for vaild input
    AND R0,R0,R0
    BRnz END
    AND R1,R1,R1
    BRnz END
```

算法测试

使用shell的$RANDOM环境变量来生成随机数，然后写入tb文件，执行

> $ lc3sim -s tb

来作为测试文件，并用python检测输出是否正确

```
#! /bin/bash
# 输入lut表格, 新编辑测试文件故覆盖写入
echo "file rshiftlut" > tb
# 打开rshift.obj
echo "file rshift" >> tb
# 随机生成R0
R0=$((RANDOM%32768))
echo "register R0 #$R0" >> tb
# 随机生成R1
R1=$((RANDOM%32768))
echo "register R1 #$R1" >> tb
```

```
# 在303a(HALT)处设置断点
echo "break set 303a" >> tb
# 执行
echo "finish" >> tb
# END
echo "quit" >> tb
echo "R0:$R0,R1:$R1"
./lc3sim -s tb | tail -4
python gcd.py $R0 $R1
```

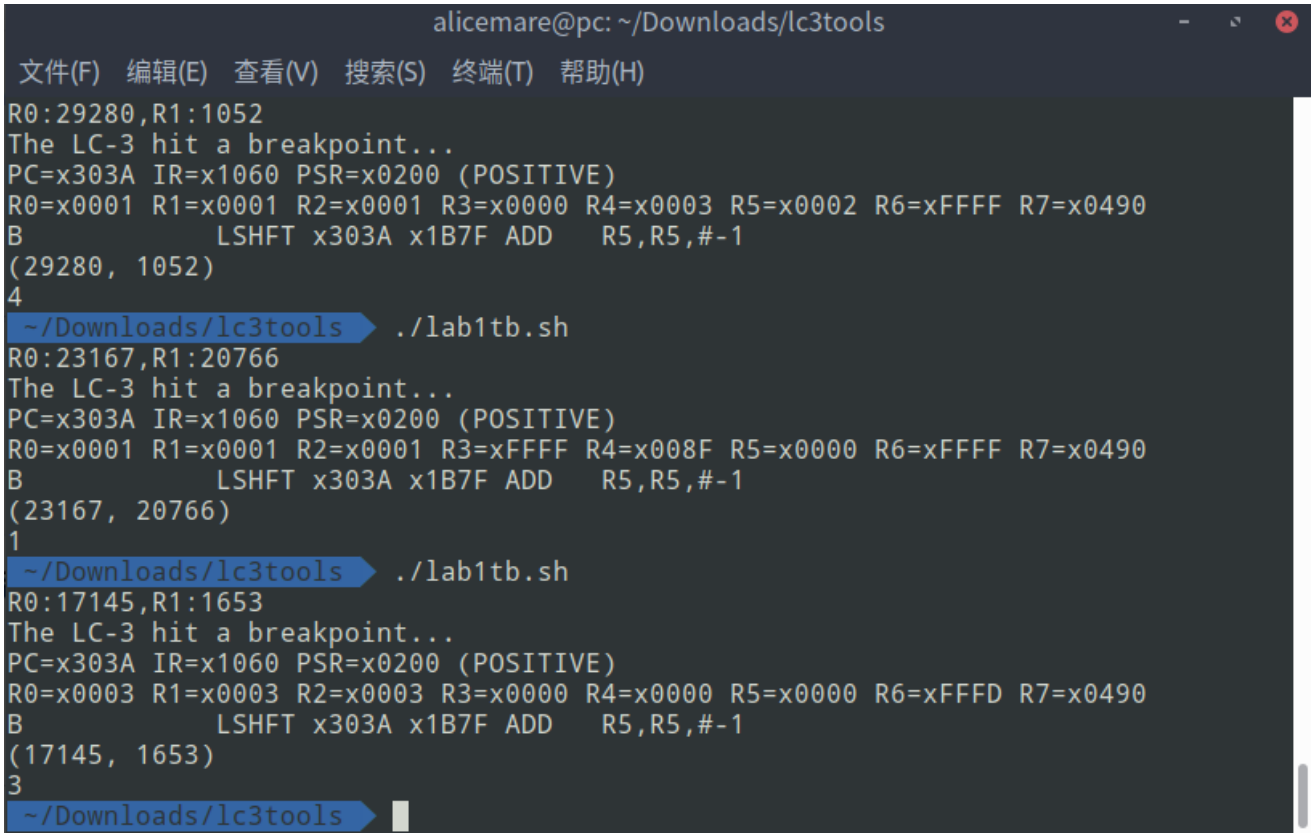
python求gcd文件:

```
#! /bin/python3
from sys import argv
def gcdofint(a,b):
    if (b==0):
        return a
    else:
        return gcdofint(b,a%b)
######
a=int(argv[1])
b=int(argv[2])
print(a,b)
print(gcdofint(a,b))
```

测试结果

惨败

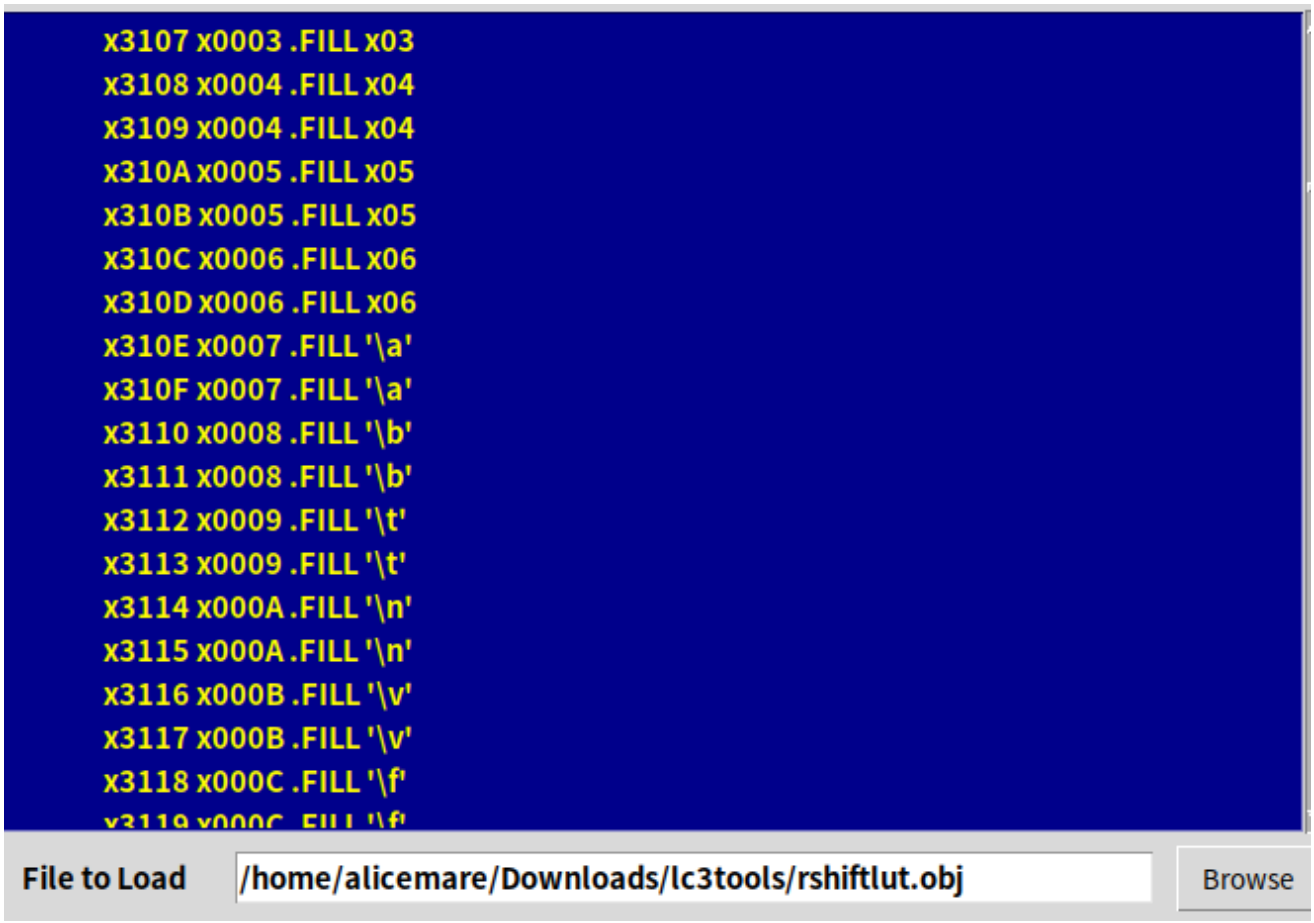
事实上大部分测试都是失败的...(这个截图的算好的了)



bug发现

在我在lc3sim中加载了lut.obj之后发现很多内存单元并不是我想的那样是8位的移位结果，而是ASCII码表...并且想了很多方法，除非是一个个点击这些内存单元然后修改为应有的值，才能保证程序正常输出结果

很显然在查表查到这些单元的时候就会失败...所以我的程序基本上都是只有那些互质的才能输出结果....因为这些互质的不用查表



实验总结

在这次实验感觉学到了很多东西

1. shell方面，我学习了如何用shell编程script来批处理文件，~~自动化的感觉真是好，很有成就感~~
2. 算法方面，知道了求gcd除了传统的更相减损，辗转相除外还有更容易在计算机体系上实现的stein算法
3. 汇编方面，感受到了以前程序员在和昂贵而有限内存空间和不够用的寄存器斗争的勇气和智慧，同时感受到了抽象给现代程序员们带来的幸福生活
4. 了解到了一些关于程序性能优化的方法

最后，谢谢款待，(咸鱼表示尽力了2333