**计原大实验报告**

组号302

组员：肖迪，杨煜，张艺庆

**系统整体测试**

经过测试，系统可以正常的使用，可以在我们做出来的CPU上运行监控程序，监控程序A，R，D，U，G各项命令运行正常。 在单步运行的时，通过VGA接口，显示屏幕上可以显示当前刚刚从指令寄存器取出的指令汇编代码（经过解码的），ps2小键盘运行也是正常的，可以通过小键盘的不同的按键选择不同的CPU运行的频率。系统整体通过了助教的检查验收，我们也拍摄了视频录制了系统正常运行的状态。

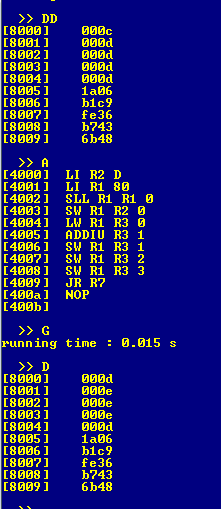
**系统运行速度测试**

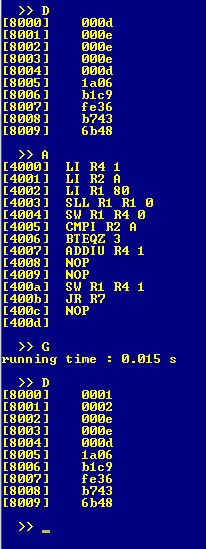
我们小组的CPU运行主频是12.5MHz，在这个频率下，运行助教的提供的五段测试代码，运行的时间如下表：

|  |  |
| --- | --- |
| 测试代码 | 运行时间（s） |
| 1 | 16．0 |
| 2 | 22．0 |
| 3 | 12．0 |
| 4 | 18．0 |
| 5 | 12．0 |

**系统编程测试**

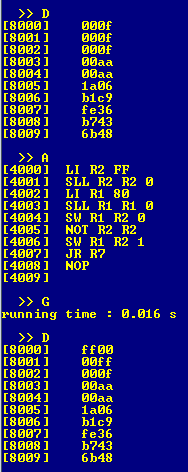
能够成功的运行监控程序，这就说明系统基本正确，但还有一些可能出现的冲突情况我们没有测试，另外还有监控程序中没有的扩展指令我们没有测试。

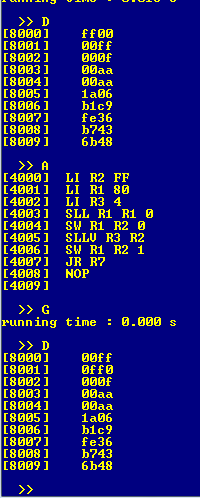
左图是**测试“load后使用”**这个数据冲突的，这个数据冲突在流水线中必须要加nop作为气泡，[4004]的load指令的目标地址是R3，然后下一条指令就要读R3，这就会产生一个冲突，从[4006]开始三条指令就是将R3的值写入[8001]开始的三个地址，可见，程序正确处理了这个情况，写入的值是加1之后的新值。

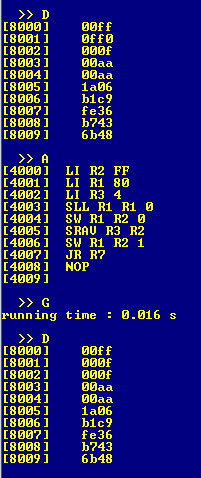
左图是**测试转发单元**，和**延迟槽**的代码：

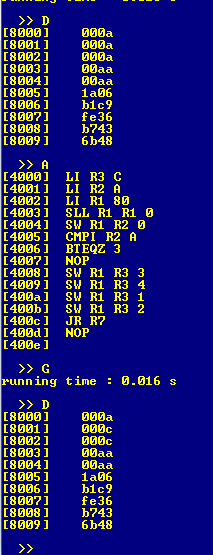
若前一条指令的目标寄存器是后一条指令的源寄存器，就会产生数据冲突，必须使用转发，在[4002]和之后一条指令之间就存在这样的关系。

延迟槽是分支和跳转指令后面的一条指令，无论分支是否执行，这条指令都要被执行，[4006]这条分支指令之后的一条指令就是延迟槽，虽然这里分支指令是被执行了，但是延迟槽中的指令照样要被执行，R4的值要被加1，所以我们看到[8001]和[8000]内存地址的结果就是不一样的，[8001]的结果被加了1，说明延迟槽是正常工作的。

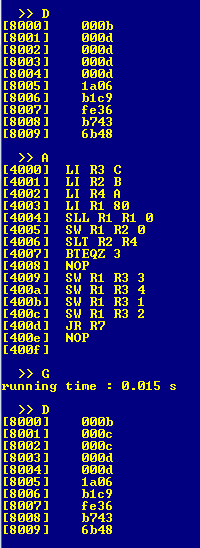
左图是测试我们302小组的第一条**扩展指令NOT**的，我们小组第一条扩展指令是NOT，测试代码中，将寄存器R2的值设置为0x00FF，然后取非，可以看到，结果变成了0xFF00，我分别将结果储存在了[8000]开头的内存中，这将是之后测试所将采用的模式。

左图是第二条**扩展指令SLLV**, 测试代码中，将寄存器R2的值设置为0x00FF，然后左移4位，结果变为0x0FF0，我分别将结果储存在了[8000]开头的内存中。

左图是测试**第三条扩展指令SRAV**的，和上一条类似，将寄存器R2的值设置为0x00FF，然后右移4位，结果变为0x000F，我分别将结果储存在了[8000]开头的内存中。



左图是测试**扩展指令CMPI的**，本条指令是比较一个寄存器和另外一个立即数的值，然后根据比较结果改变标志寄存器的T。本条指令应该结合分支指令使用，由于R2的值是0xA，而立即数的值也是0xA，所以CMPI运行之后的结果是将标志寄存器置为0，下一条[4006]的跳转指令就会产生跳转，跳转到的位置是[400a]，跳过了[4008] 和[4009]的两条将内存修改成c的指令，结果只是将[8001]和[8002]两个内存地址修改成了c，而不是不发生跳转的4个。



类似的，左图是测试**扩展指令SLT**的，这条指令也是一条根据不同的条件而改变标志寄存器的值。在[4006]的SLT指令比较了两个寄存器R2和R4的值，然后置标志寄存器为0，下一条[4007]的跳转指令就会产生跳转，跳转到的位置是[400b]，跳过了[4009] 和[400a]的两条将内存修改成c的指令，结果只是将[8001]和[8002]两个内存地址修改成了c，而不是不发生跳转的4个。