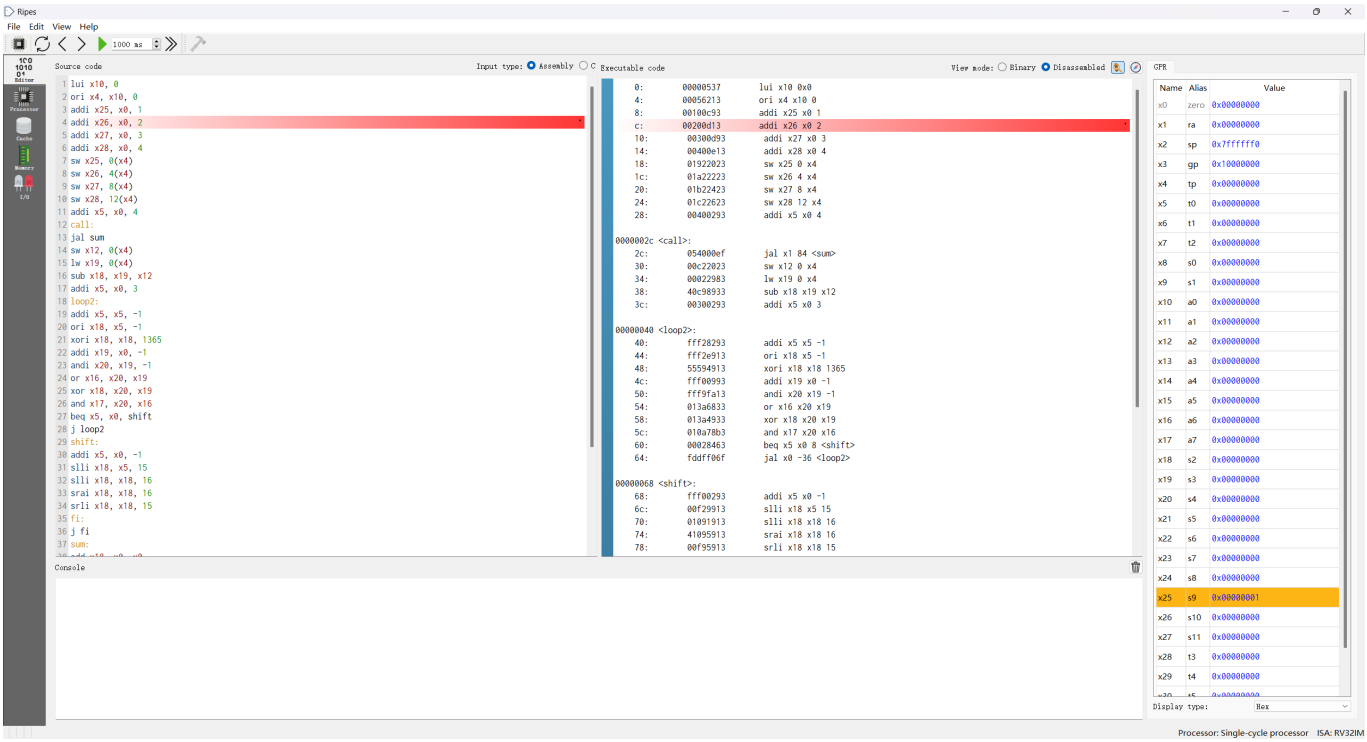
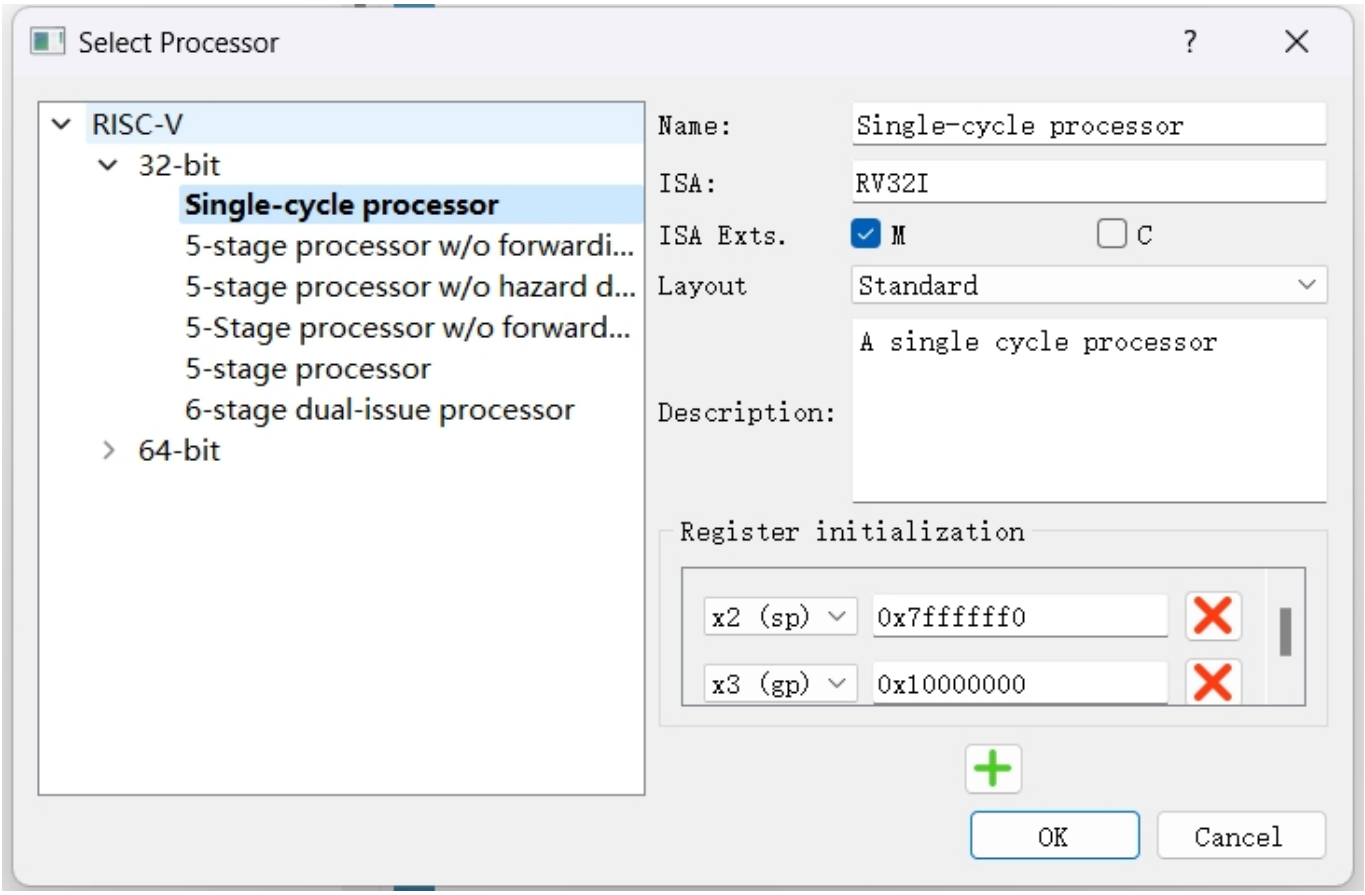
Lab 2 报告

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**：

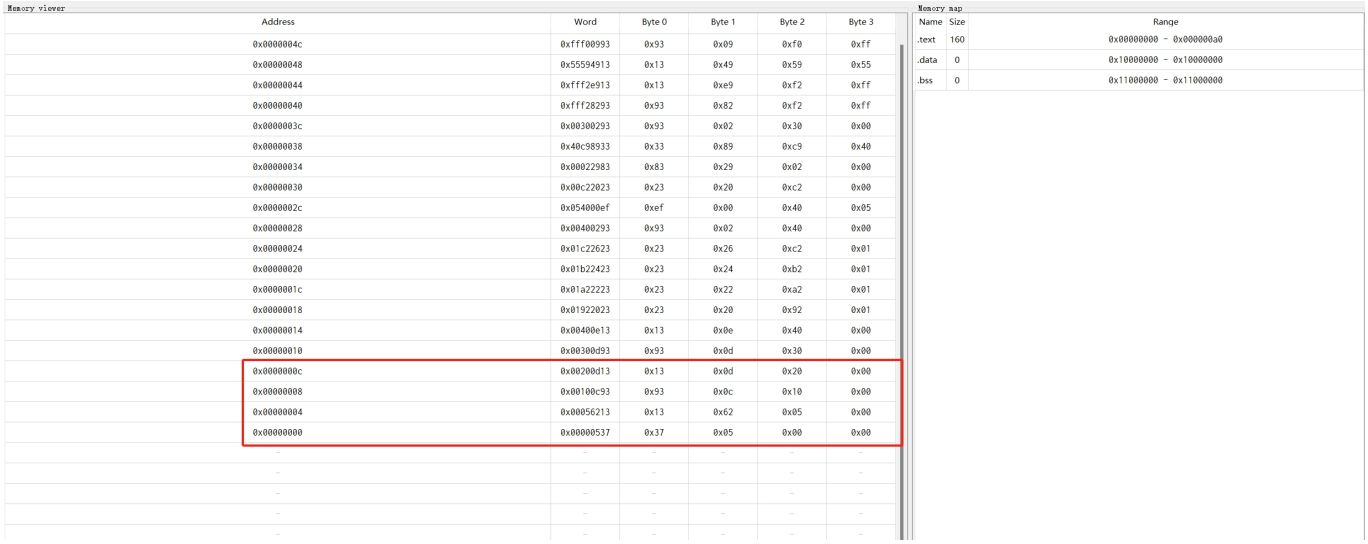


通过指令的持续运行，根据相关信息，我们可以完成表格1的内容（表格1的内容在'表1.pdf'中）

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：



Address	Word	Byte 0	Byte 1	Byte 2	Byte 3
0x0000004c	0xfff00993	0x93	0x09	0xf0	0xff
0x00000048	0x55594913	0x13	0x49	0x59	0x55
0x00000044	0xfff2e913	0x13	0xe9	0xf2	0xff
0x00000040	0xfff28293	0x93	0x82	0xf2	0xff
0x0000003c	0x00300293	0x93	0x02	0x30	0x00
0x00000038	0x40c98933	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

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

写内存之后：

Memory viewer

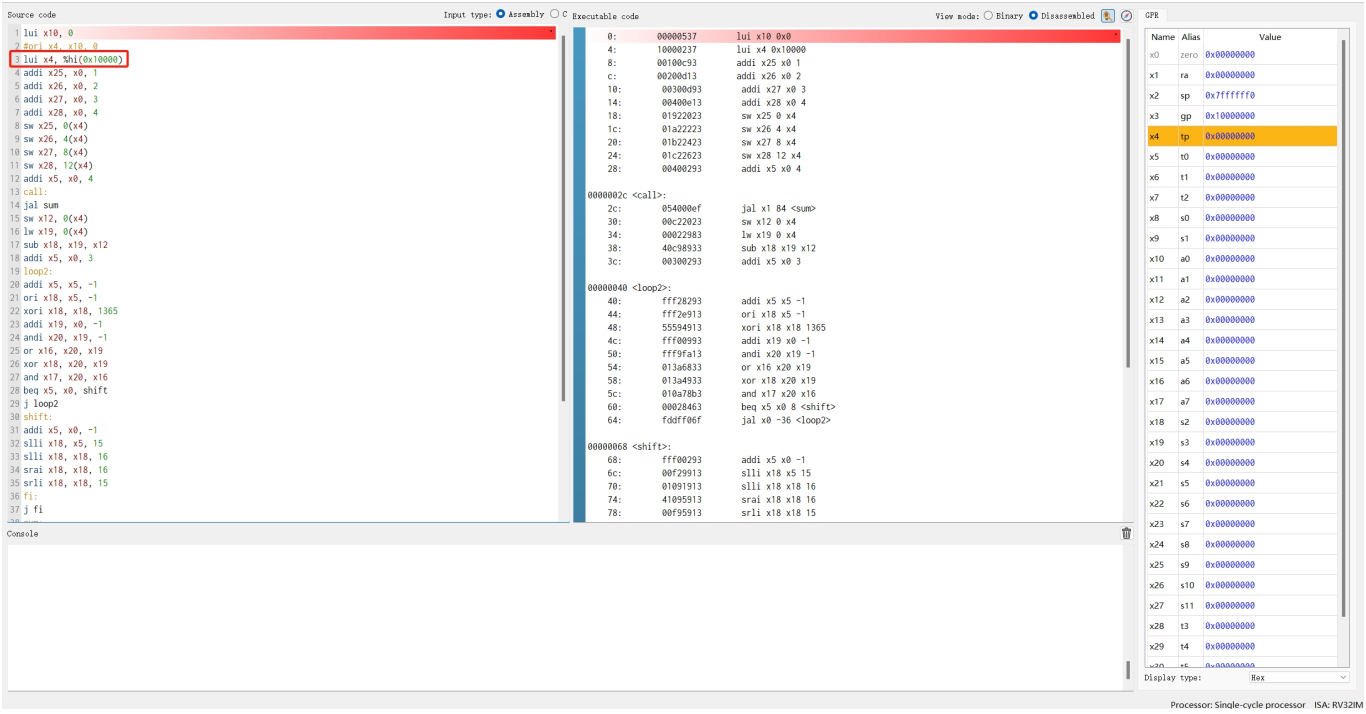
Address	Word	Byte 0	Byte 1	Byte 2	Byte 3
0x0000004c	0xfff00993	0x93	0x09	0xf0	0xff
0x00000048	0x55594913	0x13	0x49	0x59	0x55
0x00000044	0xfff2e913	0x13	0xe9	0xf2	0xff
0x00000040	0xfff28293	0x93	0x82	0xf2	0xff
0x0000003c	0x00300293	0x93	0x02	0x30	0x00
0x00000038	0x40c98933	0x33	0x89	0xc9	0x40
0x00000034	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	0x00000004	0x04	0x00	0x00	0x00
0x00000008	0x00000003	0x03	0x00	0x00	0x00
0x00000004	0x00000002	0x02	0x00	0x00	0x00
0x00000000	0x00000001	0x01	0x00	0x00	0x00

Memory map

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

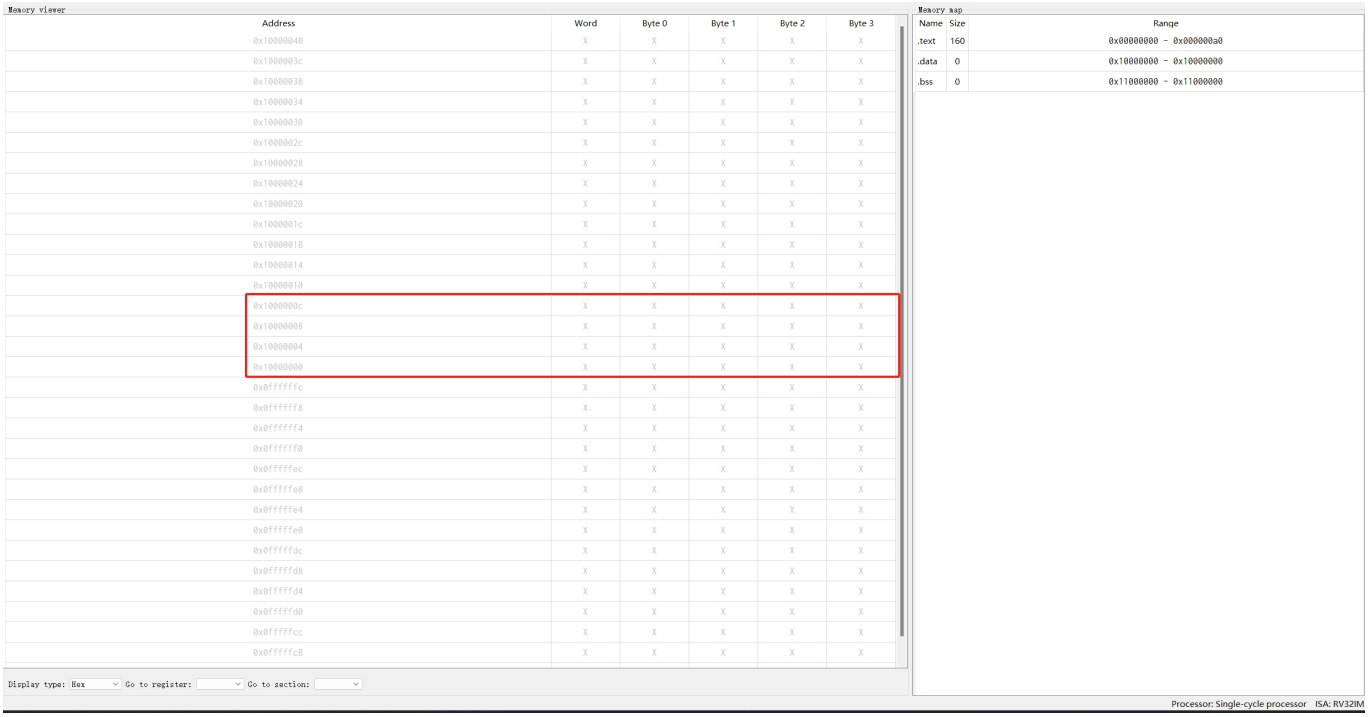
如果我们想把数据写在 '.data' 部分，我们可以改变对 x4 寄存器的操作，使其存储的值变为**0x10000000**，这样数据便会存在 .data 部分。下面是截图：

代码修改

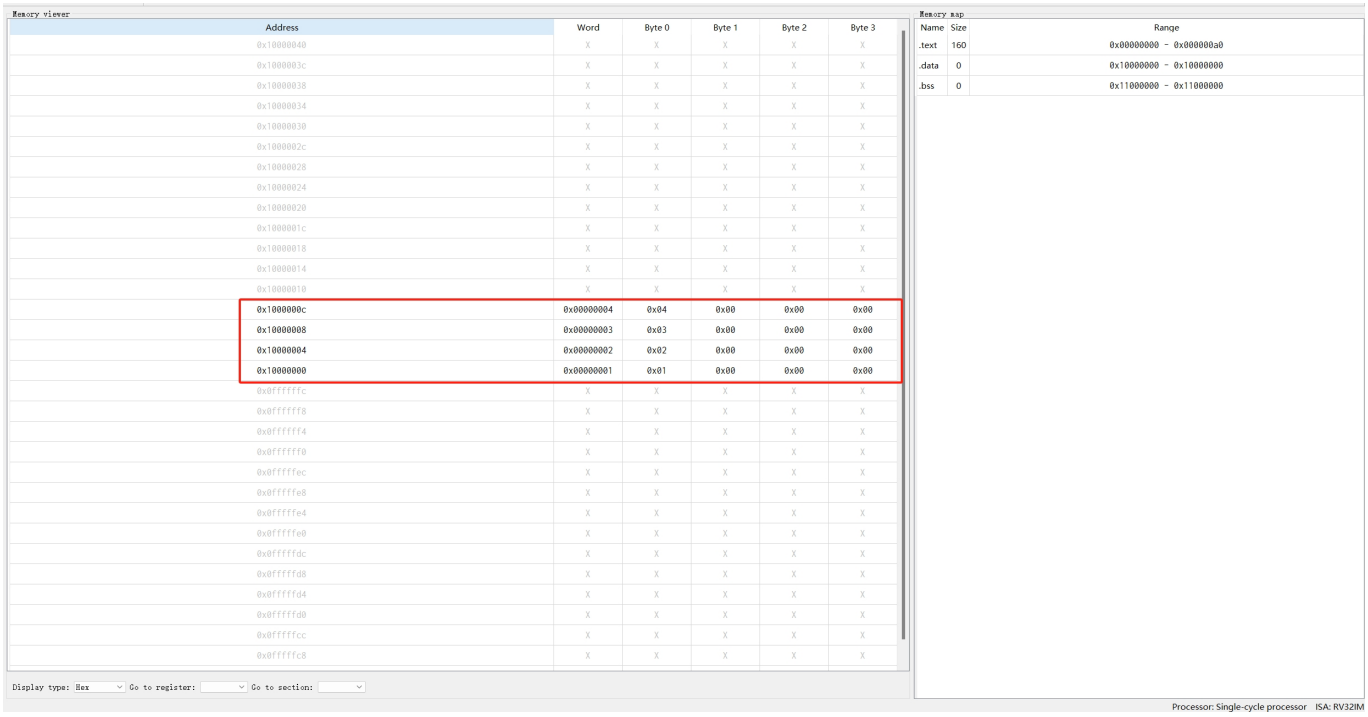


代码简要说明：lui指令的全称是Load Upper Immediate，它的功能是把一个20位的立即数加载到寄存器的高20位，低12位为0。它的格式是：lui rd, imm

写入 .data 部分之前



写入 .data 部分之后



通过对代码的修改，我们成功将数据写入 .data 部分