版本

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **更新记录** | 文档名 | | 实验指导书\_lab2.1 | |
| 版本号 | | 0.3 | |
| 创建人 | | 计算机组成原理教学组 | |
| 创建日期 | | 2022/1/1 | |
| **更新历史** | | | | |
| **序号** | **更新日期** | **更新人** | **版 号** | **更新内容** |
| 1 | 2022/1/1 | 陈颖琪 | 0.1 | RISC-V CPU模型虚拟仿真 |
| 2 | 2022/2/24 | 陈颖琪 | 0.2 | 仿真平台说明，添加ripes |
| 3 | 2022/3/29 | 陈颖琪 | 0.3 | 修正了表1中的两处错误 |
|  |  |  |  |  |

**文档错误反馈:**

本文档出现错误请联系：

Yingqichen@sjtu.edu.cn

### 实验二 之 RISC-V CPU虚拟仿真部分

### 1、实验目的

1、熟悉指令和汇编代码。

2、产生指令存储器初始值文件。

## 2、实验要求

### 2.1 分析指令执行过程

**2.1.1仿真平台**

**可以1、2两个仿真平台为主来完成实验，3仅供参考。**

**仿真软件1**：Ripes v2.2.4

RISC-V 32/64b单周期及流水线架构CPU模型仿真软件,可在canvas上本课程内下载。本部分实验和后续的lab5实验可以此平台为主。

**仿真平台2**：WebRISCV

RISC\_V 32/64b 五级流水线架构CPU模型在线仿真平台，主要用于理解流水线执行过程。在后续的lab5实验中也会使用。

<https://webriscv.dii.unisi.it/index.php>

or

<http://10.119.1.50:81/>

**辅助仿真平台3**：

RISC\_V 32b单周期架构CPU模型在线仿真平台，便于理解单条指令执行过程，仅供参考。

<http://tice.sea.eseo.fr/riscv/>

**2.1.2 仿真平台输入用测试代码段**

本次实验用到如下汇编代码段，可直接于仿真平台1、2输入用，供参考。

lui x10, 0

ori x4, x10, 0

addi x25, x0, 1

addi x26, x0, 2

addi x27, x0, 3

addi x28, x0, 4

sw x25, 0(x4)

sw x26, 4(x4)

sw x27, 8(x4)

sw x28, 12(x4)

addi x5, x0, 4

call:

jal sum

sw x12, 0(x4)

lw x19, 0(x4)

sub x18, x19, x12

addi x5, x0, 3

loop2:

addi x5, x5, -1

ori x18, x5, -1

xori x18, x18, 1365

addi x19, x0, -1

andi x20, x19, -1

or x16, x20, x19

xor x18, x20, x19

and x17, x20, x16

beq x5, x0, shift

j loop2

shift:

addi x5, x0, -1

slli x18, x5, 15

slli x18, x18, 16

srai x18, x18, 16

srli x18, x18, 15

fi:

j fi

sum:

add x18, x0, x0

loop:

lw x19, 0(x4)

addi x4, x4, 4

add x18, x18, x19

addi x5, x5, -1

bne x5, x0, loop

slli x12, x18, 0

jr ra

仿真平台3输入用。如果使用仿真平台3，对应前述代码段的写法需要有些改变。跳转的偏移量必须是实际的偏移量带符号数字值。并且不支持j、jr等伪代码。以下的修改后代码供参考。第一句设置的地址0xd0000000是为了便于通过仿真页面看到输出值。

lui x10, 0xd0000000

ori x4, x10, 0

addi x5, x0, 4

jal x1, +84

sw x2, 0(x4)

lw x9, 0(x4)

sub x8, x9, x4

addi x5, x0, 3

addi x5, x5, -1

ori x8, x5, -1

xori x8, x8, 1365

addi x9, x0, -1

andi x10, x9, -1

or x6, x10, x9

xor x8, x10, x9

and x7, x10, x6

beq x5, x0, +8

jal x0, -36

addi x5, x0, -1

slli x8, x5, 15

slli x8, x8, 16

srai x8, x8, 16

srli x8, x8, 15

jal x0, 0

add x8, x0, x0

lw x9, 0(x4)

addi x4, x4, 4

add x8, x8, x9

addi x5, x5, -1

bne x5, x0, -16

slli x2, x8, 0

jalr x0, x1, 0

**2.1.3补全表格理解代码段含义**

在选择仿真平台1（也可选择其他平台，做对比）输入测试代码段，设置单周期CPU模式，通过单步执行仿真理解代码执行的操作，得出其每条指令机器码，补全对指令的描述，或者指令执行后的输出。填入表格1，补全后四列（红色字体）内容。

表1 测试用代码段功能描述表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Address(HEX) | 标号 | 仿真平台2输入代码（这样输入才不会出错） | 仿真平台2显示代码 | 机器码（BIN） | 机器码（HEX） | 指令类型 | 描述 |
| 0 |  | lui x10, 0 | **lui a0, 0** | 00000000000000000000010100110111 | 00000537 | U | #initializex10 =base address 0 |
| 4 |  | ori x4, x10, 0 | **oritp, a0, 0** | 00000000000001010110001000010011 | 00056213 | I | #x4<- base address x10 + offset 0 =0 |
| 8 |  | addi x25, x0, 1 | **addi s9, x0, 1** | 00000000000100000000110010010011 | 00100c93 | I | #initialize x25 = 1 |
| c |  | addi x26, x0, 2 | **addi s10, x0, 2** | 00000000001000000000110100010011 | 00200d13 | I | # initialize x26 = 2 |
| 10 |  | addi x27, x0, 3 | **addi s11, x0, 3** | 00000000001100000000110110010011 | 00300d93 | I | # initialize x27 = 3 |
| 14 |  | addi x28, x0, 4 | **addi t3, x0, 4** | 00000000010000000000111000010011 | 00400e13 | I | # initialize x28 = 4 |
| 18 |  | sw x25, 0(x4) | **sw s9, 0(tp)** | 00000001100100100010000000100011 | 01922023 | I | #[0] = 1 |
| 1c |  | sw x26, 4(x4) | **sw s10, 4(tp)** | 00000001101000100010001000100011 | 01a22223 | I | # [4] = 2 |
| 20 |  | sw x27, 8(x4) | **sw s11, 8(tp)** | 00000001101100100010010000100011 | 01b22423 | I | # [8] = 3 |
| 24 |  | sw x28, 12(x4) | **sw t3, 12(tp)** | 00000001110000100010011000100011 | 01c22623 | I | # [12] = 4 |
| 28 |  | addi x5, x0, 4 | **addi t0, x0, 4** | 00000000010000000000001010010011 | 00400293 | I | # x5 = 4，循环次数 |
| 2c | Call： | Call: jal sum | **jalra, 128** | 00000101010000000000000011101111 | 054000ef | UJ | # call function sum  跳转到 pc = 80 |
| 30 |  | sw x12, 0(x4) | **sw a2, 0(tp)** | 00000000110000100010000000100011 | 00c22023 | S | #[10] <- 0x0000000a  (x12=0x0000000a) |
| 34 |  | lw x19, 0(x4) | **lw s3, 0(tp)** | 00000000000000100010100110000011 | 00022983 | I | #x19<- [10]  ([10]=0x0000000a) |
| 38 |  | sub x18, x19, x12 | **sub s2, s3, a2** | 01000000110010011000100100110011 | 40c98933 | R | #x18= 0 |
| 3c |  | addi x5, x0, 3 | **addi t0, x0, 3** | 00000000001100000000001010010011 | 00300293 | I | #x5=3 |
| 40 | loop2: | loop2:addi x5, x5, -1 | **addi t0, t0, -1** | 11111111111100101000001010010011 | fff28293 | I | #x5 <- (x5-1)，循环次数-1 |
| 44 |  | ori x18, x5, -1 | **ori s2, t0, -1** | 11111111111100101110100100010011 | fff2e913 | I | #x18= 0xffffffff ，（x18 = x5 or 12bit 立即数有符号扩展0xffffffff） |
| 48 |  | xori x18, x18, 1365 | **xori s2, s2, 1365** | 01010101010110010100100100010011 | 55594913 | I | #X18=0xfffffaaa |
| 4c |  | addi x19, x0, -1 | **addi s3, x0, -1** | 11111111111100000000100110010011 | fff00993 | I | #X19=0xffffffff |
| 50 |  | andi x20, x19, -1 | **andi s4, s3, -1** | 11111111111110011111101000010011 | fff9fa13 | I | #X20=0xffffffff，（X20=0xffffffff and 0xffffffff） |
| 54 |  | or x16, x20, x19 | **or a6, s4, s3** | 00000001001110100110100000110011 | 013a6833 | R | #X16=0xffffffff |
| 58 |  | xor x18, x20, x19 | **xor s2, s4, s3** | 00000001001110100100100100110011 | 010a78b3 | R | #X18=0x00000000 |
| 5c |  | and x17, x20, x16 | **and a7, s4, a6** | 00000001000010100111100010110011 | 010a78b3 | R | #X17=0xffffffff |
| 60 |  | beq x5, x0, shift | **beq t0, x0, 104** | 00000000000000101000010001100011 | 00028463 | SB | #Ifx5 = 0  Goto shift after finished loop2 4 times，goto pc= 68 |
| 64 |  | j loop2 | **jal x0, 64** | 11111101110111111111000001101111 | fddff06f | UJ | #Loop Loop2 for 4 times，goto pc=40 |
| 68 | shift: | shift:addi x5, x0, -1 | **addi t0, x0, -1** | 11111111111100000000001010010011 | fff00293 | I | #X5=0xffffffff |
| 6c |  | slli x18, x5, 15 | **slli s2, t0, 15** | 00000000111100101001100100010011 | 00f29913 | I | #X18=0xffff8000 |
| 70 |  | slli x18, x18, 16 | **slli s2, s2, 16** | 00000001000010010001100100010011 | 01091913 | I | #X18=0x80000000 |
| 74 |  | srai x18, x18, 16 | **srai s2, s2, 16** | 01000001000010010101100100010011 | 41095913 | I | #X18=0xffff8000 |
| 78 |  | srli x18, x18, 15 | **srli s2, s2, 15** | 00000000111110010101100100010011 | 00f95913 | I | #X18=0x0001ffff |
| 7c | finish: | finish:j finish | **jal x0, 124** | 00000000000000000000000001101111 | 0000006f | UJ | #Endhere |
| 80 | sum: | sum:add x18, x0, x0 | **add s2, x0, x0** | 00000000000000000000100100110011 | 00000933 | R | #X18 = 0 |
| 84 | loop: | loop:lw x19, 0(x4) | **lw s3, 0(tp)** | 00000000000000100010100110000011 | 00022983 | I | #X19 <- [x4] |
| 88 |  | addi x4, x4,4 | **additp, tp,4** | 00000000010000100000001000010011 | 00420213 | I | #x4 <- x4 + 4 |
| 8c |  | add x18, x18, x19 | **add s2, s2, s3** | 00000001001110010000100100110011 | 01390933 | R | #X18= x18 + [x4],  x18= 1 |
| 90 |  | addi x5, x5, -1 | **addi t0, t0, -1** | 11111111111100101000001010010011 | fff28293 | I | #x5 <- (x5-1)，循环次数-1 |
| 94 |  | bne x5, x0, loop | **bne t0, x0, 132** | 11111110000000101001100011100011 | fe0298e3 | SB | #loop循环累加 4次，结果存于：x18 |
| 98 |  | slli x12, x18, 0 | **slli a2, s2, 0** | 00000000000010010001011000010011 | 00091613 | I | #X12<- x18 ，X12 = 0x0000000a，函数调用结果存于：x12 |
| 9c |  | Jr ra | **jalr x0, 0(ra)** | 00000000000000001000000001100111 | 00008067 | I | #函数sum调用返回，回到pc = 30 |

### 2.2 指令存储器IP例化初始化文件产生

完成表1后，即可产生vivado FPGA开发平台指令存储器IP例化时的初始化文件：sc\_irom.coe。以便为后续CPU核设计提供必要的支持。该文件为文本格式，但是以coe作为后缀。格式如下。第一行表示机器码是16进制格式。第二行等号后开始列出代码段的机器码，可以用空格或者回车符分隔每条指令最终以分号结束。

memory\_initialization\_radix=16;

memory\_initialization\_vector=00000537

00056213

00100c93

00200d13

00300d93

00400e13

01922023

01a22223

01b22423

01c22623

00400293

054000ef

00c22023

00022983

40c98933

00300293

fff28293

fff2e913

55594913

fff00993

fff9fa13

013a6833

010a78b3

010a78b3

00028463

fddff06f

fff00293

00f29913

01091913

41095913

00f95913

0000006f

00000933

00022983

00420213

01390933

fff28293

fe0298e3

00091613

00008067

；

将前面补全的表1中的16进制机器码逐行写入，存为sc\_irom.coe。你也可以通过其他汇编器产生机器码后自己编写小程序自动存为该格式文件。

## 3 辅助仿真平台说明 仅供参考

#### 3.1仿真平台3

以下仿真平台3截图中的程序段比仿真平台2文本列出的测试代码有以下几点区别：

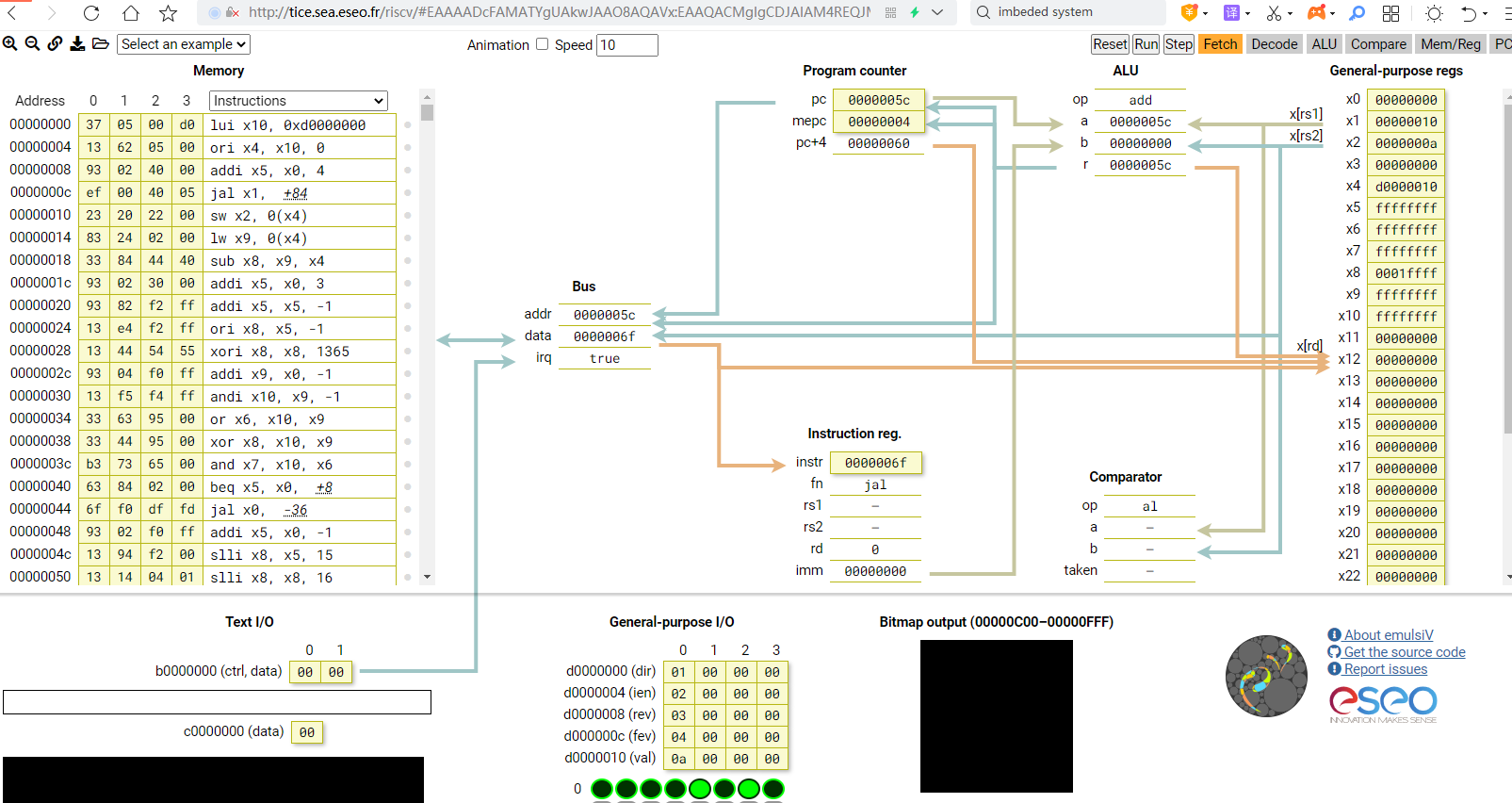
1. 少第3-10行
2. 使用的几个寄存器号不同
3. 读写的数据设置为基地址0xd0000000开始的连续五个字，该地址为仿真平台设定的一个界面上可直观读写的General-purpose I/O口。
4. 跳转指令里的偏移量写法与仿真平台2不同，需要明确数值。

请按照给出的仿真平台3 测试程序输入并单步执行仿真，查看仿真结果。理解指令的执行过程，协助理解仿真平台2测试程序及其仿真结果，完成对表1的补充。

给出的这两段程序其主要功能是完全一样的，只是为了在不同仿真平台看到更直观的仿真结果所作的更改：

在仿真平台3上仿真时，第1句改为lui x10, 0xd0000000，去掉3-10行，读取地址0xd00000000处开始的四个的IO口进行后续处理。其他不变。

在仿真平台2上仿真时，第2句可改为ori x4, x10, 1024，其他不变。对地址1024处开始的数据存储器进行操作和后续处理。以便于观察数据存储器内的变化结果。



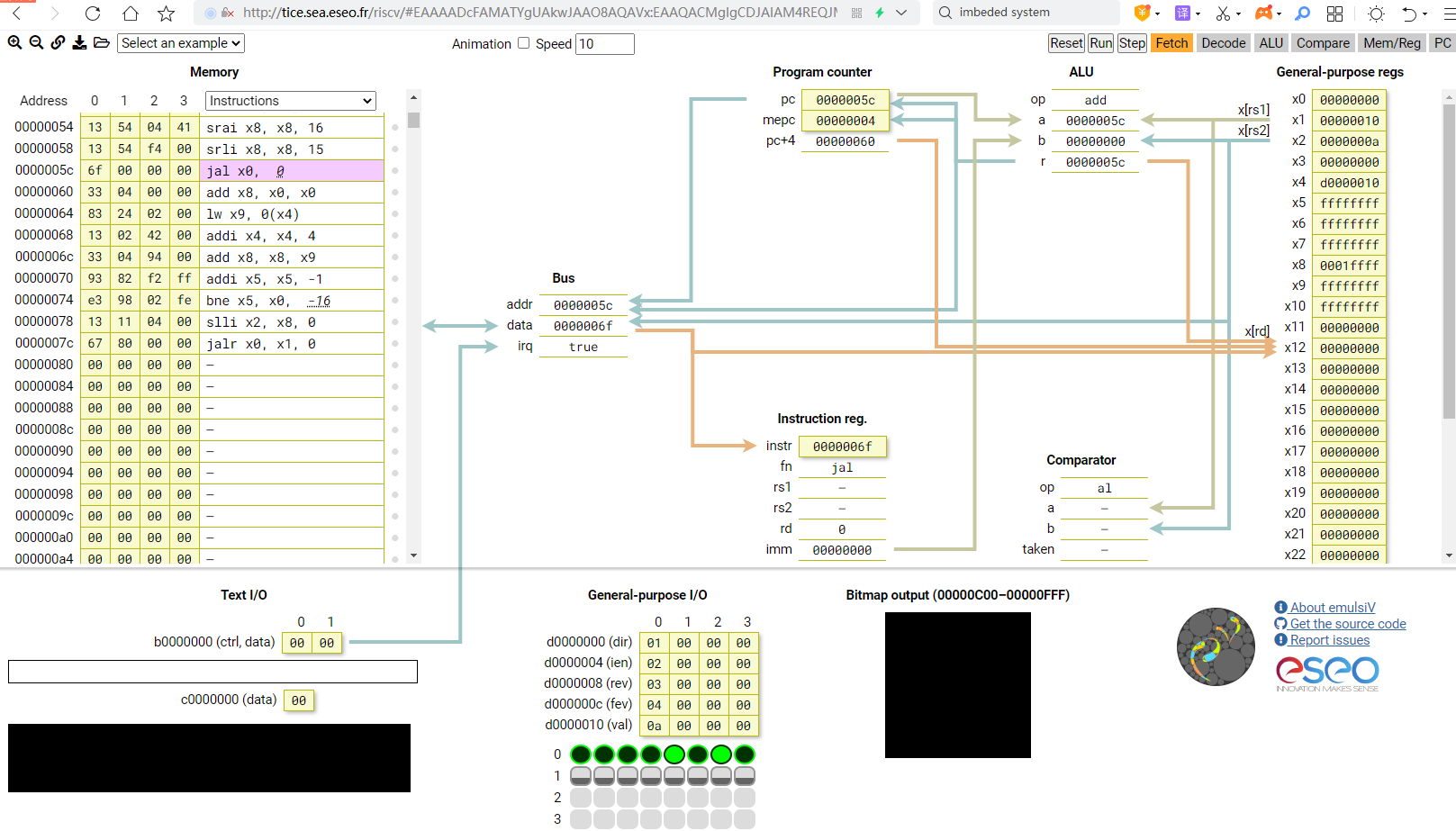
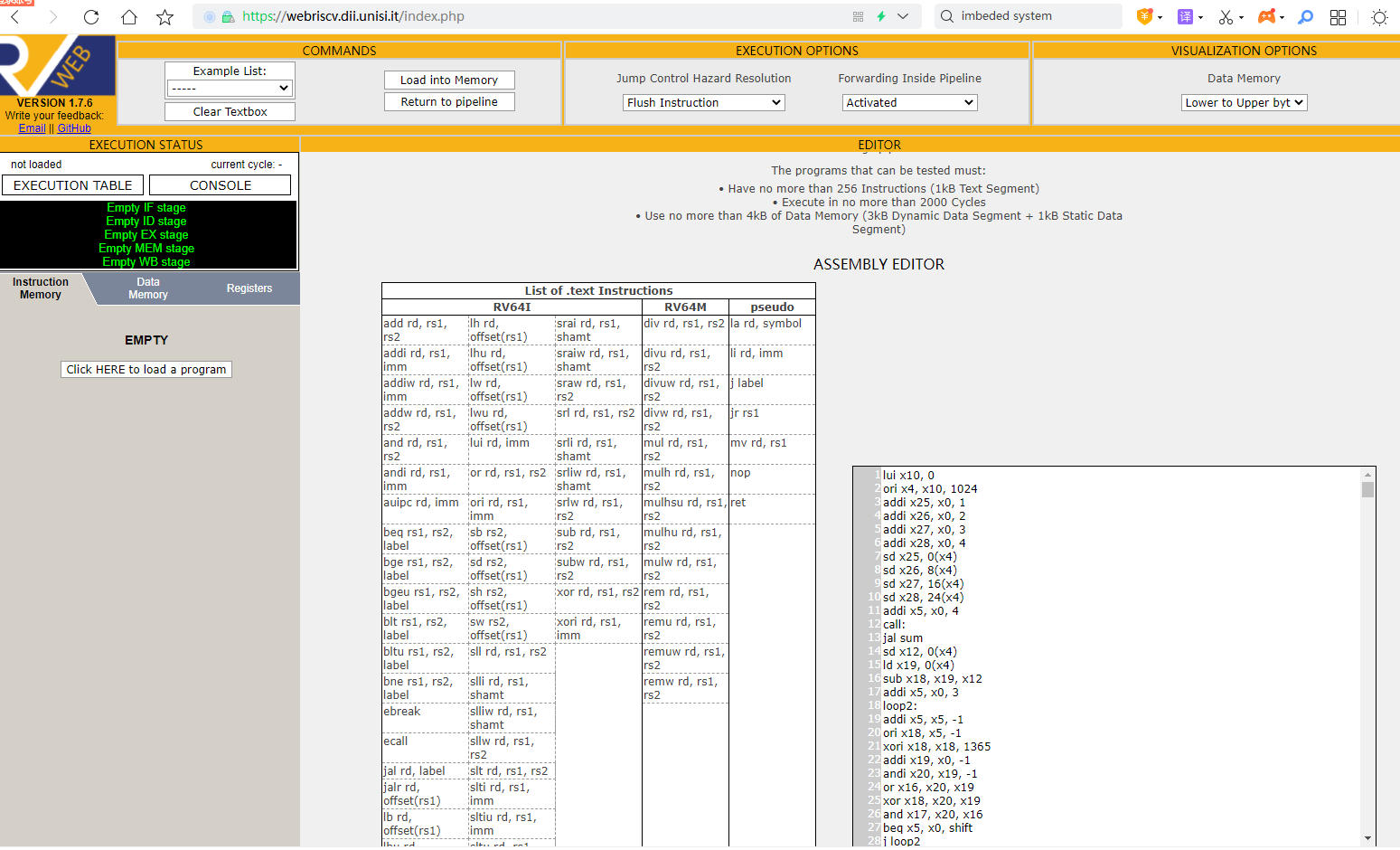
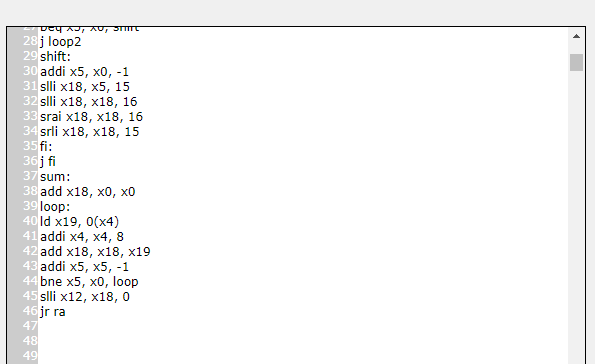


图1

#### 3.2仿真平台2

程序加载与仿真界面截图，如图2和图3。在仿真平台2上仿真时，第2句可改为ori x4, x10, 1024，其他不变。对地址1024处开始的数据存储器进行操作和后续处理。以便于观察数据存储器内的变化结果。



 图2

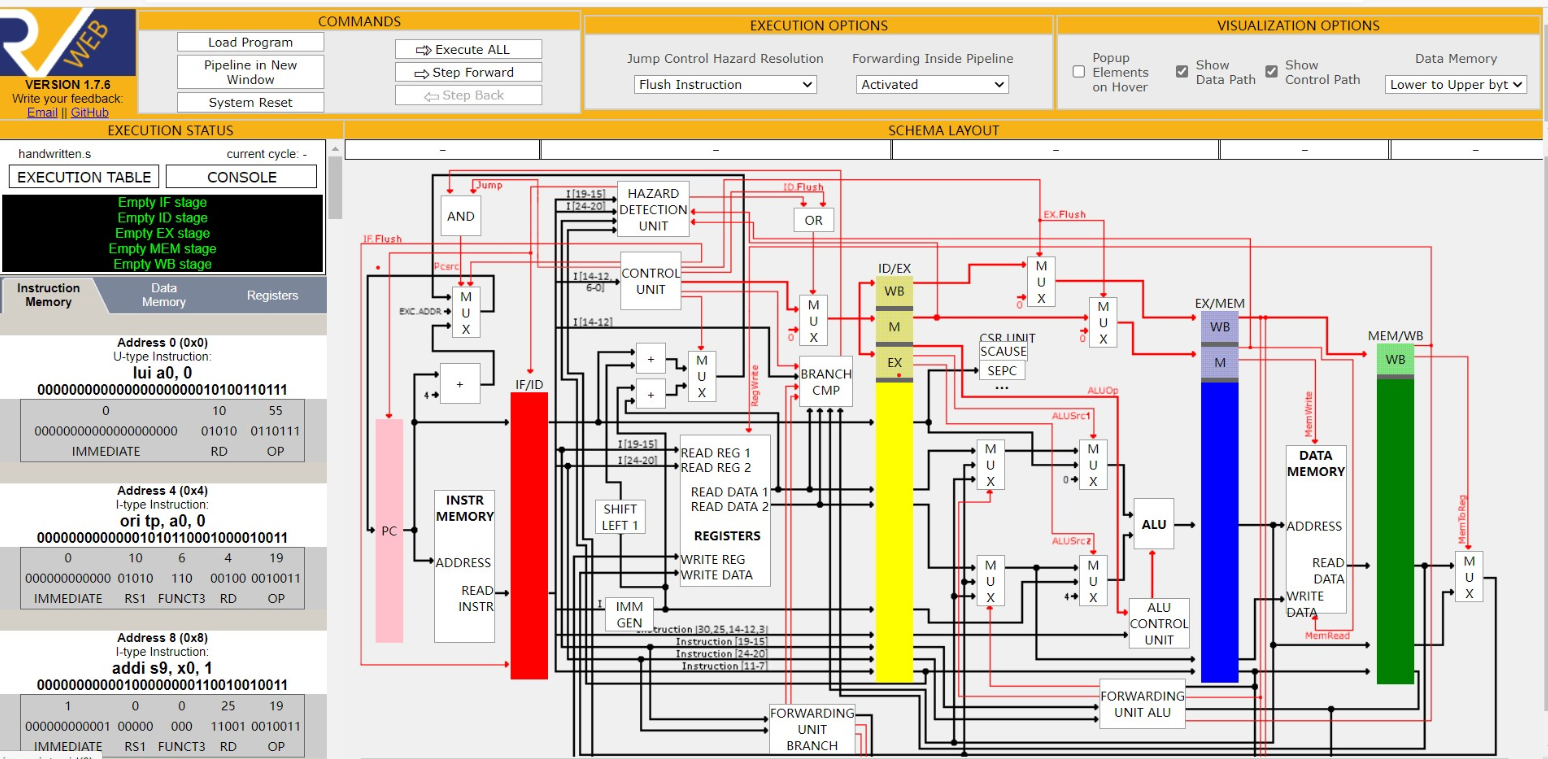


图3

仿真平台2测试程序加载后的指令和机器码截图，供参考。