计算机系统结构实验

实验（三）指令调度和分支延迟

姓　　名	熊恪峥
学　　号	22920202204622
日　　期	2023年4月19日
学　　院	信息学院
课程名称	计算机系统结构

实验（三）指令调度和分支延迟

目录

1 实验目的

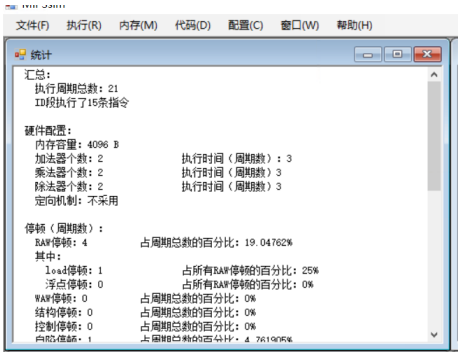
1. 加深对循环级并行性、指令调度技术、循环展开技术以及寄存器换名技术的理解；
2. 熟悉用指令调度技术来解决流水线中的数据相关的方法；
3. 了解指令调度、循环展开等技术对CPU性能的改进。

2 实验内容

2.1 用指令调度技术解决流水线中的结构冲突与数据冲突

调度前，关闭定向功能，执行结果如图 1a。总周期33，RAW6次，WAW2次。

(a) 调度前



(b) 调度后

可以发现代码中1、11、12；1、2、3；1，4；1，5，6；1，7；等行需要进行调度以消除冲突，为此，可以把有冲突的指令放置在较远的位置，这样就可以减少冲突的发生。进行调度之后代码如代码 1所示，总周期减少到了21，RAW减少到了4次，WAW减少到了0次。

代码 1 修改后

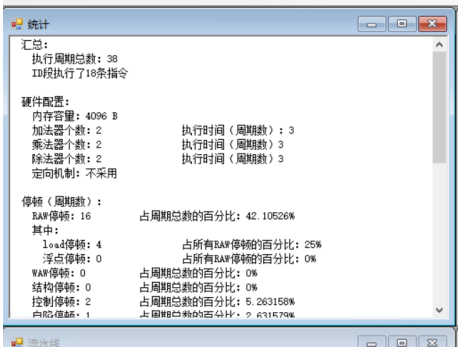
```
.text
main:
ADDIU  $r1,$r0,A
MUL    $r22,$r20,$r14
LW     $r2,0($r1)
MUL    $r24,$r26,$r14
ADD    $r4,$r0,$r2
LW     $r6,4($r1)
SW     $r4,0($r1)
ADD    $r8,$r6,$r1
MUL    $r12,$r10,$r1
ADD    $r18,$r16,$r1
ADD    $r16,$r12,$r1
SW     $r18,16($r1)
LW     $r20,8($r1)
TEQ $r0,$r0
.data
A:
.word 4,6,8
```

因此，加速比如(1)。

$$S = \frac{33}{21} = 1.571 \tag{1}$$

2.2 用延迟分支减少分支指令对性能的影响

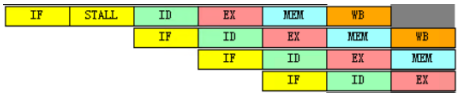
通过菜单中“配置”、“延迟槽”选项关闭分支延迟，执行结果如图 2a，时钟周期如图 2c。可以发现在分支指令中进行了较多的停顿。总共执行了38个周期。



(a) 不使用延迟

(b) 使用延迟

(c) 不使用延迟的时钟周期



(d) 使用延迟的时钟周期

假设延迟槽有一个，将指令LW放入延迟槽中，并通过上述菜单选项打开延迟槽功能，执行结果如图 2b，时钟周期如图 2d。周期总数降至25，并且从时钟周期看出分支指令的停顿减少了。

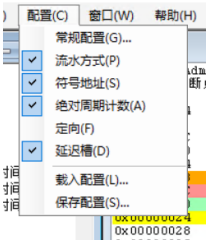
代码 2 修改后

```
.text
main:
ADDI  $r2,$r0,1024
ADD   $r3,$r0,$r0
ADDI  $r4,$r0,8
loop:
LW    $r1,0($r2)
ADDI  $r3,$r3,4
ADDI  $r1,$r1,1
SUB   $r5,$r4,$r3
SW    $r1,0($r2)
BGTZ  $r5,loop
LW    $r1,0($r2)
ADD   $r7,$r0,$r6
TEQ   $r0,$r0
```

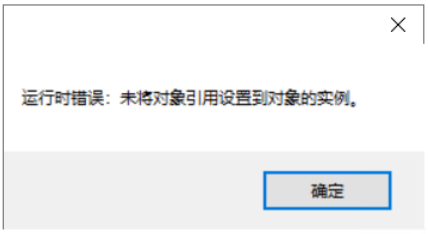
2.3 补充实验

2.3.1 解决不能使用L.D和S.D的问题

补充实验中需要用到浮点数加载和写回的指令L.D和S.D，然而由于MIPSSim固有的问题，这些指令直接使用时汇报如图 3a的错误。为了解决这个错误，需要如图 3b启用延时槽，这样这些指令的运行就能正常。



(a) 启动延迟槽



(b) L.D报错信息

2.3.2 实现

在补充实验中，我实现了算法 1中的循环，并且为了演示控制冲突，在分支指令后增加了一个运算操作。代码如代码 3所示。

算法 1 实现的循环运算

1: **for** $i \in \{1, \ldots, 16\}$ **do**
2: $x[i] \leftarrow x[i] + x[i]$

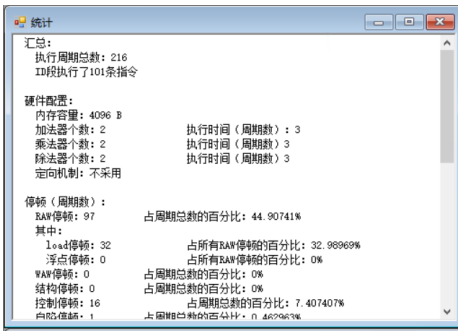
代码 3 展开前

```
.text
main:
ADDI  $r2,$r0,1024
ADDI  $r3,$r0,16
loop:
L.D    $f1,0($r2)
ADD.D  $f1,$f1,$f1
S.D    $f1,0($r2)
ADDI  $r2,$r2,4
ADDI   $r3,$r3,-1
BGTZ  $r3,loop
ADD   $r7,$r0,$r6
TEQ   $r0,$r0
```

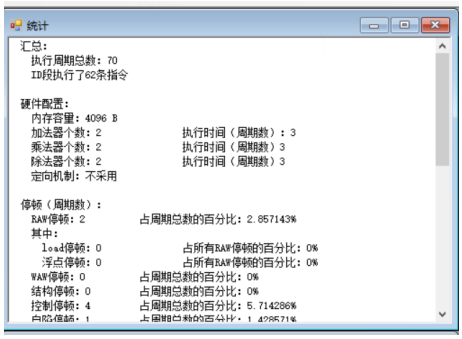
执行展开前的代码，结果如图 4a所示，总周期为216，RAW97次，WAW0次。进行循环展开、并根据指令的换名和调度，然后将指令调度到延迟槽中，得到代码 4。结果如图 4b所示，

代码 4 展开后

```
.text
main:
ADDI  $r2,$r0,1024
ADDI  $r3,$r0,16
loop:
ADDI   $r3,$r3,-4
L.D    $f1,0($r2)
L.D    $f10,4($r2)
L.D    $f11,8($r2)
L.D    $f12,12($r2)
ADD.D  $f1,$f1,$f1
ADD.D  $f10,$f10,$f10
ADD.D  $f11,$f11,$f11
ADD.D  $f12,$f12,$f12
S.D    $f1,0($r2)
S.D    $f10,4($r2)
S.D    $f11,8($r2)
S.D    $f12,12($r2)
BGTZ   $r3,loop
ADDI   $r2,$r2,16
ADD    $r7,$r0,$r6
TEQ    $r0,$r0
```



(a) 调度前



(b) 调度后

思考题：

1. **当定向技术打开和关闭时结果是否有差异？**有，图 4b中展示的是使用延迟槽的结果。在未使用延迟槽时，时钟周期数稍多。需要注意的是，由于上节提到的问题， *在实验时使用了删除了L.D和S.D的程序比较性能。*

2. **Stall是否越少越好？**是的，通过减少Stall，可以使程序执行加快提高CPU的性能。当然，这可能降低程序的可读性，但是由编译器来完成这项工作时就不会有可读性的问题产生。

3 实验总结

在此次实验中，我加深了对循环级并行性、指令调度技术、循环展开技术以及寄存器换名技术的理，熟悉了用指令调度技术来解决流水线中的数据相关的方法，了解了指令调度、循环展开等技术对CPU性能的改进。并且通过实际编写代码的方式体验了循环展开减少停顿的方法。