# 实验一 MIPS 指令系统和MIPS体系结构

陈朴炎 2021211138

## 1 实验目的

(1) 了解和熟悉指令级模拟器。

(2) 熟练掌握MIPSsim模拟器的操作和使用方法。

(3) 熟悉MIPS指令系统及其特点，加深对MIPS指令操作语义的理解。

(4) 熟悉MIPS体系结构。

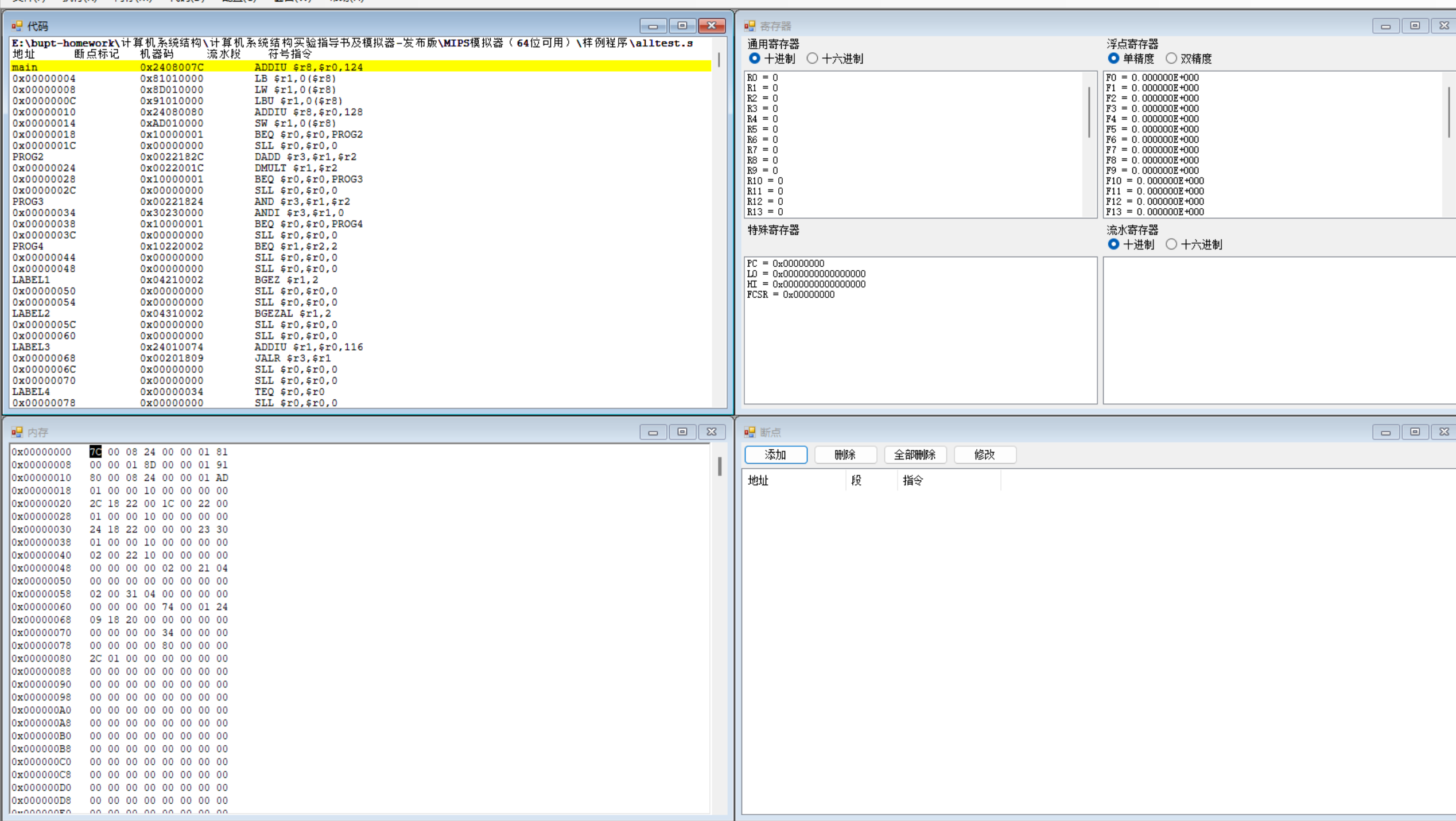
## 2 实验平台

实验平台采用指令级和流水线操作级模拟器MIPSsim。

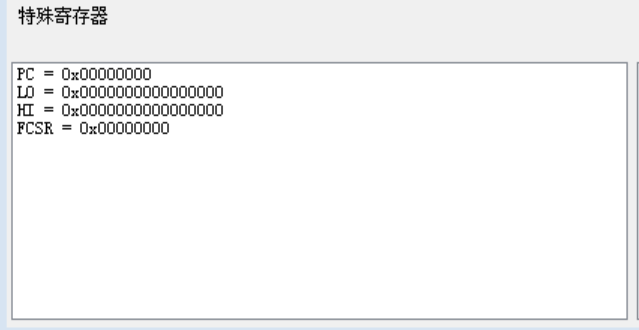
## 3 实验内容和步骤

首先要阅读附录中的MIPSsim模拟器的使用方法，然后了解MIPSsim的指令系统和汇编语言。之后在MIPSsim模拟器中载入样例程序alltest.s，在模拟器中进行非流水执行，查看并分析每条指令执行后相关寄存器的值。

首先配置非流水线方式，并载入了alltest.s程序。如下图所示：

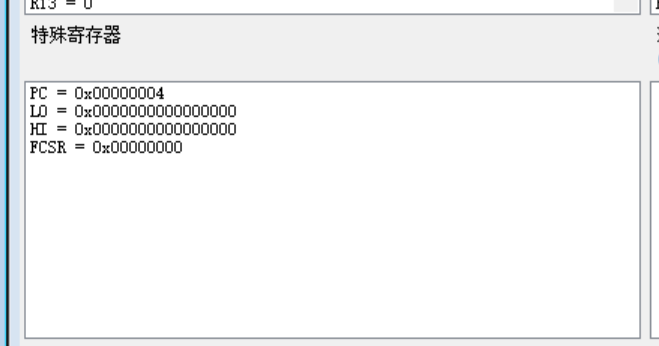


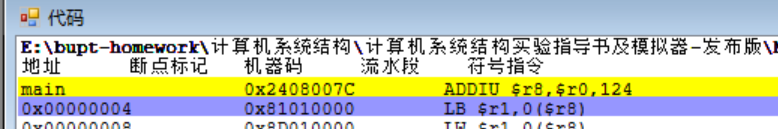
此时，特殊寄存器内的值如下所示：



PC值为 [PC]=0x00000000。

按下F7，单步执行一条指令，PC变为0x00000004,如下所示：



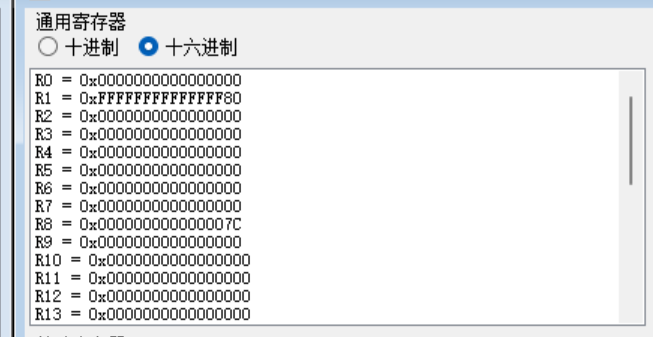


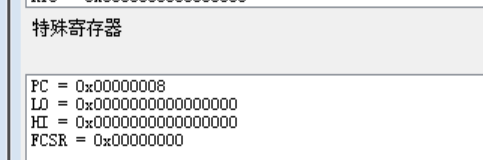
下一条地址0x00000004

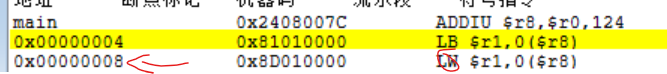
机器码为0x81010000 符号指令为LB $r1, 0($r8)

LB表示是一条有符号载入字节指令。

再按下F7，查看R1的值，为0xFFFFFFFFFFFFFF80, 如下：

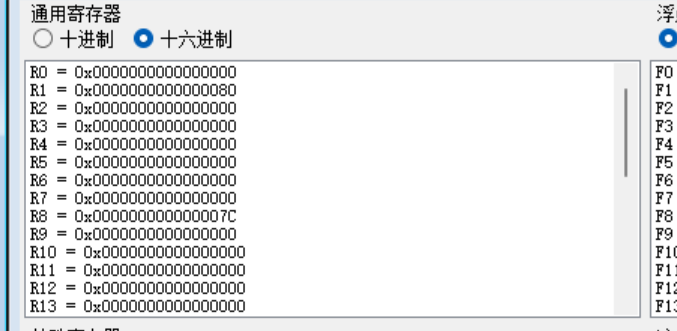




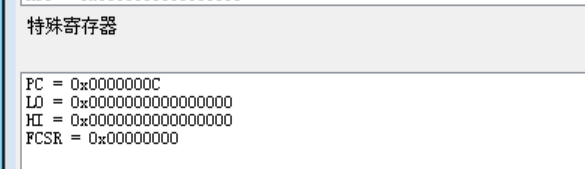


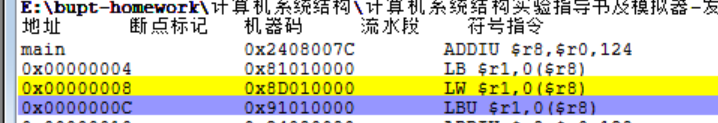
下一条指令地址为PC=0x00000008。LW表示这是一条有符号载入字指令。

再按下F7，R1值变为0x0000000000000080，如下



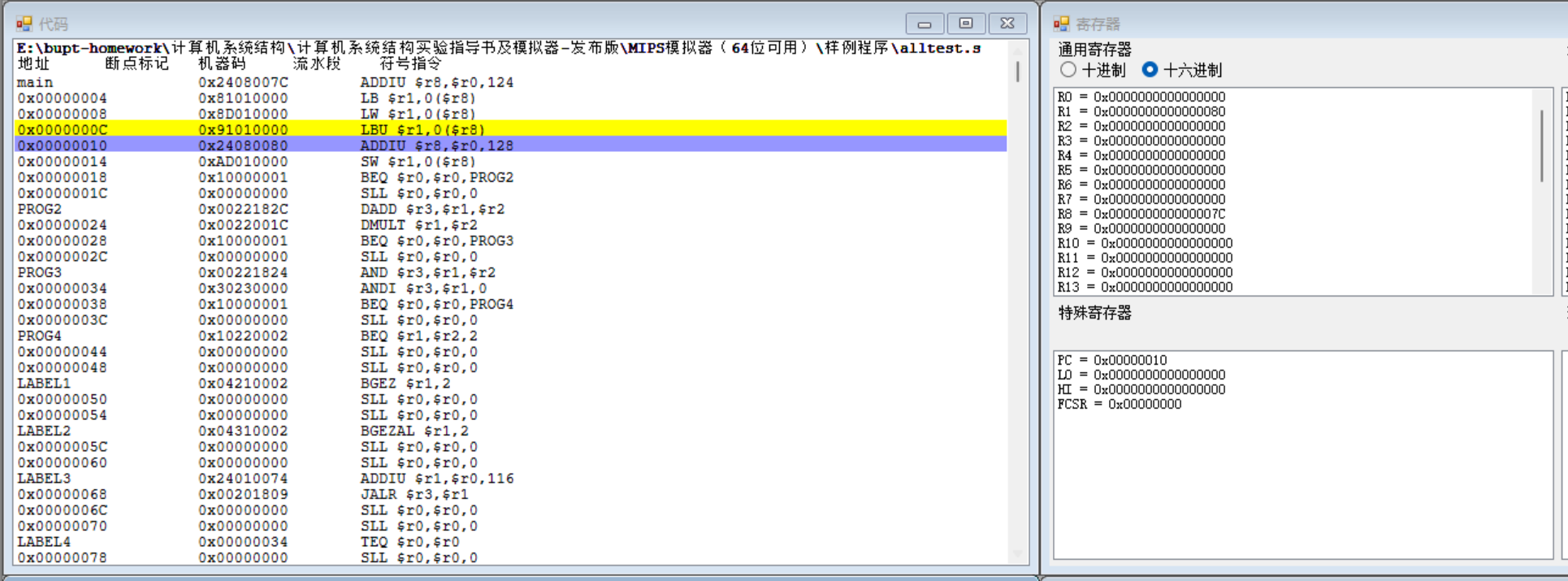
下一条指令的地址为[PC]=0x0000000C,是一条无符号载入字节指令。



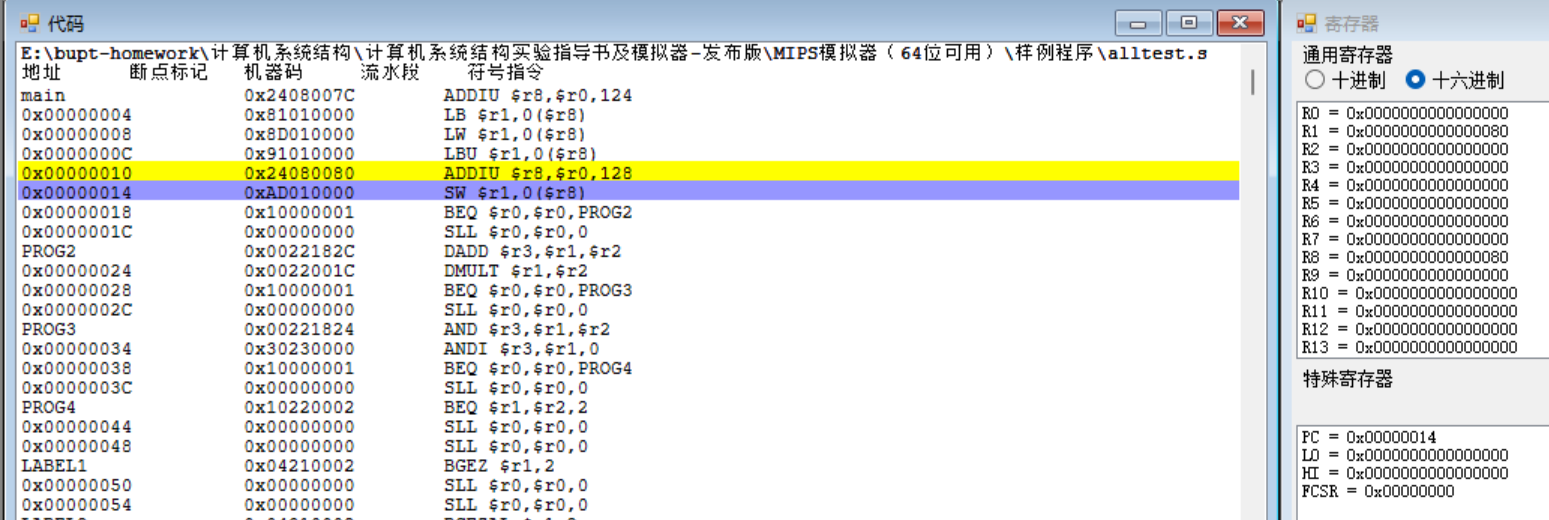


LBU表示是字节指令，无符号指令。

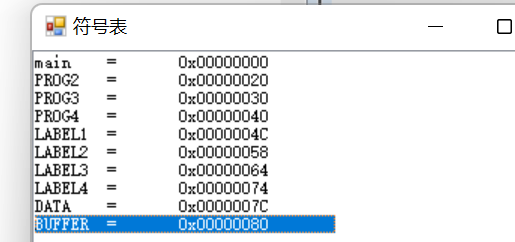
再次执行一条指令，R1还是原来的值。PC += 0x00000004 = 0x00000010



当[PC]=0x00000014时，该地址的指令是一条保存字的指令。



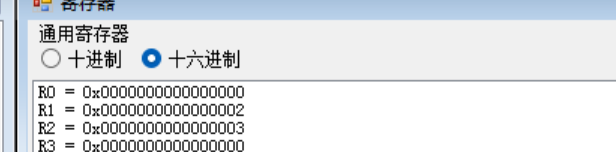
查看内存 -> 符号表内容，如下，BUFFER为0x00000080



接下来执行算数运算类指令，步骤如下：

1. 双击寄存器窗口中的R1，将其值修改为2.

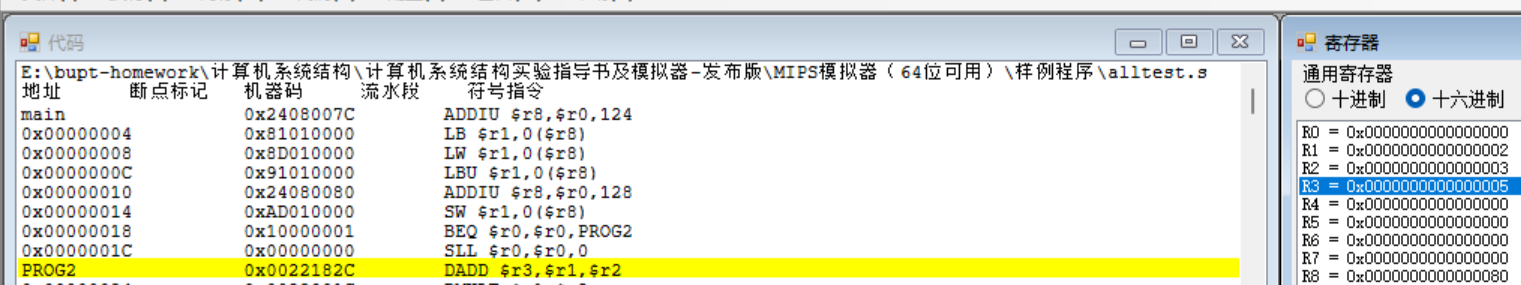
2. 双击寄存器窗口中的R2，将其值修改为3.



3. 单步执行1条指令。

4. 下一条指令地址为 0x00000020，是一条加法指令

5. 单步执行1条指令



6. 可以发现R3的值为0x0000000000000005，为R1和R2的值相加。

7. 下一条的指令的地址为[PC]=0x00000024，是一条乘法指令。

8. 单步执行1条指令



9. 查看L0、HI的值，

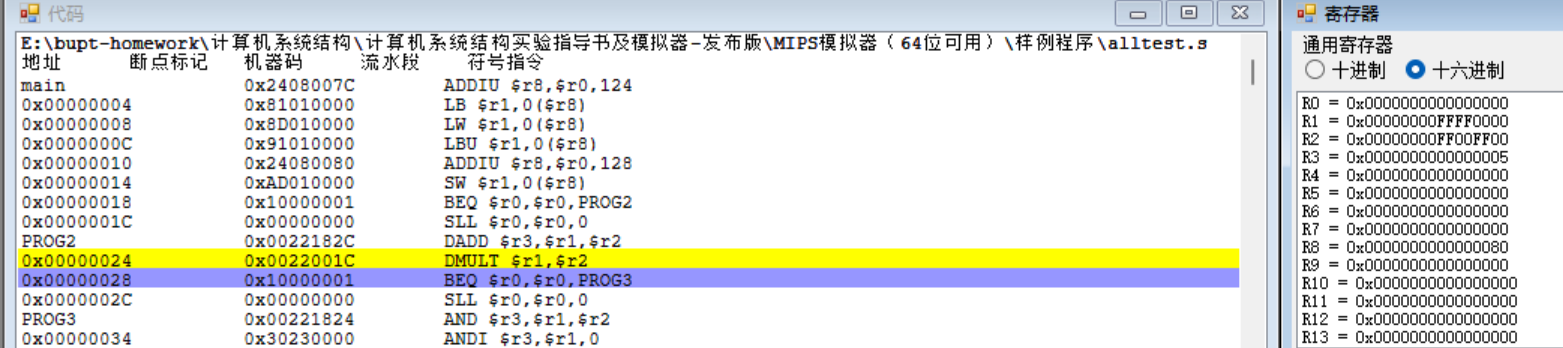
[L0]=0x0000000000000006

[HI]=0x0000000000000000

接下来执行逻辑运算类指令，步骤如下：

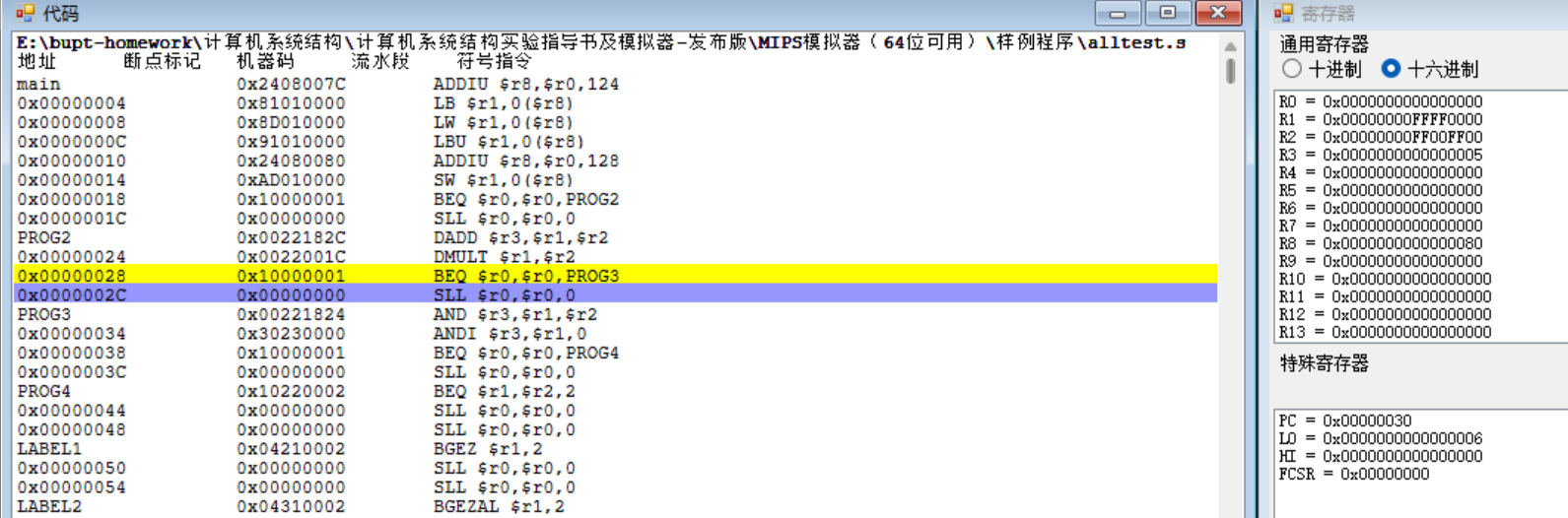
1. 双击“寄存器”窗口中的R1，将其值修改为0xFFFF0000。

2. 双击“寄存器”窗口中的R2，将其值修改为0xFF00FF00。



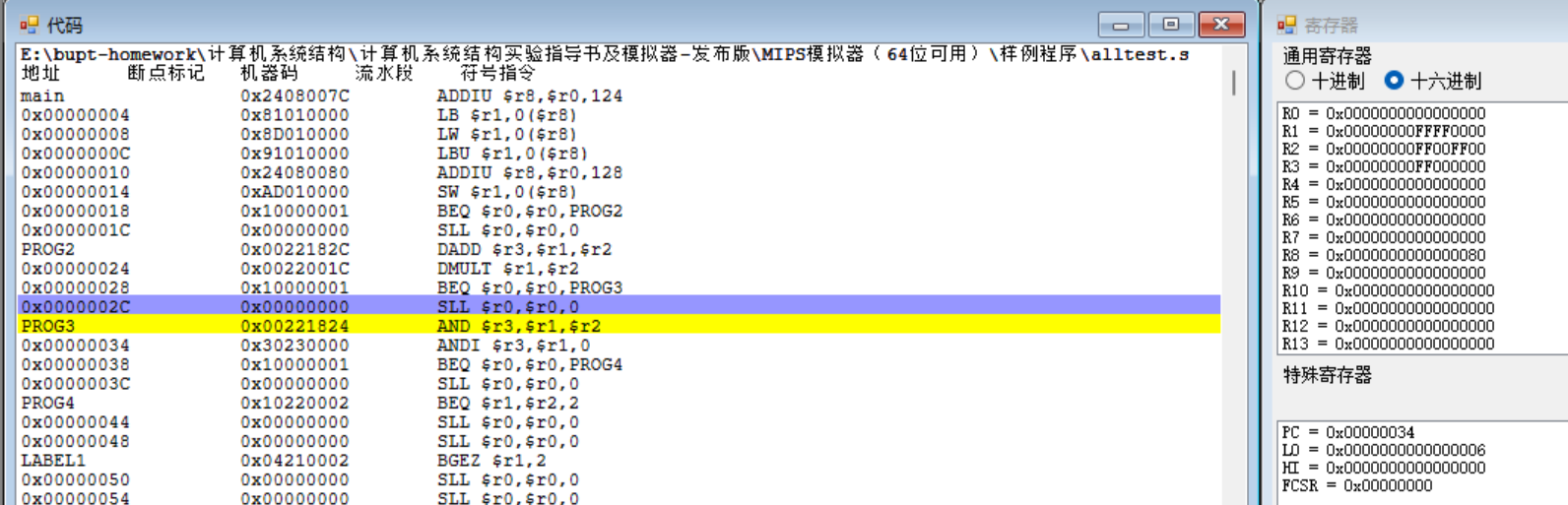
3. 单步执行

4. 下一条指令地址为0x00000030，是一条逻辑与运算指令，第二个操作数寻址方式是寄存器直接寻址。



5. 单步执行

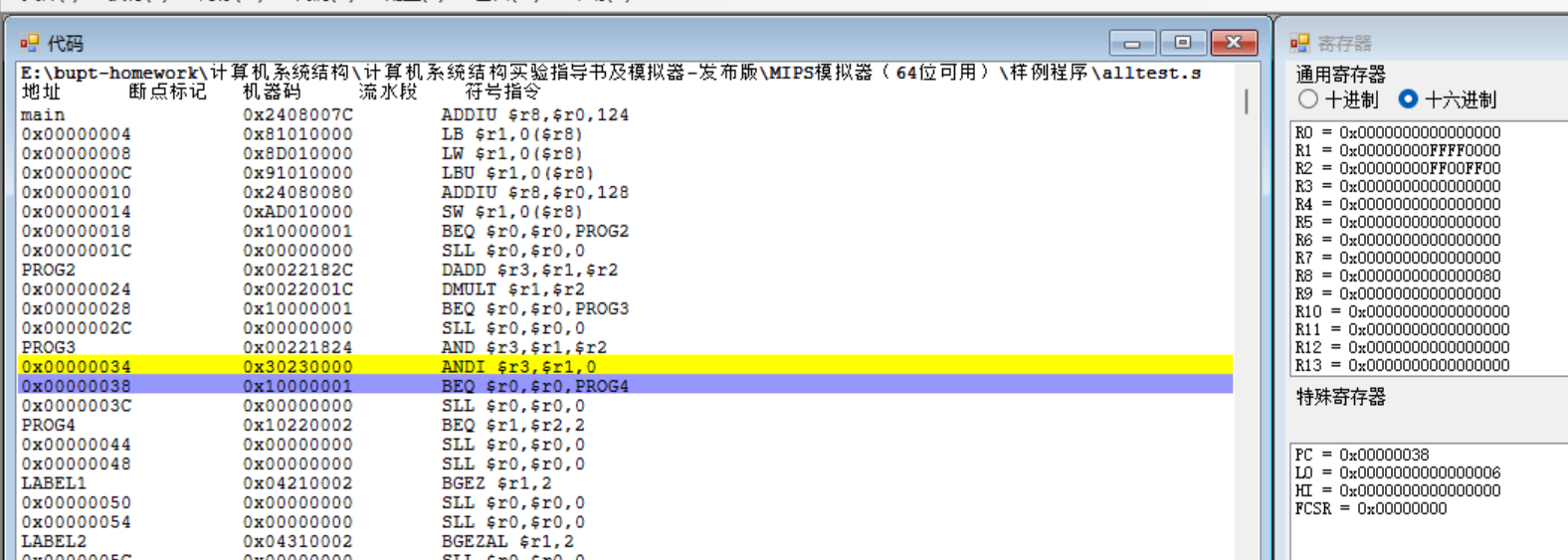
6. 查看R3的值，R3为0x00000000FF000000



7. 下一条指令地址为0x00000034是一条逻辑与运算指令，第二个操作数寻址方式是寄存器直接寻址。

8. 单步执行

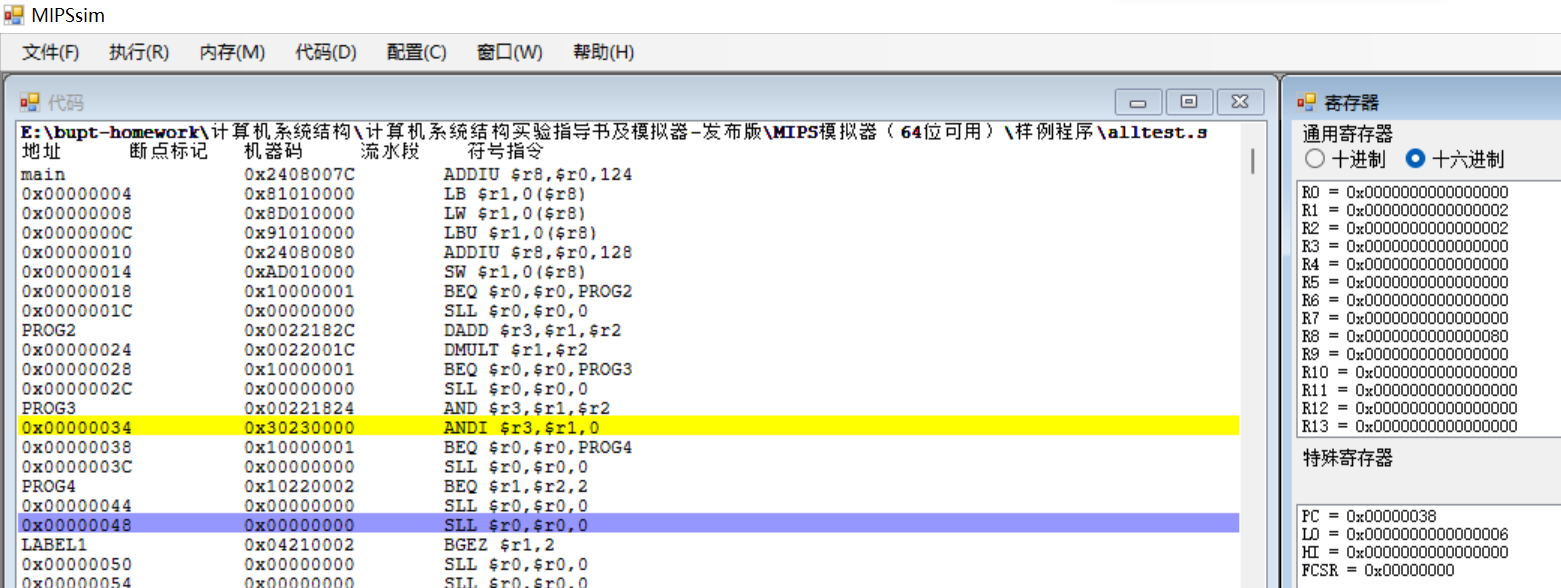
9. 查看R3的值，为0x00000000FF00FF00



接下来执行控制转移类指令。步骤如下：

1. 双击“寄存器”窗口中的R1，将其值修改为2。

2. 双击“寄存器”窗口中的R2，将其值修改为2。



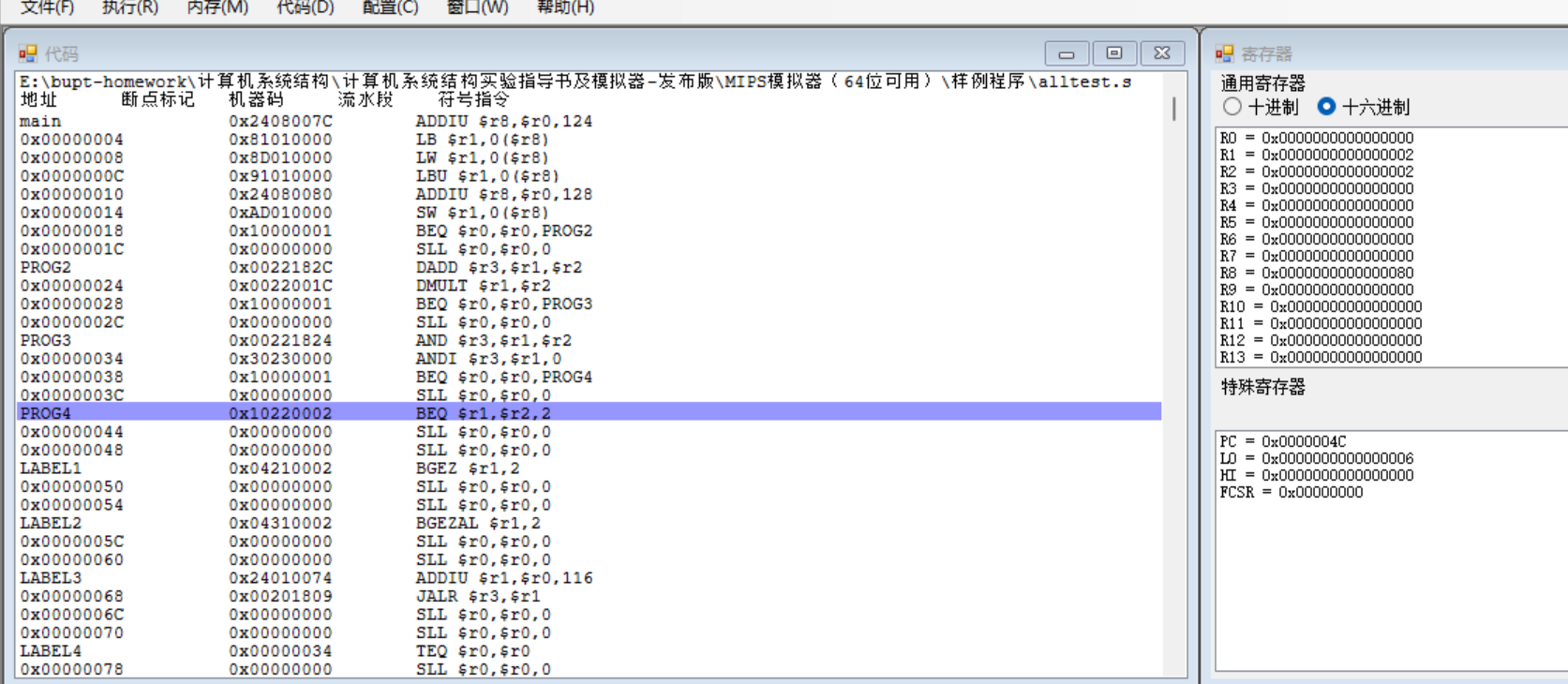
3. 单步执行1条指令。

4. 下一条指令地址为0x0x00000040，是一条BEQ指令，其测试条件是r1==r2， 目标地址为0x0000004C



5. 单步执行1条指令。

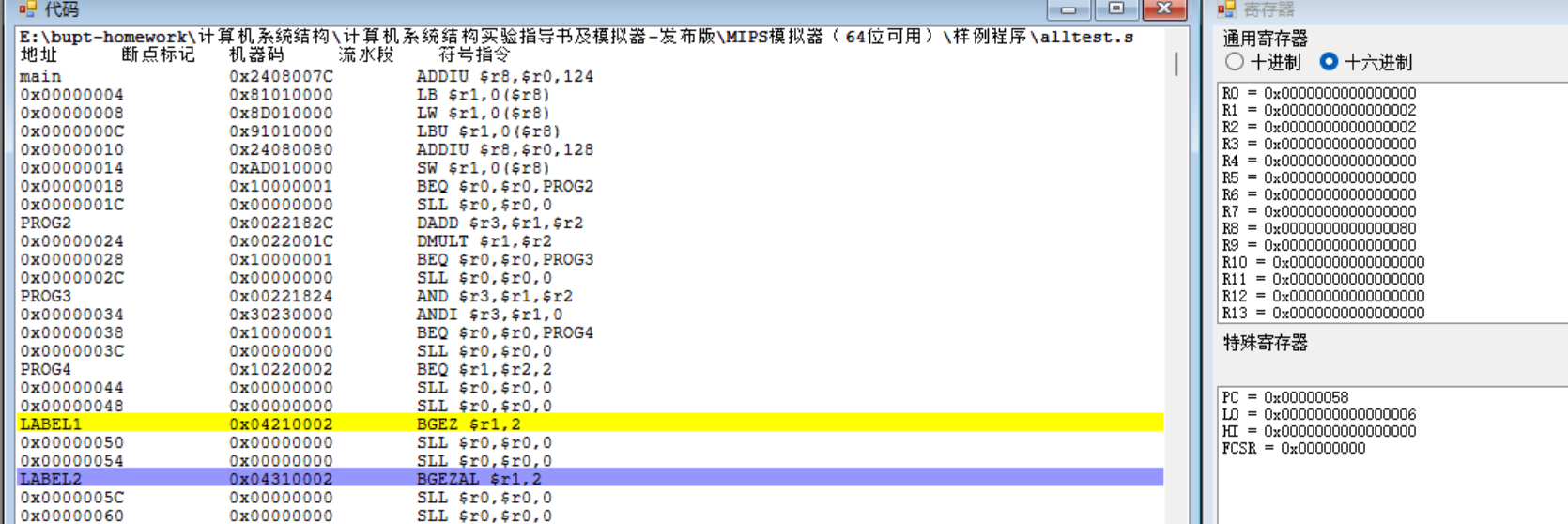
6. 查看PC的值，[PC]=0x0000004C，表明分支成功。因为刚刚把R1和R2都设置为了2，所以测试成功。



7. 下一条指令是一条 BGEZ 指令，其测试条件是r1>=0，目标地址为 0x00000058

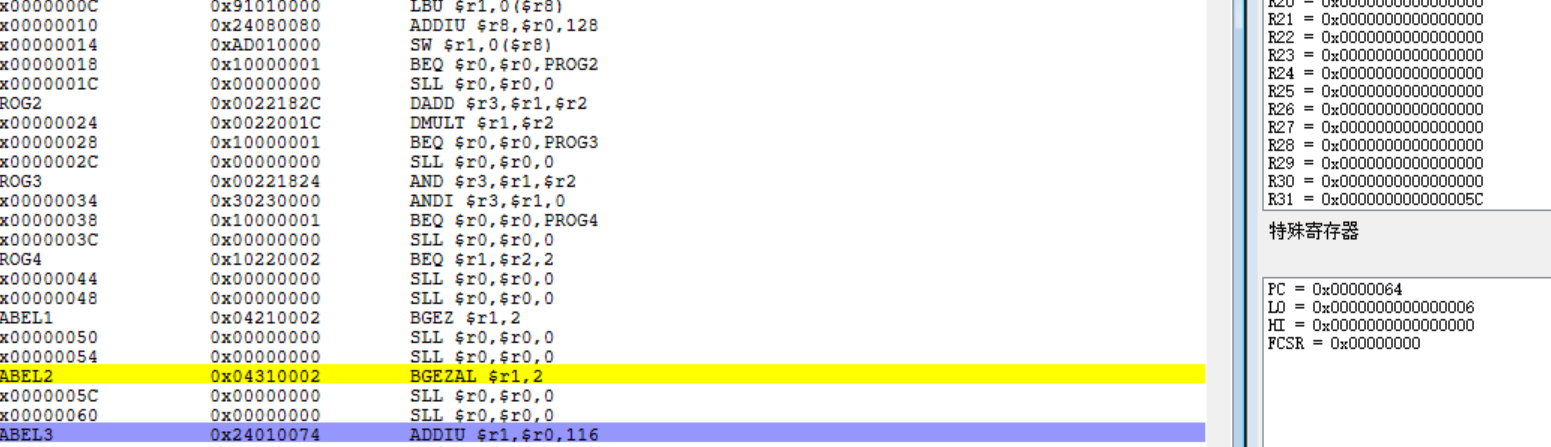
8. 单步执行1条指令。

9. 查看PC的值，[PC]=0x00000058，表明分支成功。



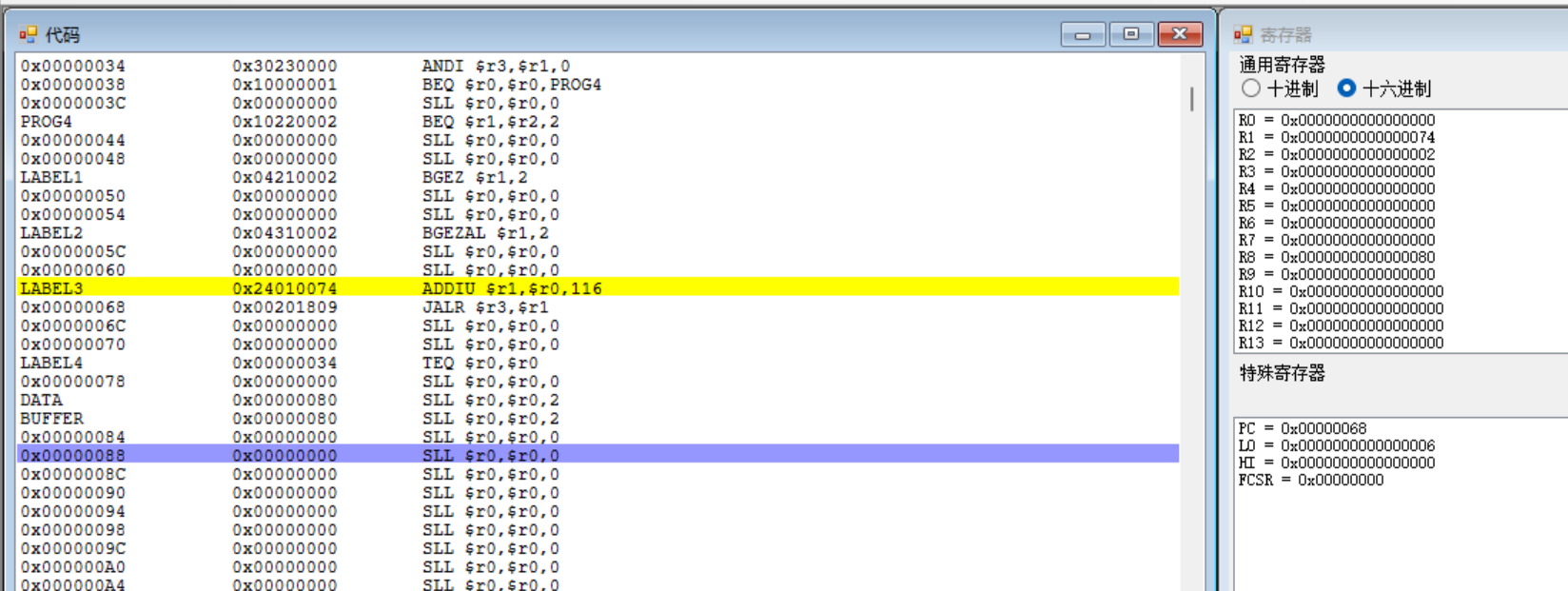
10. 下一条指令是一条 BGEZAL 指令，其测试条件是r1>=0，目标地址为 0x00000064

11. 单步执行1条指令。



12. 查看PC的值[PC]=0x00000064，表明分支成功；查看R31 的值，[R31]=0x000000000000005C

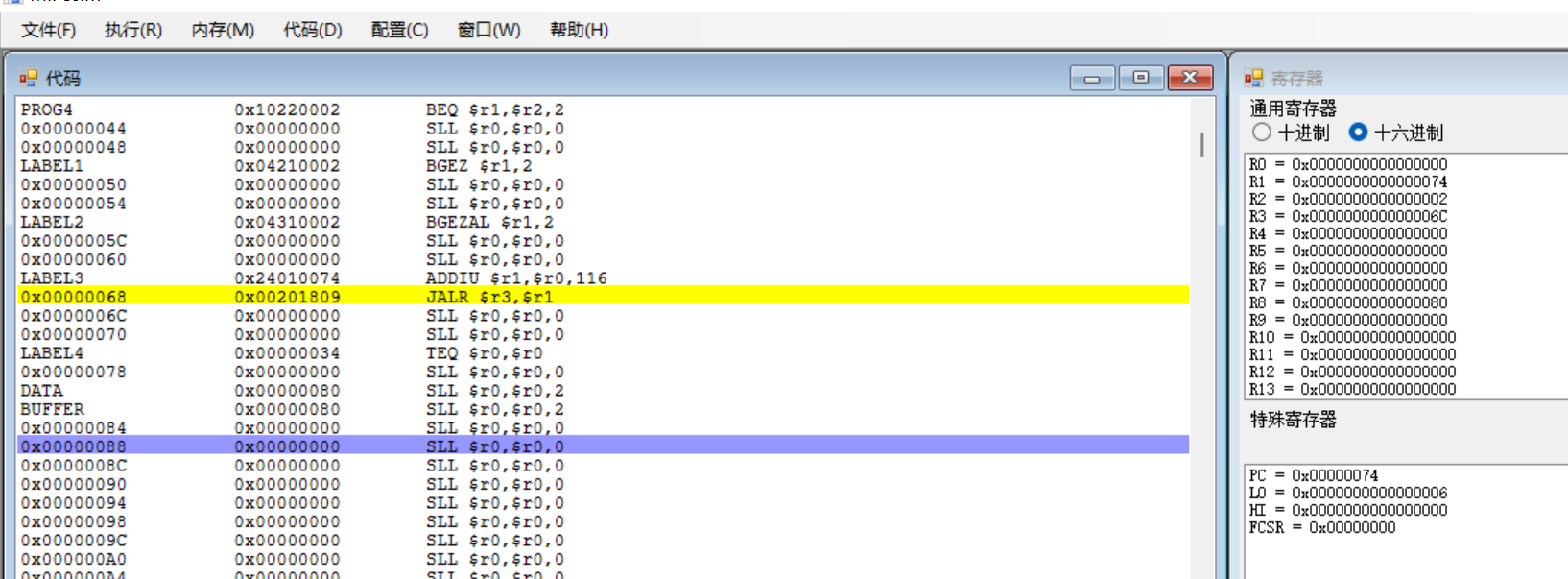
13. 单步执行1条指令。



14. 查看R1的值，[R1]=0x0000000000000074

15. 下一条指令地址为0x00000068，是一条 JALR 指令，保存目标地址的寄存器 为R1，保存返回地址的目标寄存器为R3。

16. 单步执行1条指令。



17. 查看PC和R3的值，[PC]=0x00000074，[R3]=0x000000000000006C。

## 5 实验总结

实验中的指令及作用如下：

|  |  |
| --- | --- |
| ADDIU $R8, $R0, 124 | R8=R0+124 |
| LB $R1, 0($R8) | 从[R8+0]位置读取字节到R1 |
| LW $R1, 0($R8) | 从[R8+0]位置读取字到R1 |
| LBU $R1, 0($R8) | 从[R8+0]位置读取无符号字节到R1 |
| SW $R1, 0($R8) | 将R0数据存到[R8+0]位置处 |
| BEQ $R0, $R0, PROG2 | 条件转移指令（到PROG2处） |
| DADD $R3, $R1, $R2 | R3=R1+R2 |
| DMULT $R1, $R2 | 两定点寄存器相乘 |
| AND $R3, $R1, $R2 | R3=R1与R2 |
| ANDI $R3, $R1, 0 | R3=R1与0 |
| BGEZ $R1, 2 | 条件转移指令（偏移量2） |
| JALR $R3, $R1 | 使用寄存器的跳转指令 |