**流水线及流水线中的冲突/相关**

**实验目的**

1. 加深对计算机流水线基本概念的理解。

2. 理解MIPS结构如何用5段流水线来实现，理解各段的功能和基本操作。

3. 加深对结构冲突、数据冲突、控制相关的理解，理解这冲突/相关对CPU性能的影响。

4. 进一步理解解决数据冲突和控制相关的方法，掌握如何应用定向技术减少数据冲突引起的停顿。进一步理解延迟分支技术对CPU性能的改进。

**实验内容和步骤**

首先要掌握MIPSsim模拟器的使用方法。见文档《MIPSsim使用手册》。

1. 启动MIPSsim。

2. 根据预备知识中关于流水线各段操作的描述，进一步理解流水线窗口中各段的功能，掌握各流水寄存器的含义。（用鼠标双击各段，就可以看到各流水寄存器的内容）。

3. 熟悉MIPSsim模拟器的操作和使用方法。

可以先载入一个样例程序（在本模拟器所在的文件夹下的“样例程序”文件夹中），然后分别以单步执行一个周期、执行多个周期、连续执行、设置断点等的方式运行程序，观察程序的执行情况，观察CPU中寄存器和存储器的内容的变化，特别是流水寄存器内容的变化。

4. 选择配置菜单中的“流水方式”，使模拟器工作于流水方式下。

**5. 观察程序在流水线中的执行情况，步骤如下：**

(1)选择MIPSsim的“文件”→“载入程序”选项来加载pipeline.s（在模拟器所在文件夹下的“样例程序”文件夹中）。

(2)关闭定向功能。这是通过在“配置”→“定向”（使该项前面没有“√”号）来实现的。

(3)用单步执行一周期的方式（“执行”菜单中）或用F7执行该程序，观察每一周期中，各段流水寄存器内容的变化、指令的执行情况（“代码”  窗口）以及时钟周期图。

(4)当执行到第13个时钟周期时，各段分别正在处理的指令是：

IF：LW $r4,60($r6) 取指

ID：ADDI $r3,$r0,25 读寄存器

EX：ADDI $r1,$r1,-1 执行

MEM：ADDI $r6,$r0,8 访存

WB：ADD $r2,$r1,$r0 回写

画出这时的时钟周期图：

图形用户界面, 应用程序, 表格

描述已自动生成

6. 这时各流水寄存器中的内容为：（下图分别为十进制与十六进制形式）

IF/ID.IR：2361655356

手机屏幕截图

描述已自动生成文本

描述已自动生成IF/ID.NPC：48

ID/EX.A：0

ID/EX.B：0

ID/EX.Imm：25

ID/EX.IR：537067545

EX/MEM.ALUo：4

EX/MEM.IR：539099135

MEM/WB.LMD：0

MEM/WB.ALUo：8

MEM/WB.IR：537264136

**7. 观察和分析结构冲突对CPU性能的影响，步骤如下：**

（1）加载structure\_hz.s（在模拟器所在文件夹下的“样例程序”文件夹中）。

（2）执行该程序，找出存在结构冲突的指令fadd , 对以及导致结构冲突的部件：浮点加法器。

（3）记录由结构冲突引起的停顿时钟周期数 35 ，计算停顿时钟周期数占总执行周期数的百分比：67.308%。

（4）把导致结构冲突的部件的个数改为4个。

（5）再次重复上述（1）～（3）的工作，分析结构冲突对CPU性能的影响，讨论解决结构冲突的方法：

结构冲突对CPU性能的影响：当发生冲突时，流水线会出现停顿从而降低CPU的性能。

解决结构冲突的方法：在流水线处理机中设置相互独立的指令寄存器和数据寄存器。

**8. 观察数据冲突并用定向技术（forwarding）来减少停顿，步骤如下：**

（1）全部复位。

（2）加载data\_hz.s（在模拟器所在文件夹下的“样例程序”文件夹中）。

（3）关闭定向功能。这是通过在“配置”→“定向”（使该项前面没有“√”号）来实现的。

（4）用单步执行一个周期的方式（F7）执行该程序，同时查看时钟周期图，列出在什么时刻发生了RAW（先写后读）冲突：在Cycle 4, 6, 7, 9, 10, 13, 14, 17, 18, 20, 21, 25, 26, 28, 29, 32, 33, 36, 37, 39, 40, 44, 45, 47, 48, 51, 52, 55, 56, 58, 59时，发生RAW冲突。

（5）记录数据冲突引起的停顿时钟周期数 31 以及程序执行的总时钟周期数 65 ，计算停顿时钟周期数占总执行周期数的百分比 47.692% 。

（6）复位CPU。

（7）打开定向功能。这是通过在“配置”→“定向”（使该项前面有一个“√”号）来实现的。

（8）用单步执行一周期的方式（F7）执行该程序，同时查看时钟周期图，列出在什么时刻发生了RAW（先写后读）冲突，并与（3）的结果进行比较： 在第5,9,13,17,21,25,29,33,37发生了RAW冲突，可以看到，通过定向技术，大大

减少了RAW冲突数目 。

（9）记录数据冲突引起的停顿时钟周期数 9 以及程序执行的总时钟周期数 43 。计算采用定向技术后性能提高的倍数65/43=1.51。

**9. 用延迟分支减少分支指令对性能的影响。**

（1）全部复位。

（2）载入branch.asm。

（3）关闭延迟分支功能。这是通过在“配置”→“延迟槽”选项来实现的。

（4）执行该程序。观察并记录发生分支延迟的时刻： Cycle 6, 9, 13, 21, 24, 28 。

（5）记录执行该程序所花的总时钟周期数 38 。

（6）假设延迟槽为一个，对branch.asm进行指令调度，然后保存到“delayed-branch.asm”中。

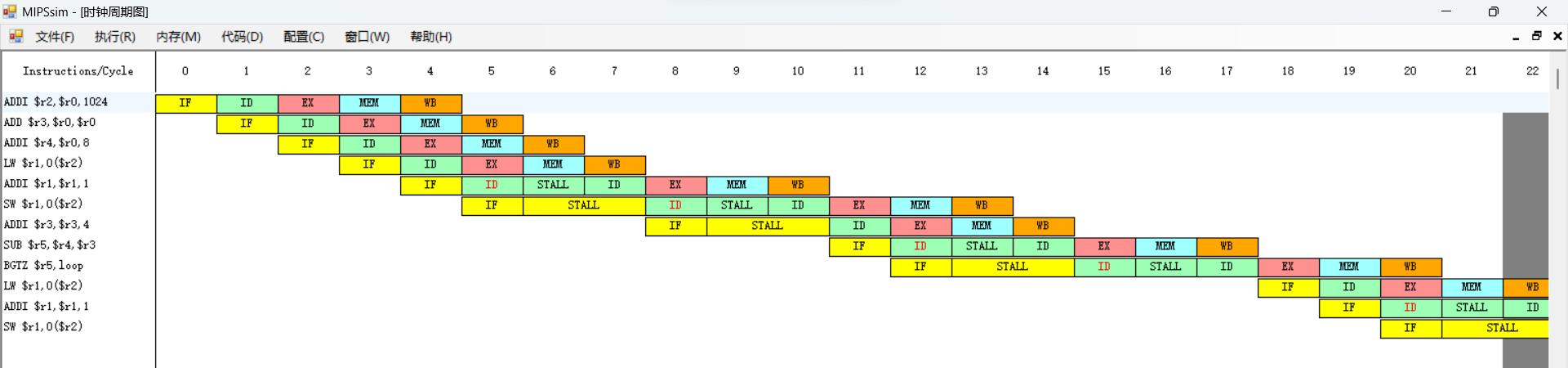
（7）载入delayed-branch.asm。

（8）打开延迟分支功能。

（9）执行该程序，观察其时钟周期图。

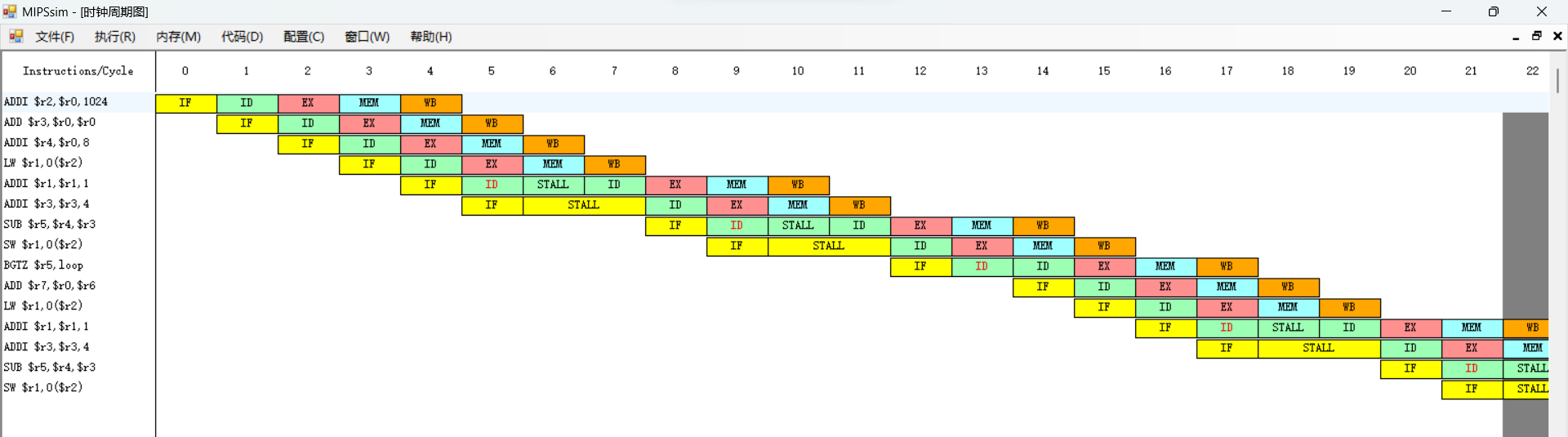
（10）记录执行该程序所花的总时钟周期数 31 。

（11）对比上述两种情况下的时钟周期图：



图表

描述已自动生成



图表, 瀑布图

描述已自动生成

（12）根据记录结果，比较没采用延迟分支和采用了延迟分支的性能。论述延迟分支对于提高CPU性能的作用：

在使用延迟槽后，指令在运行到跳转指令时，不会出现延迟等待，则能够提高CPU的性能。