# 实验二 五级流水线与流水线冒险

## 实验目的

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

## 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

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

## 实验环境

硬件：桌面PC

软件：Windows、WinMIPS64

## 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(int i = 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(r0)

ld r10,0(r10)

dadd r12,r9,r10

sd r12,0(r11)

daddi r19,r19,1

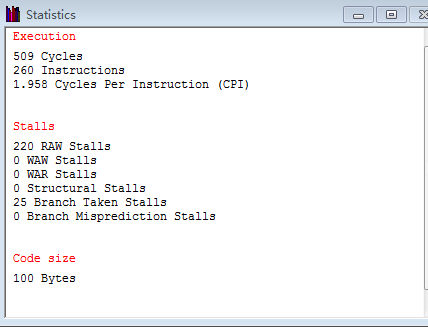
j loop2

exit2:daddi r17,r17,1

j loop1

exit1: halt

将这段代码加载到WinMIPS64中，运行结果如下（Statistic窗口结果）：



从Statistic窗口可以看出，本程序运行过程中总共产生了220处RAW的数据相关。接下来，我们一一列举出来这些数据相关产生的代码，并进行分析和优化。

1. 调整指令序列

在这一部分，我们进行优化的方式为调整指令序列:

**第一个调整的部分：**



上诉的两条语句产生了两个RAW的阻塞，此处，我们可以通过将“ld r16，len（r0）”这条语句换到上面，即start：处，如下：

start:ld r16,len(r0)

daddi r17,r0,0

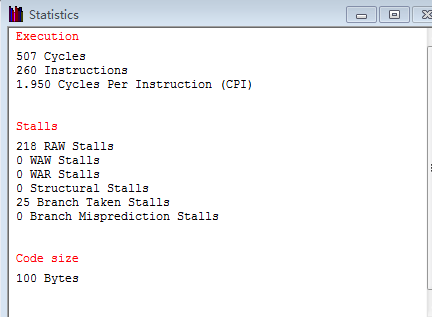
daddi r21,r0,a

daddi r22,r0,b

daddi r23,r0,c

loop1: slt r8,r17,r16

通过调整序列来规避这个数据相关，效果如下：



将此图与上面的结果图进行对比，我们可以发现，RAW阻塞的次数减少了两次。

**第二个调整的部分：**



上述的语句产生了四个RAW数据相关，此处，我们也进行序列的调整，调整如下：

ld r9,0(r0)

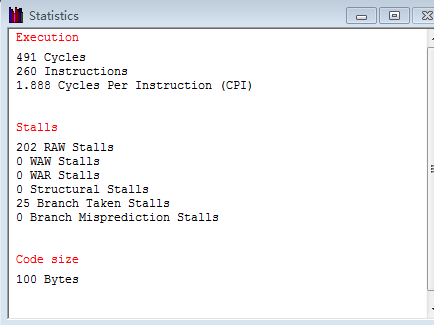
ld r10,0(r10)

daddi r19,r19,1

dadd r12,r9,r10

sd r12,0(r11)

通过将“daddi r19，r19，1”这条语句上移，我们可以减少一次RAW的数据相关，运行后结果如下：



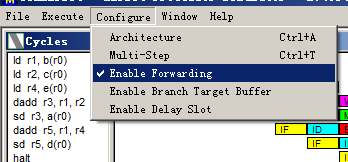
将结果图与之前的结果对照，很明显发现，RAW阻塞的次数减少了16次。效果还是比较明显。

鉴于此程序相对简单，可以通过调整指令序列来优化流水线相关的地方较少，但是上面的一些调整还是能明显的看出这种优化的有效性。

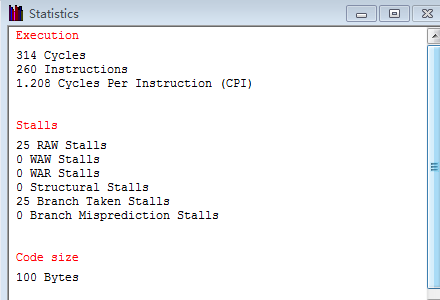
1. Forwarding功能开启

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，结果如下：



查看结果，可以看到RAW阻塞从202次变成了25次，效果十分显著。

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,r16

ddiv r18,r18,r19

daddi r20,r20,1

daddi r21,r21,1

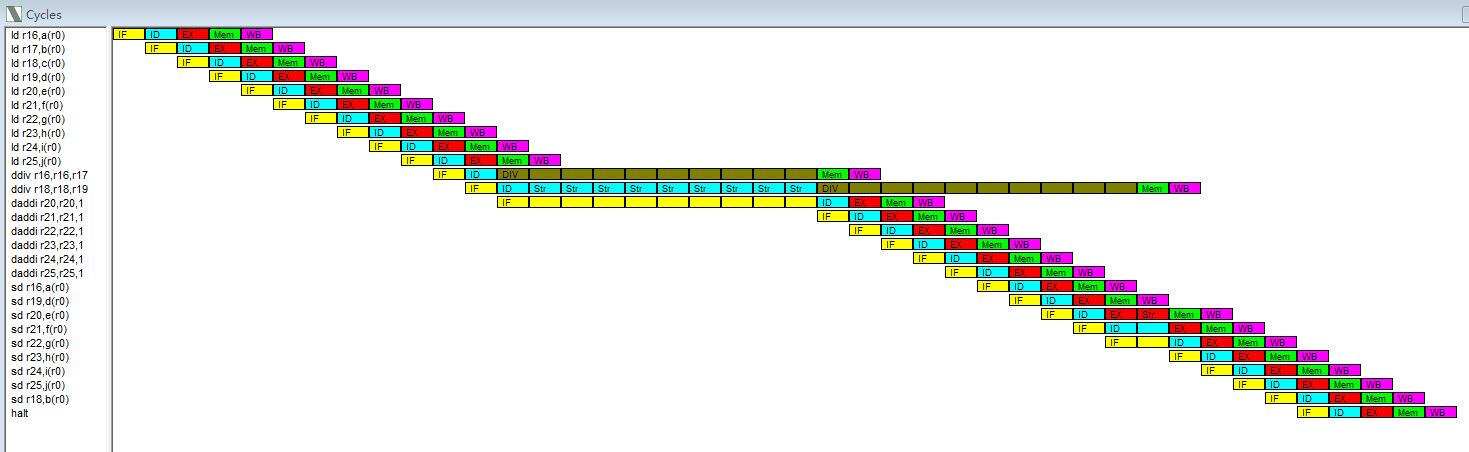
daddi r22,r22,1

daddi r23,r23,1

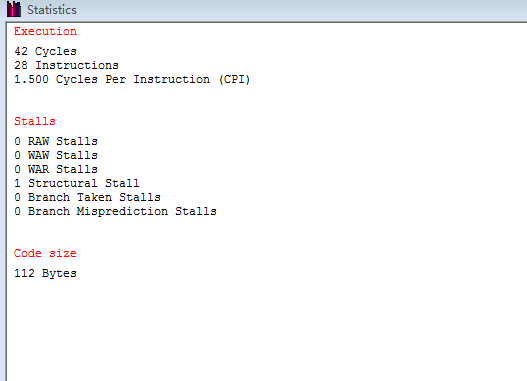
daddi r24,r24,1

halt

上面的指令运行，在***Cycle***窗口结果如下：



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。

代码调整如下：

.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

j: .word 6

.text

start:

ld r16,a(r0)

ld r17,b(r0)

ld r18,c(r0)

ld r19,d(r0)

ddiv r16,r16,r17

ld r20,e(r0)

ld r21,f(r0)

ld r22,g(r0)

ld r23,h(r0)

ld r24,i(r0)

ld r25,j(r0)

daddi r20,r20,1

daddi r21,r21,1

daddi r22,r22,1

ddiv r18,r18,r19

daddi r23,r23,1

daddi r24,r24,1

daddi r25,r25,1

sd r16,a(r0)

sd r19,d(r0)

sd r20,e(r0)

sd r21,f(r0)

sd r22,g(r0)

sd r23,h(r0)

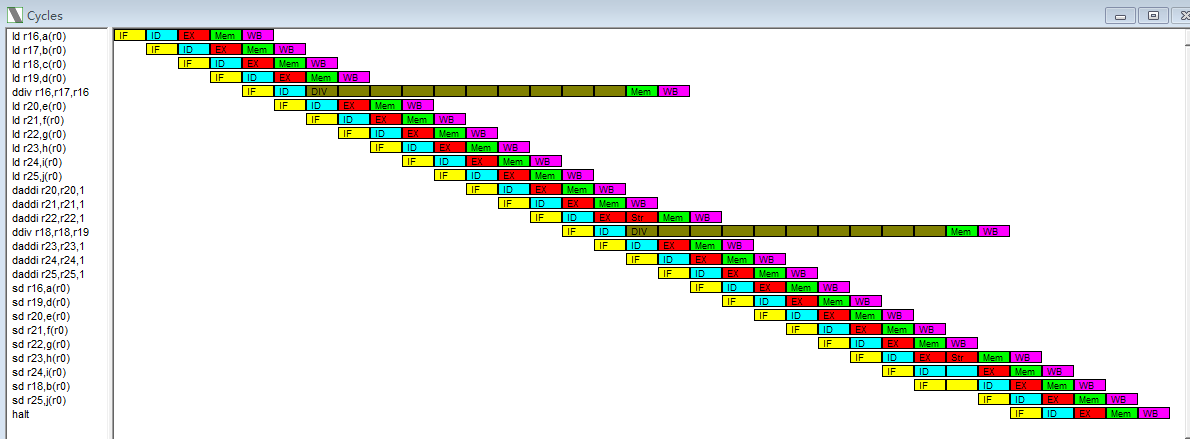
sd r24,i(r0)

sd r25,j(r0)

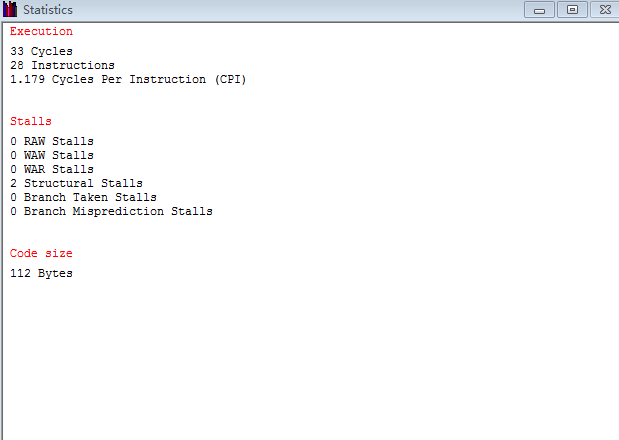
sd r18,b(r0)

halt

运行调整后的代码，在***Cycle***窗口结果如下：



在***Statistics***窗口的结果如下：



通过对比，我们可以发现，优化后的代码在运行时减少了9个周期，优化效果显著。

1. Perf观察结构优化效果
2. 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。