

# 计算机组成原理 实验三报告

姓名：曾郅琛 学号：PB20071431 实验日期：2022-4-7

## 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以及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 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

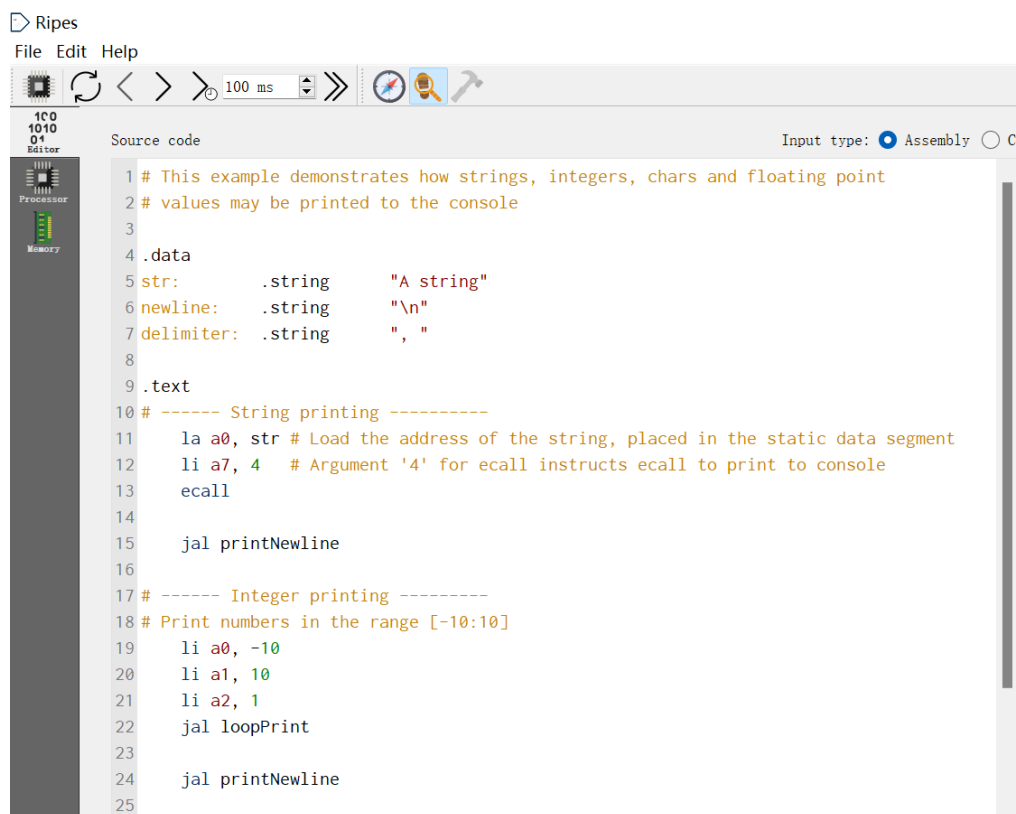
## 三、实验平台：

Ripes、RARS

## 四、实验过程：

### 4.1. 理解并仿真 RIPES 示例汇编程序

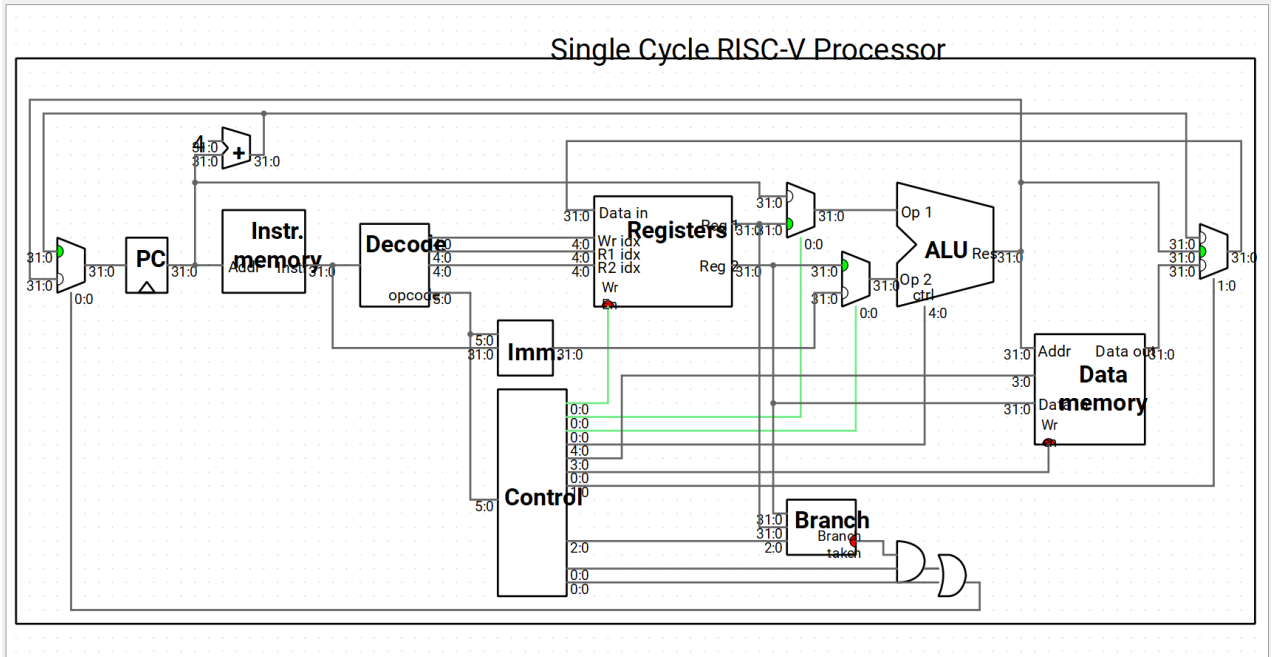
加载 Ripes 示例：Console Printing



The screenshot shows the Ripes software interface. The top menu bar includes 'File', 'Edit', and 'Help'. Below the menu is a toolbar with icons for running, stepping through code, and other debugging functions. The main window is titled 'Source code' and displays the following assembly code:

```
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
```

### 选择单周期 CPU 数据通路



## 4.2 设计汇编程序——计算斐波那契—卢卡斯数列

### 代码展示:

```

1. .data
2. a:
3. .word 0
4. n:
5. .word 8
6. .text
7. li s0 1
8. li s1 2
9. la t1 a
10. lw t0 n
11. addi s3 t0 0
12. addi t0 t0 -2
13. beq s3 s0 fo
14. beq s3 s1 ft
15. loop:
16. add s2 s1 s0
17. addi t0 t0 -1
18. addi s0 s1 0
19. addi s1 s2 0
20. beq zero t0 ft
21. jal a0 loop
22. ft:
23. sw s1 (t1)
24. jal a0 exit
25. fo:
26. sw s0 (t1)
27. exit:

```

## 代码解释：

### (1) 初始值

```
1. .data
2. a:
3. .word 0
4. n:
5. .word 8
6. .text
7. li s0 1
8. li s1 2
9. la t1 a
10. lw t0 n
```

初始定义.data 迭代次数为8，利用 RICV 中伪代码 li 和 la 分别将数列前两项赋给 s0、s1 以及把初始值赋给 t1，t0，以后面比较与运算。

### (2) 主循环

```
1. addi s3 t0 0
2. addi t0 t0 -2
3. beq s3 s0 fo
4. beq s3 s1 ft
5. loop:
6. add s2 s1 s0
7. addi t0 t0 -1
8. addi s0 s1 0
9. addi s1 s2 0
10. beq zero t0 ft
11. jal a0 loop
```

当循环结束时，beq 分支判断分别进入两个分支（fo，ft）。在正常循环中进入 loop 循环，s2 为斐波拉契数后继数  $s2=s1+s0$ ，迭代数 t0 减一，后用 s0，s1 储存原来值，进入下一个循环。同时判断 0 与 t0 的大小，判断循环是否结束。最后跳转指令 jal 回到 loop:

### (3) 结束分支

```
1. ft:
2. sw s1 (t1)
3. jal a0 exit
4. fo:
5. sw s0 (t1)
6. exit:
```

寄存器数据:

Data Segment				
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x00000000	0x00000001	0x00000008	0x00000000	0x00000000
0x00000020	0x00000002	0x00000000	0x00000000	0x00000000
0x00000040	0x00000003	0x00000000	0x00000000	0x00000000
0x00000060	0x00000005	0x00000000	0x00000000	0x00000000
0x00000080	0x00000008	0x00000000	0x00000000	0x00000000
0x000000a0	0x0000000d	0x00000000	0x00000000	0x00000000
0x000000c0	0x00000015	0x00000000	0x00000000	0x00000000
0x000000e0	0x00000022	0x00000000	0x00000000	0x00000000
0x00000100	0x00000000	0x00000000	0x00000000	0x00000000
0x00000120	0x00000000	0x00000000	0x00000000	0x00000000
0x00000140	0x00000000	0x00000000	0x00000000	0x00000000
0x00000160	0x00000000	0x00000000	0x00000000	0x00000000

## 生成.coe 文件

采用记事本分别打开生成的 ins.coe 和 data.coe，在文档的最开始加上以下

语句后保存：memory\_initialization\_radix =

16;memory\_initialization\_vector =

ins.coe - 记事本

文件 编辑 查看

```
memory_initialization_radix = 16;
memory_initialization_vector =
00100413
00200493
ffffd317
ff830313
ffffd297
ff42a283
00028993
ffe28293
02898463
00998e63
00848933
fff28293
00048413
00090493
00500463
fedff56f
00932023
0080056f
00832023
```

data.coe - 记事本

文件 编辑 查看

```
memory_initialization_radix = 16;
memory_initialization_vector =
00000001
00000002
00000003
00000005
00000008
0000000d
00000015
00000022
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
```

## 五、心得体会：

本次实验较为容易，因为之前 11xx 老师在课内作业已经训练过汇编代码编写，斐波拉契数列在 lab1 也完成了 FPGA 实现。总体比较简单。