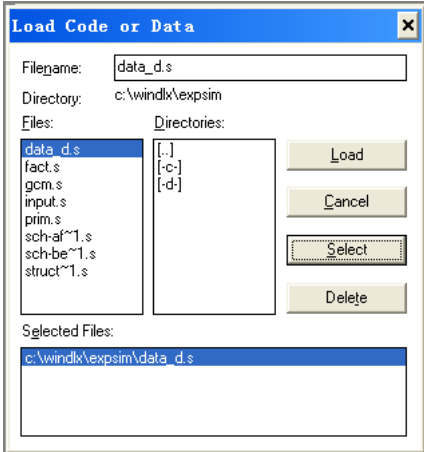
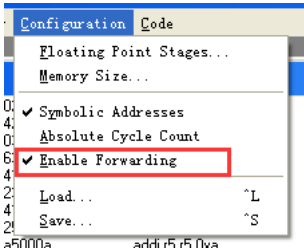
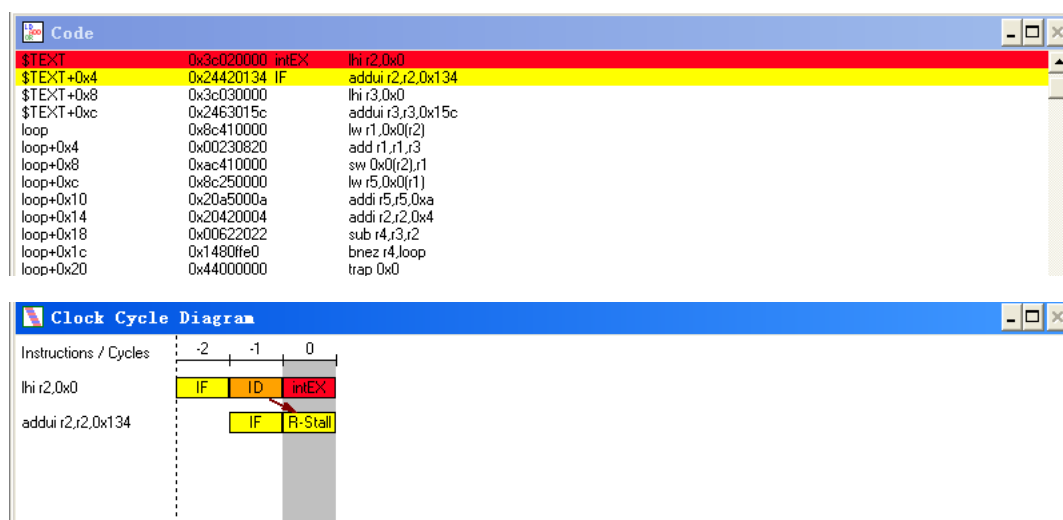


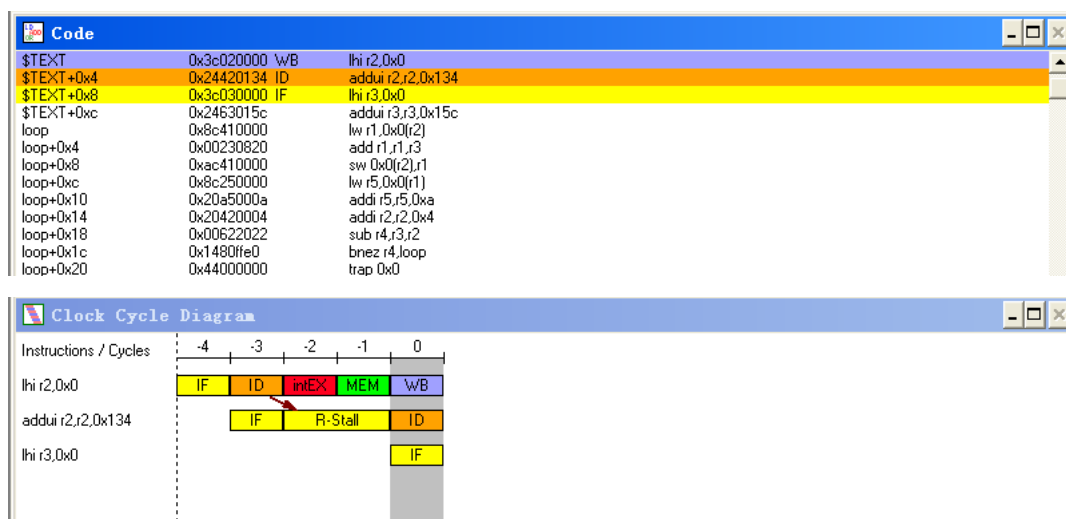
# 山东大学计算机科学与技术学院

## 计算机体系结构课程实验报告

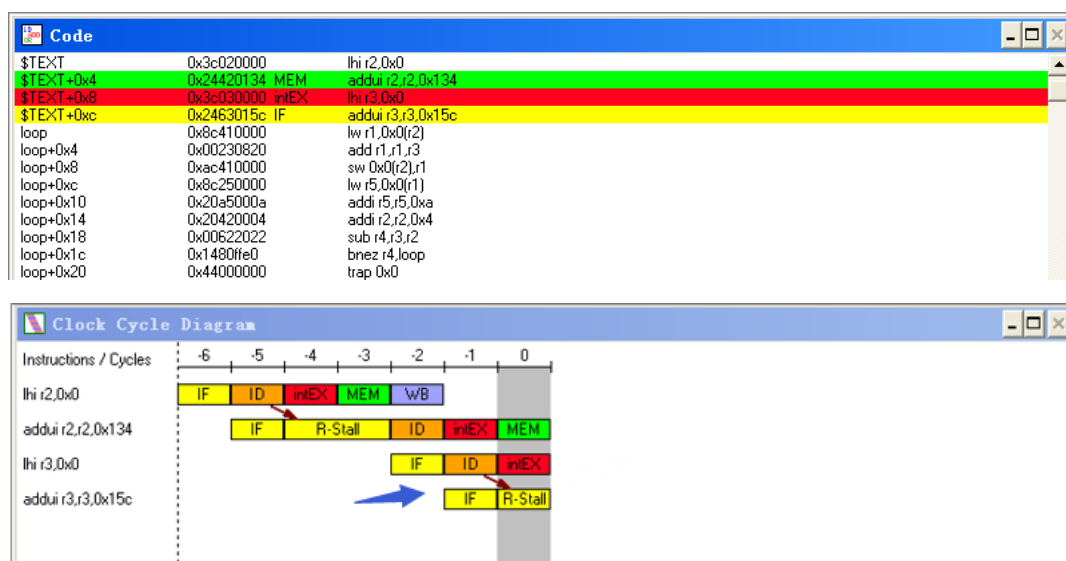
学号：201800130031	姓名：来苑	班级：计科 1 班
实验题目：实验五 数据相关		
实验学时：2 学时	实验日期：2021. 6. 4	
实验目的： 通过本实验，加深对数据相关的理解，掌握如何使用定向技术来减少相关带来的暂停。		
硬件环境： 机房		
软件环境： WindowsXp		
实验步骤与内容： 1. 在不采用定向技术的情况下，用 WinDLX 模拟器执行程序 data_d.s 1.1 初始化和配置环境，装入程序 data_d.s  1.2 关闭定向技术  1.3 单步执行程序 单步执行到此处时，产生了第一个数据相关。指令 ADDUI R2, R2, A & 0xFFFF 需要等待指令 LHI R2, (A>>16) & 0xFFFF 将 R2 写回内存，由于没有开启定向技术，所以等待 MEM 周期结束，指令 ADDUI R2, R2, A & 0xFFFF 才可以继续执行。		



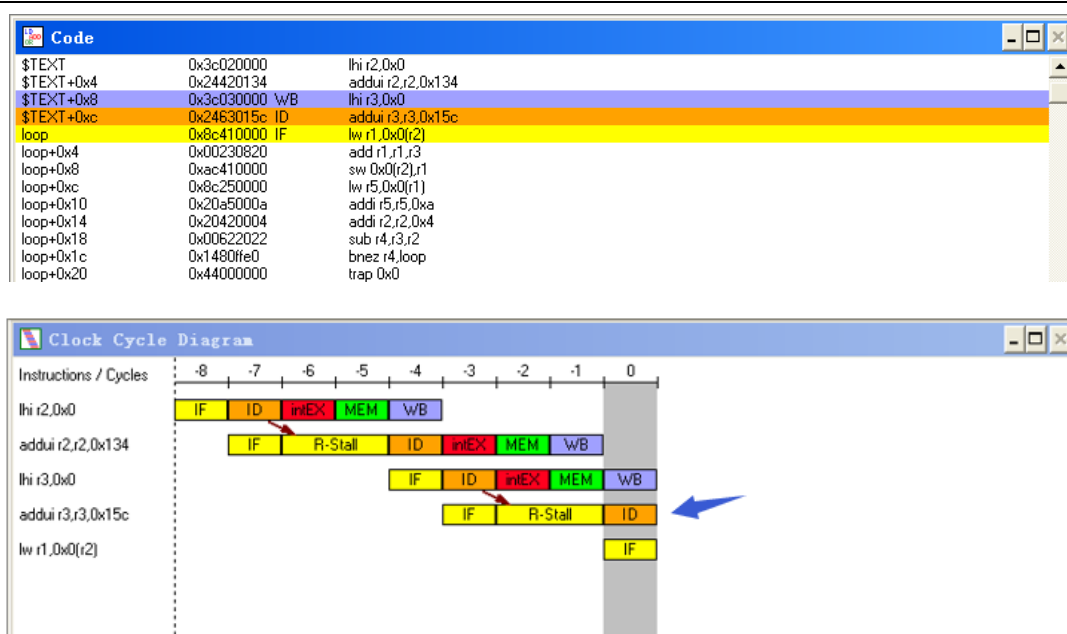
待指令 LHI R2, (A>>16) & 0xFFFF 的 MEM 周期结束后，下一条指令 ADDUI R2, R2, A & 0xFFFF 才得以进入 ID 周期：



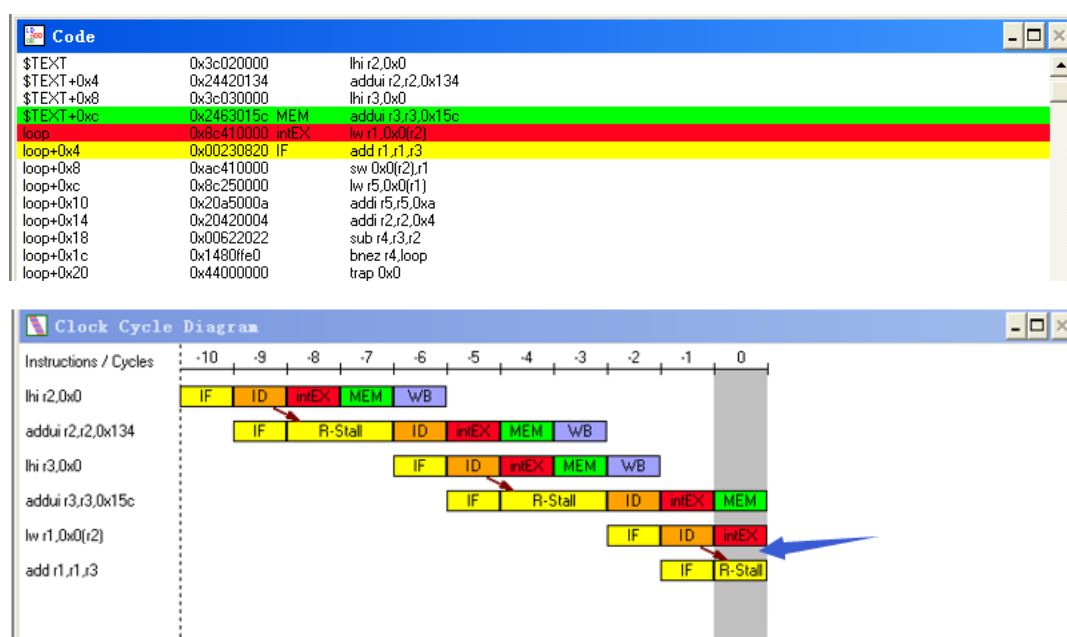
继续执行，此处出现了第二个数据相关。与第一个数据相关相似，这里指令 ADDUI R3, R3, B&0xFFFF 需要等待指令 LHI R3, (B>>16)&0xFFFF 将寄存器 R3 写回内存。



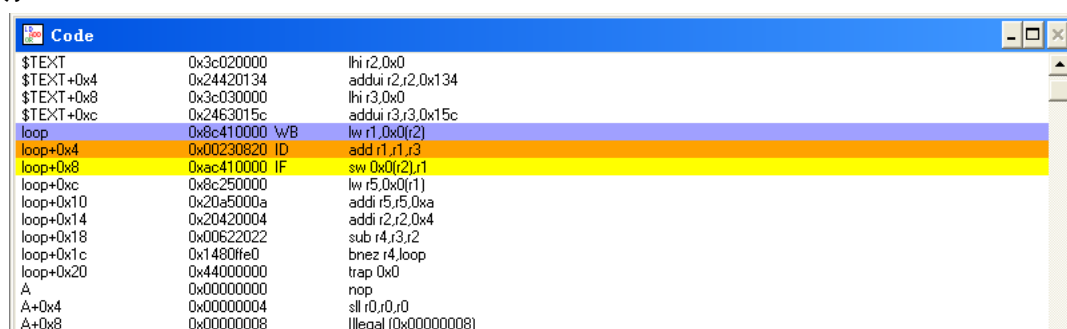
待指令 LHI R3, (B>>16)&0xFFFF 的 MEM 周期结束后，下一条指令 ADDUI R3, R3, B&0xFFFF 才得以进入 ID 周期：

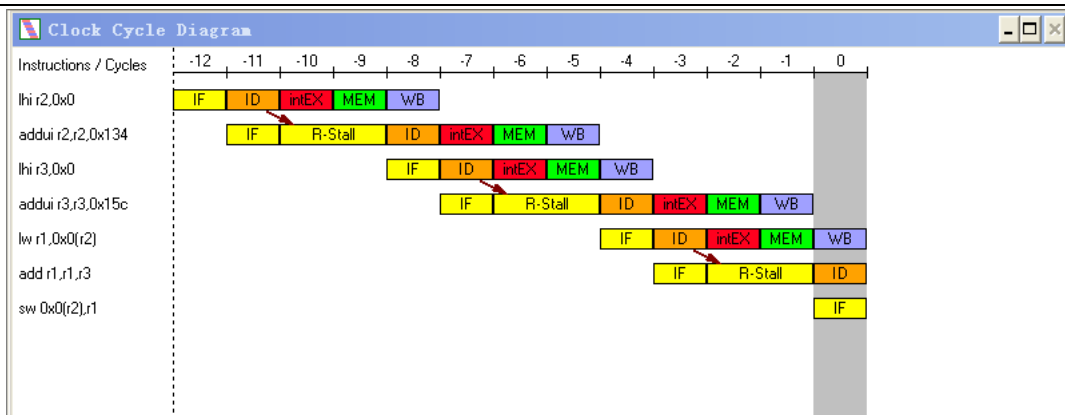


继续执行，此处出现了第三个数据相关。指令 **ADD R1, R1, R3** 需要等待指令 **LW R1, 0 (R2)**将寄存器 R1 写回内存。



待指令 **LW R1, 0 (R2)**的 MEM 周期结束后，下一条指令 **ADD R1, R1, R3** 才得以进入 ID 周期：

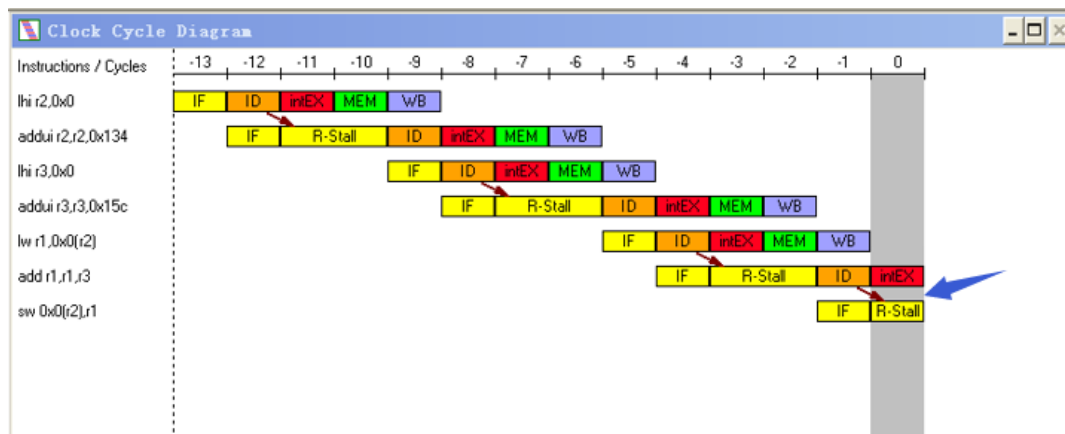




继续执行，此处出现第四个数据相关。指令 **SW 0(R2), R1** 需要等待指令 **ADD R1, R1, R3** 计算出 R1，并将寄存器 R1 写回内存。

```

Code
$TEXT      0x3c020000      lhi r2,0x0
$TEXT+0x4  0x24420134      addui r2,r2,0x134
$TEXT+0x8  0x3c030000      lhi r3,0x0
$TEXT+0xc  0x2463015c      addui r3,r3,0x15c
loop       0x8c410000      lw r1,0x0(r2)
loop+0x4   0x00230820      add r1,r1,r3
loop+0x8   0xac410000      sw 0x0(r2),r1
loop+0xc   0x8c250000      lw r5,0x0(r1)
loop+0x10  0x20a5000a      addi r5,r5,0xa
loop+0x14  0x20420004      addi r2,r2,0x4
loop+0x18  0x00622022      sub r4,r3,r2
loop+0x1c  0x1480ffe0      bnez r4,loop
loop+0x20  0x44000000      trap 0x0
  
```



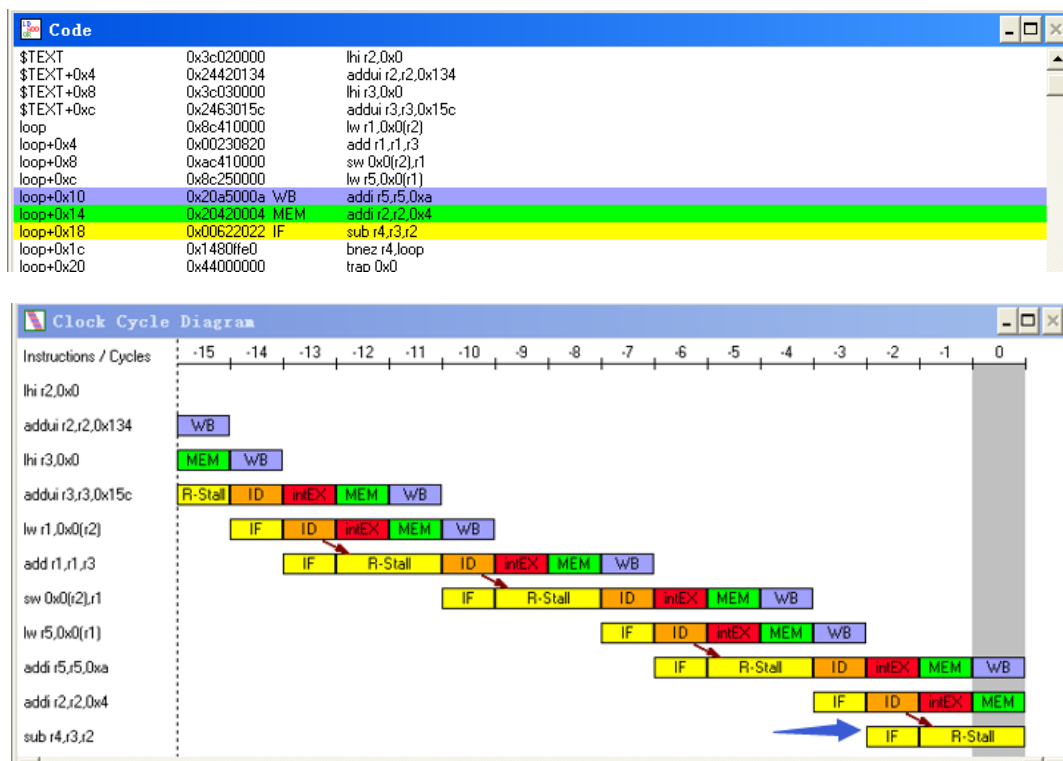
待指令 **ADD R1, R1, R3** 进入写回周期时，下一条指令 **SW 0(R2), R1** 才得以进入 ID 周期。（这里应当是假设前一条指令在前半个周期内写回 R1 寄存器，下一条指令在后半个周期读取 R1 寄存器）。

```

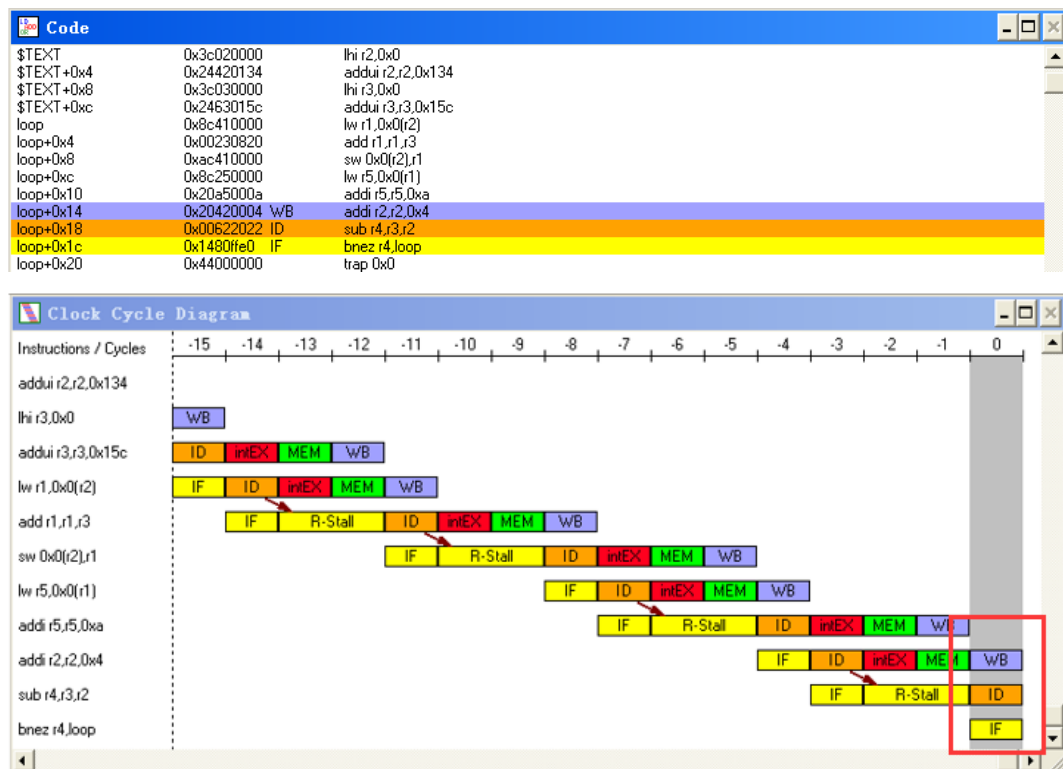
Code
$TEXT      0x3c020000      lhi r2,0x0
$TEXT+0x4  0x24420134      addui r2,r2,0x134
$TEXT+0x8  0x3c030000      lhi r3,0x0
$TEXT+0xc  0x2463015c      addui r3,r3,0x15c
loop       0x8c410000      lw r1,0x0(r2)
loop+0x4   0x00230820      add r1,r1,r3
loop+0x8   0xac410000      sw 0x0(r2),r1
loop+0xc   0x8c250000      lw r5,0x0(r1)
loop+0x10  0x20a5000a      addi r5,r5,0xa
loop+0x14  0x20420004      addi r2,r2,0x4
loop+0x18  0x00622022      sub r4,r3,r2
loop+0x1c  0x1480ffe0      bnez r4,loop
loop+0x20  0x44000000      trap 0x0
  
```



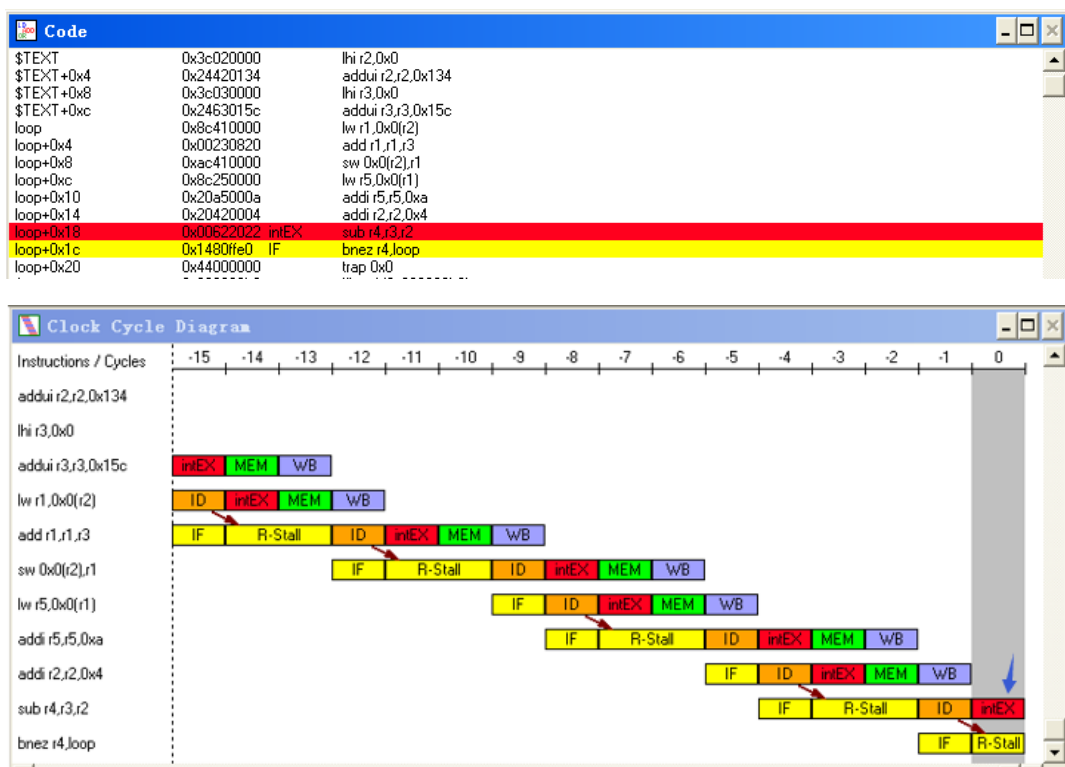
继续执行，此处出现第六个数据相关。指令 **SUB R4, R3, R2** 需要等待指令 **ADDI R2, R2, #4** 执行完毕才能进入 ID 周期。



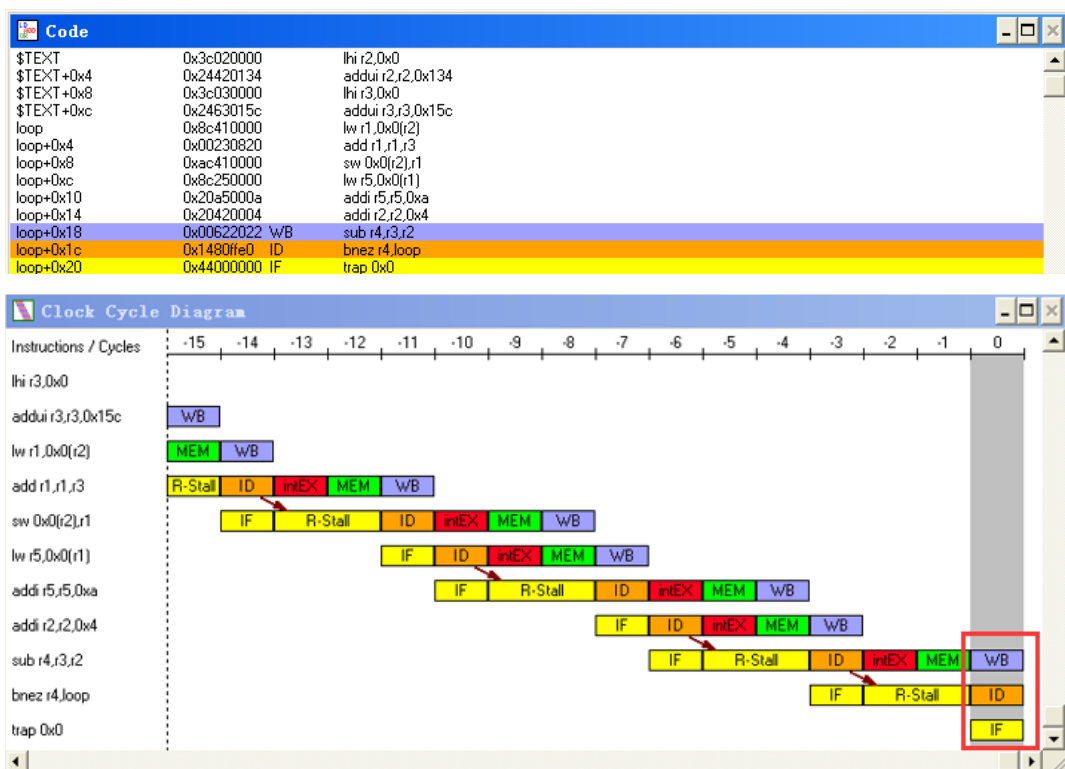
待指令 **ADDI R2, R2, #4** 进入写回周期时，下一条指令 **SUB R4, R3, R2** 才得以进入 ID 周期。（这里应当是假设前一条指令在前半个周期内写回 R2 寄存器，下一条指令在后半个周期读取 R2 寄存器）。



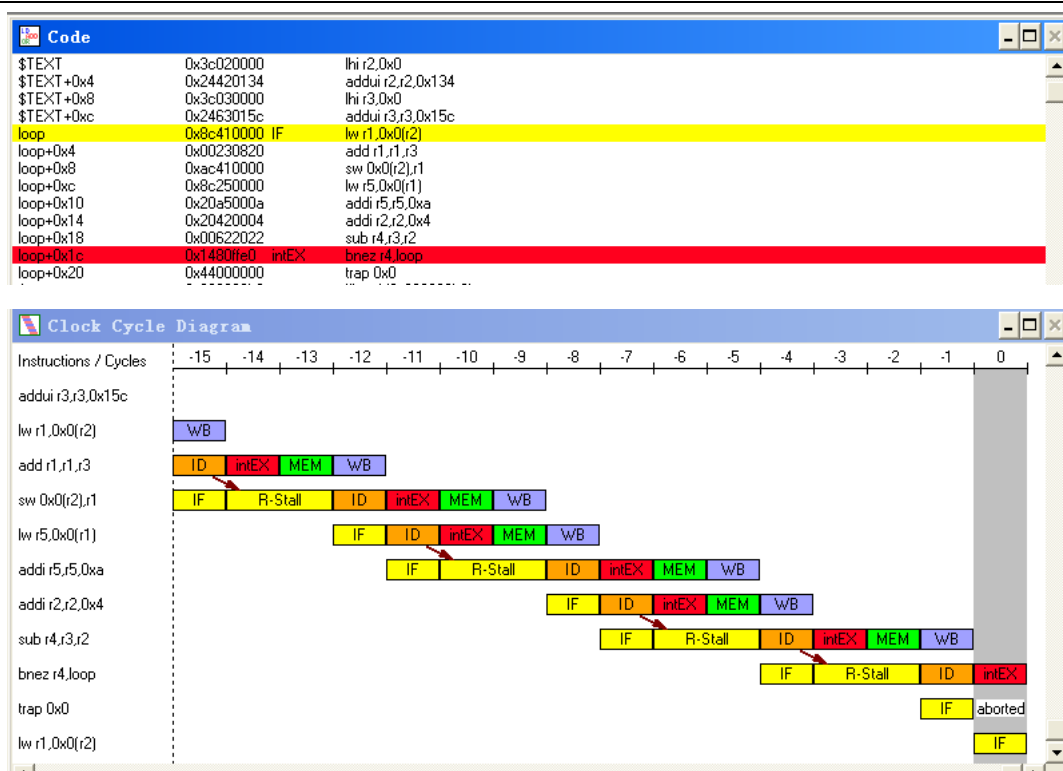
继续执行，此处出现第七个数据相关。指令 **BNEZ R4, loop** 需要等待指令 **SUB R4, R3, R2** 执行完毕才能进入 ID 周期。



待指令 SUB R4, R3, R2 进入写回周期时，下一条指令 BNEZ R4, loop 才得以进入 ID 周期。（这里应当是假设前一条指令在前半个周期内写回 R4 寄存器，下一条指令在后半个周期读取 R4 寄存器）。



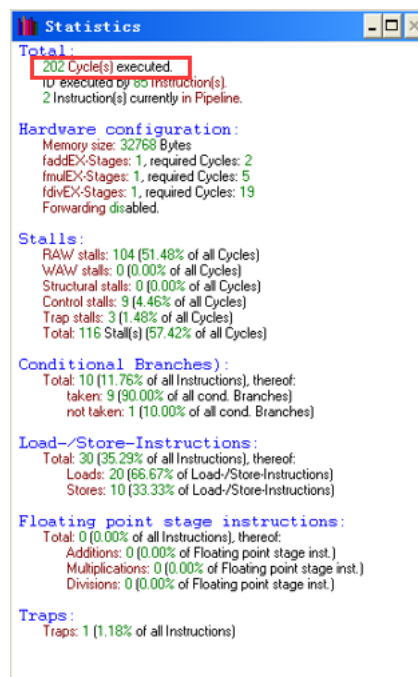
至此，一个循环结束。一共有十个循环。



- 记录数据相关引起的暂停时钟周期数以及程序执行的总时钟周期数，计算暂停时钟周期数占总执行周期数的百分比。

进入周期前，有 4 个暂停周期，每个循环会产生 10 个暂停周期，一共 10 个循环，所以中共有 104 个暂停周期。

查 Statistic 得到总周期数为 202 个，所以暂停时钟周期数栈总执行周期数的  $104/202=51.48\%$ 。

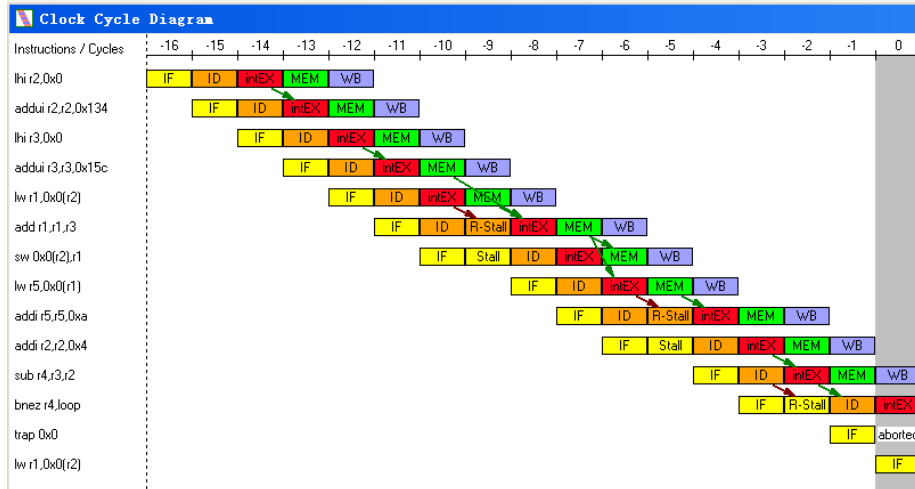


- 在采用定向技术的情况下，用 WinDLX 模拟器再次运行程序 data\_d.s 打开定向技术以后单步执行程序

查看一个循环，可以看到，原来的数据相关被解决了，但又产生了新的数据相关和结构相关。需要注意的是 lw 指令与 lhi 指令不同。lhi 指令不需要访存，而 lw 指令在译码周



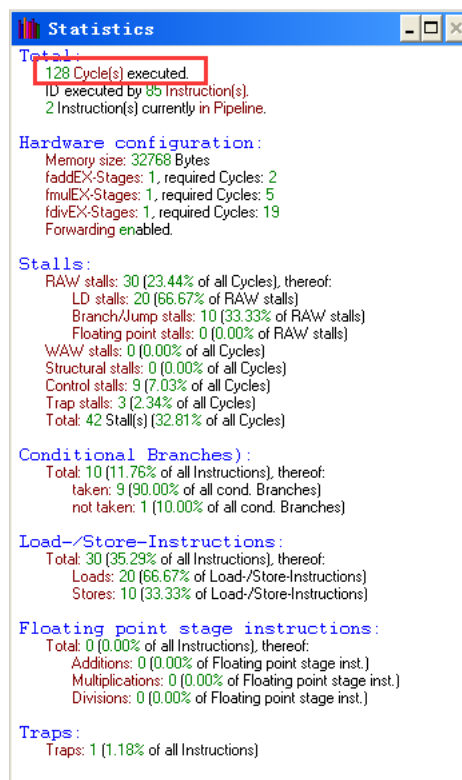
期内读取寄存器的值，在执行/有效地址计算周期 intEX 内计算有效地址，然后在访存周期 MEM 才取得数据，所以不能通过 ALU 到通用寄存器组的通路，即定向技术，来解决数据相关，所以会有前两个红色箭头的出现，并且害引起了结构相关。第三个红色箭头是由于定向技术发生在 ALU 计算后，即 intEX 周期中，所以此处会有一个不可避免的数据相关。



4. 记录数据相关引起的暂停时钟周期数以及程序执行的总时钟周期数，计算暂停时钟周期数占总执行周期数的百分比。

一共 10 个循环，每个循环有 3 个暂停时钟周期，所以一共是 30 个暂停时钟周期。暂停时钟周期数占总执行周期数百分比等于  $30/128=23.44\%$ 。

可见，虽然定向技术不能避免所有的数据相关，但是极大地提高了效率。



5. 根据上面记录的数据，计算采用定向技术后性能提高的倍数。

$202/128=1.58$

提高了约 0.58 倍。

#### 结论分析与体会：

通过本次实验，我对流水线的五个阶段有了更深的理解，也加深了对 load 指令执行流程的认识。同时，我也进一步体会到了数据相关对性能的较大影响，以及定向技术能够有效地解决数据相关。