**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**

**指 导 教 师： 刘刚**

**报告人： 吴嘉楷 学号： 2022150168 班级： 国际班**

**实 验 时 间： 2024年11月14日**

**实验报告提交时间： 2024年11月16日**

**教务处制**

**一、实验目标：**

1. 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
2. 通过指令顺序调整，或旁路与预测技术来提高流水线效率
3. 更加了解流水线细节和其指令的改善方法
4. 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。

1. 调整指令序列

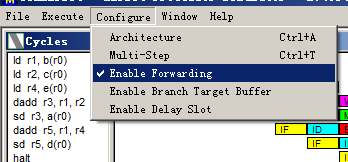
在这一部分，我们利用指令调整的方法对数据相关代码进行优化，规避数据相关。

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。

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，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。

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

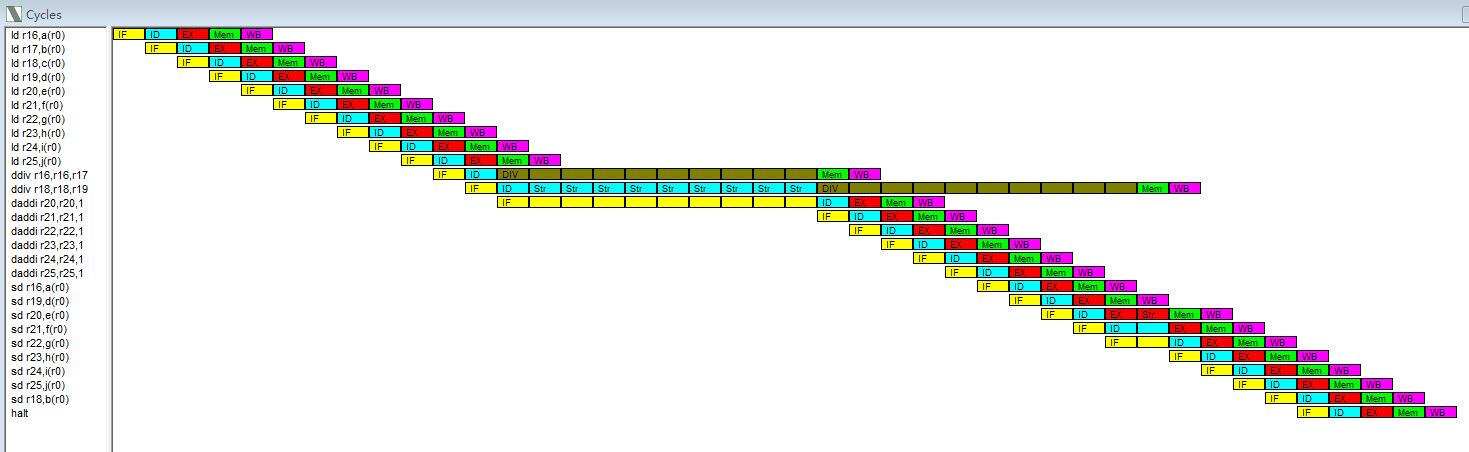
daddi r22,r22,1

daddi r23,r23,1

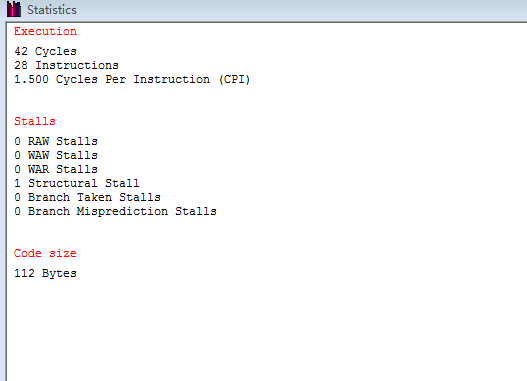
daddi r24,r24,1

halt

上面的指令运行，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。

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**

1. **代码测试与初步分析**
2. 创建文件matrix\_add.s，部分代码如图1所示，并将其load到WINMIPS64中。

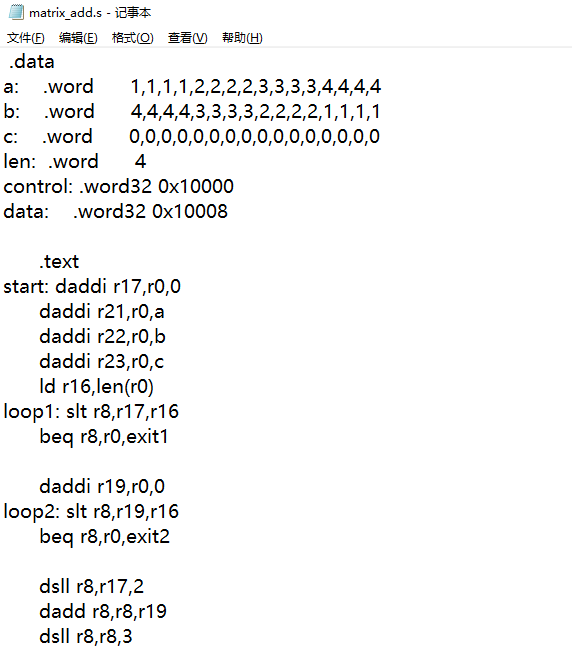


图 1 部分代码

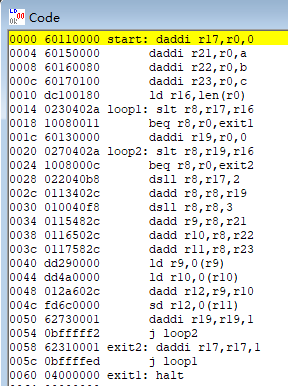


图2 代码load到WINMIPS64中

1. 未启用forwarding，运行原始代码并观察Statistics窗口，如图3所示。产生了220个RAW Stalls，即存在数据相关导致流水线暂停。

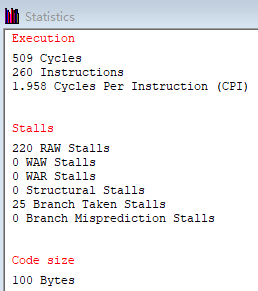


图 3 Statistics窗口

1. 观察代码，可以发现多处数据相关，如图4所示。

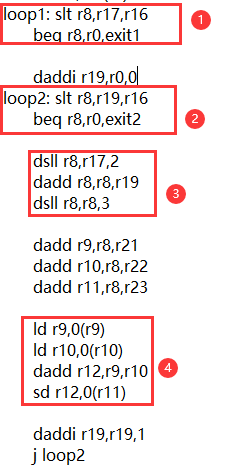


图 4 多处数据相关

在①处，slt指令中使用了r8寄存器，而下一条beq的指令也使用到了r8寄存器，产生了数据相关，②处同理。

在③处，dsll指令修改了r8寄存器的值，在下一条指令中r8寄存器的值被使用，产生数据相关。

在④处，dadd 指令使用了在前面的 ld 指令中加载的数据，存在数据相关。相加之后的指令又继续调用了r12寄存器的值，又产生了数据相关。

1. **调整指令序列优化与分析**
2. 根据上述分析，观察到几条daddi指令的执行与顺序无关，且所使用的寄存器也与数据相关处无关，故对数据相关处的指令顺序进行调整，如图5所示，而①②处的调整与判断指令的顺序不可随意更改，因此未调整①②处的指令顺序。

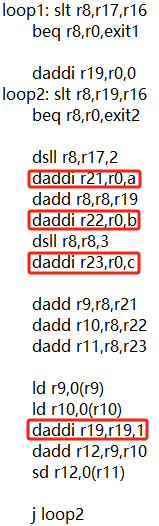


图 5 调整指令序列

1. 调整后运行代码，结果如图6所示：

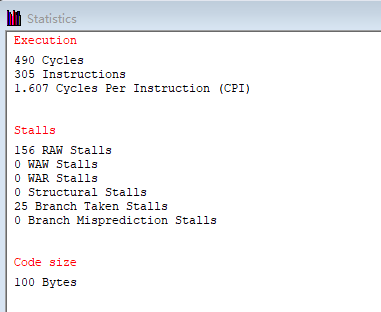


图 6 调整指令序列后Statistics窗口

RAW Stalls的值变为156，比初始结果减少了64，而Cycles从509变为了490，说明指令序列调整成功减少了RAW相关的次数，流水线效率提高。

1. **Forwarding功能开启优化与分析**
2. 在Configure选项中开启forwarding

运行原始代码，结果如图7所示，RAW相关的次数减少到了42，Cycles从509变为了360：

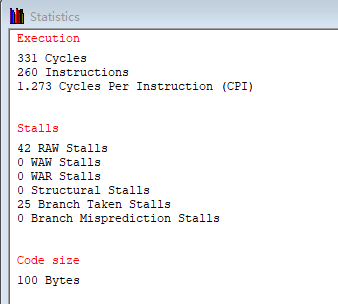


图 7 Forwarding开启(原始代码)

而使用指令序列调整后的代码，运行结果如图8 所示，RAW相关的次数减少到了26，Cycles从490变为了360，说明效率提高。

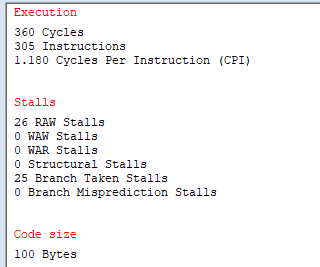


图 8 Forwarding开启(调整指令序列后)

1. 以序列调整后的代码为例，观察比较其Forwarding开启前后Cycles窗口的区别。

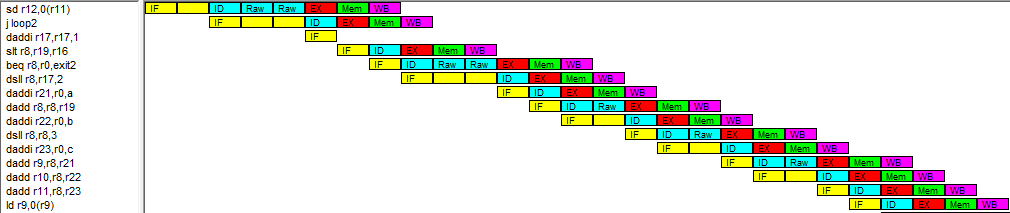


图 9 开启前Cycles窗口

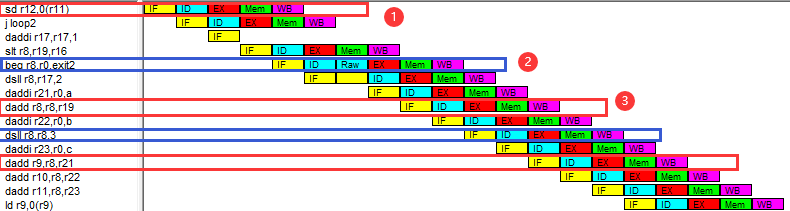


图 10 开启后Cycles窗口

结合图3所示的原始代码中存在的多处数据相关，分析可知，开启Forwarding后，图4第④处sd产生的数据相关被消除了（对应图10的①），图4第①②处存在的两处RAW相关被减为了一个（对应图10的②），第③④在调整指令序列后，仍存在一个RAW相关，开启之后被消除（对应10的③）。

1. **结构相关优化与分析**
2. 说明：给出的代码运行结果，与截图并不相同，如图11所示，并不包含结构相关。

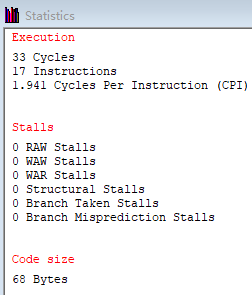


图 11 不包含结构相关

1. **根据截图的Cycles窗口调整代码**，运行结果出现结构相关，如图12所示。

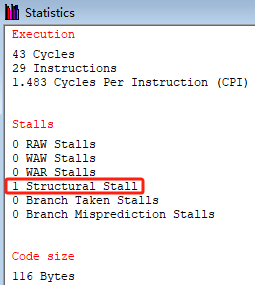


图 12 包含结构相关

修改后的代码如下：

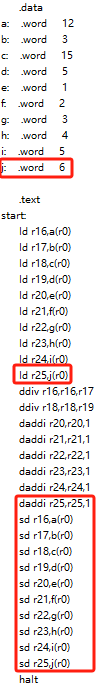


图13 修改后的代码截图

1. 观察此代码发现，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：



图14 阻塞了9个周期

为了提高流水线的效率，可以调整指令序列。因为两条除法指令仅使用到了r16，r17，r18，r19寄存器，因此可以将数据载入其他寄存器和使用其他寄存器进行加法的指令塞入其中，修改后的代码如图15所示。

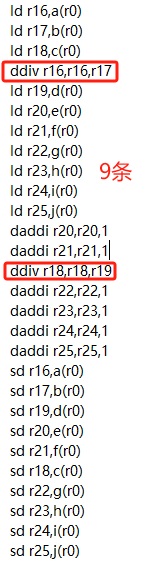


图 15 调整指令序列(除法)

1. 运行代码并观察，Statistics窗口如图16所示，Cycles窗口如图17所示。Cycles从43降至34，指令运行周期减少，说明调整序列后的流水线效率提高。

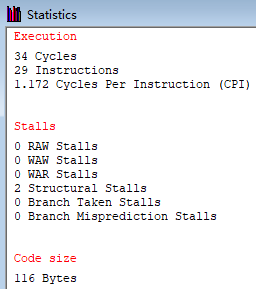


图 16 Statistics窗口(除法)

1. 从图15和图16可以看出，存在两个结构相关，与原本相比多了一个。

**分析原因：**

原始代码存在连续的除法指令，第二个除法指令阻塞9个周期，**虽然阻塞，但并没有被视作结构相关**，执行到MEM阶段时，会与后面的某一条指令发生结构相关。

而调整指令序列后，两条指令都会分别与后面的某一条指令产生结构相关。虽然结果显示结构相关数增加，但Cycles值是减小的，说明经过指令序列调整，流水线的效率提高。

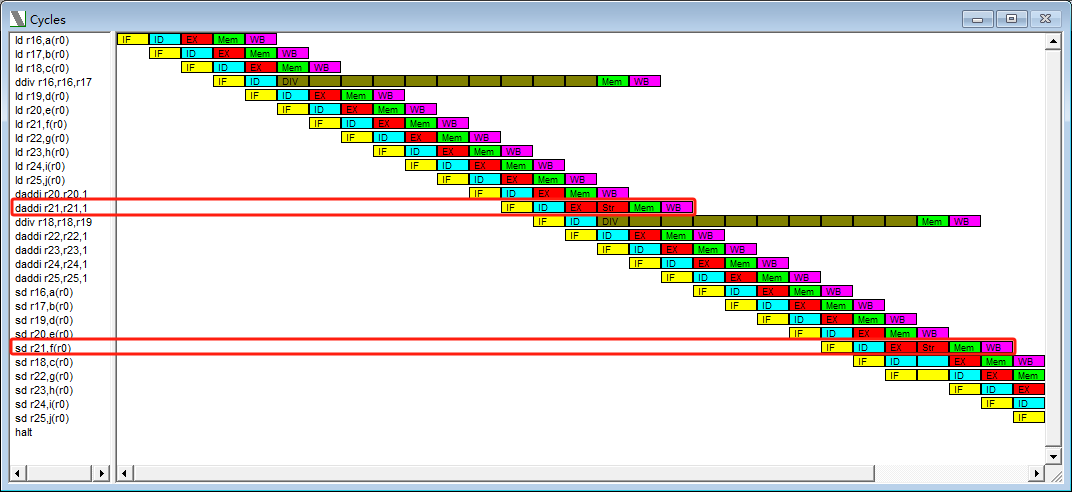
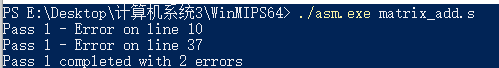


图 17 Cycles窗口(除法)

**六、实验总结与体会**

**问题：**

1. 矩阵相加的代码语法检验错误



解决：根据错误提示分析，发现“:”后面未加上空格，补上后语法正确。



图 16 语法正确性检验

1. 实验给出的除法指令存在缺陷：没有将计算结果存储回内存当中，且运行结果与截图不符。

解决：

1. 在代码末尾加上sd指令，将计算结果存储回内存当中。
2. 观察给出的Cycles窗口截图，发现没有执行j = j+1 的操作，据此增补代码，并使用相应的代码进行后续的实验。

**总结与体会**

1. 通过本次实验，我对MIPS的五级流水线有了更加深入的理解，深刻理解了数据相关性对流水线性能的影响，并学会了如何简单观察与识别代码中存在的数据相关。
2. 我学会了通过调整指令序列和启用 Forwarding 功能来优化流水线性能的方法。Forwarding具有以下作用

①减少数据冒险： 启用 Forwarding 功能后，在执行需要等待数据的指令时，直接从计算单元传递数据到需要的指令，避免了等待写入寄存器的时间，减少了数据冒险。

②性能提升： Forwarding 提高了整体性能，减少了由于数据相关性而导致的流水线停顿，使得指令流水线更加充分利用。

1. 我学会了如何通过分析Cycles窗口来识别流水线中的冒险，并结合statistics窗口的结果来优化指令，提高流水线效率。
2. 我对结构相关有了进一步的了解，经过学习，我掌握了解决结构冒险的初步方法。如插入气泡，适当增加硬件，或设计独立硬件，进行流水线分段等。

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