

实验报告

RISC-V 基本指令集模拟器设计与实现

智能 1501 班 201508010426 丁增

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](#)，其中共有_47_条指令。

所需写入的指令为 LHU，XOR，SRL

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 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。

具体指令编码内容如下

```

case LHU:
    cout << "Do LHU" << endl;
    R[rd] = readByte(Imm11_0ItypeSignExtended + src1) & 0x0000ffff;
    break;
    default:
        cout << "ERROR: Unknown funct3 in LOAD instruction " <<
        IR << endl;
    }
    break;
case XOR:
    cout << "Do XOR " << endl;
    R[rd] = R[rs1] ^ R[rs2];
    break;
case SRL:
    cout << "DO SRL" << 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	
CPU	core i5-6500U	
内存	DDR3 4GB	
操作系统	Windows 10	

为了验证程序的可行性，在模拟器程序里面添加了测试程序,详情如下

```
writeWord(4, (0x00<<25) | (2<<20) | (1<<15) | (XOR << 12) | (3 << 7) | (ALURRR));
```

```
writeWord(16, (0x400<<20) | (1<<15) | (LHU<<12) | (3<<7) | (LOAD));
```

```
writeWord(20, (0x00<<25) | (2<<20) | (1<<15) | (SRL<< 12) | (0<< 7) | (ALURRR));
```

指令作用分别为：将地址为 R[rs1]的通用寄存器的值，与地址为 R[rs2]的通用寄存器的值进行逻辑“异或”运算，运算结果保存到地址为 R[rd]的通用寄存器中

LHU 从内存中指定的加载地址处，读取一个字，保存到地址为 R[rs1]的通用寄存器中。该指令有地址对齐要求，要求加载地址的最低两位为 00

SRL 将地址为 rt 的通用寄存器的值，向 R[rs1]右移，空出来的位置使用 0 填充，结果保存到地址为 R[rd]的通用寄存器中。

测试记录

模拟器运行过程的截图如下：

第一条指令 XOR 运行后模拟器的输出

```

Registers before executing the instruction 00x0
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 XOR
Registers after executing the instruction
PC=0x4 IR=0x20c1b3
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
Continue simulation (Y/n)? [Y]

```

第二条指令 LHU 运行后模拟器的输出

```

D:\uo10\cpudesign\2018\simulator\cpusim.exe
0x0 R[1a]=0x0 R[1b]=0x0 R[1c]=0x0 R[1d]=0x0 R[1e]=0x0 R[1f]=0x0
Do LHU
Registers after executing the instruction
PC=0x10 IR=0x4000d183
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
Continue simulation (Y/n)? [Y]
Registers before executing the instruction 00x10
PC=0x10 IR=0x4000d183
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

```

第三条指令 SRL 运行后模拟器的输出

```

DO SRL
Registers after executing the instruction
PC=0x14 IR=0x20d1b3
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
Continue simulation (Y/n)? [Y]

```

分析和结论

从测试记录来看，模拟器实现了对二进制指令的读入，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。

