湖南大學

HUNAN UNIVERSITY

大三小学期
实验报告

题　　目：RISC-V 的基本整数指令集 RV32I 的模拟器设计

学生姓名：孙文源

学生学号：201608030110

专业班级：通信 1601

指导老师：吴强

一、 实验目的

设计一个 CPU 模拟器，能模拟 CPU 指令集的功能。

二、 实验条件

CPU：core i5-6200U
操作系统：Windows10 家庭版
内存：DDR3 4GB

三、 实验内容

作为 CPU 模拟器，需要考虑到的基本执行过程包括：取指，译码，指令执行三个基本过程，之后重复循环。因此模拟器的流程框架为：

```
PC = 0;
while（）{
        IR = readWord(PC);
        decode(IR);
        switch（op）
PC = NextPC;}
```

四、 指令测试过程

1. LUI

将 IR 指令的前 20 位的立即数当成结果的高 20 位，低 12 位填 0，放进 rd 为下标的寄存器。

```
case LUI:
    cout << "Do LUI" << endl;
    R[rd] = Imm31_12UtypeZeroFilled;
    break;
```

```
writeWord(0, (0xffff0 << 12) | (4 << 7) | (LUI));//LUI
```

测试结果：

```
PC=0x4 IR=0xffff0237
R[0]=0x0 R[1]=0x0 R[2]=0x0 R[3]=0x0 R[4]=0xffff0000 R[5]=0x0 R[6]=0x0 R[7]=0x0 R[8]=0x0 R[9]=0x
0 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]
```

2. AUIPC

将高 20 位表示的立即数加到 pc 上，并将结果写入 rd 寄存器，但 pc 本身值不变

```
case AUIPC:
    cout << "Do AUIPC" << endl;
    cout << "PC = " << PC << endl;
    cout << "Imm31_12UtypeZeroFilled = " << Imm31_12UtypeZeroFilled << endl;
    R[rd] = PC + Imm31_12UtypeZeroFilled;
    break;
```

```
writeWord(4, (1 << 12) | (5 << 7) | (AUIPC));//AUIPC
```

测试结果：

```
PC=0x8 IR=0x1297
R[0]=0x0 R[1]=0x0 R[2]=0x0 R[3]=0x0 R[4]=0xffff0000 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
[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Continue simulation (Y/n)? [Y]
```

3. SW/SH/SB

SW/SH/SB 分别将寄存器 rs2 中的低 32/16/8/位到储存器中。将 rs2 寄存器中的低 32 位存到内存地址为：基址是 rs1+偏移量。

```
case SB:
    cout << "Do SB" << endl;
    char sb_d1;
    unsigned int sb_a1;
    sb_d1=R[rs2] & 0xff;
    sb_a1 = R[rs1] +Imm11_0StypeSignExtended;
    writeByte(sb_a1, sb_d1);
    cout<<*((uint32_t*)&(M[512]))<<endl;
    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 SW:
    cout << "DO SW" << endl;
    //unsigned int imm_temp;
    uint32_t _swData;
    _swData=R[rs2] & 0xffffffff;
    unsigned int _swR;
    _swR = R[rs1] + Imm11_0StypeSignExtended;
    cout << "SW Addr and Data are: " << _swR << ", " << _swData << endl;
    writeWord(_swR, _swData);
    cout<<*((uint32_t*)&(M[1024]))<<endl;
    break;
```

writeWord(4, (0x0<<25) | (4<<20) | (0<<15) | (SW << 12) | (4 << 7) | (STORE));//SW

测试结果：

```
DO SW
SW Addr and Data are: 4, ffff0000
ffff0000
Registers after executing the instruction
PC=0x8 IR=0x402223
R[0]=0x0 R[1]=0x0 R[2]=0x0 R[3]=0x0 R[4]=0xffff0000 R[5]=0x0 R[6]=0x0 R[7]=0x0 R[8]=0x0 R[9]=0x
0 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]
```

4. LW/LH/LB

LW 将 32 位值复制到 rd 中，LH 从储存器中读取 16 位，然后将其符号扩展到 32 位，保存到 dr 中。LHU 指令读取存储器 16 位，然后 0 扩展到 32 位，再保存到 rd 中。LB/LBU 则是读取 8 位。

```
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;
    cout<<'1'<<*((uint32_t*)&(M[1024]))<<endl;
    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;
    cout<<'1'<<*((uint32_t*)&(M[4]))<<endl;
    break;
```

```
case LW:
    cout << "Do LW" << endl;
    unsigned int temp_LW,temp_LW_UP;
    temp_LW=readByte(src1+Imm11_0ItypeSignExtended);
    temp_LW_UP=temp_LW>>31;
    if(temp_LW_UP==1){
        temp_LW=0x00000000 | temp_LW;
    }else{
        temp_LW=0xffffffff & temp_LW;
    }
    R[rd]=temp_LW;
    break;
```

writeWord(4, (0x4<<20) | (0<<15) | (5<<12) | (3<<7) | (LOAD));//LB

测试结果：

```
LB Address is: 4
10
Registers after executing the instruction
PC=0x8 IR=0x400183
R[0]=0x0 R[1]=0x0 R[2]=0x0 R[3]=0x83 R[4]=0xffff0000 R[5]=0x0 R[6]=0x0 R[7]=0x0 R[8]=0x0 R[9]=
x0 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]=0
0 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[
d]=0x0 R[1e]=0x0 R[1f]=0x0
Continue simulation (Y/n)? [Y]
```

5. SLL/SRL/SRA

分别逻辑左右移，算术右移。被移位的是 rs1,移动 rs2 的低 5 位。

```
case SLL:
    cout<<"DO SLL"<<endl;
    unsigned int rsTransform;
    rsTransform=R[rs2]&0x1f;
    R[rd]=R[rs1]<<rsTransform;
    break;
```

```
 case SRL:
cout<<"DO SRL"<<endl;
            R[rd]=R[rs1]>>R[rs2];
    break;
 case SRA:
      cout<<"DO SRA"<<endl;
      R[rd]=(int)src1>>src2;
    break;
```

writeWord(8, (0<<25) | (3<<20) | (4<<15) | (SLL<<12) | (5<<7) | (ALURRR));//SLL

测试结果：

```
DO SLL
Registers after executing the instruction
PC=0xc IR=0x3212b3
R[0]=0x0 R[1]=0x0 R[2]=0x0 R[3]=0x1000 R[4]=0xffff0000 R[5]=0xffff0000 R[6]=0x0 R[7]=0x0 R
x0 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
 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
]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Continue simulation (Y/n)? [Y]
```

6. ADD/SUB

ADD/SUB 分别用于执行加减法、忽略溢出。

```
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;
```

writeWord(8, (ADD<<25) | (3<<20) | (4<<15) | (ADDSUB << 12) | (5 << 7) | (ALURRR));//ADD

测试结果：

```
Do ADD
Registers after executing the instruction
PC=0xc IR=0x3202b3
R[0]=0x0 R[1]=0x0 R[2]=0x0 R[3]=0x1000 R[4]=0xffff0000 R[5]=0xffff1000 R[6]=0x0 R[7]=0x0 R
x0 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
 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
]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Continue simulation (Y/n)? [Y]
```

7. ADDI/SLTI

ADDI：将 12 位有符号立即数和 rs 相加，溢出忽略，直接使用结果的最低 32bit，并存入 rd。SLTI：如果 rs 小于立即数(都是有符号整数),将 rd 置 1,否则置 0。

```
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;
```

writeWord(4, (1<<20) | (4<<15) | (ADDSUB << 12) | (3<<7) | (ALUIMM));//ADDI

测试结果：

```
Do ADDI
Registers after executing the instruction
PC=0x8 IR=0x120193
R[0]=0x0 R[1]=0x0 R[2]=0x0 R[3]=0xffff0001 R[4]=0xffff0000 R[5]=0x0 R[6]=0x0 R[7]=0x0 R[8]=
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]=0x
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[1
x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Continue simulation (Y/n)? [Y]
```

8. ANDI/ORI/XORI

ANDI/ORI/XORI：rs 与有符号 12 位立即数进行 and,or,xor 操作。

```
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;
case ANDI:
    cout << "DO ANDI"<<endl;
    R[rd]=R[rs1]&Imm11_0ItypeSignExtended;
    break;
```

writeWord(4, (0x100<<20) | (4<<15) | (XORI << 12) | (2 << 7) | (ALUIMM));//XORI

测试结果：

```
Do XORI
Registers after executing the instruction
PC=0x8 IR=0x10024113
R[0]=0x0 R[1]=0x0 R[2]=0xffff0100 R[3]=0x0 R[4]=0xffff0000 R[5]=0x0 R[6]=0x0 R[7]=0x0 R[8]=0x
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
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]
x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Continue simulation (Y/n)? [Y]
```

9. SLLI/SRLI/SRAI

SLLI：逻辑左移，低位移入 0。SRLI：逻辑右移，高位移入 0。

SRAI：算数右移，符号移入高位

```
case SLLI:
    cout << "Do SLLI " << endl;
    R[rd]=src1<<shamt;
    break;
case SHR:
    switch(funct7) {
        case SRLI:
            cout << "Do SRLI" << endl;
            R[rd]=src1>>shamt;
            break;
        case SRAI:
            cout << "Do SRAI" << endl;
            R[rd] = ((int)src1) >> shamt;
            cout<<rd<<endl;
            break;
```

writeWord(4, (0<<25) | (1<<20) | (4<<15) | (SHR<<12) | (2<<7) | (ALUIMM));//SRLI

测试结果：

```
Do SRLI
Registers after executing the instruction
PC=0x8 IR=0x125113
R[0]=0x0 R[1]=0x0 R[2]=0x7fff8000 R[3]=0x0 R[4]=0xffff0000 R[5]=0x0 R[6]=0x0 R[7]=0x0 R[8]=0x
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
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]
x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Continue simulation (Y/n)? [Y]
```

10. BEQ/BNE

rs1(==/!=)rs2，分别在相等或者不等时，发生跳转。

```
case BEQ:
    cout << "DO BEQ" << endl;
    if(src1==src2){
        NextPC = PC + Imm12_1BtypeSignExtended;
    }
    break;
case BNE:
    cout << "Do BNE " << endl;
    if(src1!=src2){
        NextPC = PC + Imm12_1BtypeSignExtended;
    }
    break;
```

writeWord(4, (0<<25) | (0<<20) | (1<<15) | (BEQ<<12) | (0x10<<7) | (BRANCH)) ;//BEQ

测试结果：

```
DO BEQ
Registers after executing the instruction
PC=0x14 IR=0x8863
R[0]=0x0 R[1]=0x0 R[2]=0x0 R[3]=0x0 R[4]=0xffff0000 R[5]=0x0 R[6]=0x0 R[7]=0x0 R[8]=0x0 R[9]=0
0 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]=0x
 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[1
]=0x0 R[1e]=0x0 R[1f]=0x0
Continue simulation (Y/n)? [Y]
```

11. AND/OR/XOR

rs1 与 rs2 进行 and,or,xor 操作

```
case XOR:
    cout << "Do XOR " << endl;
    R[rd]=R[rs1]^R[rs2];
    break;
case OR:
    cout << "Do OR" << endl;
        R[rd]=R[rs1]|R[rs2];
     break;
case AND:
    cout << "Do AND" << endl;
        R[rd]=R[rs1]&R[rs2];
    break;
```

writeWord(4, (0<<25) | (4<<20) | (1<<15) | (OR<<12) | (2<<7) | (ALURRR));//OR

测试结果：

```
Do OR
Registers after executing the instruction
PC=0x8 IR=0x40e133
R[0]=0x0 R[1]=0x0 R[2]=0xffff0000 R[3]=0x0 R[4]=0xffff0000 R[5]=0x0 R[6]=0x0 R[7]=0x0 R
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
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
x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Continue simulation (Y/n)? [Y]
```

12. SLTIU

SLTIU：和 SLTI 功能一致，但处理的是无符号数

```
case SLTIU:
    cout << "Do SLTIU" << endl;
    if(src1<(unsigned int)Imm11_0ItypeSignExtended)
        R[rd] = 1;
    else
        R[rd] = 0;
    break;
```

writeWord(4, (1<<20) | (5<<15) | (3<<12) | (2<<7) | (ALUIMM));//SLTIU

测试结果：

```
PC=0x8 IR=0x12b113
R[0]=0x0 R[1]=0x0 R[2]=0x1 R[3]=0x0 R[4]=0xffff0000 R[5]=0x0 R[6]=0x0 R[7]=0x0 R[8]=
0 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]=0
 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[
]=0x0 R[1e]=0x0 R[1f]=0x0
Continue simulation (Y/n)? [Y]
```

13. SLT/SLTU

SLT/SLTU 分别用于执行有无符号数的比较 if rs1<rs2）rd=1;else rd=0;

同样，STLU rd,x0,rs2 用来判定 rs2 是否为 0.

```
case SLT:
    cout << "Do SLT " << endl;
    if((int)src1<(int)src2){
        R[rd]=1;
    }else{
        R[rd]=0;
    }
    break;
case SLTU:
    cout << "Do SLTU" << endl;
    if(src2!=0){
        R[rd]=1;
    }else{
        R[rd]=0;
    }
    break;
```

writeWord(4, (0<<25) | (4<<20) | (1<<15) | (3<<12) | (3<<7) | (ALURRR));//SLTU

测试结果：

```
Do SLTU
Registers after executing the instruction
PC=0x8 IR=0x40b1b3
R[0]=0x0 R[1]=0x0 R[2]=0x0 R[3]=0x1 R[4]=0xffff0000 R[5]=0x0 R[6]=0x0 R[7]=0x0 R[8]=0x0 R[9]=0x
0 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]
```