《计算机系统结构》

流水线及流水线中的冲突

学院：计算机学院（国家示范性软件学院）

班级：2018211314

姓名：李志毅

学号：2018211582

实验二 流水线及流水线中的冲突

一、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）加深对数据冲突和资源冲突的理解，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。

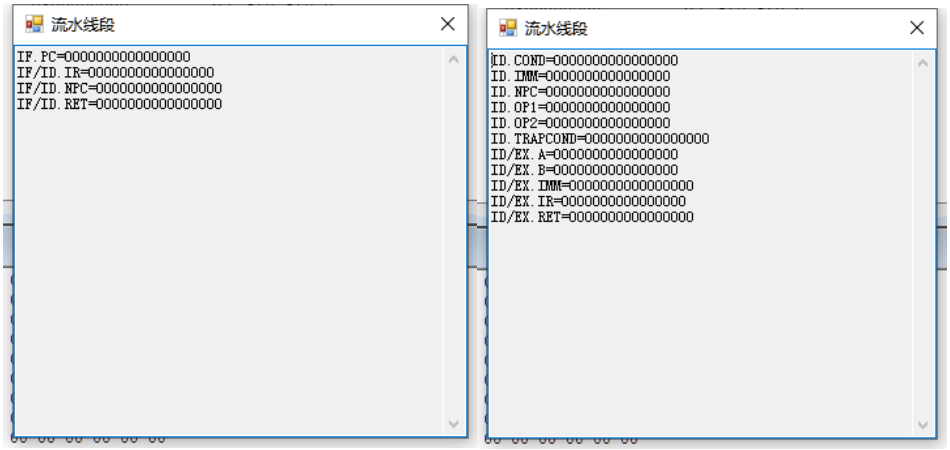
二、实验环境

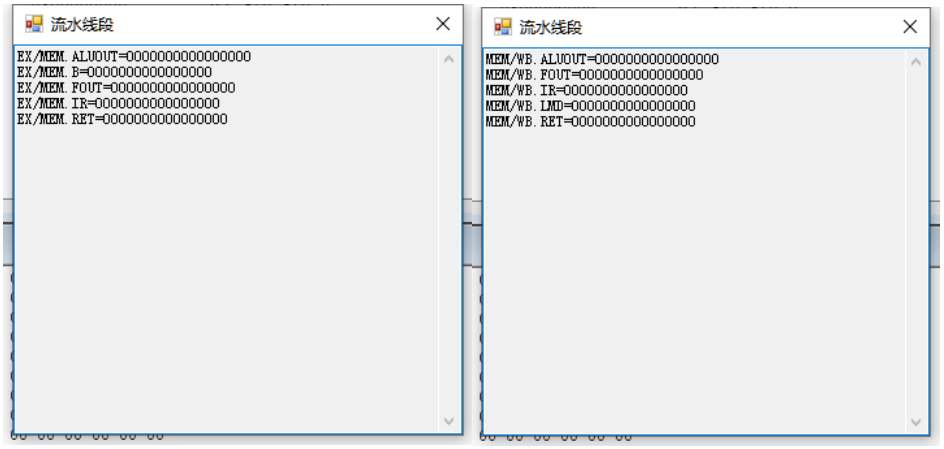
指令级和和流水线操作级模拟器 MIPSsim

三、实验步骤

1. 启动 MIPSsim

2.鼠标双击各段，查看各流水寄存器的内容





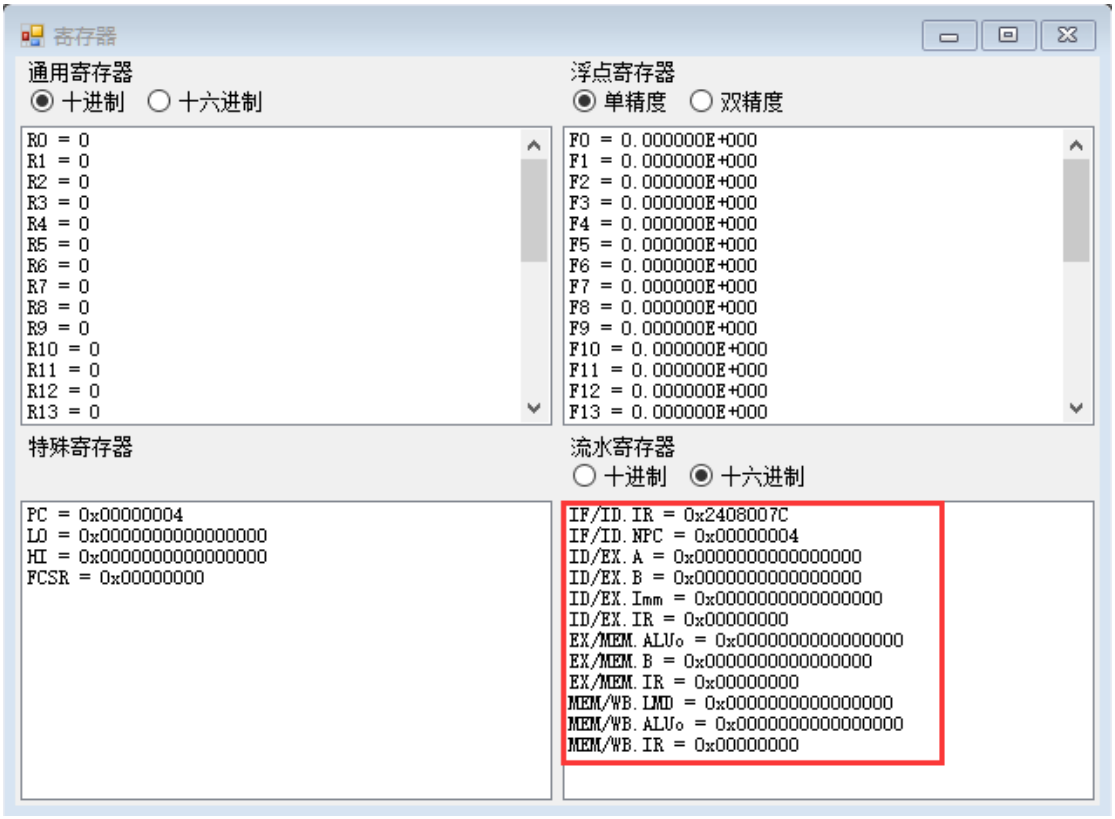
寄存器	含义
IF/ID.IR	流水段 IF 与 ID 之间的指令寄存器
IF/ID.NPC	流水段 IF 与 ID 之间的下一指令程序计数器
ID/EX.A	流水段 ID 与 EX 之间的第一操作数寄存器
ID/EX.B	流水段 ID 与 EX 之间的第二操作数寄存器
ID/EX.Imm	流水段 ID 与 EX 之间的立即数寄存器
ID/EX.IR	存放从 IF/ID.IR 传过来的指令
EX/MEM.ALUOUT	流水段 EX 与 MEM 之间的 ALU 计算结果寄存器
EX/MEM.IR	存放从 ID/EX.IR 传过来的指令
MEM/WB.LMD	流水段 MEM 与 WB 之间的数据寄存器，用于存放从存储器 读出的数据
MEM/WB.ALUOUT	存放从 EX/MEM.ALUo 传过来的计算结果
MEM/WB.IR	存放从 EX/MEM.IR 传过来的指令

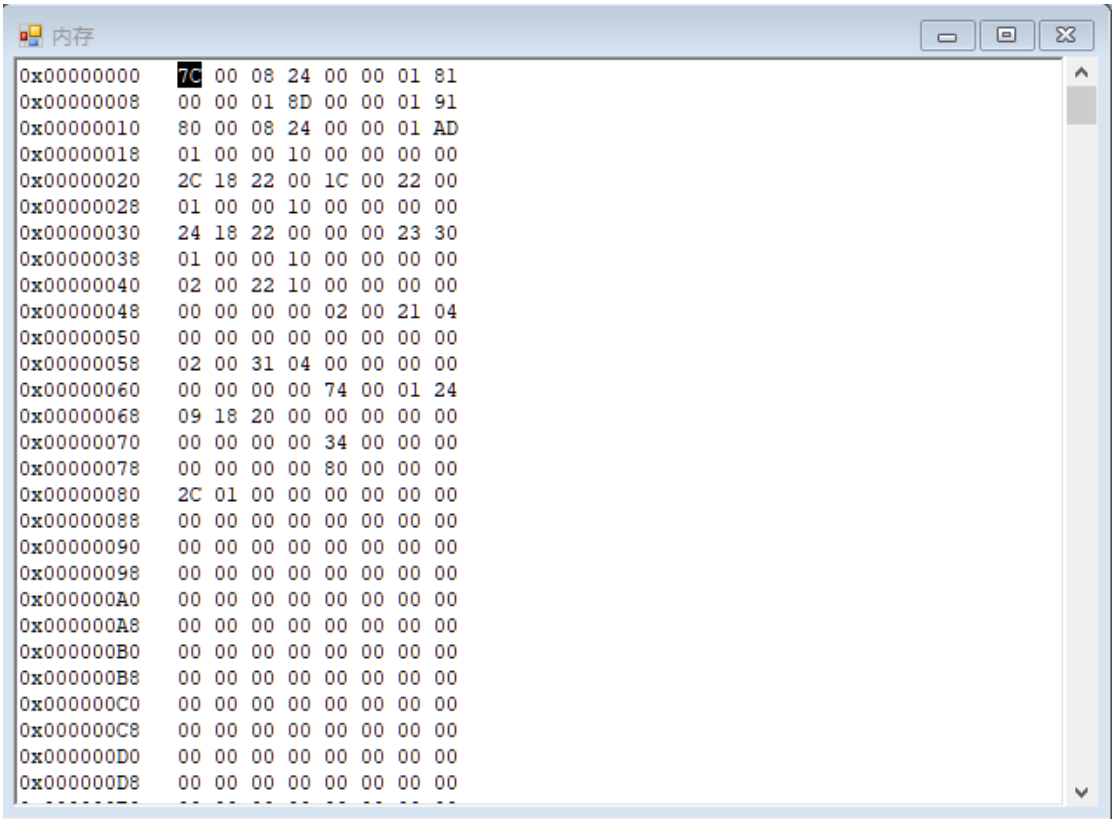
3.载入样例程序，分别以单步执行一个周期、执行多个周期、连续执行、设置断点等方式运行程序，观察程序的执行情况。观察 CPU 中寄存器和存储器内容的变化，特别是流水寄存器内容的变化。

载入 alltest.s 样例文件，首先选择单步执行一个周期的方式

3.1 单步执行一个周期

按下一次：

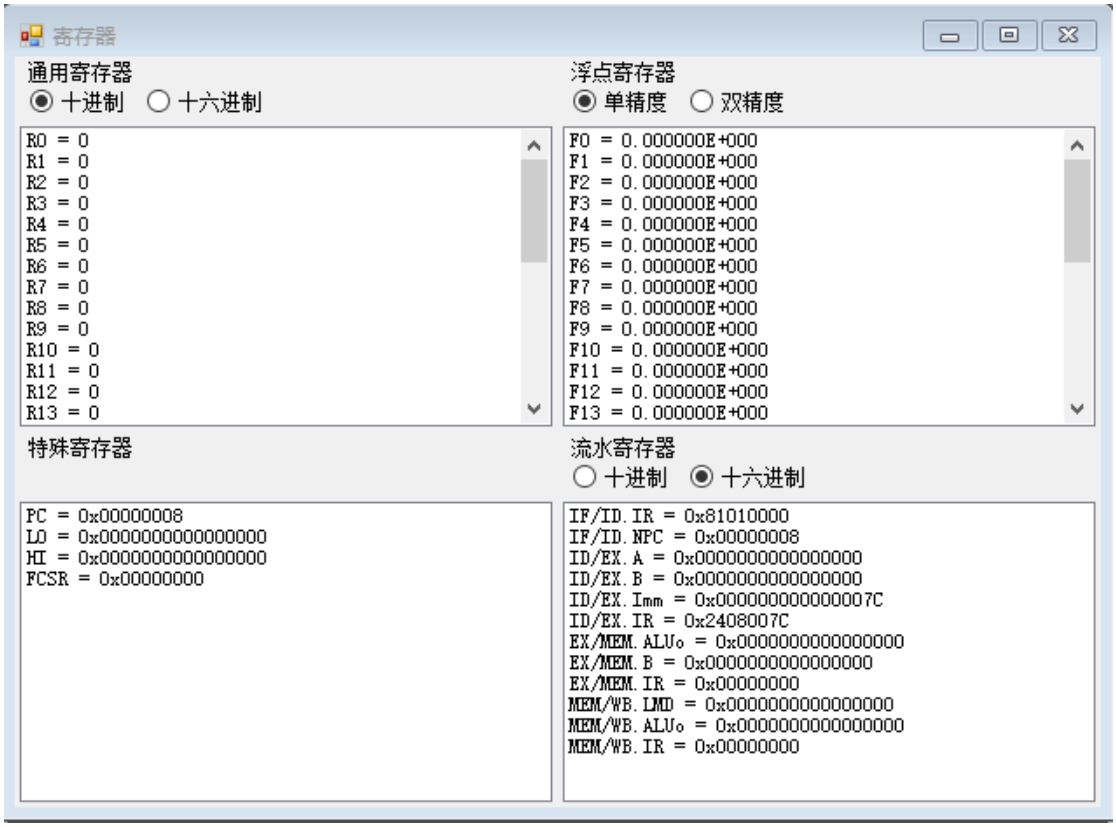




按下一次后，IF/ID.IR 的值变为**0x2408007C**，IF/ID.NPC 的值变为**0x00000004**，说明下一指令程序计数器值即 PC 值为 0x4，流水段 IF 与 ID 之间的指令寄存器

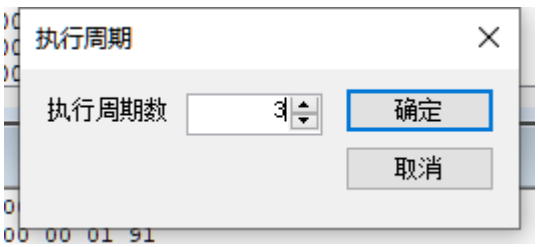
的值为 **0x2408007C**

再次按下一次后，可以看到 IF/ID.IR 的值变为 **0x81010000**，IF/ID.NPC 的值变为 **0x00000008**，ID/EX.Imm 的值变为 **0x7C**，即 **124**。说明立即数寄存器的值为 124，ID/EX.IR 中存放值 **0x2408007C**，为上一周期从 IF/ID.IR 传过来的指令。之后按下单步执行一个周期后，会使得程序流水向后推进一个周期，引起寄存器的变化。就不一一分析所有过程了。

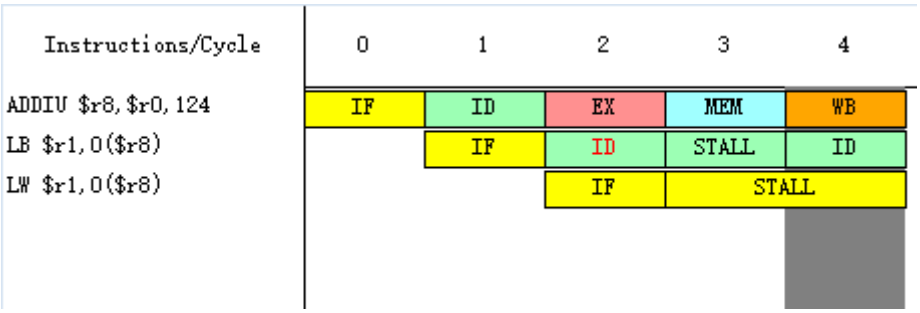


3.2 执行多个周期

设置执行多个周期的值为 3



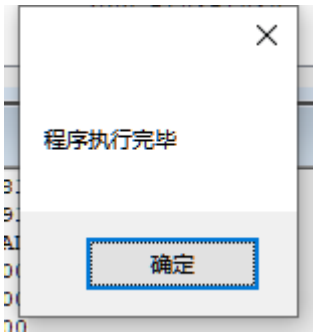
观察流水寄存器内容的值：



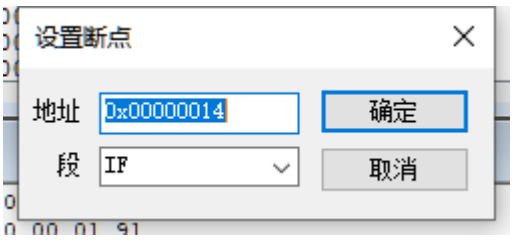
除了 PC 和 IR 在变化外，可以看到 EX/MEM.ALUOUT 中的值变为 **0x7C**，EX/MEM.IR 中的值为 **0x2408007C**，MEM/WB.ALUOUT 的值变为 **0x7C**，MEM/WB.IR 中的值变为 **0x2408007C**，即 ALU 计算结果为 0x7C，指令已传递到 MEM/WB.IR

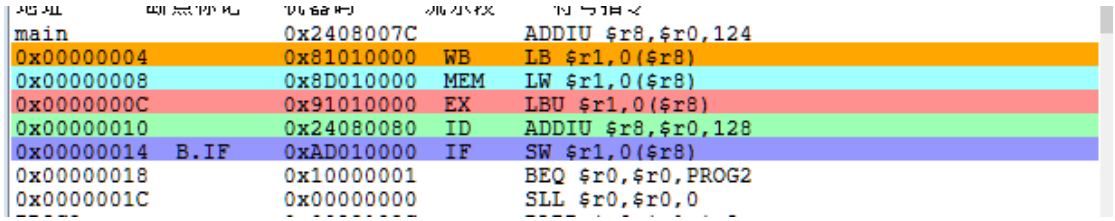
3.3 连续执行

直接执行完所有指令

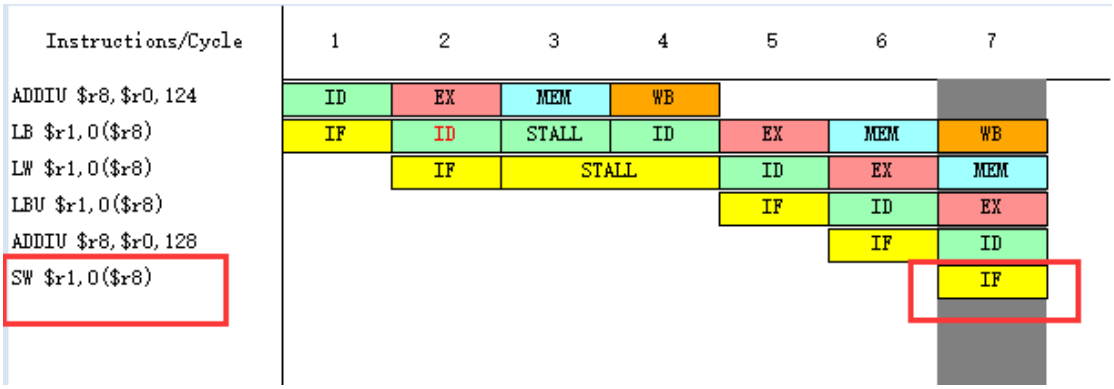


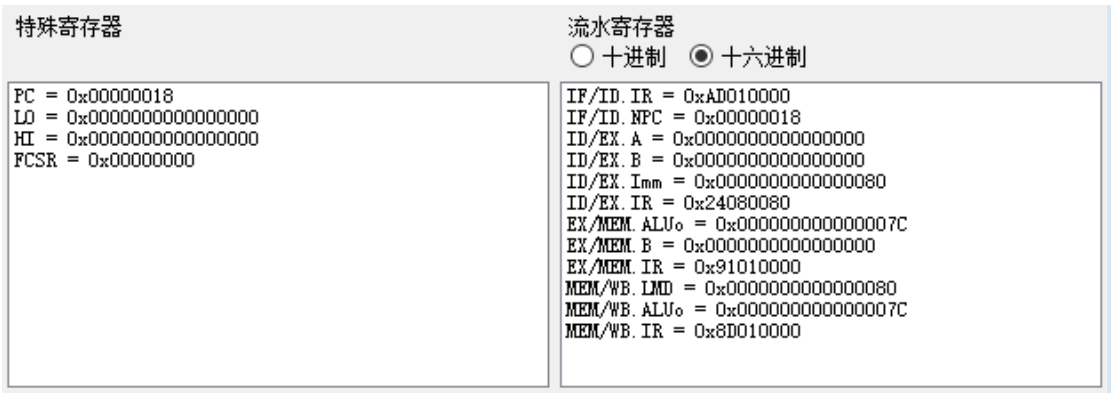
3.4 设置断点



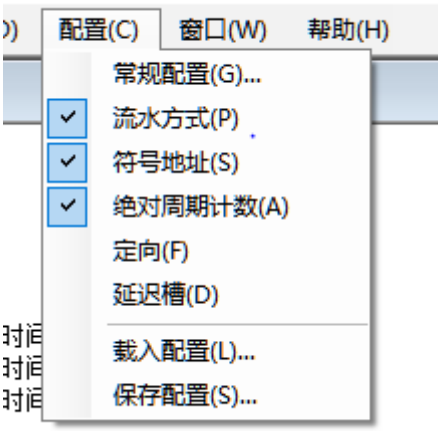


设置断点后，点击连续执行，可以看到程序在执行到断点处时停止

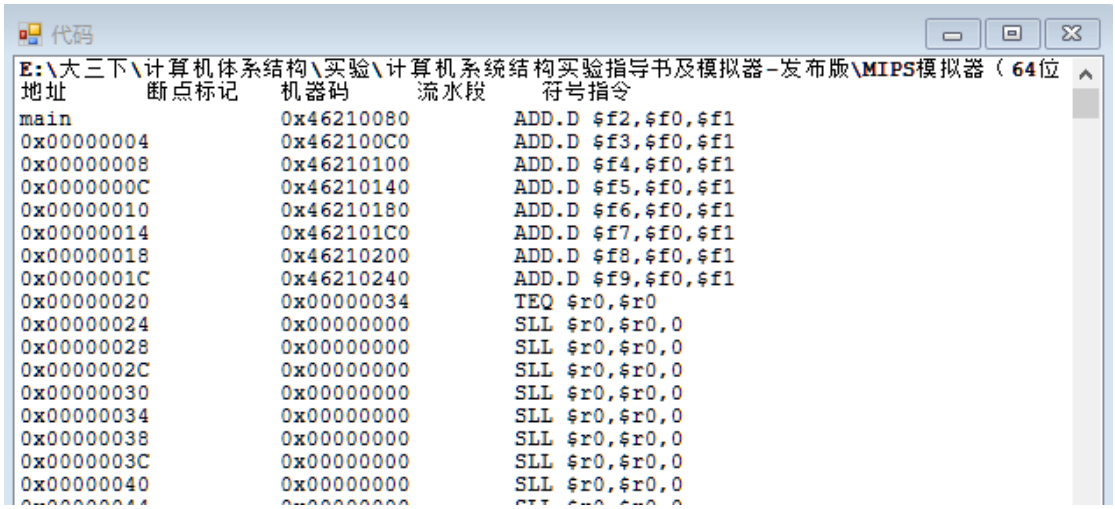




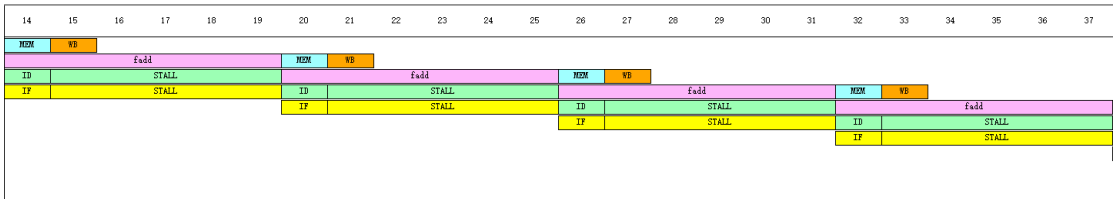
4. 选择配置菜单中的“流水方式”选项，使模拟器工作于流水方式下



5. 观察程序下流水方式下的执行情况
6. 观察和分析结构冲突对 CPU 性能的影响
6.1 加载 structure_hz.s



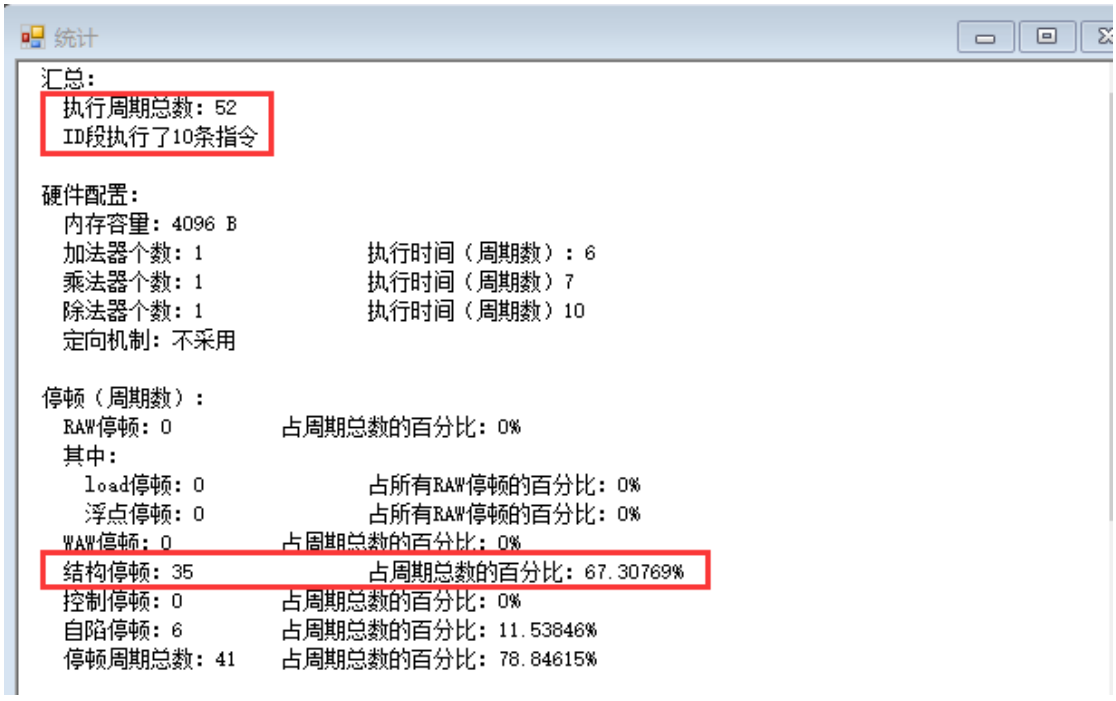
6.2 执行该程序，找出存在结构冲突的指令对以及导致结构冲突的部件



如图所示，可以看出引起结构冲突的指令为 fadd 指令，导致冲突的部件为浮点加法器部件。

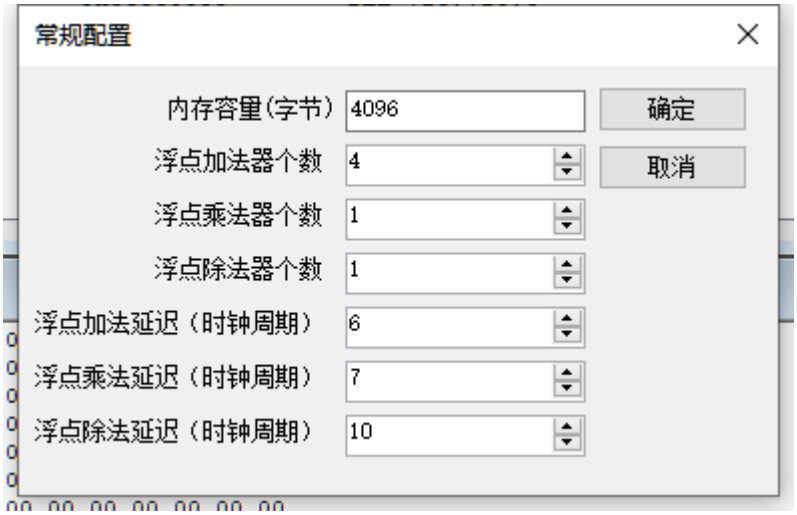
6.3 记录由结构冲突引起的停顿周期数，计算停顿周期数占总执行周期数的百分比

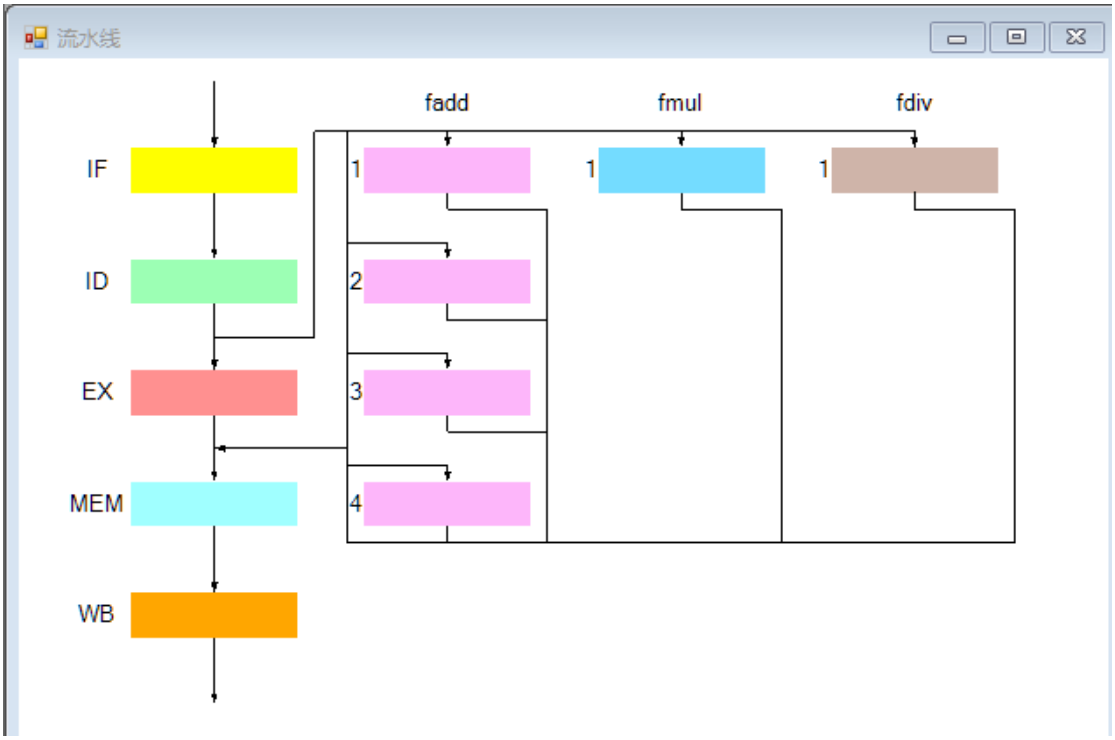
通过观察统计界面可以得到：



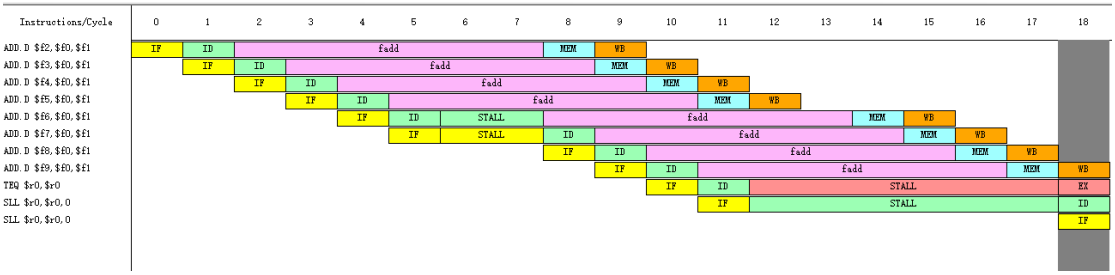
因为结构冲突而停顿的周期数为 35，占总执行周期数的 67.30769%

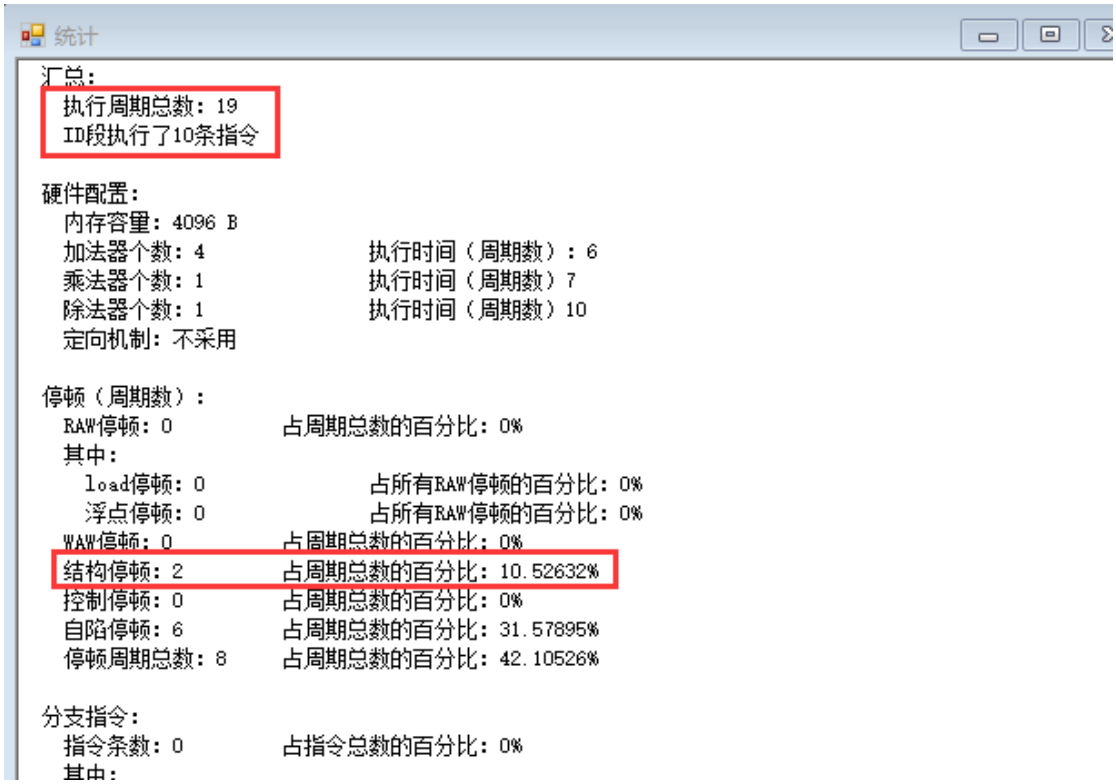
6.4 把浮点加法器的个数改为 4 个





6.5 重复 6.1-6.3 的步骤





执行周期总数为 19，结构停顿周期数为 2，占周期总数的 10.526332%

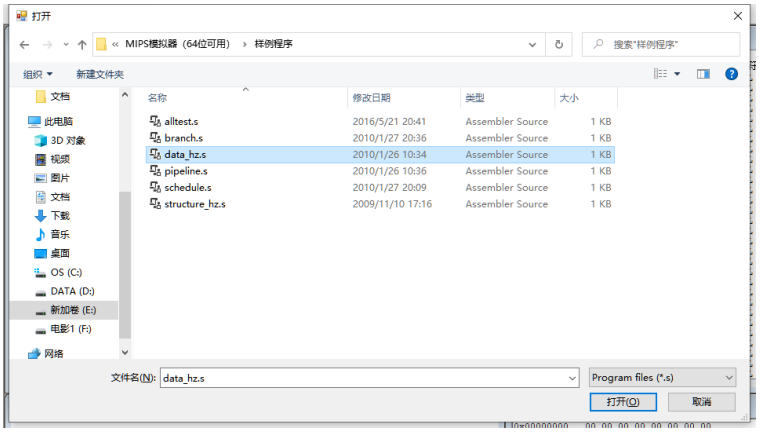
6.6 分析结构冲突对 CPU 性能的影响，讨论解决结构冲突的方法

由上面的实验可以看出，结构冲突会导致 CPU 性能下降，影响执行周期总数，导致流水上产生多于的空操作，严重影响 CPU 的性能。解决结构冲突的办法有很多，在本次实验中是通过增加浮点加法器的个数来解决的，此外还可以通过暂停一个时钟周期，取后一条指令的操作，或设置两个独立的存储器分别存放操作数和指令，以及采取指令预存技术等方法来解决结构冲突。

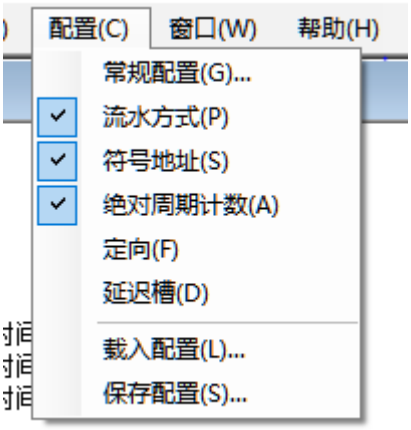
7. 观察数据冲突并用定向技术来减少停顿

7.1 全部复位

7.2 加载 data_hz.s

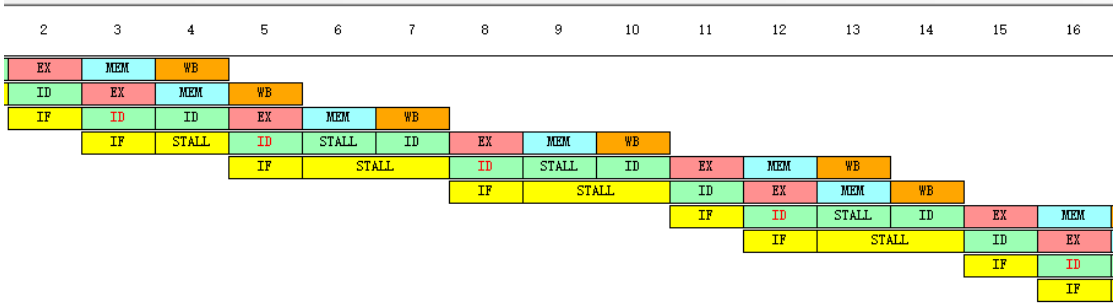


7.3 关闭定向功能

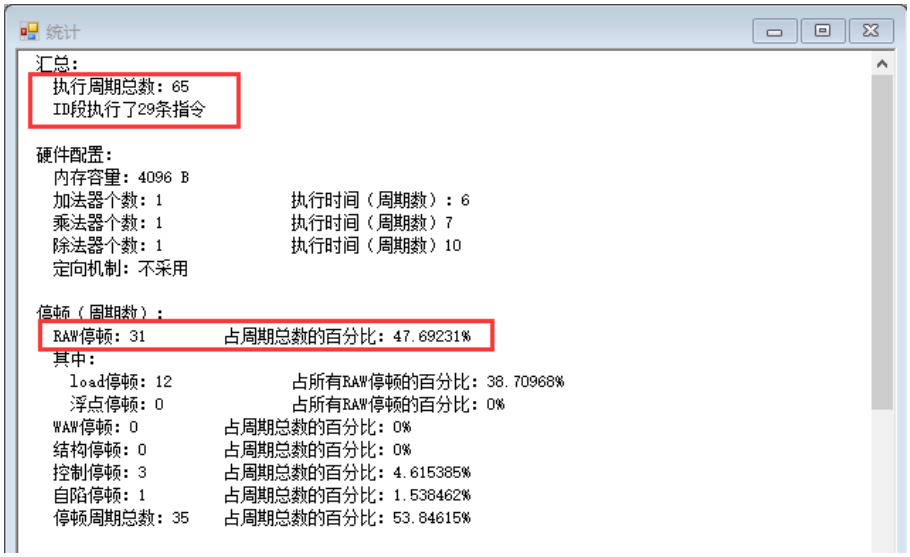


7.4 用单步执行一个周期的方式执行该程序，观察时钟周期图，列出什么时刻发生了 RAW 冲突

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。



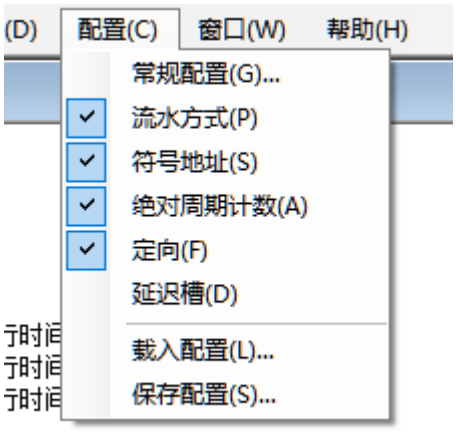
7.5 记录数据冲突引起的停顿周期数以及程序执行的总时钟周期数，计算停顿周期数占总执行周期数的百分比



执行周期总数 65 个，RAW 冲突 31 个，占总执行周期数的 47.69231%

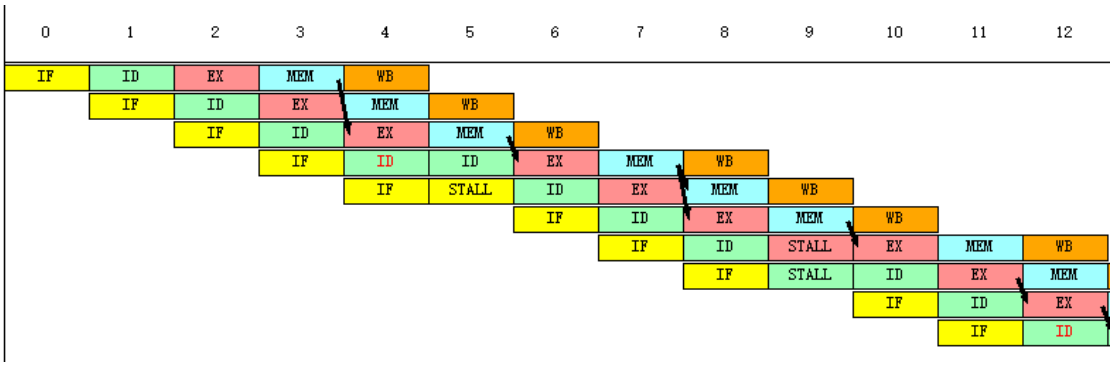
7.6 复位 CPU

7.7 打开定向功能

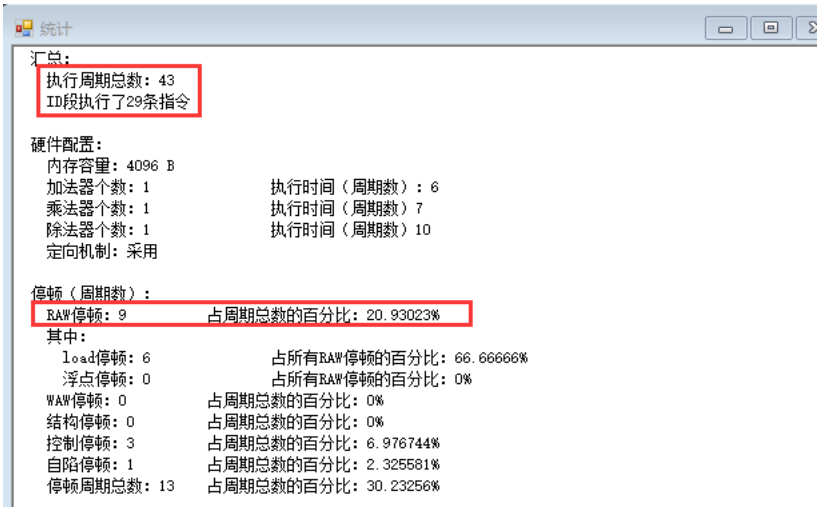


7.8 用单步执行一个周期的方式执行该程序，查看时钟周期图，列出什么时刻发生了 RAW 冲突，并与步骤 3）的结果比较。

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。



7.9 记录数据冲突引起的停顿周期数以及程序执行的总周期数。计算采用定向以后性能 比原来提高多少。



执行周期总数 43 个，结构停顿周期数 9 个，占总执行周期数的 20.93023%

性能提高了近 65/43=1.51 倍

四、实验问题与心得

本次实验使用指令级和流水线操作级模拟器 MIPSsim 分析了示例程序流水线过程，加深了我对于计算机流水线基本概念的理解，理解了 MIPS 结构是如何使用 5 段流水线来实现的，理解了各段的功能和基本操作，加深了我对数据冲突和结构冲突的理解，以及采用定向技术解决数据冲突带来的好处和性能的提升，进一步掌握了解决数据冲突的方法，掌握了如何应用定向技术来减少数据冲突引起的停顿。本次实验让我受益匪浅。