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

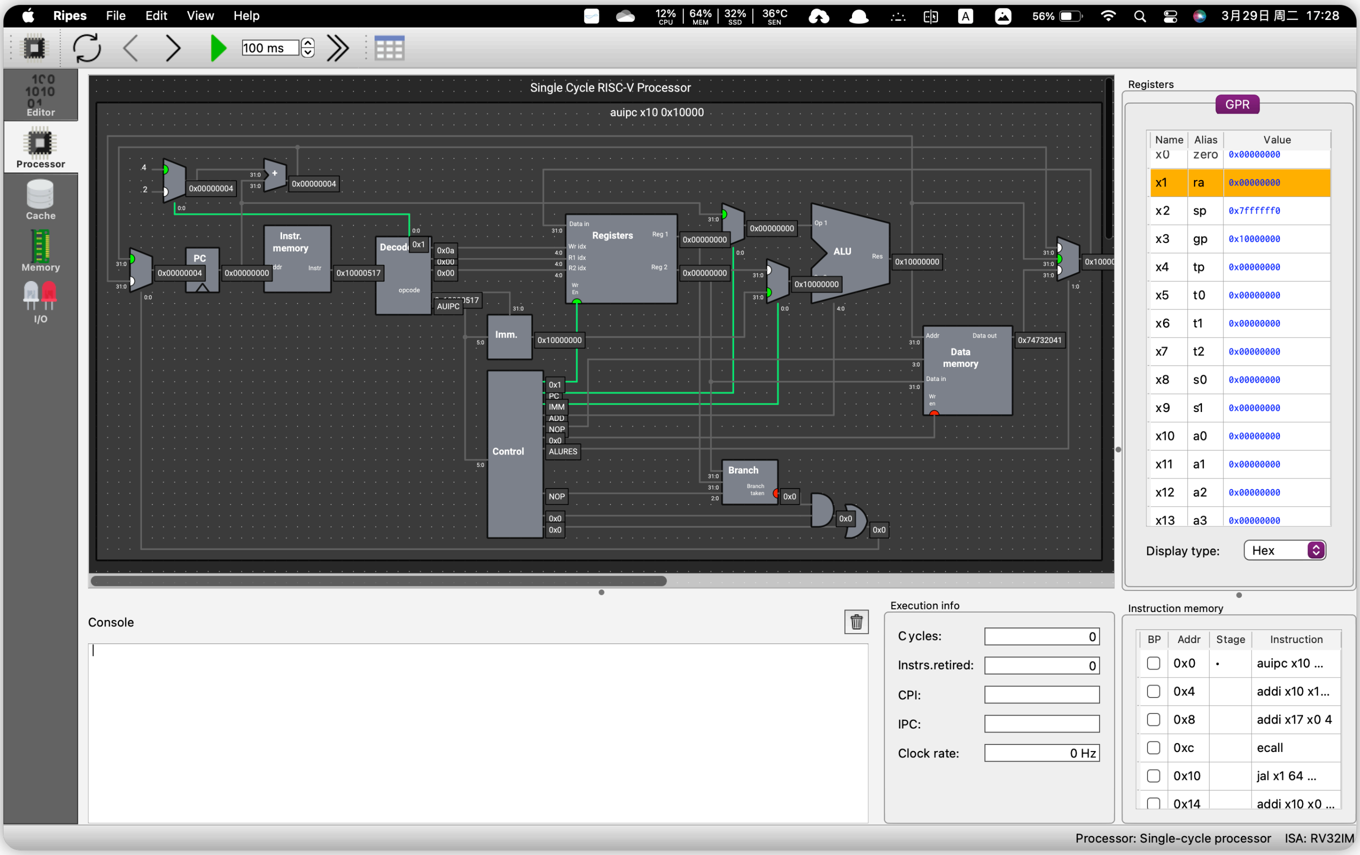
Step 1 : 理解并仿真RIPES示例汇编程序

`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。




```
# 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:
    mv t0 a0
    mv t1 a1
loop:
    # Print value in a0 as specified by argument a2
    mv a0 t0
    mv a7 a2
    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:
    # Exit program
    li a7, 10
```

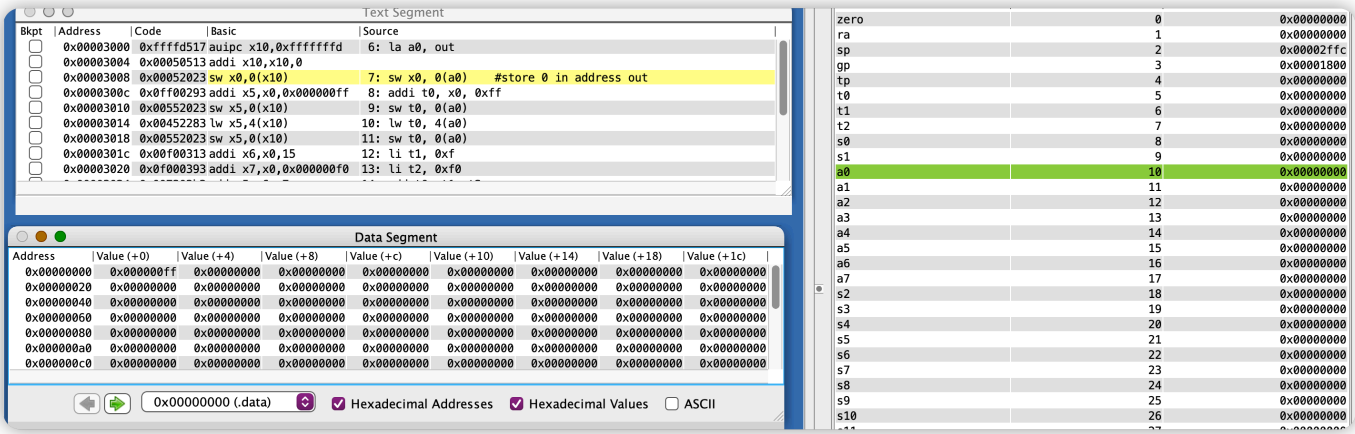
```
    ecall
```

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

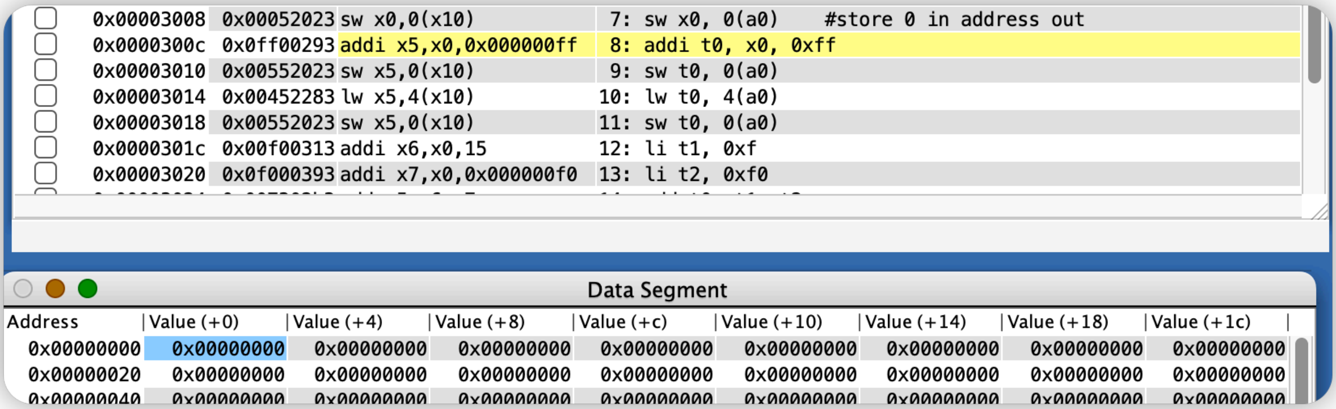
```
.data
out: .word 0xff
in: .word 0

.text
la a0, out
sw x0, 0(a0)  #store 0 in address out
addi t0, x0, 0xff
sw t0, 0(a0)
lw t0, 4(a0)
sw t0, 0(a0)
li t1, 0xf
li t2, 0xf0
add t0, t1, t2

li t2, 0xf
beq t1, t2, equal
jal exit

equal:
li t3, 0xee
jal exit

exit:
li a7, 10
ecall
```

单步运行至0x3008 a0被置为0。

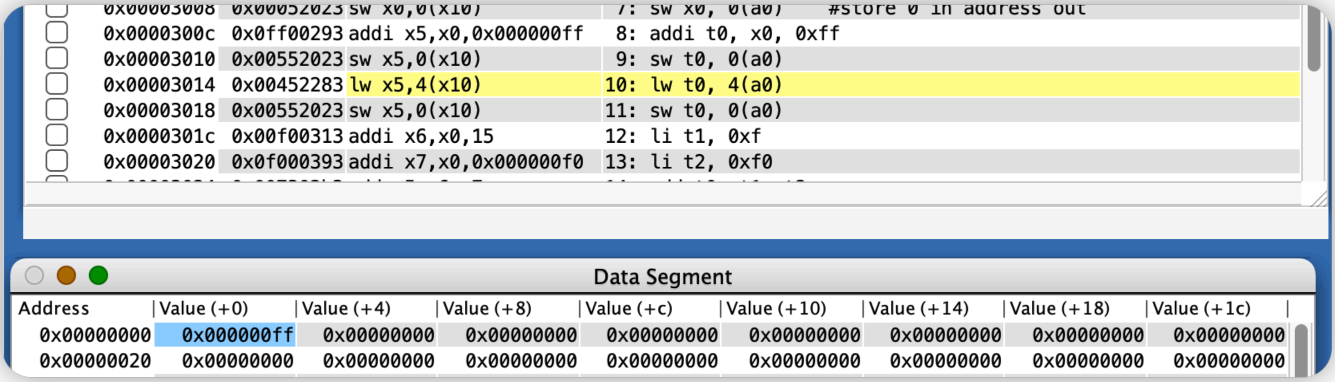


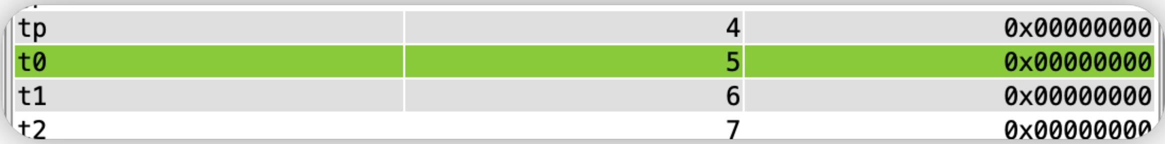
下一步.data(+0)处数据被置为0，说明sw运行正确。

addi将0xff存入x5寄存器 结果正确。

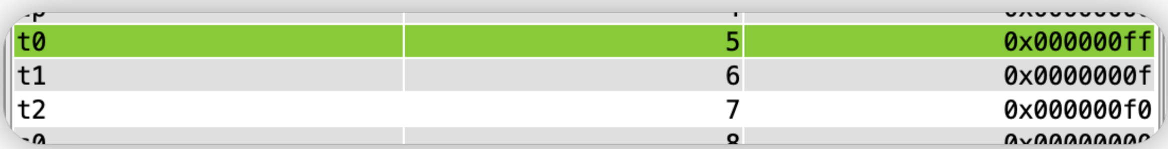


lw将t0寄存器置0

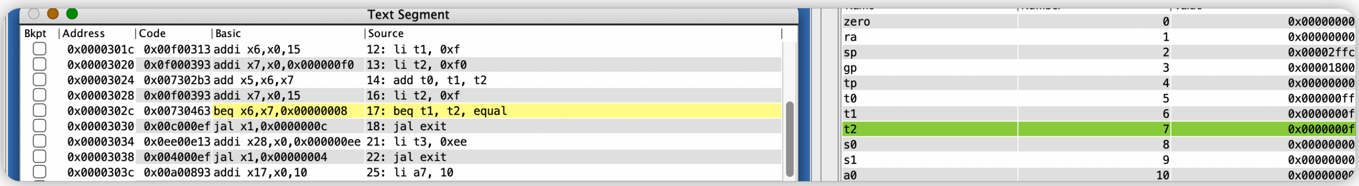




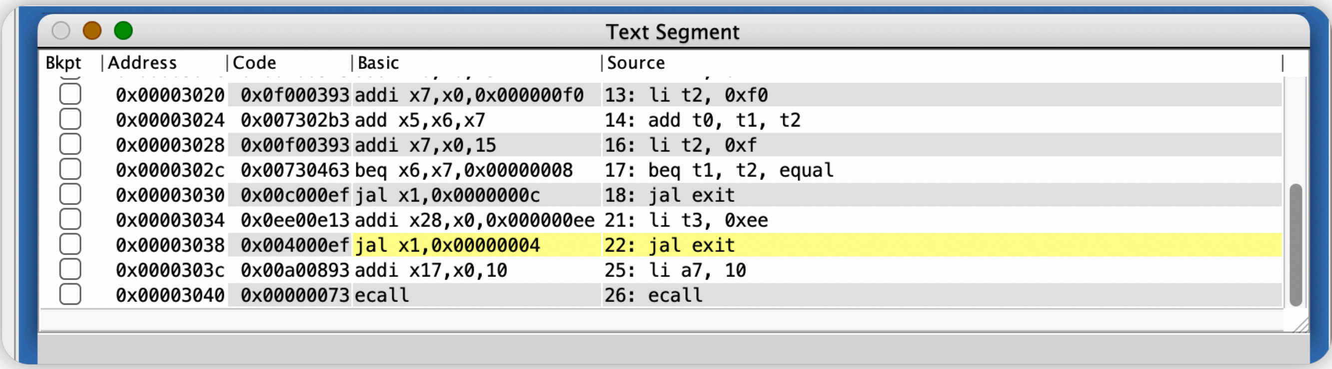
add操作t3 = t1 + t2.



beq操作 若t1 = t2时跳转到标签equal处：



jal跳转到exit函数：



至此 验证完成。

ins.coe

```
memory_initialization_radix = 16;
memory_initialization_vector =
ffffd517
00050513
00052023
0ff00293
00552023
00452283
00552023
00f00313
0f000393
007302b3
00f00393
00730463
00c000ef
0ee00e13
004000ef
00a00893
00000073
```

data.coe

```
memory_initialization_radix = 16;
memory_initialization_vector =
000000ff
00000000
...
```

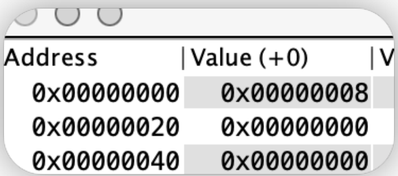
Step 3：

fib.asm

```
.data
out: .word 0
in: .word 4 #存储输入输出
```

```
.text
li t1, 1
li t2, 2
la a0, out
lw t3, 4(a0)
beq t1, t3, first
beq t2, t3, second #特殊判断 n = 1、2的情况
addi t3, t3, -2
loop: #循环 一般的情况
add t4, t1, t2
addi t1, t2, 0
addi t2, t4, 0
addi t3, t3, -1
bgtz t3, loop
sw t4, 0(a0)
jal exit

first:
sw t1, 0(a0)
jal exit

second:
sw t2, 0(a0)
jal exit

exit:
li a7, 10
ecall
```

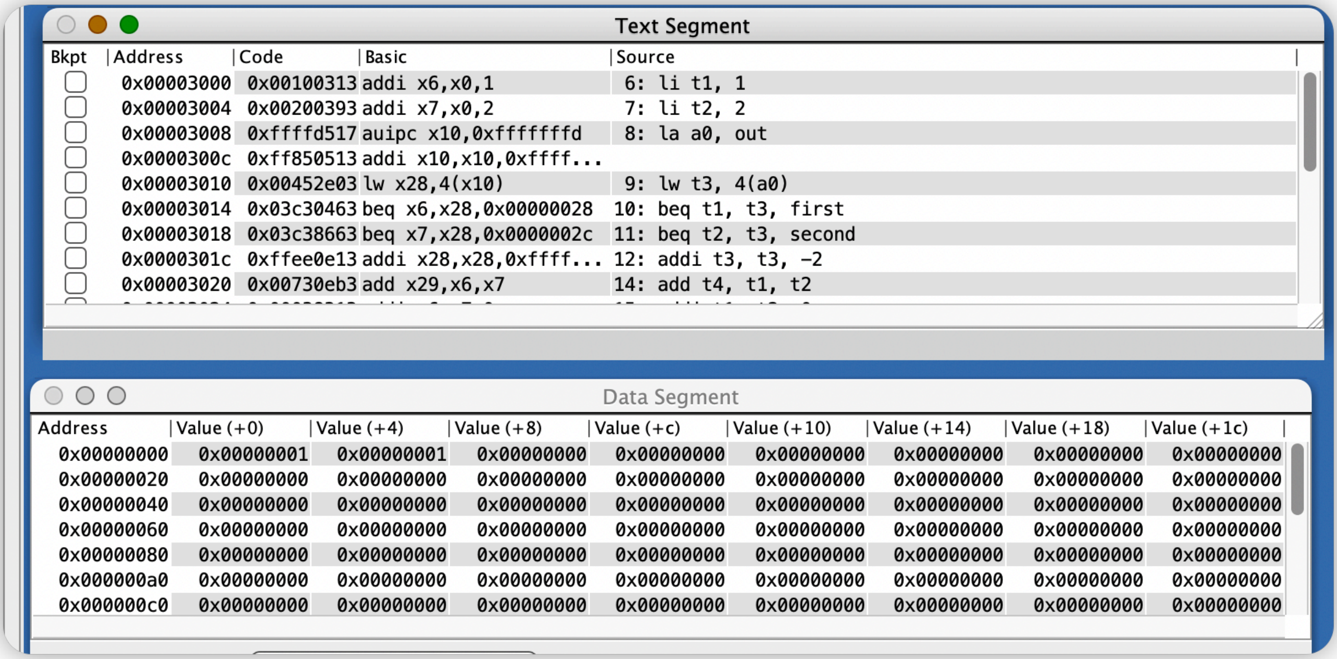
e.x 当在输入地址的数据为5时（即`in: .word 5`）输出如下：

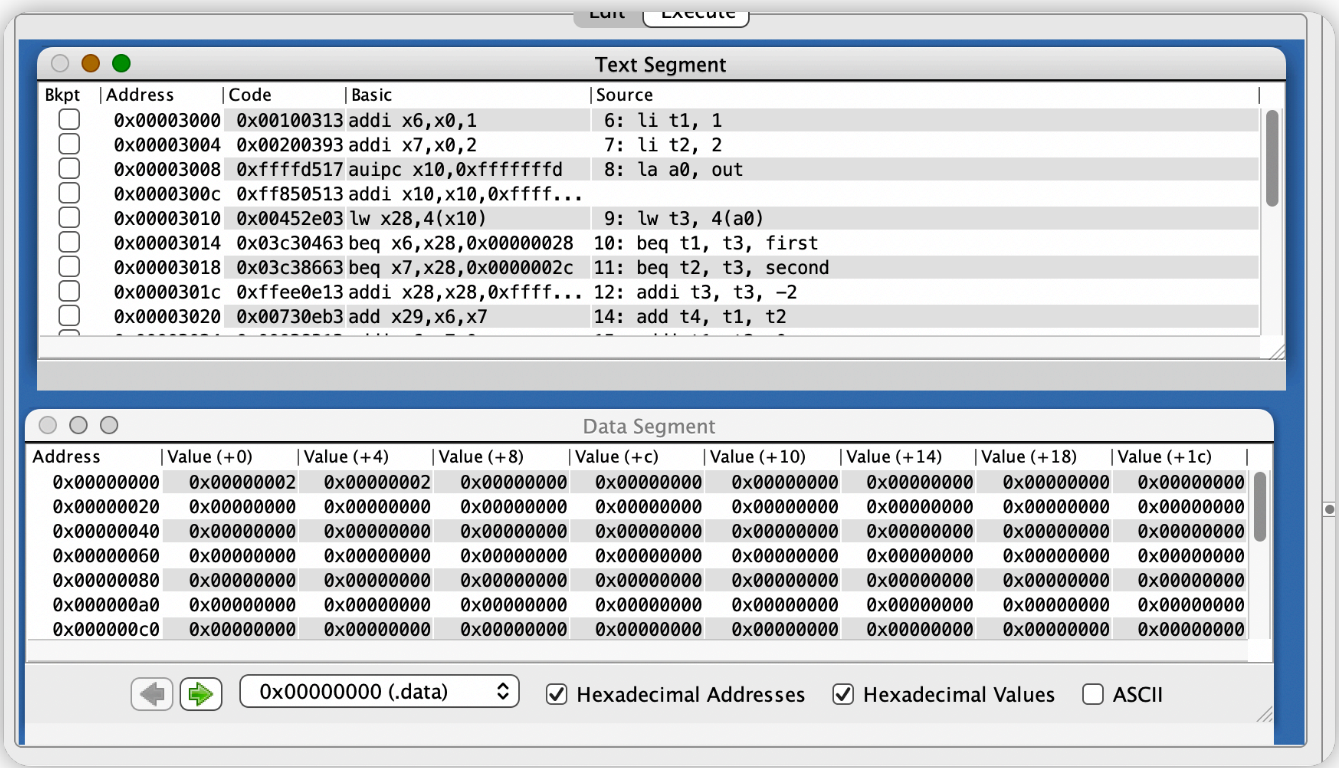


Address	Value (+0)
0x00000000	0x00000008
0x00000020	0x00000000
0x00000040	0x00000000

结果为 8 正确。

当在输入地址的数据为1、2时 输出如下：





结果正确。

fib.coe

```
memory_initialization_radix = 16;
memory_initialization_vector =
00100313
00200393
ffffd517
ff850513
00452e03
```

```
03c30463
03c38663
ffee0e13
00730eb3
00038313
000e8393
fffe0e13
ffc048e3
01d52023
014000ef
00652023
00c000ef
00752023
004000ef
00a00893
00000073
```

fib_data.coe

```
memory_initialization_radix = 16;
memory_initialization_vector =
00000002
...
```

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。