实验一 MIPS汇编实验

一、 实验目的

1、学习MARS、logisim的使用；

2、学习MIPS汇编语言，熟悉并理解MIPS指令系统；

3、了解程序在单周期MIPS-CPU搭建的SOC中的运行。

二、 实验题目

1、实验题目一-输入输出实验

LED和拨码开关的端口地址如下：

表1 LED、拨码开关端口地址

接口部件	首地址
低 16 位 LED	0XFFFFFC60
高 8 位 LED	0XFFFFFC62
低 16 位拨码开关	0XFFFFFC70
高 8 位拨码开关	0XFFFFFC72

程序见Exercise3.1.asm。

实现功能：不断地从拨码开关读出数据，并将数据输出到对应的LED灯上。拨码开关为1时LED灯亮。

根据实验指导书，运行程序，熟悉实验过程。

2、实验题目二-乘法运算

以原码一位乘为基础，设计一个数的平方和立方2种运算。由拨码开关SW3-SW0输入一个数，SW22为平方，SW23为立方，均为高电平有效。结果输出到YLD15~YLD0同时保存到存储器中。一次运算结束后使用SW21清零（高电平有效），等待进行下一次运算。

3、实验题目三-节日彩灯

设计一个有24个发光二极管的彩灯程序。循环执行，每隔大约半秒变换一次，可设置如下变换模式：

模式1：LED灯从两边向中间依次点亮，再从中间向两边依次熄灭。

模式2：从左向右依次亮1、2、3…24盏LED灯，至24盏LED灯全亮后，从左向右灭1、2、3…24盏LED灯，至所有灯全灭。

模式3：由拨码开关控制LED灯点亮连续X位（X由拨码开关输入），并循环右移。

Logisim时钟频率设置为4.1KHz：

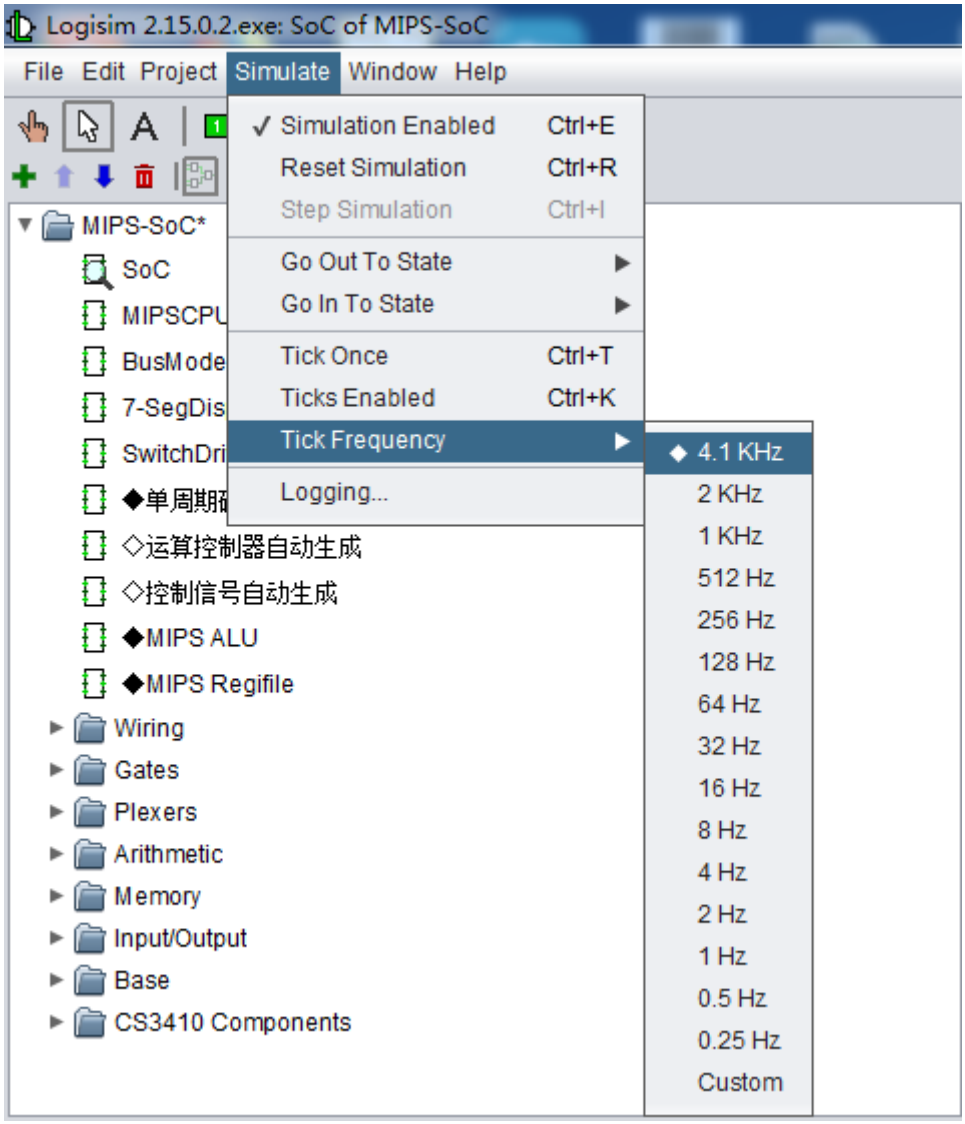


图1 logisim频率设置

三、 实验步骤

1、MIPS程序汇编

MARS（MIPS汇编程序和运行时模拟器）是一个轻量级的交互式开发环境（IDE），用于使用MIPS汇编语言进行编程。具有代码提示，模拟运行，调试，统计等功能。

1）界面简介

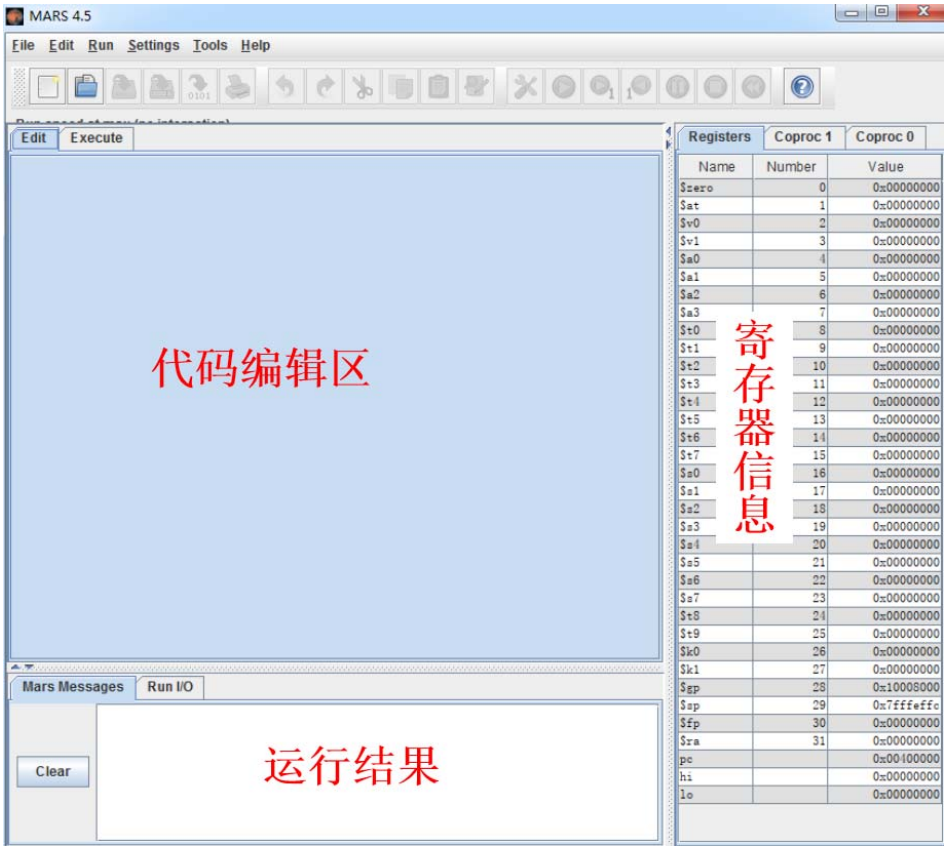


图2 MARS界面图

2）运行方法

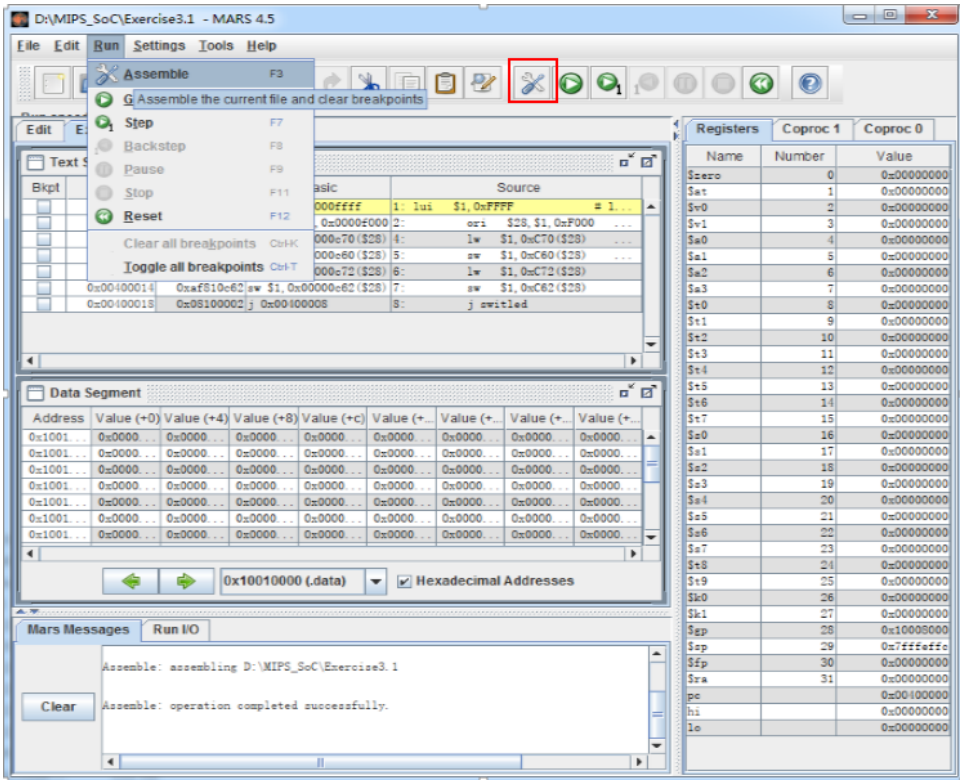


图3 MARS运行

3）导出十六进制

程序汇编后可以利用File菜单中的Dump Memory功能将代码段和数据段导出，采用十六进制文本的方式导出到 “**.hex”，即可在LOGISIM中加载到RAM或ROM中。

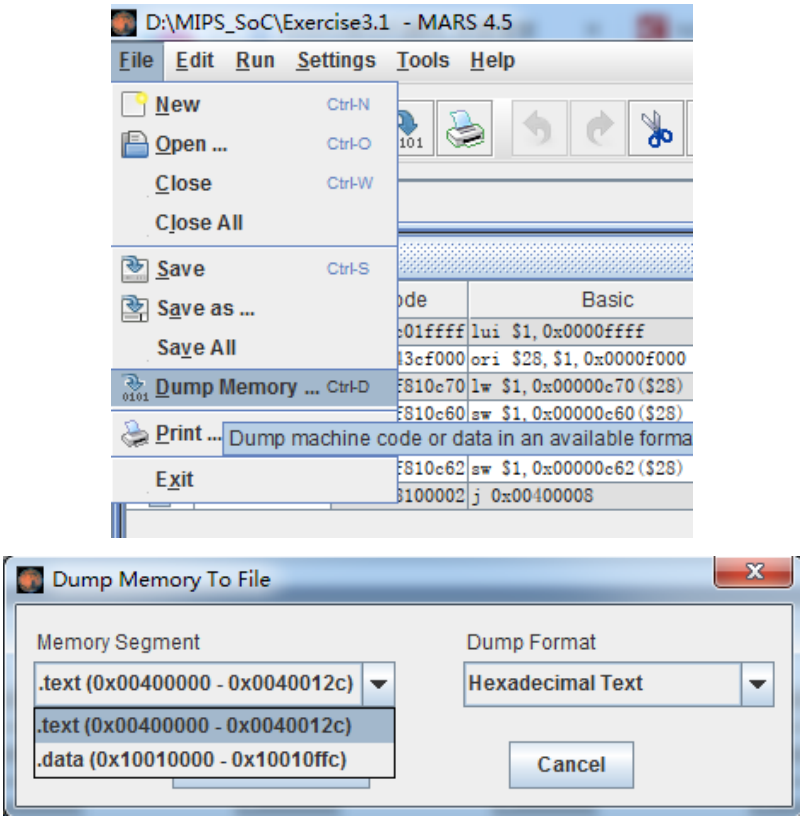


图4 导出十六进制文件

注意：.TEXT是存储在指令存储器中，.DATA 生成的数据是存储在数据存储器中。

2、程序运行

1）SOC电路

在logisim中打开“MIPS-SoC.circ”。图为MIPS单周期CPU、指令存储器、数据存储器及外设构成的SOC电路。

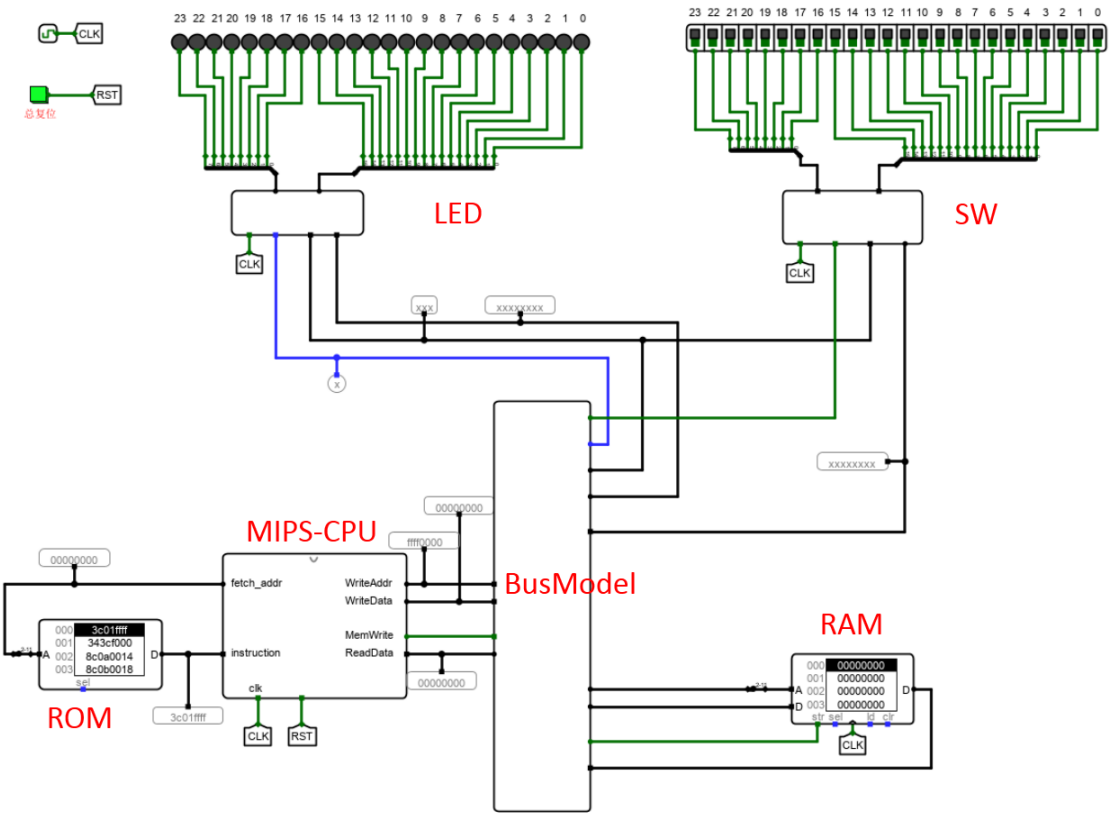


图5 SOC电路图

支持23条指令，如下表：

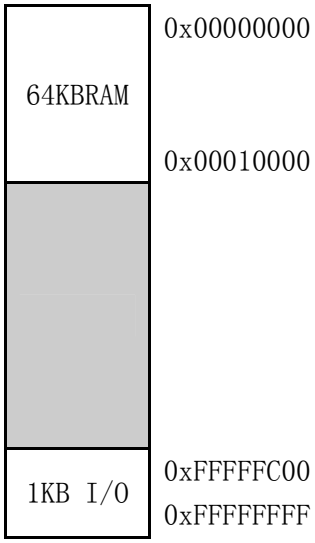
表2 23条MIPS汇编指令表

助记符	指令格式						示 例	示例含义	操 作 及 解 释
BIT #	31..26	25..21	20..16	15..11	10..6	5..0			
R-类型	op	rs	rt	rd	shamt	func			
add	000000	rs	rt	rd	00000	100000	add $1,$2,$3	$1=$2+S3	(rd)←(rs)+(rt); rs=$2,rt=$3,rd=$1
addu	000000	rs	rt	rd	00000	100001	addu $1,$2,$3	$1=$2+S3	(rd)←(rs)+(rt); rs=$2,rt=$3,rd=$1
sub	000000	rs	rt	rd	00000	100010	sub $1,$2,$3	$1=$2-S3	(rd)←(rs)-(rt); rs=$2,rt=$3,rd=$1
and	000000	rs	rt	rd	00000	100100	and $1,$2,$3	$1=$2&S3	(rd)←(rs)&(rt); rs=$2,rt=$3,rd=$1
or	000000	rs	rt	rd	00000	100101	or $1,$2,$3	$1=$2|S3	(rd)←(rs) | (rt); rs=$2,rt=$3,rd=$1
nor	000000	rs	rt	rd	00000	100111	nor $1,$2,$3	$1= ~($2 | S3)	(rd)←~((rs) | (rt)); rs=$2,rt=$3,rd=$1
slt	000000	rs	rt	rd	00000	101010	slt $1,$2,$3	if($2<$3) $1=1 else $1=0	if (rs< rt) rd=1 else rd=0;rs＝$2，rt=$3, rd=$1
sll	000000	00000	rt	rd	shamt	000000	sll $1,$2,10	$1=$2<<10	(rd)←(rt)<<shamt,rt=$2,rd=$1,shamt=10
srl	000000	00000	rt	rd	shamt	000010	srl $1,$2,10	$1=$2>>10	(rd)←(rt)>>shamt, rt=$2, rd=$1, shamt=10, (逻辑右移)
sra	000000	00000	rt	rd	shamt	000011	sra $1,$2,10	$1=$2>>10	(rd)←(rt)>>shamt, rt=$2, rd=$1, shamt=10, (算术右移，注意符号位保留)
jr	000000	rs	00000	00000	00000	001000	jr $31	goto $31	(PC)←(rs)
I-类型	op	rs	rt	immediate					
addi	001000	rs	rt	immediate			addi $1,$2,10	$1=$2+10	(rt)←(rs)+(sign-extend)immediate,rt=$1,rs=$2
addiu	001001	rs	rt	immediate			addiu $1,$2,10	$1=$2+10	(rt)←(rs)+(sign-extend)immediate,rt=$1,rs=$2
andi	001100	rs	rt	immediate			andi $1,$2,10	$1=$2&10	(rt)←(rs)&(zero-extend)immediate,rt=$1,rs=$2
ori	001101	rs	rt	immediate			ori $1,$2,10	$1=$2|10	(rt)←(rs)|(zero-extend)immediate,rt=$1,rs=$2
lui	001111	00000	rt	immediate			lui $1,10	$1=10*65536	(rt)←immediate<<16 & 0FFFF0000H，将 16 位立即数放到目的寄存器高 16 位，目的寄存器的低 16 位填 0
lw	100011	rs	rt	offset			lw $1,10($2)	$1=Memory[$2+10]	(rt)←Memory[(rs)+(sign_extend)offset], rt=$1,rs=$2
sw	101011	rs	rt	offset			sw $1,10($2)	Memory[$2+10] =$1	Memory[(rs)+(sign_extend)offset]←(rt), rt=$1,rs=$2
beq	000100	rs	rt	offset			beq $1,$2,40	if($1=$2)	if ((rt)=(rs)) then (PC)←(PC)+4+((Sign-Extend)

						goto PC+4+40	offset<<2), rs=$1, rt=$2
bne	000101	rs	rt	offset	bne $1,$2,40	if($1≠$2) goto PC+4+40	if ((rt)≠(rs)) then (PC)←(PC)+4+((Sign-Extend) offset<<2) , rs=$1, rt=$2
slti	001010	rs	rt	immediate	slti $1,$2,10	if($2<10) $1=1 else $1=0	if ((rs)<(Sign-Extend)immediate) then (rt)←1; else (rt)←0, rs=$2, rt=$1
J-类型	op	address					
j	000010	address			j 10000	goto 10000	(PC)←((Zero-Extend) address<<2), address=10000/4
jal	000011	address			jal 10000	$31=PC+4 goto 10000	($31)←(PC)+4; (PC)←((Zero-Extend) address<<2), address=10000/4

采用I/O统一编制的方式，将32位地址的存储空间分区，其中高1024个字节用来做I/O。

表3 地址空间



2）程序测试

在指令存储器和数据存储器上点击右键，选择“清空内容”，然后再选择“编辑内容”。 把“**.hex”文件的内容粘贴到编辑器中，即：将测试程序的汇编完成的指令存放到指令存储器中。

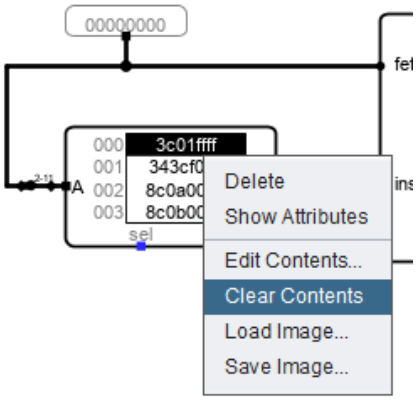


图6 编辑存储器内容

使用ctrl + k ： 时钟连续，ctrl + t ： 时钟单步，可以看到程序执行情况。

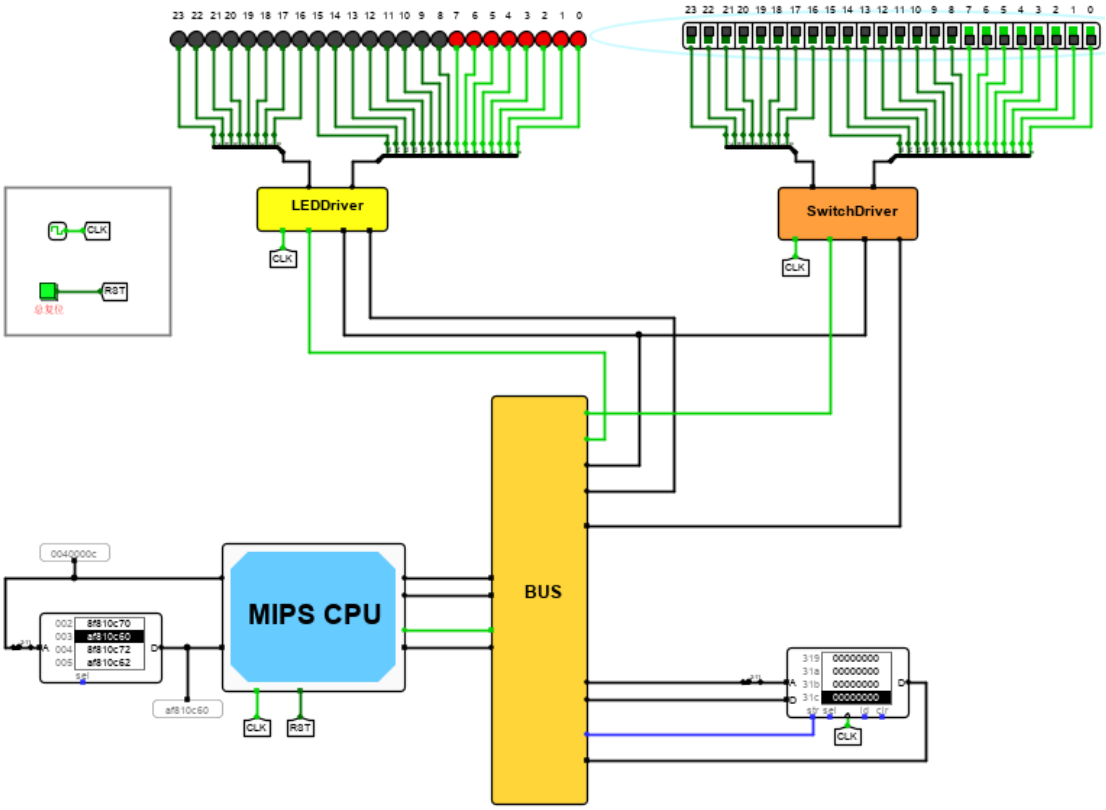


图7 程序执行情况