

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/I/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=0xffffffffaaa
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	S B	#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 = 0x0001ffff
84	loop: p:	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= 0x0001ffff
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	S B	#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 = [ra]

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。

Address 0 (0x0)	Address 16 (0x10)	Address 32 (0x20)
U-type Instruction: lui a0, 0	I-type Instruction: addi s11, x0, 3	S-type Instruction: sw s11, 8(tp)
00000000000000000000000000000000	00000000011000000001100110011001	000000011011001000100110000100011
0 10 55	3 0 0 27 19	8 27 4 2 35
00000000000000000000000000000000	000000000001100000 000 11011 0010011	000000001000 11011 00100 010 0100011
IMMEDIATE RD OP	IMMEDIATE RS1 FUNCT3 RD OP	IMMEDIATE RS2 RS1 FUNCT3 OP
Address 4 (0x4)	Address 20 (0x14)	Address 36 (0x24)
I-type Instruction: ori tp, a0, 0	I-type Instruction: addi t3, x0, 4	S-type Instruction: sw t3, 12(tp)
00000000000001010110001000010011	00000000010000000000111000010011	00000001110000100010011000100011
0 10 6 4 19	4 0 0 28 19	12 28 4 2 35
000000000000 01010 110 00100 0010011	000000000100 00000 000 11100 0010011	000000001100 11100 00100 010 0100011
IMMEDIATE RS1 FUNCT3 RD OP	IMMEDIATE RS1 FUNCT3 RD OP	IMMEDIATE RS2 RS1 FUNCT3 OP
Address 8 (0x8)	Address 24 (0x18)	Address 40 (0x28)
I-type Instruction: addi s9, x0, 1	S-type Instruction: sw s9, 0(tp)	I-type Instruction: addi t0, x0, 4
000000000001000000000110010010011	000000001100100100010000000100011	00000000010000000000001010010011
1 0 0 25 19	0 25 4 2 35	4 0 0 5 19
000000000001 00000 000 11001 0010011	000000000000 11001 00100 010 0100011	000000000100 00000 000 00101 0010011
IMMEDIATE RS1 FUNCT3 RD OP	IMMEDIATE RS2 RS1 FUNCT3 OP	IMMEDIATE RS1 FUNCT3 RD OP
Address 12 (0xc)	Address 28 (0x1c)	Address 44 (0x2c)
I-type Instruction: addi s10, x0, 2	S-type Instruction: sw s10, 4(tp)	I-type Instruction: jal ra, 128
000000000100000000000110100010011	00000000110100100010001000100011	000001010100000000000000011010111
2 0 0 26 19	4 26 4 2 35	42 1 111
000000000010 00000 000 11010 0010011	000000000100 11010 00100 010 0100011	000000000000000101010 00001 1101111
IMMEDIATE RS1 FUNCT3 RD OP	IMMEDIATE RS2 RS1 FUNCT3 OP	ADDRESS RD OP
Address 48 (0x30)	Address 64 (0x40)	Address 80 (0x50)
S-type Instruction: sw a2, 0(tp)	I-type Instruction: addi t0, t0, -1	I-type Instruction: andi s4, s3, -1
0000000011000010001000000000010011	1111111111100101000001010010011	1111111111110011111010000010011
0 12 4 2 35	-1 5 0 5 19	-1 19 7 20 19
000000000000 01100 00100 010 0100011	111111111111 00101 000 00101 0010011	111111111111 10011 111 10100 0010011
IMMEDIATE RS2 RS1 FUNCT3 OP	IMMEDIATE RS1 FUNCT3 RD OP	IMMEDIATE RS1 FUNCT3 RD OP
Address 52 (0x34)	Address 68 (0x44)	Address 84 (0x54)
I-type Instruction: lw s3, 0(tp)	I-type Instruction: ori s2, t0, -1	R-type Instruction: or a6, s4, s3
0000000000000001000101001100000011	111111111110101101001000100010011	00000001001110100110100000110011
0 4 2 19 3	-1 5 6 18 19	0 19 20 6 16 51
000000000000 00100 010 10011 0000011	111111111111 00101 110 10010 0010011	0000000 10011 10100 110 10000 0110011
IMMEDIATE RS1 FUNCT3 RD OP	IMMEDIATE RS1 FUNCT3 RD OP	FUNCT7 RS2 RS1 FUNCT3 RD OP
Address 56 (0x38)	Address 72 (0x48)	Address 88 (0x58)
R-type Instruction: sub s2, s3, a2	I-type Instruction: xori s2, s2, 1365	R-type Instruction: xor s2, s4, s3
010000001100100110001001001100011	01010101010110010100100100010011	000000001001110100100100100110011
32 12 19 0 18 51	1365 18 4 18 19	0 19 20 4 18 51
0100000 01100 10011 000 10010 0110011	010101010101 10010 100 10010 0010011	0000000 10011 10100 110 10010 0110011
FUNCT7 RS2 RS1 FUNCT3 RD OP	IMMEDIATE RS1 FUNCT3 RD OP	FUNCT7 RS2 RS1 FUNCT3 RD OP
Address 60 (0x3c)	Address 76 (0x4c)	Address 92 (0x5c)
I-type Instruction: addi t0, x0, 3	I-type Instruction: addi s3, x0, -1	R-type Instruction: and a7, s4, a6
000000000011000000000001010010011	11111111111100000000100110010011	00000001000010100111100010110011
3 0 0 5 19	-1 0 0 19 19	0 16 20 7 17 51
000000000011 00000 000 00101 0010011	111111111111 00000 000 10011 0010011	0000000 10010 10100 111 10001 0110011
IMMEDIATE RS1 FUNCT3 RD OP	IMMEDIATE RS1 FUNCT3 RD OP	FUNCT7 RS2 RS1 FUNCT3 RD OP

Address 96 (0x60)				
SB-type Instruction:				
beq t0, x0, 104				
0000000000000101000010001100011				
4	0	5	0	99
000000000100	00000	00101	000	1100011
IMMEDIATE	RS2	RS1	FUNCT3	OP

Address 100 (0x64)				
UJ-type Instruction:				
jal x0, 64				
111111011101111111110000001101111				
-18	0	111		
111111111111111101110	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				
00000000111100101001100100010011				
15	5	1	18	19
000000001111	00101	001	10010	0010011
IMMEDIATE	RS1	FUNCT3	RD	OP

Address 128 (0x80)				
R-type Instruction:				
add s2, x0, x0				
000000000000000000000100100110011				
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)				
000000000000000100010100110000011				
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				
000000000100001000000001000010011				
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 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				
1000000100001001010101100100010011				
-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				
0000000000000000000000000001101111				
0	0	111		
0000000000000000000000	00000	1101111		
ADDRESS		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
1111111111000	00000	00101	001	1100011
IMMEDIATE	RS2	RS1	FUNCT3	OP

Address 152 (0x98)				
I-type Instruction:				
slli a2, s2, 0				
0000000000000010010001011000010011				
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)				
00000000000000000000000001100111				
0	1	0	0	103
000000000000	00001	000	00000	1100111
IMMEDIATE	RS1	FUNCT3	RD	OP