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：通信 1502 班
学号：201508030227
姓名：曾佳丽

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

1.采用 C/C++编写程序

2.模拟器的输入是二进制的机器指令文件

3.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

请见这里，其中基本指令集共有 47 条。我完成的五条指令是，JALR SH SUB ORI BLGU

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（包括运算和写会）

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 ADD:
            cpu.regs[inst.rd] = cpu.regs[rs] + cpu.regs[rt];
            break;
        case /*其它操作码*/ :
            /* 执行相关操作 */
            break;
        default:
            cout << "无法识别的操作码：” << inst.opcode;
    }
}
```

其中 while 循环条件可以根据需要改为模拟终止条件

具体指令内容如下：

```
case JALR:
    cout << "DO JALR" << endl;
    R[rd]=PC+4;
    NextPC=R[rs1]+Imm20_1JtypeSignExtended;
    break;
```

```
case BGEU:
    cout<<"Do BGEU"<<endl;

    if(src1>=src2){
        NextPC=PC+Imm12_1BtypeSignExtended;
    }
    break;
```

```
case SH:
    cout<<"Do SH"<<endl;
    uint16_t j;
    j=R[rs2]&0xffff;
    unsigned int x;
    x = R[rs1] + Imm11_0StypeSignExtended;
    writeHalfWord(x,j);
    break;
```

```
case ORI:
    cout<<"Do ORI"<<endl;
    R[rd]=R[rs1]|Imm11_0ItypeSignExtended;
    break;
case SUB:
    cout<<" Do SUB"<<endl;
    R[rd]=R[rs1]-R[rs2];
    break;
```

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 i5-6500U	
内存	DDR3 4GB	
操作系统	Windows 8	中文版

测试记录

模拟器的测试输入指令：

```
void progMem() {
    writeWord(0, (0xfffff << 12) | (2 << 7) | (LUI));
    writeWord(4, (1 << 12) | (5 << 7) | (AUIPC));
    writeWord(8, (0x20<<25) | (5<<20) | (0<<15) | (SW << 12) | (0 << 7) | (STORE));
    writeWord(12, (0x400<<20) | (0<<15) | (LB<<12) | (3<<7) | (LOAD));

    writeWord(16, (0x0<<25) | (4<<20) | (1<<15) | (BGEU<<12) | (0x8<<7) | (BRANCH));
    writeWord(24, (0x20<<25) | (7<<20) | (0<<15) | (SH << 12) | (0 << 7) | (STORE));


    writeWord(28, (SUB<<25) | (6<<20) | (4<<15) | (ADDSUB << 12) | (9 << 7) | (ALURRR));
    writeWord(32, (0x400<<20) | (2<<15) | (ORI<<12) | (4<<7) | (ALUIMM));
    writeWord(36,(0x400)<<20|(1<<15)|(JALR<<12)|(4<<7)|(JALR));
}
```

JALR BGEU SH ORI SUB 指令的作用：

JALR:直接跳转指令，无条件跳转到由寄存器 rs1 指定的指令，并将下一条指令的地址保存到寄存器 rd（默认为 31）中

BGEU:为无符号比较，当 R[rs1]>=R[rs2]时进行跳转

SH：SH 指令取寄存器 R[rs2]的低位存储 16 位值到存储器。有效的字节地址是通过将寄存器 R[rs1]添加到符号扩展的 12 位偏移来获得的

ORI: 在寄存器 R[rs1]上执行位 OR 的逻辑操作，并立即将符号扩展 12 位，将结果放在 R[rd]中

SUB:进行加法运算，R[rs1]-R[rs2]，将结果存入 R[rd]中

模拟器运行过程的截图如下：

进行 5 条指令之前由执行过娶她指令后的地址和各个寄存器的值

```
Registers bofore executing the instruction @0x0
PC=0x0 IR=0x0
R[0]=0x0 R[1]=0x0 R[2]=0x0 R[3]=0x0 R[4]=0x0 R[5]=0x0 R[6]=0x0 R[7]=0x0 R[8]=0x0
 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0 R[10]=0x0 R[11]=
0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0 R[18]=0x0 R[19]=
0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Do LUI
Registers after executing the instruction
PC=0x4 IR=0xfffff137
R[0]=0x0 R[1]=0x0 R[2]=0xfffff000 R[3]=0x0 R[4]=0x0 R[5]=0x0 R[6]=0x0 R[7]=0x0 R
[8]=0x0 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0 R[10]=0x0
 R[11]=0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0 R[18]=0x0
 R[19]=0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Continue simulation (Y/n)? [Y]
Y
Registers bofore executing the instruction @0x4
PC=0x4 IR=0xfffff137
R[0]=0x0 R[1]=0x0 R[2]=0xfffff000 R[3]=0x0 R[4]=0x0 R[5]=0x0 R[6]=0x0 R[7]=0x0 R
[8]=0x0 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0 R[10]=0x0
 R[11]=0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0 R[18]=0x0
 R[19]=0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Do AUIPC
PC = 4
Imm31_12UtypeZeroFilled = 1000
Registers after executing the instruction
PC=0x8 IR=0x1297
R[0]=0x0 R[1]=0x0 R[2]=0xfffff000 R[3]=0x0 R[4]=0x0 R[5]=0x1004 R[6]=0x0 R[7]=0x
0 R[8]=0x0 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0 R[10]=
0x0 R[11]=0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0 R[18]=
0x0 R[19]=0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Continue simulation (Y/n)? [Y]
Registers bofore executing the instruction @0x8
PC=0x8 IR=0x1297
R[0]=0x0 R[1]=0x0 R[2]=0xfffff000 R[3]=0x0 R[4]=0x0 R[5]=0x1004 R[6]=0x0 R[7]=0x
0 R[8]=0x0 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0 R[10]=
0x0 R[11]=0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0 R[18]=
0x0 R[19]=0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
DO SW
SW Addr and Data are: 400, 1004
Registers after executing the instruction
PC=0xc IR=0x40502023
R[0]=0x0 R[1]=0x0 R[2]=0xfffff000 R[3]=0x0 R[4]=0x0 R[5]=0x1004 R[6]=0x0 R[7]=0x
0 R[8]=0x0 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0 R[10]=
0x0 R[11]=0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0 R[18]=
0x0 R[19]=0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
```

```
Continue simulation (Y/n)? [Y]
Y
Registers bofore executing the instruction @0xc
PC=0xc IR=0x40502023
R[0]=0x0 R[1]=0x0 R[2]=0xfffff000 R[3]=0x0 R[4]=0x0 R[5]=0x1004 R[6]=0x0 R[7]=0x
0 R[8]=0x0 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0 R[10]=
0x0 R[11]=0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0 R[18]=
0x0 R[19]=0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
DO LB
LB Address is: 400
Registers after executing the instruction
PC=0x10 IR=0x40000183
R[0]=0x0 R[1]=0x0 R[2]=0xfffff000 R[3]=0x4 R[4]=0x0 R[5]=0x1004 R[6]=0x0 R[7]=0x
0 R[8]=0x0 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0 R[10]=
0x0 R[11]=0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0 R[18]=
0x0 R[19]=0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
```

运行结果如下：

BGEU 运行结果如下

```
Do BGEU
Registers after executing the instruction
PC=0x18 IR=0x40f463
R[0]=0x0 R[1]=0x0 R[2]=0xfffff000 R[3]=0x4 R[4]=0x0 R[5]=0x1004 R[6]=0x0 R[7]=0x
0 R[8]=0x0 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0 R[10]=
0x0 R[11]=0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0 R[18]=
0x0 R[19]=0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Continue simulation (Y/n)? [Y]
```

测试指令分别取 R[4]为 R[rs1]和 R[1]为 R[rs2]进行比较，因为两者皆为 0，所以跳转条件不成立，则 PC=PC+4+4=0x18。测试结果正确。

SH 运行结果如下：

```
Continue simulation (Y/n)? [Y]
Y
Registers bofore executing the instruction @0x18
PC=0x18 IR=0x40f463
R[0]=0x0 R[1]=0x0 R[2]=0xfffff000 R[3]=0x4 R[4]=0x0 R[5]=0x1004 R[6]=0x0 R[7]=0x
0 R[8]=0x0 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0 R[10]=
0x0 R[11]=0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0 R[18]=
0x0 R[19]=0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Do SH
Registers after executing the instruction
PC=0x1c IR=0x40701023
R[0]=0x0 R[1]=0x0 R[2]=0xfffff000 R[3]=0x4 R[4]=0x0 R[5]=0x1004 R[6]=0x0 R[7]=0x
0 R[8]=0x0 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0 R[10]=
0x0 R[11]=0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0 R[18]=
0x0 R[19]=0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
```

SUB 指令运行后模拟器的输出

```
Continue simulation (Y/n)? [Y]
Registers bofore executing the instruction @0x1c
PC=0x1c IR=0x40701023
R[0]=0x0 R[1]=0x0 R[2]=0xfffff000 R[3]=0x4 R[4]=0x0 R[5]=0x1004 R[6]=0x0 R[7]=0x
0 R[8]=0x0 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0 R[10]=
0x0 R[11]=0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0 R[18]=
0x0 R[19]=0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
 Do SUB
Registers after executing the instruction
PC=0x20 IR=0x406204b3
R[0]=0x0 R[1]=0x0 R[2]=0xfffff000 R[3]=0x4 R[4]=0x0 R[5]=0x1004 R[6]=0x0 R[7]=0x
0 R[8]=0x0 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0 R[10]=
0x0 R[11]=0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0 R[18]=
0x0 R[19]=0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Continue simulation (Y/n)? [Y]
```

测试指令分别取 R[4]为 R[rs1]和 R[2]为 R[rs2]相减，原值分别为 0x0 和 0x0,结果为 0x0 存入 R[a]中，测试结果正确。

ORI 指令运行结果

```
Continue simulation (Y/n)? [Y]
Y
Registers bofore executing the instruction @0x20
PC=0x20 IR=0x406204b3
R[0]=0x0 R[1]=0x0 R[2]=0xfffff000 R[3]=0x4 R[4]=0x0 R[5]=0x1004 R[6]=0x0 R[7]=0x
0 R[8]=0x0 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0 R[10]=
0x0 R[11]=0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0 R[18]=
0x0 R[19]=0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Do ORI
Registers after executing the instruction
PC=0x24 IR=0xf2676213
R[0]=0x0 R[1]=0x0 R[2]=0xfffff000 R[3]=0x4 R[4]=0xffffff26 R[5]=0x1004 R[6]=0x0
R[7]=0x0 R[8]=0x0 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0
 R[10]=0x0 R[11]=0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0
 R[18]=0x0 R[19]=0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
```

测试指令取R[4]为R[rd],原值为0x0，取R[2]为R[rs1],原值为0xfffff000,为Imm11_0ItypeSignExtended赋值为0x100,与R[rs1]进行OR,结果为0x104存入R[9]中，测试结果正确。

JALR指令执行结果

```
Continue simulation (Y/n)? [Y]
Registers bofore executing the instruction @0x24
PC=0x24 IR=0x40016213
R[0]=0x0 R[1]=0x0 R[2]=0xfffff000 R[3]=0x4 R[4]=0xfffff400 R[5]=0x1004 R[6]=0x0
R[7]=0x0 R[8]=0x0 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0
 R[10]=0x0 R[11]=0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0
 R[18]=0x0 R[19]=0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0

DO JALR
Registers after executing the instruction
PC=0x6f400 IR=0x4006f267
R[0]=0x0 R[1]=0x0 R[2]=0xfffff000 R[3]=0x4 R[4]=0x28 R[5]=0x1004 R[6]=0x0 R[7]=0
x0 R[8]=0x0 R[9]=0x0 R[a]=0x0 R[b]=0x0 R[c]=0x0 R[d]=0x0 R[e]=0x0 R[f]=0x0 R[10]
=0x0 R[11]=0x0 R[12]=0x0 R[13]=0x0 R[14]=0x0 R[15]=0x0 R[16]=0x0 R[17]=0x0 R[18]
=0x0 R[19]=0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Continue simulation (Y/n)? [Y]
```

分析和结论

从测试记录来看，模拟器实现了对十六进制指令文件的读入，实现了JALR BGEU SH SUB ORI五条指令的功能