**计算机系统结构实验**

**实验三：DLX 流水线运行分析**

**学 院： 信息工程学院**

**班 级： 网络工程二班**

**学 号： 2015551621**

**姓 名： 王康**

**实验地点： 新计算机中心**

**实验时间： 2017.12.05**

**指导教师： 杨奇为**

**一、实验目的**

通过本实验，熟悉 WinDLX 模拟器的操作和使用，了解 DLX 指令集结构及其特点。

1. **实验设备环境**

WinDLX要求的硬件平台是IBM PC兼容机，WinDLX是一个Windows应用程序，运行于Windows 3.0以上的操作系统。

1. **实验原理**

(1) 用 WinDLX 模拟器执行求阶乘程序 fact.s 。执行步骤详见“WinDLX教程” 。 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。该程序中调用了 input.s 中的输入子程序，这个子程序用于读入正整数。

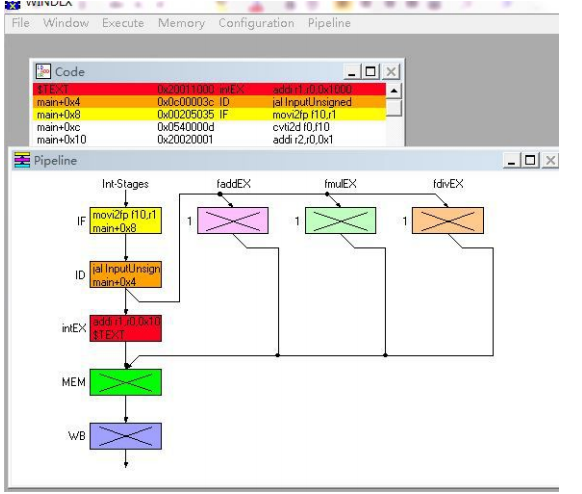
（2） 输入数据 “3” 采用单步执行方法， 完成程序并通过上述使用 WinDLX， 总结 WinDLX的特点。

1. 注意观察变量说明语句所建立的数据区，理解 WinDLX 指令系统。

**三、实验步骤**

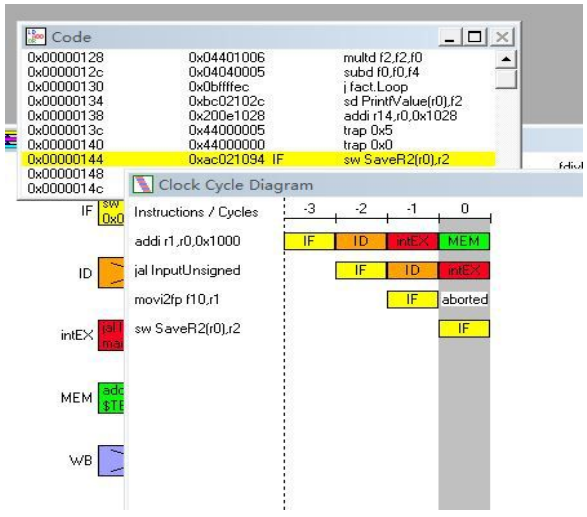
**1.**双击 Code 图标，从左到右依次为代表存储器内容的三栏信息：地址 (符号或数字)、命令的十六进制机器代码和汇编命令。

点击主窗口中的 Execution 开始模拟。按 F7 键开始执行。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。命令“jalInputUnsigned”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。



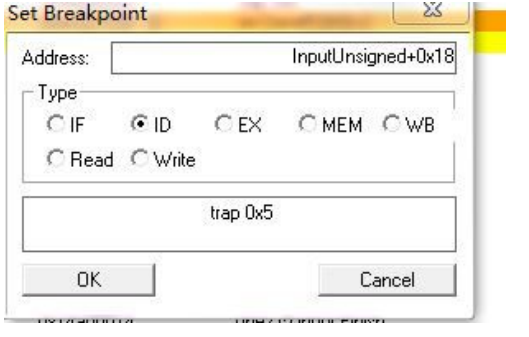
**2.**再次按下 F7 键，代码窗口中的颜色会再改变，红色表明命令处入第三段“intEX”。 再按下 F7，在代码窗口中，黄色出现在更下面的位置，并且是唯一彩色行。

打开 Clock Cycle Diagram 窗口。它显示流水线的时空图。



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”。其原因是：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.**当通过 Code 窗口观察代码时，接下来的几条指令几近一样，它们都是 sw-操作：将寄存器中的数写入存储器中。重复按 F7 将很枯燥，因此使用断点加快此过程。现在，指向 Code 窗口中包含命令 trap 0x5 的 0x0000013c 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。

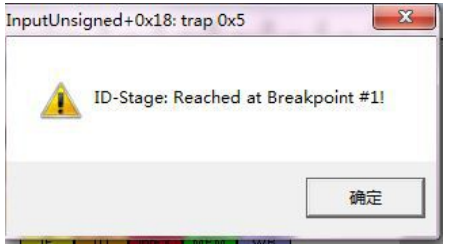


在 Code 窗口中， trap 0x5 行上出现 了“BID”，它表示当本指令在译码段时，程序中止执行。



现在点击 Execution / Run 或按 F5，模拟就继续运行，弹出窗口，输入14。

会出现一个对话框提示你“ID-Stage: reached at Breakpoint #1”，按"确认"按钮关闭。

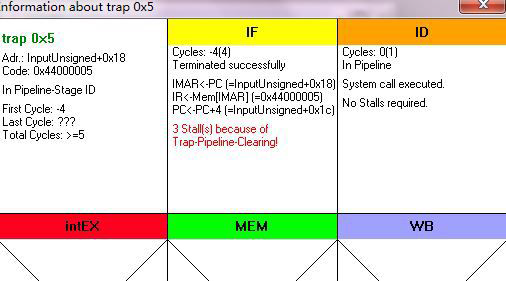


点击 Clock cycle diagram 窗口中的 trap 0x5 行，你将看到模拟正处于时钟周期 14。

trap 0x5 行如下所示：



原因是：无论何时遇到一条trap 指令时，DLX 处理器中的流水线将被清空。在Information 窗口（双击trap 行弹出）中，在IF 段显示消息“3 stall(s) because of Trap-Pipeline-Clearing!”。



指令 trap 0x5 已经写到屏幕上，你可以通过点击主窗口菜单条上的 Execute /Display DLX-I/O 来查看。

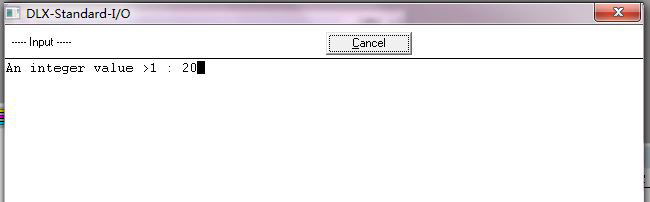


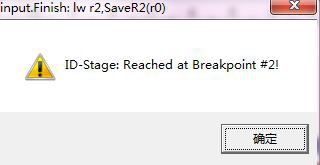
**四、实验要求**

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，点击 Code / Set Breakpoint / OK，在这

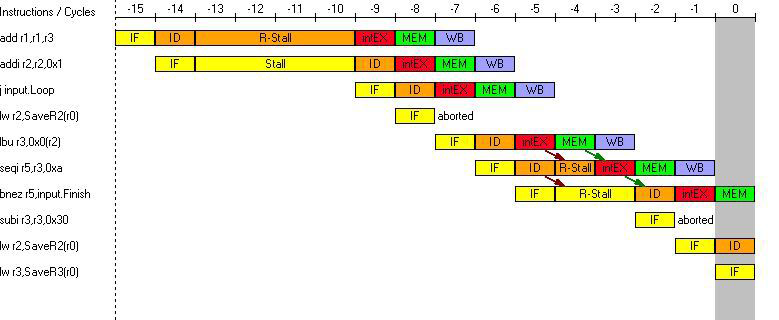
一行上设置一个断点。采用同样的方法，在地址 0x000001a4（指令 jar r31）处设置断点。

现在按 F5 继续运行。 这时， 会弹出 DLX-Standard-I/O 窗口，键入 20 然后按 Enter，模拟继续运行到断点 # 2 处。

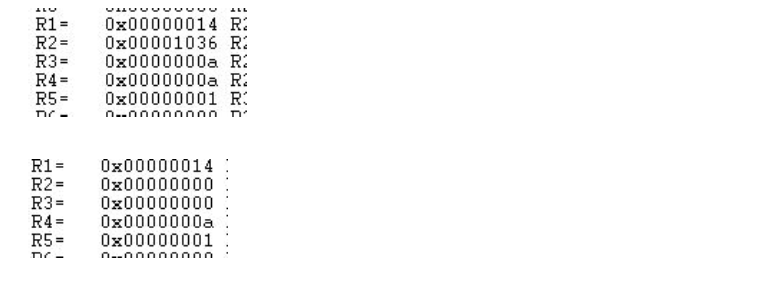




在Clock cycle diagram 窗口中，在指令之间出现了红和绿的箭头。红色箭头表示需要一个暂停，箭头指向处显示了暂停的原因。R-Stall（R-暂停）表示引起暂停的原因是RAW。绿色箭头表示定向技术的使用。



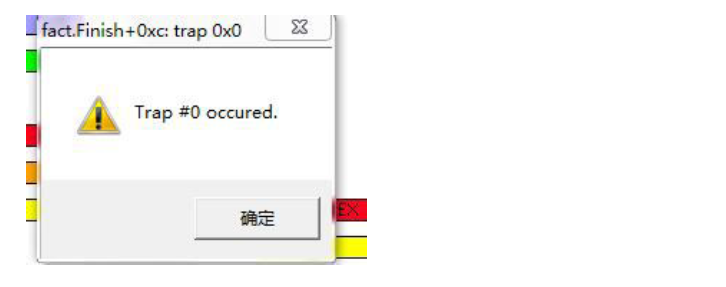
双击主窗口中的Register 图标。Register 窗口会显示各个寄存器中的内容。看一下R1 到R5 的值。按F5 使模拟继续运行到下一个断点处，有些值将发生改变，指令lw 从主存中取数到寄存器中。



**定向技术带来的加速比**

点击 Execute / Multiple Cycles 或者按 F8 键，在新出现的窗口中输入 17 ，然后按 Enter 键，模拟程序将继续运行 17 个时钟周期。向上滚动 Clock cycle diagram 窗口，直到看到指令周期 72 到 78。在 EX 段，两个浮点操作（multd and subd）分别在不同的部件上运行，它们都需要多个周期才能结束。因而在它们之后的下一条指令能取指，译码和执行，然后暂停一个周期以允许 subd 完成 MEM 段。

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)。





Statistics 窗口中的各种统计数字：总的周期数(215) 和暂停数 (17 RAW, 25 Control, 12 Trap; 54 Total) ，然后关闭窗口。

点击 Configuration 中的 Enable Forwarding 使定向无效（去掉小钩），打开断点Breakpoints 图标并点击 Breakpoints 菜单，删除所有断点，然后按 F5，键入 20 后，按Enter ，模拟程序一直运行到结束。重新查看静态窗口，控制暂停和 Trap 暂停仍然是同样的值，而 RAW 暂停从 17 变成了 53，总的模拟周期数增加到 236。

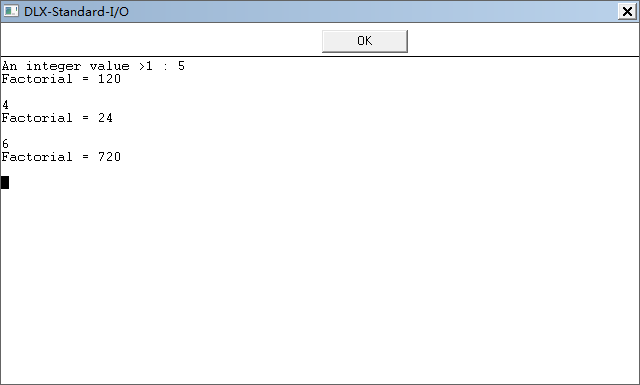


利用这些值，计算定向技术带来的加速比：236 / 215 = 1.098

所以可知 DLX forwarded比 DLX not forwarded 快 9.8%

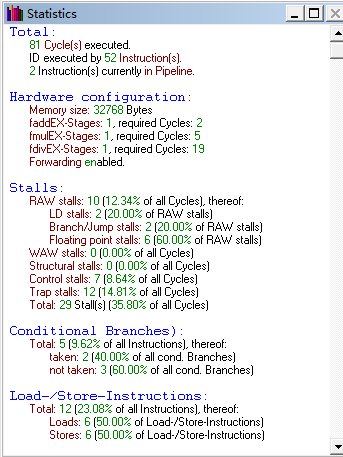
**五、实验作业**

计算自己的定向技术带来的加速比(根据按F5的次数，每个人的数据会有不同)。

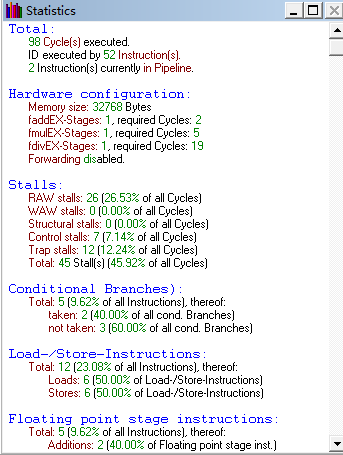


第一次F5

使用定向

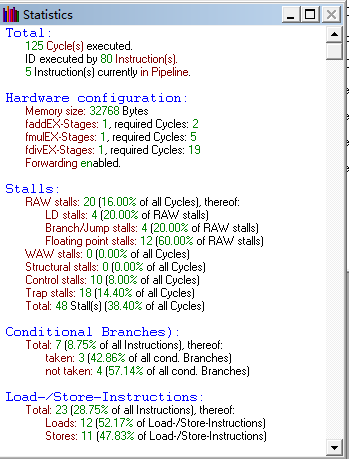


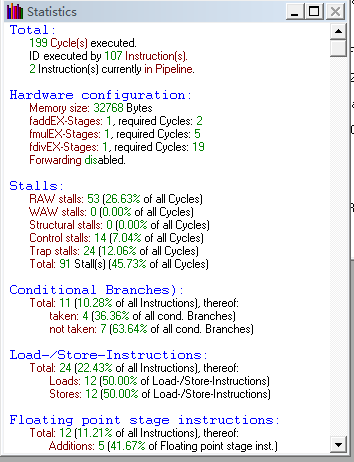
不使用定向技术



加速比：98/81=1.21

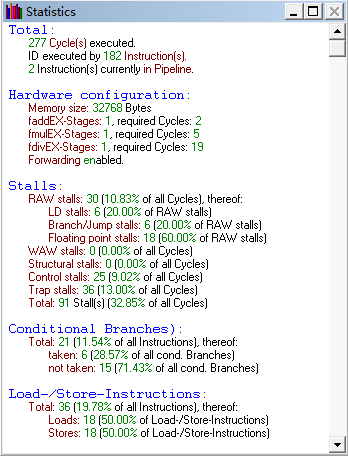
第二次F5

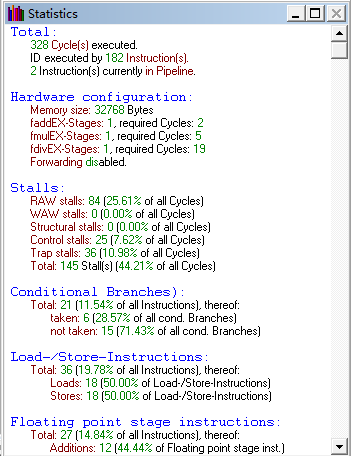




加速比：199/125=1.59

第三次F5

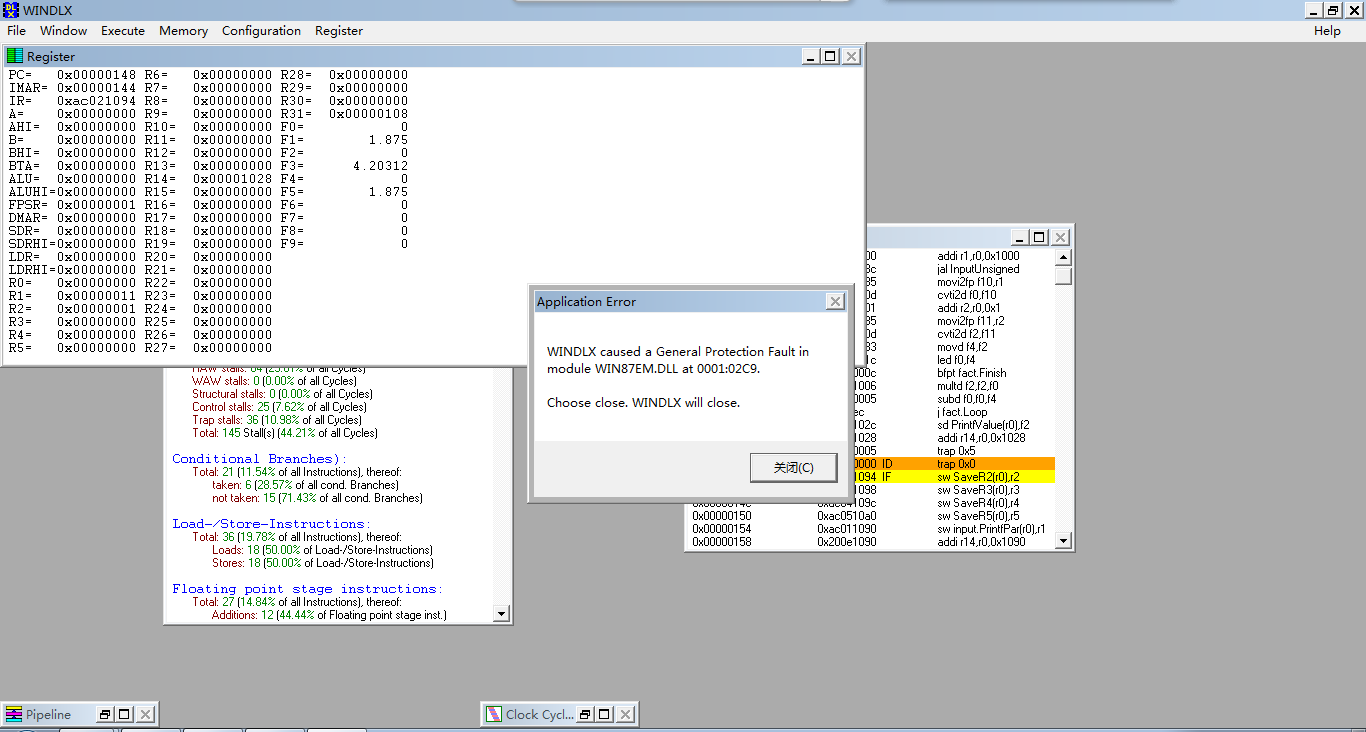




加速比：328/277=1.18

**六、实验遇到问题**

打开查看寄存器查看寄存器状态窗口时程序直接停止运行：



解决方法：

由于软件自身的问题，只要不使用该窗口时就可可解决该问题

**七、实验总结**

通过本次实验我对WinDLX有了更深一步的了解，通过对装入DLX汇编语言程序进行仿真运行，模拟出机器内部的运行状态。并且学会了以定向技术的方式来运行程序。通过流水线直观的观察到定向技术带来的作用。

同时也学会了一些常见问题的解决方法，为以后将要从事的工作有很大帮助。