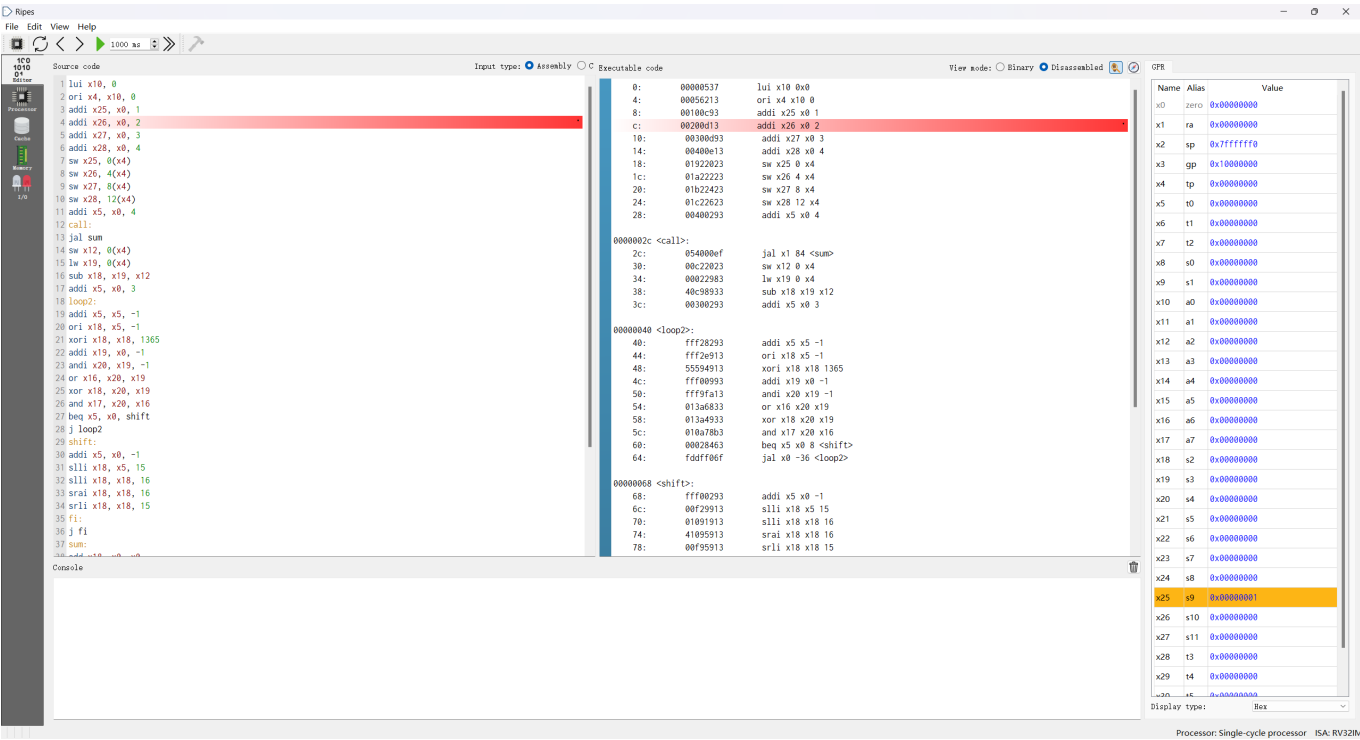


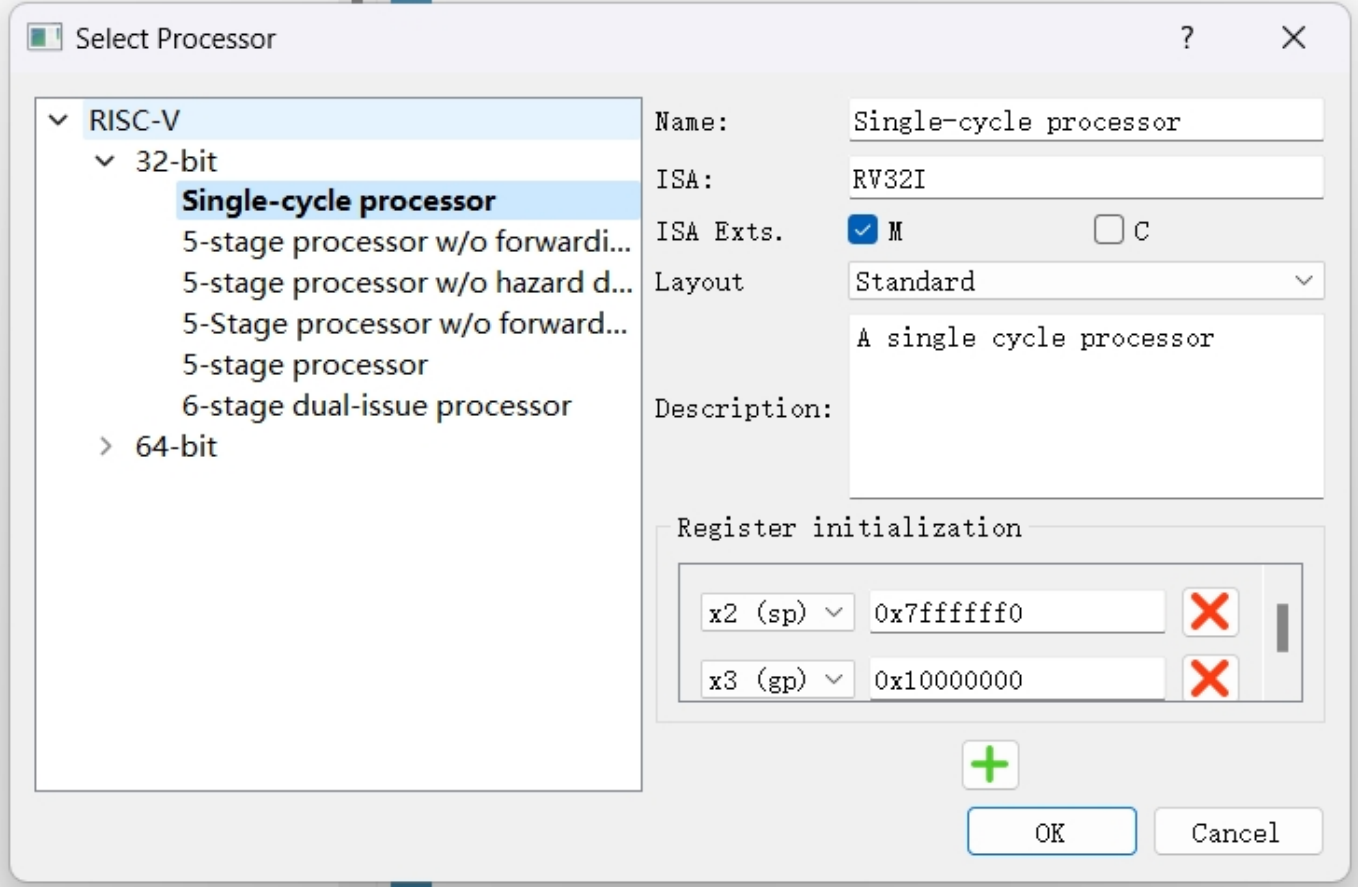
# Lab 2 报告

## 仿真说明

通过将doc文档中所给的代码复制到 Ripes 软件中，我们可以得到如下图一样的效果图，其中软件最右侧位我们提供了当指令运行时，各寄存器所存储的数值



随后，我们将处理器设置成单周期CPU模式：



通过指令的持续运行，根据相关信息，我们可以完成表格1的内容（表格1的内容在‘表1.pdf’中）

思考题

通过对内存结构的观察，我们可以发现初始代码将数据写在了内存的 '.text' 部分，以下是截图

写内存之前：

Memory viewer							Memory map		
Address	Word	Byte 0	Byte 1	Byte 2	Byte 3		Name	Size	Range
0x0000004c	0xffff0893	0x93	0x09	0xf0	0xff		.text	160	0x00000000 - 0x000000a0
0x00000048	0x55594913	0x13	0x49	0x59	0x55		.data	0	0x10000000 - 0x10000000
0x00000044	0xffff2e913	0x13	0xe9	0xf2	0xff		.bss	0	0x11000000 - 0x11000000
0x00000040	0xffff28293	0x93	0x82	0xf2	0xff				
0x0000003c	0x00300293	0x93	0x02	0x30	0x00				
0x00000038	0x40c58933	0x33	0x89	0xc9	0x40				
0x00000034	0x00022983	0x83	0x29	0x02	0x00				
0x00000030	0x00c22023	0x23	0x20	0xc2	0x00				
0x0000002c	0x054000ef	0xef	0x00	0x40	0x05				
0x00000028	0x00400293	0x93	0x02	0x40	0x00				
0x00000024	0x01c22623	0x23	0x26	0xc2	0x01				
0x00000020	0x01b22423	0x23	0x24	0xb2	0x01				
0x0000001c	0x01a22223	0x23	0x22	0xa2	0x01				
0x00000018	0x01922023	0x23	0x20	0x92	0x01				
0x00000014	0x00400e13	0x13	0x0e	0x40	0x00				
0x00000010	0x00300d93	0x93	0x0d	0x30	0x00				
0x0000000c	0x00200d13	0x13	0x0d	0x20	0x00				
0x00000008	0x00100c93	0x93	0x0c	0x10	0x00				
0x00000004	0x00056213	0x13	0x62	0x05	0x00				
0x00000000	0x00000537	0x37	0x05	0x00	0x00				
-	-	-	-	-	-				
-	-	-	-	-	-				
-	-	-	-	-	-				
-	-	-	-	-	-				

写内存之后：

Memory viewer						Memory map	
Address	Word	Byte 0	Byte 1	Byte 2	Byte 3	Name	Size
0x0000004c	0xffff0893	0x93	0x09	0xf0	0xff		
0x00000048	0x55594913	0x13	0x49	0x59	0x55		
0x00000044	0xffff2e913	0x13	0xe9	0xf2	0xff		
0x00000040	0xffff28293	0x93	0x82	0xf2	0xff		
0x0000003c	0x00300293	0x93	0x02	0x30	0x00		
0x00000038	0x40c38933	0x33	0x89	0xc9	0x40		
0x00000034	0x00022983	0x83	0x29	0x02	0x00		
0x00000030	0x00c22023	0x23	0x20	0xc2	0x00		
0x0000002c	0x054000ef	0xef	0x00	0x40	0x05		
0x00000028	0x00400293	0x93	0x02	0x40	0x00		
0x00000024	0x01c22623	0x23	0x26	0xc2	0x01		
0x00000020	0x01b22423	0x23	0x24	0xb2	0x01		
0x0000001c	0x01a22223	0x23	0x22	0xa2	0x01		
0x00000018	0x01922023	0x23	0x20	0x92	0x01		
0x00000014	0x00400e13	0x13	0x0e	0x40	0x00		
0x00000010	0x00300d93	0x93	0xd	0x30	0x00		
0x0000000c	0x00000004	0x04	0x00	0x00	0x00		
0x00000008	0x00000003	0x03	0x00	0x00	0x00		
0x00000004	0x00000002	0x02	0x00	0x00	0x00		
0x00000000	0x00000001	0x01	0x00	0x00	0x00		

		Range
.text	160	0x00000000 - 0x000000a0
.data	0	0x10000000 - 0x10000000
.bss	0	0x11000000 - 0x11000000

如果我们想把数据写在 '.data' 部分，我们可以改变对 x4 寄存器的操作，使其存储的值变为 **0x10000000**，这样数据便会存在 .data 部分。下面是截图：

代码修改

```
1 lui x10, 0
2 sri x4, x10, 0
3 lui x4, 0x10000000
4 addi x25, x0, 1
5 addi x26, x0, 2
6 addi x27, x0, 3
7 addi x28, x0, 4
8 sw x25, 0(x4)
9 sw x26, 4(x4)
10 sw x27, 8(x4)
11 sw x28, 12(x4)
12 addi x5, x0, 4
13 call:
14 jal sum
15 sw x12, 0(x4)
16 lw x19, 0(x4)
17 sub x18, x19, x12
18 addi x5, x0, 3
19 loop2:
20 addi x5, x5, -1
21 ori x18, x5, -1
22 xori x18, x18, 1365
23 addi x19, x0, -1
24 andi x20, x19, -1
25 or x16, x20, x19
26 xor x18, x20, x19
27 and x17, x20, x16
28 beq x5, x0, shift
29 j loop2
30 shift:
31 addi x5, x0, -1
32 slli x18, x5, 15
33 slli x18, x18, 16
34 srai x18, x18, 16
35 slli x18, x18, 15
36 fi:
37 j fi
--
```

```
0: 00000537 lui x10 0x0
4: 10000237 lui x4 0x10000
8: 00100c93 addi x25 x0 1
c: 00200d13 addi x26 x0 2
10: 00300d93 addi x27 x0 3
14: 00400e13 addi x28 x0 4
18: 01922023 sw x25 0 x4
1c: 01a22223 sw x26 4 x4
20: 01b22423 sw x27 8 x4
24: 01c22623 sw x28 12 x4
28: 00400293 addi x5 x0 4

0000002c <call>:
2c: 054000ef jal x1 84 <sum>
30: 00c22023 sw x12 0 x4
34: 00022983 lw x19 0 x4
38: 40c38933 sub x18 x19 x12
3c: 00300293 addi x5 x0 3

00000040 <loop2>:
40: fff28293 addi x5 x5 -1
44: fff2e913 ori x18 x5 -1
48: 55594913 xori x18 x18 1365
4c: fff00993 addi x19 x0 -1
50: fff9fa13 andi x20 x19 -1
54: 013a6833 or x16 x20 x19
58: 013a4933 xor x18 x20 x19
5c: 010a7863 and x17 x20 x16
60: 00028463 beq x5 x0 0 <shift>
64: fddff06f jal x0 -36 <loop2>

00000068 <shift>:
68: fff00293 addi x5 x0 -1
6c: 00f29913 slli x18 x5 15
70: 01091913 slli x18 x18 16
74: 41095913 srai x18 x18 16
78: 00f95913 slli x18 x18 15
```

Name	Alias	Value
x0	zero	0x00000000
x1	ra	0x00000000
x2	sp	0x7fffffff
x3	gp	0x10000000
x4	tp	0x00000000
x5	t0	0x00000000
x6	t1	0x00000000
x7	t2	0x00000000
x8	s0	0x00000000
x9	s1	0x00000000
x10	a0	0x00000000
x11	a1	0x00000000
x12	a2	0x00000000
x13	a3	0x00000000
x14	a4	0x00000000
x15	a5	0x00000000
x16	a6	0x00000000
x17	a7	0x00000000
x18	a2	0x00000000
x19	s3	0x00000000
x20	s4	0x00000000
x21	s5	0x00000000
x22	s6	0x00000000
x23	s7	0x00000000
x24	s8	0x00000000
x25	s9	0x00000000
x26	s10	0x00000000
x27	s11	0x00000000
x28	t3	0x00000000
x29	t4	0x00000000

Display type: Hex

代码简要说明：lui指令的全称是Load Upper Immediate，它的功能是把一个20位的立即数加载到寄存器的高20位，低12位为0。它的格式是：lui rd, imm

写入 .data 部分之前

Memory viewer							Memory map		
Address	Word	Byte 0	Byte 1	Byte 2	Byte 3		Name	Size	Range
0x10000040	X	X	X	X	X		.text	160	0x00000000 - 0x000000a0
0x1000003c	X	X	X	X	X		.data	0	0x10000000 - 0x10000000
0x10000038	X	X	X	X	X		.bss	0	0x10000000 - 0x10000000
0x10000034	X	X	X	X	X				
0x10000030	X	X	X	X	X				
0x1000002c	X	X	X	X	X				
0x10000028	X	X	X	X	X				
0x10000024	X	X	X	X	X				
0x10000020	X	X	X	X	X				
0x1000001c	X	X	X	X	X				
0x10000018	X	X	X	X	X				
0x10000014	X	X	X	X	X				
0x10000010	X	X	X	X	X				
0x1000000c	X	X	X	X	X				
0x10000008	X	X	X	X	X				
0x10000004	X	X	X	X	X				
0x10000000	X	X	X	X	X				
0x0fffffc	X	X	X	X	X				
0x0fffffb	X	X	X	X	X				
0x0fffff4	X	X	X	X	X				
0x0fffff0	X	X	X	X	X				
0x0ffffec	X	X	X	X	X				
0x0ffffe8	X	X	X	X	X				
0x0ffffe4	X	X	X	X	X				
0x0ffffe0	X	X	X	X	X				
0x0ffffdc	X	X	X	X	X				
0x0ffffd8	X	X	X	X	X				
0x0ffffd4	X	X	X	X	X				
0x0ffffd0	X	X	X	X	X				
0x0ffffcc	X	X	X	X	X				
0x0ffffc8	X	X	X	X	X				

Display type: HexGo to registers:Go to section:

Processor: Single-cycle processorISA: RV32IM

写入 .data 部分之后

Memory viewer							Memory map		
Address	Word	Byte 0	Byte 1	Byte 2	Byte 3		Name	Size	Range
0x10000040	X	X	X	X	X		.text	160	0x00000000 - 0x000000a0
0x1000003c	X	X	X	X	X		.data	0	0x10000000 - 0x10000000
0x10000038	X	X	X	X	X		.bss	0	0x10000000 - 0x10000000
0x10000034	X	X	X	X	X				
0x10000030	X	X	X	X	X				
0x1000002c	X	X	X	X	X				
0x10000028	X	X	X	X	X				
0x10000024	X	X	X	X	X				
0x10000020	X	X	X	X	X				
0x1000001c	X	X	X	X	X				
0x10000018	X	X	X	X	X				
0x10000014	X	X	X	X	X				
0x10000010	X	X	X	X	X				
0x1000000c	0x00000004	0x04	0x00	0x00	0x00				
0x10000008	0x00000003	0x03	0x00	0x00	0x00				
0x10000004	0x00000002	0x02	0x00	0x00	0x00				
0x10000000	0x00000001	0x01	0x00	0x00	0x00				
0x0fffffc	X	X	X	X	X				
0x0fffffb	X	X	X	X	X				
0x0fffff4	X	X	X	X	X				
0x0fffff0	X	X	X	X	X				
0x0ffffec	X	X	X	X	X				
0x0ffffe8	X	X	X	X	X				
0x0ffffe4	X	X	X	X	X				
0x0ffffe0	X	X	X	X	X				
0x0ffffdc	X	X	X	X	X				
0x0ffffd8	X	X	X	X	X				
0x0ffffd4	X	X	X	X	X				
0x0ffffd0	X	X	X	X	X				
0x0ffffcc	X	X	X	X	X				
0x0ffffc8	X	X	X	X	X				

Display type: HexGo to registers:Go to section:

Processor: Single-cycle processorISA: RV32IM

通过对代码的修改，我们成功将数据写入 .data 部分