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：201900130133	姓名：施政良	班级： 四班
实验题目：熟悉 WinDLX 的使用		
实验学时：2	实验日期： 2022-04-21	

实验目的：

（1）通过本实验，熟悉 WinDLX 模拟器的操作和使用。

（2）了解 DLX 指令集结构 及其特点。

硬件环境：

WinDLX (是一个基于 Windows 的 DLX 模拟器)

软件环境：

Windows 7

实验步骤与内容：

实验内容

本次实验主要涉及 xxxx,具体的实验步骤可以划分为如下几个步骤

（1） 用 WinDLX 模拟器执行求阶乘程序 facts 。执行步骤详见“WinDLX 教程”。

（2） 该这个程序说明浮点指令的使用。该程序从标准输入读入一个整数，求其阶乘，然后将结果输出。程序中调用了 input.s 中的输入子程序，这个子程序用于读入正整数。

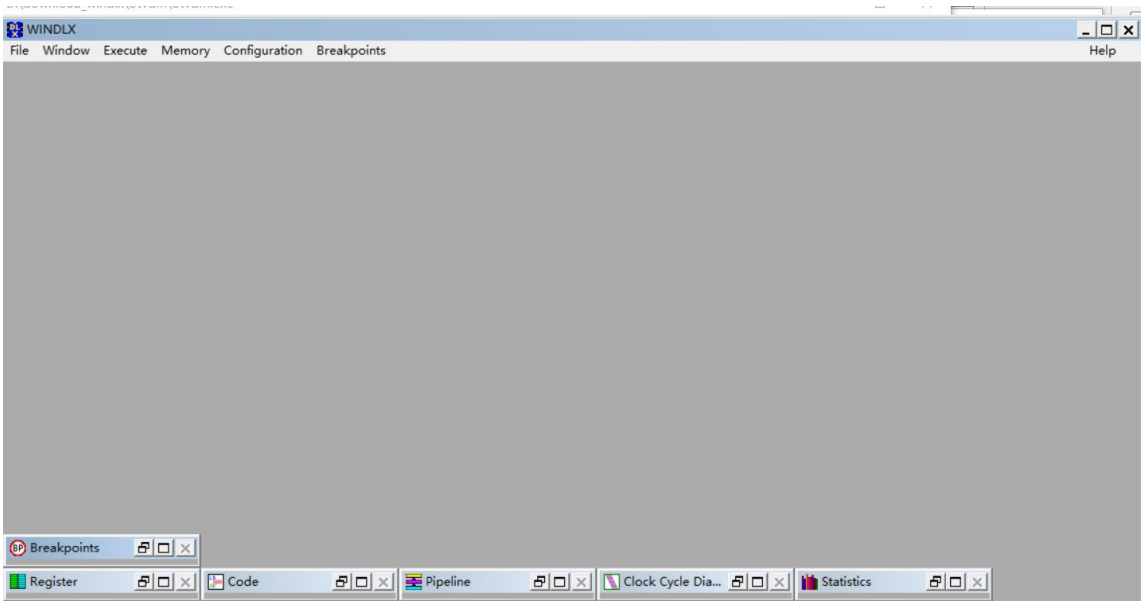
（3） 输入数据“3”采用单步执行方法，完成程序并通过上述使用 WinDLX，总结 WinDLX 的特点。

（4） 注意观察变量说明语句所建立的数据区，理解 WinDLX 指令系统。

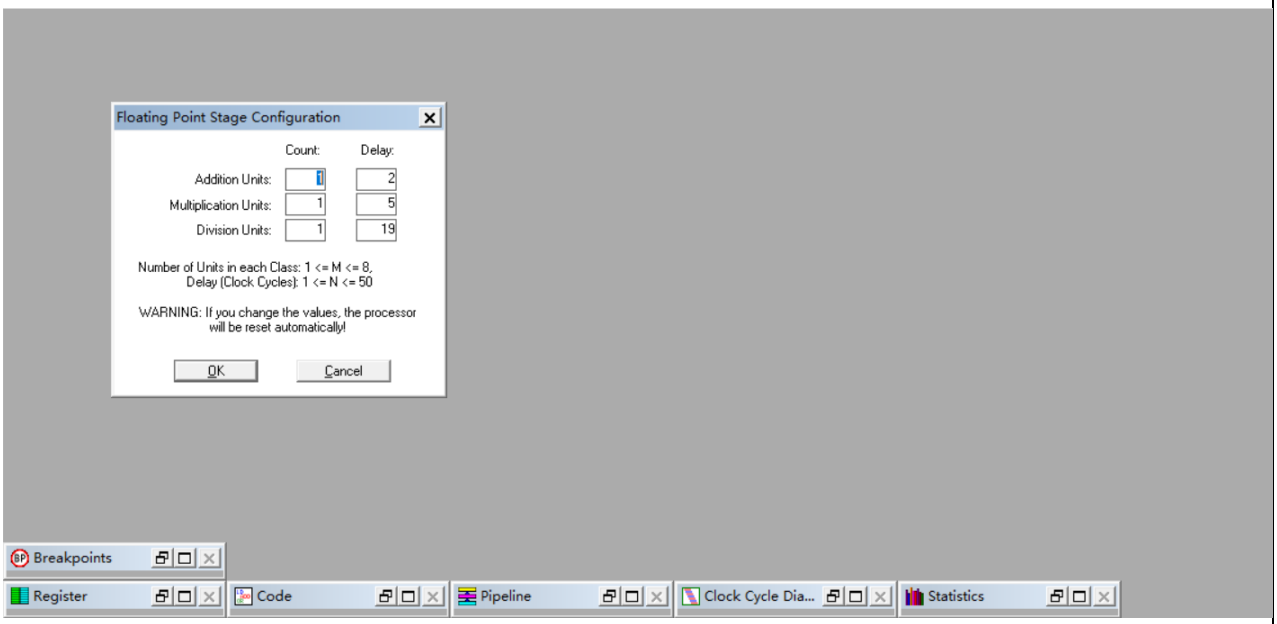
具体实验过程

1. 配置 WinDLX

首先双击 WinDLX 图标，运行 WinDLX 可执行文件，之后将出现 一个带有六个图标的主窗口，如下图所示。



为了初始化模拟器，点击 File 菜单中的 Reset all 菜单项，弹出一个“Reset DLX”对话框。然后点击窗口中的“确认”按钮即可。WinDLX 可以在多种配置下工作。你可以改变流水线的结构和时间要求、存储器大小和其他几个控制模拟的参数。点击 Configuration / Floating Point Stages（点击 Configuration 打开菜单，然后点击 Floating Point Stages 菜单项），选择如下标准配置：



点击 Configuration / Memory Size ，可以设置模拟处理器的存储器大小。应设置为 0x8000，然后，点击 OK 返回主窗口。

2. 装载测试程序

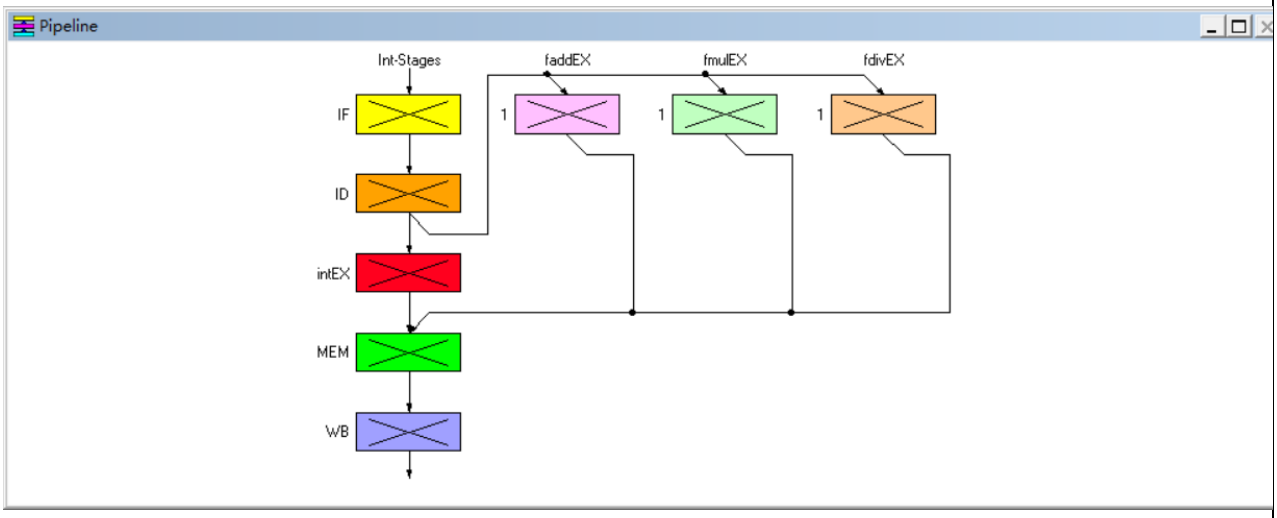
在开始模拟之前，至少应装入一个程序到主存。为此，选择 File / Load Code or Data，窗口中 会列出目录中所有汇编程序。 我们在前面已经提到，fact.s 计算一个整型值的阶乘； input.s 中包含一个子程序，它读标准 输入（键盘）并将值存入 DLX 处理器的通用寄存器 R1 中。按如下步骤操作，可将这两个文件装 入主存。

3. 模拟

在主窗口中，你可以看见六个图标，它们分别为“Register”，“Code”，“Pipeline”，“Clock Cycle Diagram”，“Statistics”和“Breakpoints”。点击其中任何一个将弹出一个新窗口（子窗 口）。在模拟过程中将介绍每一个窗口的特性和用法

3.1 Pipeline 窗口

首先分析 DLX 处理器的内部结构。双击图标 Pipeline，出现一个子窗口，窗 口中用图表形示显示了 DLX 的五段流水线。

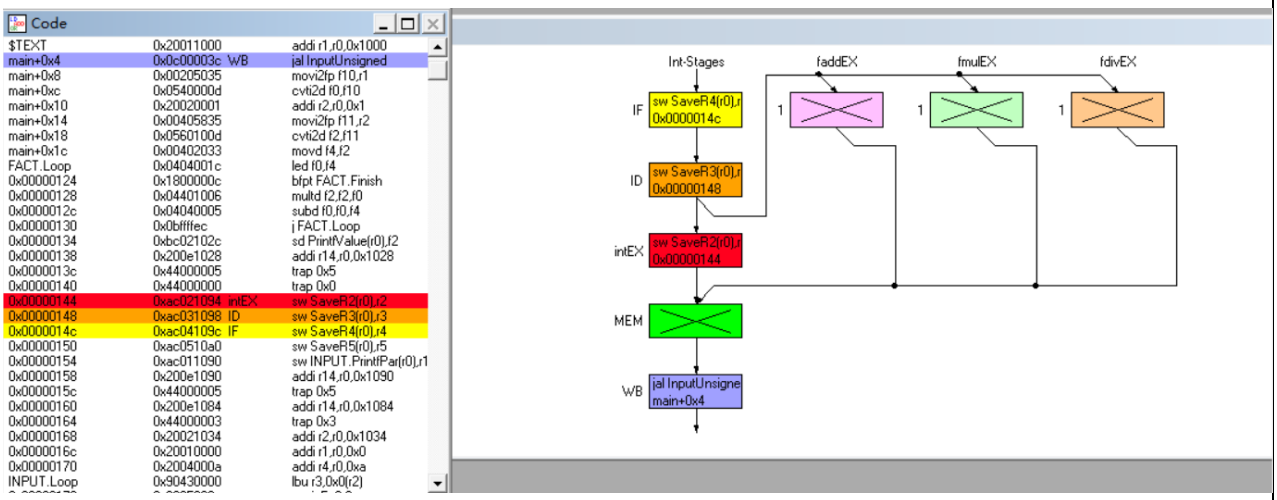


此图显示 DLX 处理器的五个流水段和浮点操作 (加 / 减, 乘和除)的单元。

3.2 Code 窗口

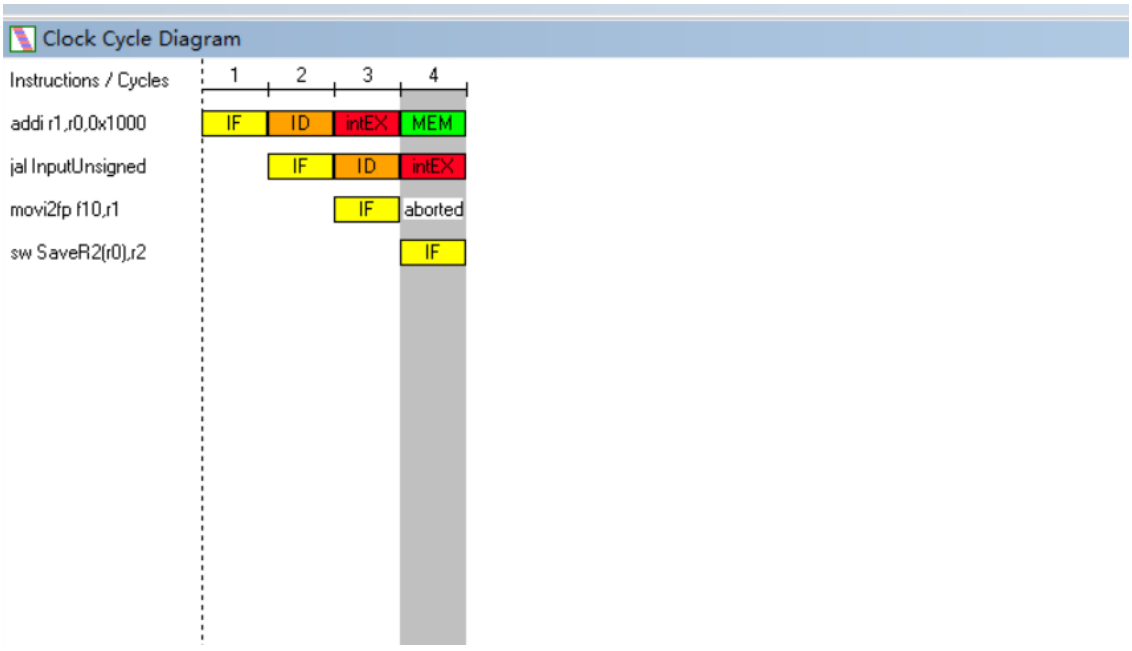
双击 Code 图标，可看到代表存储器内容的三栏信息，从左到右依次 为：地址 (符号或数字)、命令的十六进制机器代码和汇编命令。 现在，点击主窗口中的 Execution 开始模拟。在出现的下拉式菜单中，点击 Single Cycle 或 按 F7 键。 这时，窗口中带有地址“$TEXT”的第一行变成黄色。按下 F7 键，模拟就向前执行一步， 第一行的颜色变成橘黄色，下一行变成黄色.。这些不同颜色指明命令处于流水线的哪一段。如果 Pipeline 窗口已经关闭，请双击相应图标重新打开它。 如果窗口足够大，你能够看到命令“jal

InputUnsigned”在 IF 段，“addi r1, r0, 0x1000”在第二段 ID。其他方框中带有一个“X”标志，表明没有处理有效信息



3.3 Clock Cycle Diagram 窗口

使所有子窗口图标化，然后打开 Clock Cycle Diagram 窗口。它显示流水线的时空图



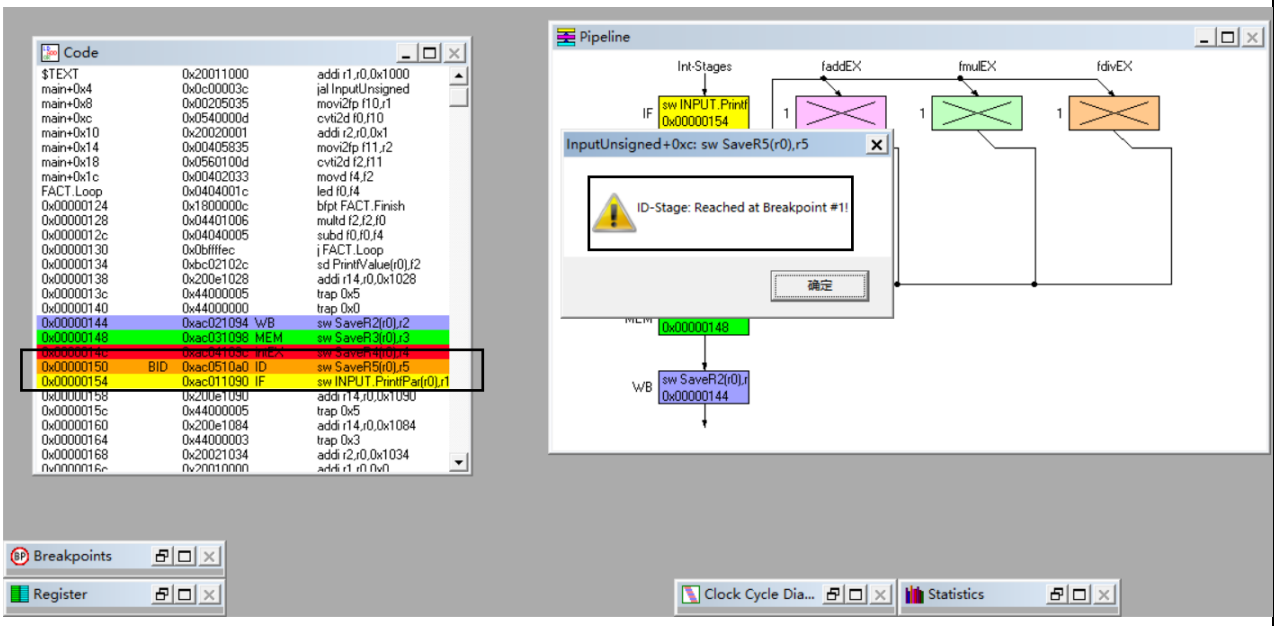
观察窗口视图，看到模拟正在第四时钟周期，其中第一条命令正在 MEM 段，第二条命令在 intEX 段，第四条命令在 IF 段。而第三条命令指示为“aborted”。

出现上述指令流水的原因是：第二条命令（jal）是无 条件分支指令， 但只有在第三个时钟周期， jal 指令被译码后才知道，这时，下一条命令 movi2fp 已经取出，但需执行的下一条命令在另一个地址处，因而，movi2fp 的执行应被取消，在流水线 中留下气泡。

3.4 Breakpoint 窗口

之后打开 Code 窗口观察代码，可以发现接下来的几条指令几乎都是 sw-操作，即将寄存器中的数写入存储器中。为了加快处理过程，可以使用断点。

指向 Code 窗口中包含命令 trap 0x5 的 0x0000015c 行，此命令是写屏幕的系统调用单击命令行，然后点击主窗口菜单 Code，单击 Set Breakpoint （确保命令行仍被标记！），将弹出 一个新的“Set Breakpoint”窗口。通过此窗口，你可以选择命令运行到流水线的哪一阶段时，程 序停止执行。缺省为 ID 段。点击 OK 关闭窗口。



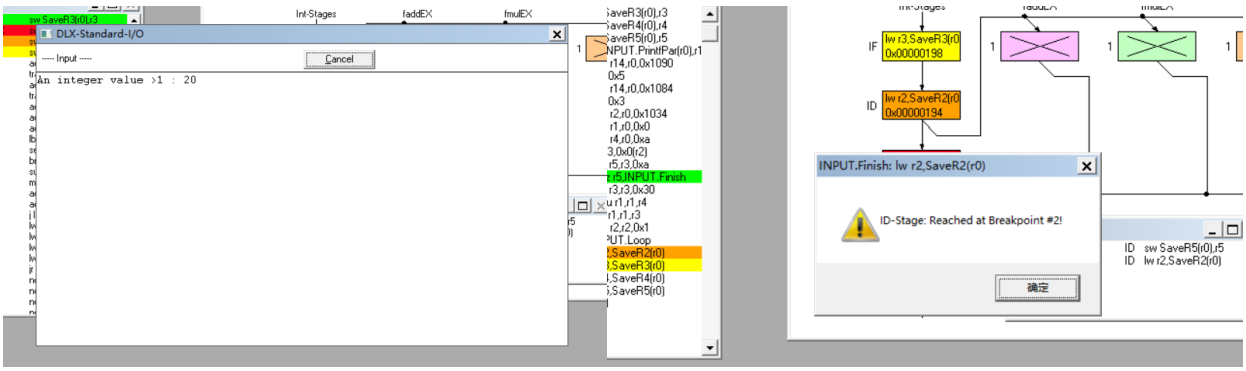
在 Code 窗口中， trap 0x5 行上出现 了“BID”，它表示当本指令在译码段时，程序中止执行。

只要单击图标 Reakpoints，弹出一个小窗口，其中显示了所有断点。

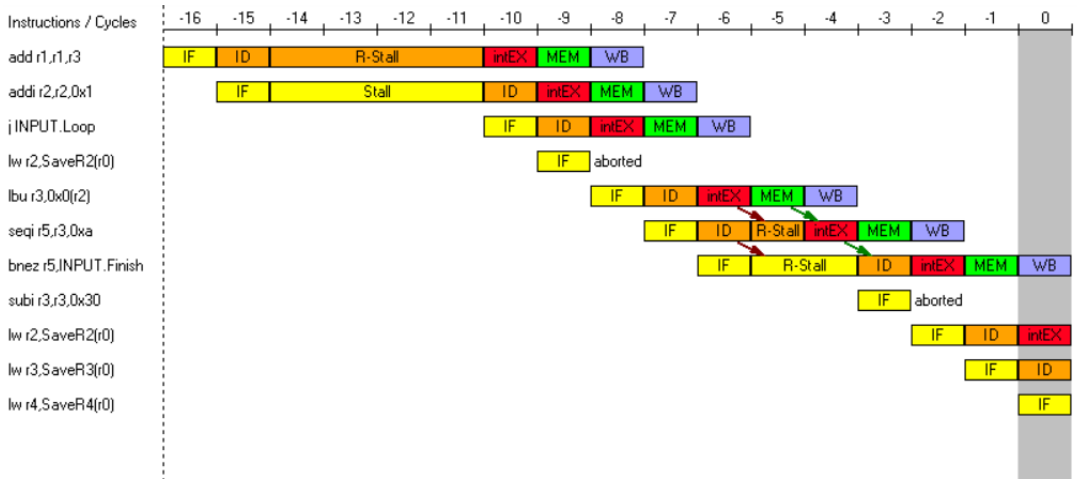
重新使窗口图标化。 现在你只要点击 Execution / Run 或按 F5，模拟就继续运行。会出现一个对话框提示你 “ID-Stage: reached at Breakpoint #1”，按"确认"按钮关闭。 点击 Clock cycle diagram 窗口中的 trap 0x5 行，你将看到模拟正处于时钟周期 14。trap 0x5 行如 下所示：

3.5 Register 窗口

为进一步模拟，点击 Code 窗口，用箭头键或鼠标向下滚动到地址为 0x00000194 的那一行 （指令是 lw r2, SaveR2(r0)），点击此行，然后按 Ins 键，或点击 Code / Set Breakpoint / OK，在 这一行上设置一个断点。采用同样的方法，在地址 0x000001a4（指令 jar r31）处设置断点。现在 按 F5 继续运行。这时，会弹出 DLX-Standard-I/O 窗口，在信息“An integer value >1:”后鼠标闪 烁，键入 20 然后按 Enter，模拟继续运行到断点 # 2 处

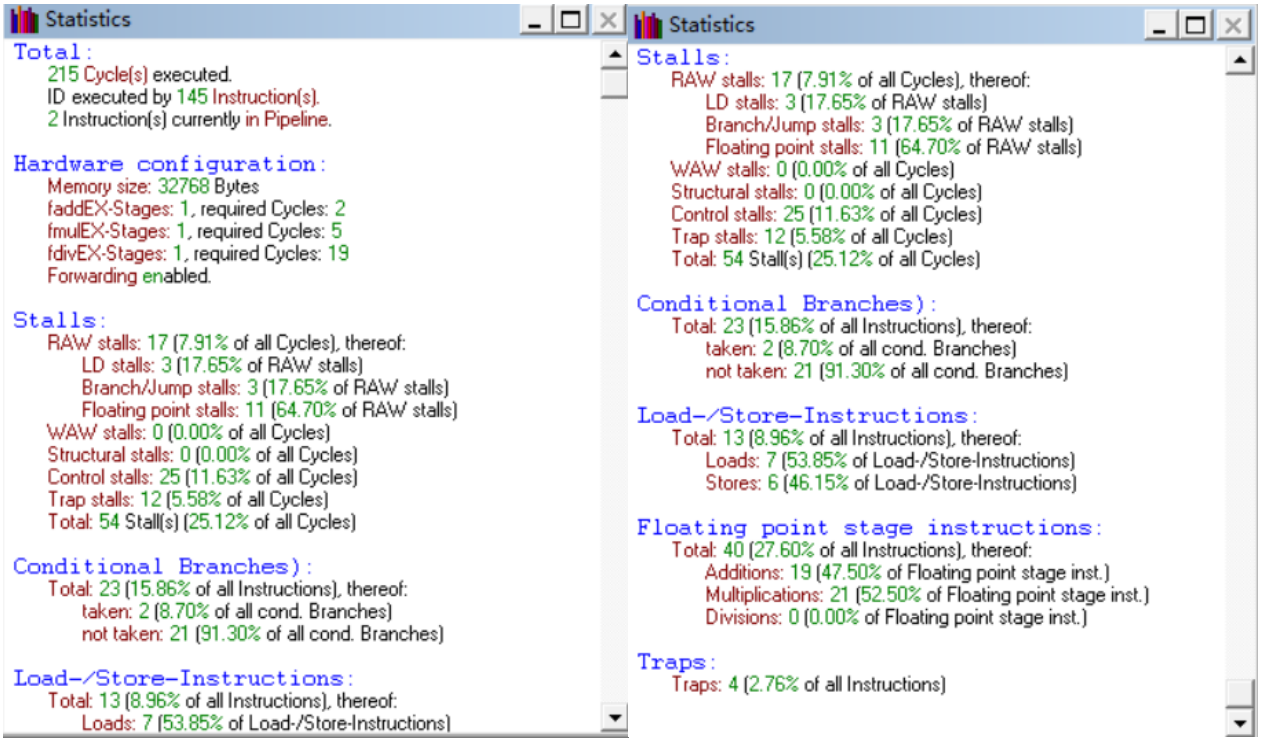


在 Clock cycle diagram 窗口中，在指令之间出现了红和绿的箭头。红色箭头表示需要一个暂 停，箭头指向处显示了暂停的原因。R-Stall（R-暂停）表示引起暂停的原因是 RAW。绿色箭头 表示定向技术的使用



3.6 Statistics 窗口

在上述实验的基础上，按 F5 使程序完成执行，出现消息“Trap #0 occurred”表明最后一条指令 trap 0 已经执行， Trap 指令中编号“0”没有定义，只是用来终止程序。双击图标 Statistics。Statistics 窗口提供各 个方面的信息：模拟中硬件配置情况、暂停及原因、条件分支、Load/Store 指令、浮点指令和 traps。 窗口中给出事件发生的次数和百分比，如 RAW stalls：17(7.91 % of all Cycles)。 在静态窗口中我们可以比较一下不同配置对模拟的影响



结论分析与体会：

结论分析

1. 关于指令流水中的相关（流水线冲突）

分析：

流水线中的相关是指相邻或相近的指令因存在某种关联，后面的指令不能在原指定的时钟周期开始执行。

一般来说，流水线中的相关主要分为如下三种类型：

(1) 结构相关：当硬件资源满足不了指令重叠执行的要求，而发生资源冲突时，就发生了结 构相关。

(2) 数据相关：当一条指令需要用到前面指令的执行结果，而这些指令均在流水线中重叠执 行时，就可能引起数据相关。

(3) 控制相关：当流水线遇到分支指令和其它能够改变 PC 值的指令时，就会发生控制相 关。

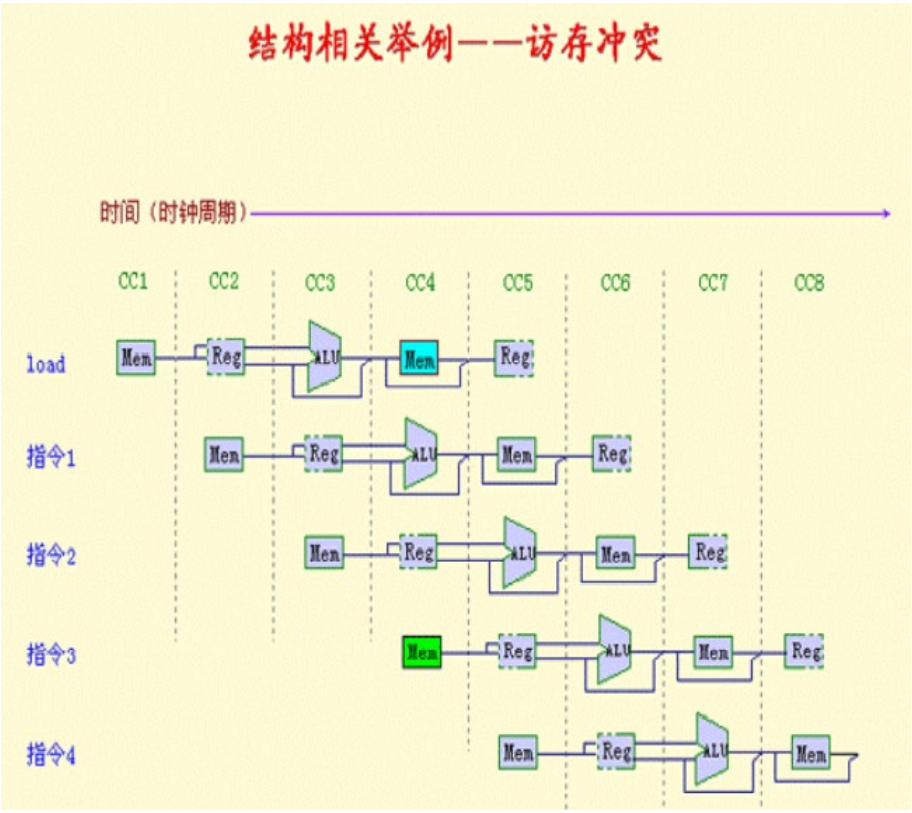
一旦流水线中出现相关，必然会给指令在流水线中的顺利执行带来许多问题，如果不能很好 地解决相关问题，轻则影响流水线的性能，甚至导致错误的执行结果。消除相关的基本方法是让流水线暂停执行某些指令，而继续执行其它一些指令。

2. 流水线的相关结构

分析：

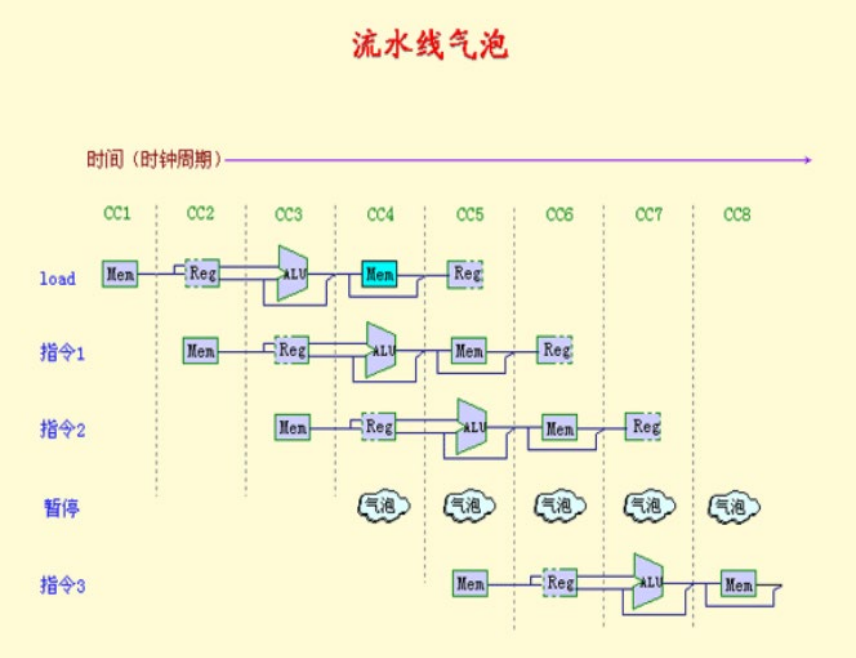
如果某些指令组合在流水线中重叠执行时，产生资源冲突，则称该流水线有结构相关。为了能够在流水线中顺利执行指令的所有可能组合，而不发生结构相关，通常需要采用流水化功能单元的方法或资源重复的方法

许多流水线机器都是将数据和指令保存在同一存储器中。如果在某个时钟周期内，流水线既 要完成某条指令对数据的存储器访问操作，又要完成取指令的操作，那么将会发生存储器访问冲突问题，产生结构相关。如下图所示。



为了解决这个问题，可以让流水线完成前一条指令 对数据的存储器访问时，暂停取后

一条指令的操作，如下图所示。该周期称为流水线的一 个暂停周期。暂停周期一般也称为流水线气泡，或简称为气泡。从图 3.3.2 可以看出，在流水线中插入暂停周期可以消除这种结构相关



由上可知，为消除结构相关而引入的暂停将影响流水线的性能。同时，为了避免结构相关，也可以考虑采用资源重复的方法。比如，在流水线机器中设置相互独立的指令存储器和数据存储器；也可 以将 Cache 分割成指令 Cache 和数据 Cache。

3. 在实际设计时，是否允许存在指令结构的相关

分析：

在实际设计中允许存在指令结构的相关。主要处于以下两点原因

（1） 减少功能单元的延迟。

（2） 减少硬件代价，如果为了避免结构相关而将流水线中的 所有功能单元完全流水化，或者设置足够的硬件资源，那么所带来的硬件代价必定很大.

体会

经过本次实验，我从流水线的角度分析了程序的运行。一般程序的执行大体分为五个部分，包括取指（IF）,译码（ID）,取数操作（OF），执行操作（EX）以及写回操作（WB）。并且，通过这种指令流水的方式，充分利用了不同硬件之间的并行性，可以显著的提高程序执行的速度。即在流水线过程中，只有第一个指令需要占用 5 个独立的周期，其余的指令都只需要占用独立的一个周期，剩下的都是和其余的指令周期交叉。

指令流水线也存在很多细节问题，例如在程序的运行过程中可能会遇到一些情况使得

流水线无法正确执行后续指令而引起流水线阻塞或停顿，这种现象称为流水线冲突或流水线冒险。在实际中可以采用编译器优化指令执行顺序，加入气泡，加入额外的旁路缓解流水线冒险的现象。

附录
实验中涉及到的汇编程序代码如下所示

1. INPUT.s 文件

```
;-------------------------------------------------------------------------
-
;Subprogram call by symbol "InputUnsigned"
;expect the address of a zero-terminated prompt string in R1
;returns the read value in R1
;changes the contents of registers R1,R13,R14
;-------------------------------------------------------------------------
-

        .data

        ;*** Data for Read-Trap
ReadBuffer: .space      80
ReadPar:    .word       0,ReadBuffer,80

        ;*** Data for Printf-Trap
PrintfPar:  .space      4

SaveR2:     .space      4
SaveR3:     .space      4
SaveR4:     .space      4
SaveR5:     .space      4


        .text

        .global     InputUnsigned
InputUnsigned:
        ;*** save register contents
```

```
        sw      SaveR2,r2
        sw      SaveR3,r3
        sw      SaveR4,r4
        sw      SaveR5,r5

        ;*** Prompt
        sw      PrintfPar,r1
        addi        r14,r0,PrintfPar
        trap        5

        ;*** call Trap-3 to read line
        addi        r14,r0,ReadPar
        trap        3

        ;*** determine value
        addi        r2,r0,ReadBuffer
        addi        r1,r0,0
        addi        r4,r0,10    ;Decimal system

Loop:       ;*** reads digits to end of line
        lbu     r3,0(r2)
        seqi        r5,r3,10    ;LF -> Exit
        bnez        r5,Finish
        subi        r3,r3,48    ;??
        multu       r1,r1,r4    ;Shift decimal
        add     r1,r1,r3
        addi        r2,r2,1     ;increment pointer
        j       Loop

Finish:     ;*** restore old register contents
        lw      r2,SaveR2
        lw      r3,SaveR3
        lw      r4,SaveR4
        lw      r5,SaveR5
        jr      r31     ; Return
```

2. FACT.s 文件

```
;-------------------------------------------------------------------------
; Program begin at symbol main
```

```
3. ; requires module INPUT
4. ; read a number from stdin and calculate the factorial (type: double)
5. ; the result is written to stdout
6. ;-------------------------------------------------------------------------
7.
8.          .data
9. Prompt:      .asciiz     "An integer value >1 : "
10.
11.PrintfFormat:    .asciiz     "Factorial = %g\n\n"
12.         .align      2
13.PrintfPar:   .word       PrintfFormat
14.PrintfValue:     .space      8
15.
16.
17.         .text
18.         .global main
19.main:
20.         ;*** Read value from stdin into R1
21.         addi        r1,r0,Prompt
22.         jal     InputUnsigned
23.
24.         ;*** init values
25.         movi2fp     f10,r1      ;R1 -> D0    D0..Count register
26.         cvti2d      f0,f10
27.         addi        r2,r0,1     ;1 -> D2 D2..result
28.         movi2fp     f11,r2
29.         cvti2d      f2,f11
30.         movd        f4,f2       ;1-> D4  D4..Constant 1
31.
32.         ;*** Break loop if D0 = 1
33.Loop:        led     f0,f4       ;D0<=1 ?
34.         bfpt        Finish
35.
36.         ;*** Multiplication and next loop
37.         multd       f2,f2,f0
38.         subd        f0,f0,f4
39.         j       Loop
40.
41.Finish:      ;*** write result to stdout
42.         sd      PrintfValue,f2
```

```
43.         addi        r14,r0,PrintfPar
44.         trap        5
45.
46.         ;*** end
47.         trap        0
```