HUNAN UNIVERSITY

RISC-V 基本指令集模拟器设计与实现

班级：智能 1602

学号：201608010723

姓名：施园

一、实验目的

完成一个模拟 RISC-V 的基本整数指令集 RV32 的模拟器设计

二、实验要求

1、硬件设计采用 VHDL 或 Verilog 语言，软件设计采用 C/C++或 SystemC 语言，其它语言例如 Chisel、MyHDL 等也可选。

2、实验报告采用 markdown 语言，或者直接上传 PDF 文档

3、实验最终提交所有代码和文档。

三、实验过程

因为在暑假的时候做过相似的工作，所以本次实验相对顺利一些。在本工作中采用基本指令集 47 条指令中的 5 条，分别为 FENCE.I，EBREAK,CSRRS,CSRRWI,CSRRCI。FENCE.I 指令在 CPU 乱序执行的时候生效，相当于单周期的 nop 指令,EBREAK 指令是跳转到系统约定好的调试入口地址，在本次模拟中将调试入口地址设置为 4。

根据 cpu 执行指令的顺序：1、取指 2、译码 3、执行(包含运算和写回)，模拟器设计如下:

```
while(c != 'n') {
    cout << "Registers bofore executing the instruction @0x" << std::hex << PC << endl;
    showRegs();
    IR = readWord(PC);
    NextPC = PC + WORDSIZE;
    decode(IR);
    switch(opcode) {
        case LUI:
            cout << "Do LUI" << endl;
            R[rd] = Imm31_12UtypeZeroFilled;
            break;
        case AUIPC:
            cout << "Do AUIPC" << endl;
            cout << "PC = " << PC << endl;
            cout << "Imm31_12UtypeZeroFilled = " << Imm31_12UtypeZeroFilled << endl;
            R[rd] = PC + Imm31_12UtypeZeroFilled;
            break;
        case JAL:
            cout << "Do JAL"<<endl;
            R[rd]=PC+4;
            NextPC = PC+ Imm20_1JtypeSignExtended;
            break;
        case JALR:
            cout <<"DO JALR"<<endl;
            R[rd]=PC+4;
            NextPC=R[rs1]+Imm20_1JtypeSignExtended;
            break;
        case BRANCH:
            switch(funct3) {
                case BEQ:
                    cout <<"DO BEQ"<<endl;
                    }
                    break;
                case BNE:
                    cout <<"Do BNE "<<endl;
                    if(src1!=src2){
                        NextPC = PC + Imm12_1BtypeSignExtended;
                    }
                    break;
                case BLT:
                    cout <<"Do BLT"<<endl;
                    if((int)src1<(int)src2){
                        NextPC = PC + Imm12_1BtypeSignExtended;
                    }
                    break;
                case BGE:
                    cout <<"Do BGE"<<endl;
                    if((int)src1 >= (int)src2)
                        NextPC = PC + Imm12_1BtypeSignExtended;
                    break;
                case BLTU:
                    cout <<"Do BLTU"<<endl;
                    if(src1<src2){
                        NextPC=PC+Imm12_1BtypeSignExtended;
                    }
                    break;
                case BGEU:
                    cout<<"Do BGEU"<<endl;
                    if(src1>=src2){
                    NextPC=PC+Imm12_1BtypeSignExtended;
                    }
                    break;
```

```
        }
        break;
    case LOAD:
        switch(funct3) {
            case LB:
                cout <<"DO LB"<<endl;
                unsigned int LB_LH,LB_LH_UP;
                cout << "LB Address is: " << src1+Imm11_0ItypeSignExtended << endl;
                LB_LH=readByte(src1+Imm11_0ItypeSignExtended);
                LB_LH_UP=LB_LH>>7;
                if(LB_LH_UP==1){
                LB_LH=0xffffff00 & LB_LH;
                }else{
                LB_LH=0x000000ff & LB_LH;
                }
                R[rd]=LB_LH;
                break;
            case LH:
                cout << "Do LH " << endl;
                unsigned int temp_LH,temp_LH_UP;
                temp_LH=readHalfWord(src1+Imm11_0ItypeSignExtended);
                temp_LH_UP=temp_LH>>15;
                if(temp_LH_UP==1){
                    temp_LH=0xffff0000 | temp_LH;
                }else{
                    temp_LH=0x0000ffff & temp_LH;
                }
                R[rd]=temp_LH;
                break;
            case LW:
                cout << "Do LW" << endl;
```

```
                cout << "ERROR: Unknown funct3 in STORE instruction " << IR << endl;
        }
        break;
    case ALUIMM:
        switch(funct3) {
            case ADDI:
                cout << "Do ADDI" << endl;
                R[rd]=src1+Imm11_0ItypeSignExtended;
                break;
            case SLTI:
                cout << "Do SLTI" << endl;
                if(src1<Imm11_0ItypeSignExtended)
                    R[rd] = 1;
                else
                    R[rd] = 0;
                break;
            case SLTIU:
                cout <<"Do SLTIU" << endl;
                if(src1<(unsigned int)Imm11_0ItypeSignExtended)
                    R[rd] = 1;
                else
                    R[rd] = 0;
                break;
            case XORI:
                cout <<"Do XORI" << endl;
                R[rd]=(Imm11_0ItypeSignExtended)^R[rs1];
                break;
            case ORI:
                cout<<"Do ORI"<<endl;
                R[rd]=R[rs1]|Imm11_0ItypeSignExtended;
                break;
```

```
                }
                break;
            default:
                cout<<"ERROR: Unknown funct3 in ALUIMM instruction " << IR << endl;
        }
        break;
    case ALURRR:
        switch(funct3) {
            case ADDSUB:
                switch(funct7) {
                    case ADD:
                        cout << "Do ADD" << endl;
                        R[rd]=R[rs1]+R[rs2];
                        break;
                    case SUB:
                        cout<<" Do SUB"<<endl;
                        R[rd]=R[rs1]-R[rs2];
                        break;
                    default:
                        cout << "ERROR: Unknown funct7 in ALURRR ADDSUB instruction " << IR << endl;
                }
                break;
            case SLL:
                cout<<"DO SLL"<<endl;
                unsigned int rsTransform;
                rsTransform=R[rs2]&0x1f;
                R[rd]=R[rs1]<<rsTransform;
                break;
            case SLT:
                cout << "Do SLT " << endl;
                if((int)src1<(int)src2){
```

对模拟器进行测试(其中操作系统为 windows10)结果如下:

测试输入如下:

1、writeWord(0,0x0013ab73);//000000000001 0011 1010 1011 01110011

2、writeWord(4,0x0013db73);//000000000001 0011 1101 1011 01110011

3、writeWord(8,0x0013fb73);//000000000001 0011 1111 1011 01110011

4、writeWord(12,0x0000100f);//000000000000 0000 0001 0000 00001111

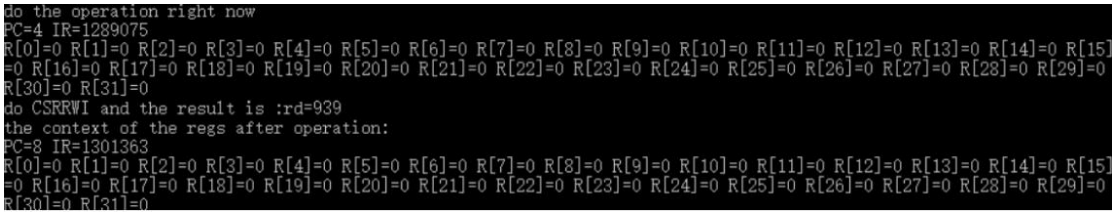
5、writeWord(16,0x00100073);//000000000001 0000 0000 0000 01110011

其中对应的指令分别为:CSRRS,CSRRWI,CSRRCI,FENCE.I,EBREAK

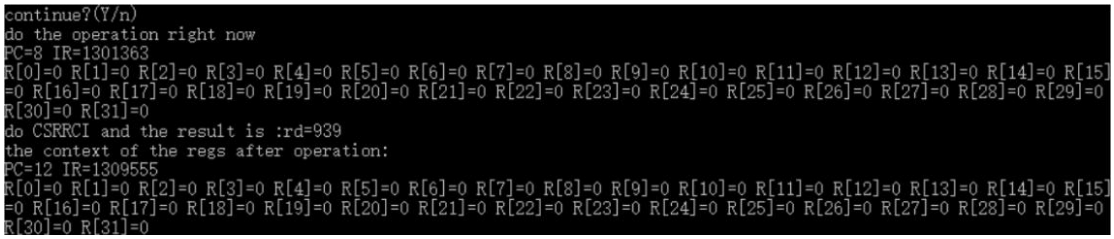
其中 CSRRS 指令执行如下：

```
do the operation right now
PC=0 IR=0
R[0]=0 R[1]=0 R[2]=0 R[3]=0 R[4]=0 R[5]=0 R[6]=0 R[7]=0 R[8]=0 R[9]=0 R[10]=0 R[11]=0 R[12]=0 R[13]=0 R[14]=0 R[15]
=0 R[16]=0 R[17]=0 R[18]=0 R[19]=0 R[20]=0 R[21]=0 R[22]=0 R[23]=0 R[24]=0 R[25]=0 R[26]=0 R[27]=0 R[28]=0 R[29]=0
R[30]=0 R[31]=0
do CSRRS and the result is :rd=943
the context of the regs after operation:
PC=4 IR=1289075
R[0]=0 R[1]=0 R[2]=0 R[3]=0 R[4]=0 R[5]=0 R[6]=0 R[7]=0 R[8]=0 R[9]=0 R[10]=0 R[11]=0 R[12]=0 R[13]=0 R[14]=0 R[15]
=0 R[16]=0 R[17]=0 R[18]=0 R[19]=0 R[20]=0 R[21]=0 R[22]=0 R[23]=0 R[24]=0 R[25]=0 R[26]=0 R[27]=0 R[28]=0 R[29]=0
R[30]=0 R[31]=0
```

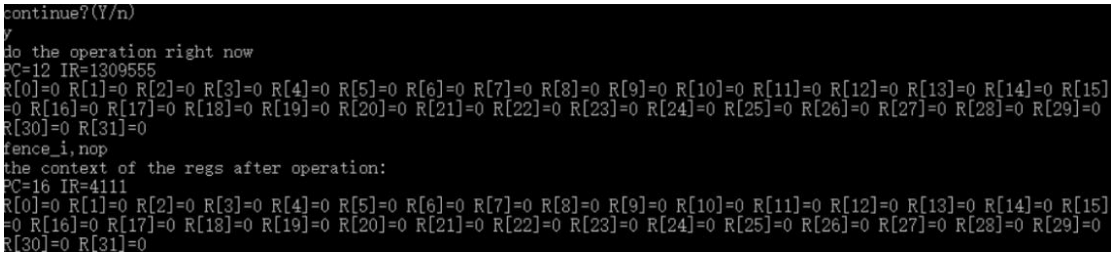
CSRRWI 指令执行如下：



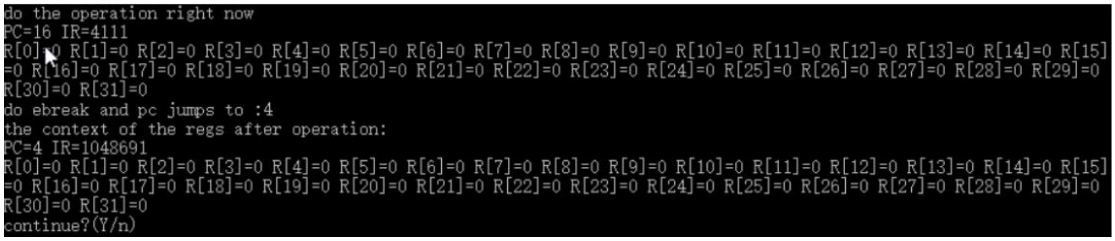
CSRRCI 指令执行如下：



FENCE.I 指令执行如下：



EBREAK 执行如下：



四、分析和总结

从测试记录可以看到，模拟器实现了对二进制指令文件的读入，指令功能的模拟，CPU 以及存储器状态的输出。综上所述，所编写的模拟器实现了所要求的的功能，完成了实验目标。