实验报告

一、实验名称：RISC-V 基本指令集模拟器设计与实现

班级：智能 1602

学号：201608010719

姓名：吕志恒

二、实验目标

完成一个模拟 RISC-V 的基本整数指令集 RV32I 的模拟器设计。

三、实验设计

考虑到 CPU 的执行顺序为：取指、译码、执行。

（1）取指，将指令写入寄存器：（只展示部分代码）

```
void Program(){
    /*U类指令*/
    WriteWord(0,(0x12345<<12)|(1<<7)|(LUI));
    WriteWord(4,(0x2<<12)|(2<<7)|(ALUPC));
    /*J类指令*/
    WriteWord(8,(0<<31)|(4<<21)|(0<<20)|(0<<12)|(3<<7)|(JAL));
    WriteWord(16,(12<<20)|(5<<15)|(0<<12)|(4<<7)|(JALR));
    /*B类指令*/

    WriteWord(24,(0<<31)|(0<<25)|(6<<20)|(5<<15)|(0<<12)|(4<<8)|(0<<7)|(BType));

    WriteWord(32,(0<<31)|(0<<25)|(6<<20)|(3<<15)|(1<<12)|(6<<8)|(0<<7)|(BType));

    WriteWord(44,(0<<31)|(0<<25)|(3<<20)|(6<<15)|(4<<12)|(4<<8)|(0<<7)|(BType));

    WriteWord(52,(0<<31)|(0<<25)|(6<<20)|(3<<15)|(5<<12)|(4<<8)|(0<<7)|(BType));

    WriteWord(60,(0<<31)|(0<<25)|(3<<20)|(6<<15)|(6<<12)|(4<<8)|(0<<7)|(BType));

    WriteWord(68,(0<<31)|(0<<25)|(6<<20)|(3<<15)|(7<<12)|(4<<8)|(0<<7)|(BType));
```

（2）译码：

```
void Decode(unsigned int IR){
    opcode= IR & 0x7f;
    rd= (IR>>7)& 0x1f;
    r1= (IR>>15)&0x1f;
    r2= (IR>>20)&0x1f;
    func3=(IR>>12)&0x7;
    func7=(IR>>25)&0x7f;

    imm31_12U = (IR>>12)& 0xfffff;

    imm31J=(IR>>31) & 1;
    imm30_21J=(IR>>21) & 0x3ff;
    imm20J=(IR>>20) &1;
    imm19_12J=(IR>>12) & 0xff;

    imm31_20JR=IR>>20;
    /*B类指令*/
    imm31B=imm31J;
    imm30_25B=(IR>>25)& 0x3f;
    imm11_8B=(IR>>8)&0xf;
    imm7B=(IR>>7)&0x1;
    /*I类指令*/
    imm31_20L=IR>>20;
    /*S类指令*/
    imm31S=imm31J;
    imm30_25S=(IR>>25)&0x3f;
    imm11_7S=(IR>>7) & 0x1f;
    /*I类指令*/

    imm_sign_31_20I=(int)IR>>20;
    shamt=(IR>>20)&0x1f;

    imm31_12U_0 = imm31_12U<<12;
    imm_sign_31_12J=(imm31J<<20)&0xfff0000|(imm19_12J<<12)|(imm20J<<11)|(imm30_21J);
    imm_sign_31_25B_11_7B=(imm31B<<12)&0xffff000|(imm7B<<11)|(imm30_25B<<5)|(imm11_8B);
    imm_sign_31_25S_11_7S=(imm31S<<12)&0xffff000|(imm30_25S<<5)|imm11_7S;
```

（3）执行：(只展示部分指令）

```
Decode(IR) ; //译码
switch(opcode){
    case LUI:{
        cout<<"执行LUI指令:将立即数作为高20位，低12位用0填充，结果放进rd寄存器"<<endl;
        R[rd]=imm31_12U_0;
        break;
    }
    case ALUPC:{
        cout<<"执行ALUPC指令，将立即数作为高20位，低12位用0填充，结果加上此时PC值放入rd寄存器，PC值本身不变"<<endl;
        R[rd]=imm31_12U_0+PC;
        break;
    }
    case JAL:{
        cout<<"执行JAL指令，将立即数有符号扩展*2+pc作为新的pc值，并将原pc+4放进rd寄存器"<<endl;
        R[rd]=PC+4;
        nextPC=PC+imm_sign_31_12J*2;
        break;
    }
    case JALR:{
        cout<<"执行JALR指令，将指令高12位作为立即数有符号扩展*2+r1作为新的pc值，并将原pc+4放进rd"<<endl;
        R[rd]=PC+4;
        nextPC=R[r1]+imm31_20JR*2;
        break;
    }
```

从图中可以看到 LUI、ALUPC、JAL 等指令。

四、测试平台

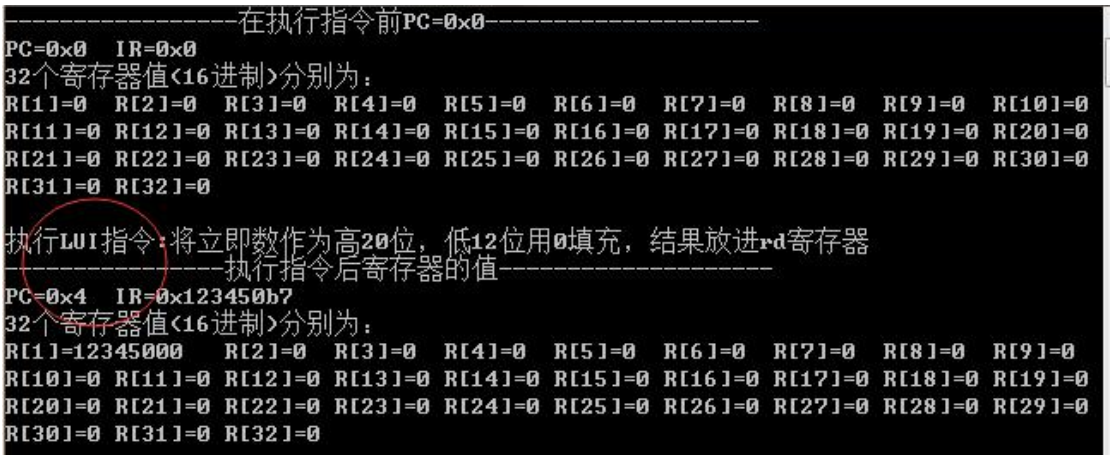
模拟器在以下平台进行测试：

部件	配件	备注
CPU	Core i7-7200U	
内存	DDR 16GB	
操作系统	Windows7 旗舰版	

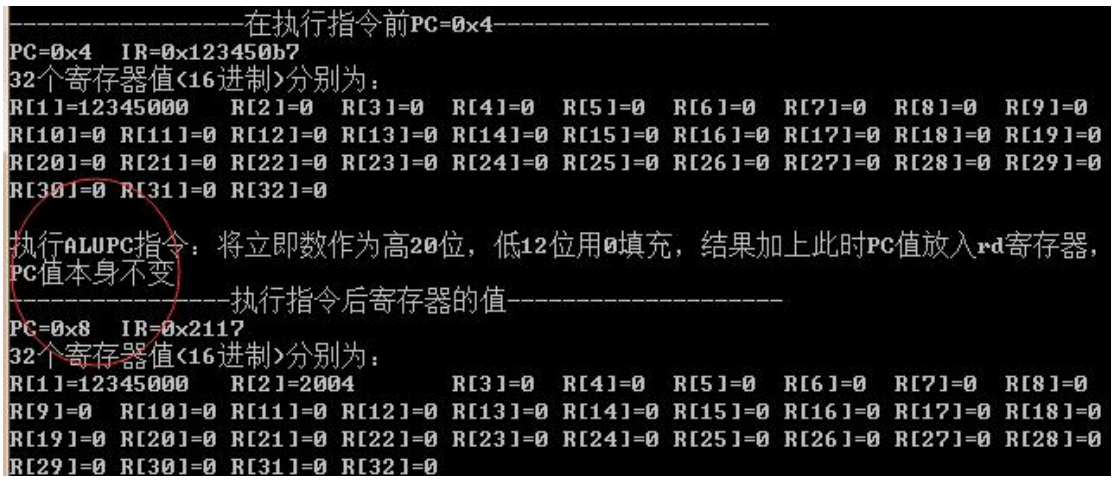
五、测试记录

以 LUI、ALUPC、JAL、JALR 指令为例子：

（1）LUI：

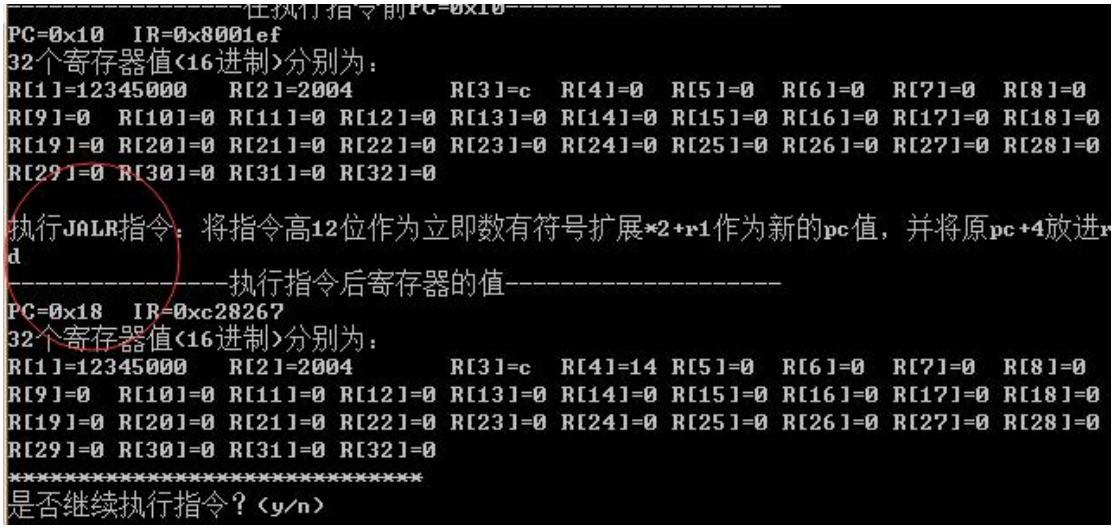


（2）ALUPC

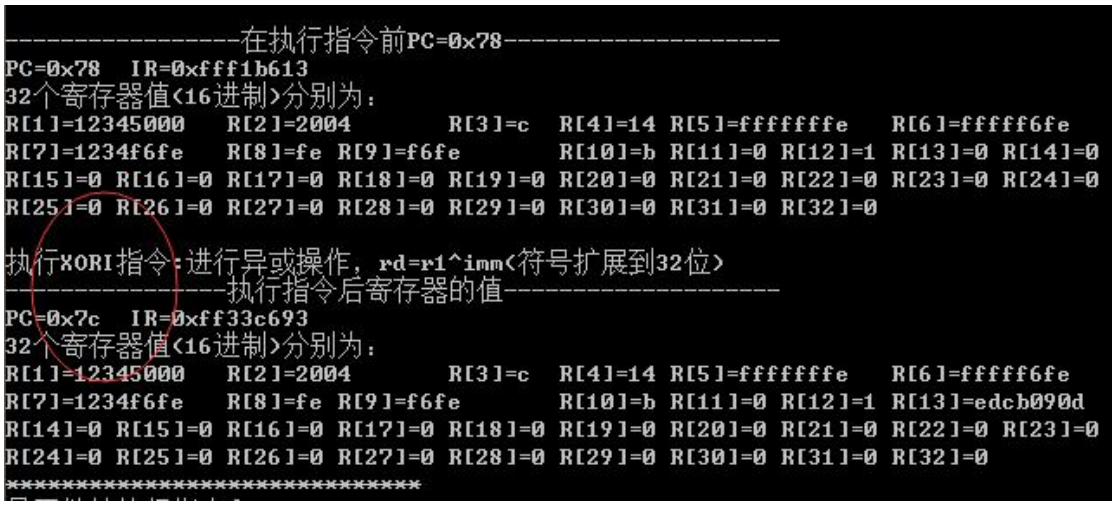


(3)执行 JAL 指令：

（4）指令 JALR 指令：



（5）其它指令（AND、XOR）



六、收获与体会

本次实验基本实现了 CPU 模拟器的功能，由于是使用 C++语言设计，中间涉及到指针，遇到的报错很多，比如重定位问题。但好在后来耐心调试解决了。写出模拟器的难点就在于怎么把存进的指令取出来同时 Pc 的值正确变化，本模拟器只能顺序执行指令，但我也想过改成支持外界输入指令执行，但没有想好如何更改 pc 的值。（我的思路是这样的：外界输入可以看成一次中断，将中断前 PC 的值保存下来，再去执行新输入的指令，执行完恢复 pc 的值，但理想总是美好的，调试了很久发现都没有用就放弃了。）本模拟器还有一个比较不完美的地方，程序退出的时候内存状态异常，这个异常是由指针引起的，目前还没有解决的办法。（Process terminated with status -1073741510 (2527 minute(s), 31 second(s))）