

**计算机系统结构**



**流水线及流水线中的冲突**

**姓 名 鄭毓恒**

**学 院 计算机学院**

**专 业 计算机科学与技术**

**班 级 2020211302**

**学 号 2020211262**

**任课教师 邝坚**

**2023年 4 月**

**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。

**实验平台**

指令级和流水线操作级模拟器 MIPSsim。

**实验内容和步骤**

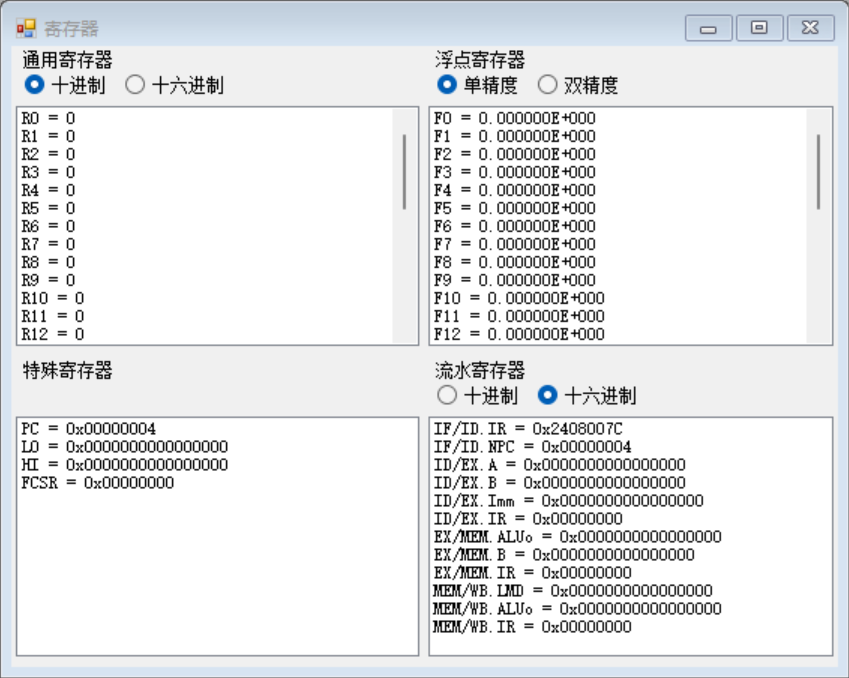
双击流水线各段，看到各流水线寄存器的内容。

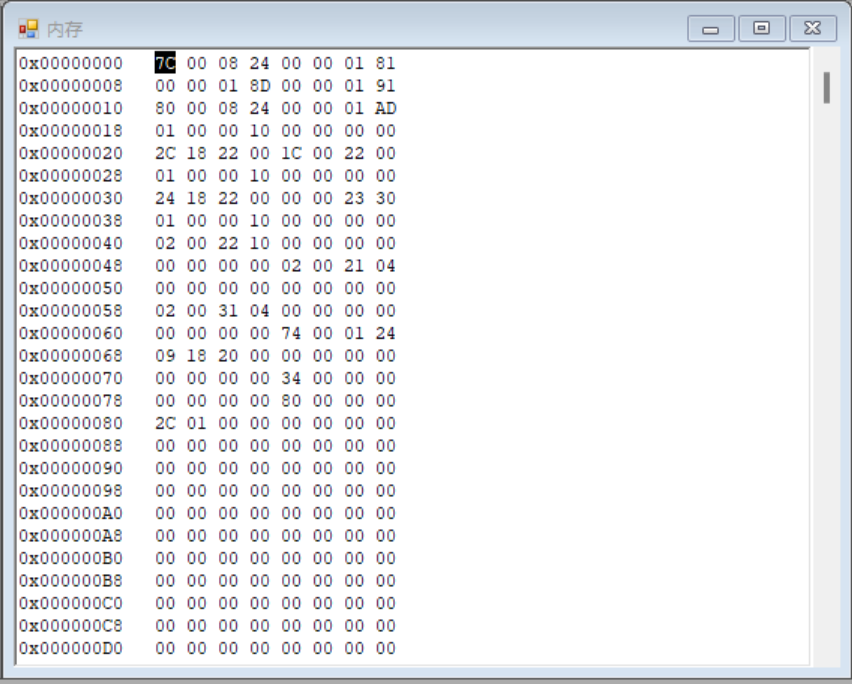
|  |  |
| --- | --- |
| 流水线段 | 内容 |
| IF |  |
| ID |  |
| EX |  |
| MEM |  |
| WB |  |

各流水线寄存器的含义如下

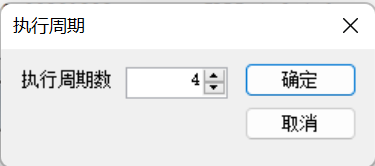
|  |  |
| --- | --- |
| 流水线寄存器 | 含义 |
| IF/ID.IR | 流水段 IF 与 ID 之间的指令寄存器 |
| IF/ID.NPC | 流水段 IF 与 ID 之间的下一指令程序计数器 |
| ID/EX.A | 流水段 ID 与 EX 之间的第一操作数寄存器 |
| ID/EX.B | 流水段 ID 与 EX 之间的第二操作数寄存器 |
| ID/EX.Imm | 流水段 ID 与 EX 之间的立即数寄存器 |
| ID/EX.IR | 存放从 IF/ID.IR 传过来的指令 |
| EX/MEM.ALUo | 流水段 EX 与 MEM 之间的 ALU 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.ALUo | 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 |

载入样例程序alltest.s，以单步执行一个周期的方式运行程序。一个周期后，IF/ID.IR的值变为0x2408007C，IF/ID.NPC的值变为0x00000004。说明IF段执行取指令操作后，得到指令0x2408007C，下一指令程序计数器PC值为0x4。

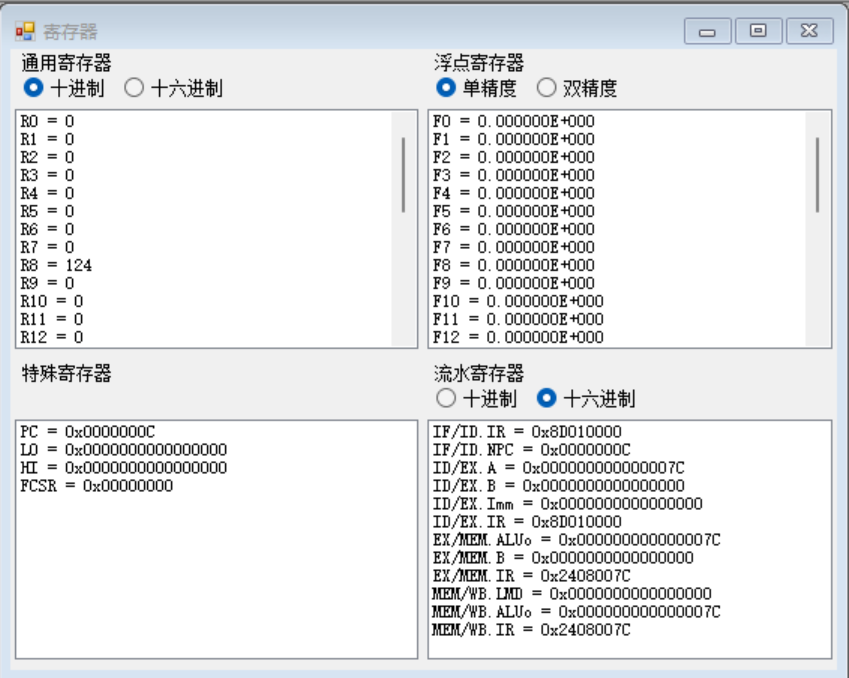




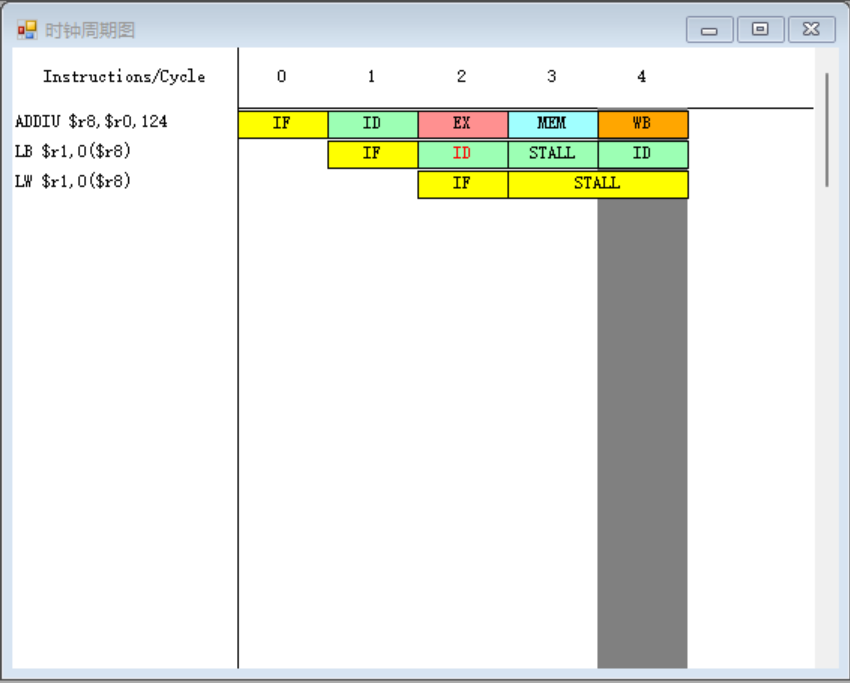
接着，执行4个周期。



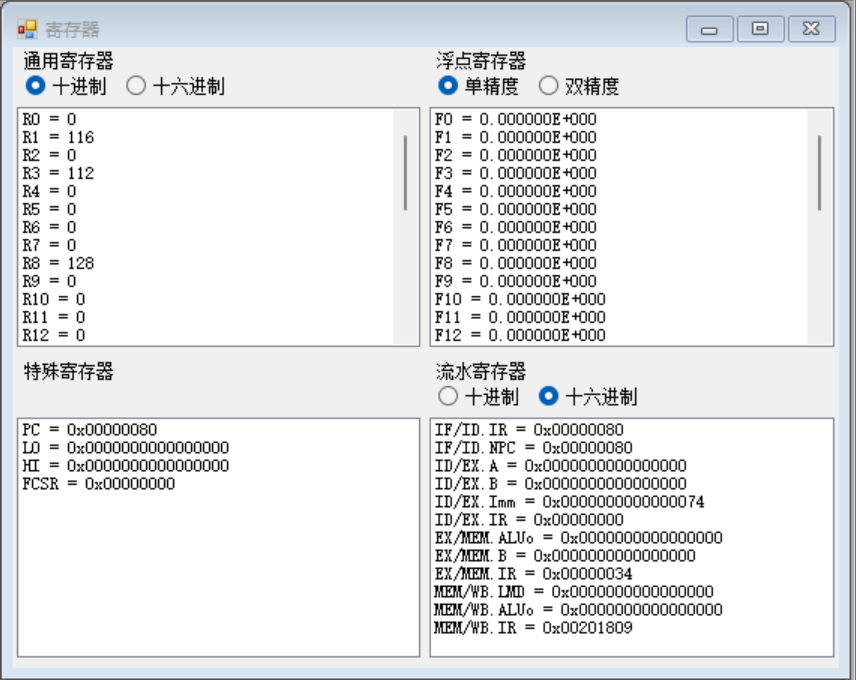
执行完成后，根据时钟周期图可知，当前流水线正在处理三条指令，分别在WB段、ID段和IF段。EX/MEM.IR和MEM/WB.IR寄存器的值是0x2408007C，证明这两段目前只处理过程序的第一条指令。EX/MEM.ALUo和MEM/WB.ALUo的值为0x7C，也就是第一条指令的结果是0x7C。



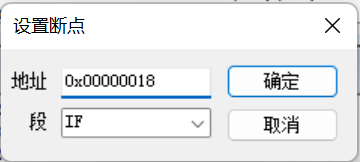
第一条指令更改了寄存器r8的值，而第二条指令需要寄存器r8的值。因此，第一条指令完成WB段后，第二条指令才会完成ID段操作，等待进入EX段执行。ID/EX.A的值是0x7C，也就是第一条指令的结果和寄存器r8的值，表示第二条指令在ID段完成处理。IF/ID.IR的值是0x8D010000，也就是第三条指令。



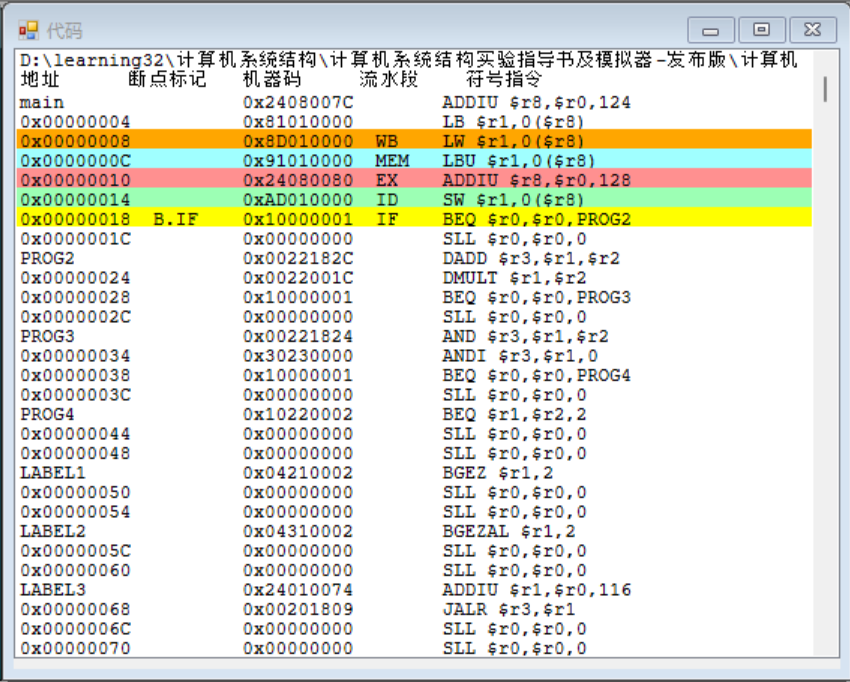
以连续执行的方式，执行剩下所有指令。



CPU复位，为程序设置断点。

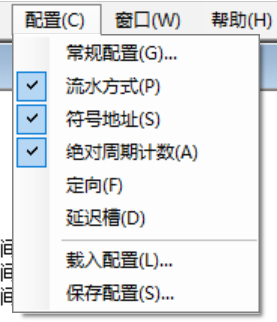


连续执行指令。当IF段执行到地址为0x18的指令，触发断点，程序暂停运行。

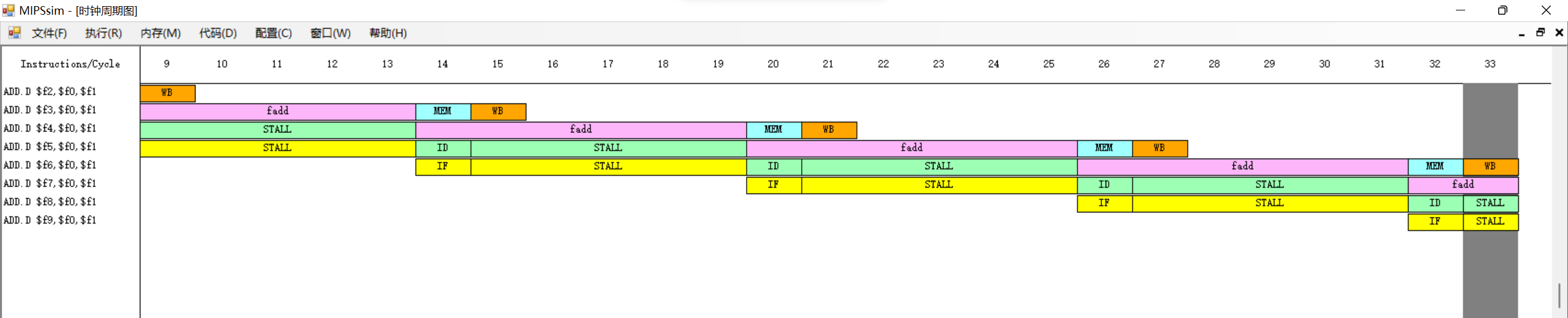




选择配置菜单中的“流水方式”选项，使模拟器工作于流水方式下。



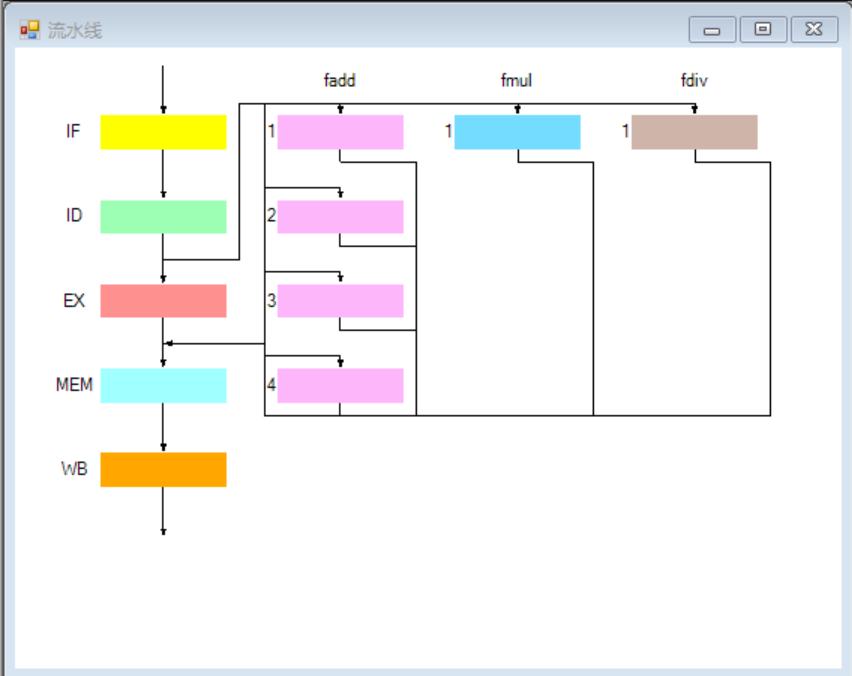
加载样例程序structure\_hz.s。观察时钟周期图，fadd指令需要六个时钟周期执行，存在结构冲突。导致冲突的部件为浮点加法器部件。



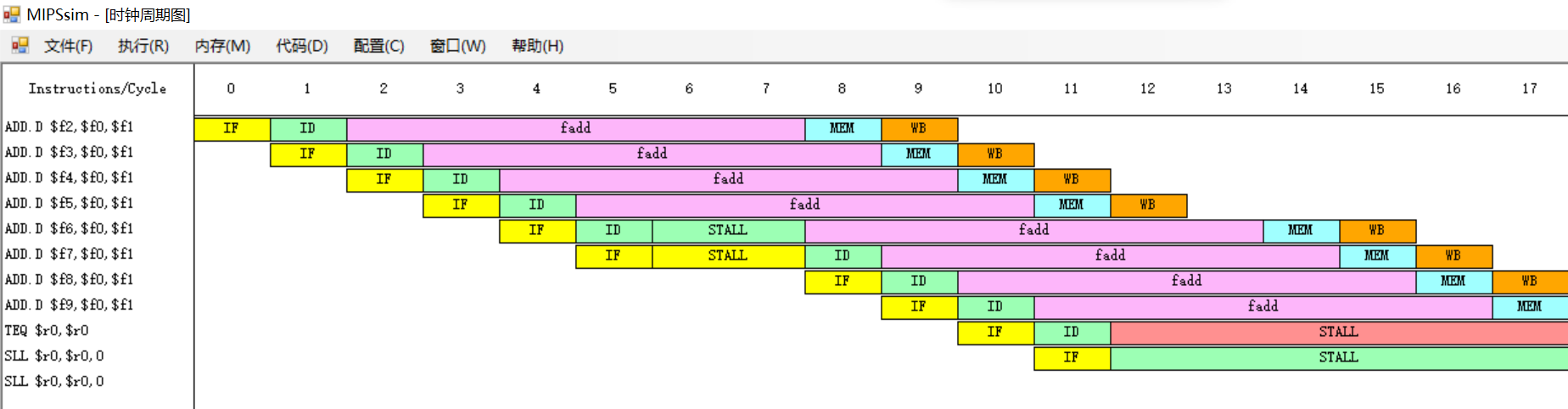
由结构冲突引起的结构停顿周期有35个，占周期总数的67.30769%。



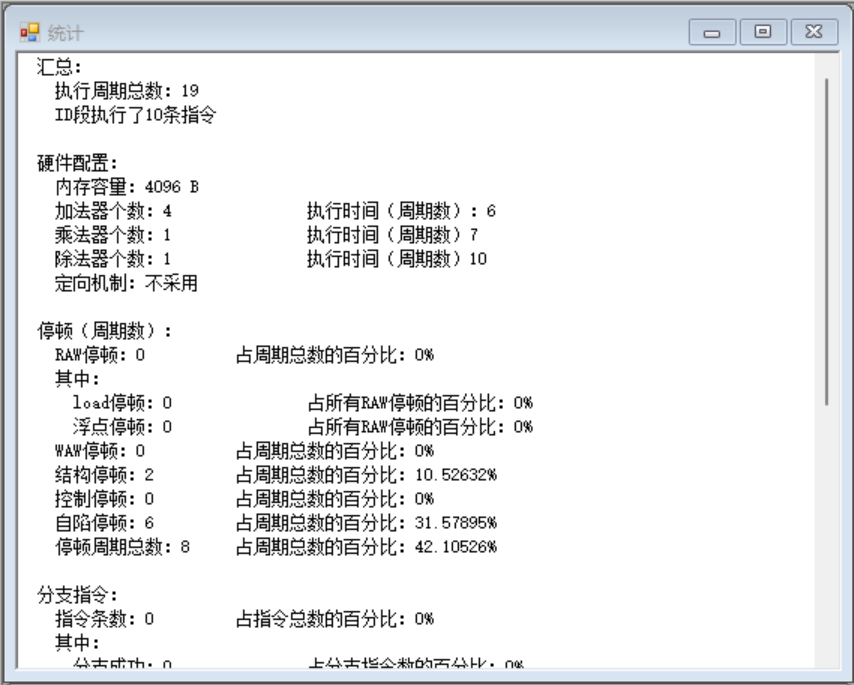
把浮点加法器数量改为4个。



执行程序，发现较少的结构冲突停顿周期。

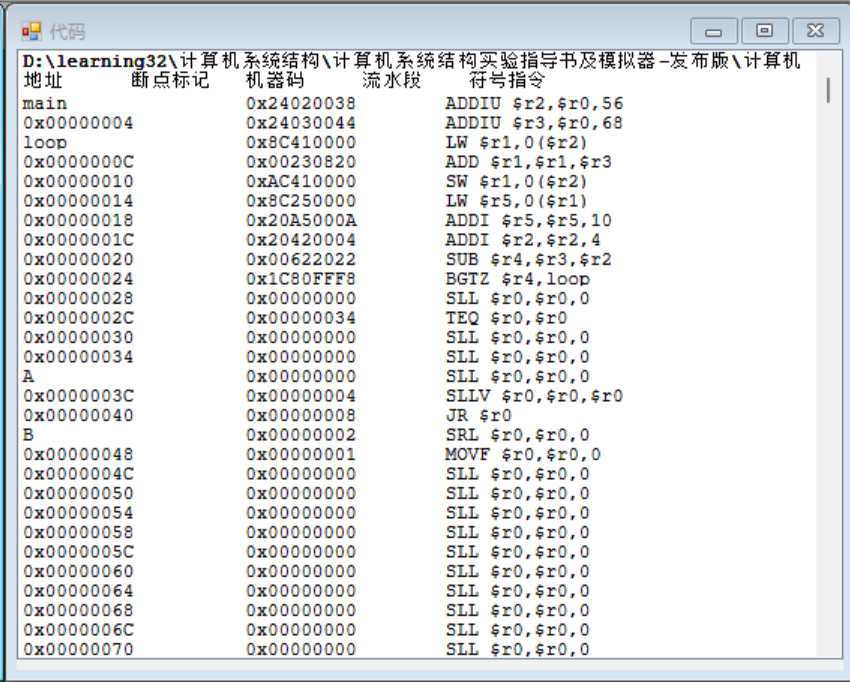


由结构冲突引起的停顿周期数有2个，占周期总数的10.52632%。

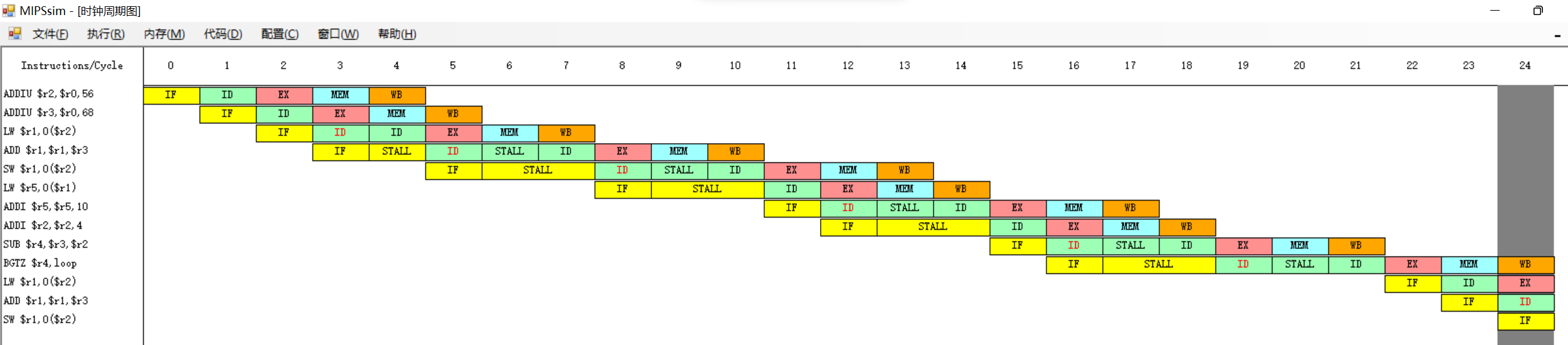


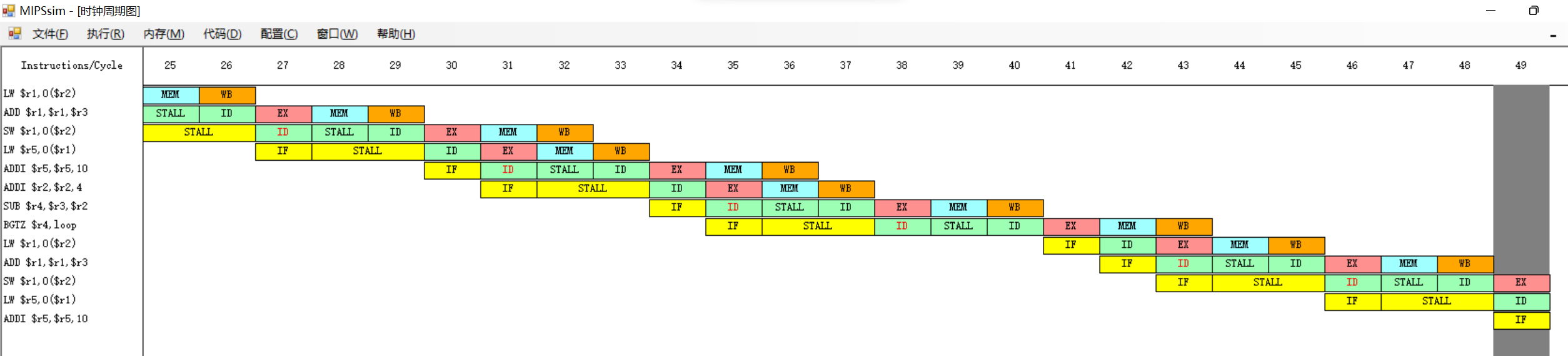
结构冲突会使流水线出现停顿，增加执行周期总数，从而降低流水线的效率和实际的加速比甚至导致错误的执行结果，降低CPU性能。结构冲突由硬件资源满足不了指令重叠执行的要求而导致，因此本次实验中通过增加运算器的数量解决结构冲突。除此之外，还可以通过在取下一条指令前插入暂停周期，或者通过指令和数据的分离存取处理解决。

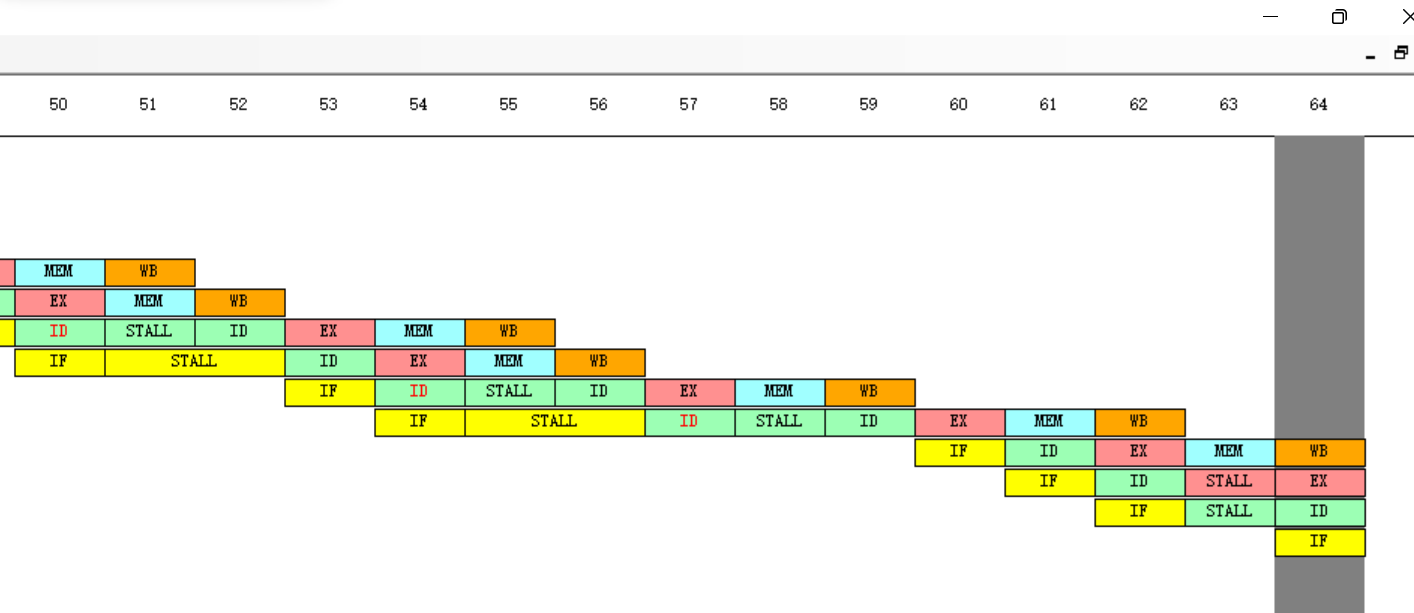
全部复位后，加载样例程序data\_hz.s，关闭定向功能。



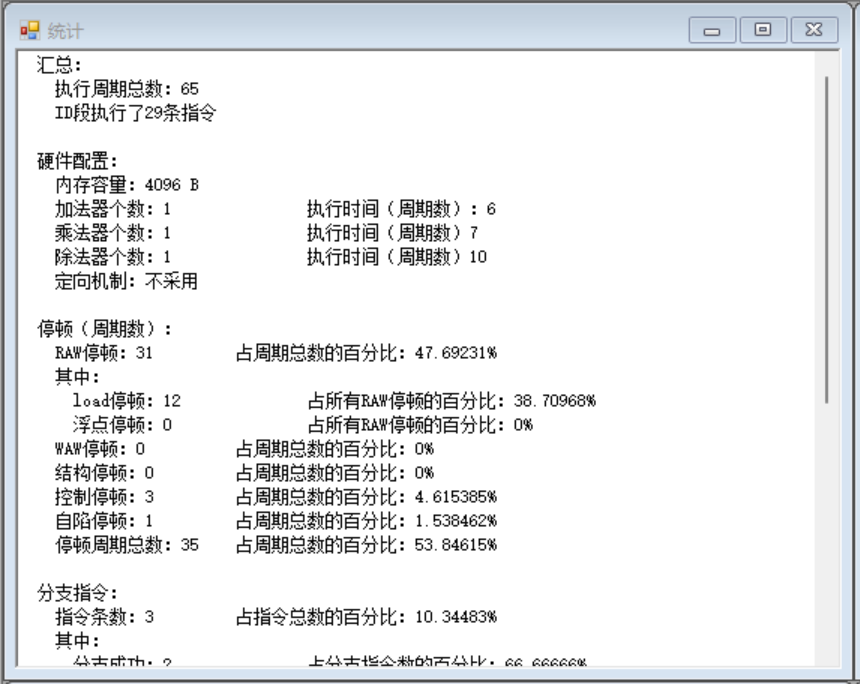
执行程序，发现在时钟周期3、5、8、12、16、19、24、27、31、35、38、43、46、50、54、57，发生了RAW冲突。



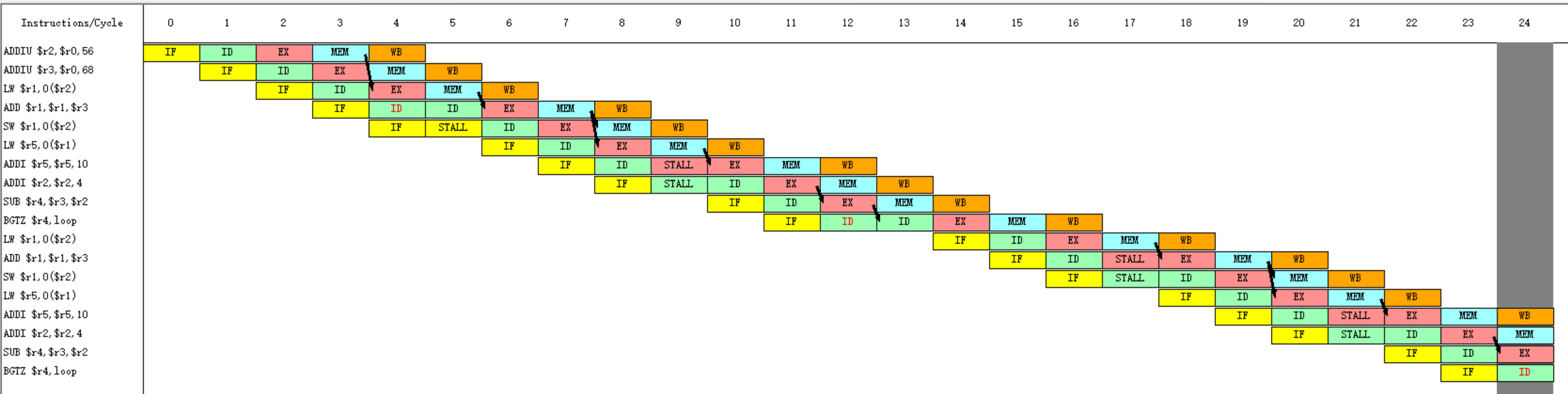


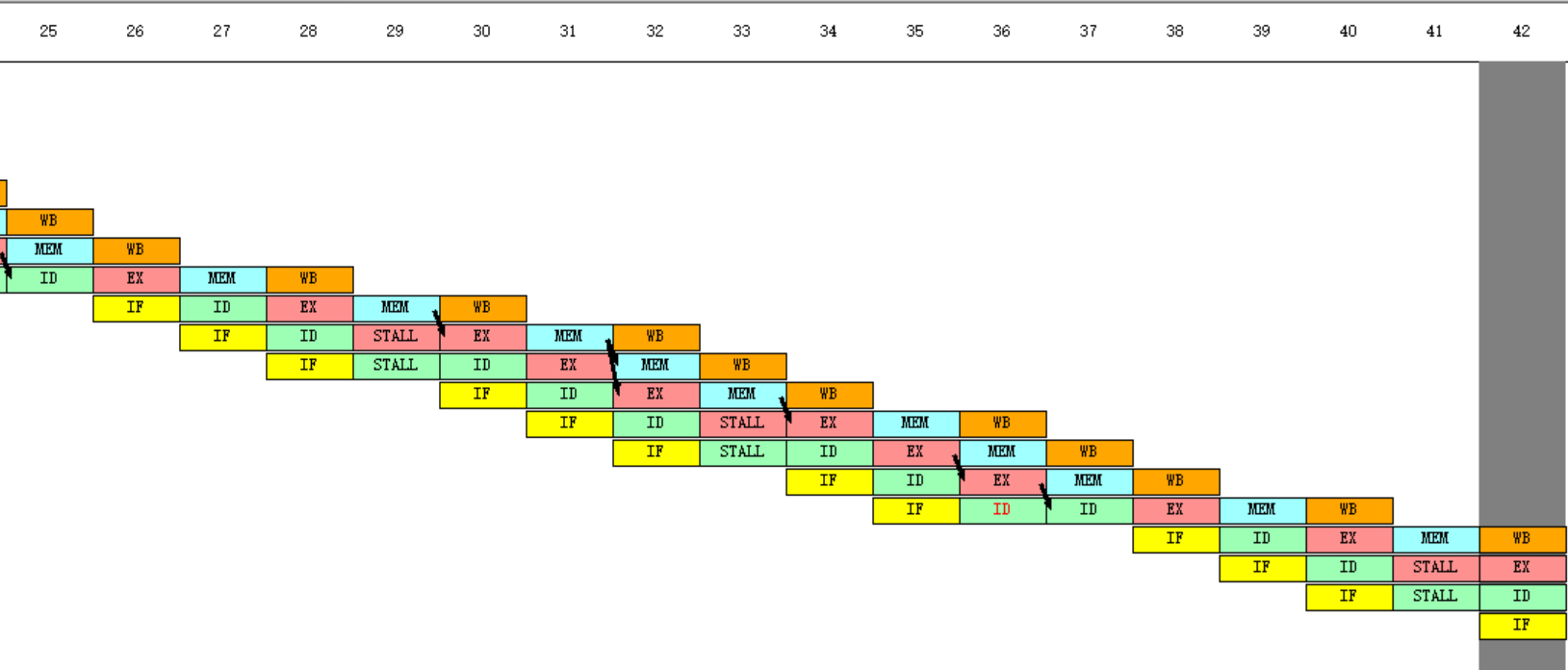


执行周期总数有65个，由数据冲突引起的RAW停顿有31个，占周期总数的47.69231%。

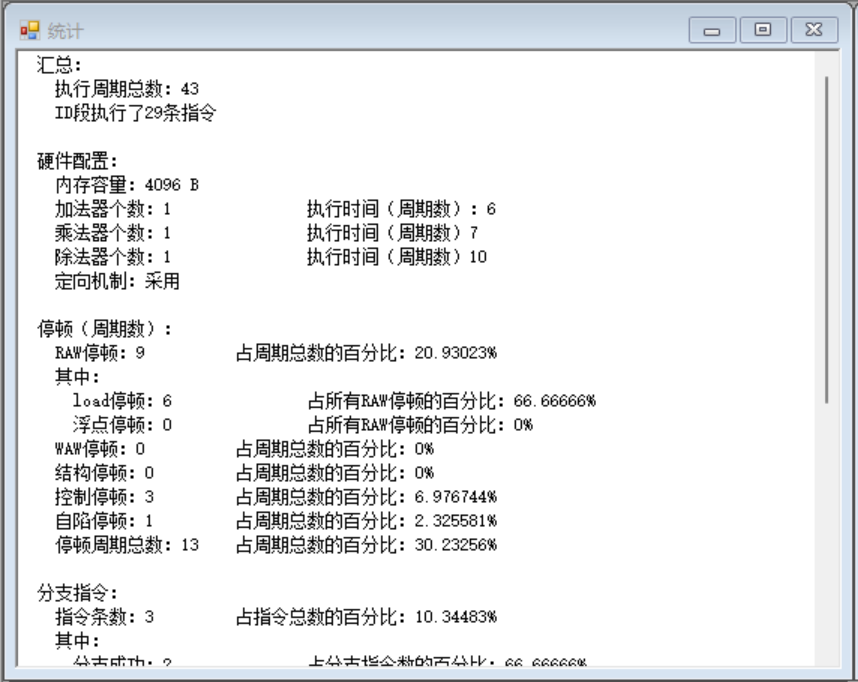


复位CPU，打开定向功能。执行程序，在周期4、12、24和36发送RAW冲突。有了定向功能后，发送RAW冲突的周期少了很多。





执行周期总数有43个，由数据冲突引起的RAW冲突有9个，占周期总数的20.93023%。性能提升了65/43=1.51倍。



**实验总结**

通过使用指令级和流水线操作级模拟器MIPSsim执行示例程序，让我更加理解流水线的基本概念和各段的功能和基本操作。同时，通过测试和分析结构冲突和数据冲突，了解到两种冲突对CPU性能的影响。通过分析它们的出现原因，可以找到解决方法，例如增加硬件资源减少结构冲突，以及使用定向技术减少数据冲突引起的停顿。