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：实验一 熟悉 WinDLX 的使用		
实验学时：2 学时	实验日期：2021.4.29	

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。

硬件环境：

机房

软件环境：

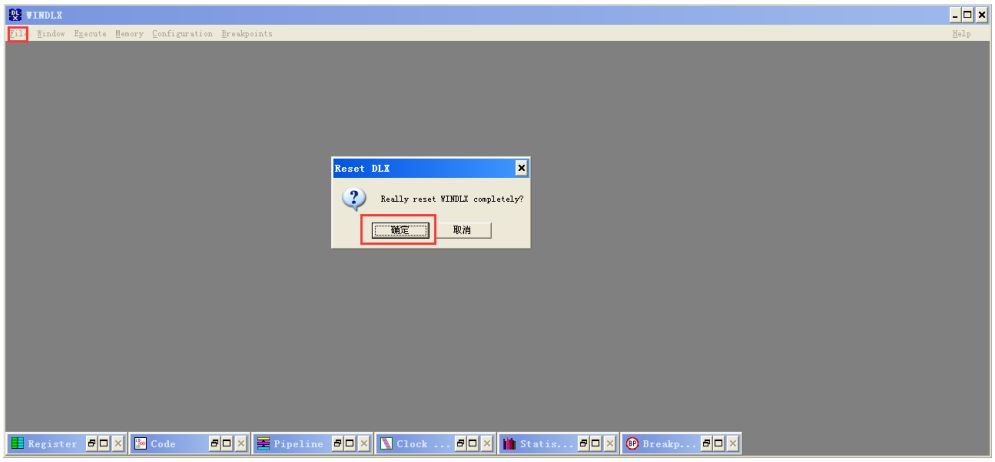
Windows xp

实验步骤与内容：

1. 用 WinDLX 模拟器执行求阶乘程序 facts

1.1 配置

首先初始化模拟器，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。



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：

	Count	Delay
Addition Units:	1	2
Multiplication Units:	1	5
Division Units:	1	19

最后，设置模拟处理器的存储器大小为设置为 0x8000。

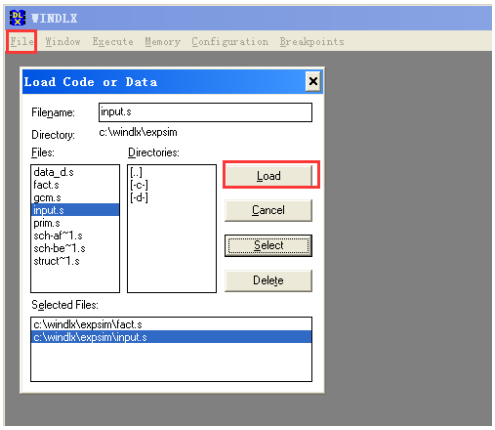
1.2 装载

选择 File / Load Code or Data，窗口中会列出目录中所有汇编程序。其中，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。

* 点击 fact.s
* 点击 select 按钮
* 点击 input.s
* 点击 select 按钮
* 点击 load 按钮

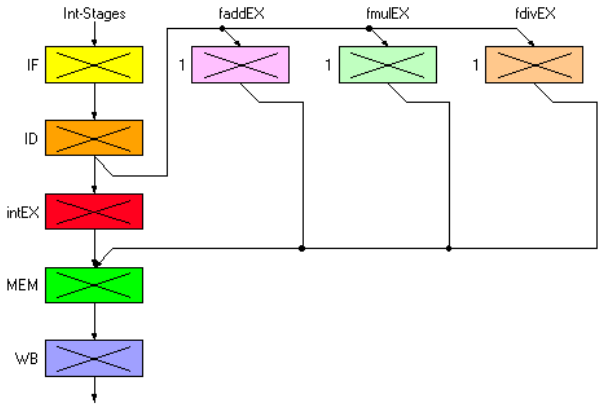
选择文件的顺序很关键，它决定了文件在存储器中出现的顺序。



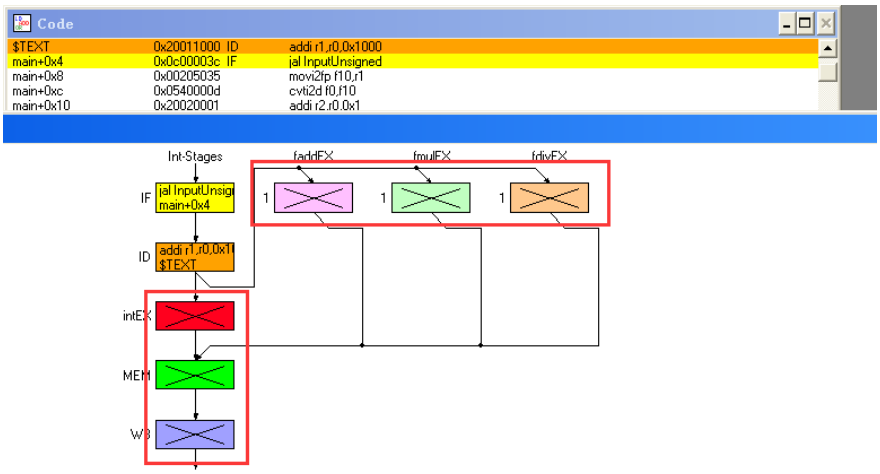
1. 3 模拟

首先打开 Pipeline 窗口查看五段流水线。

五段流水线：取指令周期(IF)->指令译码/读寄存器周期(ID)->执行/有效地址计算周期(EX)->存储器访问/分支完成周期(MEM)->写回周期(WB)。

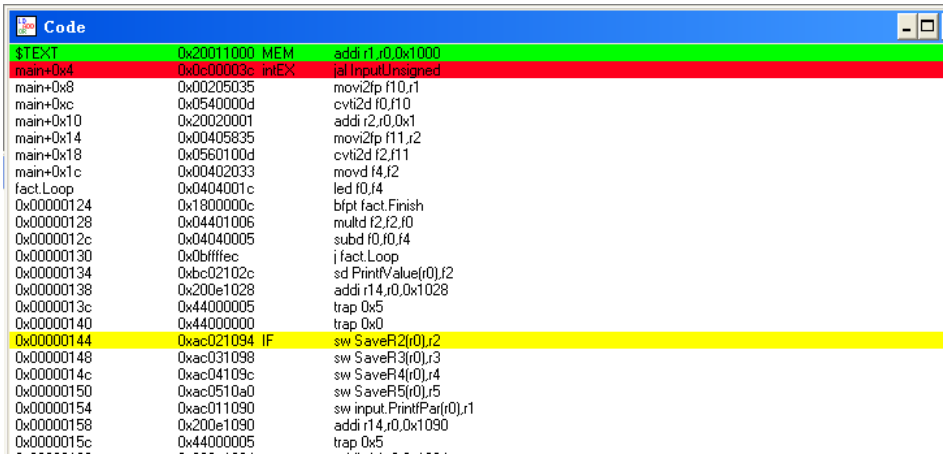


之后，打开 code 窗口，每次执行单个时钟，直到如下：

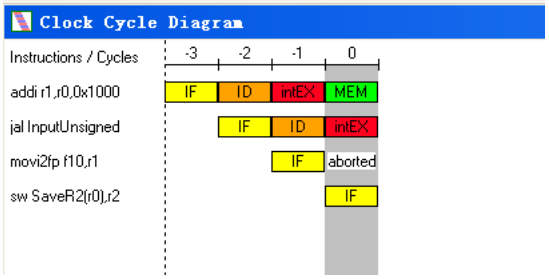


红色框内的方框中带有一个“X”，表示没有处理有效信息。

继续单步执行，直到如下情况：

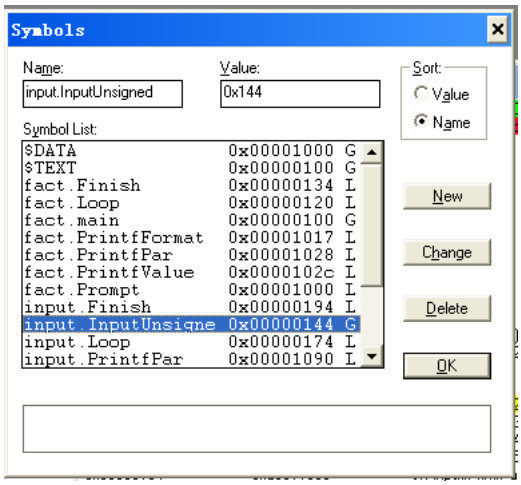


此时发现，IF，intEX 和 MEM 段正在使用而 ID 段没有。查看 Clock Cycle Diagram 窗口发现第三条指令知识为“aborted”，这是因为**第二条命令（jal）是无条件分支指令， 但只有在第三个时钟周期，jal 指令被译码后才知道，这时，下一条命令 movi2fp 已经取出，但需执行的下一条命令在另一个地址处，因而，movi2fp 的执行应被取消，在流水线中留下气泡。**

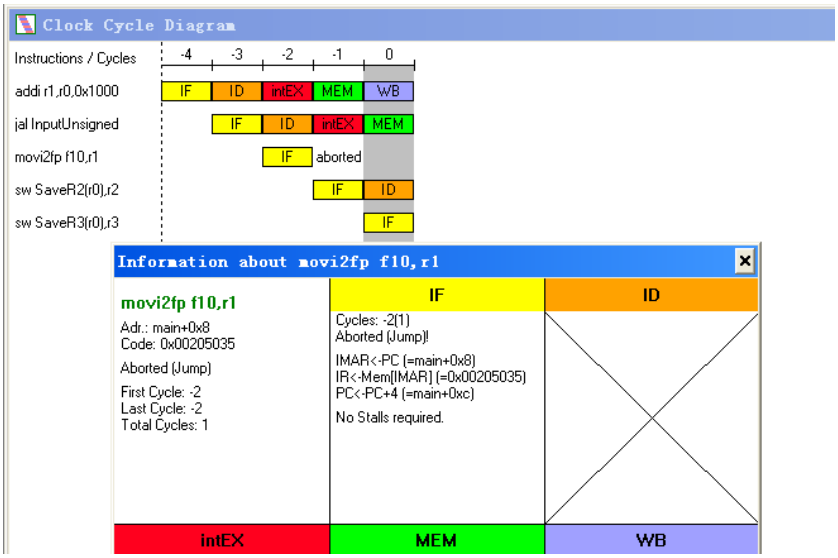


jal 的分支地址命名为“InputUnsigned”。为找到此符号地址的实际值，点击主窗口中的 Memory 和 Symbols，出现的子窗口中显示相应的符号和对应的实际值。在 “Sort”：区域选定“name”，使它们按名称排序，而不是按数值排序。数字后的“G”代表全局符号， “L”代表局部符号。

“input”中的“InputUnsigned”是一个全局符号，它的实际值为 0x144 ，用作地址。

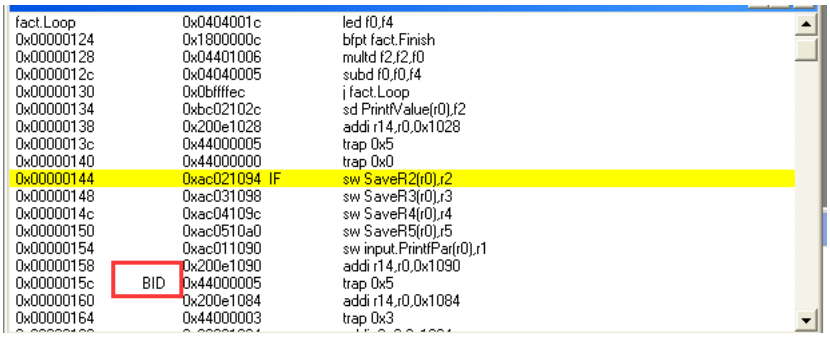


再一次点击 F7 ，第一条命令（addi）到达流水线的最后一段。对准 Clock cycle diagram 窗口中相应命令，双击第三行（movi2fp），在跳出的 Information 窗口中看到它只执行了第一段（IF）， 这是因为出现跳转而被取消。



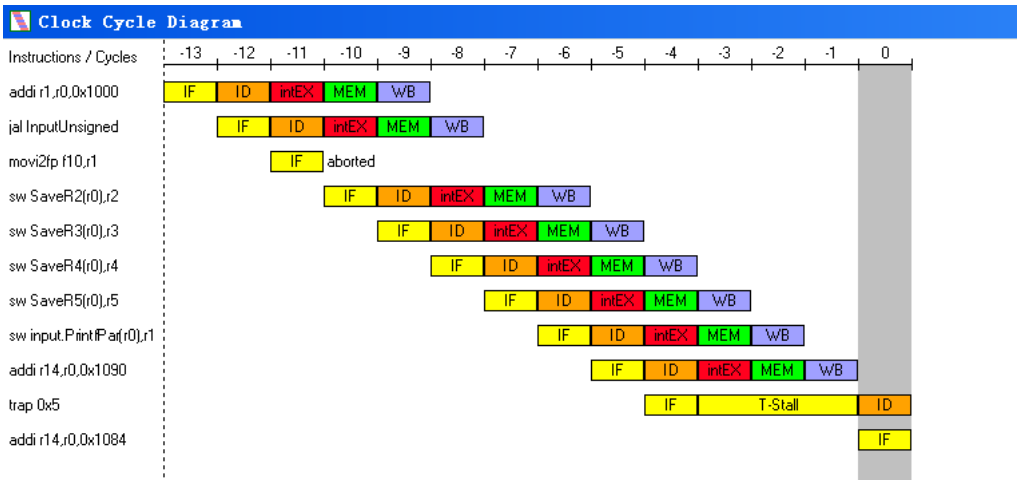
Code 窗口中包含命令 trap 0x5 的 0x0000015c 行，是写屏幕的系统调用命令。

单击命令行，然后点击主窗口菜单 Code，设置断点。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。

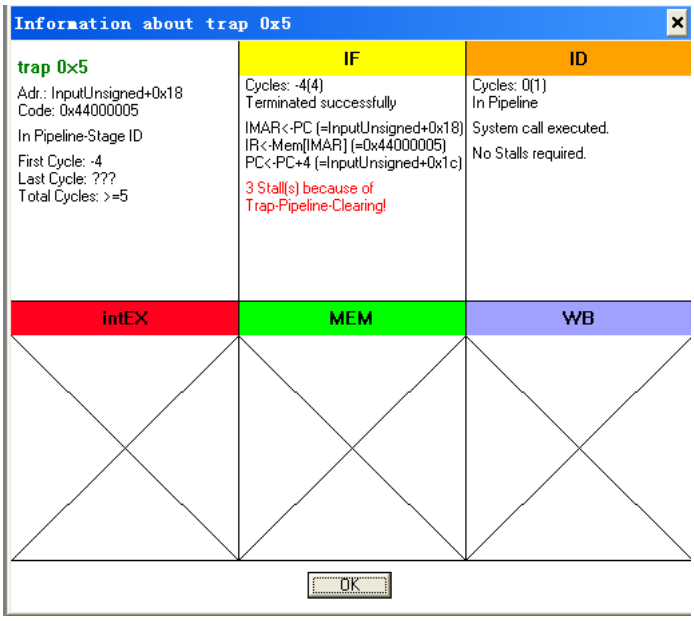


只要点击 Execution / Run 或按 F5，模拟就继续运行。

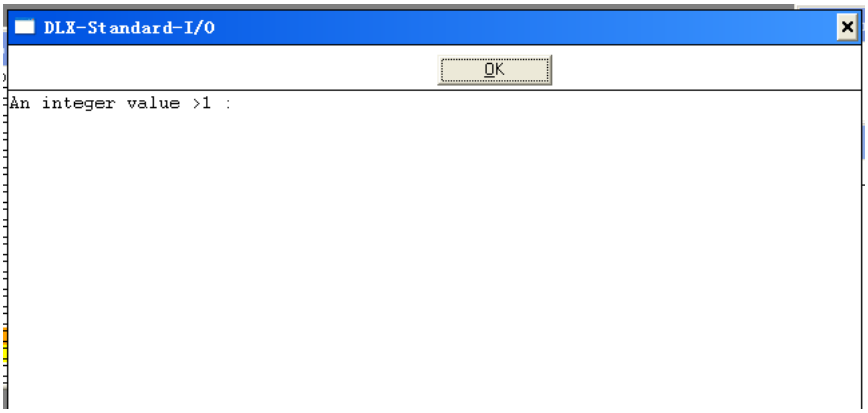
点击 Clock cycle diagram 窗口中的 trap 0x5 行，可以看到模拟正处于时钟周期 14。trap 0x5 行如下所示



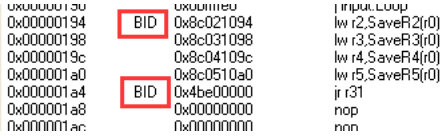
注意：无论何时遇到一条 trap 指令时，DLX 处理器中的流水线将被清空。在 Information 窗口（双击 trap 行弹出）中，在 IF 段显示消息“3 stall(s) because of Trap-Pipeline-Clearing!”。



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

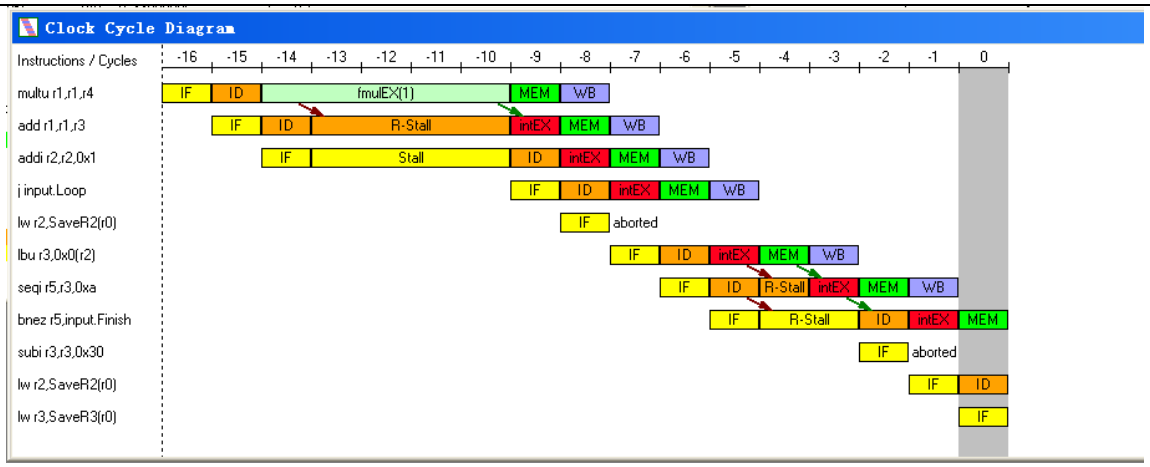


为进一步模拟，点击 Code 窗口，在地址为 0x00000194 的那一行（指令是 lw r2, SaveR2(r0)）设置一个断点。采用同样的方法，在地址 0x000001a4（指令 jar r31）处设置断点。



按 F5 继续运行。在跳出的窗口中键入 20，然后回车。

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。**



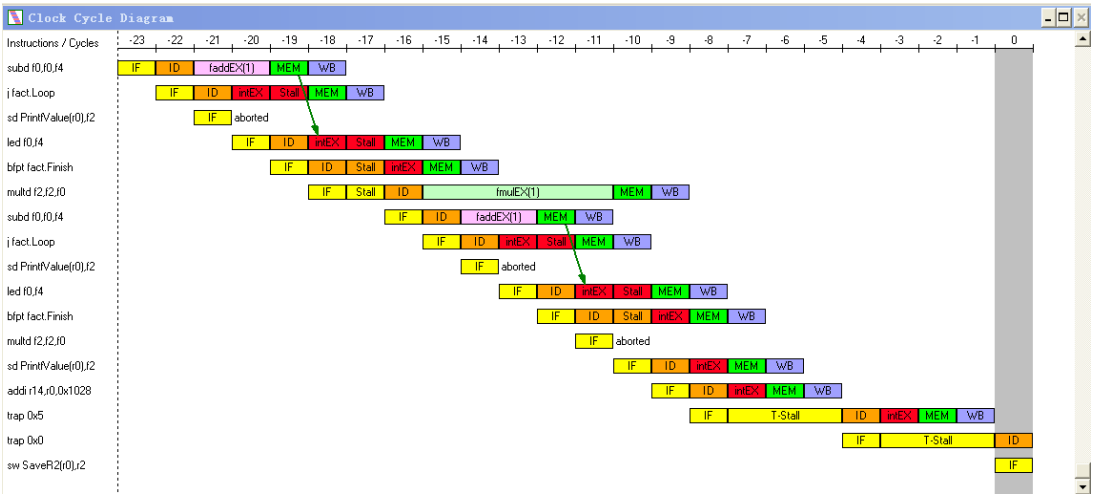
最后查看 Register 窗口。

Before：

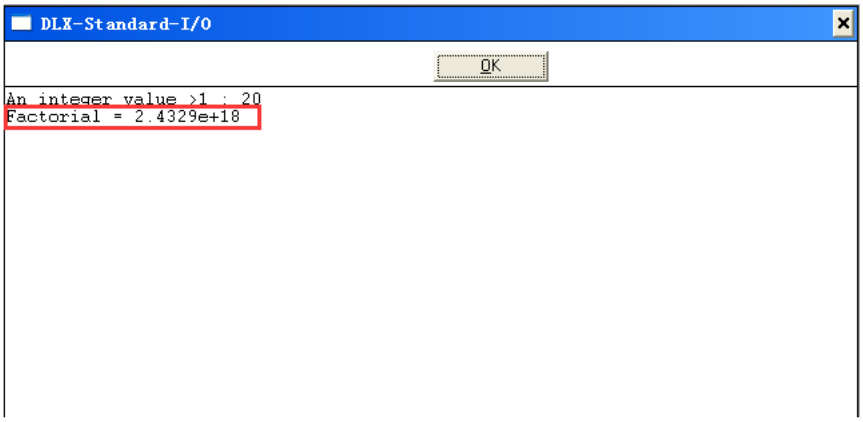
```
PC=    0x0000019c R9=    0x00000000 F2=              0 F27=
IMAR=  0x00000198 R10=   0x00000000 F3=              0 F28=
IR=    0x8c031098 R11=   0x00000000 F4=              0 F29=
A=     0x00000000 R12=   0x00000000 F5=              0 F30=
AHI=   0x00000000 R13=   0x00000000 F6=              0 F31=
B=     0x00000000 R14=   0x00001084 F7=              0 D0=
BHI=   0x00000000 R15=   0x00000000 F8=              0 D2=
BTA=   0x00000000 R16=   0x00000000 F9=              0 D4=
ALU=   0x00000000 R17=   0x00000000 F10=             0 D6=
ALUHI=0x00000000 R18=    0x00000000 F11=             0 D8=
FPSR=  0x00000000 R19=   0x00000000 F12=             0 D10=
DMAR=  0x00000000 R20=   0x00000000 F13=             0 D12=
SDR=   0x00000000 R21=   0x00000000 F14=             0 D14=
SDRHI=0x00000000 R22=    0x00000000 F15=             0 D16=
LDR=   0x00000000 R23=   0x00000000 F16=             0 D18=
LDRHI=0x00000000 R24=    0x00000000 F17=             0 D20=
R0=    0x00000000 R25=   0x00000000 F18=             0 D22=
R1=    0x00000014 R26=   0x00000000 F19=             0 D24=
R2=    0x00001036 R27=   0x00000000 F20=             0 D26=
R3=    0x0000000a R28=   0x00000000 F21=             0 D28=
R4=    0x0000000a R29=   0x00000000 F22=             0 D30=
R5=    0x00000001 R30=   0x00000000 F23=             0
R6=    0x00000000 R31=   0x00000108 F24=             0
R7=    0x00000000 F0=              0 F25=             0
R8=    0x00000000 F1=              0 F26=             0
```

After：

```
PC=    0x00000148 R9=    0x00000000 F2=       -0.18882
IMAR=  0x00000144 R10=   0x00000000 F3=        385.763
IR=    0xac021094 R11=   0x00000000 F4=              0
A=     0x00000000 R12=   0x00000000 F5=          1.875
AHI=   0x00000000 R13=   0x00000000 F6=              0
B=     0x00000000 R14=   0x00001028 F7=              0
BHI=   0x00000000 R15=   0x00000000 F8=              0
BTA=   0x00000000 R16=   0x00000000 F9=              0
ALU=   0x00000000 R17=   0x00000000
ALUHI=0x00000000 R18=    0x00000000
FPSR=  0x00000001 R19=   0x00000000
DMAR=  0x00000000 R20=   0x00000000
SDR=   0x00000000 R21=   0x00000000
SDRHI=0x00000000 R22=    0x00000000
LDR=   0x00000000 R23=   0x00000000
LDRHI=0x00000000 R24=    0x00000000
R0=    0x00000000 R25=   0x00000000
R1=    0x00000018 R26=   0x00000000
R2=    0x00000001 R27=   0x00000000
R3=    0x00000000 R28=   0x00000000
R4=    0x00000000 R29=   0x00000000
R5=    0x00000000 R30=   0x00000000
R6=    0x00000000 R31=   0x00000108
R7=    0x00000000 F0=              0
R8=    0x00000000 F1=          1.875
```



程序运行结束后，查看 I/O 界面，可以得到阶乘结果。



最后查看 Statistic 窗口。

开启定向：

```
Total:
    215 Cycle(s) executed.
    ID executed by 145 Instruction(s).
    2 Instruction(s) currently in Pipeline.

Hardware configuration:
    Memory size: 32768 Bytes
    faddEX-Stages: 1, required Cycles: 2
    fmulEX-Stages: 1, required Cycles: 5
    fdivEX-Stages: 1, required Cycles: 19
    Forwarding enabled.

Stalls:
    RAW stalls: 17 (7.91% of all Cycles), thereof:
        LD stalls: 3 (17.65% of RAW stalls)
        Branch/Jump stalls: 3 (17.65% of RAW stalls)
        Floating point stalls: 11 (64.70% of RAW stalls)
    WAW stalls: 0 (0.00% of all Cycles)
    Structural stalls: 0 (0.00% of all Cycles)
    Control stalls: 25 (11.63% of all Cycles)
    Trap stalls: 12 (5.58% of all Cycles)
    Total: 54 Stall(s) (25.12% of all Cycles)

Conditional Branches):
    Total: 23 (15.86% of all Instructions), thereof:
        taken: 2 (8.70% of all cond. Branches)
        not taken: 21 (91.30% of all cond. Branches)

Load-/Store-Instructions:
    Total: 13 (8.96% of all Instructions), thereof:
        Loads: 7 (53.85% of Load-/Store-Instructions)
        Stores: 6 (46.15% of Load-/Store-Instructions)

Floating point stage instructions:
    Total: 40 (27.60% of all Instructions), thereof:
        Additions: 19 (47.50% of Floating point stage inst.)
        Multiplications: 21 (52.50% of Floating point stage inst.)
        Divisions: 0 (0.00% of Floating point stage inst.)

Traps:
    Traps: 4 (2.76% of all Instructions)
```

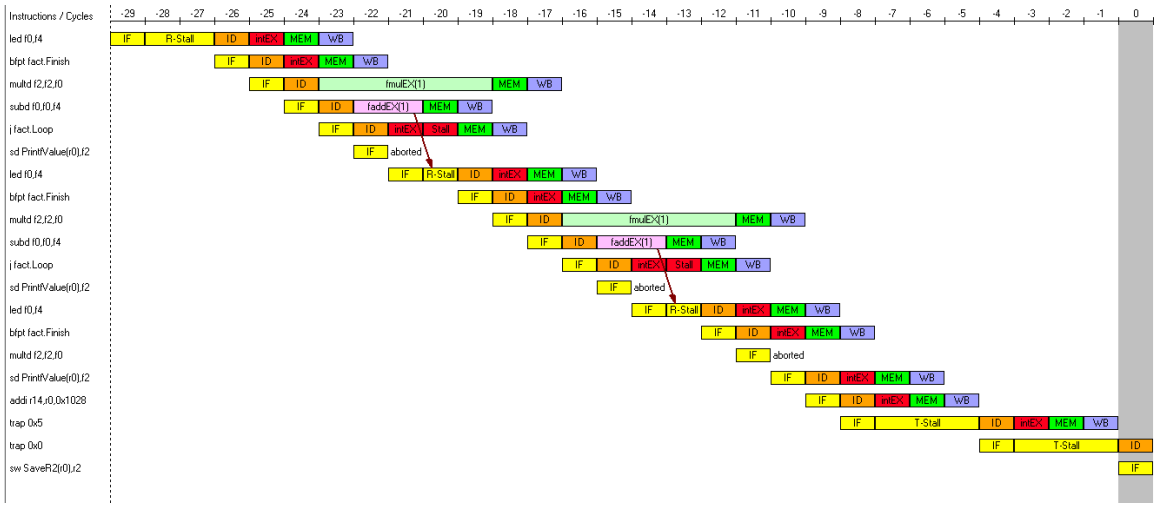
关闭定向：

```
Total:
    236 Cycle(s) executed.
    ID executed by 145 Instruction(s).
    2 Instruction(s) currently in Pipeline.

Hardware configuration:
    Memory size: 32768 Bytes
    faddEX-Stages: 1, required Cycles: 2
    fmulEX-Stages: 1, required Cycles: 5
    fdivEX-Stages: 1, required Cycles: 19
    Forwarding disabled.

Stalls:
    RAW stalls: 53 (22.46% of all Cycles)
    WAW stalls: 0 (0.00% of all Cycles)
    Structural stalls: 0 (0.00% of all Cycles)
    Control stalls: 25 (10.59% of all Cycles)
    Trap stalls: 12 (5.08% of all Cycles)
    Total: 90 Stall(s) (38.14% of all Cycles)

Conditional Branches):
    Total: 23 (15.86% of all Instructions), thereof:
        taken: 2 (8.70% of all cond. Branches)
        not taken: 21 (91.30% of all cond. Branches)

Load-/Store-Instructions:
    Total: 13 (8.96% of all Instructions), thereof:
        Loads: 7 (53.85% of Load-/Store-Instructions)
        Stores: 6 (46.15% of Load-/Store-Instructions)

Floating point stage instructions:
    Total: 40 (27.60% of all Instructions), thereof:
        Additions: 19 (47.50% of Floating point stage inst.)
        Multiplications: 21 (52.50% of Floating point stage inst.)
        Divisions: 0 (0.00% of Floating point stage inst.)

Traps:
    Traps: 4 (2.76% of all Instructions)
```

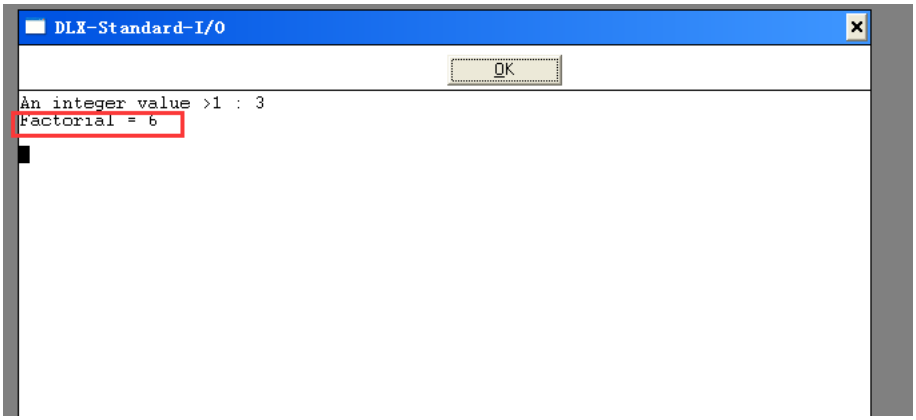
定向技术带来的加速比： 236 / 215 = 1.098

$DLX_{forwarded}$ 比 $DLX_{not\ forwarded}$ 快 9.8%。

2. 输入数据“3”采用单步执行方法。



得到结果 3!=6。



开启定向：

Total:
81 Cycle(s) executed.
ID executed by 52 Instruction(s).
2 Instruction(s) currently in Pipeline.

Hardware configuration:
Memory size: 32768 Bytes
faddEX-Stages: 1, required Cycles: 2
fmulEX-Stages: 1, required Cycles: 5
fdivEX-Stages: 1, required Cycles: 19
Forwarding enabled.

Stalls:
RAW stalls: 10 (12.34% of all Cycles), thereof:
LD stalls: 2 (20.00% of RAW stalls)
Branch/Jump stalls: 2 (20.00% of RAW stalls)
Floating point stalls: 6 (60.00% of RAW stalls)
WAW stalls: 0 (0.00% of all Cycles)
Structural stalls: 0 (0.00% of all Cycles)
Control stalls: 7 (8.64% of all Cycles)
Trap stalls: 12 (14.81% of all Cycles)
Total: 29 Stall(s) (35.80% of all Cycles)

Conditional Branches):
Total: 5 (9.62% of all Instructions), thereof:
taken: 2 (40.00% of all cond. Branches)
not taken: 3 (60.00% of all cond. Branches)

Load-/Store-Instructions:
Total: 12 (23.08% of all Instructions), thereof:
Loads: 6 (50.00% of Load-/Store-Instructions)
Stores: 6 (50.00% of Load-/Store-Instructions)

Floating point stage instructions:
Total: 5 (9.62% of all Instructions), thereof:
Additions: 2 (40.00% of Floating point stage inst.)
Multiplications: 3 (60.00% of Floating point stage inst.)
Divisions: 0 (0.00% of Floating point stage inst.)

Traps:
Traps: 4 (7.69% of all Instructions)

关闭定向：

Total:
98 Cycle(s) executed.
ID executed by 52 Instruction(s).
2 Instruction(s) currently in Pipeline.

Hardware configuration:
Memory size: 32768 Bytes
faddEX-Stages: 1, required Cycles: 2
fmulEX-Stages: 1, required Cycles: 5
fdivEX-Stages: 1, required Cycles: 19
Forwarding disabled.

Stalls:
RAW stalls: 26 (26.53% of all Cycles)
WAW stalls: 0 (0.00% of all Cycles)
Structural stalls: 0 (0.00% of all Cycles)
Control stalls: 7 (7.14% of all Cycles)
Trap stalls: 12 (12.24% of all Cycles)
Total: 45 Stall(s) (45.92% of all Cycles)

Conditional Branches):
Total: 5 (9.62% of all Instructions), thereof:
taken: 2 (40.00% of all cond. Branches)
not taken: 3 (60.00% of all cond. Branches)

Load-/Store-Instructions:
Total: 12 (23.08% of all Instructions), thereof:
Loads: 6 (50.00% of Load-/Store-Instructions)
Stores: 6 (50.00% of Load-/Store-Instructions)

Floating point stage instructions:
Total: 5 (9.62% of all Instructions), thereof:
Additions: 2 (40.00% of Floating point stage inst.)
Multiplications: 3 (60.00% of Floating point stage inst.)
Divisions: 0 (0.00% of Floating point stage inst.)

Traps:
Traps: 4 (7.69% of all Instructions)

定向技术带来的加速比： 98 / 81 = 1.210

结论分析与体会：

通过 WinDLX，可以从多个角度——代码角度、流水线时钟角度等——观察一个程序的执行，通过单步调试，可以对程序和计算机的理解更透彻。

同时，WinDLX 可以在多种配置下工作，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，使用方便。