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 本号**	**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		00000537		#initialize x10 =base address 0
4		ori x4, x10, 0	oritp, a0, 0		00056213		#x4<- base address x10 + offset 0 =0
8		addi x25, x0, 1	addi s9, x0, 1		00100c93	I	#initialize x25 = 1
c		addi x26, x0, 2	addi s10, x0, 2				
10		addi x27, x0, 3	addi s11, x0, 3				
14		addi x28, x0, 4	addi t3, x0, 4				
18		sw x25, 0(x4)	sw s9, 0(tp)				#[0] = 1
1c		sw x26, 4(x4)	sw s10, 4(tp)				# [4] = 2
20		sw x27, 8(x4)	sw s11, 8(tp)				
24		sw x28, 12(x4)	sw t3, 12(tp)				
28		addi x5, x0, 4	addi t0, x0, 4				# x5 = 4，循环次数
2c	Call：	Call: jal sum	jalra, 128				# call function sum 跳转到 pc = 80
30		sw x12, 0(x4)	sw a2, 0(tp)				#[10] <- 0x0000000a (x12=0x0000000a)
34		lw x19, 0(x4)	lw s3, 0(tp)				#x19<- [10] ([10]=0x0000000a)
38		sub x18, x19, x12	sub s2, s3, a2			R	#x18= 0
3c		addi x5, x0, 3	addi t0, x0, 3				#x5=3
40	loop2:	loop2:addi x5, x5, -1	addi t0, t0, -1				
44		ori x18, x5, -1	ori s2, t0, -1				#x18= 0xffffffff ，（x18 = x5 or 12bit 立即数

							有符号扩展 0xffffffff）
48		xori x18, x18, 1365	xori s2, s2, 1365				#X18=
4c		addi x19, x0, -1	addi s3, x0, -1				#X19=
50		andi x20, x19, -1	andi s4, s3, -1				#X20=0xffffffff ，（ X20=0xffffffff and 0xffffffff）
54		or x16, x20, x19	or a6, s4, s3				#X16=
58		xor x18, x20, x19	xor s2, s4, s3				#X18=
5c		and x17, x20, x16	and a7, s4, a6				#X17=
60		beq x5, x0, shift	beq t0, x0, 104				#Ifx5 = 0 Goto shift after finished loop2 4 times，goto pc= 68
64		j loop2	jal x0, 64				#Loop Loop2 for 4 times，goto pc=40
68	shift:	shift:addi x5, x0, -1	addi t0, x0, -1				#X5=0xffffffff
6c		slli x18, x5, 15	slli s2, t0, 15				#X18=0xffff8000
70		slli x18, x18, 16	slli s2, s2, 16				#X18=
74		srai x18, x18, 16	srai s2, s2, 16		41095913		#X18=
78		srli x18, x18, 15	srli s2, s2, 15				#X18=
7c	finish:	finish:j finish	jal x0, 124				#Endhere
80	sum:	sum:add x18, x0, x0	add s2, x0, x0				#X18 =
84	loop:	loop:lw x19, 0(x4)	lw s3, 0(tp)				#X19 <- [x4]
88		addi x4, x4,4	additp, tp,4				#x4 <-
8c		add x18, x18, x19	add s2, s2, s3				#X18= x18 + [x4], x18=
90		addi x5, x5, -1	addi t0, t0, -1				#x5 <- (x5-1)，循环次数-1
94		bne x5, x0, loop	bne t0, x0, 132			SB	#loop 循环累加次，结果存于：

98		slli x12, x18, 0	slli a2, s2, 0			I	#X12<- x18 ， X12 = 0x0000000a，函数调用结果存于：
9c		Jr ra	jalr x0, 0(ra)			I	#函数 sum 调用返回，回到 pc =

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

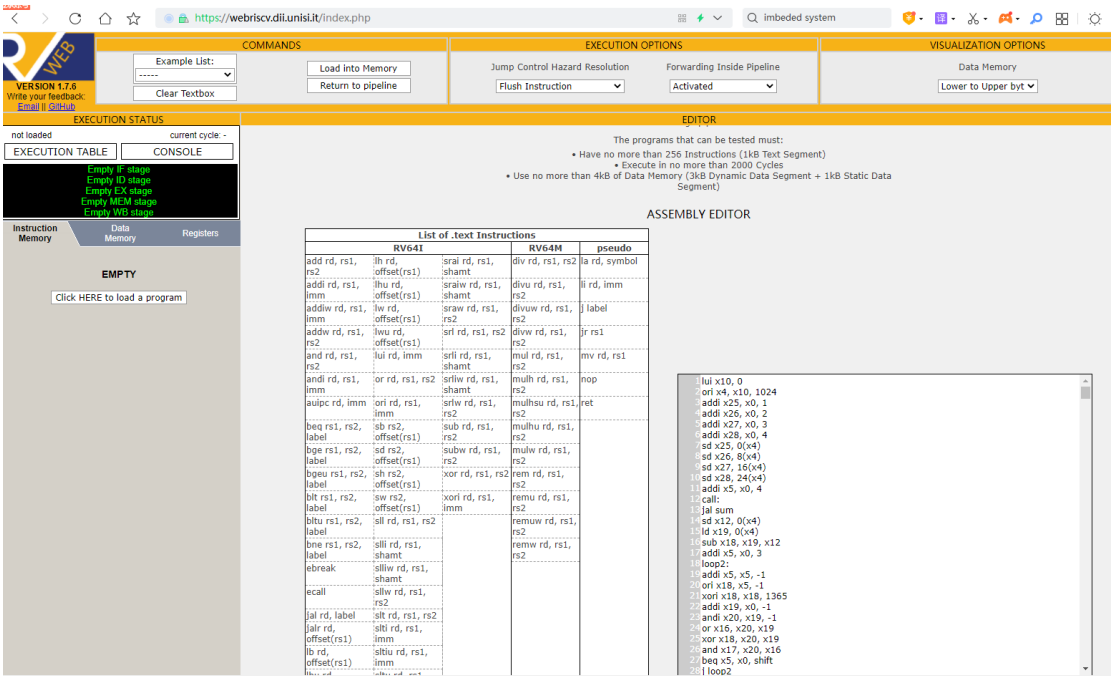
以上代码段最开始将数据 1，2，3，4 写到了哪些存储器地址单元?可以查看 memory 页面，找到被更新的单元，作截图。

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。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。

再进一步，如果用前一个平台来仿真第二句修改后的（也即 ori x4, x10, 1024），查看 memory 页面，找到被更新的单元的地址并与之前的截图作对比，理解对应的语句功能含义。

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。




```
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:
ld x19, 0(x4)
addi x4, x4, 8
add x18, x18, x19
addi x5, x5, -1
bne x5, x0, loop
slli x12, x18, 0
jr ra
```

图 2

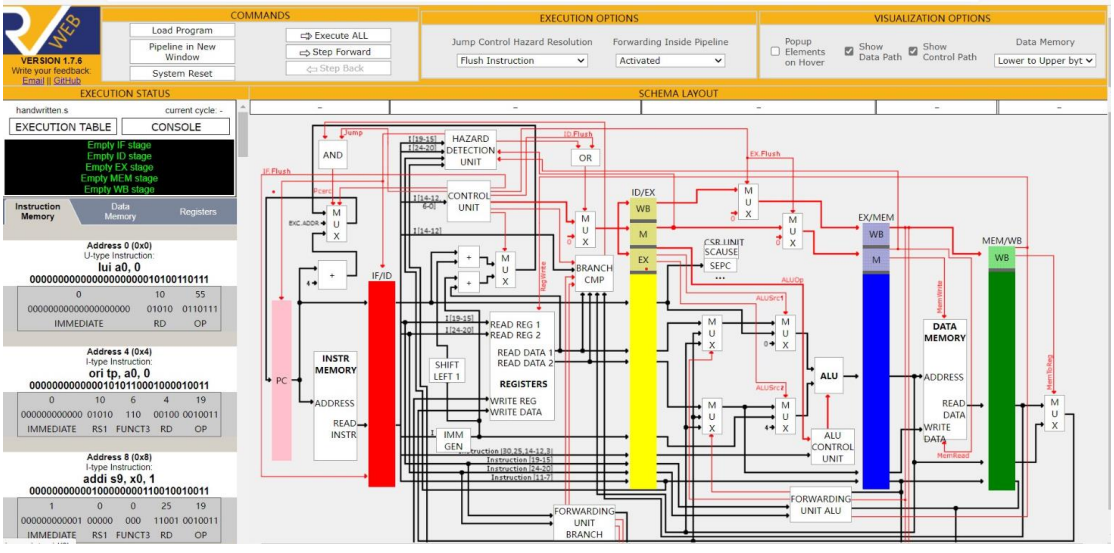
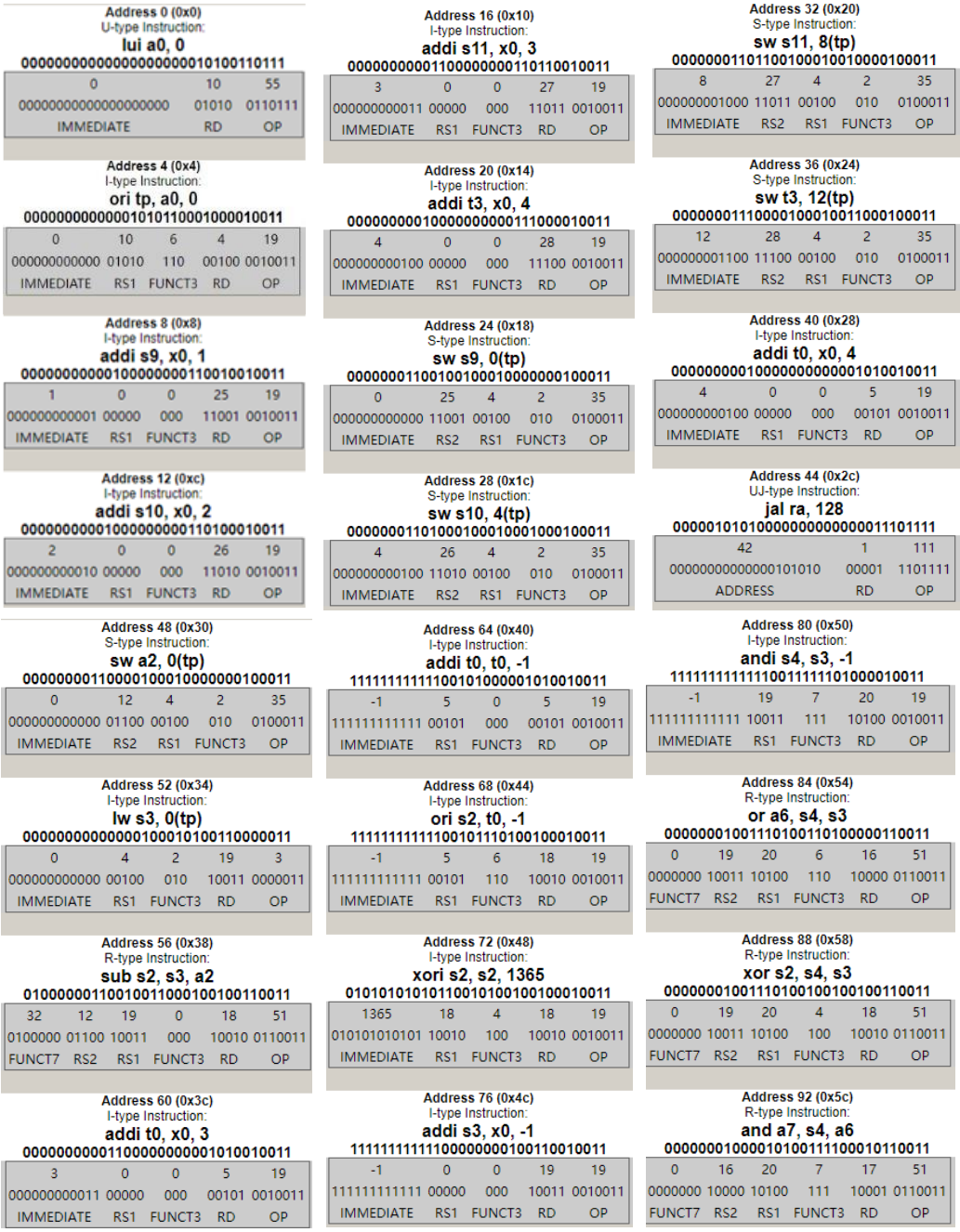


图 3

仿真平台 2 测试程序加载后的指令和机器码截图，供参考。



Address 96 (0x60)
SB-type Instruction:

beq t0, x0, 104

00000000000000010100001000110011

4	0	5	0	99
000000000100	00000	00101	000	1100011
IMMEDIATE	RS2	RS1	FUNCT3	OP

Address 100 (0x64)
UJ-type Instruction:

jal x0, 64

11111101110111111111000001101111

-18		0	111
11111111111111101110		00000	1101111
ADDRESS		RD	OP

Address 104 (0x68)
I-type Instruction:

addi t0, x0, -1

11111111111100000000001010010011

-1	0	0	5	19
111111111111	00000	000	00101	0010011
IMMEDIATE	RS1	FUNCT3	RD	OP

Address 108 (0x6c)
I-type Instruction:

slli s2, t0, 15

00000000111100010100110010010011

15	5	1	18	19
000000001111	00101	001	10010	0010011
IMMEDIATE	RS1	FUNCT3	RD	OP

Address 112 (0x70)
I-type Instruction:

slli s2, s2, 16

00000001000010010001100100010011

16	18	1	18	19
000000010000	10010	001	10010	0010011
IMMEDIATE	RS1	FUNCT3	RD	OP

Address 116 (0x74)
I-type Instruction:

srai s2, s2, 16

10000001000010010101100100010011

-2032	18	5	18	19
100000010000	10010	101	10010	0010011
IMMEDIATE	RS1	FUNCT3	RD	OP

Address 120 (0x78)
I-type Instruction:

srli s2, s2, 15

00000000111110010101100100010011

15	18	5	18	19
000000001111	10010	101	10010	0010011
IMMEDIATE	RS1	FUNCT3	RD	OP

Address 124 (0x7c)
UJ-type Instruction:

jal x0, 124

00000000000000000000000001101111

0		0	111
00000000000000000000		00000	1101111
ADDRESS		RD	OP

Address 128 (0x80)
R-type Instruction:

add s2, x0, x0

00000000000000000000100100110011

0	0	0	0	18	51
0000000	00000	00000	000	10010	0110011
FUNCT7	RS2	RS1	FUNCT3	RD	OP

Address 132 (0x84)
I-type Instruction:

lw s3, 0(tp)

00000000000000010001010011000011

0	4	2	19	3
000000000000	00100	010	10011	0000011
IMMEDIATE	RS1	FUNCT3	RD	OP

Address 136 (0x88)
I-type Instruction:

addi tp, tp, 4

00000000010000100000001000010011

4	4	0	4	19
000000000100	00100	000	00100	0010011
IMMEDIATE	RS1	FUNCT3	RD	OP

Address 140 (0x8c)
R-type Instruction:

add s2, s2, s3

00000001001110010000100100110011

0	19	18	0	18	51
0000000	10011	10010	000	10010	0110011
FUNCT7	RS2	RS1	FUNCT3	RD	OP

Address 144 (0x90)
I-type Instruction:

addi t0, t0, -1

11111111111100101000001010010011

-1	5	0	5	19
111111111111	00101	000	00101	0010011
IMMEDIATE	RS1	FUNCT3	RD	OP

Address 148 (0x94)
SB-type Instruction:

bne t0, x0, 132

11111110000000101001100011100011

-8	0	5	1	99
111111111000	00000	00101	001	1100011
IMMEDIATE	RS2	RS1	FUNCT3	OP

Address 152 (0x98)
I-type Instruction:

slli a2, s2, 0

00000000000010010001011000010011

0	18	1	12	19
000000000000	10010	001	01100	0010011
IMMEDIATE	RS1	FUNCT3	RD	OP

Address 156 (0x9c)
I-type Instruction:

jalr x0, 0(ra)

00000000000000001000000001100111

0	1	0	0	103
000000000000	00001	000	00000	1100111
IMMEDIATE	RS1	FUNCT3	RD	OP