

计算机组成原理 实验报告

姓名：张艺耀

学号：PB20111630

实验日期：2022- 3 - 29

实验题目

汇编程序设计

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)
- 理解CPU调试模块PDU的使用方法

实验平台

FPGAOL Ripes Rars

实验过程

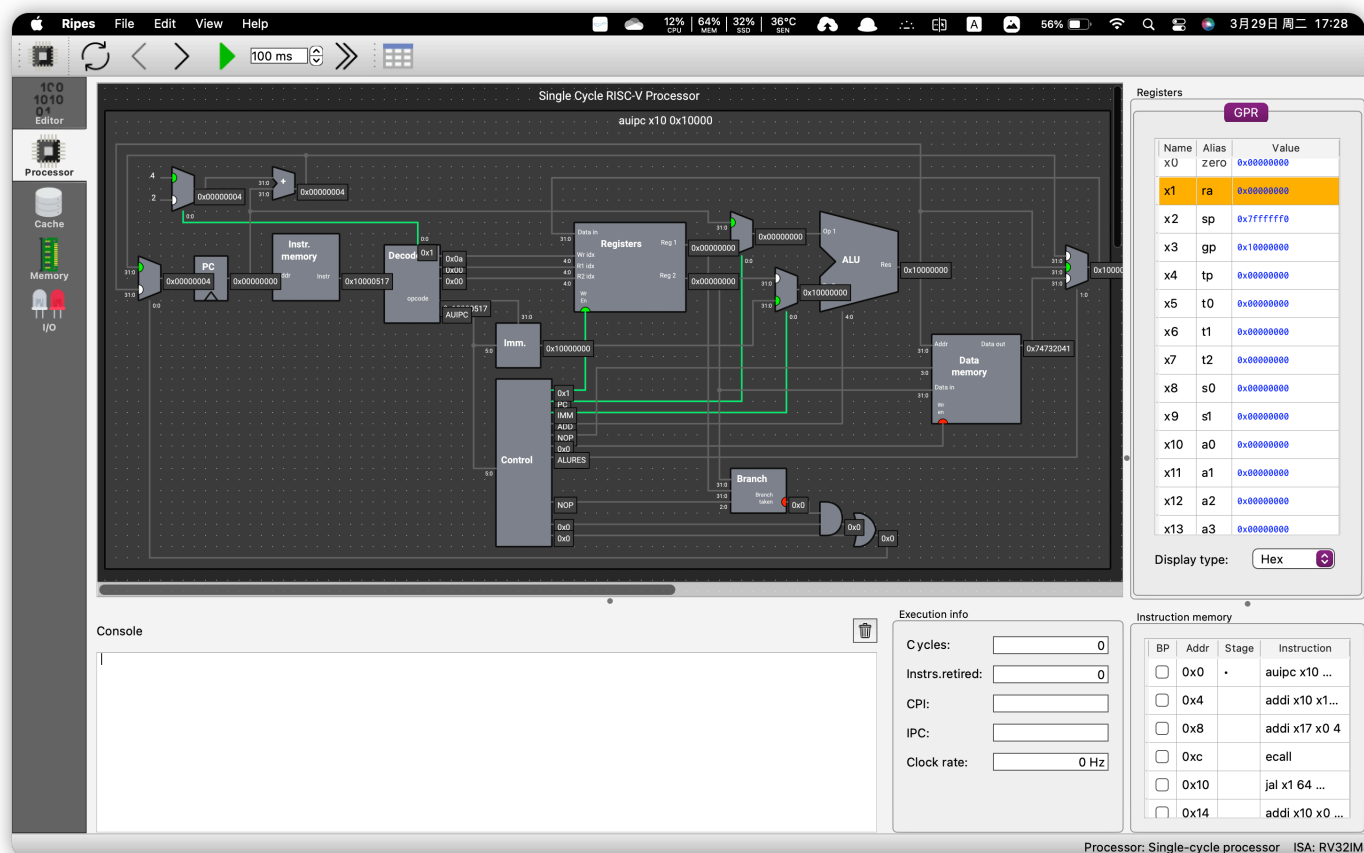
Step 1: 理解并仿真RIPES示例汇编程序

`consolePrinting.s` 程序打印字符串、数字、字符、浮点数到控制台。

在 `String printing` 中，程序先将str的地址存到a0寄存器中 再调用系统调用向控制台输出打印此字符串。
之后的 `jal printNewLine` 是函数调用，用于打印换行符，其中a7中的值决定了ecall调用打印数据的类型。

在 `Integer printing` 中，程序先把数字的范围（-10到10）分别存到a0和a1寄存器中，之后调用函数 `loopPrint` 循环输出整数和分隔符到控制台。

之后的浮点数和字符的输出类似以上两个。



```
1  # This example demonstrates how strings, integers, chars and floating point
2  # values may be printed to the console
3
4  .data
5  str:      .string      "A string"
6  newline:  .string      "\n"
7  delimiter: .string      ", "
8
9  .text
10 # ----- String printing -----
11     la a0, str # Load the address of the string, placed in the static data segment
12     li a7, 4   # Argument '4' for ecalls instructs ecalls to print to console
13     ecalls
14
15     jal printNewline
16
17 # ----- Integer printing -----
18 # Print numbers in the range [-10:10]
19     li a0, -10
20     li a1, 10
21     li a2, 1
22     jal loopPrint
23
```

```

24     jal printNewline
25
26 # ----- Float printing -----
27 # Print an approximation of Pi (3.14159265359)
28     li a0, 0x40490FDB
29     li a7, 2
30     ecall
31
32     jal printNewline
33
34 # ----- ASCII character printing -----
35 # Print ASCII characters in the range [33:53]
36     li a0, 33
37     li a1, 53
38     li a2, 11
39     jal loopPrint
40
41     # Finish execution
42     jal exit
43
44 # ===== Helper routines =====
45 printNewline:
46     la a0, newline
47     li a7, 4
48     ecall
49     jr x1
50
51 # --- LoopPrint ---
52 # Loops in the range [a0;a1] and prints the loop invariant to console
53 # a0: range start
54 # a1: range stop
55 # a2: print method (ecall argument)
56 loopPrint:
57     mv t0 a0
58     mv t1 a1
59 loop:
60     # Print value in a0 as specified by argument a2
61     mv a0 t0
62     mv a7 a2
63     ecall
64     # Print a delimiter between the numbers
65     li a7, 4
66     la a0, delimiter
67     ecall
68     # Increment
69     addi t0, t0, 1
70     ble t0, t1, loop
71     jr x1
72
73 exit:
74     # Exit program
75     li a7, 10

```

Step 2 : 设计汇编程序 验证6条指令功能

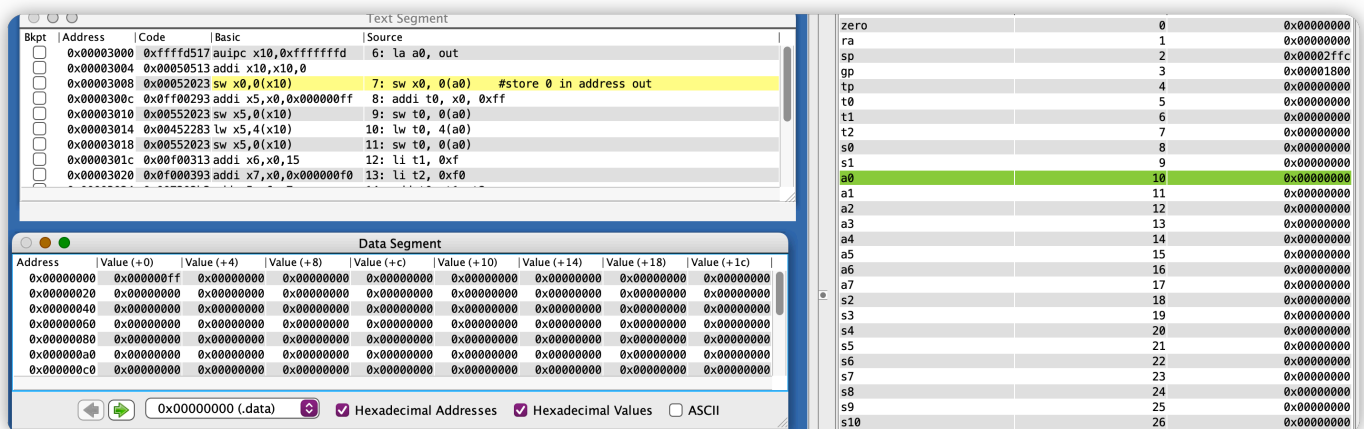
需要验证的指令为sw lw add addi beq jal

```

1  .data
2  out: .word 0xff
3  in: .word 0
4
5  .text
6  la a0, out
7  sw x0, 0(a0) #store 0 in address out
8  addi t0, x0, 0xff
9  sw t0, 0(a0)
10 lw t0, 4(a0)
11 sw t0, 0(a0)
12 li t1, 0xf
13 li t2, 0xf0
14 add t0, t1, t2
15
16 li t2, 0xf
17 beq t1, t2, equal
18 jal exit
19
20 equal:
21 li t3, 0xee
22 jal exit
23
24 exit:
25 li a7, 10
26 ecall

```

单步运行至0x3008 a0被置为0。



下一步.data(+0)处数据被置为0，说明sw运行正确。

addi将0xff存入x5寄存器 结果正确。

<input type="checkbox"/>	0x00003008	0x00052023	sw x0,0(x10)	7: sw x0, 0(a0) #store 0 in address out
<input type="checkbox"/>	0x0000300c	0x0ff00293	addi x5,x0,0x000000ff	8: addi t0, x0, 0xff
<input type="checkbox"/>	0x00003010	0x00552023	sw x5,0(x10)	9: sw t0, 0(a0)
<input type="checkbox"/>	0x00003014	0x00452283	lw x5,4(x10)	10: lw t0, 4(a0)
<input type="checkbox"/>	0x00003018	0x00552023	sw x5,0(x10)	11: sw t0, 0(a0)
<input type="checkbox"/>	0x0000301c	0x00f00313	addi x6,x0,15	12: li t1, 0xf
<input type="checkbox"/>	0x00003020	0x0f000393	addi x7,x0,0x000000f0	13: li t2, 0xf0

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

lw将t0寄存器置0

<input type="checkbox"/>	0x00003008	0x00052023	sw x0,0(x10)	7: sw x0, 0(a0) #store 0 in address out
<input type="checkbox"/>	0x0000300c	0x0ff00293	addi x5,x0,0x000000ff	8: addi t0, x0, 0xff
<input type="checkbox"/>	0x00003010	0x00552023	sw x5,0(x10)	9: sw t0, 0(a0)
<input type="checkbox"/>	0x00003014	0x00452283	lw x5,4(x10)	10: lw t0, 4(a0)
<input type="checkbox"/>	0x00003018	0x00552023	sw x5,0(x10)	11: sw t0, 0(a0)
<input type="checkbox"/>	0x0000301c	0x00f00313	addi x6,x0,15	12: li t1, 0xf
<input type="checkbox"/>	0x00003020	0x0f000393	addi x7,x0,0x000000f0	13: li t2, 0xf0

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00000000	0x000000ff	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000

add操作t3 = t1 + t2.

t0	5	0x000000ff
t1	6	0x0000000f
t2	7	0x000000f0

beq操作 若t1 = t2时跳转到标签equal处:

Text Segment				Register File			
Bkpt	Address	Code	Basic	Source	Register	Value	Comment
<input type="checkbox"/>	0x0000301c	0x00f00313	addi x6,x0,15	12: li t1, 0xf	zero	0	0x00000000
<input type="checkbox"/>	0x00003020	0x0f000393	addi x7,x0,0x000000f0	13: li t2, 0xf0	ra	1	0x00000000
<input type="checkbox"/>	0x00003024	0x007302b3	add x5,x6,x7	14: add t0, t1, t2	sp	2	0x00002ffc
<input type="checkbox"/>	0x00003028	0x00f00393	addi x7,x0,15	16: li t2, 0xf	gp	3	0x00001800
<input type="checkbox"/>	0x0000302c	0x00730463	beq x6,x7,0x00000008	17: beq t1, t2, equal	tp	4	0x00000000
<input type="checkbox"/>	0x00003030	0x00c000ef	jal x1,0x0000000c	18: jal exit	t0	5	0x000000ff
<input type="checkbox"/>	0x00003034	0x00e00e13	addi x28,x0,0x000000ee	21: li t3, 0xee	t1	6	0x0000000f
<input type="checkbox"/>	0x00003038	0x004000ef	jal x1,0x00000004	22: jal exit	t2	7	0x000000f0
<input type="checkbox"/>	0x0000303c	0x00a00093	addi x17,x0,10	25: li a7, 10	s0	8	0x00000000
					s1	9	0x00000000
					a0	10	0x00000000

jal跳转到exit函数:

Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00003020	0x0f000393	addi x7,x0,0x000000f0	13: li t2, 0xf0
<input type="checkbox"/>	0x00003024	0x007302b3	add x5,x6,x7	14: add t0, t1, t2
<input type="checkbox"/>	0x00003028	0x0f000393	addi x7,x0,15	16: li t2, 0xf
<input type="checkbox"/>	0x0000302c	0x00730463	beq x6,x7,0x00000008	17: beq t1, t2, equal
<input type="checkbox"/>	0x00003030	0x00c000ef	jal x1,0x0000000c	18: jal exit
<input type="checkbox"/>	0x00003034	0x0ee00e13	addi x28,x0,0x000000ee	21: li t3, 0xee
<input type="checkbox"/>	0x00003038	0x004000ef	jal x1,0x00000004	22: jal exit
<input type="checkbox"/>	0x0000303c	0x00a00893	addi x17,x0,10	25: li a7, 10
<input type="checkbox"/>	0x00003040	0x00000073	ecall	26: ecall

至此 验证完成。

ins.coe

```

1 memory_initialization_radix = 16;
2 memory_initialization_vector =
3 ffffd517
4 00050513
5 00052023
6 0ff00293
7 00552023
8 00452283
9 00552023
10 00f00313
11 0f000393
12 007302b3
13 00f00393
14 00730463
15 00c000ef
16 0ee00e13
17 004000ef
18 00a00893
19 00000073

```

data.coe

```

1 memory_initialization_radix = 16;
2 memory_initialization_vector =
3 000000ff
4 00000000
5 ...

```

Step 3:

fib.asm

```

1 .data
2 out: .word 0
3 in: .word 4 #存储输入输出

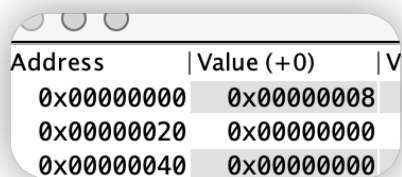
```

```

4
5 .text
6 li t1, 1
7 li t2, 2
8 la a0, out
9 lw t3, 4(a0)
10 beq t1, t3, first
11 beq t2, t3, second #特殊判断 n = 1、2的情况
12 addi t3, t3, -2
13 loop: #循环 一般的情况
14 add t4, t1, t2
15 addi t1, t2, 0
16 addi t2, t4, 0
17 addi t3, t3, -1
18 bgtz t3, loop
19 sw t4, 0(a0)
20 jal exit
21
22 first:
23 sw t1, 0(a0)
24 jal exit
25
26 second:
27 sw t2, 0(a0)
28 jal exit
29
30 exit:
31 li a7, 10
32 ecall

```

e.x 当在输入地址的数据为5时（即 `in: .word 5`）输出如下：



Address	Value (+0)	V
0x00000000	0x00000008	
0x00000020	0x00000000	
0x00000040	0x00000000	

结果为 8 正确。

当在输入地址的数据为1、2时 输出如下：

Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00003000	0x00100313	addi x6,x0,1	6: li t1, 1
<input type="checkbox"/>	0x00003004	0x00200393	addi x7,x0,2	7: li t2, 2
<input type="checkbox"/>	0x00003008	0xffffd517	auipc x10,0xffffffff	8: la a0, out
<input type="checkbox"/>	0x0000300c	0xff850513	addi x10,x10,0xffff...	
<input type="checkbox"/>	0x00003010	0x00452e03	lw x28,4(x10)	9: lw t3, 4(a0)
<input type="checkbox"/>	0x00003014	0x03c30463	beq x6,x28,0x00000028	10: beq t1, t3, first
<input type="checkbox"/>	0x00003018	0x03c38663	beq x7,x28,0x0000002c	11: beq t2, t3, second
<input type="checkbox"/>	0x0000301c	0xffee0e13	addi x28,x28,0xffff...	12: addi t3, t3, -2
<input type="checkbox"/>	0x00003020	0x00730eb3	add x29,x6,x7	14: add t4, t1, t2

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00000000	0x00000001	0x00000001	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00003000	0x00100313	addi x6,x0,1	6: li t1, 1
<input type="checkbox"/>	0x00003004	0x00200393	addi x7,x0,2	7: li t2, 2
<input type="checkbox"/>	0x00003008	0xffffd517	auipc x10,0xffffffff	8: la a0, out
<input type="checkbox"/>	0x0000300c	0xff850513	addi x10,x10,0xffff...	
<input type="checkbox"/>	0x00003010	0x00452e03	lw x28,4(x10)	9: lw t3, 4(a0)
<input type="checkbox"/>	0x00003014	0x03c30463	beq x6,x28,0x00000028	10: beq t1, t3, first
<input type="checkbox"/>	0x00003018	0x03c38663	beq x7,x28,0x0000002c	11: beq t2, t3, second
<input type="checkbox"/>	0x0000301c	0xffee0e13	addi x28,x28,0xffff...	12: addi t3, t3, -2
<input type="checkbox"/>	0x00003020	0x00730eb3	add x29,x6,x7	14: add t4, t1, t2

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00000000	0x00000002	0x00000002	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

☒ Hexadecimal Addresses
 ☒ Hexadecimal Values
 ☐ ASCII

结果正确。

fib.coe

```

1 memory_initialization_radix = 16;
2 memory_initialization_vector =
3 00100313
4 00200393
5 fffffd517
6 ff850513
7 00452e03

```



```
8 03c30463
9 03c38663
10 ffee0e13
11 00730eb3
12 00038313
13 000e8393
14 fffe0e13
15 ffc048e3
16 01d52023
17 014000ef
18 00652023
19 00c000ef
20 00752023
21 004000ef
22 00a00893
23 00000073
```

fib_data.coe

```
1 memory_initialization_radix = 16;
2 memory_initialization_vector =
3 00000002
4 ...
```

心得体会

通过本次实验，学到了很多关于RISC-V汇编语言编写的知识。

最重要的是Ripes和Rars这两个软件的使用。由于以前使用过LC-3 tools、gdb并且接触过LC-3的汇编语言和x86汇编语言 且RISC-V语言与其有很多相似之处，本次实验上手较快。