**深 圳 大 学 实 验 报 告**

**课 程 名 称： 计算机系统(3)**

**实验项目名称： 处理器结构实验一**

**学 院： 计算机与软件学院**

**专 业： 计算机科学与技术/软件工程**

**指 导 教 师： 王 毅**

**报告人： 郑雨婷 学号： 2021150122 班级： 高性能**

**实 验 时 间： 2023年11月2日**

**实验报告提交时间： 2023年11月19日**

**教务处制**

**一、实验目标：**

了解MIPS的五级流水线，和在运行过程中的所产生的各种不同的流水线冒险

通过指令顺序调整，或旁路与预测技术来提高流水线效率

更加了解流水线细节和其指令的改善方法

更加熟悉MIPS指令的使用

**二、实验内容**

观察一段代码并运行，观察其中的流水线冒险，并记录统计统计信息。

对所给的代码进行指令序列的调整，以期避免数据相关，并记录统计信息。

启动forward功能，以获得性能提升，并且记录统计信息。

（选做：用perf记录x86中的数据相关于指令序列调整后的时间统计、

调整指令，以避免连续乘法间的阻塞。）

**三、实验环境**

硬件：桌面PC

软件：Windows，WinMIPS64仿真器

**四、****实验步骤及说明**

首先，我们给出一段C代码，该段代码实现的是两个矩阵相加。

设有4\*4矩阵A和4\*4矩阵B相加，得到4\*4矩阵C：

for(inti = 0; i< 4; i++)

For(int j = 0; j < 4; j++)

C[i][j] = A[i][j] + B[i][j];

根据上述的C代码，我们将其转换成MIPS语言，然后运行，并进行分析。

MIPS代码如下：

.data

a: .word 1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4

b: .word 4,4,4,4,3,3,3,3,2,2,2,2,1,1,1,1

c: .word 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

len: .word 4

control: .word32 0x10000

data: .word32 0x10008

.text

start:daddi r17,r0,0

daddi r21,r0,a

daddi r22,r0,b

daddi r23,r0,c

ld r16,len(r0)

loop1: slt r8,r17,r16

beq r8,r0,exit1

daddi r19,r0,0

loop2: slt r8,r19,r16

beq r8,r0,exit2

dsll r8,r17,2

dadd r8,r8,r19

dsll r8,r8,3

dadd r9,r8,r21

dadd r10,r8,r22

dadd r11,r8,r23

ld r9,0**(r9**)

ld r10,0(r10)

dadd r12,r9,r10

sd r12,0(r11)

daddi r19,r19,1

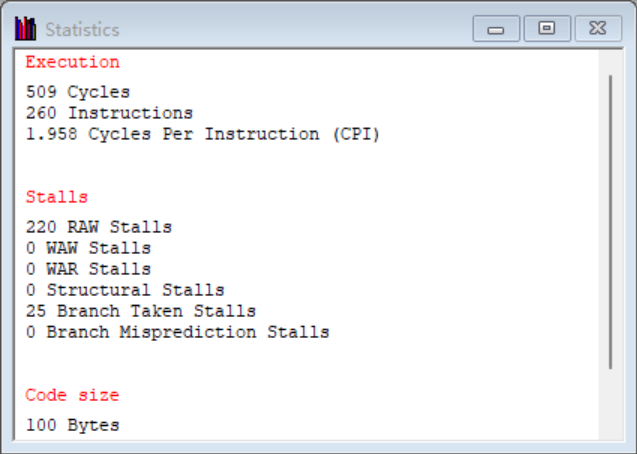
j loop2

exit2:daddi r17,r17,1

j loop1

exit1: halt

实验前请保证winMIPS64配置中“Enable Forwarding”没有选中。将这段代码加载到WinMIPS64中，运行后观察结果（提供Statistic窗口截图）。从Statistic窗口记录：本程序运行过程中总共产生了多少次RAW的数据相关。接下来，我们对产生数据相关的代码逐个分析，请列出产生数据相关的代码，并在下一步中进行分析和优化。



运行以上代码后得到以上截图，可以看到，运行过程中总共产生了220次RAW的数据相关。产生数据相关的代码：

1. ld r16,len(r0)

loop1: slt r8,r17,r16

2. slt r8,r17,r16

beq r8,r0,exit1

3. daddi r19,r0,0

loop2: slt r8,r19,r16

4. slt r8,r19,r16

beq r8,r0,exit2

5. dsll r8,r17,2

dadd r8,r8,r19

6. dadd r8,r8,r19

dsll r8,r8,3

7. dsll r8,r8,3

dadd r9,r8,r21

8. ld r10,0(r10)

dadd r12,r9,r10

9. dadd r12,r9,r10

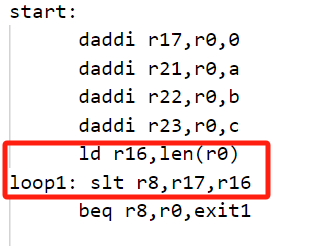
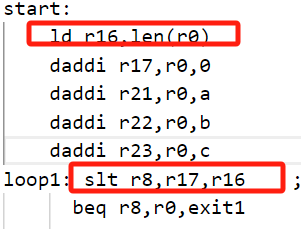
sd r12,0(r11)

1. 调整指令序列

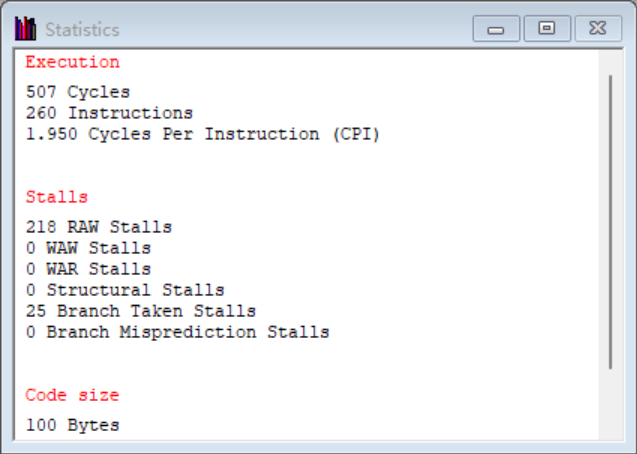
在这一部分，我们利用指令调整的方法对数据相关代码进行优化，规避数据相关。

通过调整序列来规避这个数据相关，在statics窗口中记录其效果。将此结果与初始的结果进行对比，报告RAW相关的次数减少的数量。

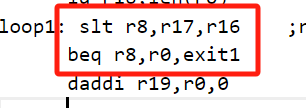
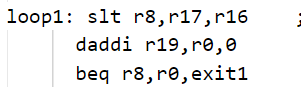
我们知道，数据相关是指一条指令必须等待另一条指令的完成而造成流水线暂停的情况。例如，红框圈出的两行代码中，ld r16,len(r0)在还未将数据写入r16时，下一条指令进入到EX阶段，这时就需要等待，产生了RAW相关。

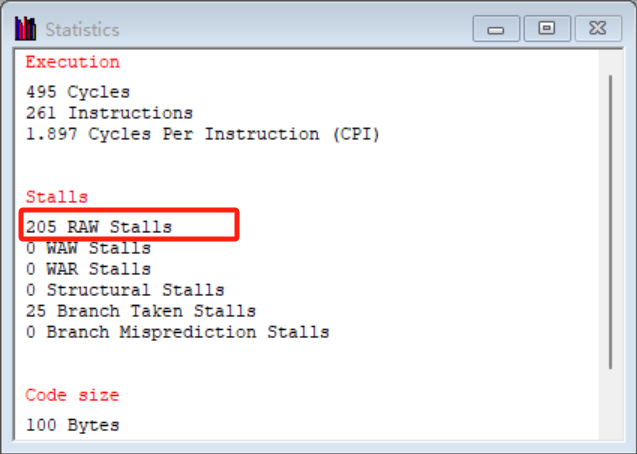
将这两行代码隔开之后，再次运行，可以发现RAW相关减少至218.



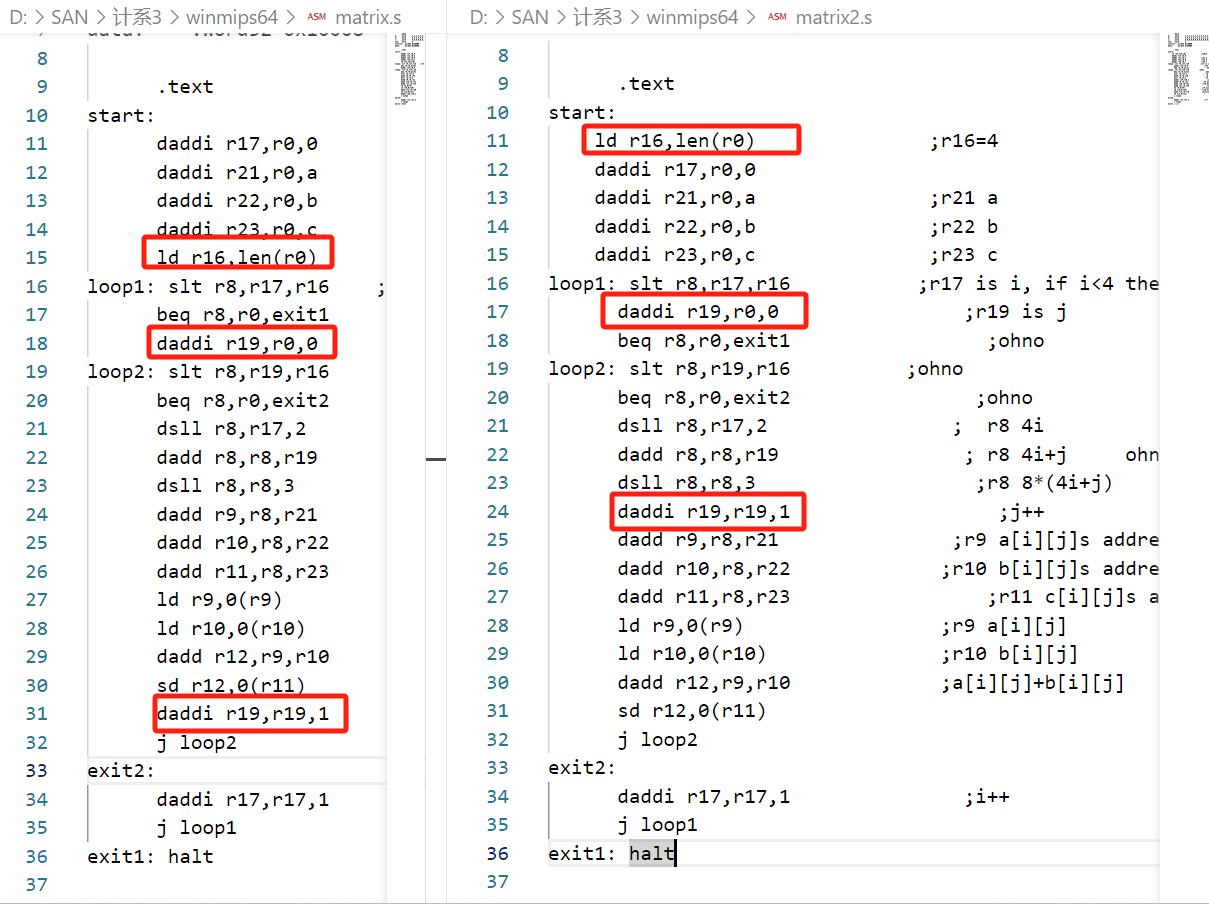
同理，对下图左侧的代码顺序进行调整，变为右侧，将数据相关的两行代码隔开。

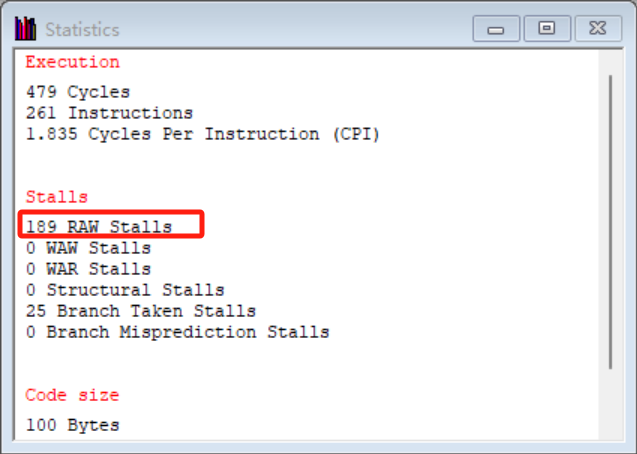
再次运行，可以发现RAW相关减少至205.



包括每次内层循环对j进行加一操作后，又立马进行判断是否小于4，这也会产生RAW。所以进行如下调整：



再次运行，可以发现RAW相关减少至189.

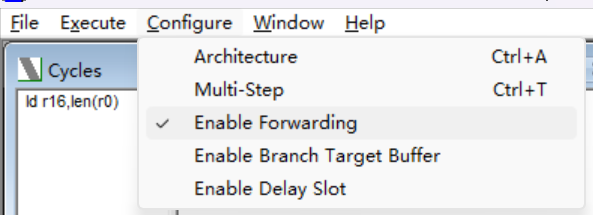


因此，我们可以总结出，想要消除数据相关RAW，最简单最直接的方法就是在不改变程序的原意的前提下，将产生数据相关的两行代码尽量隔开。减少必须等待上一条指令完成的情况。

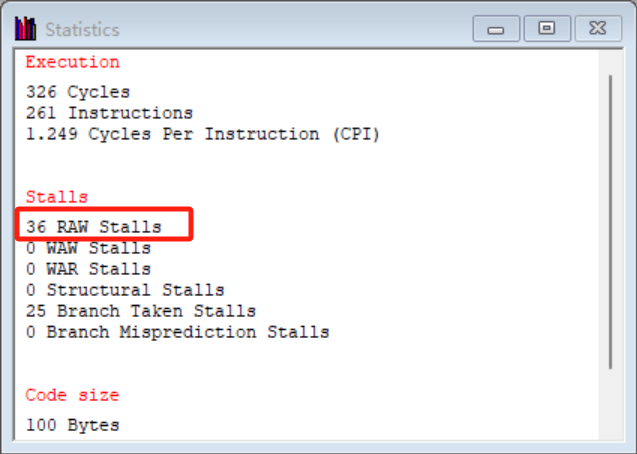
1. Forwarding功能开启

接下来，我们要展示Forwarding功能的优化效果。

首先，我们要知道如何开启Forwarding功能。法如下：点开***configure***下拉窗口，给***Enable Forwarding***选项左侧点上勾。



开启了Forwarding功能之后，我们再运行，查看结果，发现仅有36个RAW数据相关。解释哪些数据相关的问题得到解决，并以截图说明问题解决前后的差异所在。



对比开启Forwarding功能前后的Cycles窗口，查看减少的RAW。

1. ld r10,0(r10)

dadd r12,r9,r10

若没有开启Forwading功能，add指令需要等待ld指令将数据写回r10之后，才可以进行加法运算EX阶段，所以需要两次RAW。



在开启Forwarding功能之后，在ld指令EX之后，建立旁路，将r10的数据发送给add指令，这样就可以减少一次RAW。



2. slt r8,r19,r16

beq r8,r0,exit2

若没有开启Forwading功能，beq指令需要等待slt指令将数据写回r8之后，才可以进行运算EX阶段，所以需要两次流水线阻塞。



在开启Forwarding功能之后，在slt指令执行EX之后，建立旁路，将r8的数据发送给beq指令，此时beq立马可以进行EX，这样就可以减少一次RAW。



3. dsll r8,r17,2

dadd r8,r8,r19

若没有开启Forwading功能，add指令需要等待sll指令将数据写回r8之后，才可以进行运算EX阶段，所以需要两次流水线阻塞。



在开启Forwarding功能之后，在sll指令执行EX之后，建立旁路，将r8的数据发送给add指令，此时add立马可以进行EX，这样就可以解决RAW。



4. dadd r8,r8,r19

dsll r8,r8,3

若没有开启Forwading功能，sll指令需要等待add指令将数据写回r8之后，才可以进行运算EX阶段，所以需要两次流水线阻塞。



在开启Forwarding功能之后，在add指令执行EX之后，建立旁路，将r8的数据发送给sll，此时sll立马可以进行EX，这样就可以解决RAW。



5. dsll r8,r8,3

dadd r9,r8,r21

与问题2相同。若没有开启Forwading功能，add指令需要等待sll指令将数据写回r8之后，才可以进行运算EX阶段，所以需要两次流水线阻塞。



在开启Forwarding功能之后，在sll指令执行EX之后，建立旁路，将r8的数据发送给add指令，此时add立马可以进行EX，这样就可以解决RAW。



6. ld r9,0(r9)

ld r10,0(r10)

dadd r12,r9,r10

若没有开启Forwading功能，dadd 指令的 rs 和 rt 分别是两个 ld 指令的 rt，因此需要两个ld 指令都执行完 WB 之后，dadd 指令才能执行EX阶段。因此有 2 个 RAW。



开启 forwarding 后：两个 ld 指令在执行完 EX 之后，就将还没存进 r9 和 r10 的结果传给 dadd 指令的 ID/EX 部分，因此减少了 2 个 RAW。但由于要等 ld 指令完成 WB 之后，dadd指令才能执行 Mem 部分，因此还有 1 个 RAW。



7. dadd r12,r9,r10

sd r12,0(r11)

若没有开启Forwading功能，dadd 指令的 rd 寄存器是 sd 指令的 rt 寄存器，因此在 dadd 指令执行完 WB之后，sd 指令才能执行 EX阶段，因此有 2 个 RAW。



开启 forwarding 后：dadd 指令执行完 EX 之后就可以把数据直接传给 sd 的 ID/EX 部分，因此减少了 2 个 RAW。



1. 结构相关优化

流水线中的结构相关，指的是流水线中多条指令在同一时钟周期内争用同一功能部件现象。即因硬件资源满足不了指令重叠执行的要求而发生的冲突。

在WinMIPS64中，我们可以在除法中观察到这种现象。要消除这种结构相关，我们可以采取调整指令位置的方法进行优化。在这个部分，我们首先给出几条C代码，然后将该代码翻译成MIPS代码（为了观察的方便，我们这里MIPS代码并不是逐一翻译，而是调整代码，使得其他部分数据相关已经优化，而两条除法指令连续出现），运行并查看结果。接着，调整代码序列，重新运行。观察优化效果。

下面是给出的C代码：

a = a / b

c = c / d

e = e + 1

f = f + 1

g = g + 1

h = h + 1

i = i + 1

j = j + 1

根据上述的C代码，我们给出数据相关优化的指令如下：

.data

a: .word 12

b: .word 3

c: .word 15

d: .word 5

e: .word 1

f: .word 2

g: .word 3

h: .word 4

i: .word 5

.text

start:

ld r16,a(r0)

ld r17,b(r0)

ld r18,c(r0)

ld r19,d(r0)

ld r20,e(r0)

ld r21,f(r0)

ld r22,g(r0)

ld r23,h(r0)

ld r24,i(r0)

ddiv r16,r16,**r17**

ddiv r18,r18,r19

daddi r20,r20,1

daddi r21,r21,1

daddi r22,r22,1

daddi r23,r23,1

daddi r24,r24,1

st r16,a(r0)

st r17,b(r0)

st r18,c(r0)

st r19,d(r0)

st r20,e(r0)

st r21,f(r0)

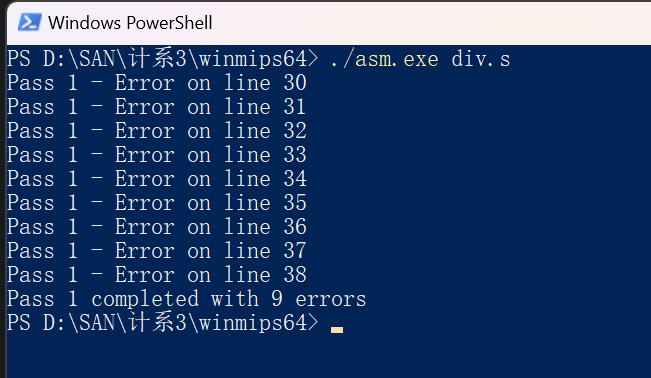
st r22,g(r0)

st r23,h(r0)

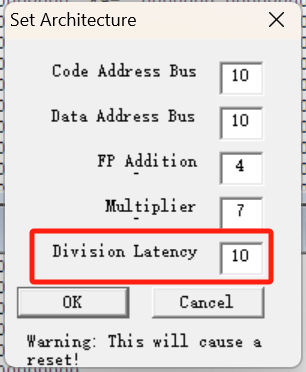
st r24,i(r0)

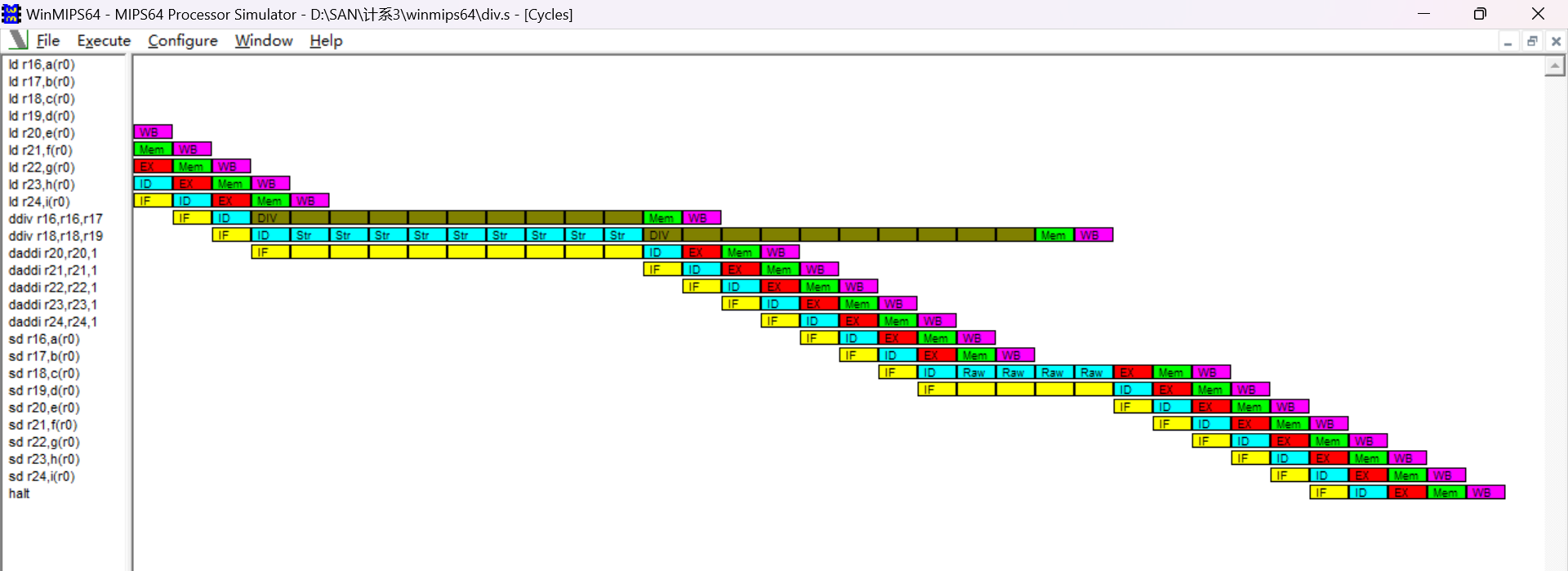
halt

上面的指令编译，发现错误，将st指令改为sd指令后编译成功。

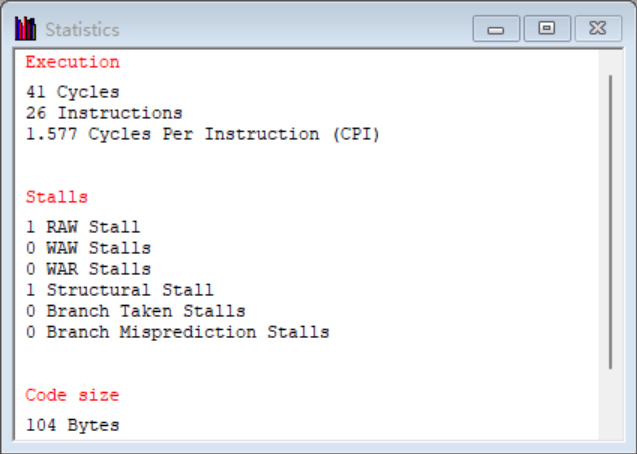


运行程序，在***Cycle***窗口结果如下（程序运行前请将configure->architecture->division latency改为10）：





在***Statistics***窗口的结果如下：



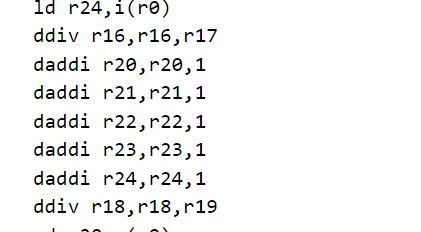
通过观察，我们可以发现，两个连续的除法产生了明显的结构相关，第二个除法为了等待上一个除法指令在执行阶段所占用的资源，阻塞了9个周期。

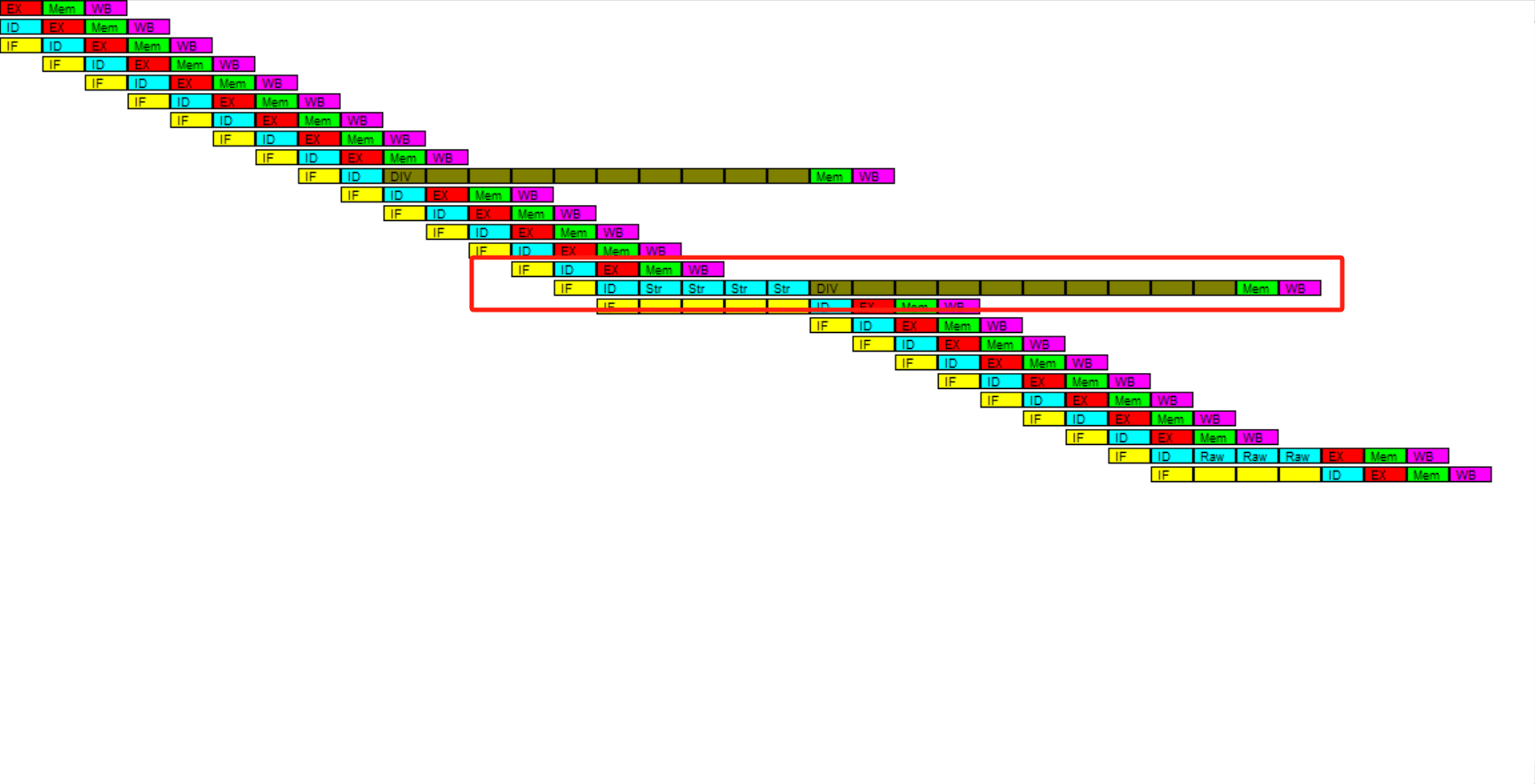
显然，这样的连续的除法所导致的结构相关极大的降低了流水线效率，为了消除结构相关，我们需要做的是调整指令序列，将其他无关的指令塞入两条连续的除法指令中。

给出指令序列的调整方案并给出流水线工作状态的截图，做出解释。

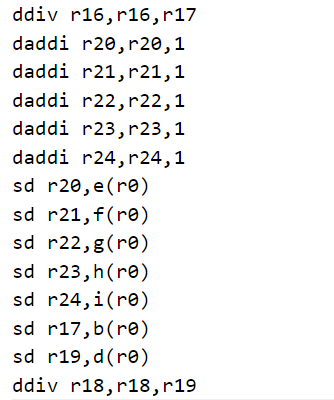
因为第二个除法为了等待上一个除法指令在执行阶段所占用的资源，阻塞了 9 个周期，要调整指令序列以减少结构相关，可以在两个连续的除法指令之间插入其他指令，以充分利用流水线中的各个阶段。

好在这个程序的加位运算结果与除法运算互不相干，只需要可以将加法运算的5条指令插入两个除法运算之间，但是因为阻塞了九个周期，只插入了五条指令，应该还未消除结构相关。





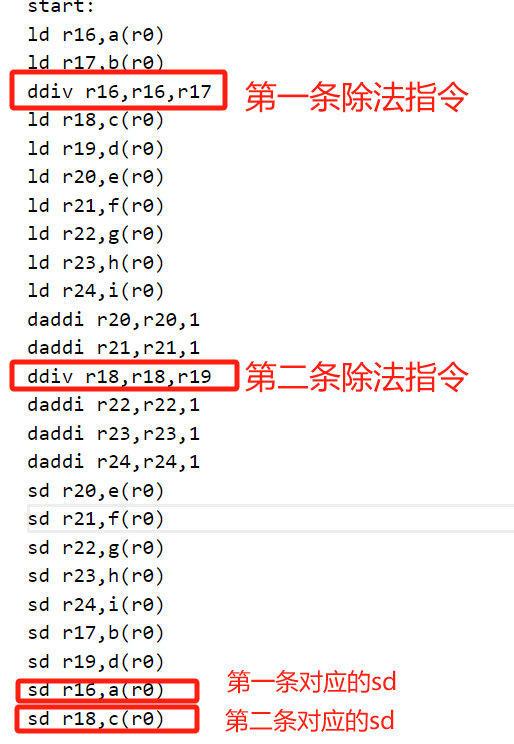
如上图，果然还是阻塞了四个周期，并未消除结构相关。还需要四条指令，可以将储存加法结果的sd指令放到两个除法之间，得到如下代码：



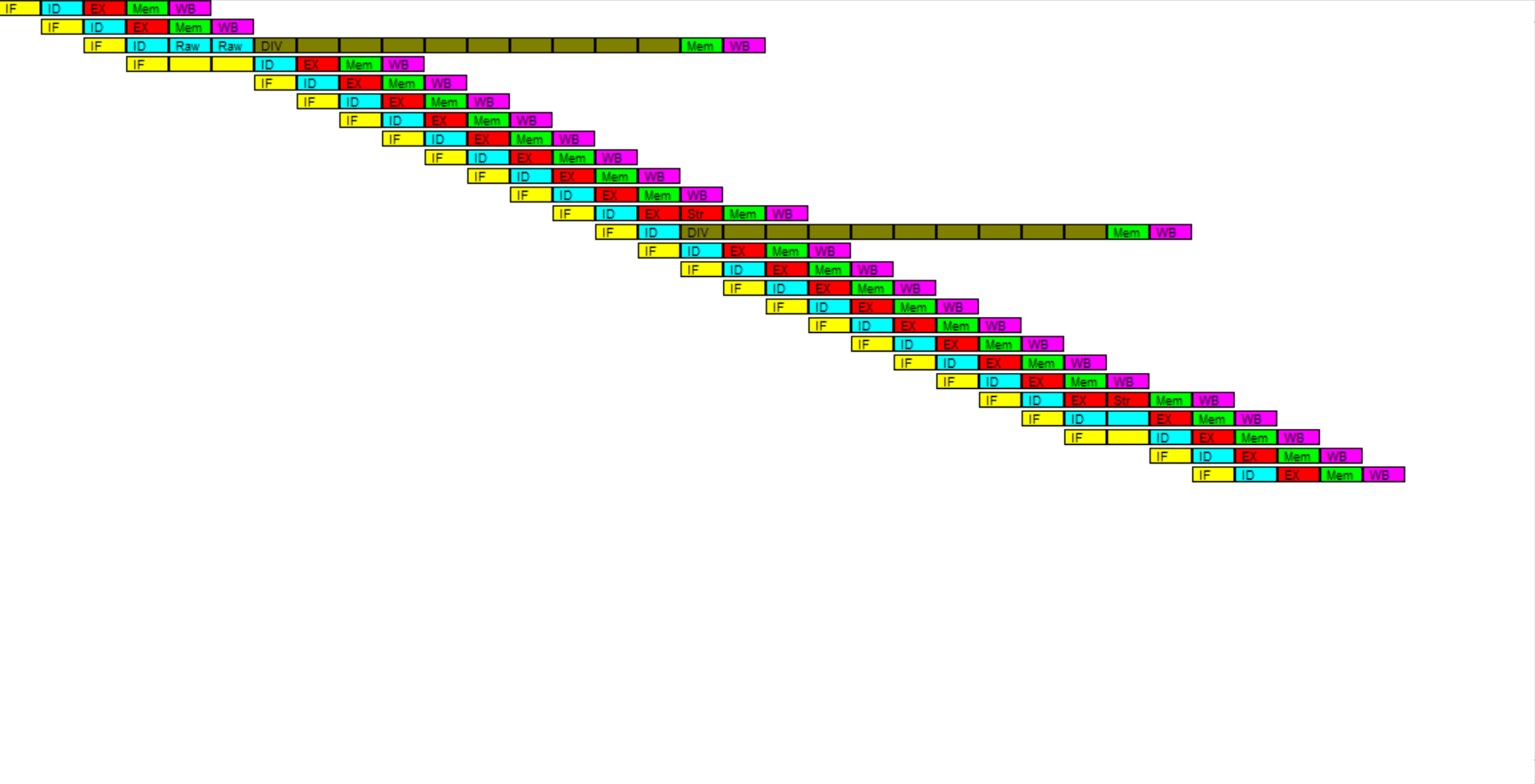
此时再次运行，查看cycles窗口发现，第二条除法指令没有被阻塞，成功消除了这个结构相关。



需要注意的是，这样做导致在存储除法运算结果的sd指令中出现了RAW数据相关，这是因为除法指令的执行需要十个时钟周期。在store数据至内存地址中时，除法指令还没有执行完毕，没有执行WB阶段。要一直阻塞至除法指令WB执行完后，sd指令才能进入EX阶段。因此，第一条除法指令和第二条除法指令、除法指令和对应的sd指令之间，都需要间隔九条指令。



此时的流水线工作状态的截图如下，可以看到，即消除了结构相关又消除了数据相关。



1. 提交报告

记录实验过程，保存实验截图，给出分析结果，形成实验报告。初始代码准备（10分），后面每个优化方法各30分。

**五、实验结果**

在第一部分中，了解了什么是RAW数据相关。通过指令顺序的调整，将RAW从220次减少到189次。在第二部分中，打开forwarding功能，对程序进行了优化。通过对比开启Forwarding前后的流水线工作截图，找出优化的代码。分析了forwarding的工作机制。在第三部分中，了解了什么是结构相关。通过指令顺序的调整，将结构相关的影响消除，有效地减少了流水线的阻塞。

**五、实验总结与体会**

通过本次实验，我深入了解了MIPS的五级流水线结构以及在运行过程中可能产生的不同种类的流水线冒险。

首先，在实验的第一部分，我学习了什么是RAW数据相关，通过对一段代码的观察和运行，详细记录了其中出现的流水线冒险，特别是RAW数据相关的情况。通过对指令序列的调整，成功将RAW冒险从220次减少到189次，有效提高了流水线的效率。在实验的第二部分，我启动了forwarding功能，通过对程序进行了优化。通过比对开启Forwarding前后的流水线工作截图，找出了优化的代码，并深入分析了forwarding的工作机制。这一步骤使得数据能够更快地传递到需要的阶段，有效减少了流水线冒险，进一步提高了整体性能最后，在实验的第三部分，我了解了什么是结构相关。通过对指令顺序的调整，成功消除了结构相关的影响，减少了流水线的阻塞。这个过程中，我进一步体会到了流水线优化对整体性能的重要性。

总体而言，通过这次实验，我不仅对MIPS五级流水线的工作原理有了更深刻的理解，还学到了如何优化流水线性能的方法。

|  |
| --- |
| **指导教师批阅意见：**  **成绩评定：**  指导教师签字：王毅  2023年11月28日 |
| 备注： |