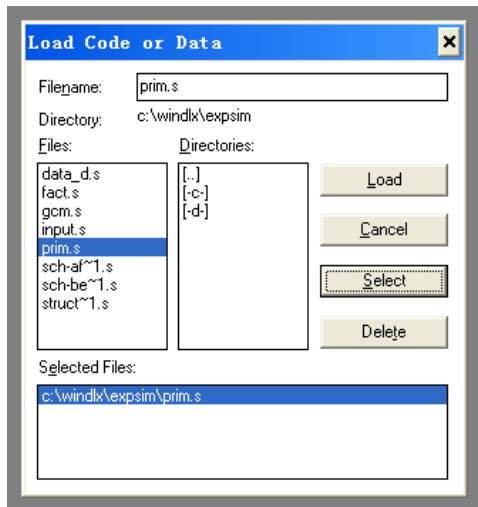


山东大学计算机科学与技术学院

计算机体系结构课程实验报告

学号：201800130031	姓名：来苑	班级：计科 1 班
实验题目：实验三 用 WinDLX 模拟器完成求素数程序		
实验学时：2 学时	实验日期：2021. 5. 21	
<p>实验目的：</p> <p>通过实验，熟练掌握 WINDLX 的操作方法，特别注意在单步执行 WinDLX 程序中，流水线中指令的节拍数。</p>		
<p>硬件环境：</p> <p>机房</p>		
<p>软件环境：</p> <p>WindowsXp</p>		
<p>实验步骤与内容：</p> <p>1. 用 WinDLX 模拟器执行求素数程序 prim.s</p> <p>1.1 初始化和配置环境，装入程序 prim.s</p> <div data-bbox="549 1086 1029 1590"></div> <p>1.2 阅读代码</p> <div data-bbox="247 1662 1342 1886"><pre>1 ; ***** WINDLX Exp.2: Generate prime number table ***** 2 ; ***** (c) 1991 Günther Raidl ***** 3 ; ***** Modified 1992 Maziar Khosravipour ***** 4 5 ;----- 6 ; Program begins at symbol main 7 ; generates a table with the first 'Count' prime numbers from 'Table' 8 ;----- 9</pre></div>		

```

10      .data
11
12      ;*** size of table
13      .global      Count
14      Count:      .word      10          ; 获得10个素数
15      .global      Table
16      Table:      .space     Count*4    ; 每个数4个字节
17      ;----- 获得10个素数, 每个素数占4个字节 -----
18
19      .text
20      .global main
21      main:
22      ;*** Initialization 初始化下标和当前值
23      addi        r1,r0,0      ; 数组下标
24      addi        r2,r0,2      ; 数, 从2开始
25
26      ;*** 判断(R2)能否被数组里的数整除
27      NextValue:  addi        r3,r0,0    ; 从table中的第1个数开始, 初始化循环值
28      Loop:      seq         r4,r1,r3    ; (r1)=(r3), 则r4=1, 已遍历数组
29                  bnez        r4,IsPrim  ; r4不为0, r4为素数, 跳转
30                  lw          r5,Table(r3) ; 取table中当前下标的数
31                  divu        r6,r2,r5    ; r6=r2/r5
32                  multu       r7,r6,r5    ; r7=r6*r5
33                  subu        r8,r2,r7    ; r8=r2-r7 计算余数
34                  beqz        r8,IsNoPrim ; 余数为0, 能被整除, r2不是素数
35                  addi        r3,r3,4
36                  j           Loop
37
38      IsPrim:     ;*** Write value into Table and increment index ;是素数, 写入table, 下标+“1”
39                  sw          Table(r1),r2 ; 数插入数组
40                  addi        r1,r1,4     ; 下标+“1”
41
42      ;*** 'Count' reached?
43      lw          r9,Count      ; (r9)=Count
44      srli        r10,r1,2      ; 右移两位, 相当于除以4 (r1加的1是以4个字节为单位的“1”)
45      sge         r11,r10,r9    ; r10>=r9, r11=1
46      bnez        r11,Finish
47
48      IsNoPrim:   ;*** Check next value ; 不是素数, 判断下一个数
49                  addi        r2,r2,1     ; increment R2
50                  j           NextValue
51
52      Finish:     ;*** end
53      trap        0

```

可以看到, 在 Loop 循环中, 有乘法指令和除法指令, 在后面的执行中, 要着重观察一下这两条指令的节拍数。

```

28      Loop:      seq         r4,r1,r3    ; (r1)=(r3), 则r4=1, 已遍历数组
29                  bnez        r4,IsPrim  ; r4不为0, r4为素数, 跳转
30                  lw          r5,Table(r3) ; 取table中当前下标的数
31                  divu        r6,r2,r5    ; r6=r2/r5
32                  multu       r7,r6,r5    ; r7=r6*r5
33                  subu        r8,r2,r7    ; r8=r2-r7 计算余数
34                  beqz        r8,IsNoPrim ; 余数为0, 能被整除, r2不是素数
35                  addi        r3,r3,4
36                  j           Loop
37

```

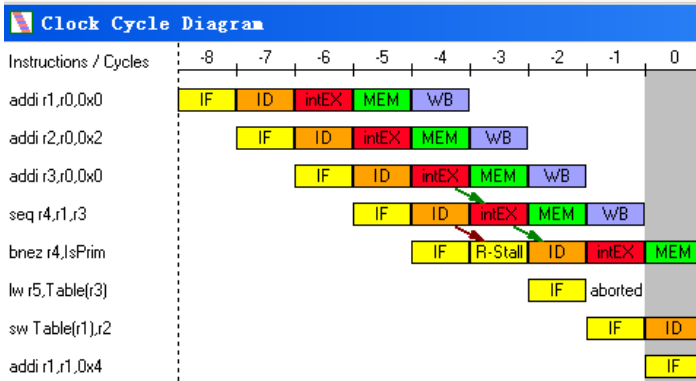
1.3 单步执行求取素数 2

为了方便跟踪, 首先在指令 `sw Table(r1),r2` 处设置断点, 此处表示找到素数, 并且将这个素数写入 table 中。

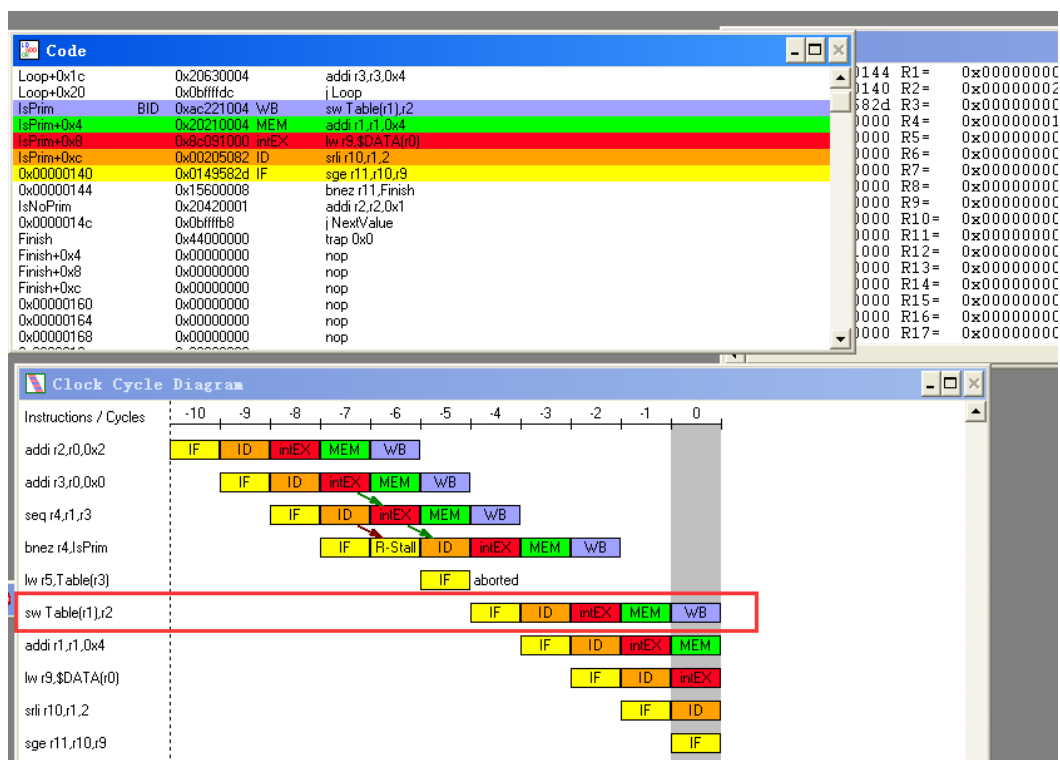
Loop+0x20	0x0bffffdc	j Loop
IsPrim	0xac221004	sw Table(r1),r2
IsPrim+0x4	0x20210004	addi r1,r1,0x4
IsPrim+0x8	0x8c091000	lw r9,\$DATA(r0)

然后单步执行程序, 直到到达断点处。

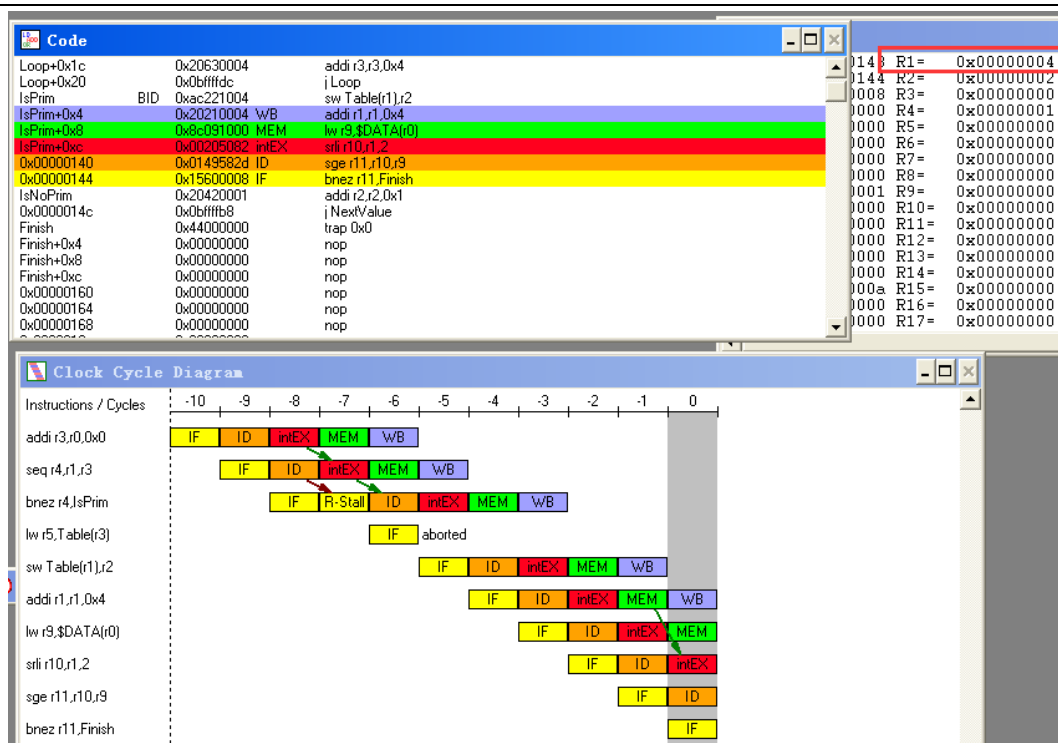
Code		
\$TEXT	0x20010000	addi r1,r0,0x0
main+0x4	0x20020002	addi r2,r0,0x2
NextValue	0x20030000	addi r3,r0,0x0
Loop	0x00232028	seq r4,r1,r3
Loop+0x4	0x1480001c	bnez r4,IsPrim
Loop+0x8	0x8c651004	lw r5,Table(r3)
Loop+0xc	0x0045301b	divu r6,r2,r5
Loop+0x10	0x00c53819	multu r7,r6,r5
Loop+0x14	0x00474023	subu r8,r2,r7
Loop+0x18	0x11000020	beqz r8,IsNoPrim
Loop+0x1c	0x20630004	addi r3,r3,0x4
Loop+0x20	0x0bffffdc	j Loop
IsPrim	BID 0xac221004	ID sw Table(r1),r2
IsPrim+0x4	0x20210004	IF addi r1,r1,0x4
IsPrim+0x8	0x8c091000	lw r9,\$DATA(r0)



继续单步执行，直到设置断点的指令执行完毕。此时，素数 2 已被写入 table 中。



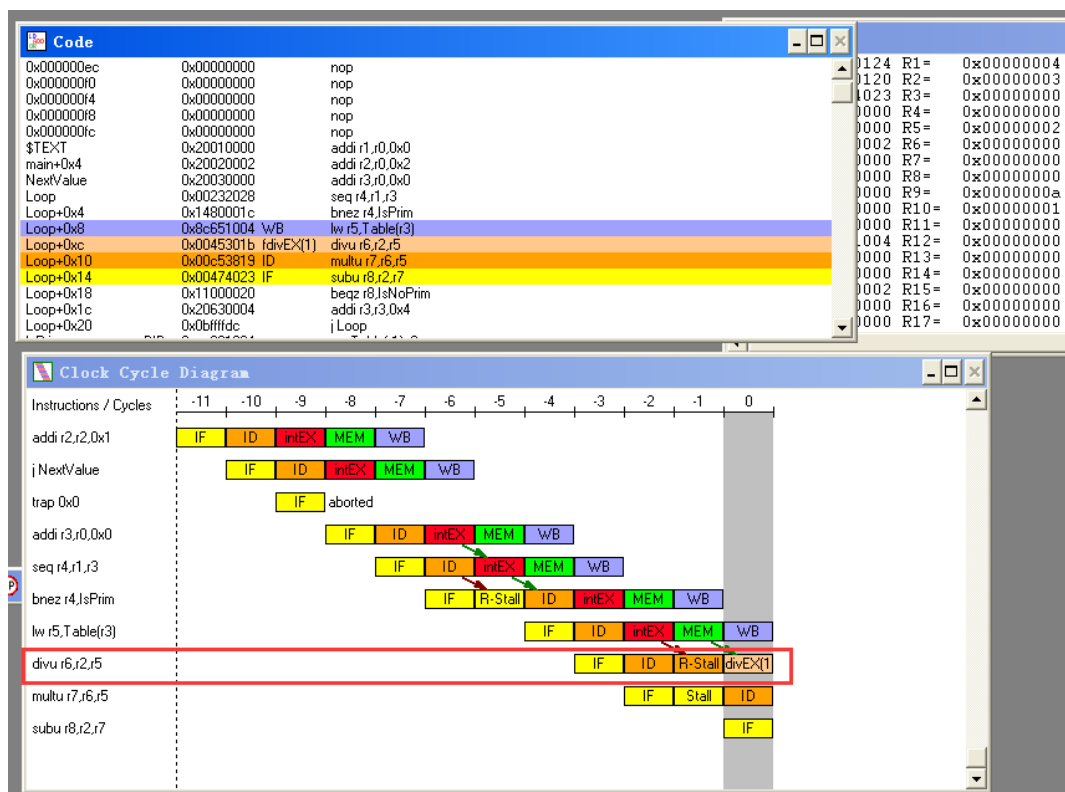
继续单步执行，此时 table 的下标+“1”，可以看到 r1 的值变成了 4 (addi r1,r1,4)。

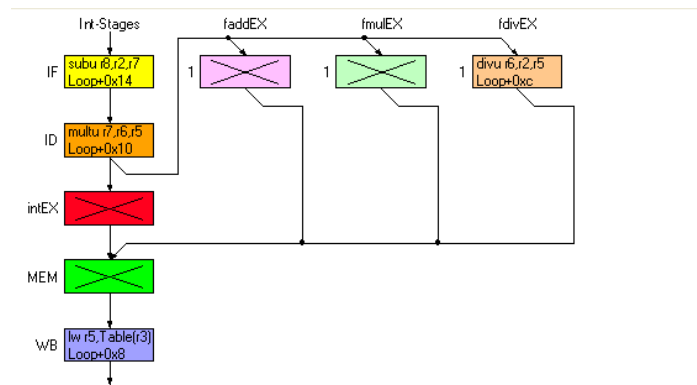


至此，素数 2 已经求取完毕，接下来观察素数 3 的求取过程。

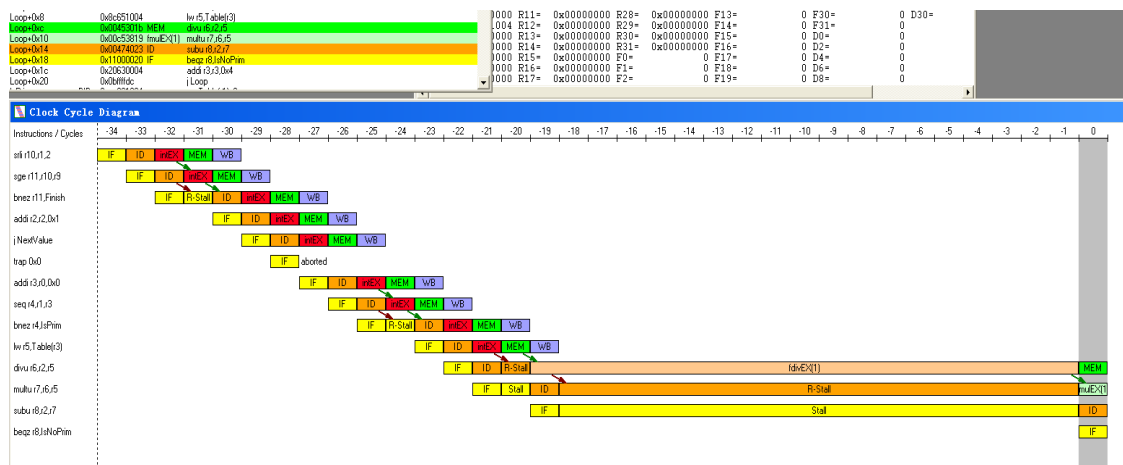
1.4 单步执行求取素数 3

继续执行，直到执行到除法指令。此时除法指令进入执行周期(EX)。

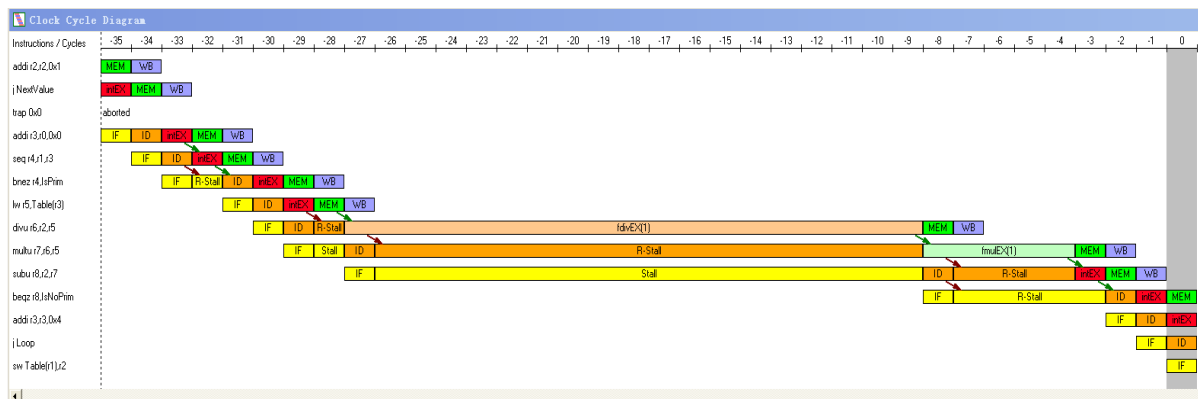




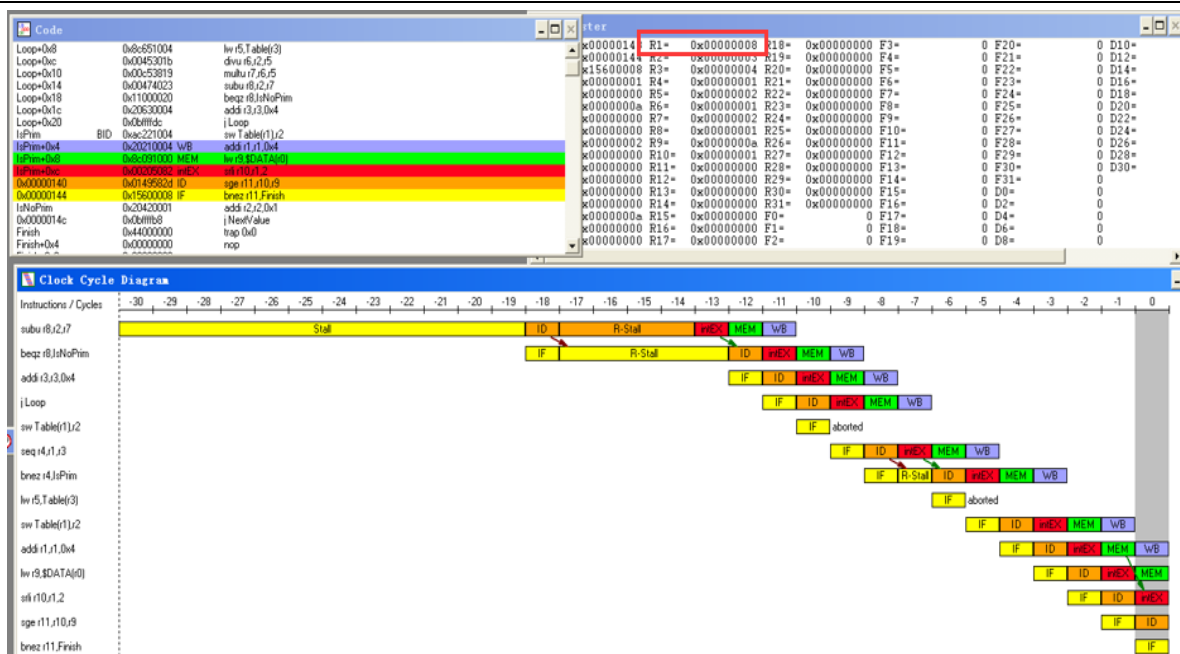
继续单步执行，可以发现，除法指令的执行周期(EX)的节拍数很大。



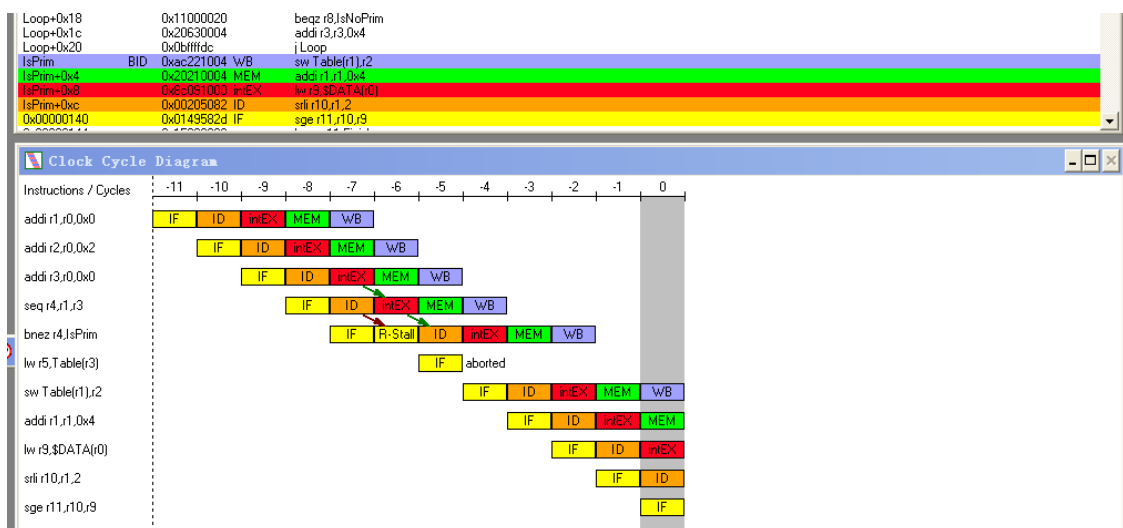
继续单步执行乘法指令，可以发现，虽然乘法指令的执行周期没有除法周期那么长，但也比一般指令，如读取指令，要长得多。



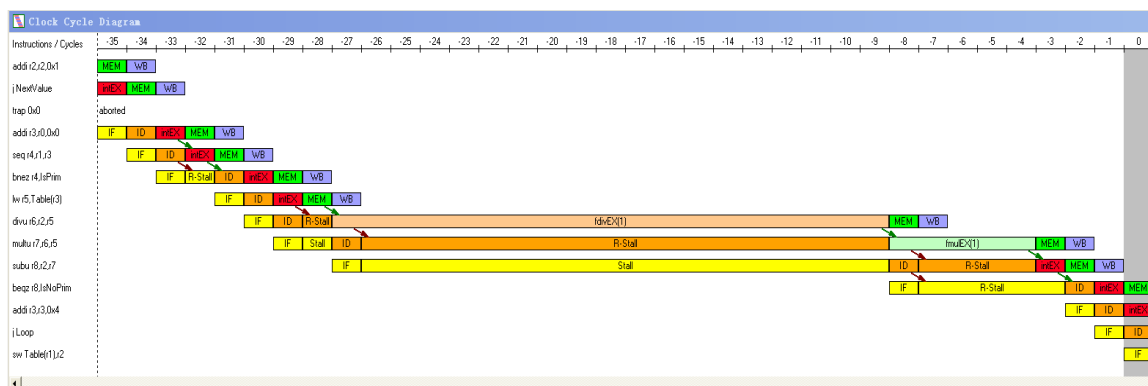
至此，素数 3 求取完毕。可以看到 R1 寄存器的内容变成了 8，即数组的下标实现了 +1，也就是说，现在已经进入求取下一个素数的阶段。



2. 观察求取素数 2 和素数 3 的指令节拍的差别 素数 2:

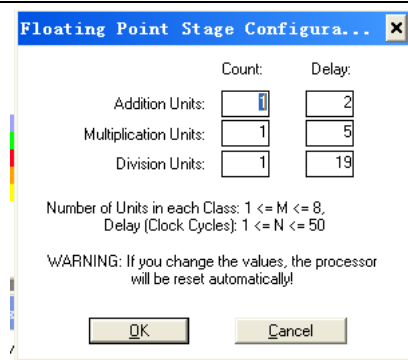


素数 3:



两者的区别在于求取素数 2 时，由于数组中没有元素，所以不需要执行乘除指令，而素数 3 的求取过程中，会有乘除指令的指令。所以，两者比较，可以很明显地认识超乘除指令指令节拍长的特点。

3. 观察 configuration/floating point slages 中的各指令执行拍数



通过这里，也可以看到，乘除指令的节拍多。

结论分析与体会：

在做这个实验前，虽然直到乘除指令的执行时间会比较长，但在试验时才深刻地感受到了乘除指令的节拍数真的很长。通过这次实验，我对指令流水和指令节拍有了更深刻的体会。