实验2 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。

2. 实验平台

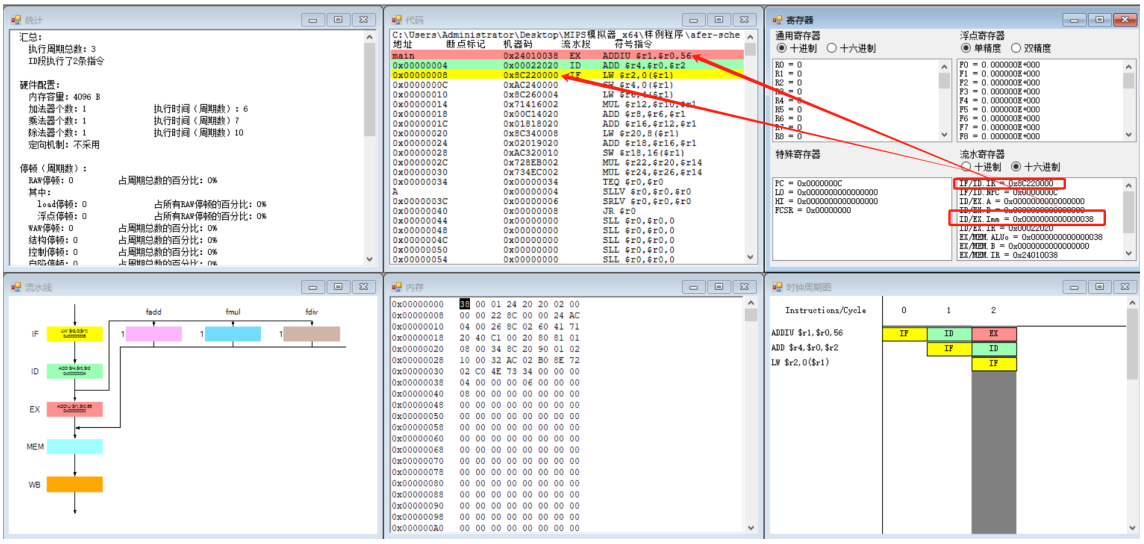
指令级和流水线操作级模拟器 MIPSsim。

3. 实验内容和步骤

（1）启动 MIPSsim。
（2）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。（鼠标双击各段，即可看到各流水寄存器的内容）
（3）载入一个样例程序（在本模拟器所在文件夹下的“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 中寄存器和存储器内容的变化，特别是流水寄存器内容的变化。

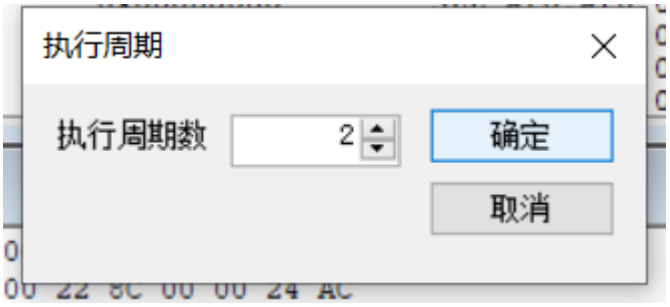
本次载入的是after-schedule.s

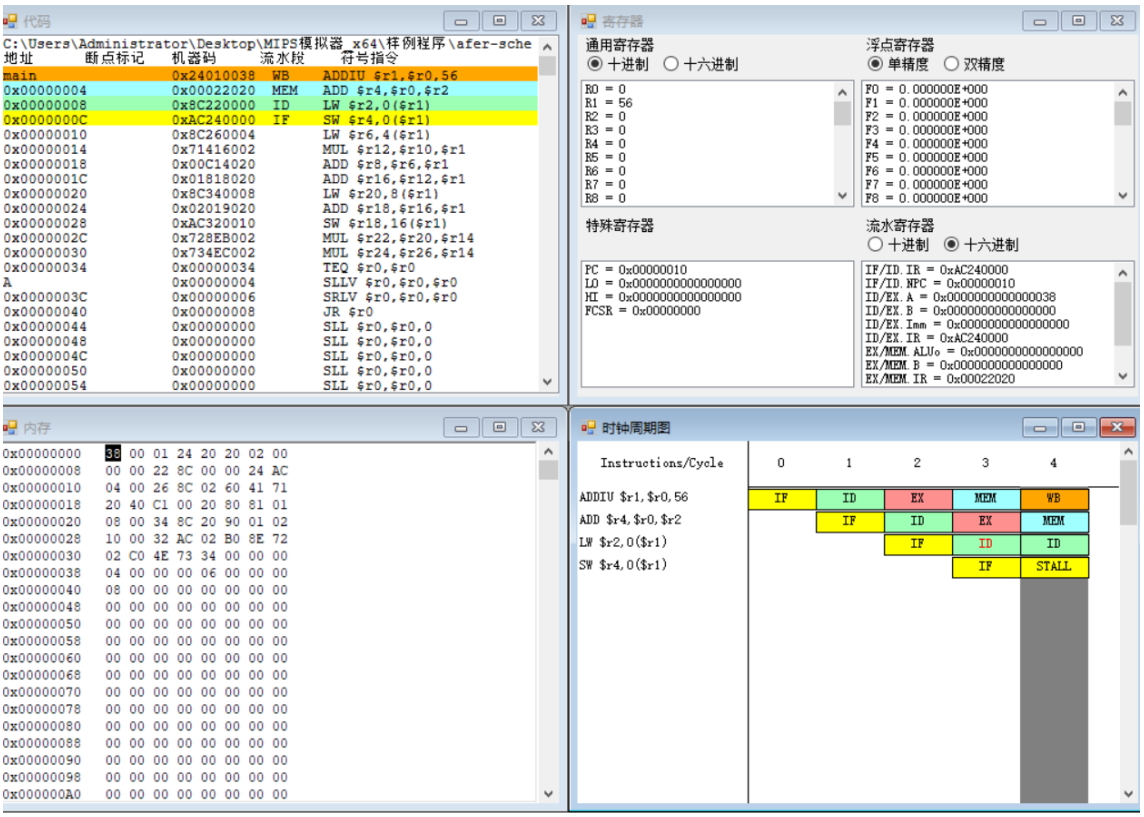
1. 单步执行一个周期



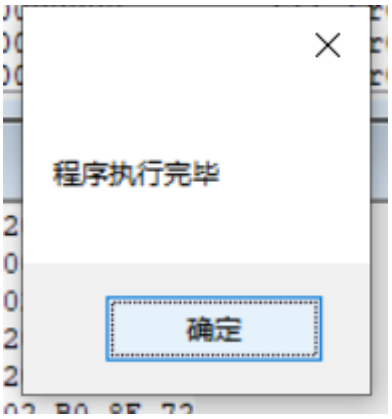
可以看到，当前处于第三个节拍，此时，存入第三条指令，IR中存放第三条指令的机器码；此时第一条指令是RX阶段，立即数是58，故Imm中存入的是0x38

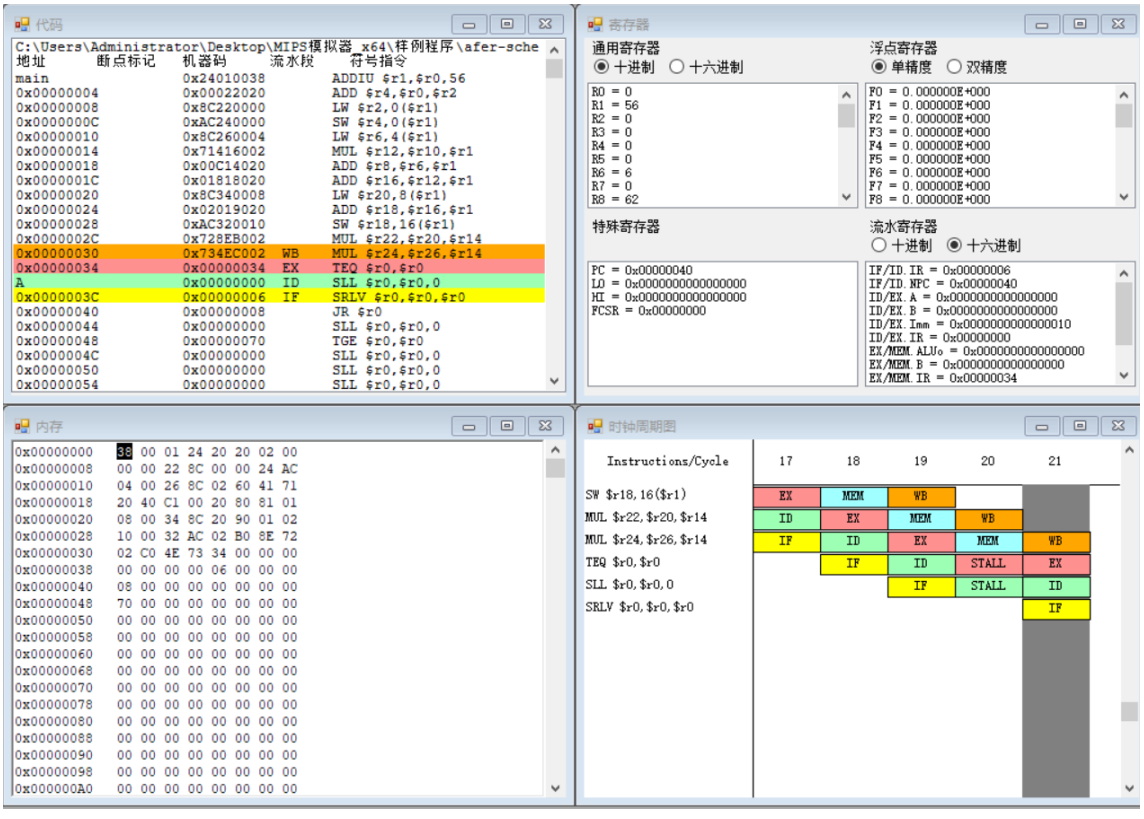
2. 执行2个周期



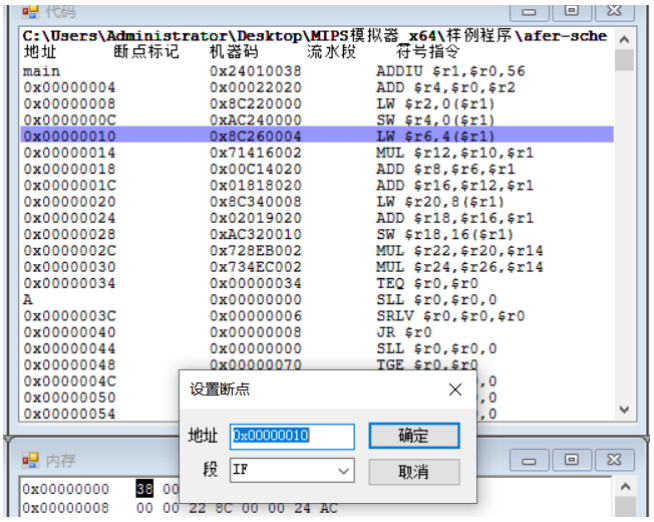

代码
C:\Users\Administrator\Desktop\MIPS模拟器_x64\样例程序\afer-sche
地址 断点标记 机器码 流水段 符号指令
main 0x24010038 WB ADDIU $r1,$r0,56
0x00000004 0x00022020 MEM ADD $r4,$r0,$r2
0x00000008 0x8C220000 ID LW $r2,0($r1)
0x0000000C 0xAC240000 IF SW $r4,0($r1)
0x00000010 0x8C260004 LW $r6,4($r1)
0x00000014 0x71416002 MUL $r12,$r10,$r1
0x00000018 0x00C14020 ADD $r8,$r6,$r1
0x0000001C 0x01818020 ADD $r16,$r12,$r1
0x00000020 0x8C340008 LW $r20,8($r1)
0x00000024 0x02019020 ADD $r18,$r16,$r1
0x00000028 0xAC320010 SW $r18,16($r1)
0x0000002C 0x728EB002 MUL $r22,$r20,$r14
0x00000030 0x734EC002 MUL $r24,$r26,$r14
0x00000034 0x00000034 TEQ $r0,$r0
A 0x00000004 SLLV $r0,$r0,$r0
0x0000003C 0x00000006 SRLV $r0,$r0,$r0
0x00000040 0x00000008 JR $r0
0x00000044 0x00000000 SLL $r0,$r0,0
0x00000048 0x00000000 SLL $r0,$r0,0
0x0000004C 0x00000000 SLL $r0,$r0,0
0x00000050 0x00000000 SLL $r0,$r0,0
0x00000054 0x00000000 SLL $r0,$r0,0
寄存器
通用寄存器
十进制 十六进制
R0 = 0
R1 = 56
R2 = 0
R3 = 0
R4 = 0
R5 = 0
R6 = 0
R7 = 0
R8 = 0
浮点寄存器
单精度 双精度
F0 = 0.000000E+000
F1 = 0.000000E+000
F2 = 0.000000E+000
F3 = 0.000000E+000
F4 = 0.000000E+000
F5 = 0.000000E+000
F6 = 0.000000E+000
F7 = 0.000000E+000
F8 = 0.000000E+000
特殊寄存器
PC = 0x00000010
LO = 0x0000000000000000
HI = 0x0000000000000000
FCSR = 0x00000000
流水寄存器
十进制 十六进制
IF/ID.IR = 0xAC240000
IF/ID.NPC = 0x00000010
ID/EX.A = 0x0000000000000038
ID/EX.B = 0x0000000000000000
ID/EX.Imm = 0x0000000000000000
ID/EX.IR = 0xAC240000
EX/MEM.ALUo = 0x0000000000000000
EX/MEM.B = 0x0000000000000000
EX/MEM.IR = 0x00022020
内存
0x00000000 38 00 01 24 20 20 02 00
0x00000008 00 00 22 8C 00 00 24 AC
0x00000010 04 00 26 8C 02 60 41 71
0x00000018 20 40 C1 00 20 80 81 01
0x00000020 08 00 34 8C 20 90 01 02
0x00000028 10 00 32 AC 02 B0 8E 72
0x00000030 02 C0 4E 73 34 00 00 00
0x00000038 04 00 00 00 06 00 00 00
0x00000040 08 00 00 00 00 00 00 00
0x00000048 00 00 00 00 00 00 00 00
0x00000050 00 00 00 00 00 00 00 00
0x00000058 00 00 00 00 00 00 00 00
0x00000060 00 00 00 00 00 00 00 00
0x00000068 00 00 00 00 00 00 00 00
0x00000070 00 00 00 00 00 00 00 00
0x00000078 00 00 00 00 00 00 00 00
0x00000080 00 00 00 00 00 00 00 00
0x00000088 00 00 00 00 00 00 00 00
0x00000090 00 00 00 00 00 00 00 00
0x00000098 00 00 00 00 00 00 00 00
0x000000A0 00 00 00 00 00 00 00 00
时钟周期图
Instructions/Cycle 0 1 2 3 4
ADDIU $r1,$r0,56 IF ID EX MEM WB
ADD $r4,$r0,$r2 IF ID EX MEM
LW $r2,0($r1) IF ID ID
SW $r4,0($r1) IF STALL

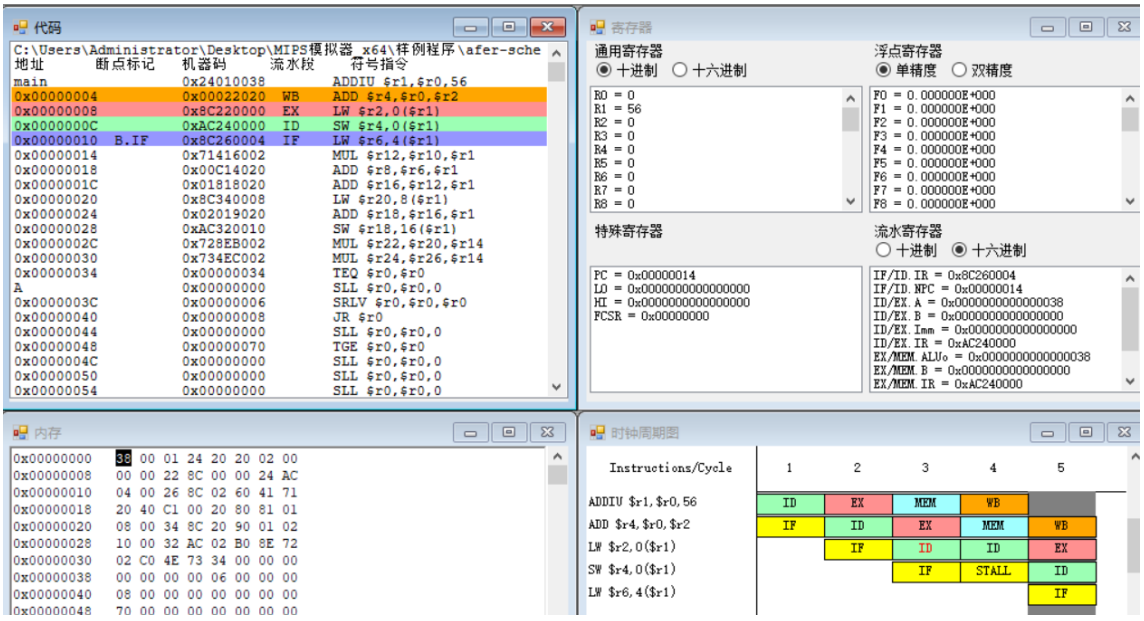
3. 连续执行


程序执行完毕
确定


代码
C:\Users\Administrator\Desktop\MIPS模拟器_x64\样例程序\afer-sche
地址 断点标记 机器码 流水段 符号指令
main 0x24010038 ADDIU $r1,$r0,56
0x00000004 0x00022020 ADD $r4,$r0,$r2
0x00000008 0x8C220000 LW $r2,0($r1)
0x0000000C 0xAC240000 SW $r4,0($r1)
0x00000010 0x8C260004 LW $r6,4($r1)
0x00000014 0x71416002 MUL $r12,$r10,$r1
0x00000018 0x00C14020 ADD $r8,$r6,$r1
0x0000001C 0x01818020 ADD $r16,$r12,$r1
0x00000020 0x8C340008 LW $r20,8($r1)
0x00000024 0x02019020 ADD $r18,$r16,$r1
0x00000028 0xAC320010 SW $r18,16($r1)
0x0000002C 0x728EB002 MUL $r22,$r20,$r14
0x00000030 0x734EC002 WB MUL $r24,$r26,$r14
0x00000034 0x00000034 EX TEQ $r0,$r0
A 0x00000000 ID SLL $r0,$r0,0
0x0000003C 0x00000006 IF SRLV $r0,$r0,$r0
0x00000040 0x00000008 JR $r0
0x00000044 0x00000000 SLL $r0,$r0,0
0x00000048 0x00000070 TGE $r0,$r0
0x0000004C 0x00000000 SLL $r0,$r0,0
0x00000050 0x00000000 SLL $r0,$r0,0
0x00000054 0x00000000 SLL $r0,$r0,0
寄存器
通用寄存器
十进制 十六进制
R0 = 0
R1 = 56
R2 = 0
R3 = 0
R4 = 0
R5 = 0
R6 = 6
R7 = 0
R8 = 62
浮点寄存器
单精度 双精度
F0 = 0.000000E+000
F1 = 0.000000E+000
F2 = 0.000000E+000
F3 = 0.000000E+000
F4 = 0.000000E+000
F5 = 0.000000E+000
F6 = 0.000000E+000
F7 = 0.000000E+000
F8 = 0.000000E+000
特殊寄存器
PC = 0x00000040
LO = 0x0000000000000000
HI = 0x0000000000000000
FCSR = 0x00000000
流水寄存器
十进制 十六进制
IF/ID.IR = 0x00000006
IF/ID.NPC = 0x00000040
ID/EX.A = 0x0000000000000000
ID/EX.B = 0x0000000000000000
ID/EX.Imm = 0x0000000000000010
ID/EX.IR = 0x00000000
EX/MEM.ALUo = 0x0000000000000000
EX/MEM.B = 0x0000000000000000
EX/MEM.IR = 0x00000034
内存
0x00000000 38 00 01 24 20 20 02 00
0x00000008 00 00 22 8C 00 00 24 AC
0x00000010 04 00 26 8C 02 60 41 71
0x00000018 20 40 C1 00 20 80 81 01
0x00000020 08 00 34 8C 20 90 01 02
0x00000028 10 00 32 AC 02 B0 8E 72
0x00000030 02 C0 4E 73 34 00 00 00
0x00000038 00 00 00 00 06 00 00 00
0x00000040 08 00 00 00 00 00 00 00
0x00000048 70 00 00 00 00 00 00 00
0x00000050 00 00 00 00 00 00 00 00
0x00000058 00 00 00 00 00 00 00 00
0x00000060 00 00 00 00 00 00 00 00
0x00000068 00 00 00 00 00 00 00 00
0x00000070 00 00 00 00 00 00 00 00
0x00000078 00 00 00 00 00 00 00 00
0x00000080 00 00 00 00 00 00 00 00
0x00000088 00 00 00 00 00 00 00 00
0x00000090 00 00 00 00 00 00 00 00
0x00000098 00 00 00 00 00 00 00 00
0x000000A0 00 00 00 00 00 00 00 00
时钟周期图
Instructions/Cycle 17 18 19 20 21
SW $r18,16($r1) EX MEM WB
MUL $r22,$r20,$r14 ID EX MEM WB
MUL $r24,$r26,$r14 IF ID EX MEM WB
TEQ $r0,$r0 IF ID STALL EX
SLL $r0,$r0,0 IF STALL ID
SRLV $r0,$r0,$r0 IF

4. 设置断点



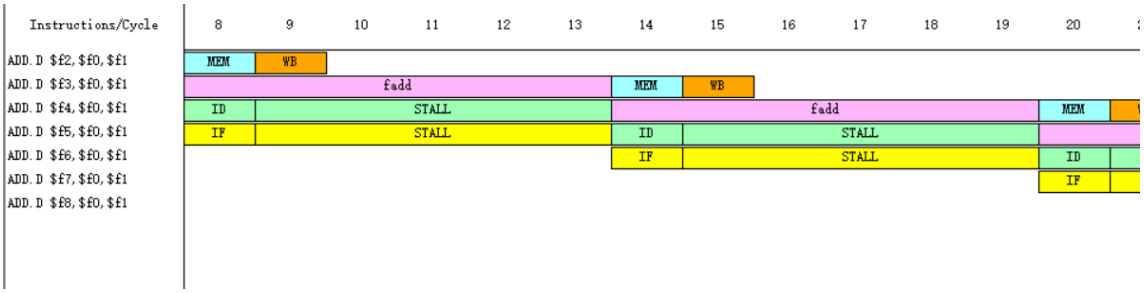


(4) 选择配置菜单中的"流水方式"选项，使模拟器工作于流水方式下。

(5) 观察程序在流水方式下的执行情况。

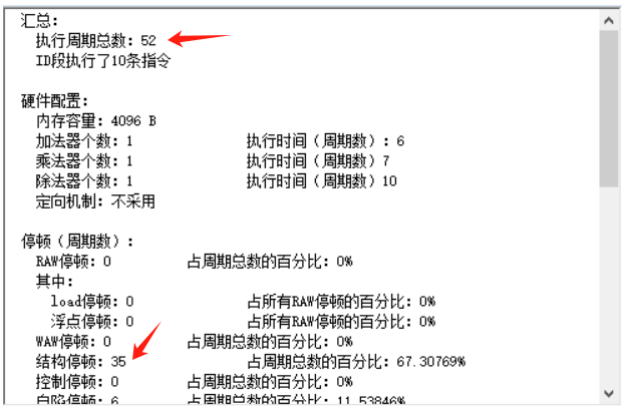
(6) 观察和分析结构冲突对 CPU 性能的影响，步骤如下：

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。



从图中可以看出，每个相邻的`ADD.D`指令都存在冲突，其中，冲突是由浮点加法器引起的

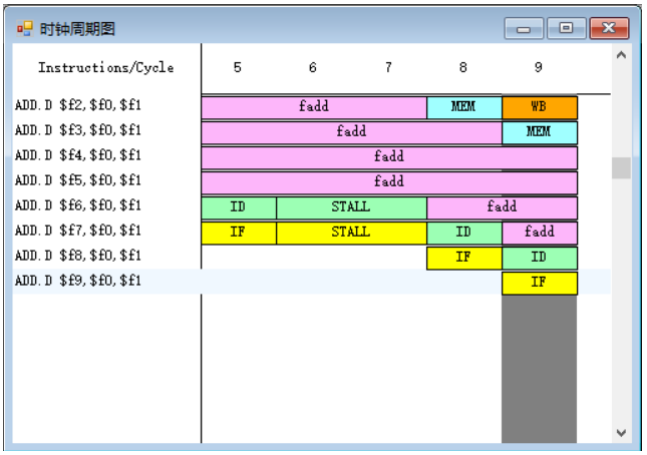
3. 记录由结构冲突引起的停顿周期数，计算停顿周期数占总执行周期数的百分比。



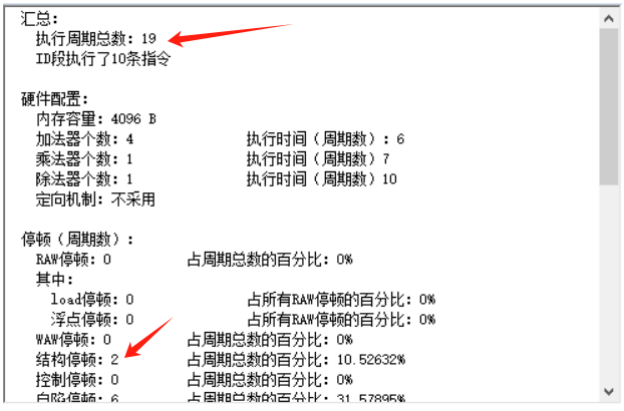
0-51，共计52个周期。其中，由结构冲突引起的停顿周期有35个，占周期总数的67.03769%

4. 把浮点加法器的个数改为 4 个。

5. 再重复 1-3 的步骤。



`ADD.D $f5,$f0,$f1` 和 `ADD.D $f6,$f0,$f1` 之间存在冲突，导致冲突的部件是浮点加法器



总周期数19个，由结构冲突引起的停顿周期数2个，占比 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。

结构冲突是计算机体系结构中的一种冲突类型，指的是处理器在执行指令序列时由于指令之间的数据或资源依赖关系而导致的冲突。这种冲突会对CPU性能产生负面影响，因为它会导致指令的执行发生延迟或停顿，从而降低了指令级并行性和整体的处理器吞吐量。

结构冲突主要涉及以下几个方面:

1. 数据访问冲突：当多条指令需要访问相同的数据存储器或寄存器时，由于数据访问端口的限制，可能会导致冲突。例如，在同时执行一条加载指令和一条存储指令时，如果它们需要访问同一个存储器位置，就会发生数据访问冲突。
2. 功能单元冲突：当多条指令需要使用相同的功能单元（如乘法器或加法器）时，由于功能单元的数量有限，可能会导致冲突。例如，如果同时执行两条乘法指令，但处理器只有一个乘法器，就会发生功能单元冲突。

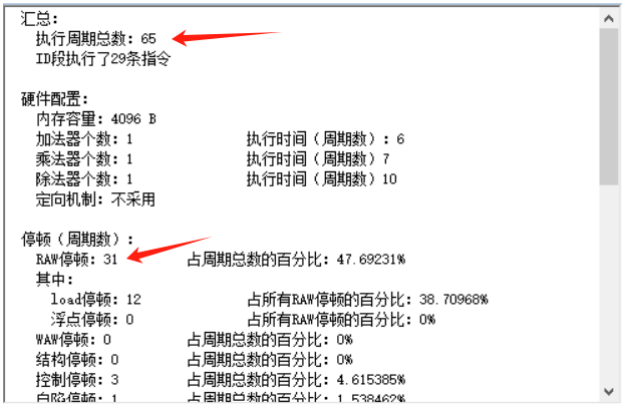
解决结构冲突的方法包括以下几种:

1. 硬件重组：通过增加硬件资源来解决结构冲突。例如，增加数据存储器的端口数、增加功能单元的数量或增加寄存器文件的读写端口数。这种方法可以提高处理器的吞吐量，但会增加硬件成本和功耗。
2. 数据和指令重排：通过对指令序列进行重排序，将具有依赖关系的指令分开或交错执行，以减少结构冲突的发生。例如，使用编译器优化技术，如乱序执行（out-of-order execution）或指令调度（instruction scheduling），来改变指令执行的顺序。这种方法不需要额外的硬件支持，但需要在编译器或处理器的微码中实现相应的逻辑。
3. 预取和缓存技术：通过预先从内存中提取数据、指令或预测访问模式来减少数据访问冲突。预取技术可以在发生结构冲突之前提前将数据加载到高速缓存中，以避免延迟。缓存技术可以利用局部性原理，将经常访问的数据存储在高速缓存中，以减少对主存的访问次数。
4. 超线程和多核处理器：通过使用超线程或多核处理器来增加处理器的并行度，以减少结构冲突的影响。超线程技术允许在同一个处理器核心上同时执行多个线程，共享处理器资源。多核处理器则将多个处理器核心集成在同一芯片上，每个核心可以独立执行指令，从而提高处理器的整体性能。

这些方法可以单独或组合使用，具体的选择取决于处理器架构、应用程序特性和性能要求。通过优化结构冲突的处理，可以提高CPU的性能和效率。

7. 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（在“配置”菜单下选择取消“定向”）。
    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。

        发生RAW冲突的周期（按照时钟周期图，从周期0开始计）：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
    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，计算停顿时钟周期数占总执行周期数的百分比。



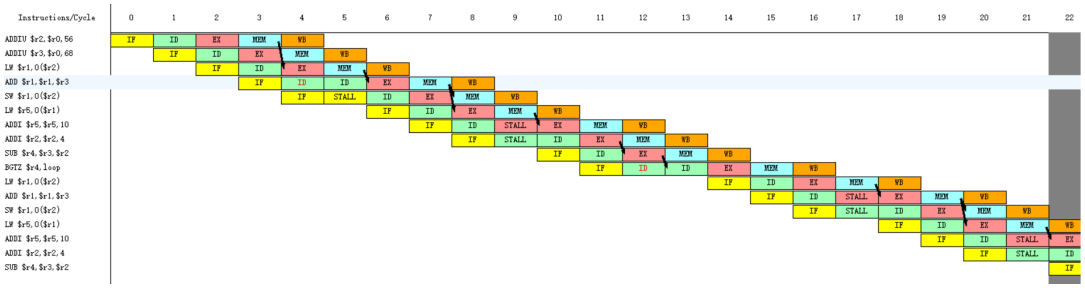
总周期数65个，由于数据冲突引起的停顿周期数31个，占比47.69231%

6. 复位 CPU。
7. 打开定向功能。
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。

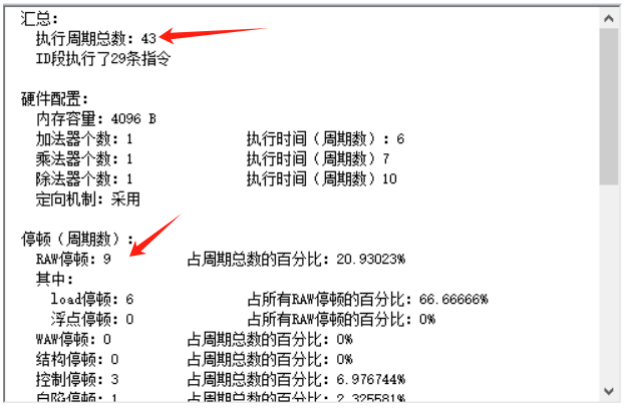
   发生RAW冲突的周期（按照时钟周期图，从周期0开始计）：5，10，13，18，22，25，30，34，37

   可见发生 RAW 冲突次数显著减少，每次冲突的停顿周期数也显著减少

   时钟周期图为：



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。



   总周期数 43 个，由数据冲突引起的停顿周期数9个，占比20.93023%。

   采用定向技术后流水机的性能是原来的$\frac{65}{43} \times 100\% = 1.51$倍