实验报告

实验名称（RISC-V基本指令集模拟器设计与实现）

班级：信息安全1501

学号：201508060122

姓名：刘菲菲

实验目标

设计一个CPU模拟器，能模拟CPU指令集的功能。

实验要求

- 采用C/C++编写程序
- 模拟器的输入是二进制的机器指令文件
- 模拟器的输出是CPU各个寄存器的状态和相关的存储器单元状态

实验内容

CPU指令集

CPU的指令集请见这里，其中基本指令集共有*47*条指令。

我的五条指令为:FENCE.I,EBREAK,CSRRS,CSRRWI,CSRRCI

其中，fence.i指令在CPU乱序执行的时候才会生效，而在单周期的cpu中，作用相当于nop；ebreak指令是跳转到系统约定好的调试入口地址，因此在本次模拟中我将这个调试入口地址设置为4。

模拟器程序框架

考虑到CPU执行指令的流程为：

1. 取指
2. 译码
3. 执行（包括运算和结果写回）

对模拟器程序的框架设计如下：

```
while(1) {
    inst = fetch(cpu.pc);
    cpu.pc = cpu.pc + 4;

    inst.decode();

    switch(inst.opcode) {
        case FENCES:
            switch (funct3) {
                case FENCE_I:
                    //TODO: 补充指令模拟代码:
                    cout<<"nop"<<endl;
                    break;
                default:
                    cout << "ERROR: unknown funct3 in
FENCES instruction " << IR << endl;
                    break;
                }
        case CSRX:
            switch (funct3) {
            case CALLBREAK:
                switch (imm11_0i) {
                    case EBREAK:
                        //TODO: 补充指令模拟代码:
                        PC = ebreakadd;
                        break;
                    default:
                        cout << "ERROR: unknown imm11_0i
in CSRX CALLBREAK instruction " << IR << endl;
                        break;
                    }
            case CSRRS:
                {
                    uint32_t temp =
readWord(rs2)&0x00000fff;
                    uint32_t temp1 = rs1 & 0x000fffff;
                    writeWord   (rd,(temp|temp1));
                    cout << "do CSRRS and the result is
:" << "rd="<<readWord(rd)<<endl;
                    break;
                }
            case CSRRWI:
            {   if (rd == 0) break;
                else
                {
                    uint32_t zmm = imm11j& 0x000001f;
                    uint32_t tem = readWord(rs2) &
0x00000fff;
```

```
                    writeWord(rd, tem);
                    writeWord(rs2, zmm);
                    cout << "do CSRRWI and the result is
:" << "rd=" << readWord(rd) << endl;
                    break;
                }
            }
            case CSRRCI:
                //TODO: 补充指令模拟代码:
                {   uint32_t zmm = imm11j & 0x000001f;
                    uint32_t tem = readWord(rs2) &
0x00000fff;
                    if (readWord(rd) != 0)
                    {
                        writeWord(rs2, zmm | tem);
                    }
                    cout << "do CSRRCI and the result is
:" << "rd=" << readWord(rd) << endl;

                    break;
                }
            default:
                cout << "ERROR: unknown funct3 in CSRX
instruction " << IR << endl;
            }
            break;
    }
}
```

其中while循环条件可以根据需要改为模拟终止条件。

测试

测试平台

模拟器在如下机器上进行了测试：

部件	配置	备注
CPU	core i5-5200U	
内存	DDR3 8GB	
操作系统	Windows 10 专业版	中文版

测试记录

模拟器的测试输入如下：

```
void progMem() {
    // 从地址0开始写入测试指令
    writeWord(0, 0x0013ab73);// 000000000001  0011 1 010 1011 0 1110011
    writeWord(4, 0x0013db73);//000000000001 /0011 /1 101 /1011 /0 1110011
    writeWord(8, 0x0013fb73);//000000000001 /0011 /1 111/1011/0 1110011
    writeWord(12, 0x0000100f);//000000000000 0000/0 001/0000/0 0001111
    writeWord(16, 0x00100073);//000000000001 0000/0 000/0000/0 1110011
}
```

依次对应的的指令为：CSRRS,CSRRWI,CSRRCI，FENCE.I,EBREAK。

模拟器运行过程的截图如下：

第一条指令运行后模拟器的输出

```
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
```

第二条指令运行后模拟器的输出

```
do the operation right now
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
do CSRRWI and the result is :rd=939
the context of the regs after operation:
PC=8 IR=1301363
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
```

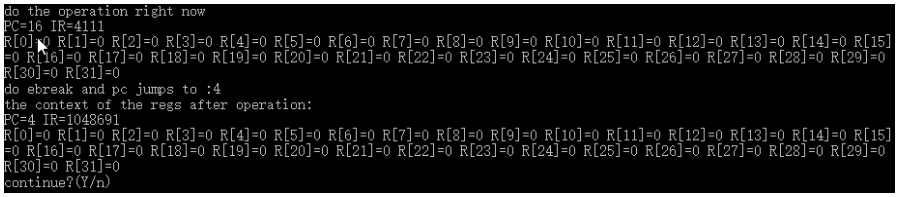
第三条指令运行后模拟器的输出

```
continue?(Y/n)
do the operation right now
PC=8 IR=1301363
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 CSRRCI and the result is :rd=939
the context of the regs after operation:
PC=12 IR=1309555
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
```

第四条指令运行后模拟器的输出

```
continue?(Y/n)
y
do the operation right now
PC=12 IR=1309555
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
fence_i,nop
the context of the regs after operation:
PC=16 IR=4111
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
```

最后一条指令运行后模拟器的输出



分析和结论

从测试记录来看，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和存储器状态的输出。

根据分析结果，可以认为编写的模拟器实现了所要求的功能，完成了实验目标。