

计算机组成原理 实验报告

姓名：宋玮 学号：PB20151793 实验日期：2022.4.3

一、实验题目：

Lab03 汇编程序设计

二、实验目的：

- 熟悉 RISC-V 汇编指令的格式
- 熟悉 CPU 仿真软件 Ripes，理解汇编指令执行的基本原理（数据通路和控制器的协调工作过程）
- 熟悉汇编程序的基本结构，掌握简单汇编程序的设计
- 掌握汇编仿真软件 RARS (RISC-V Assembler & Runtime Simulator) 的使用方法，会用该软件进行汇编程序的仿真、调试以及生成 CPU 测试需要的指令和数据文件（COE）

三、实验平台：

Ripes, Rars

四、实验过程及结果：

如下所示：

● 理解并仿真 RIPES 示例汇编程序

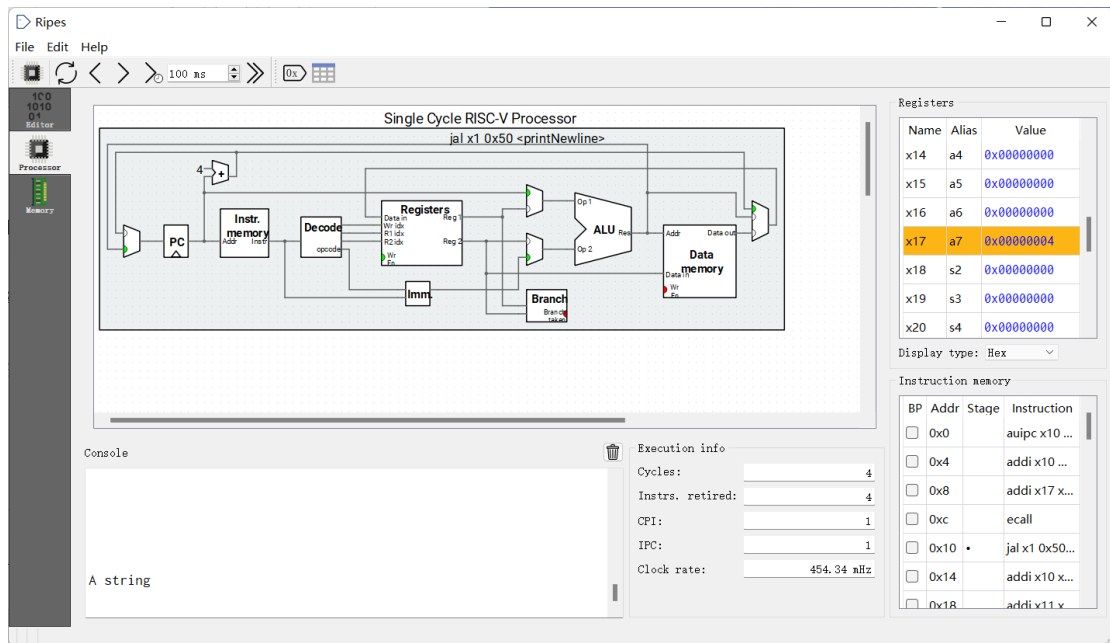
加载 Ripes 示例汇编程序 (Console Printing);

选择单周期 CPU 数据通路;

单步执行程序;

观察数据通路控制信号和寄存器内容的变化;

如下图：左边是数据通路控制信号，右边是寄存器内容；按“>”号即可单步执行。



● 用 rars 测试六条指令

1. 汇编程序 (test.asm) 如下:

```
.data
out: .word 0xff      #led, 初始全亮
in: .word 0xfe      #switch

.text
la a0, out          #仿真需要
sw x0, 0(a0)        #test sw: 全灭led
addi t0, x0, 0xff   #test addi: 全亮led
sw t0, 0(a0)
lw t0, 4(a0)        #test lw: 由switch设置led
sw t0, 0(a0)        #load in to out
addi t1, x0, 1      #test add
add t0, t1, t0      #t0=t0+t1
sw t0, 0(a0)        #load to out
addi t2, x0, 2
loop:
beq t2, x0, exit    #test beq
addi t2, t2, -1
sw t2, 0(a0)
jal x1, loop        #test jal
exit:
```

(1) test sw

D:\desktop\计算机实验3\test.asm - RARS 1.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x0400000	0x0f019517	auipc x10,0x0000fc10	6: la x0, out #所需寄存器
	0x0400004	0x00050513	addi x10,x10,0	
	0x0400008	0x00052023	sw x0,0(x10)	7: sw x0,0(x0) #test sw: 全亮led
	0x040000c	0x0ff02933	addi x5,x0,0x000000ff	8: addi t0,x0,0xff #test addi: 全亮led
	0x0400010	0x00052023	sw x5,0(x10)	9: sw t0,0(x0) #test lw: 由switch设置led
	0x0400014	0x00451283	lw x5,4(x10)	10: lw t0,4(x0)
	0x0400018	0x00052023	sw x5,0(x10)	11: sw t0,0(x0)
	0x040001c	0x00100313	addi x6,x0,1	12: addi t1,x0,1 #test add
	0x0400020	0x000302b3	add x5,x6,x5	13: add t0,t1,t0 #t0=t0+t1
	0x0400024	0x00052023	sw x0,0(x10)	14: sw t0,0(x0) #load to out
	0x0400028	0x00020393	addi x7,x0,2	15: addi t2,x0,2
	0x040002c	0x00038863	beq x7,x0,0x00000010	17: beq t2,x0,exit #test beq

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x000000ff	0x000000ff	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Labels

Label	Address
test.asm	
in	0x10010004
out	0x10010000
exit	0x040003c
loop	0x040003c

Registers

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0xffffffe0
gp	3	0x10008000
tp	4	0x10000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
a0	8	0x00000000
a1	9	0x00000000
a2	10	0x10000000
a3	11	0x00000000
a4	12	0x00000000
a5	13	0x00000000
a6	14	0x00000000
a7	15	0x00000000
s0	16	0x00000000
s1	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x04000008

Messages

Reset: reset completed.

m[out]变为 0

(2) test addi

D:\desktop\计算机实验3\test.asm - RARS 1.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x0400000	0x0f019517	auipc x10,0x0000fc10	6: la x0, out #所需寄存器
	0x0400004	0x00050513	addi x10,x10,0	
	0x0400008	0x00052023	sw x0,0(x10)	7: sw x0,0(x0) #test sw: 全亮led
	0x040000c	0x0ff02933	addi x5,x0,0x000000ff	8: addi t0,x0,0xff #test addi: 全亮led
	0x0400010	0x00052023	sw x5,0(x10)	9: sw t0,0(x0) #test lw: 由switch设置led
	0x0400014	0x00451283	lw x5,4(x10)	10: lw t0,4(x0)
	0x0400018	0x00052023	sw x5,0(x10)	11: sw t0,0(x0)
	0x040001c	0x00100313	addi x6,x0,1	12: addi t1,x0,1 #test add
	0x0400020	0x000302b3	add x5,x6,x5	13: add t0,t1,t0 #t0=t0+t1
	0x0400024	0x00052023	sw x0,0(x10)	14: sw t0,0(x0) #load to out
	0x0400028	0x00020393	addi x7,x0,2	15: addi t2,x0,2
	0x040002c	0x00038863	beq x7,x0,0x00000010	17: beq t2,x0,exit #test beq

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x000000fe	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Labels

Label	Address
test.asm	
in	0x10010004
out	0x10010000

Registers

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0xffffffe0
gp	3	0x10008000
tp	4	0x10000000
t0	5	0x000000ff
t1	6	0x00000000
t2	7	0x00000000
a0	8	0x00000000
a1	9	0x00000000
a2	10	0x10000000
a3	11	0x00000000
a4	12	0x00000000
a5	13	0x00000000
a6	14	0x00000000
a7	15	0x00000000
s0	16	0x00000000
s1	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x04000010

Messages

Reset: reset completed.

m[out]从 0 变为 0xff

(3) test lw

The screenshot shows the RARS 1.5 MIPS simulator. The main window displays the assembly code for the 'test lw' program. The registers window on the right shows the state of the MIPS registers. The 'test' register (t0) is highlighted in green, indicating it is the current register being used.

Register	Name	Number	Value
\$zero		0	0x00000000
\$ra		1	0x00000000
\$sp		2	0x7fffffc0
\$gp		3	0x10008000
\$tp		4	0x00000000
\$t0		5	0x00000000
\$t1		6	0x00000000
\$t2		7	0x00000000
\$t3		8	0x00000000
\$t4		9	0x00000000
\$t5		10	0x10010000
\$t6		11	0x00000000
\$t7		12	0x00000000
\$a0		13	0x00000000
\$a1		14	0x00000000
\$a2		15	0x00000000
\$a3		16	0x00000000
\$a4		17	0x00000000
\$a5		18	0x00000000
\$a6		19	0x00000000
\$a7		20	0x00000000
\$s0		21	0x00000000
\$s1		22	0x00000000
\$s2		23	0x00000000
\$s3		24	0x00000000
\$s4		25	0x00000000
\$s5		26	0x00000000
\$s6		27	0x00000000
\$s7		28	0x00000000
\$s8		29	0x00000000
\$s9		30	0x00000000
\$s10		31	0x00000000
\$pc			0x00400018

t0=m[in]=0xfe

随后将 t0 存入 m[out]

(4) test add

The screenshot shows the RARS 1.5 MIPS simulator. The main window displays the assembly code for the 'test add' program. The registers window on the right shows the state of the MIPS registers. The 'test' register (t0) is highlighted in green, indicating it is the current register being used.

Register	Name	Number	Value
\$zero		0	0x00000000
\$ra		1	0x00000000
\$sp		2	0x7fffffc0
\$gp		3	0x10008000
\$tp		4	0x00000000
\$t0		5	0x000000fe
\$t1		6	0x00000001
\$t2		7	0x00000000
\$t3		8	0x00000000
\$t4		9	0x00000000
\$t5		10	0x10010000
\$t6		11	0x00000000
\$t7		12	0x00000000
\$a0		13	0x00000000
\$a1		14	0x00000000
\$a2		15	0x00000000
\$a3		16	0x00000000
\$a4		17	0x00000000
\$a5		18	0x00000000
\$a6		19	0x00000000
\$a7		20	0x00000000
\$s0		21	0x00000000
\$s1		22	0x00000000
\$s2		23	0x00000000
\$s3		24	0x00000000
\$s4		25	0x00000000
\$s5		26	0x00000000
\$s6		27	0x00000000
\$s7		28	0x00000000
\$s8		29	0x00000000
\$s9		30	0x00000000
\$s10		31	0x00000000
\$pc			0x00400024

t0=0xfe

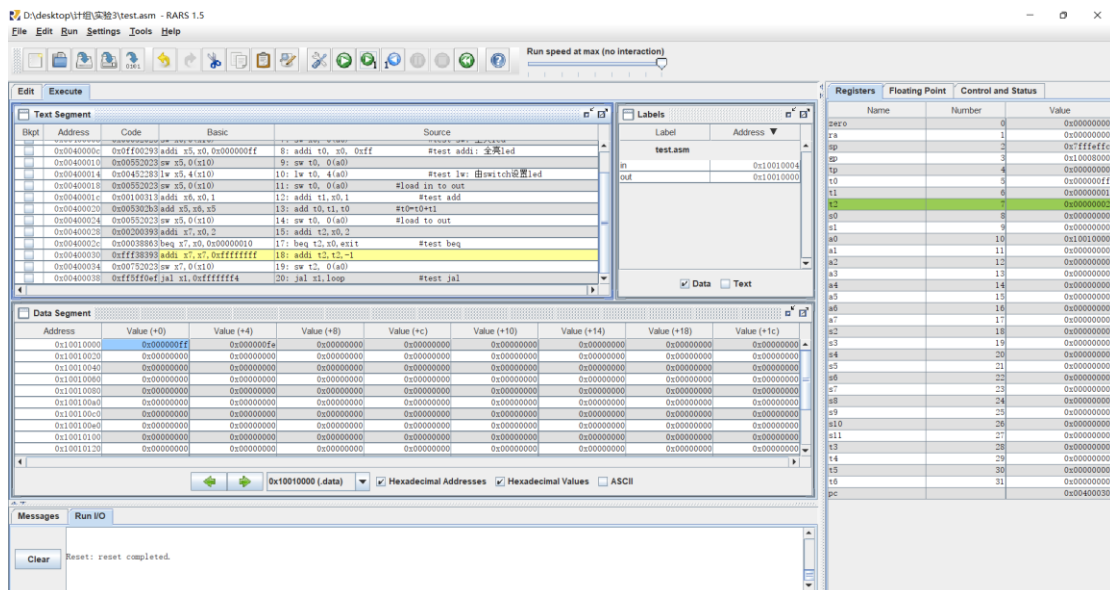
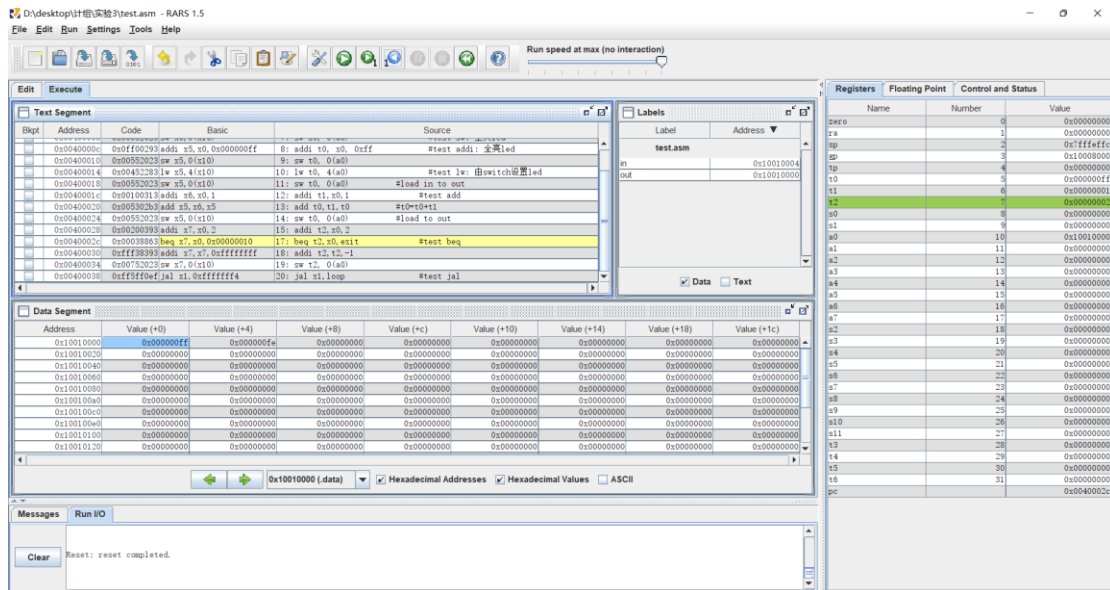
t1=1

t0=t0+t1=0xff

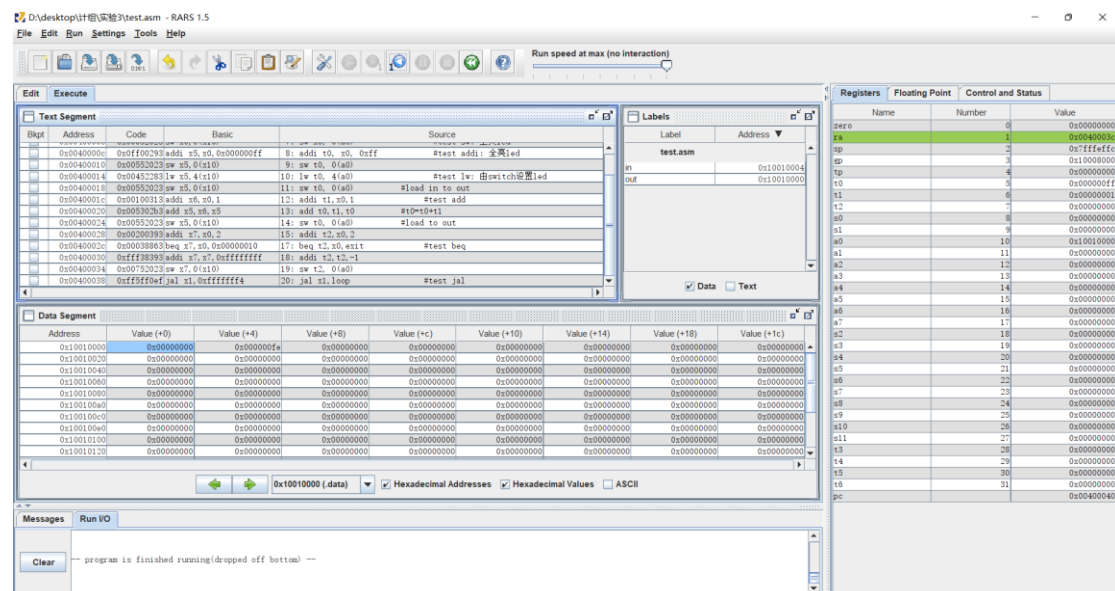
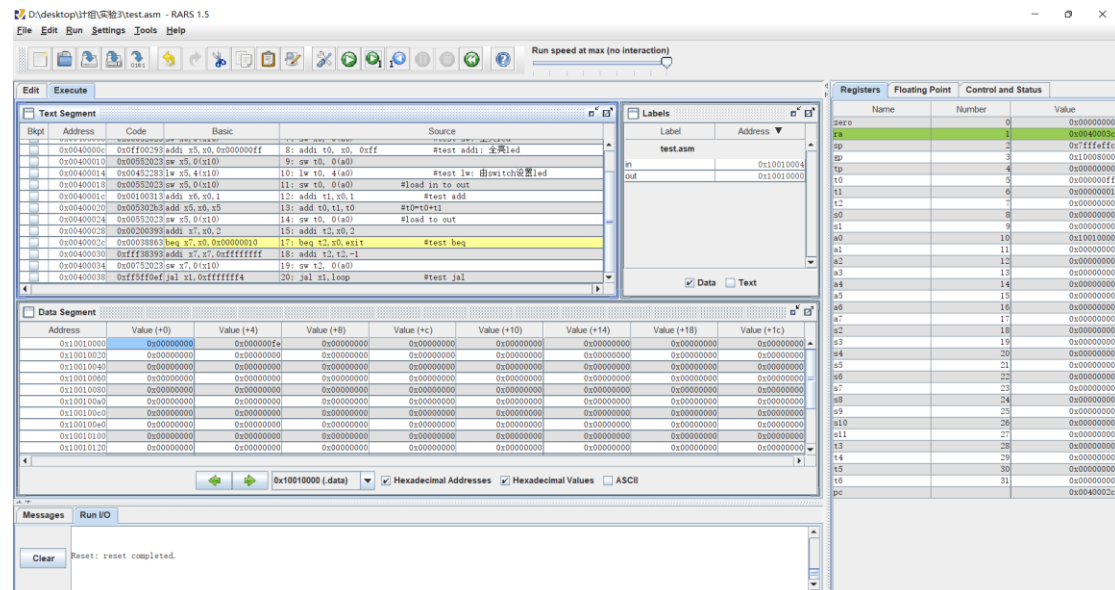
随后将 t0 存入 m[out]

(5) test beq

初始 t2=2, 不满足 t2=0, 故不跳转, 而执行下一条指令

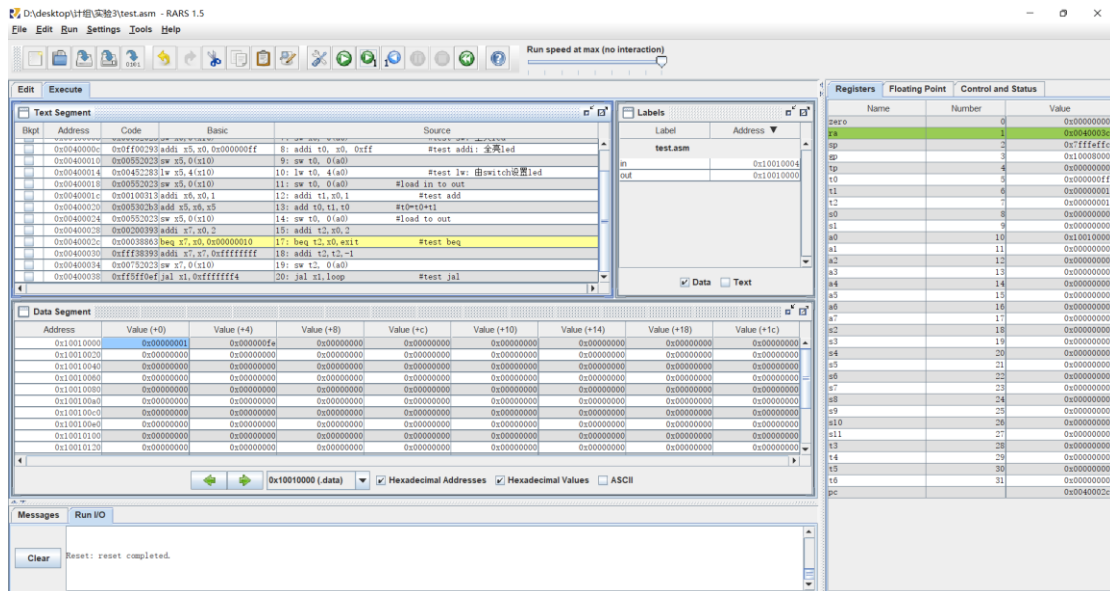
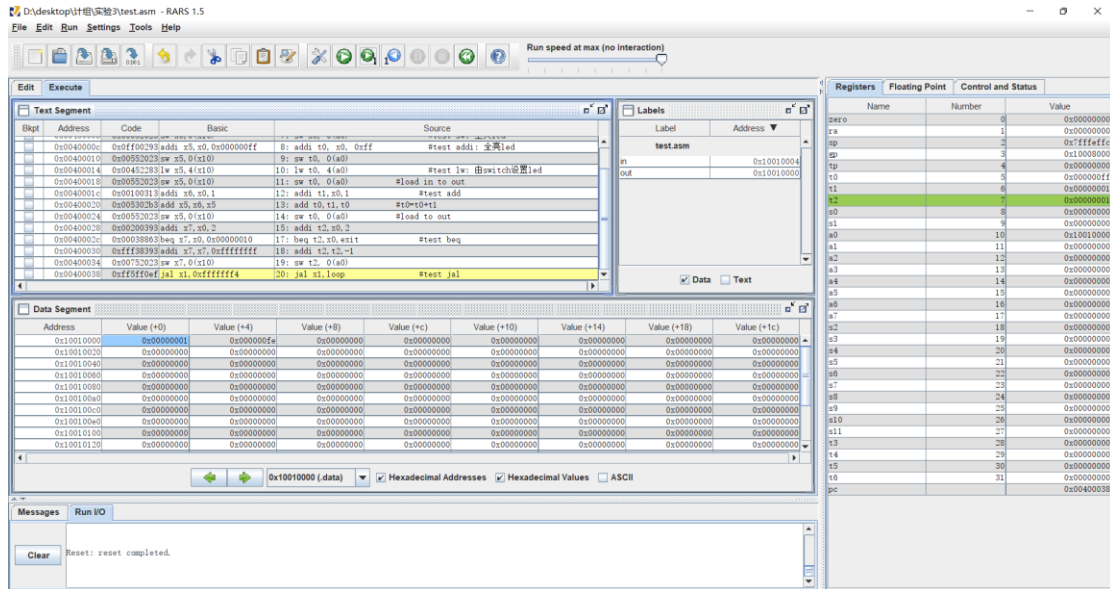


当 t2=0 时, 跳转至 exit 处, 退出程序



(6) test jal

将下一条指令地址保存在 x1 中，并跳转至循环 loop 处 (beq t2,x0,exit)



2.导出 coe 文件

testdata.coe

2022/4/3 14:50

COE 文件

11 KB

testins.coe

2022/4/3 14:51

COE 文件

1 KB

● 编写斐波那契的汇编程序

1.fibonacci 汇编程序如下:

注:

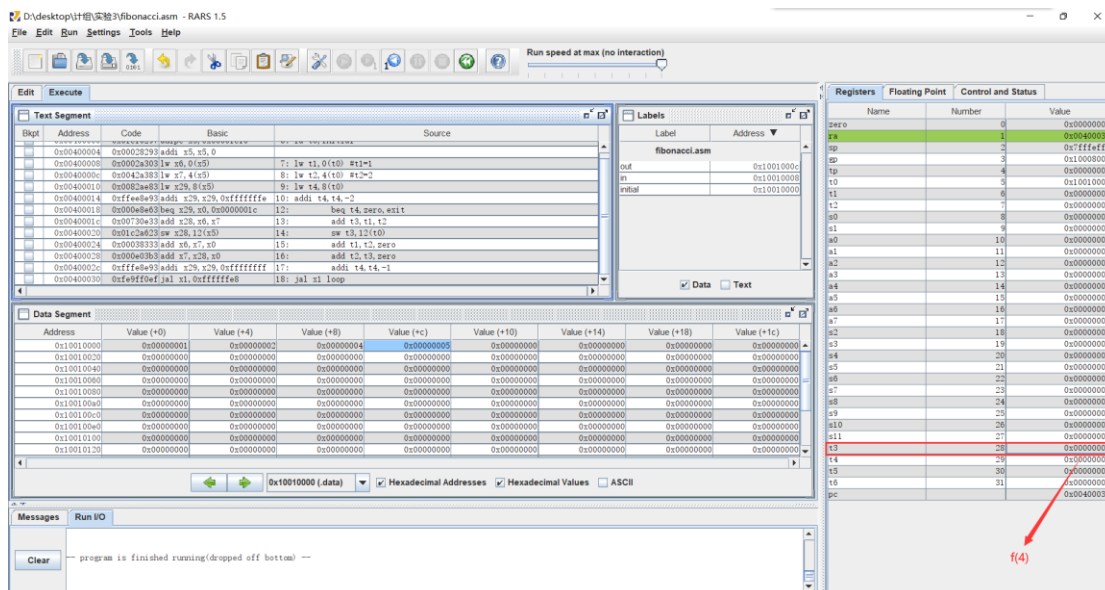
initial 是数列前两项 f1, f2;

in 是 n 值;



out 是输出 f (n);

```
.data
initial: .word 1,2
in: .word 4
out: .word 0
.text
la t0, initial
lw t1, 0(t0) #t1=1
lw t2, 4(t0) #t2=2
lw t4, 8(t0)
addi t4, t4, -2
loop:
    beq t4, zero, exit
    add t3, t1, t2
    sw t3, 12(t0)
    add t1, t2, zero
    add t2, t3, zero
    addi t4, t4, -1
jal x1 loop
exit:
```

如上图, 当 in=4, 也即 n=4 时, f(4)=5
(最后的值可在 t3 处看出, 红色方框已框出)



2.导出 coe 文件

 fibdata.coe	2022/4/3 14:16	COE 文件	11 KB
 fibins.coe	2022/4/3 14:16	COE 文件	1 KB

3.在 ripes 上运行

