北京郵電大學

实验报告

题目： 实验 2 流水线及流水线中的冲突

计算机系统结构实验小组成员信息			
班级	姓名	学号	学院
2020211314	王小龙	2020211502	计算机学院
2020211314	闻奔放	2020211504	计算机学院
2020211314	黄洪健	2020211371	计算机学院

注：红色标出的成员为本次实验的完成者

2023 年 4 月 11 日

一、实验目的

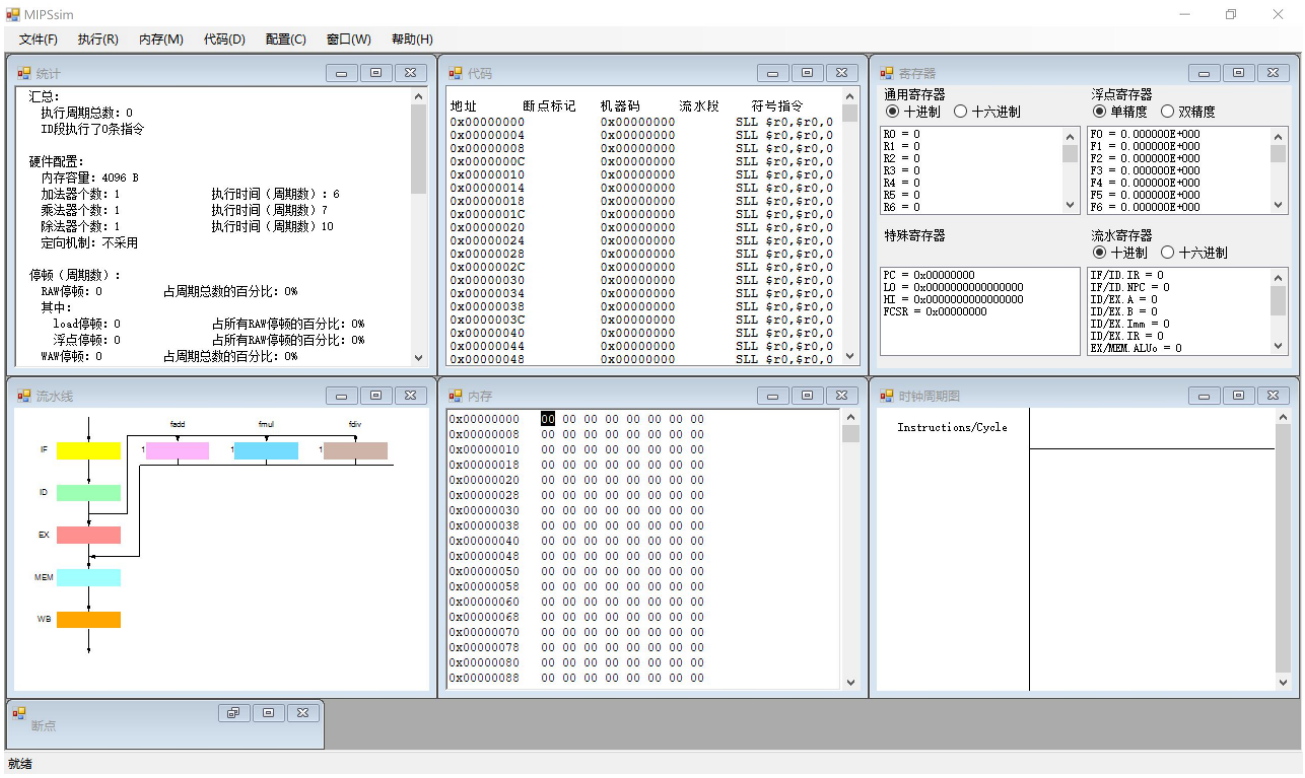
（1）加深对计算机流水线基本概念的理解。
（2）理解 MIPS 结构如何用 5 段流水线来实现，理解各段的功能和基本操作。
（3）加深对数据冲突和资源冲突的理解，理解这两类冲突对 CPU 性能的影响。
（4）进一步理解解决数据冲突的方法，掌握如何应用定向技术来减少数据冲突引起的停顿。

二、 实验环境

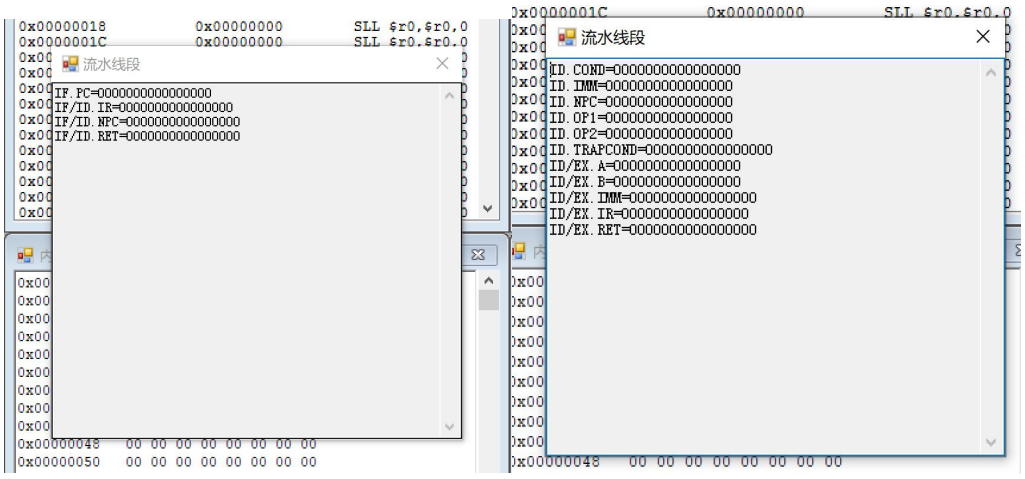
指令级和和流水线操作级模拟器 MIPSsim。

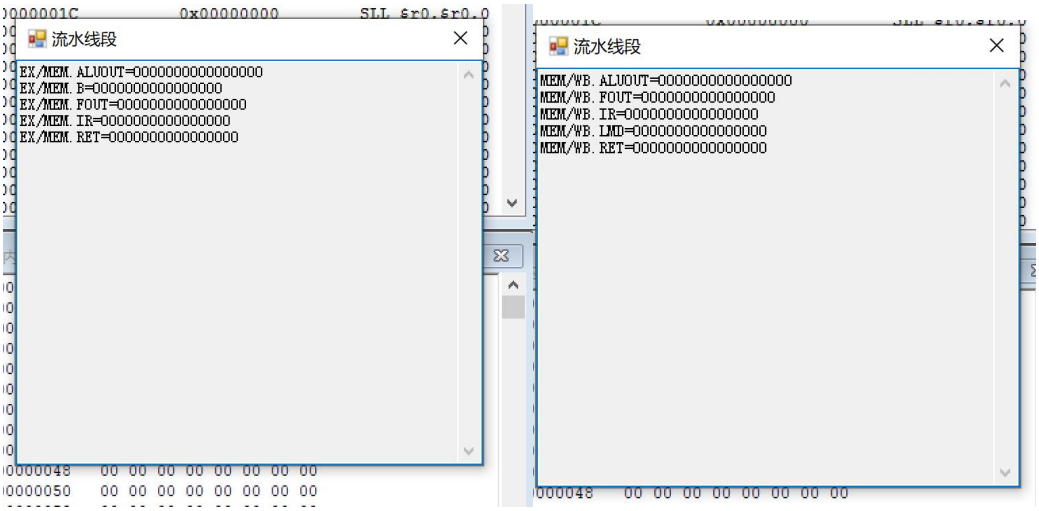
三、实验内容和步骤

3.1 启动 MIPSsim



3.2 通过鼠标双击各段，可看到各流水寄存器的内容，从而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。





通过上述操作，可以总结如下表格来说明各流水寄存器及其含义：

流水寄存器	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

3.3 载入一个样例程序，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 中寄存器和存储器内容的变化，特别是流水寄存器内容的变化。

选择载入 alltest.s 样例程序：

3.3.1 单步执行一个周期

在“执行”菜单栏下，选择“单步执行一个周期”，然后可以观察到如下变化：

可以看到，流水寄存器 IF/ID.IR 的值变为 0x2408007C，IF/ID.NPC 的值变为 0x00000004，下一指令程序计数器值（PC 值）为 0x4

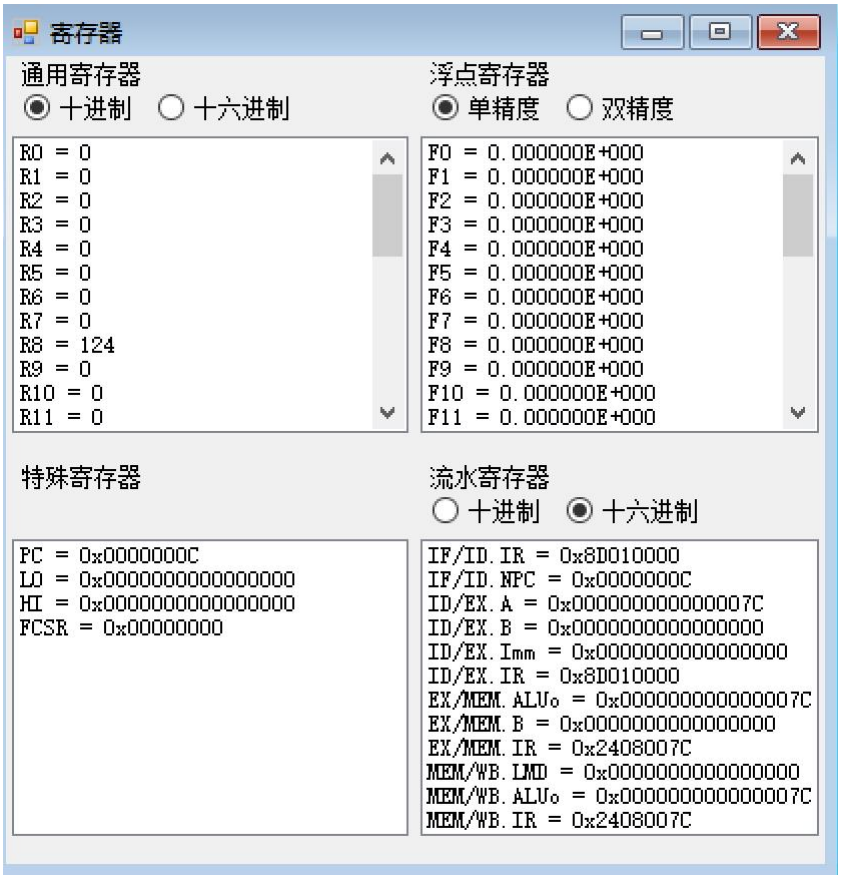
再单步执行一个周期后，可以看到如下变化：

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。ID/EX.IR 中的值为 0x2408007C，这是从上个一周期的 IF/ID.IR 传过来的指令。

之后，重复继续单步执行一个周期，与上述情况类似，不在列举。

3.3.2 执行多个周期

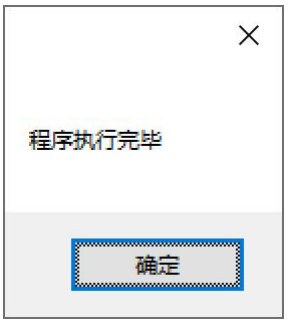
首先全部复位，然后重新载入 alltest.s 样例程序。
在菜单栏选择执行多个周期，设置执行周期数为 5，可以看到如下变化：



可以看到，IF/ID.IR 中的值变为 0x8D010000，IF/ID.NPC 的值变为 0xC，ID/EX.A 的值变为 0x7C，EX/MEM.ALUOUT 的值变为 0x7C，EX/MEM.IR 的值变为 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，指令传递到了 MEM/WB.IR。

3.3.3 连续执行

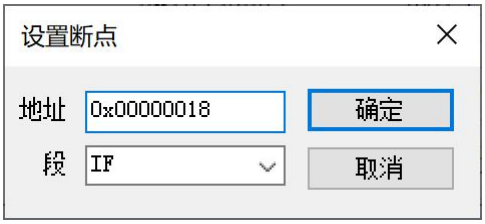
在菜单栏选择连续执行，程序显示如下界面：



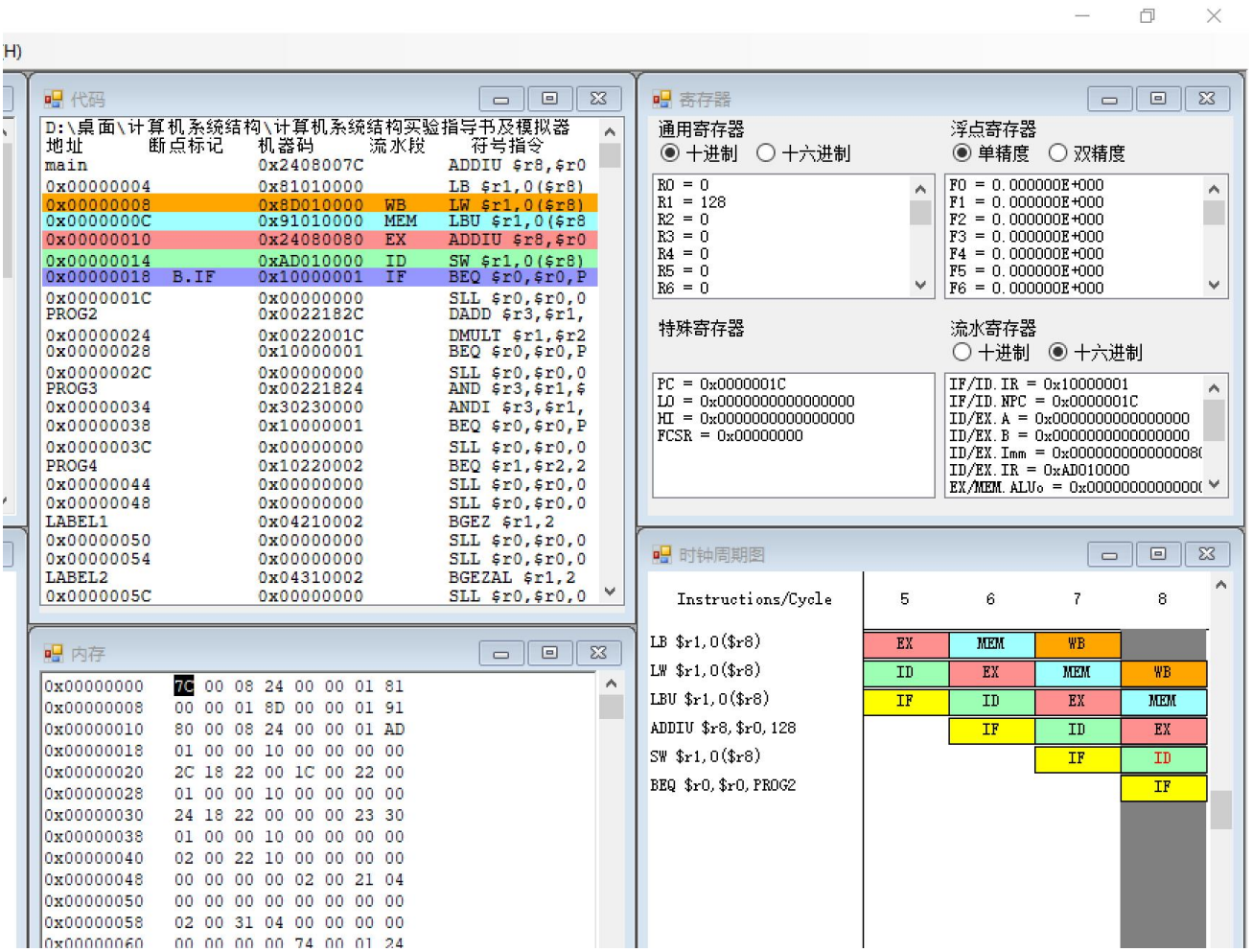
流水寄存器的值如下：

3.3.4 设置断点

通过如下界面设置断点：



选择连续执行，可以看到程序在断点处停止运行：

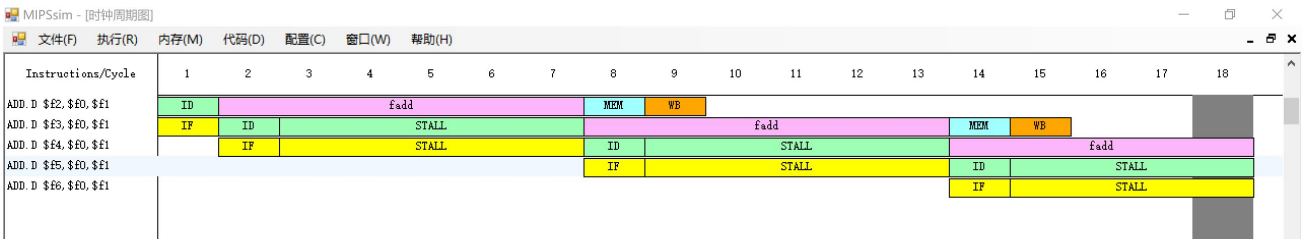


3.4 在配置菜单中选择“流水方式”，使模拟器工作于流水方式下。观察程序在流水方式下的执行情况。和上述类似，不在列举。

3.5 观察和分析结构冲突对 CPU 性能的影响

3.5.1 加载 structure_hz.s 样例程序

3.5.2 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下：



从上图可以看到，存在结构冲突的指令为 fadd 指令，导致冲突的部件为浮点加法器部件。

3.5.3 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

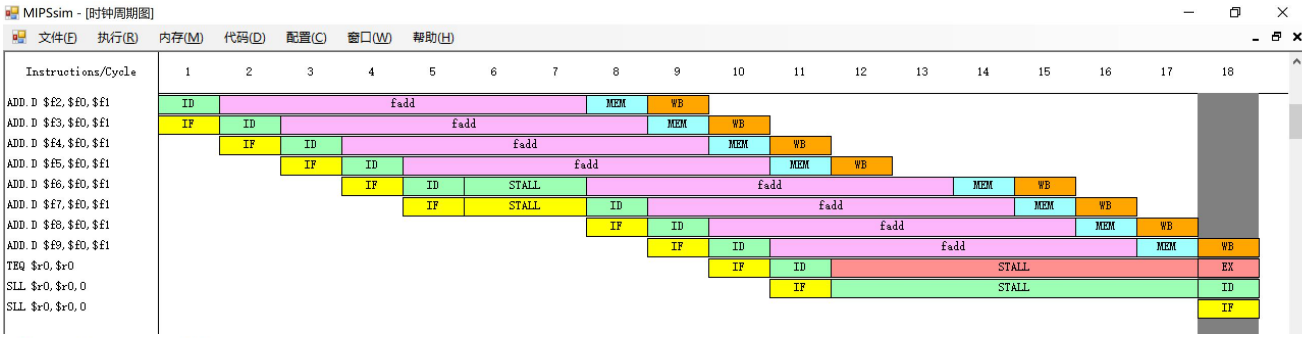
在统计窗口可以看到如下信息：

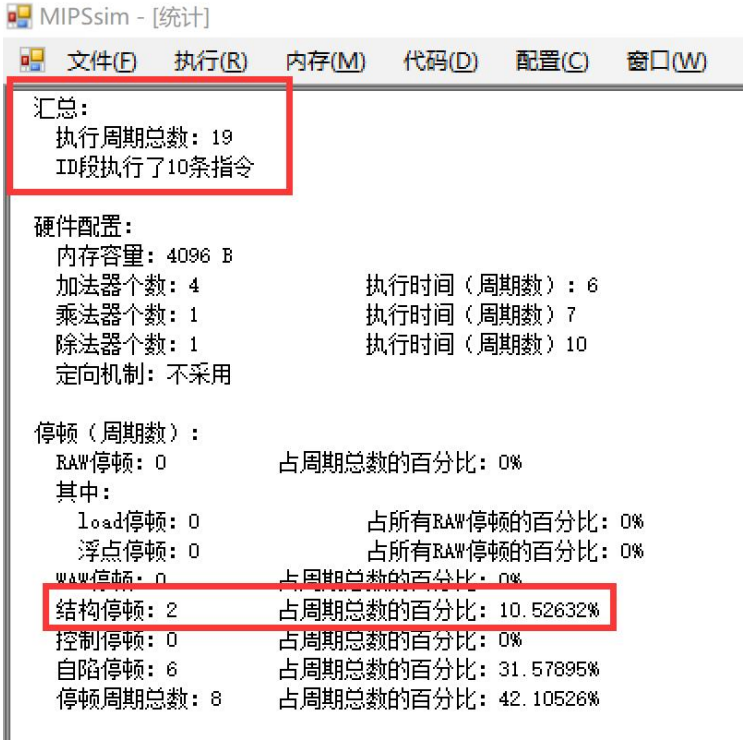
```
汇总：
  执行周期总数：52
  ID段执行了10条指令

硬件配置：
  内存容量：4096 B
  加法器个数：1                执行时间（周期数）：6
  乘法器个数：1                执行时间（周期数）7
  除法器个数：1                执行时间（周期数）10
  定向机制：不采用

停顿（周期数）：
  RAW停顿：0          占周期总数的百分比：0%
  其中：
    load停顿：0               占所有RAW停顿的百分比：0%
    浮点停顿：0               占所有RAW停顿的百分比：0%
  WAW停顿：0          占周期总数的百分比：0%
  结构停顿：35                占周期总数的百分比：67.30769%
  控制停顿：0          占周期总数的百分比：0%
  自陷停顿：6          占周期总数的百分比：11.53846%
  停顿周期总数：41     占周期总数的百分比：78.84615%

分支指令：
  指令条数：0          占指令总数的百分比：0%
  其中：
    分支成功：0               占分支指令数的百分比：0%
    分支失败：0               占分支指令数的百分比：0%

load/store指令：
  指令条数：0          占指令总数的百分比：0%
  其中：
    load：0           占load/store指令数的百分比：0%
    store：0          占load/store指令数的百分比：0%

浮点指令：
  指令条数：8          占指令总数的百分比：80%
  其中：
    加法：8           占浮点指令数的百分比：100%
    乘法：0           占浮点指令数的百分比：0%
    除法：0           占浮点指令数的百分比：0%
```

从上述红框内可以看到由结构冲突而停顿的周期数为 35，占总执行周期数的 67.30769%

3.5.4 把浮点加法器的个数改为 4 个，再重复 1-3 的步骤

这里要注意，要先修改配置，即将浮点加法器的个数改为 4 个，之后再载入程序，否则会出错。





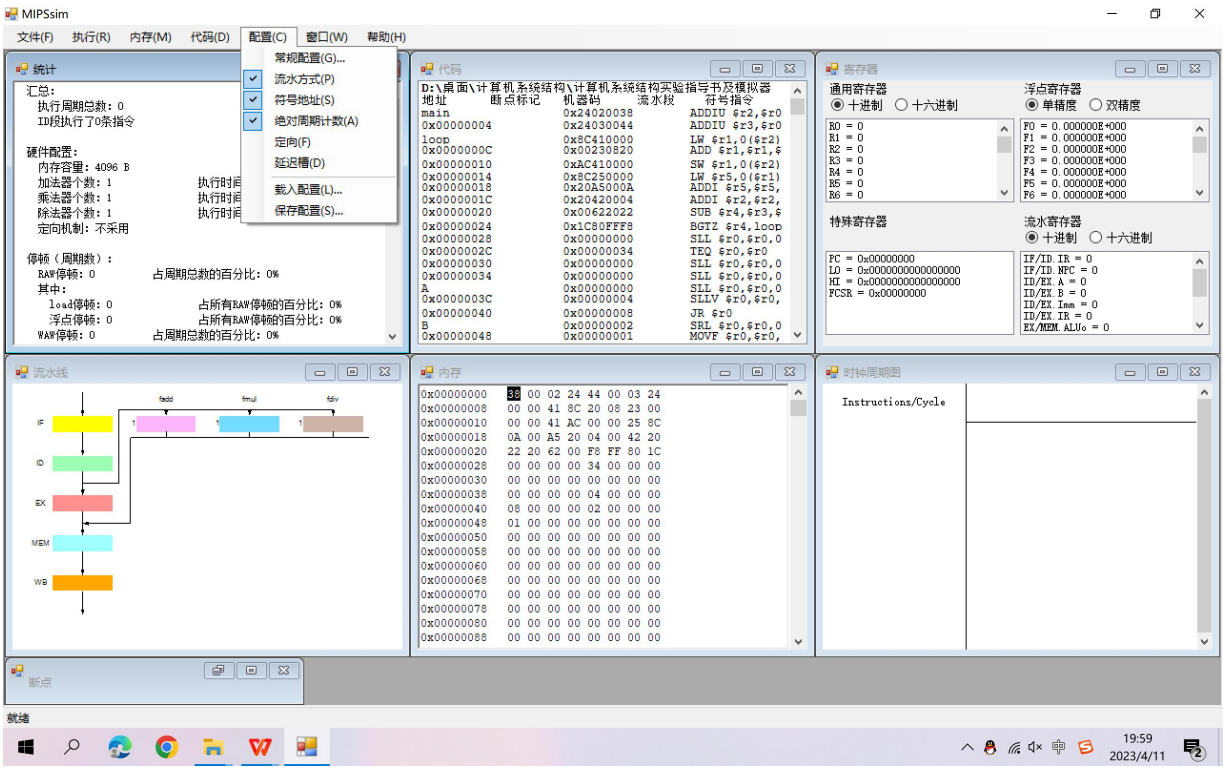
执行周期总数为 19，结构停顿周期数为 2，占周期总数的 10.526332%

3.5.5 分析结构冲突对 CPU 性能的影响，讨论解决结构冲突的方法

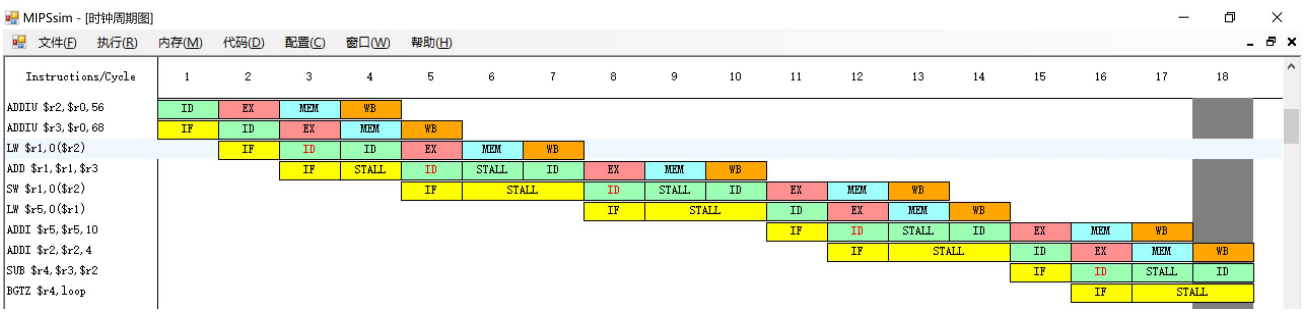
由上述内容可以看出，结构冲突会导致 CPU 性能下降，严重影响 CPU 的性能。
解决结构冲突的办法：
本次实验中采取了增加浮点加法器数量的办法，其他办法还有如设置两个独立的存储器分别存放操作数和指令，以及采取指令预存技术。

3.6 观察数据冲突并用定向技术来减少停顿

3.6.1 全部复位，加载 data_hz.s 样例程序
3.6.2 在“配置”菜单下选择取消“定向”，即关闭定向功能



3.6.3 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



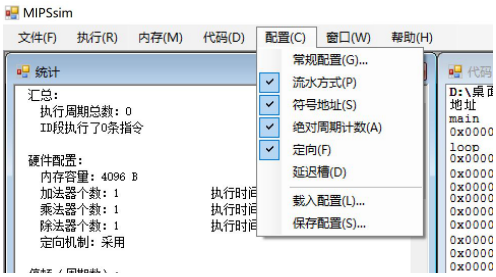
从上图可以看出，在周期 4, 6, 7, 9, 10, 13, 14, 17, 18, 20, 21, 25, 26, 28, 29, 32, 33, 36, 37, 39,

40, 44, 45, 47, 48, 51, 52, 55, 56, 58, 59 时，发生了 RAW 冲突。

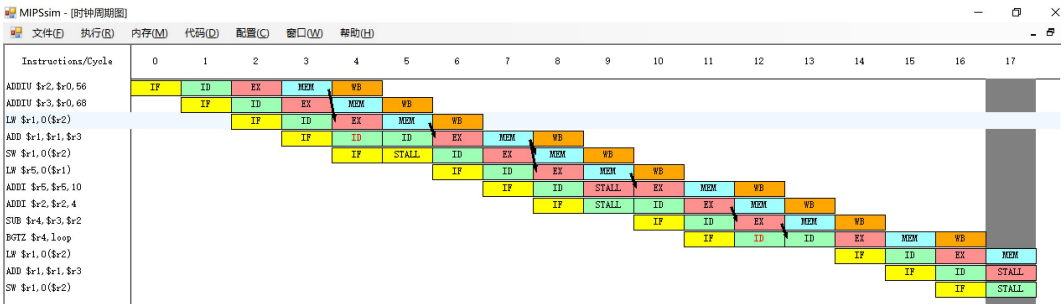
3.6.4 记录数据冲突引起的停顿周期数以及程序执行的总时钟周期数，计算停顿时钟周期数占总执行周期数的百分比

从上图看可以看出，执行周期总数 65 个，RAW 冲突 31 个，占总执行周期数的 47.69231%

3.6.5 复位 CPU，打开定向功能。

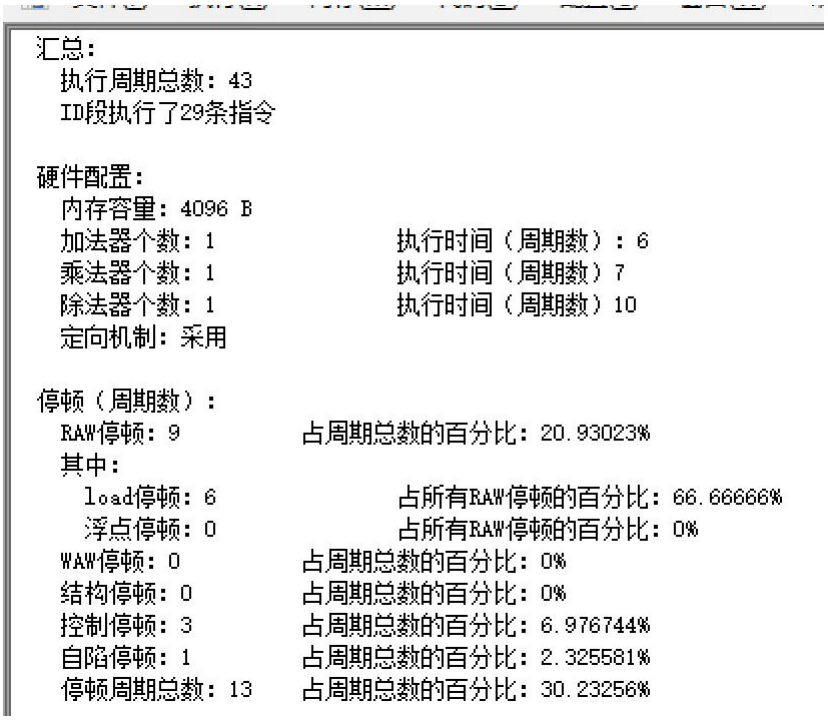


3.6.6 用单步执行一个周期的方式执行该程序，查看时钟周期图，列出什么时刻发生了 RAW 冲突，并与步骤 3）的结果比较



单步执行结果如上图，在周期 5,9,13,17,21,25,29,33,37 发生了 RAW 冲突，与上面的比较，可以看出，使用定向技术，可以很大程度上减少 RAW 冲突数目。

3.6.7 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



数据冲突引起的停顿周期数 9 个，执行周期总数 43 个，占总执行周期数的 20.93023%

性能比原来提高了 65/43=1.51 倍

四、总结体会

通过本次实验，加深了我对计算机流水线基本概念的理解，帮助我更好的理解了 MIPS 结构如何用 5 段流水线来实现，理解了各段的功能和基本操作。同时也加深了我对数据冲突和资源冲突的理解，理解了这两类冲突对 CPU 性能的影响。此外，也让我进一步理解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。总之，收获颇多。