

计算机系统结构

# 实验指导书

(试用稿)

编写：方娟 高明霞 陆帅冰

北京工业大学计算机学院

2021 年 11 月

# 目录

<b>第 1 章 WINDLX 安装及使用</b>	<b>3</b>
1.1 安装	3
1.2 使用示例	3
<b>第 2 章 WINMIPS64 安装及使用</b>	<b>9</b>
2.1 安装	9
2.2 使用示例	9
<b>第 3 章 SIMPLESCALAR 安装及使用</b>	<b>15</b>
3.1 SIMPLESCALAR 简介	15
3.2 SIMPLESCALAR 安装	15
3.3 SIMPLESCALAR 使用	21
<b>第 4 章 实验内容</b>	<b>24</b>
4.1 流水线中的相关 (WINDLX)	24
4.2 流水线中的相关 (WINMIPS64)	25
4.3 循环展开及指令调度	26
4.4 CACHE 性能分析	27
<b>第 5 章 实验要求</b>	<b>28</b>

# 第 1 章 WinDLX 安装及使用

WinDLX 处理器是 Hennessy 和 Patterson 合著的书《Computer Architecture - A Quantitative Approach》中流水线处理器的例子。WinDLX是一个基于Windows 的模拟器。WinDLX模拟器能够演示DLX 流水线如何工作。

## 1.1 安装


WinDLX 包含windlx.exe 和windlx.hlp 文件。同时，还需要一些扩展名为.s 的汇编代码文件。

1. 为 WinDLX创建目录，例如D:\WINDLX
2. 解压 WinDLX软件包或拷贝所有的WinDLX文件（至少包含windlx.exe, windlx.hlp, fact.s和input.s）到这个WinDLX目录。
3. 双击 windlx.exe 安装即可。

## 1.2 使用示例

使用 WinDLX 汇编器中的汇编文件fact.s，程序的基本功能是计算通过键盘输入的数的阶乘。需要用到文件input.s 接收从键盘输入的数。

### 1. 开始和配置WinDLX

和启动任何Windows应用程序一样，通过双击 WinDLX 图标启动WinDLX，将出现一个带有六个图标的主窗口，双击这些图标将弹出子窗口图1-1。

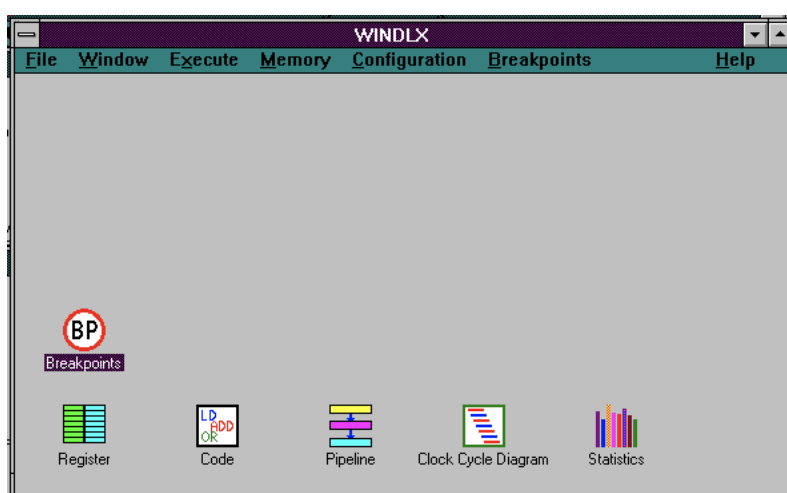


图 1-1 WinDLX 主窗口

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。可以通过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：

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

如果需要，可以通过点击相应区域来改变设置。然后，点击OK 返回主窗口。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。在 *Configuration* 菜单中的其他三个配置也可以设置， 它们是： *Symbolic addresses*, *Absolute Cycle Count* 和 *Enable Forwarding*。 点击相应菜单项后，在它的旁边将显示一个小钩。

## 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。 *fact.s* 是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。对话框中会显示信息“File(s) loaded successfully. Reset DLX?”，点击“是（Y）” 按钮确认。这样，文件就已被装入到存储器中了。现在可以开始模拟工作了。

## 3. 模拟

在主窗口中，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。

### (1) Pipeline 窗口

首先来看一下DLX处理器的内部结构。为此，双击图标*Pipeline*，出现一个子窗口，窗口

中用图表形式显示了DLX的五段流水线。尽可能地扩大此窗口，以便处于不同流水段的指令都能够在图表中显示，如图1-2所示。

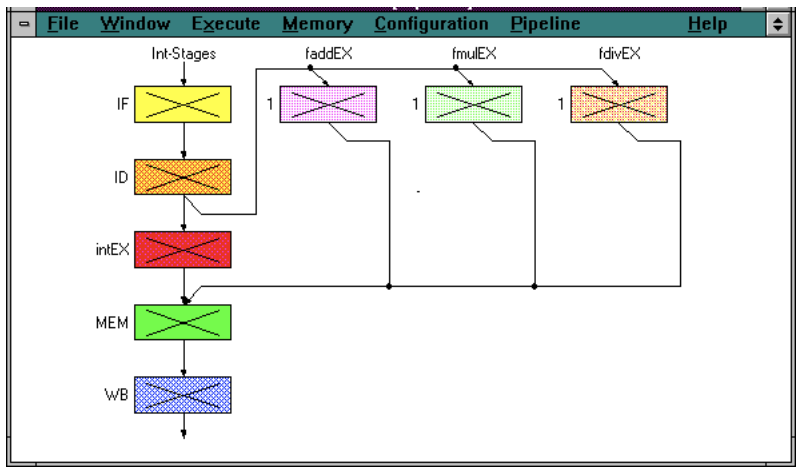


图1-2 DLX的五段流水线

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。

(2) Code 窗口

双击图标，将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。

\$TEXT	0x20011000	addi r1, r0, 0x1000
main+0x4	0x0c00003c	jal InputUnsigned

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，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。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,图形显示将变为：在代码窗口中，黄色出现在更下面的位置，并且可能是唯一彩色行。

(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，如图1-3所示。

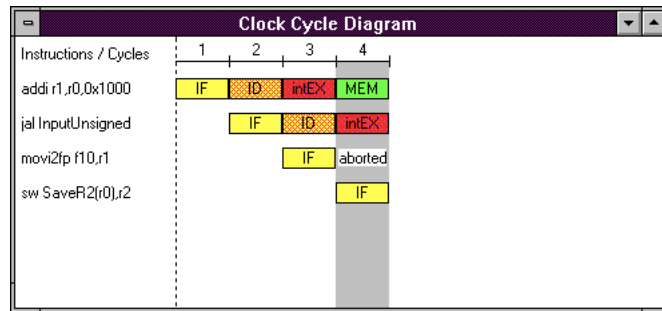


图1-3 流水线的时空图

在窗口中,将看到模拟正在第四时钟周期,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。第三条命令指示为"aborted",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。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。点击OK 按钮关闭窗口。再一次点击 F7,第一条命令(addi)到达流水线的最后一段。如果了解某条命令执行后处理器内部会发生什么?只要对准Clock cycle diagram窗口中相应命令所在行,然后双击它,弹出一个新窗口。窗口中会详细显示每一个流水段处理器内部的执行动作。这个窗口"Information about ..." 作为将来的Information 窗口。观察完后,点击OK按钮关闭窗口。双击第三行(movi2fp),会看到它只执行了第一段(IF),这是因为出现跳转而被取消。(双击Code窗口中的某一行或者Pipeline 窗口中的某一段,同样可以Information 窗口。)

#### (4) Breakpoint 窗口

当通过Code 窗口观察代码时(如果未打开,双击图标 Code),会看到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。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,如图1-4所示。通过此窗口,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。

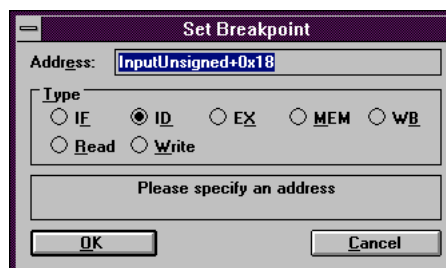


图 1-4 Set Breakpoint 窗口

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。如果想查看已定义的断点，只要单击图标Reapoints，将弹出一个窗口，其中显示了所有断点。重新使窗口图标化。

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"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，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!"。 (不要忘了按OK关闭窗口)。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。

#### (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。现在来看一下寄存器中的内容。为此，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。

如果希望不设置断点，而使模拟继续进行。办法是：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。

#### (6) Statistics 窗口

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)。

在静态窗口中可以比较一下不同配置对模拟的影响。现在我们看一看定向的作用。在前

面的模拟过程中，我们采用了定向。如果不采用定向，执行时间将会怎样呢？看一下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，会看到控制暂停和 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。利用这些值能够计算定向技术带来的加速比：

$$236 / 215 = 1.098$$

DLX forwarded比 DLX not forwarded 快9.8%。

上述过程是对WinDLX 的基本使用，可以对流水线和DLX 的操作类型有一定的了解。



## 第 2 章 WinMIPS64 安装及使用

WinMIPS64是一款基于WinDLX 处理器的指令集模拟器，WinDLX 处理器是Hennessy和Patterson 合著的书《Computer Architecture – A Quantitative Approach》中流水线处理器的例子，WinDLX 是一个基于Windows 的模拟器。WinMIPS64模拟器能够演示DLX 流水线如何工作。WinMIPS64与WinDLX在实验过程中可根据需求进行选择，两者选择一种即可。

### 2.1 安装

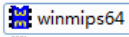
WinMIPS64包含winmips64.exe和一些扩展名为.s 的汇编代码文件。

1. 为 WinMIPS64创建目录，例如D:\winmips64
2. 解压WinMIPS64软件包或拷贝所有的WinMIPS64文件（至少包含winmips64.exe，sum.s和test\_for.s）到这个winmips64目录。
3. 双击 winmips64.exe 安装即可。

### 2.2 使用示例

使用WinMIPS64汇编器中的汇编文件sum.s，程序的基本功能是计算两个整数A, B之和。

#### 1. 开始和配置WinMIPS64

和启动任何Windows应用程序一样，通过双击 winmips64 图标  启动 WinMIPS64，将出现一个带有六个图标的主窗口，双击这些图标将弹出子窗口图2-1。

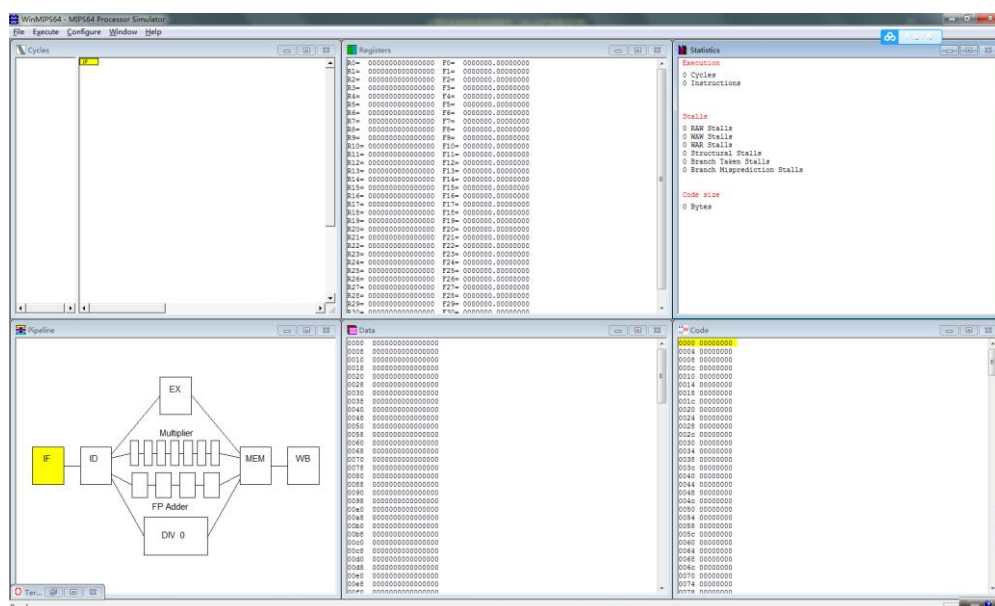


图 2-1 WinMIPS64 主窗口

为了初始化模拟器，点击File菜单中的Reset MIPS64菜单项。

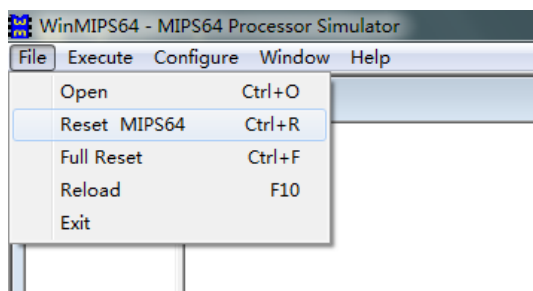


图2-2 WinMIPS64 Reset MIPS64窗口

WinMIPS64可以在多种配置下工作。可以通过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（点击Configuration打开菜单，然后点击Architecture菜单项）。

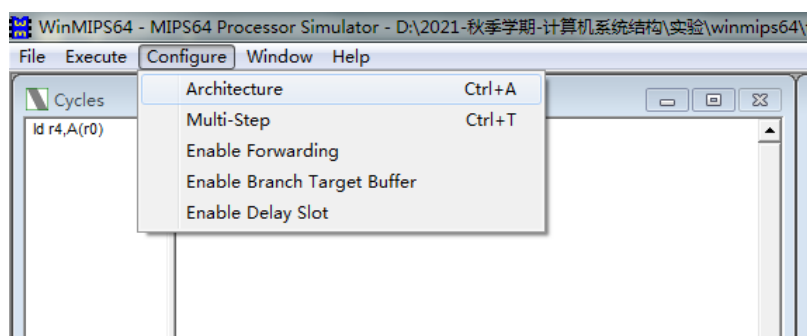


图2-3 WinMIPS64 Configuration窗口

WinMIPS64可以在多种配置下工作，通过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，具体配置参数说明：

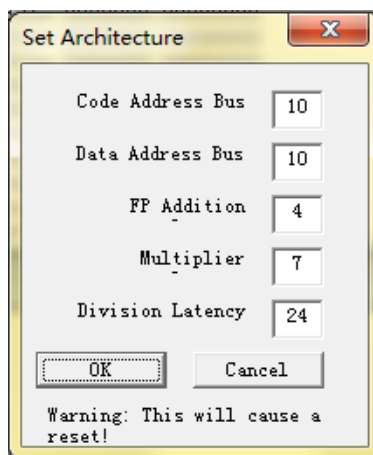


图2-4 WinMIPS64 Architecture参数设置

Code Address Bus: 代码地址总线，如果是10，则代码容量为 $2^{10}=1024$ bytes。

Data Address Bus: 数据地址总线；

FP Addition: 浮点加法延迟

Multiplier: 乘法延迟

Division Latency: 除法延迟

如果需要，可以通过点击相应区域来改变设置。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/Open，窗口中会列出目录中所有汇编程序。sum.s是计算一个加法就和程序；

```
.data
A: .word 10
B: .word 8
C: .word 0
.text
main:
ld r4,A(r0)
ld r5,B(r0)
dadd r3,r4,r5
sd r3,C(r0)
halt
```

点击Execute，下拉列表中包括Single Cycle和Multi Cycle，其中Single Cycle表示单步运行，Multi Cycle为多步运行。

## 3. 模拟

在主窗口中，可以看见六个图标，它们分别为“Register”，“Code”，“Pipeline”，“Clock Cycle Diagram”，“Statistics”和“Data”。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。

### (1) Pipeline 窗口

首先来看一下DLX处理器的内部结构。为此，双击图标Pipeline，出现一个子窗口，窗口中用图表形式显示了MIPS的五段流水线。尽可能地扩大此窗口，以便处于不同流水段的指令都能够在图表中显示，如图2-5所示。

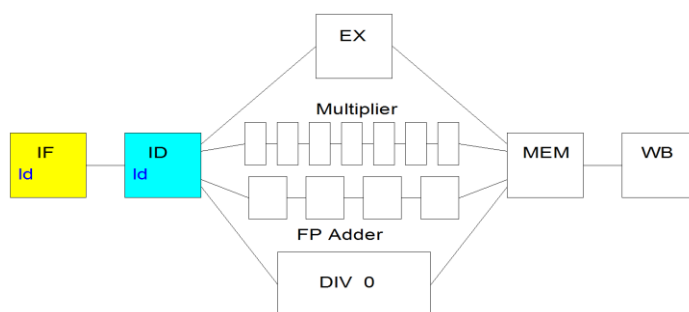


图2-5 MIPS五段流水线

## (2) Code 窗口

双击图标，将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。

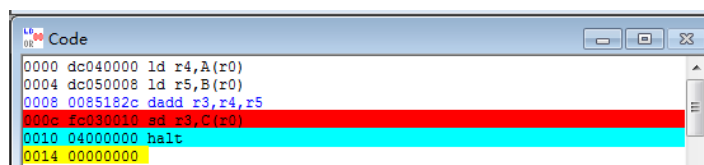


图 2-6 Code 窗口

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。

## (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，如图2-7所示。

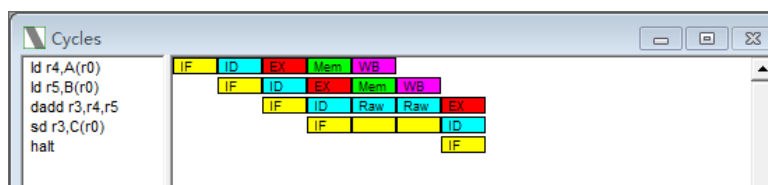


图 2-7 Clock Cycle Diagram 窗口

在窗口中，将看到第5条指令正在第一个IF段，第1条和第2条指令已经执行完毕，第3条指令由于与第2条指令出现相关。

当所有指令执行结束后，clock cycle diagram窗口显示流水线的执行过程。

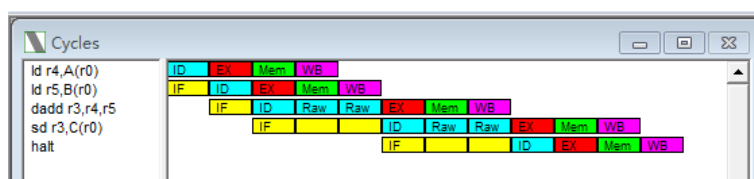


图2-8 流水线执行完成后窗口

## (4) Register窗口

现在来看一下寄存器中的内容，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。

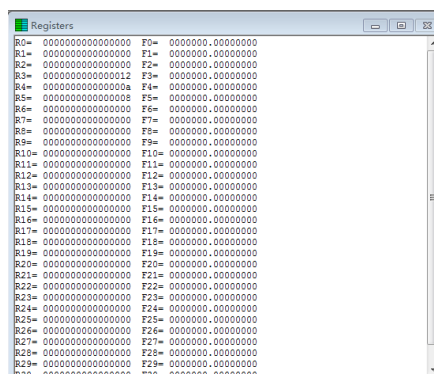


图2-9 Register窗口

按F7使模拟继续运行，有些值将发生改变。

#### (5) Data窗口

此窗口显示数据内存的内容，字节可寻址，但以64位块显示，适用于64位处理器。要编辑整数值，请双击鼠标左键。要以浮点数显示和编辑，请双击鼠标右键。

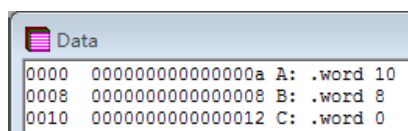


图2-10 Data窗口

#### (6) Statistics 窗口

按F4使程序完成执行，双击图标Statistics。Statistics窗口提供各个方面的信息：模拟中执行情况、出现stall情况、以及代码大小。

在静态窗口中可以比较一下不同配置对模拟的影响。现在我们看一看定向的作用。在前面的模拟过程中，如果我们采用了定向，如果采用定向技术，需要选择Enable Forwarding。

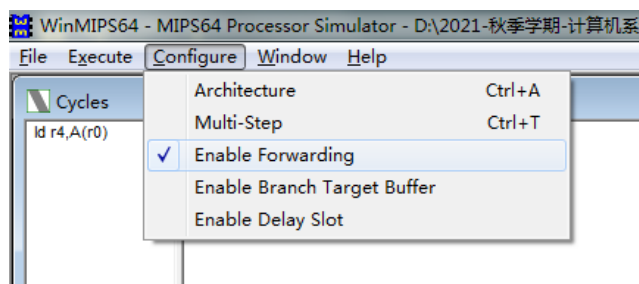
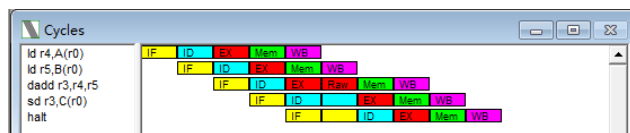
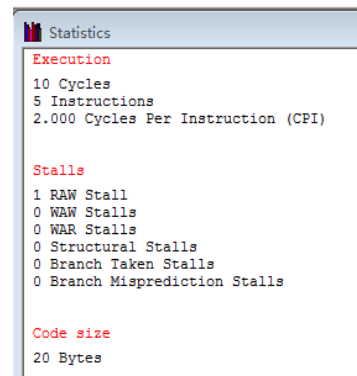


图2-11 选择采用定向技术

结果发现，同不采用定向技术相比，stall数有所减少。



(a)



(b)

图2-12 采用定向技术的流水线及统计结果

如果不采用定向，执行时间将会怎样呢？看一下Statistics窗口中的各种统计数字：

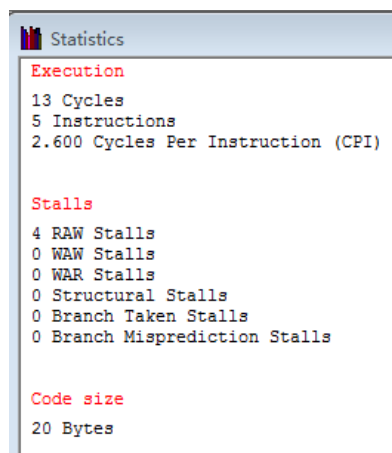


图2-13 不采用定向技术的流水线及统计结果

上述过程是对 WinMIPS64 的基本使用，可以对流水线的操作类型有一定的了解。

## 第 3 章 SimpleScalar 安装及使用

### 3.1 SimpleScalar 简介

SimpleScalar 是为计算机系统性能及功耗分析、处理器微体系结构建模、软硬件协同验证提供有效支持的工具集。它包括一系列模拟器如表1 所示，既有简单的功能模拟器Sim-safe也有复杂的sim-outorder 性能模拟器。SimpleScalar 目前支持PISA,ARM,X86 等指令集，可以运行在大多数类UNIX 平台。

表 1 SimpleScalar 包括的模拟器

模拟器	功能	特点
sim-fast	模拟计算机系统结构，速度最快	不进行指令的错误检验
sim-safe	模拟计算机系统结构	进行指令错误检验
sim-eio	简单模拟结构	进行指令检查
sim-profile	提供一些数据可以对程序进行简要分析	对程序进行简要分析
sim-cache	可以产生多级 cache 数据分析	不产生时序信息
sim-cheetah	一次运行程序代码时模拟多种 cache 配置	
sim-bpred	实现了分支预测	
sim-outorder	有两级存储器系统并支持预测执行的非常详细的乱序发射的超标量处理器	跟踪所有流水线延时的性能模拟器

### 3.2 SimpleScalar 安装

目前的 SimpleScalar 版本只能安装运行在大多数类UNIX 平台下，下边的具体安装过程以vmware（vmware workstation 6）和红帽linux 操作系统为SimpleScalar 运行环境。各安装步骤涉及到的资源统一放置在：simplescalar 实验所需资源目录下。

第一步：安装 vmware 工作站

直接运行 vmware 文件夹下的VMware-workstation-6.0.0-45731.exe，然后按照提示安装，遇到需要选择的，如果自己沒有更好选择就采用默认设置。

第二步：安装 Linux 虚拟机

在 G 盘根目录下建立simples 子目录，将simplescalar 实验所需资源/gaomxvm.rar 解压到G:\simples 目录下，形成目录G:\simples\gaomxvm

注：如无G 盘，可解压在其它驱动器中，并按（2）进行；以下文档均以G 盘此目

录为例说明。

第三步：将虚拟机添加到 vmware 工作站

打开 vmware 工作站使用工具条中 file/Open 将 G:\simples\gaomxvm\ Red Hat Linux.vmx 添加到工作站. 图 3-1 是添加成功后 vmware 中 Linux 虚拟机 gaomxLinux 的界面。

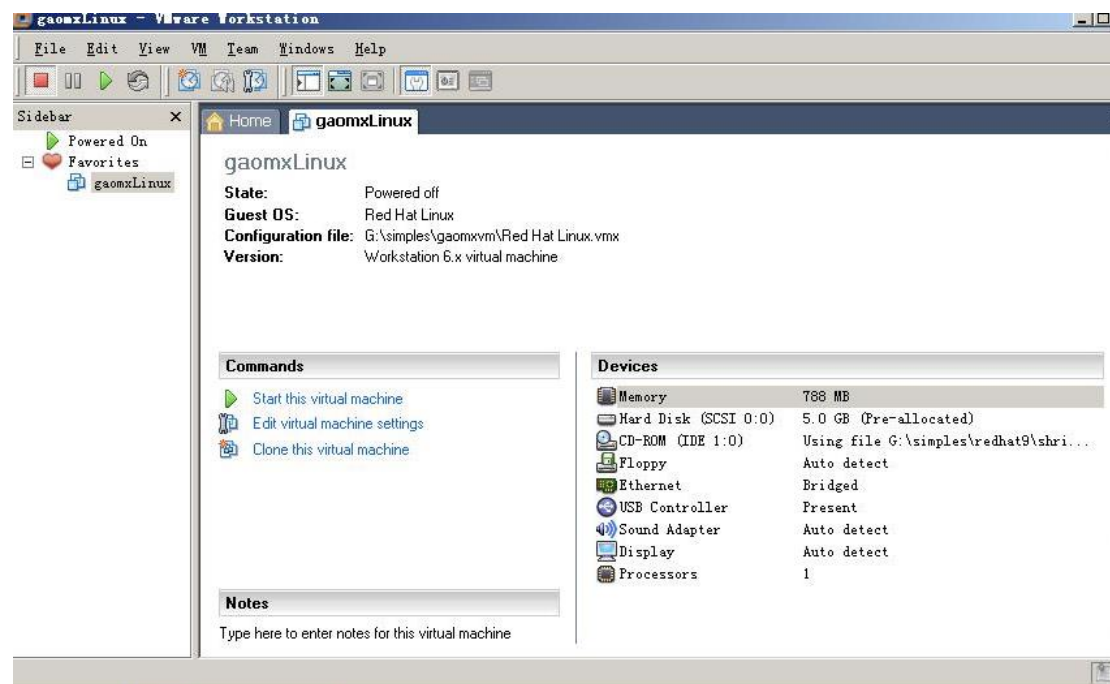


图 3-1 vmware 工作站中Linux 虚拟机gaomxLinux

第四步：创建主机和虚拟机之间的共享目录

为了方便大家在主机操作系统比如：WindowsXP，和虚拟机操作系统Linux之间互传文件，必须在两者之间建立一个共享目录。根据虚拟机所在目录不同有两种方式：

(1) 严格按照上述各步骤，将虚拟机解压到G:\simples\gaomxvm\目录下：

首先选择工作站工具条上VM菜单下的settings选项，如图3-2 所示。



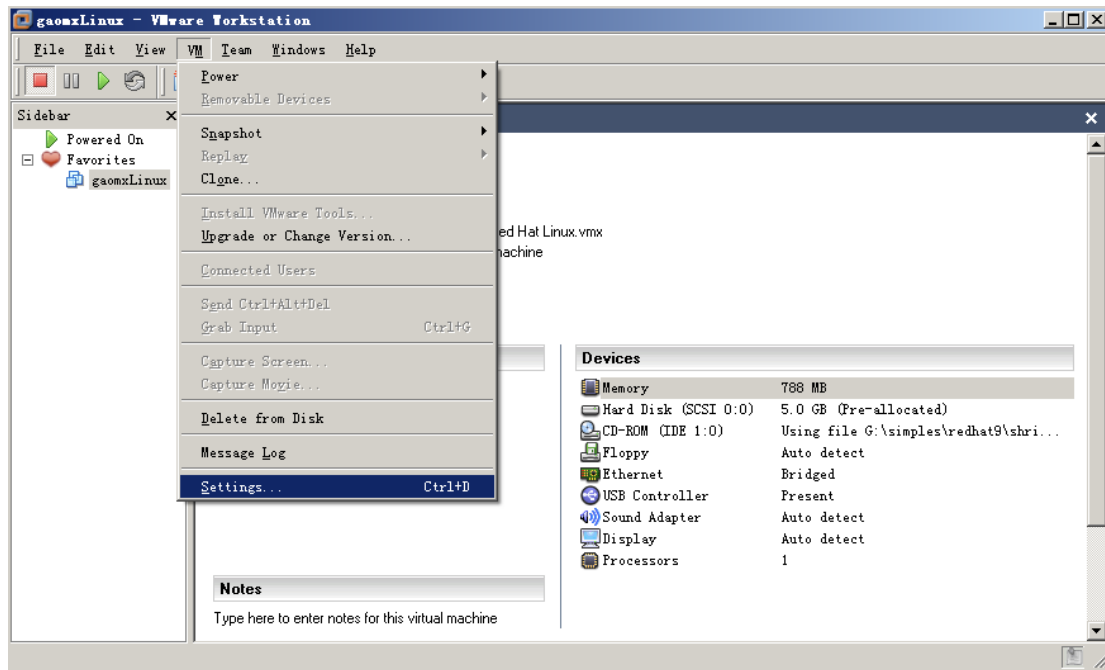


图3-2 设置共享目录过程

选中后弹出一个名为“Virtual Machine Settings”的浮动窗口如图3-3，选择选项卡 options，然后选择“Sharefolders”条目，选项卡左边就是具体设置项。如果Forders中存在一个默认设置，选中该设置，将其设置为“Always enable”。

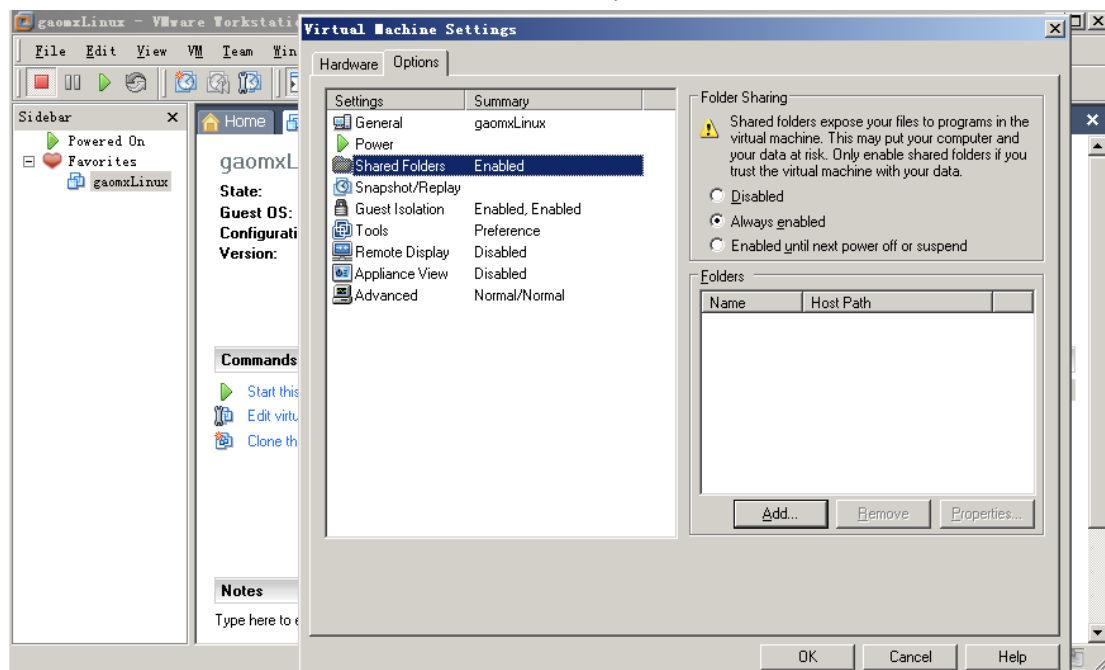


图 3-3 共享文件夹设置

如果 Folders 中不存在设置如图 3-3，选择左下按钮 Add，弹出添加向导，如图 2-4。



图 3-4 添加向导

根据向导指示，选择 Next 转到具体设置页面。在 Name 下设置在虚拟机下可以看见的目录名，此处为 gaomxsimples，在 Host folder 下设置 G:\simples 目录如图 3-5。再 Next，选择共享方式如图 3-6。完成设置返回到选项卡 options，此时 Folders 中已经存在一个刚才的设置，如图 3-7，选中该设置，将其设置为“Always enable”。

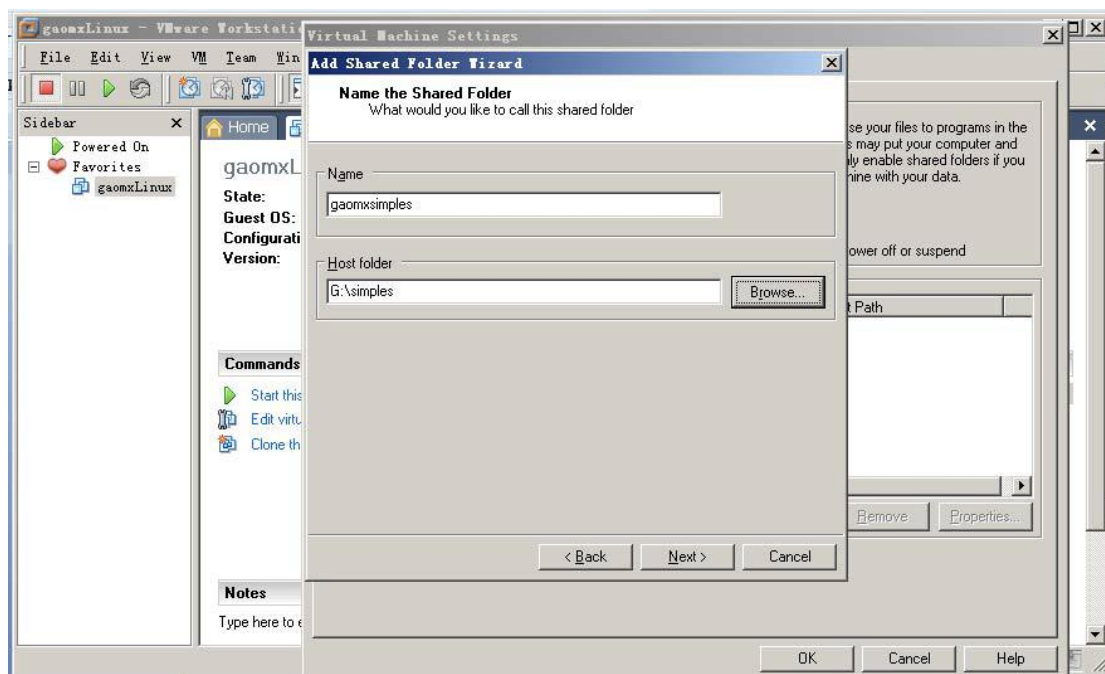


图 3-5 设置具体目录名

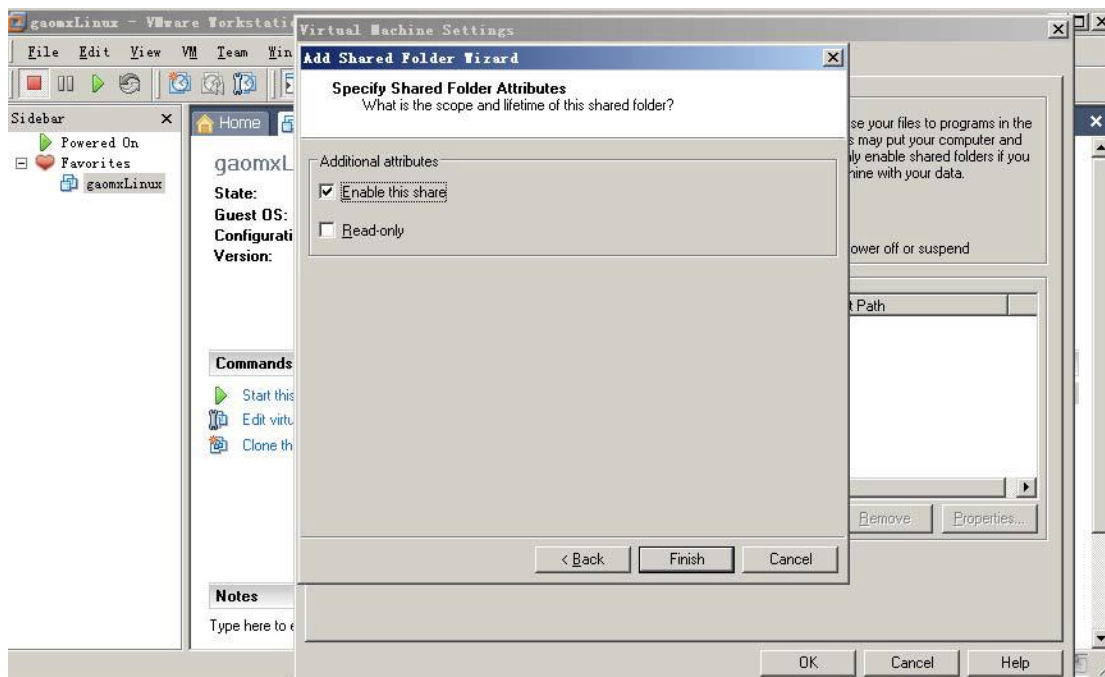


图 3-6 设置共享方式

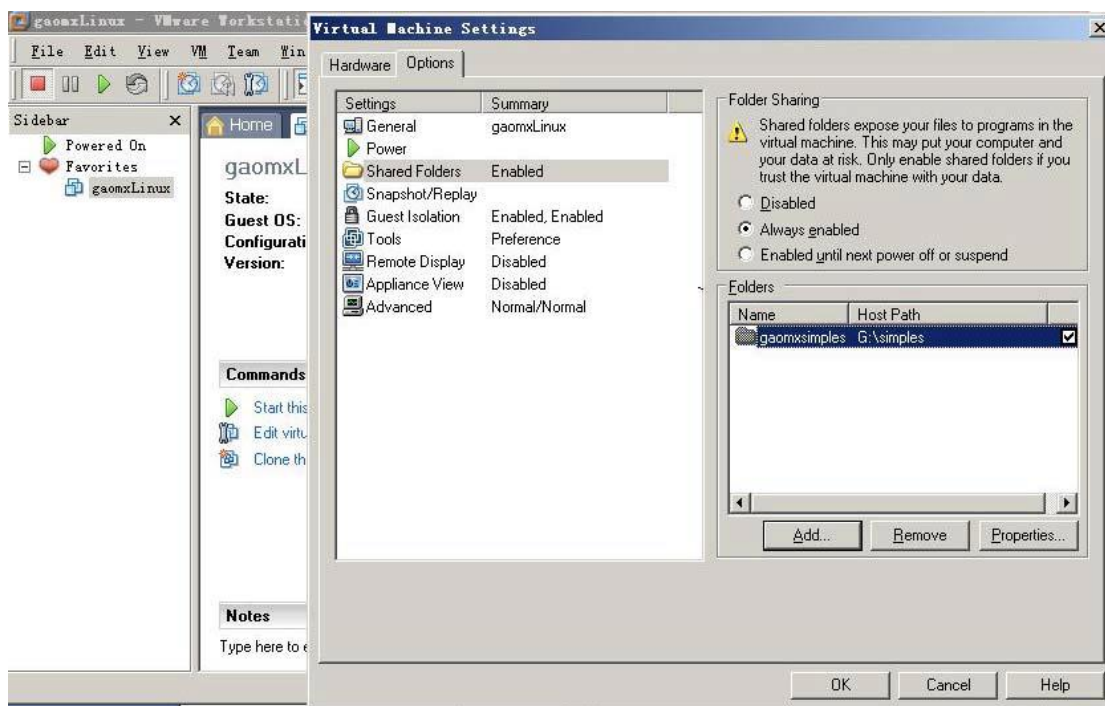


图 3-7 共享文件夹添加完成

(2) 虚拟机没有按照上述步骤解压，没有目录对照

这种情况下，需要重新在主机和虚拟中建立共享目录，具体步骤请参考 vmware 目录下文档“在 windows 下与 linux 虚拟机进行文件共享.txt”中介绍方法。

第五步：启动 gaomxvm 虚拟机

点击 gaomxvm 虚拟机中启动选项，启动该虚拟机。此过程可能需要几分钟，请耐心等待。图 2-8 是虚拟机启动后的界面。可以通过 user: root, password:123456 登录该

虚拟机。

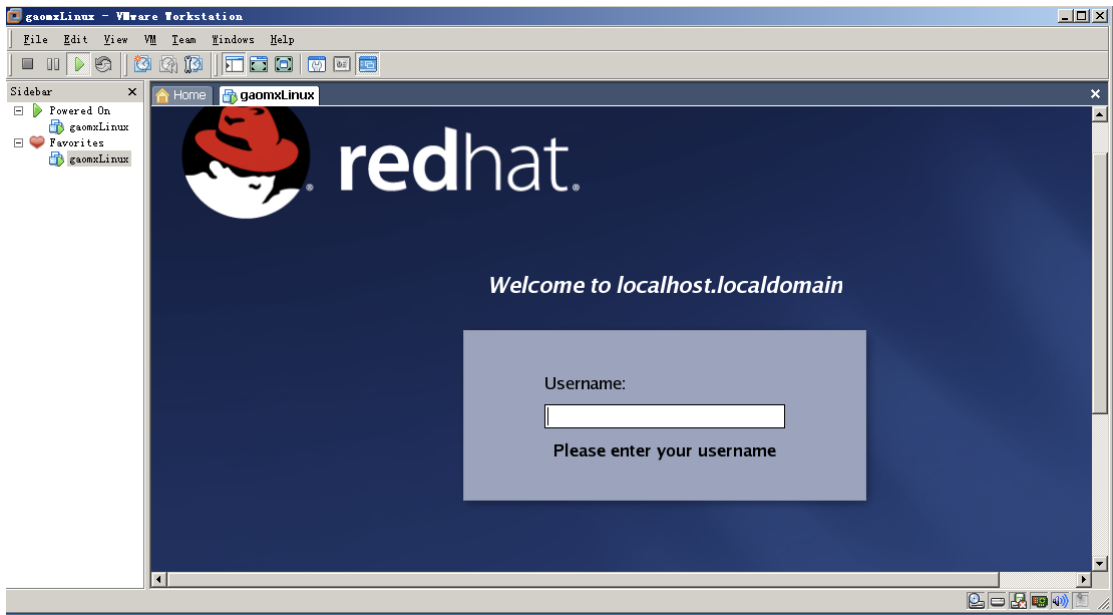


图 3-8 虚拟机启动

第六步：进入 SimpleScalar 实验环境

通过点击右键，选择 New Terminal 进入控制台环境，可以在目录 /root/下找到 simplescalar目录，这是一个SimpleScalar 模拟器的总目录。

第七步：验证共享目录是否存在

在控制台下 root/simplescalar/目录下，通过命令：`cd /mnt/hgfs` 看是否能进入该目录，并查看是否存在gaomxsimples 共享目录如图3-9，如果存在，说明第四步成功的添加了共享目录条目，否则需要通过第四步重新添加。

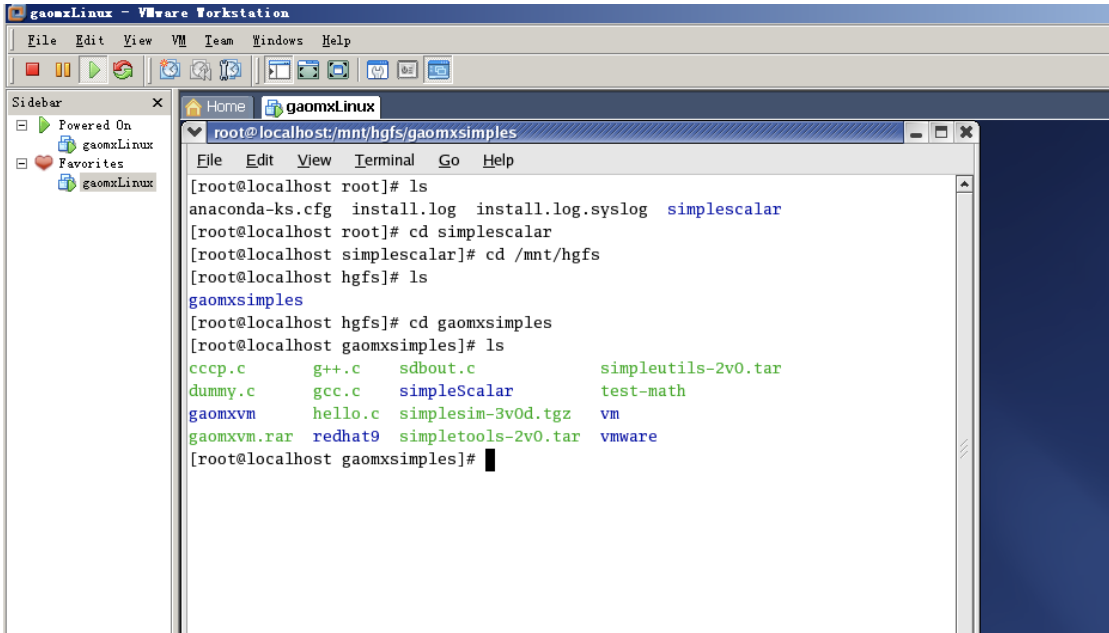


图 3-9 测试共享目录是否成功添加

### 3.3 SimpleScalar 使用

现代微机系统结构的另一重要技术是Cache。但是Cache一般位于CPU内部，即使是对汇编语言程序员也是不可见的。为了直观的建立Cache技术的各种概念，形象的学习甚至于自己动手进行Cache性能分析，设计一系列有针对性的仿真实验是个很好的教学方法。SimpleScalar工具集中有专门针对Cache技术的模拟器sim-cache和sim-cheetah，正是完成这些仿真实验的理想平台。借助这两个工具，我们在系统结构课程中增设了Cache性能分析的系列仿真实验帮助学生更好的理解和掌握Cache技术。下面以基本配置情况下对Cache失效情况的统计分析为例，说明进行Cache技术仿真实验中SimpleScalar的应用。

要使用 SimpleScalar 完成这个统计分析，需要以下三个重要步骤。第一个步骤是要配置实验环境。SimpleScalar 目前只能运行于类UNIX 平台，为了设置实验环境，我们使用了vmware（vmware workstation 5）和红帽linux 操作系统作为SimpleScalar 运行环境。第二个步骤是选择或编制测试程序。我们可以直接使用SimpleScalar 网站上Benchmarks 提供的一些经过编译的二进制测试程序，也可以自己使用某种程序语言，例如C 或汇编，编制一些典型功能的测试源程序，这些源程序通过SimpleScalar 提供的编译器编译成二进制后，模拟器就可以直接使用了。第三个步骤就是使用sim-cache，并根据要求设置其运行参数来模拟这些测试程序。经过这些步骤后，就能获得模拟结果。图2-10 是模拟得到的一部分统计数据，这个例子中使用的是用C 编写的打印“HelloWorld my name is gaomx！”的测试程序。

```
sim: ** starting functional simulation w/ caches **
Hello World my name is gaomx!

sim: ** simulation statistics **
sim_num_insn      7542 # total number of instructions executed
sim_num_refs      4107 # total number of loads and stores executed
sim_elapsed_time   1 # total simulation time in seconds
sim_inst_rate     7542.0000 # simulation speed (in insts/sec)
ill.accesses      7542 # total number of accesses
ill.hits          6999 # total number of hits
ill.misses        543 # total number of misses
ill.replacements  305 # total number of replacements
ill.writebacks    0 # total number of writebacks
ill.invalidations 0 # total number of invalidations
ill.miss_rate     0.0720 # miss rate (i.e., misses/ref)
ill.repl_rate     0.0404 # replacement rate (i.e., repls/ref)
ill.wb_rate       0.0000 # writeback rate (i.e., wrbks/ref)
ill.inv_rate      0.0000 # invalidation rate (i.e., invs/ref)
dll.accesses      4207 # total number of accesses
dll.hits          3749 # total number of hits
dll.misses        458 # total number of misses
```

图 3-10 sim-cache 统计数据实例

下面是上述实例运行的具体步骤：

第一步：在主机下编制 C 程序

我们编制一个简单的 C 语言程序如下，保存为 G:\simples\hello.c

```
#include <stdio.h>

main()
{
    printf("Hello World my name is gaomx!\n");
    return 0;
}
```

第二步：将测试程序传送到虚拟机下

在虚拟机控制台环境下 root 目录下键入命令：cd /mnt/hgfs/gaomxsimples，进入共享目录，查看刚刚保存的文件是否存在。如果存在，通过拷贝命令 cp hello.c/root/simplescalar/ 将其拷贝到/root/simplescalar/ 目录下。返回到/root/simplescalar/ 目录下，使用命令：./bin/sslittle-na-sstrix-gcc hello.c 编译该文件在/root/simplescalar/ 目录下生成a.out 文件用于测试，具体命令和控制台输出，如图3-11 所示。可以通过-o，例如：./bin/sslittle-na-sstrix-gcc hello.c -o hello.out 参数指定你要生成的文件名，而不是默认a.out。

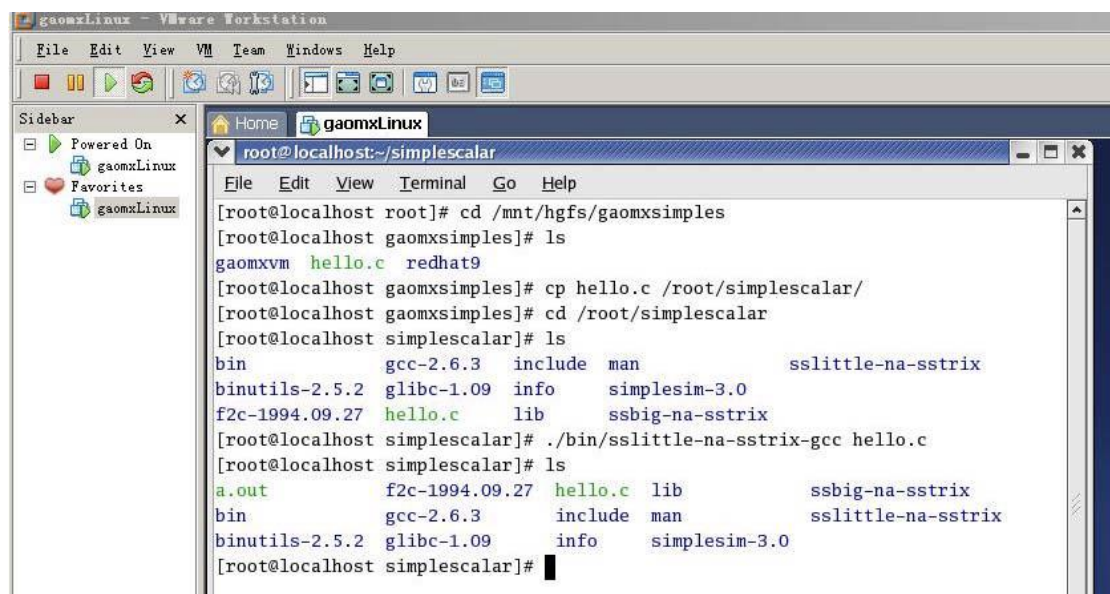
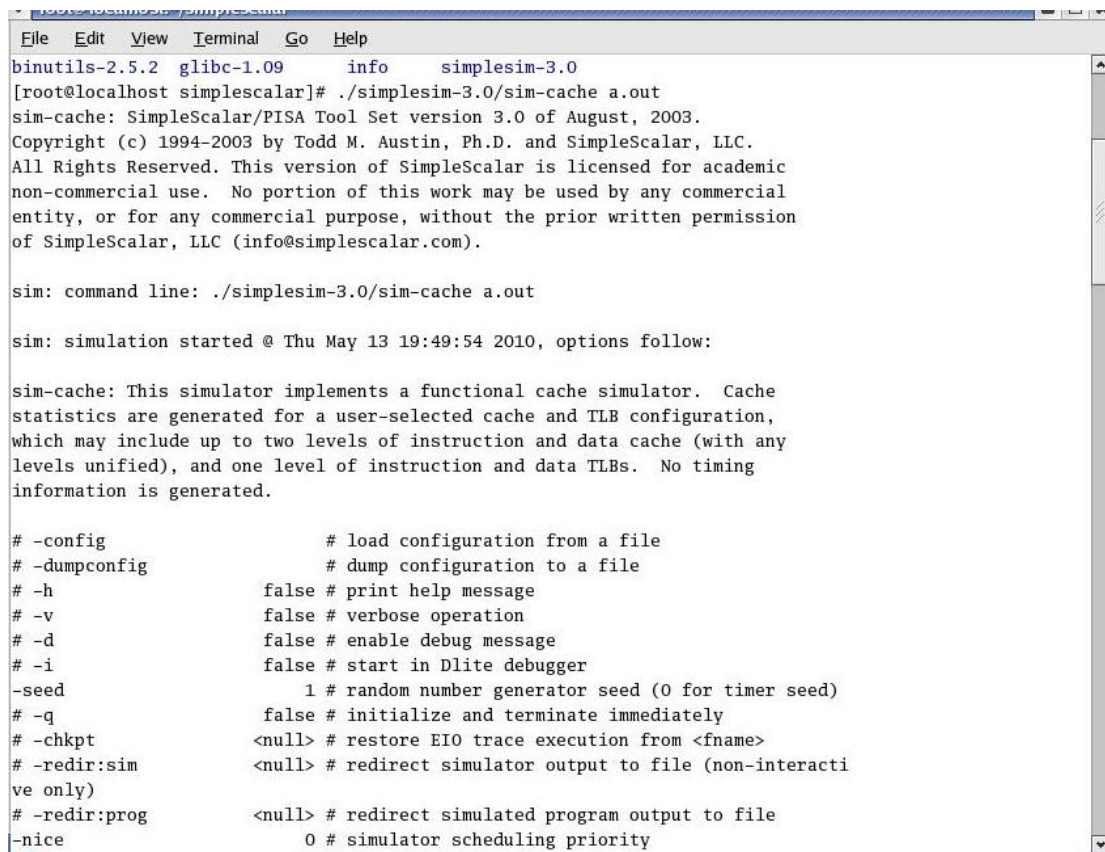


图 3-11 传送文件过程

第三步：使用 sim-cache 模拟该测试程序

此时，在/root/simplescalar/目录下，通过命令：./simplesim-3.0/sim-cache a.out 就可以对测试程序进行模拟统计，图3-12 是测试后生成的具体统计数据。因为没有设置任何参数，所以生成的统计结果从控制台输出，这些数据可以通过右键的拷贝操作直接提出到主机，生成对应文件，供我们随时分析使用。





```
File Edit View Terminal Go Help
binutils-2.5.2 glibc-1.09 info simplesim-3.0
[root@localhost simplescalar]# ./simplesim-3.0/sim-cache a.out
sim-cache: SimpleScalar/PISA Tool Set version 3.0 of August, 2003.
Copyright (c) 1994-2003 by Todd M. Austin, Ph.D. and SimpleScalar, LLC.
All Rights Reserved. This version of SimpleScalar is licensed for academic
non-commercial use. No portion of this work may be used by any commercial
entity, or for any commercial purpose, without the prior written permission
of SimpleScalar, LLC (info@simplescalar.com).

sim: command line: ./simplesim-3.0/sim-cache a.out

sim: simulation started @ Thu May 13 19:49:54 2010, options follow:

sim-cache: This simulator implements a functional cache simulator. Cache
statistics are generated for a user-selected cache and TLB configuration,
which may include up to two levels of instruction and data cache (with any
levels unified), and one level of instruction and data TLBs. No timing
information is generated.

# -config                # load configuration from a file
# -dumpconfig            # dump configuration to a file
# -h                    false # print help message
# -v                    false # verbose operation
# -d                    false # enable debug message
# -i                    false # start in Dlite debugger
-seed                    1 # random number generator seed (0 for timer seed)
# -q                    false # initialize and terminate immediately
# -chkpt                <null> # restore EIO trace execution from <fname>
# -redir:sim            <null> # redirect simulator output to file (non-interacti
ve only)
# -redir:prog          <null> # redirect simulated program output to file
-nice                    0 # simulator scheduling priority
```

图 3-12 模拟统计结果（只显示了部分）

关于 sim-cache 的部分命令参数的说明：

-cache:dl1 <config>//配置一级数据cache

-cache:dl2 <config>//配置二级数据cache

-cache:il1 <config>//配置一级指令cache

-cache:il2 <config>//配置二级指令cache

-tlb:dtlb <config>//配置数据TLB

-tlb:itlb <config>//配置指令TLB

-flush <boolean>//系统调用是否刷新

在配置 Cache 时，<config>的格式如下：

<名字>:<cache 的组数>:<块大小>:<组内块数>:<更新策略(l,f,r)>如：

-cache:dl1 dl1:256:32:4:l

表示，将一级数据cache 配置成：256 组，每组4 块，每块32 字节，使用的是L R U 替换策略。

## 第4章 实验内容

本课程的实验主要是对计算机系统结构理论知识的补充,加强学生对计算机的流水线技术和Cache 的理解。在掌握基本原理的基础上,通过使用WinDLX/WinMIPS64模拟器和SimpleScalar模拟器对流水线处理、指令调度和Cache 性能进行测试。本实验教学的目的是通过具有针对性的实验,使学生对计算机系统结构的流水线、运行流程以及提高计算机性能的经典方法有比较清晰和深入的认识,并为后续课程的学习打下良好的基础。

### 4.1 流水线中的相关 (WinDLX)

实验目的:

1. 熟练掌握WinDLX 模拟器的操作和使用,熟悉DLX 指令集结构及其特点;
2. 加深对计算机流水线基本概念的理解;
3. 进一步了解DLX 基本流水线各段的功能以及基本操作;
4. 加深对数据相关、结构相关的理解,了解这两类相关对CPU 性能的影响;
5. 了解解决数据相关的方法,掌握如何使用定向技术来减少数据相关带来的暂停。

实验平台:

WinDLX 模拟器

实验内容和步骤:

- 1.用WinDLX 模拟器执行下列三个程序:

求阶乘程序 fact.s

求最大公倍数程序 gcm.s

求素数程序 prim.s

分别以步进、连续、设置断点的方式运行程序,观察程序在流水线中的执行情况,观察CPU 中寄存器和存储器的内容。熟练掌握WinDLX 的操作和使用。

2. 用WinDLX 运行程序structure\_d.s,通过模拟找出存在资源相关的指令对以及导致资源相关的部件;记录由资源相关引起的暂停时钟周期数,计算暂停时钟周期数占总执行周期数的百分比;论述资源相关对CPU 性能的影响,讨论解决资源相关的方法。

3. 在不采用定向技术的情况下(去掉Configuration 菜单中Enable Forwarding 选项前的勾选符),用WinDLX 运行程序data\_d.s。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。在采用定向技术的情况下(勾选Enable Forwarding),用WinDLX再次运行程序data\_d.s。重复上述3 中的工作,并计算采用定向技术后性能提高的倍数。



## 4.2 流水线中的相关（WinMIPS64）

该平台与WinDLX任选一种进行实验。

实验目的：

1. 熟练掌握WinMIPS64模拟器的操作和使用，熟悉MIPS指令集结构及其特点；
2. 加深对计算机流水线基本概念的理解；
3. 进一步了解MIPS基本流水线各段的功能以及基本操作；
4. 加深对数据相关、结构相关的理解，了解这两类相关对CPU性能的影响；
5. 了解解决数据相关的方法，掌握如何使用定向技术来减少数据相关带来的暂停。

实验平台：

WinMIPS64模拟器

实验内容和步骤：

1. 用WinMIPS64模拟器执行下列三个程序：

求和程序sum.s

for循环程序test\_for.s

test\_for.s对应的C语言程序：

```
for(int i=0;i<6;i++)
{
    a[i] = a[i] + b[i] + c[i];
}
```

分别以步进、连续的方式运行程序，观察程序在流水线中的执行情况，观察CPU中寄存器和存储器的内容。熟练掌握WinMIPS64的操作和使用。

2. 用WinMIPS64运行程序test\_for.s，通过模拟找出存在资源相关的指令对以及导致资源相关的部件；记录由资源相关引起的暂停时钟周期数，计算暂停时钟周期数占总执行周期数的百分比；论述资源相关对CPU性能的影响，讨论解决资源相关的方法。

3. 在不采用定向技术的情况下（去掉Configuration菜单中Enable Forwarding选项前的勾选符），用WinMIPS64运行程序sum.s和test\_for.s。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。在采用定向技术的情况下（勾选Enable Forwarding），用WinMIPS64再次运行程序sum.s和test\_for.s。重复上述3中的工作，并计算采用定向技术后性能提高的倍数。

## 4.3 循环展开及指令调度

实验目的：

1. 加深对循环级并行性、指令调度技术、循环展开技术以及寄存器换名技术的理解；
2. 熟悉用指令调度技术来解决流水线中的数据相关的方法；
3. 了解循环展开、指令调度等技术对 CPU 性能的改进。

实验平台：

WinDLX/WinMIPS64模拟器

实验内容和步骤：

1. 用指令调度技术解决流水线中的结构相关与数据相关

(1) 用DLX汇编语言编写代码文件\*.s，程序中应包括数据相关与结构相关（假设：加法、乘法、除法部件各有2个，延迟时间都是3个时钟周期）

(2) 通过**Configuration** 菜单中的“*Floating point stages*”选项，把加法、乘法、除法部件的个数设置为2个，把延迟都设置为3个时钟周期；

(3) 用WinDLX/ WinMIPS64运行程序。记录程序执行过程中各种相关发生的次数、发生相关的指令组合，以及程序执行的总时钟周期数；

(4) 采用指令调度技术对程序进行指令调度，消除相关；

(5) 用WinDLX/ WinMIPS64运行调度后的程序，观察程序在流水线中的执行情况，记录程序执行的总时钟周期数；

(6) 根据记录结果，比较调度前和调度后的性能。论述指令调度对于提高CPU性能的意义。

2. 用循环展开、寄存器换名以及指令调度提高性能

(1) 用DLX 汇编语言编写代码文件\*.s，程序中包含一个循环次数为4 的整数倍的简单循环；

(2) 用WinDLX 运行该程序。记录执行过程中各种相关发生的次数以及程序执行的总时钟周期数；

(3) 将循环展开3次，将4个循环体组成的代码代替原来的循环体，并对程序做相应的修改。然后对新的循环体进行寄存器换名和指令调度；

(4) 用WinDLX/ WinMIPS64运行修改后的程序，记录执行过程中各种相关发生的次数以及程序执行的总时钟周期数；

(5) 根据记录结果，比较循环展开、指令调度前后的性能。

## 4.4 Cache 性能分析

实验目的：

1. 加深对 Cache 的基本概念、基本组织结构以及基本工作原理的理解；
2. 了解 Cache 的容量、相联度、块大小对Cache 性能的影响；
3. 掌握降低 Cache 失效率的各种方法，以及这些方法对Cache 性能提高的好处；
4. 理解 Cache 失效的产生原因以及Cache 的三种失效；
5. 理解 LRU 与随机法的基本思想，及它们对Cache 性能的影响。

实验平台：

SimpleScalar 模拟器

实验内容及步骤：

1. 在基本配置情况下运行程序（请指明所选的测试程序），统计Cache 总失效次数、三种不同种类的失效次数；
2. 改变Cache 容量（\*2，\*4，\*8，\*64），运行程序（指明所选的测试程序），统计各种失效的次数，并分析Cache 容量对Cache 性能的影响；
3. 改变Cache 的相联度（1 路，2 路，4 路，8 路，64 路），运行程序（指明所选的测试程序），统计各种失效的次数，并分析相联度对Cache 性能的影响；
4. 改变Cache 块大小（\*2，\*4，\*8，\*64），运行程序（指明所选的测试程序），统计各种失效的次数，并分析Cache块大小对Cache性能的影响；
5. 分别采用LRU 与随机法，在不同的Cache容量、不同的相联度下，运行程序（指明所选的测试程序）统计Cache总失效次数，计算失效率。分析不同的替换算法对Cache性能的影响。

测试程序：

用于实验的测试程序可以使用现有的，即benchmark目录下的所有程序；也可以自己用C语言直接生成各类典型程序，比如：数学运算类、输入输出类等，通过本环境提供的特定C编译器，编译生成对应的xxx.out（默认是a.out）测试程序用于实验。每个测试程序所需时间大概是 10 分钟，选择测试程序时注意从不同组中选择，以便使得出的结果不会因为对单项有所侧重而有失偏颇。每个人从中选出4~6 个测试程序进行测试。

## 第 5 章 实验要求

实验项目共占8个学时，WinDLX/ WinMIPS64模拟器实验4个学时，SimpleScalar 模拟器4个学时，每人一组独立进行上机实验，实行开放式实验教学。

基本要求：

1. 每次实验前要进行预习。
2. 实验时自行进行程序的调试、编译和执行过程。
3. 每个实验项目文件放入一个文件夹中，文件夹命名规则统一为：学号\_实验编号（若一个项目中有多个文件，则依次命名为：学号\_实验编号\_序号）。
4. 一旦发现抄袭行为，则该实验项目计 0 分。
5. 每个实验代码调试成功后，应及时让实验指导老师给予检查和登记。
6. 实验结束后按时提交实验报告以及源程序。
7. 提交源程序时应提交全部的实验文件，然后将项目文件夹中压缩打包后提交（压缩包的名称和项目文件夹名称相同）。