中国科学技术大学计算机学院

《计算机组成原理实验报告》

学生姓名：林宸昊

学生学号：PB20000034

完成日期：2022.3.31

【实验题目】汇编程序设计

【实验目的】粗略掌握RISC-V指令集，能够较熟练使用Ripes及RARS软件进行汇编程序的仿真

【实验环境】RIPES RARS

【实验内容】

【一、RIPES示例程序】

- ```
  # This example demonstrates how strings, integers, chars and floating point
  # values may be printed to the console

  .data
  str:        .string     "A string"
  newline:    .string     "\n"
  delimiter:  .string     ", "

  .text
  # ------ String printing ----------
  ```
  ```

```
    la a0, str # Load the address of the string, placed in the static data
segment
    li a7, 4   # Argument '4' for ecall instructs ecall to print to console
    ecall

    jal printNewline

# ------ Integer printing ---------
# Print numbers in the range [-10:10]
    li a0, -10
    li a1, 10
    li a2, 1
    jal loopPrint

    jal printNewline

# -------- Float printing ----------
# Print an approximation of Pi (3.14159265359)
    li a0, 0x40490FDB
    li a7, 2
    ecall

    jal printNewline

# ------ ASCII character printing ---------
# Print ASCII characters in the range [33:53]
    li a0, 33
    li a1, 53
    li a2, 11
    jal loopPrint

    # Finish execution
    jal exit

# ====== Helper routines ======
printNewline:
    la a0, newline
    li a7, 4
    ecall
    jr x1

# --- LoopPrint ---
# Loops in the range [a0;a1] and prints the loop invariant to console
# a0: range start
# a1: range stop
# a2: print method (ecall argument)
loopPrint:
    addi t0, a0 0
    addi t1, a1 0
loop:
    # Print value in a0 as specified by argument a2
    addi a0, t0, 0
    addi a7, a2, 0
    ecall
    # Print a delimiter between the numbers
    li a7, 4
    la a0, delimiter
    ecall
```

```
    # Increment
    addi t0, t0, 1
    ble  t0, t1, loop
    jr x1

exit:
    li a7, 10
    ecall
```

仿真所使用的的示例程序console printing用于展示如何将字符串，字符，整数以及浮点数打印出来。其中有如下要点：

- .data

  用于声明全局变量，可用于字符串的定义（赋值）——使用.string；

- .text

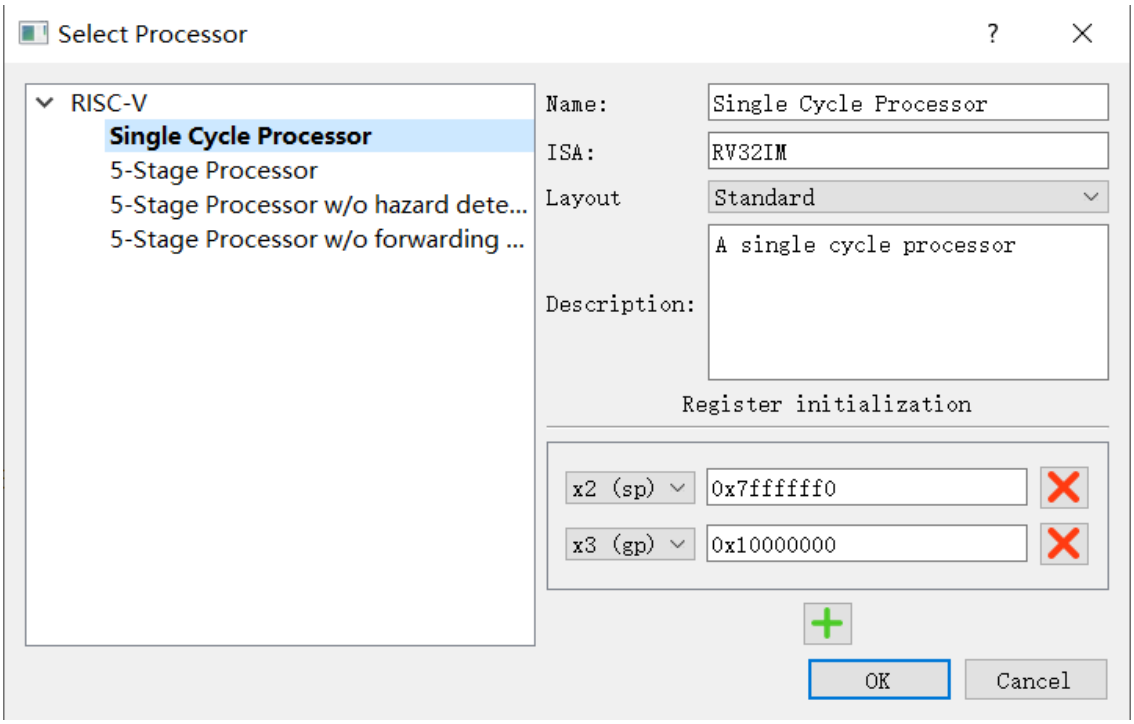
  表示代码段即程序主体部分；

- li 与 ecall

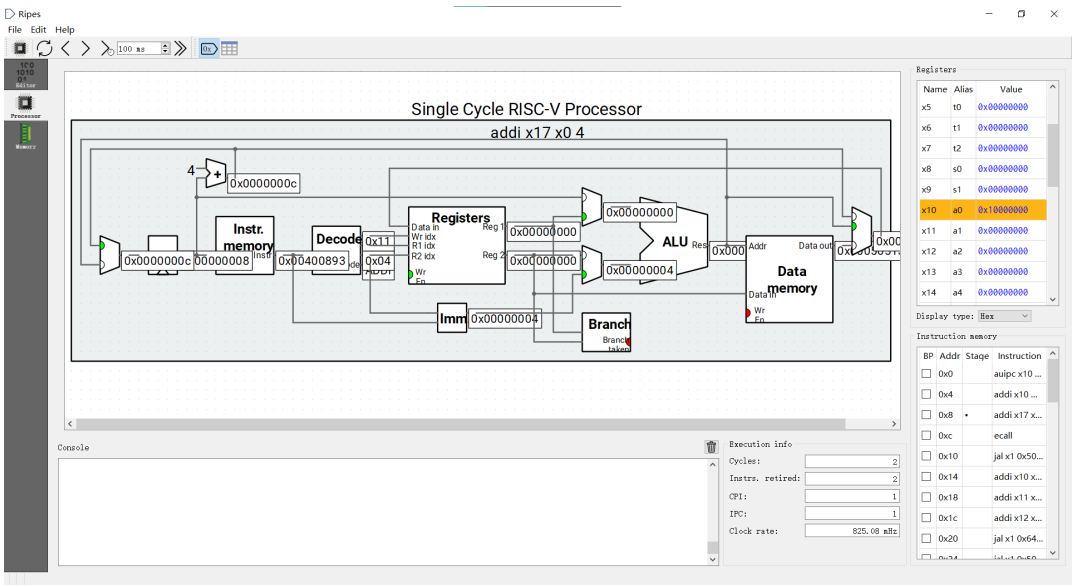
  li其实是一个伪指令，用于直接向rd中加载立即数，便于使用ecall这一系统调用调用a7中参数值所代表的可供调用的指令，有点类似ICS中的trap与陷入矢量表，比如

  ```
  # Argument '4' for ecall instructs ecall to print to console
  # Argument '10' for exit
  ```

- 使用Ripes进行仿真
  - 选择单周期



  - 单步执行，查看寄存器，实时显示通路中各结构所储存值



在指令执行中，发生数据传递的通路会暂时高亮为绿色。

【二、指令验证】

- 汇编程序

```
.text
main:
li s2, 10
# sw and lw test
sw s2, 8(s3)
lw s1, 4(s3)
lw s1, 8(s3)
# s1 should be 10 now
# add and addi test
add s1, s1, s2
# s1 should be 20 now
addi s1, s1, 20
# s1 should be 40 now
addi s2, s2, 30
# beq test
beq s2, s1, beqtest
jaltest:
li s5, 1
# s5 should be 1 now
exit:
li a7, 10
ecall
beqtest:
li s4, 1
# s4 should be 1 now
# jump to jaltest
jal s5, jaltest
```

- 最终寄存器结果

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x00002ffc
gp	3	0x00001800
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000028
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x0000000a
s2	18	0x00000028
s3	19	0x00000000
s4	20	0x00000001
s5	21	0x00000001
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		0x0000302c

- ins.coe

```
memory_initialization_radix  = 16;
memory_initialization_vector =
00a00913
0129a423
0049a483
0089a483
012484b3
01448493
01e90913
00990463
00100a93
00100a13
ff9ffaef
```

- data.coe

```
memory_initialization_radix  = 16;
memory_initialization_vector =
00000000
00000000
0000000a
……
```

【三、斐波那契数列】

- 汇编程序

```
.text
main:
# initialize, arguments '5' can be any value, which depends on how many
# numbers you want to show
li s11, 5
li s2, 1
li s3, 2
# use s1 as a counter
li s1, 1
# fibo
fibo:
# s4 = s1 + s2
add s4, s2, s3
# s2 = s3, s3 = s4
sw s3, 0(s7)
lw s2, 0(s7)
sw s4, 0(s7)
lw s3, 0(s7)
# add counter value
addi s1, s1, 1
ble s1, s11, fibo
```

- 寄存器运行结果（此处共7项）

s1	9	0x00000006
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x0000000d
s3	19	0x00000015
s4	20	0x00000015

- fibo_ins.coe

```
memory_initialization_radix  = 16;
memory_initialization_vector =
00500d93
00100913
00200993
00100493
01390a33
013ba023
000ba903
014ba023
000ba983
00148493
fe9dd4e3
```

- fibo_data.coe

```
memory_initialization_radix  = 16;
memory_initialization_vector =
00000015
00000000
00000000
……
```

【总结与思考】

- 总的来说是一次过度实验，主要是为下一次实验做准备，以及简要掌握两个软件的用法。