版本

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **更新记录** | 文档名 | | 实验指导书\_lab2.1 | |
| 版本号 | | 0.4 | |
| 创建人 | | 计算机组成原理教学组 | |
| 创建日期 | | 2022/1/1 | |
| **更新历史** | | | | |
| **序号** | **更新日期** | **更新人** | **版 号** | **更新内容** |
| 1 | 2022/1/1 | 陈颖琪 | 0.1 | RISC-V CPU模型虚拟仿真 |
| 2 | 2022/2/24 | 陈颖琪 | 0.2 | 仿真平台说明，添加ripes |
| 3 | 2022/3/29 | 陈颖琪 | 0.3 | 修正了表1中的两处错误 |
| 4 | 2023/3/14 | 陈颖琪 | 0.4 | 仿真平台3略 |

**文档错误反馈:**

本文档出现错误请联系：

Yingqichen@sjtu.edu.cn

### 实验二 之 RISC-V CPU虚拟仿真部分

### 1、实验目的

1、熟悉指令和汇编代码。

2、产生指令存储器初始值文件。

## 2、实验要求

### 2.1 仿真平台

**仿真软件1**：Ripes v2.2.4

RISC-V 32/64b单周期及流水线架构CPU模型仿真软件,下载链接：

<https://jbox.sjtu.edu.cn/l/B1IA1R>

本部分实验可以此平台为主。

**仿真平台2**：WebRISCV

RISC\_V 32/64b 五级流水线架构CPU模型在线仿真平台，主要用于理解流水线执行过程。在后续的lab5实验中也会使用。

平台链接：<http://10.119.1.50:81/> （校内网，校外访问需连接交大VPN）

或 <https://webriscv.dii.unisi.it/index.php> （备用）

**两个仿真平台均可使用。**

### 2.2 仿真测试代码段

本次实验用到如下汇编代码段，可直接于仿真平台1、2输入用，供参考。

lui x10, 0

ori x4, x10, 0

addi x25, x0, 1

addi x26, x0, 2

addi x27, x0, 3

addi x28, x0, 4

sw x25, 0(x4)

sw x26, 4(x4)

sw x27, 8(x4)

sw x28, 12(x4)

addi x5, x0, 4

call:

jal sum

sw x12, 0(x4)

lw x19, 0(x4)

sub x18, x19, x12

addi x5, x0, 3

loop2:

addi x5, x5, -1

ori x18, x5, -1

xori x18, x18, 1365

addi x19, x0, -1

andi x20, x19, -1

or x16, x20, x19

xor x18, x20, x19

and x17, x20, x16

beq x5, x0, shift

j loop2

shift:

addi x5, x0, -1

slli x18, x5, 15

slli x18, x18, 16

srai x18, x18, 16

srli x18, x18, 15

fi:

j fi

sum:

add x18, x0, x0

loop:

lw x19, 0(x4)

addi x4, x4, 4

add x18, x18, x19

addi x5, x5, -1

bne x5, x0, loop

slli x12, x18, 0

jr ra

**2.3虚拟仿真步骤与要求**

**补全表格理解代码段含义。**

1. 选择仿真平台1（也可选择其他平台，做对比）输入上述测试代码段；
2. 设置单周期CPU模式；
3. 通过单步执行仿真理解代码执行的操作，得出其每条指令机器码，补全表1中对指令的描述，或者指令执行后的输出，填入表格1，补全后四列（红色字体）内容；
4. 提交完整表格。

表1 测试用代码段功能描述表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Address(HEX) | 标号 | 仿真平台2输入代码（这样输入才不会出错） | 仿真平台2显示代码 | 机器码（BIN） | 机器码（HEX） | 指令类型 | 描述 |
| 0 |  | lui x10, 0 | lui a0, 0 | 1.1E+23 | 00000537 | U | #initialize x10 =base address 0 |
| 4 |  | ori x4, x10, 0 | oritp, a0, 0 | 1E+17 | 00056213 | I | #x4<- base address x10 + offset 0 =0 |
| 8 |  | addi x25, x0, 1 | addi s9, x0, 1 | 1E+30 | 00100c93 | R | #initialize x25 = 1 |
| c |  | addi x26, x0, 2 | addi s10, x0, 2 | 1.1E+21 | 00200d13 | R |  |
| 10 |  | addi x27, x0, 3 | addi s11, x0, 3 | 1.11E+31 | 00300d93 | R |  |
| 14 |  | addi x28, x0, 4 | addi t3, x0, 4 | 1.11E+31 | 00400e13 | R |  |
| 18 |  | sw x25, 0(x4) | sw s9, 0(tp) | 1.01E+30 | 01922023 | S | #[0] = 1 |
| 1c |  | sw x26, 4(x4) | sw s10, 4(tp) | 1.11E+31 | 01a22223 | S | # [4] = 2 |
| 20 |  | sw x27, 8(x4) | sw s11, 8(tp) | 1.11E+31 | 01b22423 | S |  |
| 24 |  | sw x28, 12(x4) | sw t3, 12(tp) | 1E+24 | 01c22623 | S |  |
| 28 |  | addi x5, x0, 4 | addi t0, x0, 4 | 1E+24 | 00400293 | R | # x5 = 4，循环次数 |
| 2c | Call： | Call: jal sum | jalra, 128 | 1E+24 | 054000ef | I | # call function sum  跳转到 pc = 80 |
| 30 |  | sw x12, 0(x4) | sw a2, 0(tp) | 1.01E+17 | 00c22023 | S | #[16] <- 0x0000000a  (x12=0x0000000a) |
| 34 |  | lw x19, 0(x4) | lw s3, 0(tp) | 1.11E+31 | 00022983 | I | #x19<- [16] （0x10）  ([16]=0x0000000a) |
| 38 |  | sub x18, x19, x12 | sub s2, s3, a2 | 1.11E+31 | 40c98933 | R | #x18= 0 |
| 3c |  | addi x5, x0, 3 | addi t0, x0, 3 | 1.11E+23 | 00300293 | R | #x5=3 |
| 40 | loop2: | loop2:addi x5, x5, -1 | addi t0, t0, -1 | 1E+24 | Fff28293 | R |  |
| 44 |  | ori x18, x5, -1 | ori s2, t0, -1 | 1E+30 | Fff2e913 | I | #x18= 0xffffffff ，（x18 = x5 or 12bit 立即数有符号扩展0xffffffff） |
| 48 |  | xori x18, x18, 1365 | xori s2, s2, 1365 | 1.11E+23 | 55594913 | R | #X18=0xfffffaaa |
| 4c |  | addi x19, x0, -1 | addi s3, x0, -1 | 1101111 | Fff00993 | R | #X19=0xffffffff |
| 50 |  | andi x20, x19, -1 | andi s4, s3, -1 | 1E+11 | Fff9fa13 | R | #X20=0xffffffff，（X20=0xffffffff and 0xffffffff） |
| 54 |  | or x16, x20, x19 | or a6, s4, s3 | 1E+17 | 013a6833 | R | #X16=0xffffffff |
| 58 |  | xor x18, x20, x19 | xor s2, s4, s3 | 1E+22 | 013a4933 | R | #X18=0 |
| 5c |  | and x17, x20, x16 | and a7, s4, a6 | 1E+24 | 010a78b3 | R | #X17=0xffffffff |
| 60 |  | beq x5, x0, shift | beq t0, x0, 104 | 1.11E+31 | 00028463 | SB | #Ifx5 = 0  Goto shift after finished loop2 4 times，goto pc= 68 |
| 64 |  | j loop2 | jal x0, 64 | 1.11E+31 | Fddff06f | I | #Loop Loop2 for 4 times，goto pc=40 |
| 68 | shift: | shift:addi x5, x0, -1 | addi t0, x0, -1 | 1E+19 | Fff00293 | R | #X5=0xffffffff |
| 6c |  | slli x18, x5, 15 | slli s2, t0, 15 | 1E+15 | 00f29913 | I | #X18=0xffff8000 |
| 70 |  | slli x18, x18, 16 | slli s2, s2, 16 | 1.1E+23 | 01091913 | I | #X18=0x80000000 |
| 74 |  | srai x18, x18, 16 | srai s2, s2, 16 | 1E+17 | 41095913 | I | #X18=0xffff8000 |
| 78 |  | srli x18, x18, 15 | srli s2, s2, 15 | 1E+30 | 00f95913 | I | #X18=0x0001ffff |
| 7c | finish: | finish:j finish | jal x0, 124 | 1.1E+21 | 0000006f | I | #Endhere |
| 80 | sum: | sum:add x18, x0, x0 | add s2, x0, x0 | 1.11E+31 | 00000933 | R | #X18 = 0 |
| 84 | loop: | loop:lw x19, 0(x4) | lw s3, 0(tp) | 1.11E+31 | 00022983 | I | #X19 <- [x4] |
| 88 |  | addi x4, x4,4 | additp, tp,4 | 1.01E+30 | 00420213 | R | #x4 <-(x4+4) |
| 8c |  | add x18, x18, x19 | add s2, s2, s3 | 1.11E+31 | 01390933 | R | #X18= x18 + [x4],  X18= 0x0000000a |
| 90 |  | addi x5, x5, -1 | addi t0, t0, -1 | 1.11E+31 | Fff28293 | R | #x5 <- (x5-1)，循环次数-1 |
| 94 |  | bne x5, x0, loop | bne t0, x0, 132 | 1E+24 | Fe0298e3 | SB | #loop循环累加 次，结果存于： x18 |
| 98 |  | slli x12, x18, 0 | slli a2, s2, 0 | 1E+24 | 00091613 | I | #X12<- x18 ，X12 = 0x0000000a，函数调用结果存于： x12 |
| 9c |  | Jr ra | jalr x0, 0(ra) | 1E+24 | 00008067 | I | #函数sum调用返回，回到pc = 30 |

### 3 指令存储器IP例化初始化文件产生

完成表1后，即可产生vivado FPGA开发平台指令存储器IP例化时的初始化文件：sc\_irom.coe。以便为后续CPU核设计提供必要的支持。该文件为文本格式，但是以coe作为后缀。格式如下。第一行表示机器码是16进制格式。第二行等号后开始列出代码段的机器码，可以用空格或者回车符分隔每条指令最终以分号（半角字符）结束。

memory\_initialization\_radix=16;

memory\_initialization\_vector=00000537

00056213

00100c93

。

。

。

。

。

；

将前面补全的表1中的16进制机器码逐行写入，存为sc\_irom.coe。以备后续实验使用。你也可以通过其他汇编器产生机器码后自己编写小程序自动存为该格式文件。

**4 思考**

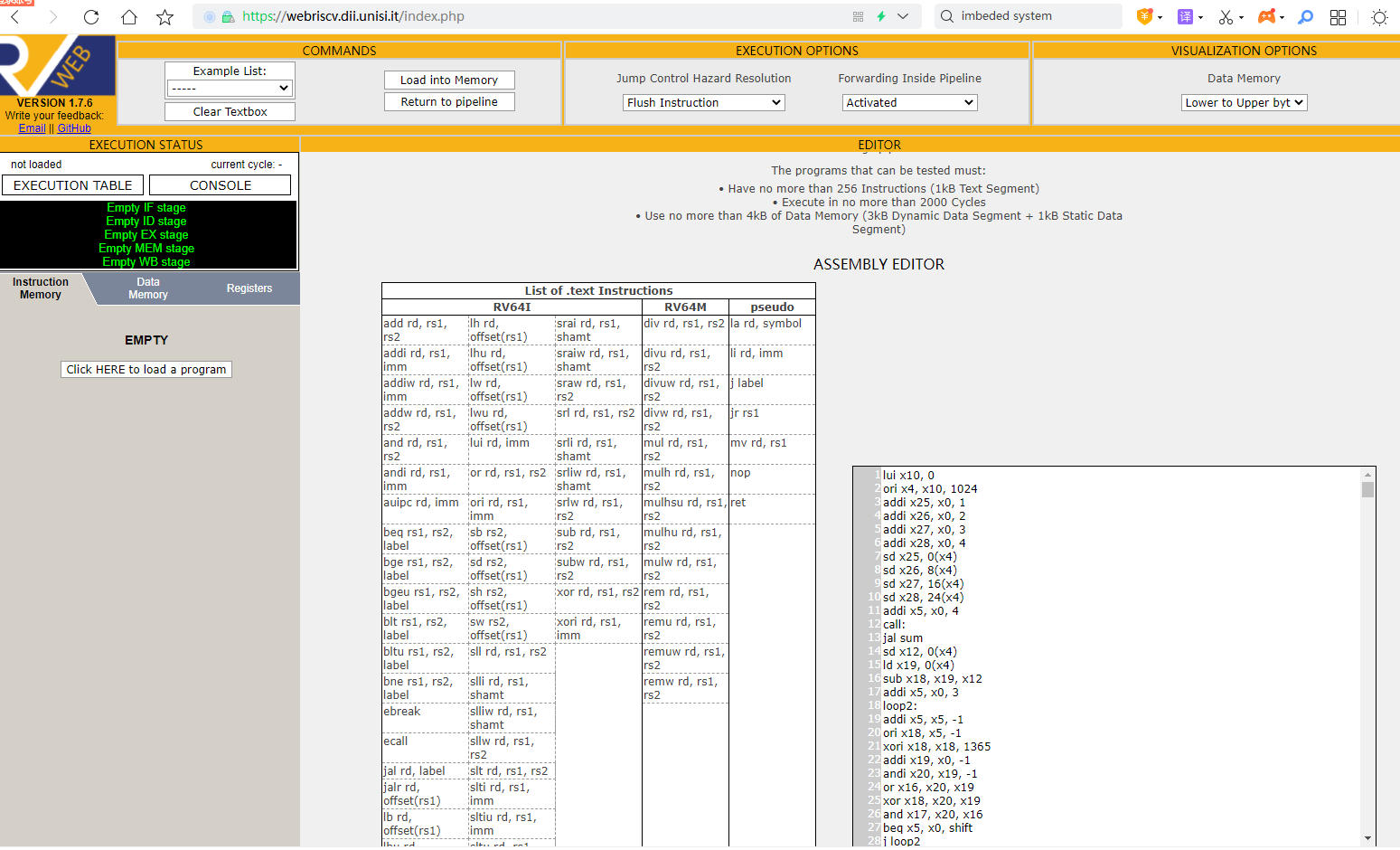
以上代码段最开始将数据1，2，3，4写到了哪些存储器地址单元?可以查看memory页面，找到被更新的单元，作截图。

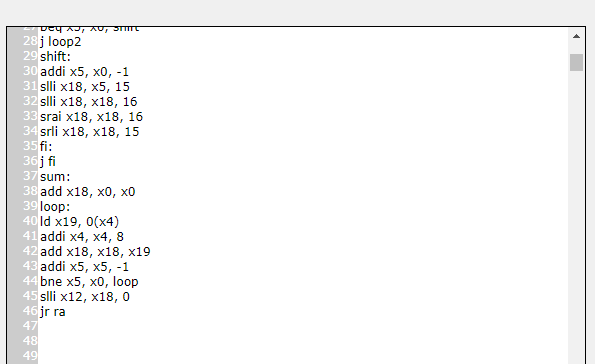
Ripes软件中是将.data段起始地址默认设定为0x10000000的。如果用Ripes来仿真，为了便于在 .Data段直接观测，如何修改测试程序的前两句，才可以将1，2，3，4写到0x10000000为基址的 .Data段起始位置? 修改后完成仿真，查看memory页面，找到.data段的起始位置，看看是否被改写?从0x10000000开始的几个单元是否被更新?理解对应的语句功能含义。

如果使用仿真平台2 完成以上仿真的话直接用该代码段是否可以? 提示：程序加载与仿真界面截图，如图2和图3。如果在仿真平台2上进行仿真，第2句可改为ori x4, x10, 1024，其他不变。因为该平台数据存储器（data段）设置的偏移量为地址1024处开始。为便于在界面直接观察数据存储器内的变化结果，可以做此修改。

**5 附录**

仿真平台2测试程序的加载界面截图示意如图2、图3 所示，供参考。



 图2

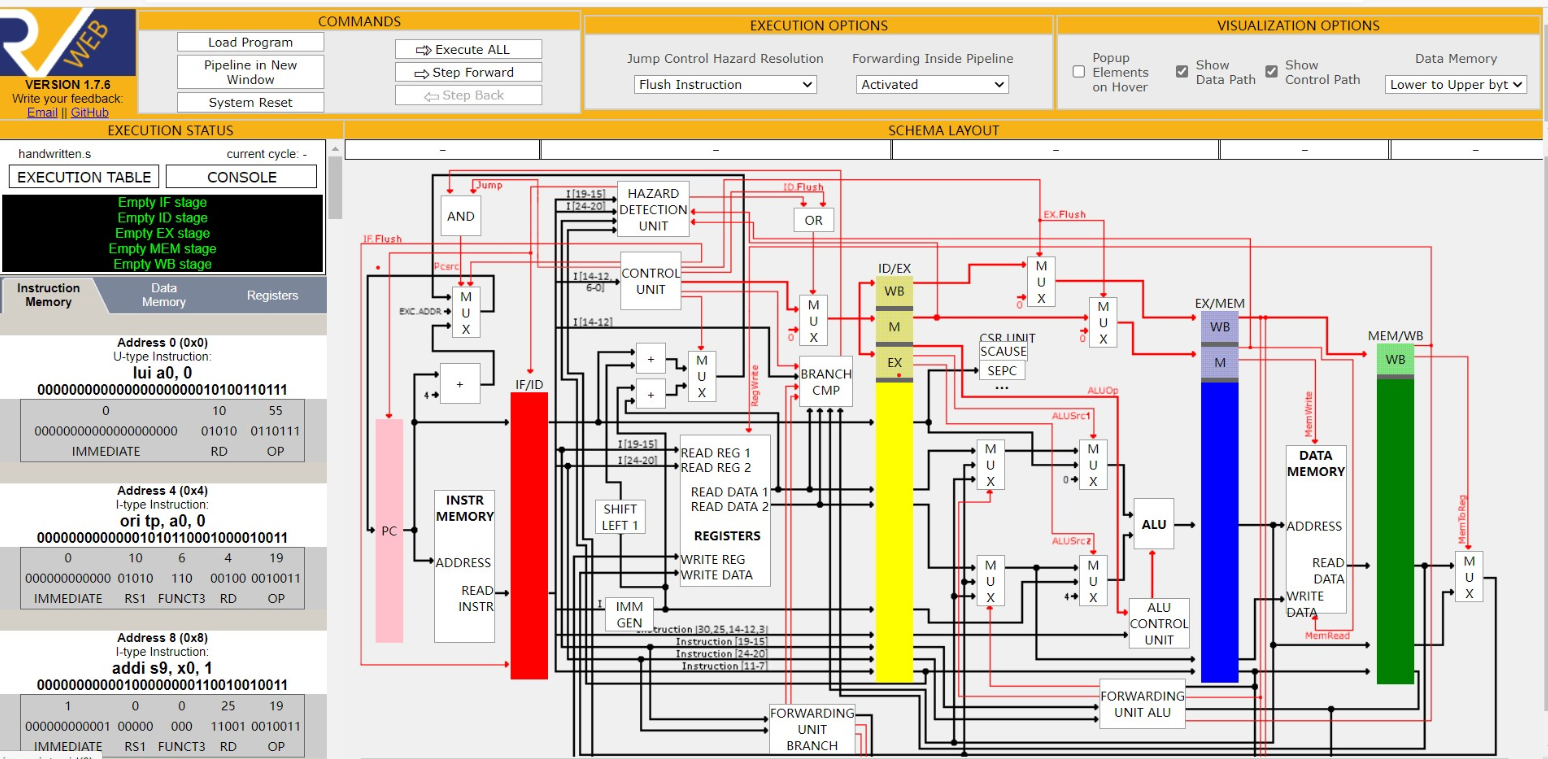


图3

仿真平台2测试程序加载后的指令和机器码截图，供参考。