

学号：202000130143	姓名： 郑凯饶	班级： 计科 1 班
实验题目：实验二 用 WinDLX 模拟器执行程序		
实验学时：2	实验日期： 4. 19	
实验目的：  通过本实验，熟练掌握 WinDLX 模拟器的操作和使用，清楚 WinDLX 五段流水线在执行具体程序时的流水情况，熟悉 DLX 指令集结构及其特点。		
硬件环境：  Dell Latitude 5411 Intel (R) Core (TM) i5-10400H CPU @ 2. 60GHz (8GPUs) , ~2. 6GHz		
软件环境： VMware Workstation 16 Player Windows 7		
实验步骤与内容：  1. 阅读汇编代码，程序用辗转相减法实现求取两个数的 gcm（最大公约数）：		
<pre>.data  ;*** Prompts for input Prompt1:  .asciiz    "First Number:" Prompt2:  .asciiz    "Second Number: "  ;*** Data for printf-Trap PrintfFormat:  .asciiz    "gcM=%d\n\n"                 .align    2 PrintfPar:     .word      PrintfFormat PrintfValue:   .space     4  .text .global main main: ;*** Read two positive integer numbers into R1 and R2 addi    r1,r0,Prompt1    ; 将该字符串的地址存储至 r1 中，后面 InputUnsigned 会取地址并输出 jal     InputUnsigned    ;read uns.-integer into R1 add     r2,r1,r0          ;R2 &lt;- R1 addi    r1,r0,Prompt2</pre>		

```

        jal    InputUnsigned    ;read uns.-integer into R1

Loop:                                ;*** Compare R1 and R2
        seq    r3,r1,r2          ;R1 == R2 ?
        bnez   r3,Result
        sgt    r3,r1,r2          ;R1 > R2 ?
        bnez   r3,r1Greater
        ; 辗转相减法求最大公约数
r2Greater: ;*** subtract r1 from r2
        sub    r2,r2,r1
        j      Loop

r1Greater: ;*** subtract r2 from r1
        sub    r1,r1,r2
        j      Loop

Result:    ;*** Write the result (R1)
        sw     PrintfValue,r1
        addi   r14,r0,PrintfPar
        trap   5

        ;*** end
        trap   0

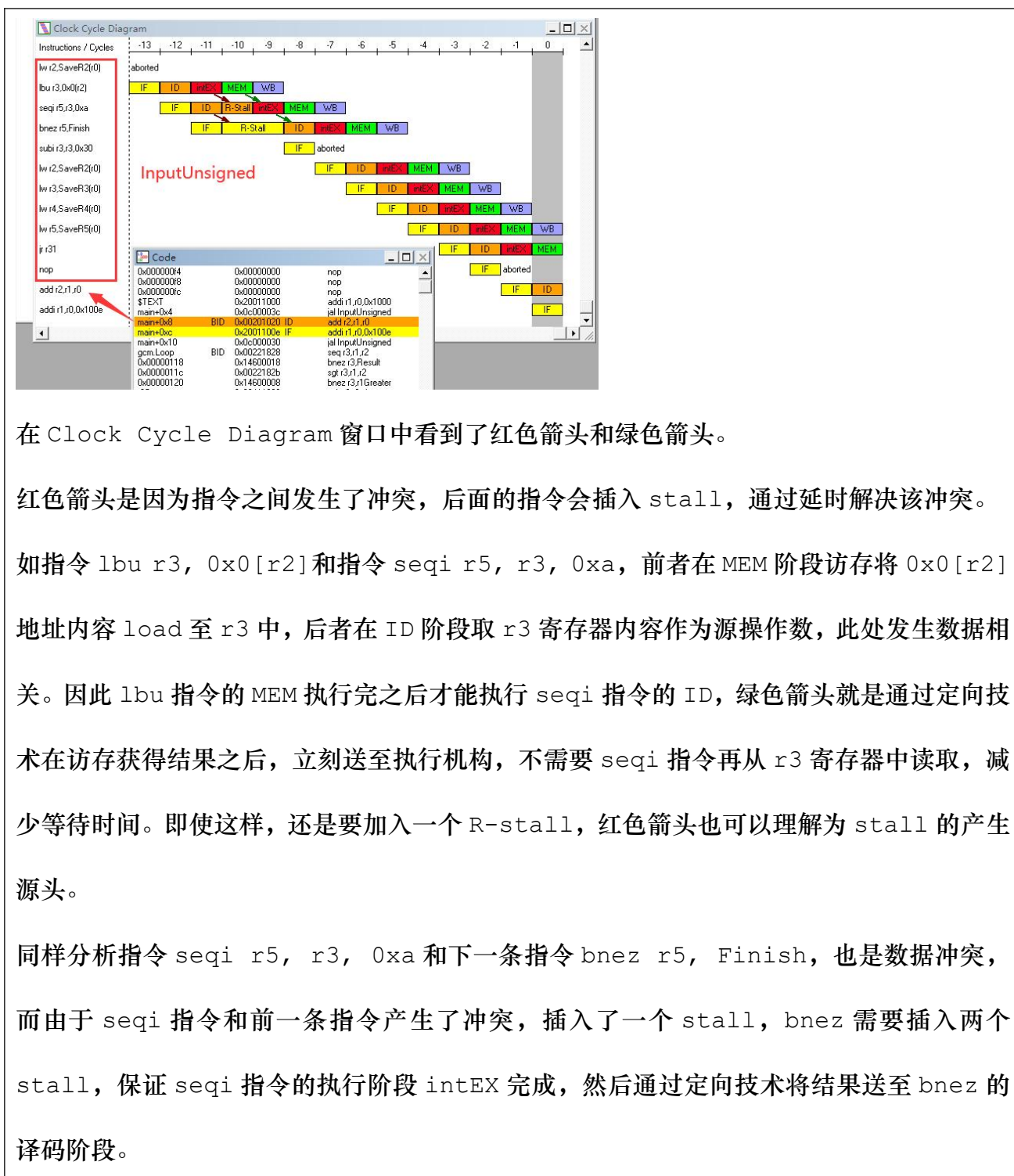
```

## 2. 装载程序并设置断点

Code		
main+0x4	0x0c00003c	jal InputUnsigned
main+0x8	BID 0x00201020	add r2,r1,r0
main+0xc	0x2001100e	addi r1,r0,0x100e
main+0x10	0x0c000030	jal InputUnsigned
gcm.Loop	BID 0x00221828	seq r3,r1,r2
0x00000118	0x14600018	bnez r3,Result
0x0000011c	0x0022182b	sgt r3,r1,r2
0x00000120	0x14600008	bnez r3,r1Greater
r2Greater	0x00411022	sub r2,r2,r1
0x00000128	0x0bffffe8	j gcm.Loop
r1Greater	0x00220822	sub r1,r1,r2
0x00000130	0x0bffffe0	j gcm.Loop
Result	0xac01102c	sw PrintfValue(r0),r1
Result+0x4	0x200e1028	addi r14,r0,0x1028
Result+0x8	0x44000005	trap 0x5
Result+0xc	BID 0x44000000	trap 0x0

## 3. 运行程序

F5 运行程序至第一个断点处 main+0x8 处,可以看到 InputUnsigned 的汇编代码被展开。



在 Clock Cycle Diagram 窗口中看到了红色箭头和绿色箭头。

红色箭头是因为指令之间发生了冲突，后面的指令会插入 stall，通过延时解决该冲突。

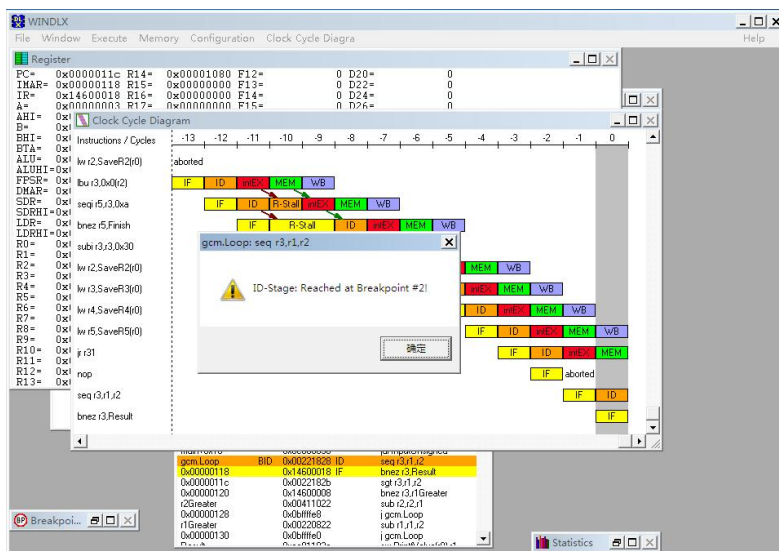
如指令 `lbu r3, 0x0[r2]` 和指令 `seqi r5, r3, 0xa`，前者在 MEM 阶段访存将 `0x0[r2]` 地址内容 load 至 `r3` 中，后者在 ID 阶段取 `r3` 寄存器内容作为源操作数，此处发生数据相关。因此 `lbu` 指令的 MEM 执行完之后才能执行 `seqi` 指令的 ID，绿色箭头就是通过定向技术在访存获得结果之后，立刻送至执行机构，不需要 `seqi` 指令再从 `r3` 寄存器中读取，减少等待时间。即使这样，还是要加入一个 R-stall，红色箭头也可以理解为 stall 的产生源头。

同样分析指令 `seqi r5, r3, 0xa` 和下一条指令 `bnez r5, Finish`，也是数据冲突，而由于 `seqi` 指令和前一条指令产生了冲突，插入了一个 stall，`bnez` 需要插入两个 stall，保证 `seqi` 指令的执行阶段 `intEX` 完成，然后通过定向技术将结果送至 `bnez` 的译码阶段。

Register									
PC=	0x00000110	R14=	0x00001080	F12=	0	D20=	0		
IMAR=	0x0000010c	R15=	0x00000000	F13=	0	D22=	0		
IR=	0x2001100e	R16=	0x00000000	F14=	0	D24=	0		
A=	0x00000006	R17=	0x00000000	F15=	0	D26=	0		
AHI=	0x00000000	R18=	0x00000000	F16=	0	D28=	0		
B=	0x00000000	R19=	0x00000000	F17=	0	D30=	0		
BHI=	0x00000000	R20=	0x00000000	F18=	0				
BT=	0x00000000	R21=	0x00000000	F19=	0				
ALU=	0x00000000	R22=	0x00000000	F20=	0				
ALUHI=	0x00000000	R23=	0x00000000	F21=	0				
FFSR=	0x00000000	R24=	0x00000000	F22=	0				
DMAR=	0x00000000	R25=	0x00000000	F23=	0				
SDR=	0x00000000	R26=	0x00000000	F24=	0				
SDRHI=	0x00000000	R27=	0x00000000	F25=	0				
LDR=	0x00000000	R28=	0x00000000	F26=	0				
LDRHI=	0x00000000	R29=	0x00000000	F27=	0				
R0=	0x00000000	R30=	0x00000000	F28=	0				
R1=	0x00000006	R31=	0x00000108	F29=	0				
R2=	0x00000000	F0=	0	F30=	0				
R3=	0x00000000	F1=	0	F31=	0				
R4=	0x00000000	F2=	0	D0=	0				
R5=	0x00000000	F3=	0	D2=	0				
R6=	0x00000000	F4=	0	D4=	0				
R7=	0x00000000	F5=	0	D6=	0				
R8=	0x00000000	F6=	0	D8=	0				
R9=	0x00000000	F7=	0	D10=	0				
R10=	0x00000000	F8=	0	D12=	0				
R11=	0x00000000	F9=	0	D14=	0				
R12=	0x00000000	F10=	0	D16=	0				
R13=	0x00000000	F11=	0	D18=	0				

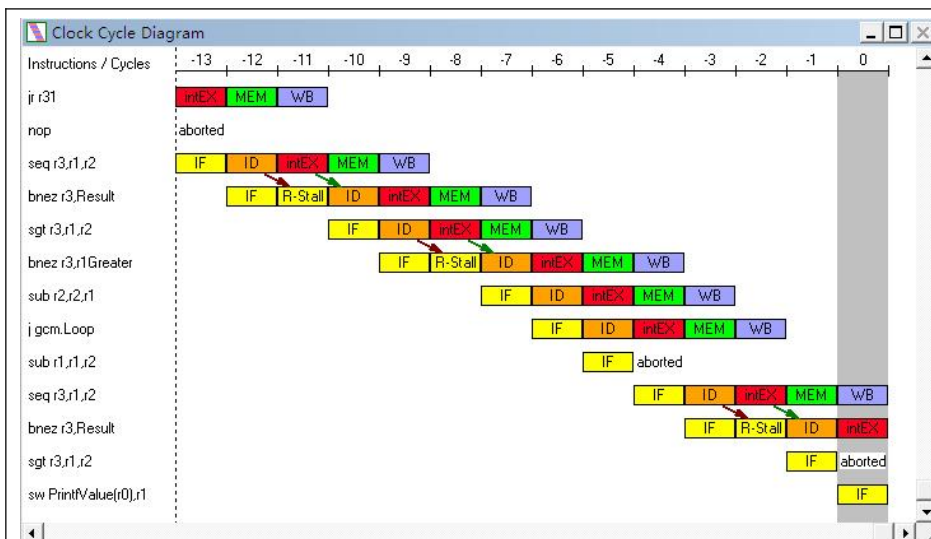
从 Register 窗口可以看到输入数被保存在 r1 寄存器中。

F5 执行到第二个断点处，此时输入完成。可以看到第一个输入被保存在了 R2，第二个输入数被保存在了 R1，准确来说调用 InputUnsigned 函数返回结果会保存在 R1 寄存器中。



Register									
PC=	0x0000011c	R14=	0x00001080	F12=	0	D20=	0		
IMAR=	0x00000118	R15=	0x00000000	F13=	0	D22=	0		
IR=	0x14600018	R16=	0x00000000	F14=	0	D24=	0		
A=	0x00000003	R17=	0x00000000	F15=	0	D26=	0		
AHI=	0x00000000	R18=	0x00000000	F16=	0	D28=	0		
B=	0x00000006	R19=	0x00000000	F17=	0	D30=	0		
BHI=	0x00000000	R20=	0x00000000	F18=	0				
BT=	0x00000000	R21=	0x00000000	F19=	0				
ALU=	0x00000000	R22=	0x00000000	F20=	0				
ALUHI=	0x00000000	R23=	0x00000000	F21=	0				
FFSR=	0x00000000	R24=	0x00000000	F22=	0				
DMAR=	0x00000000	R25=	0x00000000	F23=	0				
SDR=	0x00000000	R26=	0x00000000	F24=	0				
SDRHI=	0x00000000	R27=	0x00000000	F25=	0				
LDR=	0x00000000	R28=	0x00000000	F26=	0				
LDRHI=	0x00000000	R29=	0x00000000	F27=	0				
R0=	0x00000000	R30=	0x00000000	F28=	0				
R1=	0x00000003	R31=	0x00000114	F29=	0				
R2=	0x00000006	F0=	0	F30=	0				
R3=	0x00000000	F1=	0	F31=	0				
R4=	0x00000000	F2=	0	D0=	0				
R5=	0x00000000	F3=	0	D2=	0				
R6=	0x00000000	F4=	0	D4=	0				
R7=	0x00000000	F5=	0	D6=	0				
R8=	0x00000000	F6=	0	D8=	0				
R9=	0x00000000	F7=	0	D10=	0				
R10=	0x00000000	F8=	0	D12=	0				
R11=	0x00000000	F9=	0	D14=	0				
R12=	0x00000000	F10=	0	D16=	0				
R13=	0x00000000	F11=	0	D18=	0				

F5 继续执行，停止第三个断点，之后通过 F7 单步执行程序。可以看到程序的核心循环。



这里我们逐句分析：

Seq r3, r1, r2: 比较 r1,r2 寄存器值，如果相等，会跳转至 Result 输出结果。

这里 r1=3,r2=6 不相等，因此向下执行。

Sgt, r3, r1, r2: 比较 r1,r2 寄存器值，若 r1 大于 r2，则跳转 r1Greater 执行，这里不大于，因此向下执行。

Sub r2, r2, r1:  $r2 - r1 \rightarrow r2$ ，辗转相减法用大数减去小数，新得到的两个的 gcm 等于原来 r1,r2 中两个数的 gcm.

此时，r1=3,r=2.

I gcm Loop: 跳转至 Loop 标识符处，循环执行。

由于跳转后面一句被废弃。

Seq r3, r1, r2: 这时相等满足，跳转至 Result，后面一句被废弃。

Sw PrintValue[r0], r1: 将结果存储到指定位置，准备输出。

可以看到红绿箭头的组合重复出现了三次，这均是因为前一条指令要生成下一条条件跳转的判断条件，只能添加一个 stall 并在 IntEX 阶段后将结果定向传送至条件跳转指令的 ID 阶段。

最后得到程序的执行结果为 3.

### 结论分析与体会：

之前我没有系统学习过汇编，通过这个系列实验以及 chatgpt 的强力加持，我对汇编程序的结果有了进一步的认识。深入理解了汇编指令如何通过流水线的方式并发执行，以及这个过程会产生的各种相关性问题和对应的解决措施。