CPU测试及汇编程序设计

> 计算机组成原理实验 三
>
> PB19071501 李平治

实验目的

- 掌握**CPU**下载调试方法，以及测试数据**(COE**文件**)**的生成方法
- 熟悉汇编程序的基本结构、仿真和调试的基本方法
- 理解机器指令实现的基本原理（数据通路和控制器的协调工作过程）

实验环境

- **RARS** (*RISC-V Assembler and Runtime Simulator*)
- **Ripes** (图形化RISC-V模拟器)

仿真 RIPES 示例汇编程序——ConsolePrinting

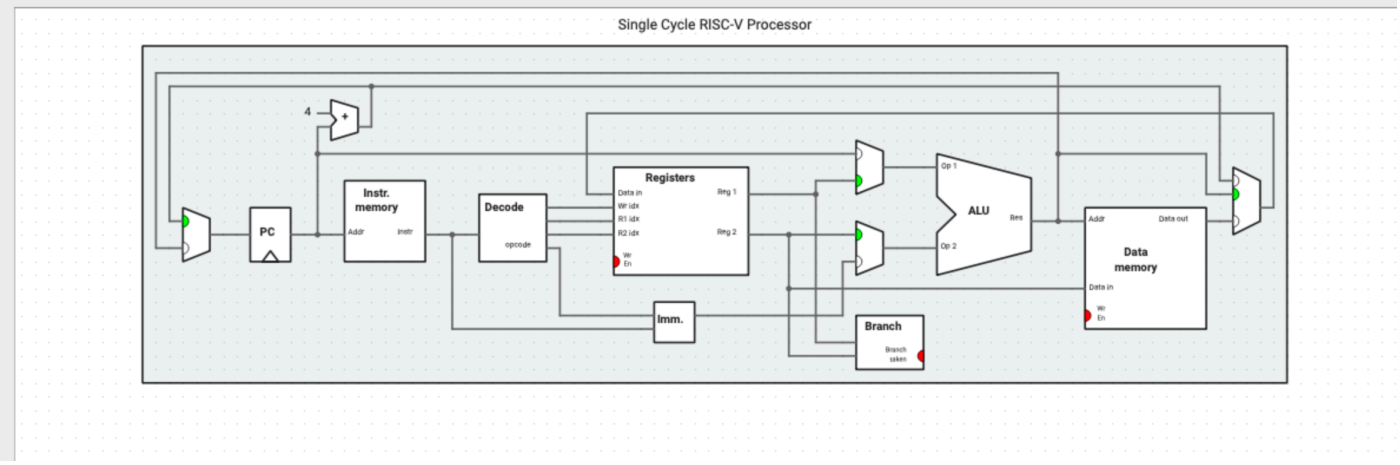
1.代码

Source code　　Input type: Assembly C　　Executable code　　View mode: Binary Disassembled

```
# 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 ---
```

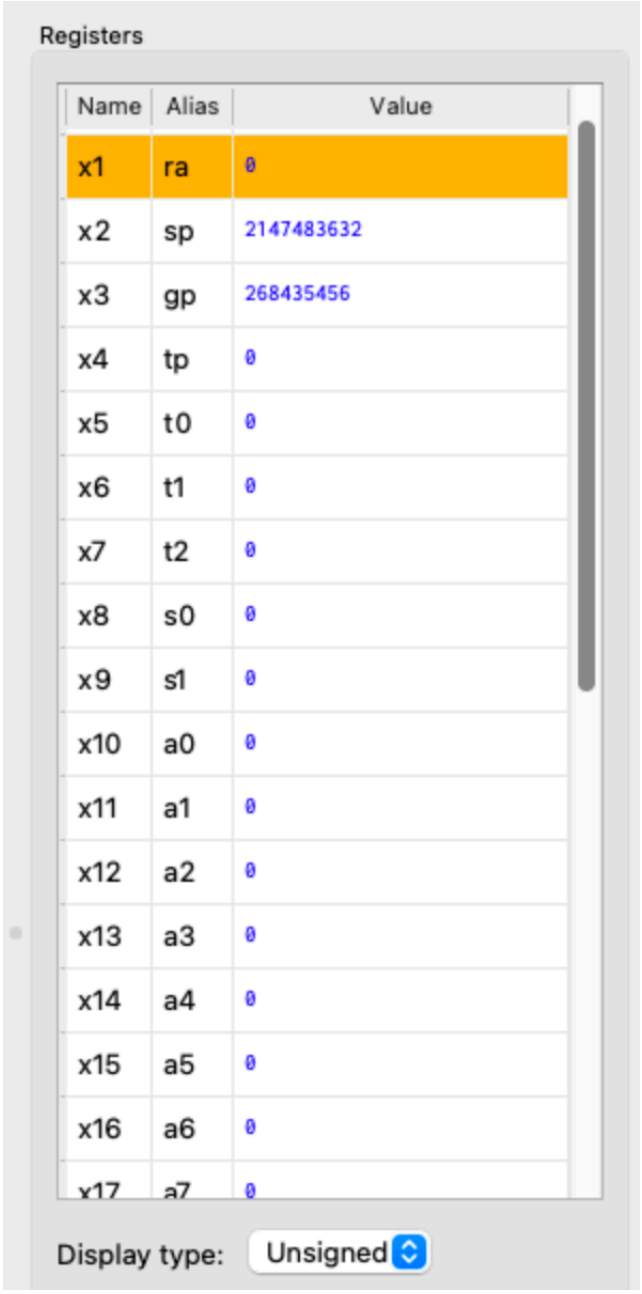
```
    0:      10000517        auipc x10 0x10000
    4:      00050513        addi x10 x10 0
    8:      00400893        addi x17 x0 4
    c:      00000073        ecall
   10:      040000ef        jal x1 0x50 <printNewline>
   14:      ff600513        addi x10 x0 -10
   18:      00a00593        addi x11 x0 10
   1c:      00100613        addi x12 x0 1
   20:      044000ef        jal x1 0x64 <loopPrint>
   24:      02c000ef        jal x1 0x50 <printNewline>
   28:      40491537        lui x10 0x40491
   2c:      fdb50513        addi x10 x10 -37
   30:      00200893        addi x17 x0 2
   34:      00000073        ecall
   38:      018000ef        jal x1 0x50 <printNewline>
   3c:      02100513        addi x10 x0 33
   40:      03500593        addi x11 x0 53
   44:      00b00613        addi x12 x0 11
   48:      01c000ef        jal x1 0x64 <loopPrint>
   4c:      048000ef        jal x1 0x94 <exit>

00000050 <printNewline>:
   50:      10000517        auipc x10 0x10000
   54:      fbc50513        addi x10 x10 -68
   58:      00400893        addi x17 x0 4
   5c:      00000073        ecall
   60:      00008067        jalr x0 x1 0

00000064 <loopPrint>:
   64:      00050293        addi x5 x10 0
   68:      00058313        addi x6 x11 0

0000006c <loop>:
   6c:      00028513        addi x10 x5 0
   70:      00060893        addi x17 x12 0
   74:      00000073        ecall
   78:      00400893        addi x17 x0 4
   7c:      10000517        auipc x10 0x10000
   80:      f9450513        addi x10 x10 -108
   84:      00000073        ecall
   88:      00128293        addi x5 x5 1
   8c:      fe5350e3        bge x6 x5 -32 <loop>
   90:      00008067        jalr x0 x1 0

00000094 <exit>:
   94:      00a00893        addi x17 x0 10
   98:      00000073        ecall
```

2.单周期CPU数据通路

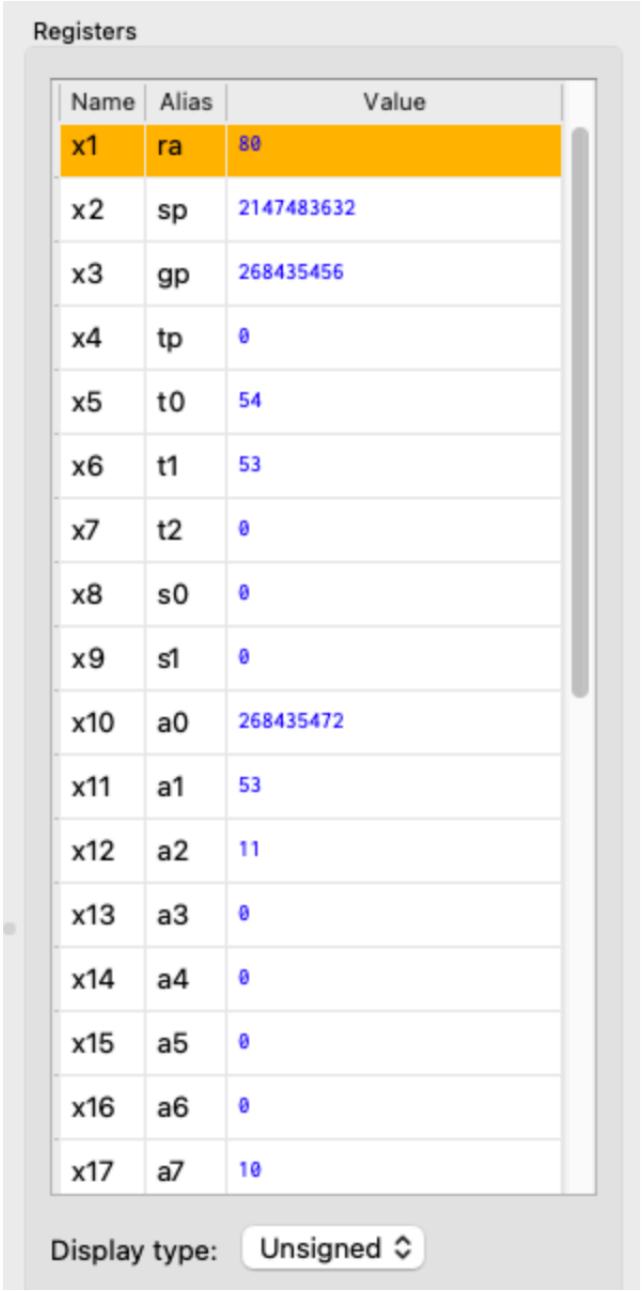


3.寄存器内容变化

- 初始:


Registers
Name
Alias
Value
x1 ra 0
x2 sp 2147483632
x3 gp 268435456
x4 tp 0
x5 t0 0
x6 t1 0
x7 t2 0
x8 s0 0
x9 s1 0
x10 a0 0
x11 a1 0
x12 a2 0
x13 a3 0
x14 a4 0
x15 a5 0
x16 a6 0
x17 a7 0
Display type:
Unsigned

- 运行结束：



汇编指令测试

1.待测试指令

- sw
- lw
- add
- addi
- beq
- jal

2.代码设计

```
li a1, 1  #load immediate number 1 into register a1
li a2, 2  #load immediate number 2 into register a2

test_sw:
  sw a1, 0(x0)
  sw a2, 4(x0)

test_lw:
  lw a3, 0(x0)
  lw a4, 4(x0)

test_add:
  add a3, a1, a2

test_addi:
  addi a4, a4, 1

test_beq:
  beq a4, a1, test_addi

test_jal:
  jal x1, exit

add1:
  addi a5, a5, 1 #This will not run if jal works

exit:
```

3.测试结果

- 运行前

  寄存器

x10	a0	0
x11	a1	0
x12	a2	0
x13	a3	0
x14	a4	0
x15	a5	0
x16	a6	0
x17	a7	0

内存

0x00000004	2098707
0x00000000	1050003

- 运行后

寄存器

x10	a0	0
x11	a1	1
x12	a2	2
x13	a3	3
x14	a4	3
x15	a5	0
x16	a6	0
x17	a7	0

内存

0x00000004	2
0x00000000	1

4.结果分析

- 寄存器a1, a2, a3, a4 和内存0x0, 0x4的结果证实:

  sw (*store word*) 的作用是将寄存器中储存一个字节到内存中；**lw** (*load word*) 的作用是从内存中加载一个字节到寄存器中.

- 寄存器a3, a4的结果证实:

  add的作用是将两个寄存器中的值相加并存入第三个寄存器；**addi**的作用是将一个寄存器中的值与立即数相加，并存入第二个寄存器中.

- 寄存器x1, a5的结果证实:

  jal (*jump and link*) 将跳转到指定label处，并在指定寄存器中存入跳转前的位置；**beq** (*branch if equal*) 的作用是判定两个寄存器中的值是否相等，若相等则跳转到指定label.

汇编程序计算*斐波那契—卢卡斯数列*

1. 设计思路

- 斐波那契—卢卡斯数列:

  从第三项开始，每一项都等于前两项之和，满足这个性质的数列称之为*斐波那契—卢卡斯数列*

- 汇编程序计算 **F-L** 数列:

  通过 **li** 指令读入初始项数值与终止项数，并将计算结果存入内存中

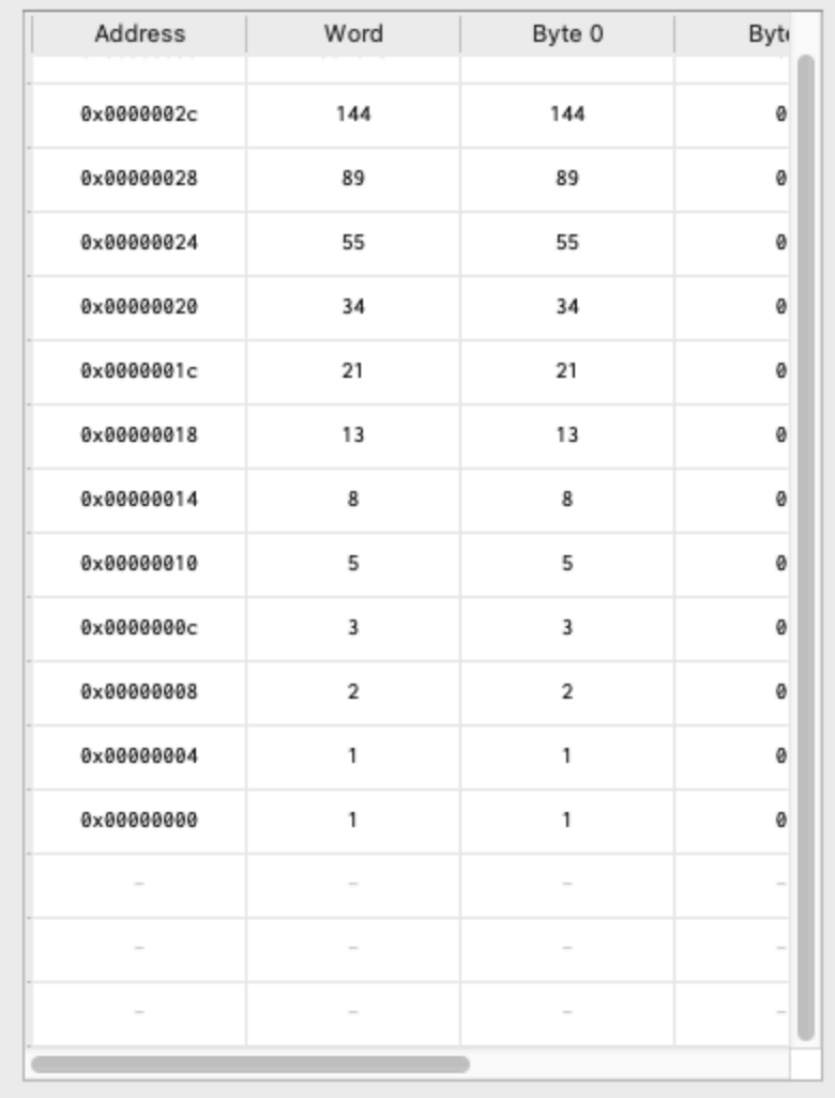
2.核心代码

```
fib:
    li a0, 12
    li a1, 1            # This is the first number, aka $a.
    li a2, 1            # This is the second number, aka $b.
    li a4, 2            # This is i
    li a6, 2            # to check a0 with
    li a7, 8
    sw a1, 0(x0)
    sw a2, 4(x0)

    blt a0, a6, cond1   # check if a0 <= 1
    beq a0, a6, end     # if a0 == 2
    bgt a0, a6, loop    # if a0 > 2, proceed to loop

loop:                   # for i in range(2, n+1): $c = $a + $b; $a = $b; $b = $c;
return $b
    add a3, a1, a2
    mv a1, a2
    mv a2, a3
    addi a4, a4, 1
    sw a2, 0(a7)
```

```
    addi a7, a7, 4
    blt a4, a0, loop
    bge a4, a0, end      # iterates n-1 times and goes to end

cond1:
    mv a0, a1            # if n==1 return the first number
    beq x0, x0, end

end:


```

3.运行结果

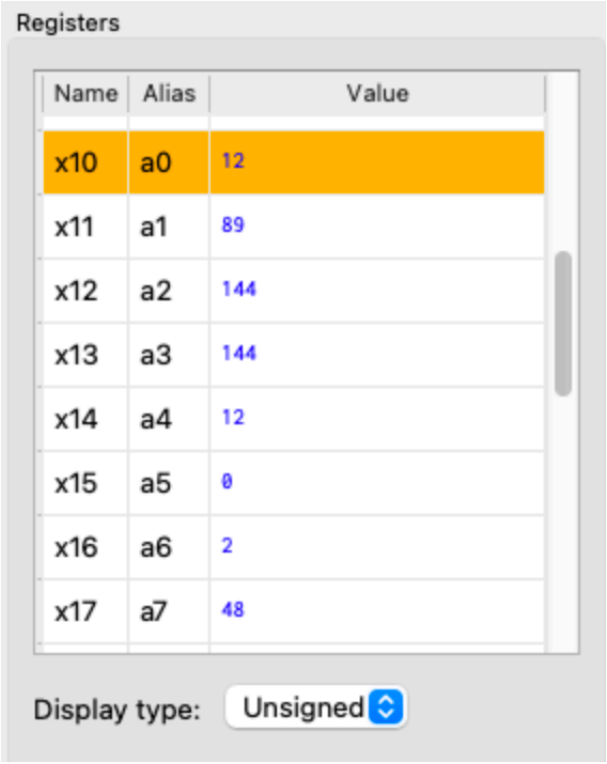
- 内存:



Address	Word	Byte 0	Byt
0x0000002c	144	144	0
0x00000028	89	89	0
0x00000024	55	55	0
0x00000020	34	34	0
0x0000001c	21	21	0
0x00000018	13	13	0
0x00000014	8	8	0
0x00000010	5	5	0
0x0000000c	3	3	0
0x00000008	2	2	0
0x00000004	1	1	0
0x00000000	1	1	0
-	-	-	-
-	-	-	-
-	-	-	-

可以看到，从下往上依次是初始值为1, 1的**FL**数列的第1项到第12项

- 寄存器



结果总结

本次试验结果符合预期，指令测试、**FLS**计算结果均与预期相同.

实验总结

- 本次试验中，掌握了汇编语言的基本指令，熟悉了**Risc-V**汇编模拟器.
- 本次试验中，熟悉了汇编语言的编写与调试，了解了**PDU**的结构与功能.