

编者的话

计算机体系结构是计算机科学与技术、网络工程、信息安全专业以及计算机应用专业本科生的一门重要专业课程，其目的是提高学生从总体结构、系统分析这一层次来研究和分析计算机系统的能力，在计算机组成原理和相关课程的基础上建立起计算机系统的完整概念。学习本门课程的目标是：初步具备计算机系统的量化分析、性能评价、设计和选型的能力。

学习计算机体系结构的过程中，将会接触到很多基本概念、原理和分析方法，对于计算机性能的分析与评测不容易领会和掌握。亟需一套较系统完整的实验工具和环境。长期以来，研究人员也开发出了各种指令模拟器和分析工具。通过这些工具，能够直观地理解计算机体系结构中的细节内容，有助于培养分析和解决问题的能力与创新精神。其中著名的威斯康辛大学的计算机体系结构主页（<http://arch-www.cs.wisc.edu/wwwarch/public/home>）提供了丰富的仿真和测评工具。我们在2006年以前使用的是基于有关模拟器的自编实验指导书。

张晨曦教授等编著《计算机系统结构实践教程》设计和编写了11个实验：指令系统和体系结构、流水线及流水线中的冲突、指令调度和延迟分支、Cache性能分析、Tomasulo算法、再定序缓冲（ROB）工作原理、多Cache一致性——监听协议、多Cache一致性——目录协议。实验内容基本覆盖了传统的计算机体系结构教学内容，有利于加深对体系结构知识的理解。特别是该书所附光盘提供了系统结构实验模拟器（可运行于Windows平台的模拟器）。这些模拟器界面友好，使用方便、直观，交互性很强。我们在信息学院2007级2008级、2009级以及软件学院2009级计应班的计算机体系结构课程实验的使用中，效果良好。

以下是2010级体系结构实验的参考范例（计算机系2009级蓝子睿同学的实验报告），希望同学们把更多的时间用于实验的测试、分析和性能评价中，提高课程的学习效率，进一步加深课程知识体系的理解。

计算机体系结构教学小组

2012年9月

计算机系统结构实践教程

作 者：张晨曦

出版社：清华大学出版社

出版时间：2010 年 5 月 1 日

ISBN：9787302223719

定价：19.00



内容简介：《计算机系统结构实践教程》设计和编写了 11 个实验：指令系统和体系结构、流水线及流水线中的冲突、指令调度和延迟分支、Cache 性能分析、Tomasulo 算法、再定序缓冲（ROB）工作原理、多 Cache 一致性——监听协议、多 Cache 一致性——目录协议。

《计算机系统结构实践教程》覆盖面广，内容丰富，有利于加深对系统结构知识的理解。《计算机系统结构实践教程》的实验既有基于 MIPS 指令集的，也有基于 SPARC 指令集的。教师可根据具体教学需要选择。《计算机系统结构实践教程》包含了实验所需相关知识的介绍，可以跟大多数系统结构教材配合使用。《计算机系统结构实践教程》所附光盘提供了我们专门为系统结构实验开发的一套运行于 Windows 平台的模拟器。这些模拟器界面友好，使用方便、直观，交互性很强。随书光盘还提供了计算机系统结构课程相关的动画和视频课件。《计算机系统结构实践教程》可作为本科生或研究生的系统结构课程以及计算机组成与结构课程的实验教材，也可作为自学者的辅助教材。

目录

第一部分 基于 MIPS 体系结构

实验 1 MIPS 指令系统和 MIPS 体系结构

- 1.1 实验目的
- 1.2 实验平台
- 1.3 实验内容和步骤
- 1.4 MIPSsim 使用手册
 - 1.4.1 启动模拟器
 - 1.4.2 MIPSsim 的窗口
 - 1.4.3 MIPSsim 的菜单
- 1.5 相关知识：MIPS 指令系统
 - 1.5.1 MIPS 的寄存器
 - 1.5.2 MIPS 的数据表示
 - 1.5.3 MIPS 的数据寻址方式
 - 1.5.4 MIPS 的指令格式
 - 1.5.5 MIPS 的部分指令介绍

实验 2 流水线及流水线中的冲突

- 2.1 实验目的
- 2.2 实验平台
- 2.3 实验内容和步骤
- 2.4 MIPSsim 使用手册
- 2.5 相关知识：流水线、相关与冲突
 - 2.5.1 一条经典的 5 段流水线
 - 2.5.2 相关与流水线冲突
 - 2.5.3 流水线的实现

实验 3 指令调度和延迟分支

- 3.1 实验目的
- 3.2 实验平台
- 3.3 实验内容和步骤
- 3.4 MIPSsim 使用手册
- 3.5 相关知识：指令调度和延迟分支
 - 3.5.1 指令调度
 - 3.5.2 延迟分支

第二部分 基于 SPARC 体系结构

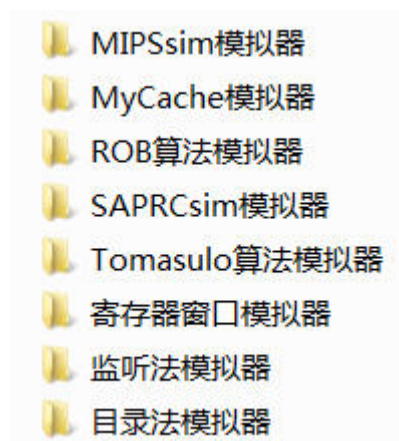
实验 4 SPARC 指令系统和 SPARC 体系结构

- 4.1 实验目的
- 4.2 实验平台
- 4.3 实验内容和步骤
- 4.4 SPARCsim 使用手册
 - 4.4.1 启动程序
 - 4.4.2 SPARCsim 的窗口

- 4.4.3 SPARCSim 的菜单
- 4.5 相关知识：UltraSPARC 指令系统
 - 4.5.1 UltraSPARC 的寄存器
 - 4.5.2 UltraSPARC 的数据表示
 - 4.5.3 UltraSPARC 的数据寻址方式
 - 4.5.4 UltraSPARC 的指令格式
 - 4.5.5 UltraSPARC 的部分指令介绍
- 实验 5 流水线及流水线中的冲突
 - 5.1 实验目的
 - 5.2 实验平台
 - 5.3 实验内容和步骤
 - 5.4 SPARCSim 使用手册
 - 5.5 相关知识：流水线、相关与冲突
 - 5.5.1 一条经典的 5 段流水线
 - 5.5.2 相关与流水线冲突
 - 5.5.3 流水线的实现
- 实验 6 指令调度和延迟分支
 - 6.1 实验目的
 - 6.2 实验平台
 - 6.3 实验内容和步骤
 - 6.4 SPARCSim 使用手册
 - 6.5 相关知识：指令调度和延迟分支
- 第三部分 Cache 和指令并行实验
- 实验 7 Cache 性能分析
 - 7.1 实验目的
 - 7.2 实验平台
 - 7.3 实验内容和步骤
 - 7.3.1 Cache 容量对不命中率的影响
 - 7.3.2 相联度对不命中率的影响
 - 7.3.3 Cache 块大小对不命中率的影响
 - 7.3.4 替换算法对不命中率的影响
 - 7.4 MyCache 模拟器的使用方法
 - 7.5 相关知识：Cache 的基本原理
 - 7.5.1 Cache 的映像规则
 - 7.5.2 查找方法
 - 7.5.3 替换算法
 - 7.5.4 写策略
 - 7.5.5 改进 Cache 性能
 - 7.5.6 3 种类型的不命中
 - 7.5.7 降低不命中率的方法
 - 7.5.8 分离 Cache 和混合 Cache
- 实验 8 Tomasulo 算法
 - 8.1 实验目的
 - 8.2 实验平台

8.3	实验内容和步骤
8.4	Tomasulo 算法模拟器的使用方法
8.5	相关知识: Tomasulo 算法
8.5.1	基本思想
8.5.2	具体算法
实验 9	再定序缓冲 (ROB) 工作原理
9.1	实验目的
9.2	实验平台
9.3	实验内容和步骤
9.4	ROB 模拟器的使用方法
9.5	相关知识: 再定序缓冲 ROB
实验 10	多 Cache 一致性——监听协议
10.1	实验目的
10.2	实验平台
10.3	实验内容和步骤
10.4	监听协议模拟器的使用方法
10.5	相关知识: 监听协议
10.5.1	基本思想
10.5.2	监听协议的实现
实验 11	多 Cache 一致性——目录协议
11.1	实验目的
11.2	实验平台
11.3	实验内容和步骤
11.4	目录协议模拟器的使用方法
11.5	相关知识: 目录协议
11.5.1	目录协议的基本思想
11.5.2	目录协议实例
附录 A	MIPSsim 的指令列表 (MIPS64 指令集的一个子集)
附录 B	MIPSsim 的指令系统 (MIPS64 指令集的一个子集)
附录 C	模拟器 MIPSsim 的汇编语言
	参考文献

随书附送的模拟器





实验报告

实验人： 蓝子睿 学号： 09388092 日期： 2011-11-06
院（系）： 信科院计算机系 专业（班级）： 09计算机科学与技术
实验题目： MIPS指令系统和MIPS体系结构

1. 实验目的

- (1) 了解和熟悉指令级模拟器。
- (2) 熟练掌握 MIPSsim 模拟器的操作和使用方法。
- (3) 熟悉 MIPS 指令系统及其特点，加深对 MIPS 指令操作语义的理解。
- (4) 熟悉 MIPS 体系结构。

2. 实验平台

实验平台采用指令级和流水线操作级模拟器 MIPSsim（随书光盘中提供）。

设计：张晨曦教授，版权所有。

开发：孙太一

3. 实验内容和步骤

首先要阅读 MIPSsim 模拟器的使用方法，然后了解 MIPSsim 的指令系统和汇编语言（见附录 A、附录 B 和附录 C）。

- (1) 启动 MIPSsim（用鼠标双击 MIPSsim.exe）。
- (2) 选择“配置”——“流水方式”选项，使模拟器工作在非流水方式下。
- (3) 参照 1.4 节的使用说明，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条指令、执行多条指令、连续执行、设置断点等方式运行程序，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。

- (4) 选择“文件”→“载入程序”选项，加载样例程序 alltest.asm，然后查看“代码”窗口，

查看程序所在的位置（起始地址为 0x00000100）。

(5) 查看“寄存器”窗口 PC 寄存器的值：[PC] = 0x00000000。

(6) 执行 load 和 store 指令，步骤如下：

①单步执行 1 条指令(F7)。

②下一条指令地址为 0x00000004，是一条有（有，无）符号载入字节（字节，半字，字）指令。

③单步执行 1 条指令(F7)。

④查看 R1 的值，[R1] = 0xFFFFFFFFFFFFFFF80。

⑤下一条指令地址为 0x00000008，是一条 有（有，无）符号载入字（字节，半字，字）指令。

⑥单步执行 1 条指令。

⑦查看 R1 的值，[R1] = 0x0000000000000080。

⑧下一条指令地址为 0x0000000C，是一条 无（有，无）符号载入字节（字节，半字，字）字节 指令。

⑨单步执行 1 条指令。

⑩ 查看 R1 的值，[R1] = 0x0000000000000080。

⑪ 单步执行 1 条指令。

⑫ 下一条指令地址为 0x00000010，是一条保存字（字，半字，字）指令。

⑬ 单步执行 1 条指令(F7)。

⑭ 查看内存 BUFFER 处字的值，值为 0x00000080。

(7) 执行算术运算类指令。步骤如下：

①双击“寄存器”窗口中的 R1，将其值修改为 2。

②双击“寄存器”窗口中的 R2，将其值修改为 3。

③单步执行 1 条指令

④下一条指令地址为 0x00000020，是一条加法指令。

⑤单步执行 1 条指令。

⑥查看 R3 的值，[R3] = 0x0000000000000005。

⑦下一条指令地址为 0x00000024，是一条乘法指令。

⑧单步执行 1 条指令。

⑨查看 LO、HI 的值，[LO] = 0x0000000000000006，[HI] = 0x0000000000000000。

(8) 执行逻辑运算类指令。步骤如下：

①双击“寄存器”窗口中的 R1，将其值修改为 0xFFFF0000。

②双击“寄存器”窗口中的 R2，将其值修改为 0xFF00FF00。

③单步执行 1 条指令。

④下一条指令地址为 0x00000030，是一条逻辑与运算指令，第二个操作数寻址方式

是寄存器直接寻址（寄存器直接寻址，立即数寻址）。

⑤单步执行 1 条指令。

⑥查看 R3 的值，[R3] = 0xFF000000。

⑦下一条指令地址为 0x00000034，是一条逻辑与指令，第二个操作数寻址方式是立即数寻址（寄存器直接寻址，立即数寻址）。

⑧单步执行 1 条指令。

⑨查看 R3 的值，[R3] = 0x0000000000000000。

(9) 执行控制转移类指令。步骤如下：

①双击“寄存器”窗口中的 R1，将其值修改为 2。

②双击“寄存器”窗口中的 R2，将其值修改为 2。

③单步执行 1 条指令。

④下一条指令地址为 0x00000040，是一条 BEQ 指令，其测试条件是[R1]=[R2]，目标地址 0x0000004C。

⑤单步执行 1 条指令。

⑥查看 PC 的值，[PC] = 0x0000004C，表明分支 成功（成功，失败）。

⑦下一条指令是一条 BGEZ 指令，其测试条件是[R1]>0，目标地址为 0x00000058。

⑧单步执行 1 条指令。

⑨查看 PC 的值，[PC] = 0x00000058，表明分支 成功（成功，失败）。

⑩下一条指令是一条 BGEZAL 指令，其测试条件是 [R1]>0，目标地址为 0x00000064。

⑪单步执行 1 条指令。

⑫查看 PC 的值，[PC] = 0x00000064，表明分支 成功（成功，失败）；查看 R31 的值，[R31] = 0x000000000000005C。

⑬单步执行 1 条指令。

⑭查看 R1 的值，[R1] = 0x0000000000000074。

⑮下一条指令地址为 0x00000068，是一条 JALR 指令，保存目标地址的寄存器为 R1，保存返回地址的目标寄存器为 R3。

⑯单步执行 1 条指令。

⑰查看 PC 和 R3 的值，[PC] = 0x00000074，[R3] = 0x000000000000006C。

4. 实验心得

通过这次实验，我对 MIPS 体系结构的计算机，尤其是对 MIPS 的指令结构有了初步的了解。总的来说，MIPS 结构的计算机特点就是指令相对简单，指令结构整齐，寄存器多，每条指令的执行周期比较短（多为 1~2 个周期）等。由于平时很少接触 MIPS 等 RISC 结构计算机，都是用 x86 结构的 CISC 计算机，现在这个实验让我对 RISC 有了解，知识面更全

面。并且我从网络上了解到 RISC 结构在嵌入式领域有很广泛的应用，很多嵌入式设备如手机、游戏机等芯片都是 RISC 架构，而我国的龙心 CPU 也是基于 RISC 架构，可看出 RISC 架构将会是以后 CPU 体系结构发展的趋势，从现在 CISC 结构 CPU 已尽量精减指令向 RISC 靠拢便可窥知一二。



实验报告

实验人： 蓝子睿 学号： 09388092 日期： 2011-11-07
院（系）： 信科院计算机系 专业（班级）： 09计算机科学与技术
实验题目： 流水线及流水线中的冲突

1. 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) 加深对数据冲突、结构冲突的理解，理解这两类冲突对 CPU 性能的影响。
- (4) 进一步理解解决数据冲突的方法，掌握如何应用定向技术来减少数据冲突引起的停顿。

2. 实验平台

实验平台采用指令级和流水线操作级模拟器 MIPSsim（随书光盘中提供）。

设计：张晨曦教授，版权所有。

开发：孙太一

3. 实验内容和步骤

首先要阅读 MIPSsim 模拟器的使用方法，然后了解 MIPSsim 的指令系统和汇编语言（见附录 A、附录 B 和附录 C）。

- (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) 参照前一个实验 1.4 节的使用说明，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”选项，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“配置”－“定向”（使该项前面没有√号）来实现的。

③用单步执行一个周期的方式（在“执行”菜单中）或按 F7 键执行该程序，观察每一个周期中，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: BEQ \$R1, \$R0, ret

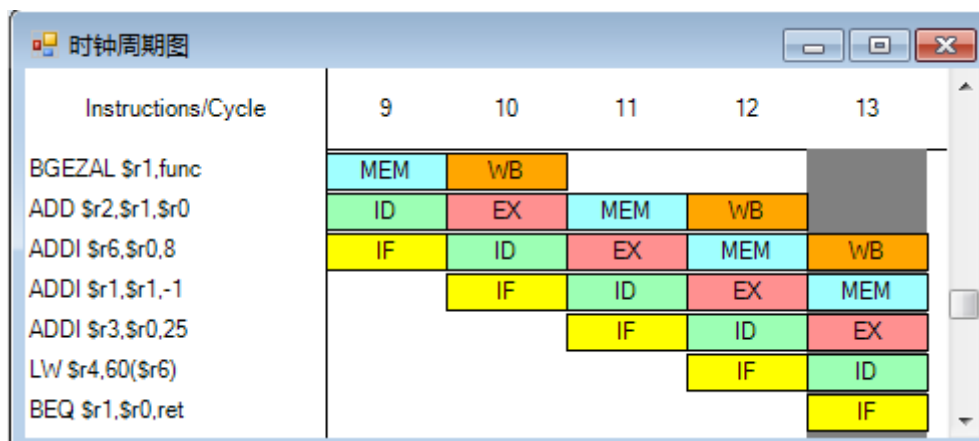
ID: LW \$R4, 60(\$R6)

EX: ADDI \$R3, \$R0, 25

MEM: ADDI \$R1, \$R1, -1

WB: ADDI \$R6, \$R0, 8

画出这时的始终周期图。



(6) 这时各流水寄存器中的内容为：

IF/ID. IR: 0000000010200004

IF/ID. NPC: 0000000000000034

ID/EX. A: 0000000000000008

ID/EX. B: 0000000000000000

ID/EX. Imm: 000000000000003C

ID/EX. IR: 000000008CC4003C

EX/MEM. ALUo: 0000000000000019

EX/MEM. IR: 0000000020030019

MEM/WB. LM: 0000000000000000

MEM/WB. ALUo: 0000000000000004

MEN/WB. IR: 000000002021FFFF

(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对以及导致结构冲突的部件。

存在结构冲突的指令是：

ADD.D \$f2, \$f0, \$f1

ADD.D \$f3, \$f0, \$f1

ADD.D \$f4, \$f0, \$f1

ADD.D \$f5, \$f0, \$f1

ADD.D \$f6, \$f0, \$f1

ADD.D \$f7, \$f0, \$f1

ADD.D \$f8, \$f0, \$f1

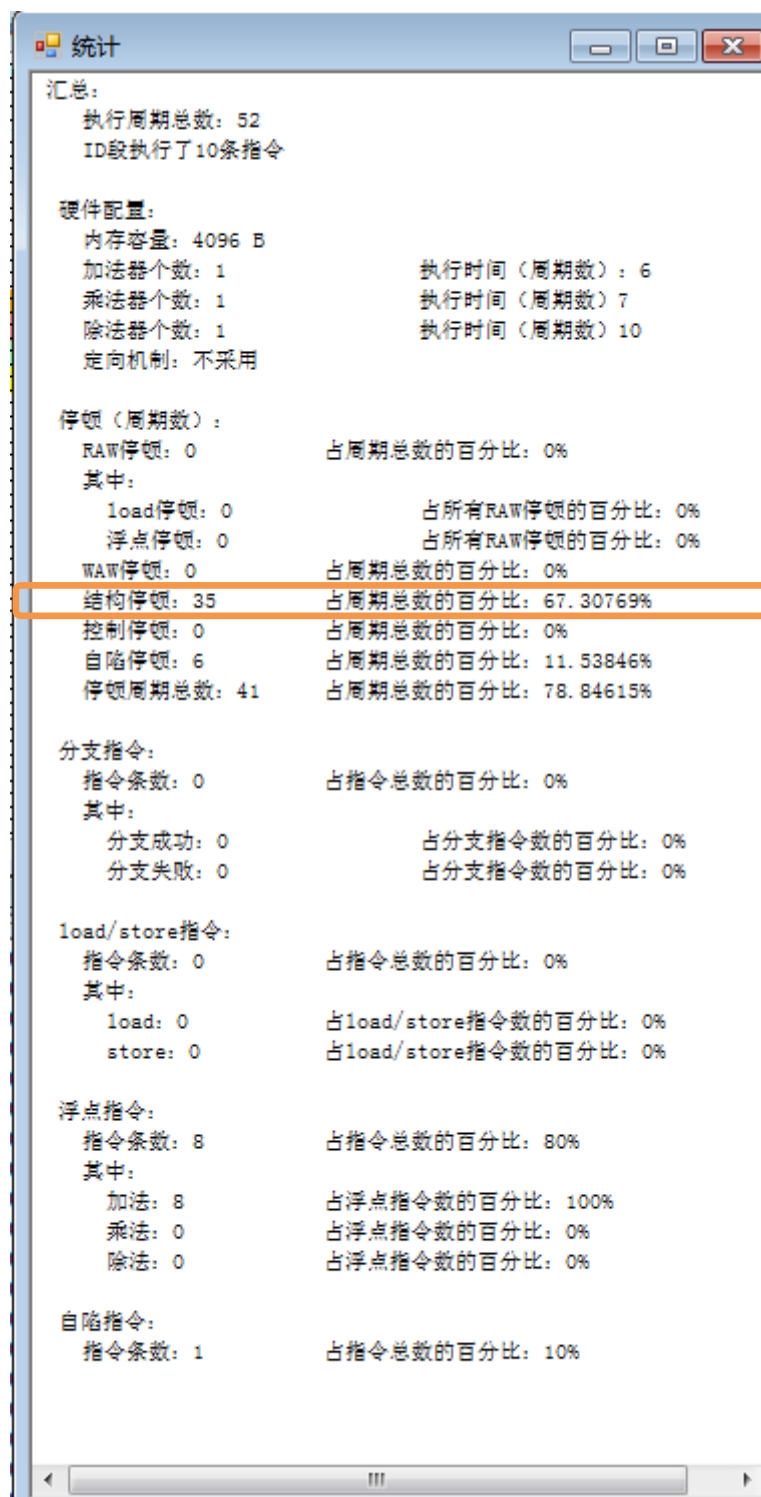
ADD.D \$f9, \$f0, \$f1

上面的程序段，所有指令都存在结构冲突（因为每条指令都是浮点加法指令）。导致结构冲突的部件是：**流水加法器**

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

停顿周期数：35 次

占总执行周期百分比：67.30769%

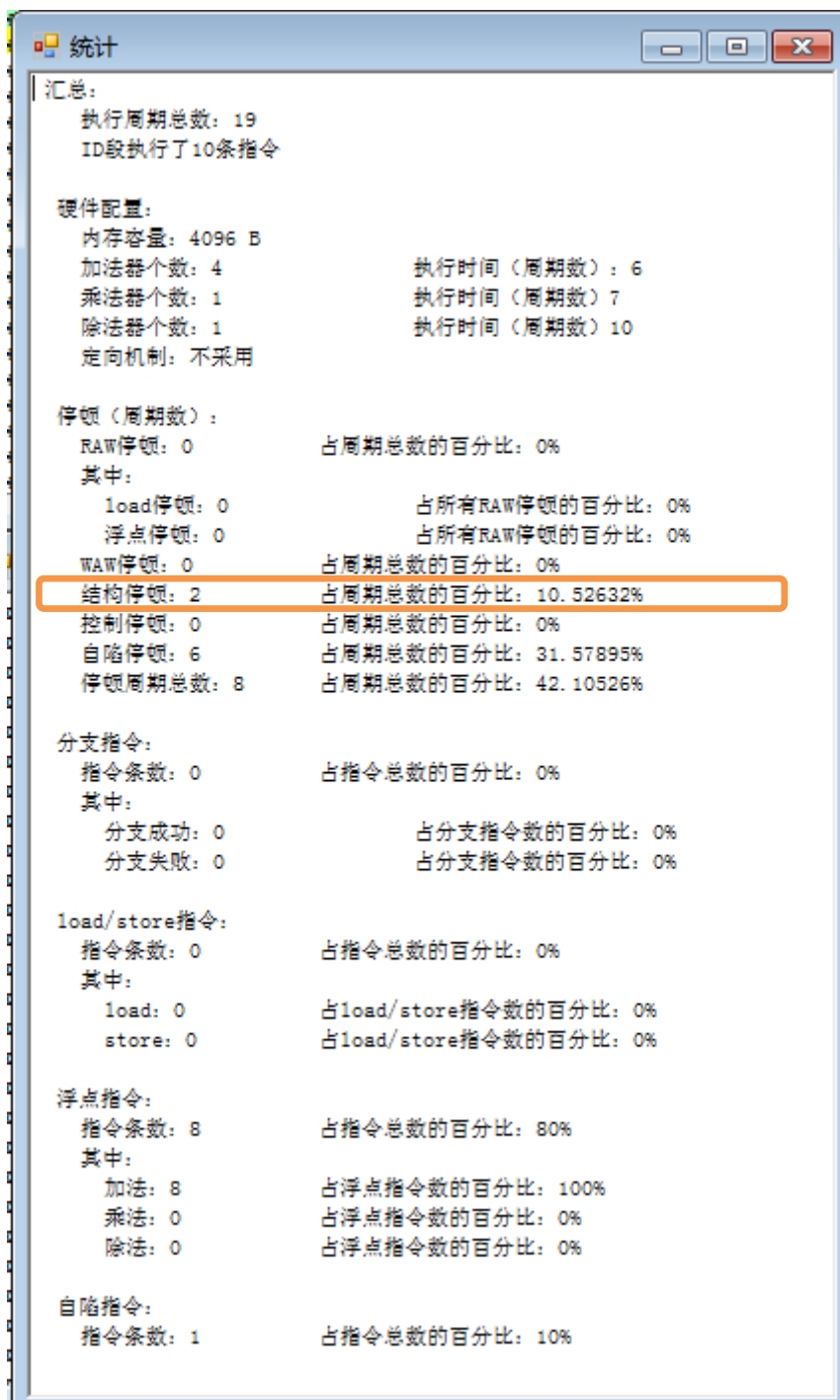


④把浮点加法器的个数改为4个。

⑤再次重复步骤①~③的工作。

停顿周期数: 2

占总执行周期百分比: 10.52632%



⑥分析结构冲突对 CPU 性能的影响，讨论解决结构冲突的方法。

结构冲突使得流水线中断，流水线不能连续地输入任务，不能充分发挥流水线的优势。从数据上可以看到，这段程序总共执行了 9 条指令（8 条加法指令+1 条自陷指令），用了 52 个周期，平均 CPI 是 5.8，效率太低。解决结构冲突的方法是重复设置存在冲突（或争用）的部件，或者对功能部件进行流水化。

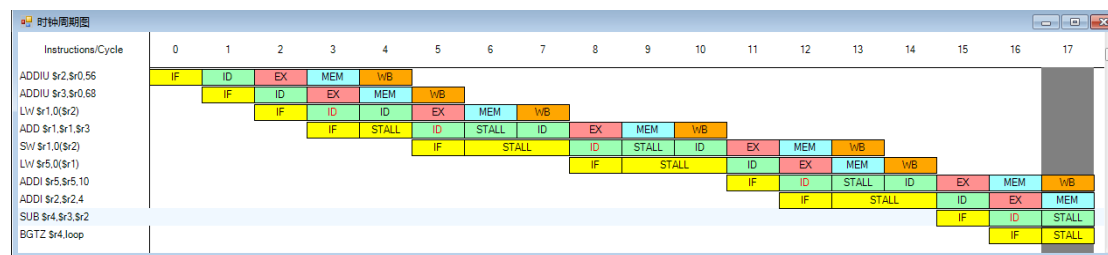
(8) 观察数据冲突并用定向技术来减少停顿，步骤如下：

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“配置”→“定向”（是该项前面没有打钩）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。



时钟周期图（前 17 个周期）

发生 RAW 冲突的指令是：

```

{
  ADDIU $R3, R0, 68
  ADD $R1, $R1, $R3
}

{
  SW $R1, 0($R2)
  LW $R5, 0($R1)
}

{
  LW $R5, 0($R1)
  ADDI $R5, $R5, 10
}

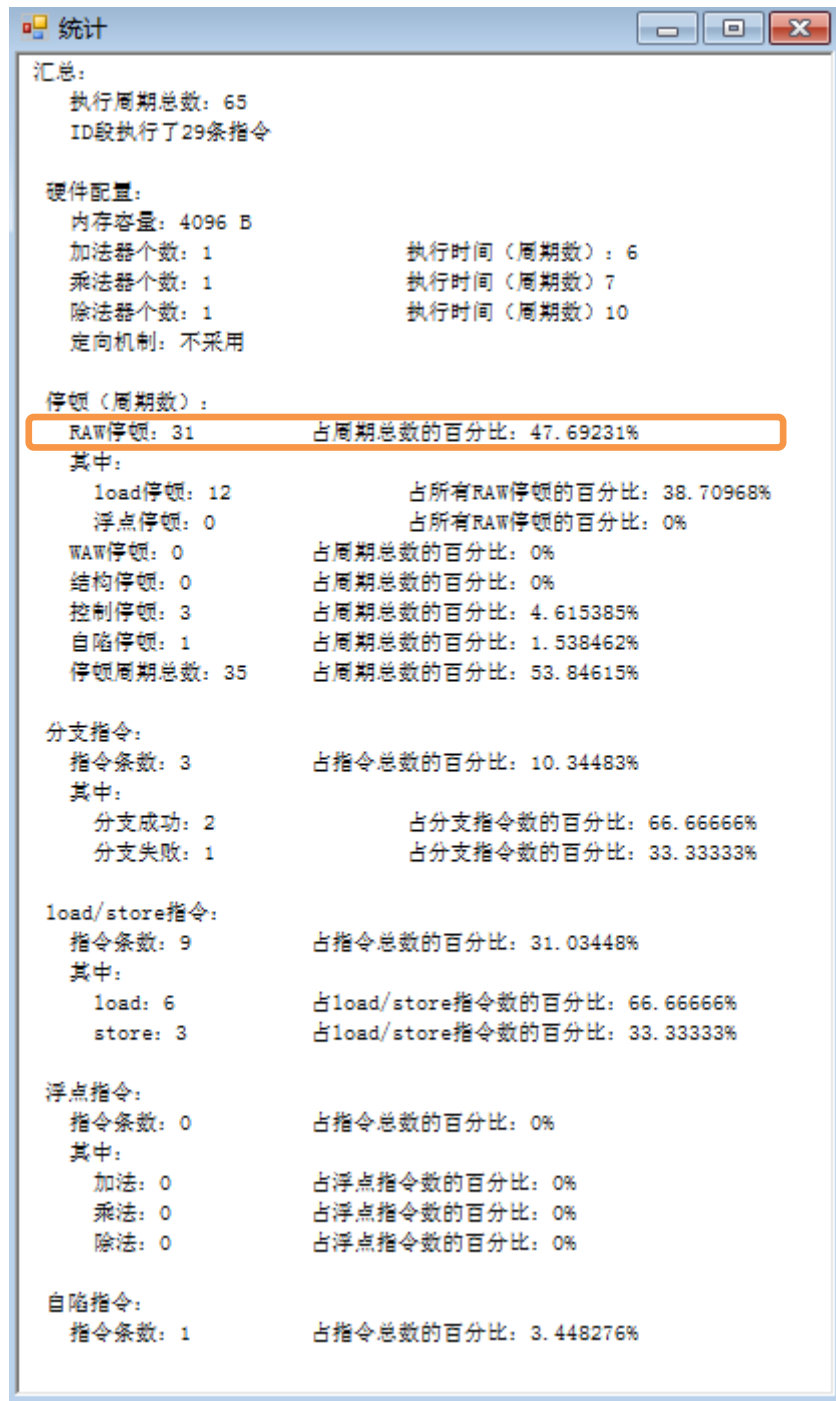
{
  ADDI $R2, $R2, 4
  SUB $R4, $R3, $R2
}

{
  SUB $R4, $R3, $R2
  BGTZ $R4, loop
}
  
```

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

停顿周期数：31

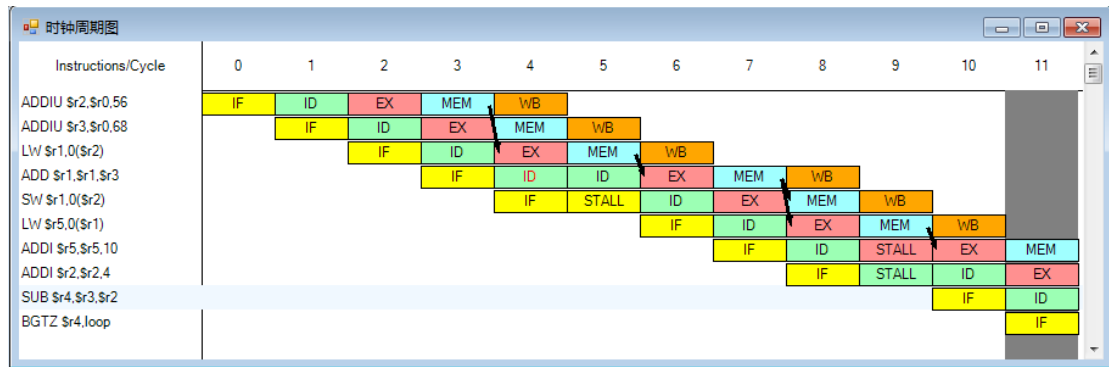
占总执行周期数的百分比：47.69231%



⑥复位 CPU。

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

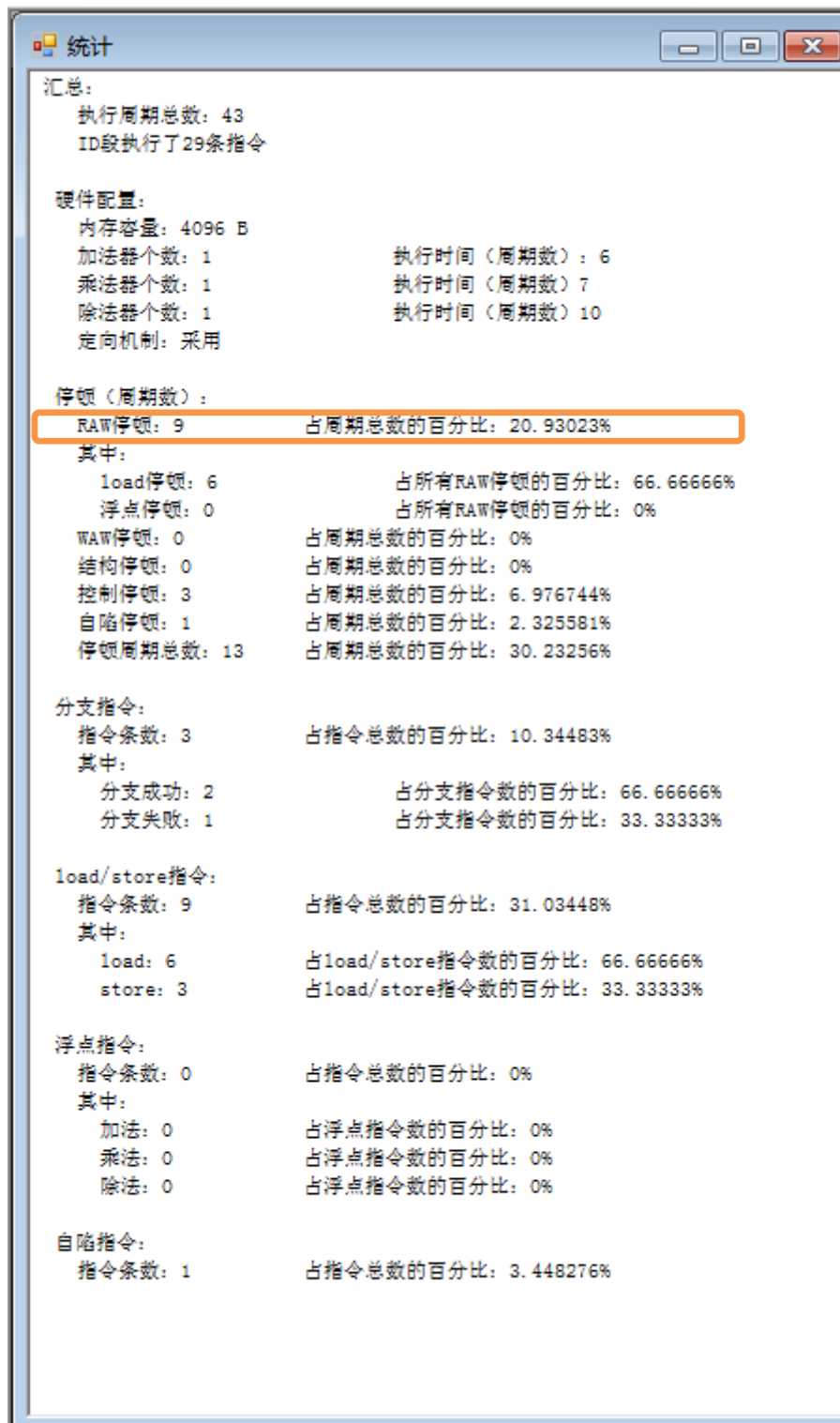
⑧用单步执行一个周期的方式(F7)执行该程序,同时查看时钟周期图,列出在什么时刻发生了RAW(先写后读)冲突,并与步骤③的结果进行比较。



从周期图上可以看到，在 LW \$R5, 0(\$R1)和 ADDI \$R5, \$R5, 10 之间发生了 RAW 冲突

停顿周期数：9

占总执行周期数百分比：20.93023%



⑨记录数据冲突引起的停顿时钟周期数以及程序执行的总时钟周期数. 计算采用定向技术后的性能是原来的几倍。

未采用定向技术之前:

停顿周期数: 31

总时钟周期数: 65

采用定向技术之后:

停顿周期数：9

总时钟周期数：43

采用了定向技术后的性能的原来的 $65/34 = 1.91$ 倍，可见性能的改善非常明显。

4. 实验心得

通过这次实验，我体会到了结构冲突和数据冲突（尤其是写后读冲突）对 CPU 资源的白白浪费程度有多严重，而解决结构冲突的方法主要是重复设置冲突部件和使功能部件流水化，解决数据冲突的方法主要是采用重定向技术。在结构冲突例子中，样例程序给出的是连续的 8 条加法指令。就这样的 8 条指令，CPU 的利用率就仅有 33% 左右。如果再多给一些连加指令（在循环中，这是很常见的，比如一个循环一万次的加法操作）CPU 的利用率就更加低得可怜。数据冲突的例子亦然，同样地造成 CPU 资源的白白浪费。这让我深深地体会到优化体系结构的重要性，也深深地明白到为什么做计算机体系结构的人会对这些浪费如此“斤斤计较”。



实验报告

实验人： 蓝子睿 学号： 09388092 日期： 2011-11-08
院（系）： 信科院计算机系 专业（班级）： 09计算机科学与技术
实验题目： 指令调度和延迟分支

1. 实验目的

- (1) 加深对指令调度技术的理解。
- (2) 加深对延迟分支技术的理解。
- (3) 熟练掌握用指令调度技术解决流水线中的数据冲突的方法。
- (4) 进一步理解指令调度技术对 CPU 性能的改进。
- (5) 进一步理解延迟分支技术对 CPU 性能的改进。

2. 实验平台

实验平台采用指令级和流水线操作级模拟器 MIPSsim（随书光盘中提供）。

设计：张晨曦教授，版权所有。

开发：孙太一

3. 实验内容和步骤

首先要阅读 MIPSsim 模拟器的使用方法，然后了解 MIPSsim 的指令系统和汇编语言（见附录 A、附录 B 和附录 C）。

- (1) 启动 MIPSsim。
- (2) 根据 2.5 节的相关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（双击各段，就可以看到各流水寄存器中的内容）。
- (3) 选择“配置”—“流水方式”选项，使模拟器工作于流水方式下。
- (4) 用指令调度技术解决流水线中的数据冲突。

①启动 MIPSsim。

②用 MIPSsim 的“文件” — “载入程序”选项来加载 schedule. 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“配置” — “定向”选项（使该项前面没有√号）来实现的。

④执行所载入的程序。通过查看统计数据 and 时钟周期图，找出并记录程序执行过程中各种冲突发生的次数、发生冲突的指令组合以及程序执行的总时钟周期数。

冲突发生次数：17 次（RAW 停顿 16 次，自陷停顿 1 次）

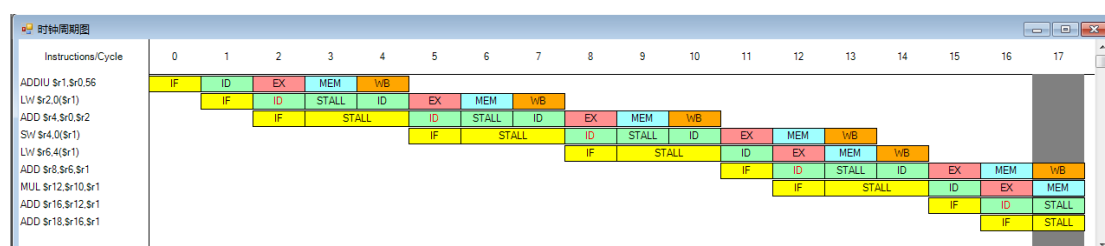
程序执行周期总数：33

发生冲突的指令组合：

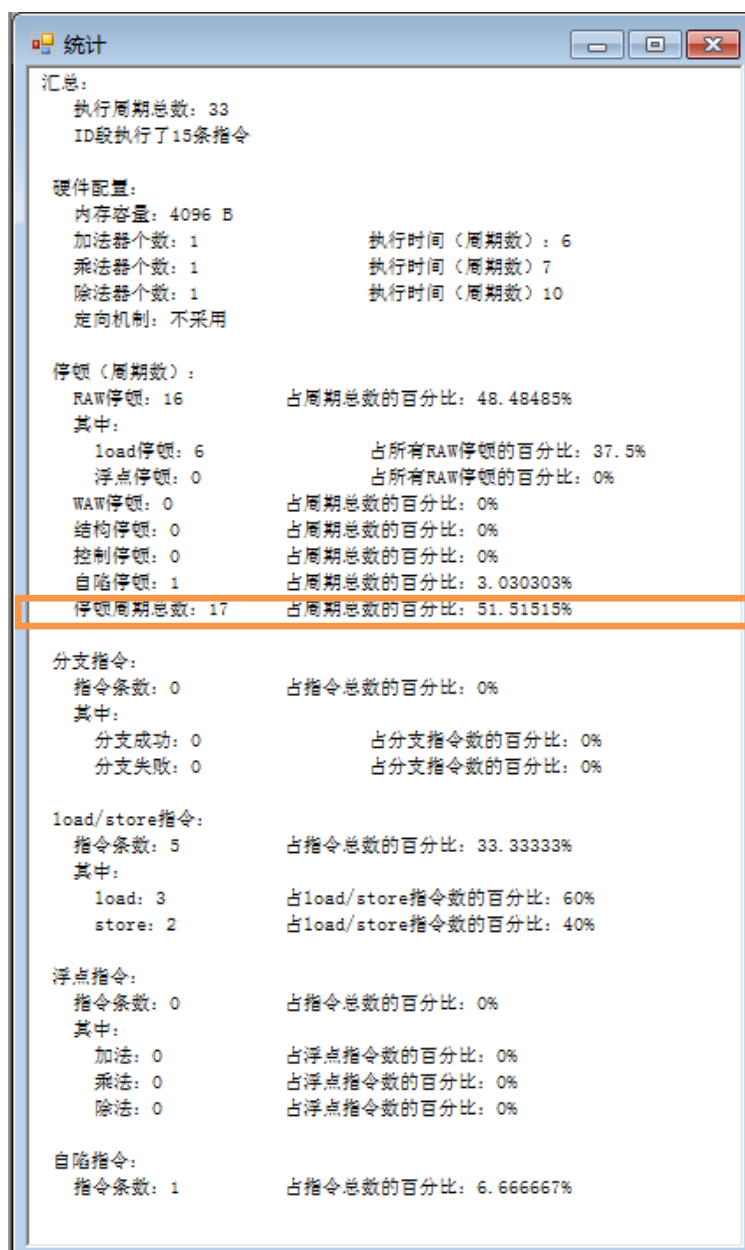
```

{ ADDIU $r1, $r0, A
  LW    $r2, 0($r1)
{ ADDIU $r1, $r0, A
  SW    $r4, 0($r1)
  LW    $r6, 4($r1)
{ LW    $r6, 4($r1)
  ADD   $r8, $r6, $r1
{ MUL   $r12, $r10, $r1
  ADD   $r16, $r12, $r1
{ ADD   $r16, $r12, $r1
  ADD   $r18, $r16, $r1
{ LW    $r20, 8($r1)
  MUL   $r22, $r20, $r14
  
```

时钟周期图：



数据统计图:



⑤采用指令调度技术对程序进行指令调度，消除冲突。将调度后的程序存到 after-schedule.s 中。

after-schedule.s 如下

```
.text
main:
ADDIU $r1, $r0, A
MUL    $r24, $r26, $r14
LW     $r6, 4($r1)
LW     $r2, 0($r1)
ADD    $r4, $r0, $r2
```

```

ADD    $r8, $r6, $r1
SW     $r4, 0($r1)
MUL    $r12, $r10, $r1
LW     $r20, 8($r1)
ADD    $r16, $r12, $r1
MUL    $r22, $r20, $r14
ADD    $r18, $r16, $r1
SW     $r18, 16($r1)
TEQ    $r0, $r0
.data
A:
.word 4, 6, 8

```

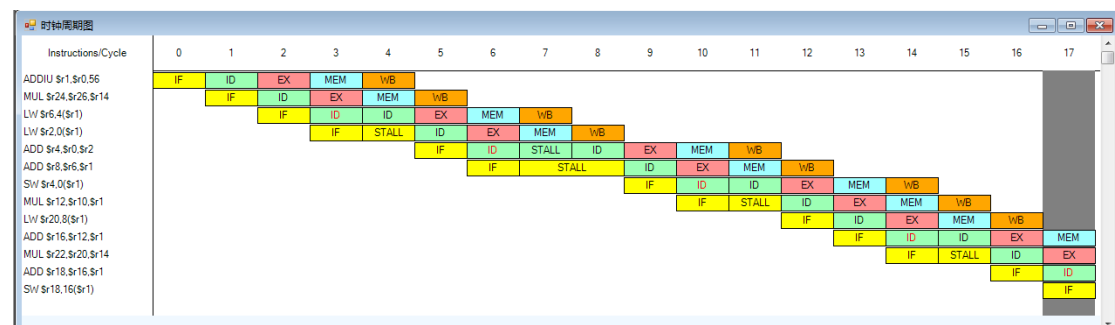
⑥载入 after-schedule.s。

⑦执行该程序。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。

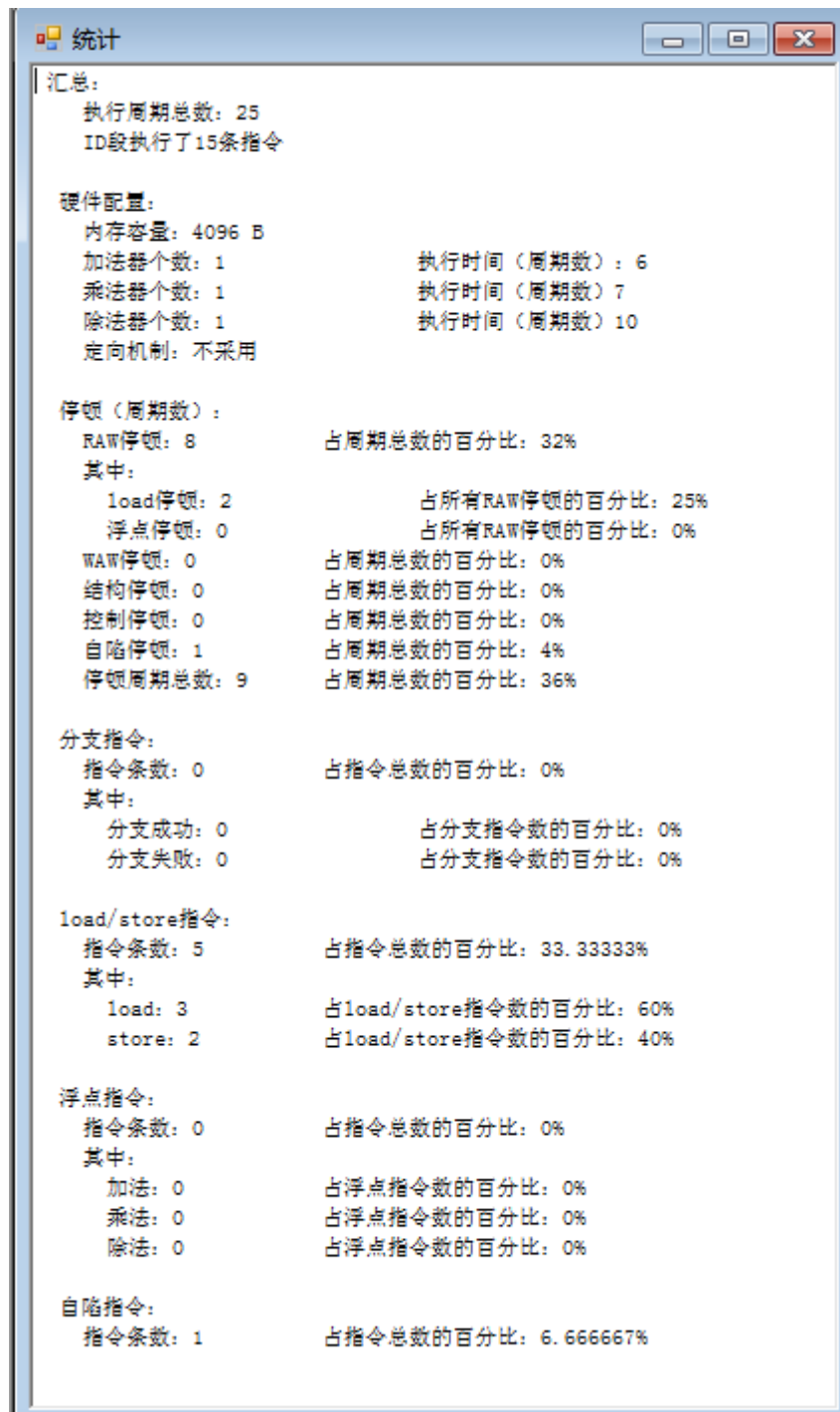
总时钟周期数：25

停顿周期数：9

时钟周期图：



数据统计图：



⑧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 性能的作用。

调度前：

冲突发生次数：17 次（RAW 停顿 16 次，自陷停顿 1 次）

程序执行周期总数：33

调度后：

停顿周期数：9 次（RAW 停顿 8 次，自陷停顿 1 次）

总时钟周期数：25

可见，调试后无论是总时钟周期数还是停顿周期数都有明显减少，其中停顿周期数减少近一半，整体加速比 $33/25=1.32$ 。调度方法是可有效减少停顿，提高 CPU 性能的（本例中，性能提升近 32%）。

(5) 用延迟分支减少分支指令对性能的影响。

①启动 MIPSsim。

②载入 branch.s。

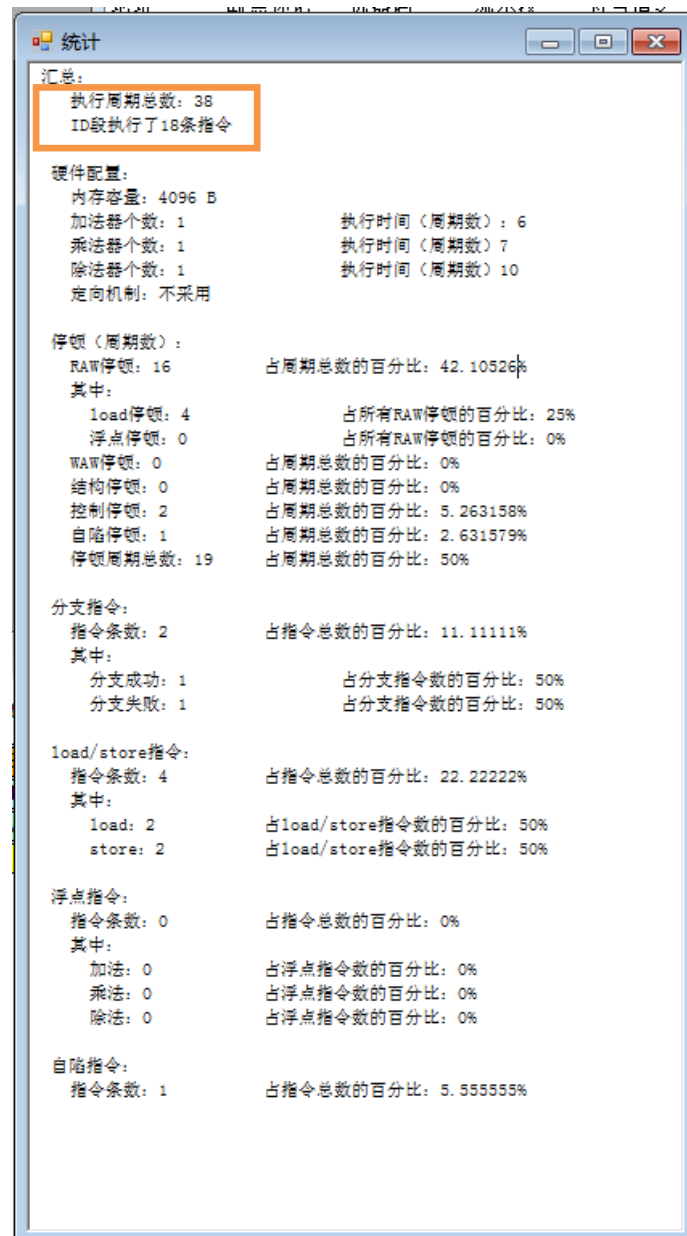
③关闭延迟分支功能。这是通过“配置”→“延迟槽”选项来实现的。

④执行该程序。观察并记录发生分支延迟的时刻。

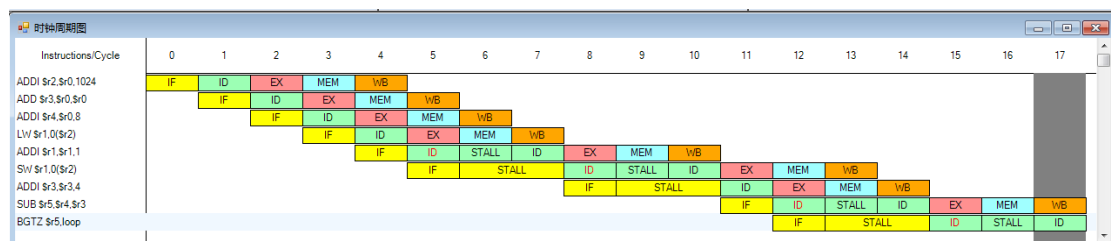
⑤记录执行该程序所用的总时钟周期数。

总时钟周期数：38

数据统计图：



时钟周期图:



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

.text

main:

ADDI \$r2, \$r0, 1024

```

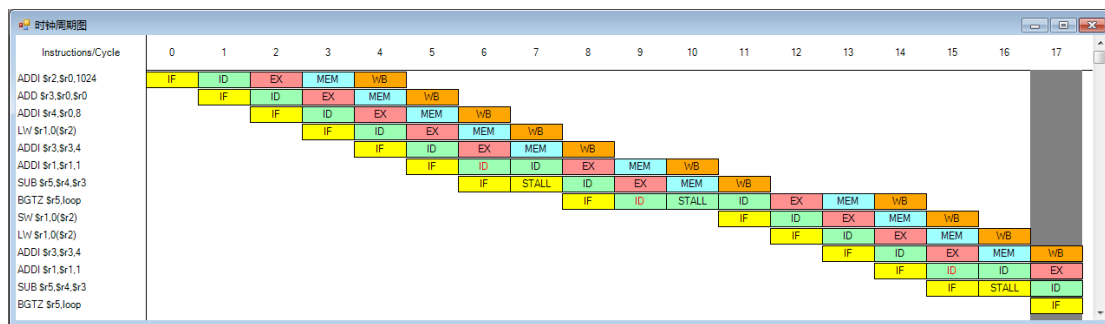
ADD    $r3, $r0, $r0
ADDI   $r4, $r0, 8
loop:
LW     $r1, 0($r2)
ADDI   $r3, $r3, 4
ADDI   $r1, $r1, 1
SUB    $r5, $r4, $r3
BGTZ   $r5, loop
SW     $r1, 0($r2)
ADD    $r7, $r0, $r6
TEQ    $r0, $r0

```

⑦载入 delayed-branch. S..

⑧打开延迟分支功能。

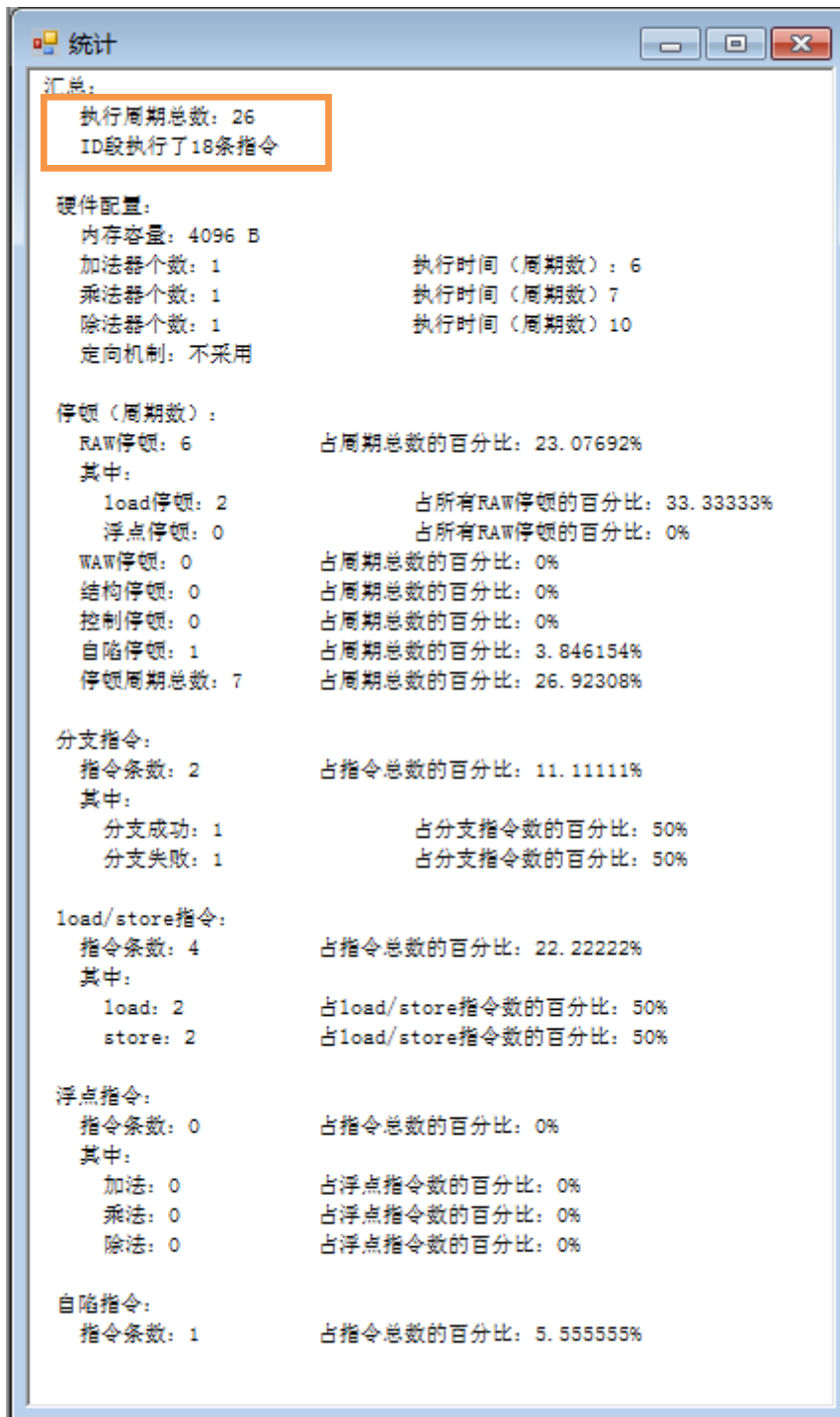
⑨执行该程序。观察其时钟周期图。



⑩记录执行该程序所用的总时钟周期数。

总时钟周期数：26

数据统计图：



⑩对比上述两种情况下的时钟周期图。

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/277013053125006061>