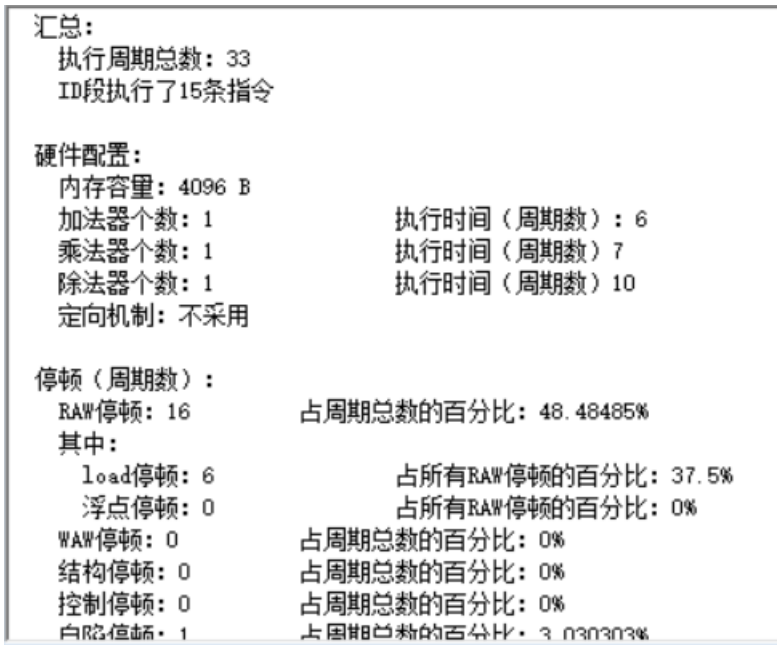
计算机系统结构实验报告

<table>
<tr><td>班级</td><td>计科 1 班</td><td>实验日期</td><td>2024..4.2</td><td>实验成绩</td><td></td></tr>
<tr><td>姓名</td><td>李梓涵</td><td>学号</td><td colspan="3">34520212201574</td></tr>
<tr><td>实验名称</td><td colspan="5">实验 3 指令调度和分支延迟</td></tr>
<tr><td>实验目的、要求</td><td colspan="5">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；
熟悉用指令调度技术来解决流水线中的数据相关的方法；
了解指令调度、循环展开等技术对 CPU 性能的改进。</td></tr>
<tr><td>实验内容、步骤及结果</td><td colspan="5">1. 用指令调度技术解决流水线中的结构冲突与数据冲突
1.1 执行载入的程序
调度前关闭定向功能，执行结果如图，周期总数为 33，RAW16 次
 
汇总:
执行周期总数: 33
ID段执行了15条指令

硬件配置:
内存容量: 4096 B
加法器个数: 1 执行时间（周期数）: 6
乘法器个数: 1 执行时间（周期数）7
除法器个数: 1 执行时间（周期数）10
定向机制: 不采用

停顿（周期数）:
RAW停顿: 16 占周期总数的百分比: 48.48485%
其中:
load停顿: 6 占所有RAW停顿的百分比: 37.5%
浮点停顿: 0 占所有RAW停顿的百分比: 0%
WAW停顿: 0 占周期总数的百分比: 0%
结构停顿: 0 占周期总数的百分比: 0%
控制停顿: 0 占周期总数的百分比: 0%
阅读代码可以发现，3 4 5 行、5 6 7 行、7 8 行、9 10 11 12 行均存在冲突。</td></tr>
</table>

```
3   ADDIU  $r1,$r0,A
4   LW     $r2,0($r1)
5   ADD    $r4,$r0,$r2
6   SW     $r4,0($r1)
7   LW     $r6,4($r1)
8   ADD    $r8,$r6,$r1
9   MUL    $r12,$r10,$r1
10  ADD    $r16,$r12,$r1
11  ADD    $r18,$r16,$r1
12  SW     $r18,16($r1)
13  LW     $r20,8($r1)
14  MUL    $r22,$r20,$r14
15  MUL    $r24,$r26,$r14
16  TEQ $r0,$r0
```

1.2 执行采用指令调度技术对程序进行指令调度，消除冲突后的程序执行总周期数降为了 21 次，且 RAW 停顿为 4 个

```
汇总:
  执行周期总数: 21
  ID段执行了15条指令

硬件配置:
  内存容量: 4096 B
  加法器个数: 1                执行时间（周期数）: 6
  乘法器个数: 1                执行时间（周期数）7
  除法器个数: 1                执行时间（周期数）10
  定向机制: 不采用

停顿（周期数）:
  RAW停顿: 4          占周期总数的百分比: 19.04762%
  其中:
    load停顿: 1               占所有RAW停顿的百分比: 25%
    浮点停顿: 0               占所有RAW停顿的百分比: 0%
  WAW停顿: 0          占周期总数的百分比: 0%
  结构停顿: 0         占周期总数的百分比: 0%
  控制停顿: 0         占周期总数的百分比: 0%
  自陷停顿: 1         占周期总数的百分比: 4.761905%
```

修改后的代码如下：

```
.text
main:
ADDIU   $r1,$r0,A
MUL     $r24,$r26,$r14
LW      $r2,0($r1)
LW      $r20,8($r1)
ADD     $r4,$r0,$r2
MUL     $r12,$r10,$r1
LW      $r6,4($r1) #3
MUL     $r22,$r20,$r14
ADD     $r16,$r12,$r1
SW      $r4,0($r1)
ADD     $r18,$r16,$r1
ADD     $r8,$r6,$r1
SW      $r18,16($r1)
TEQ $r0,$r0

.data
A:
.word 4,6,8
```

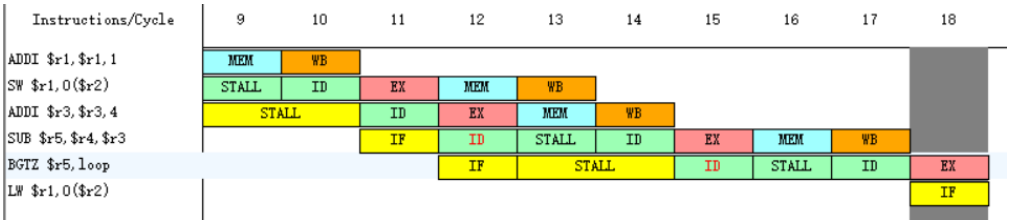
1.3 性能比较

加速比为 S = 33 / 21 = 1.57

2. 用延迟分支减少分支指令对性能的影响

2.1 执行 branch.s 并记录发生分支延迟的时刻

？？？？如图所示，在 13 到 17 周期均为分支延迟



2.2 记录总周期时钟数

如图总的周期始终数是 38

```
汇总：
  执行周期总数：38
  ID段执行了18条指令

硬件配置：
  内存容量：4096 B
  加法器个数：1                执行时间（周期数）：6
  乘法器个数：1                执行时间（周期数）7
  除法器个数：1                执行时间（周期数）10
  定向机制：不采用

停顿（周期数）：
  RAW停顿：16          占周期总数的百分比：42.10526%
  其中：
    load停顿：4                占所有RAW停顿的百分比：25%
    浮点停顿：0                占所有RAW停顿的百分比：0%
  WAW停顿：0           占周期总数的百分比：0%
  结构停顿：0          占周期总数的百分比：0%
  控制停顿：2          占周期总数的百分比：5.263158%
```

代码如下：

delayed-branch.s

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

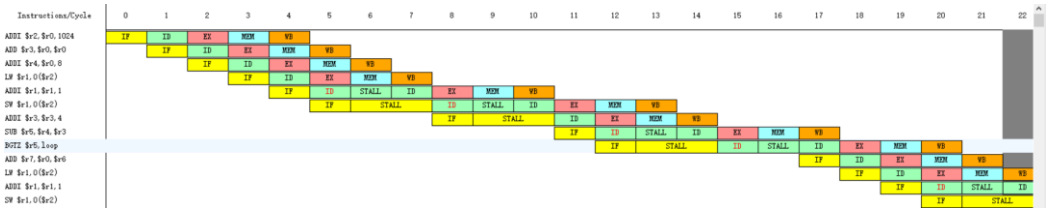
2.3 打开分支延迟功能，执行 delayed-branch.s 程序

经过增加分支延迟槽和应用指令角度技术后，周期总数降为 26

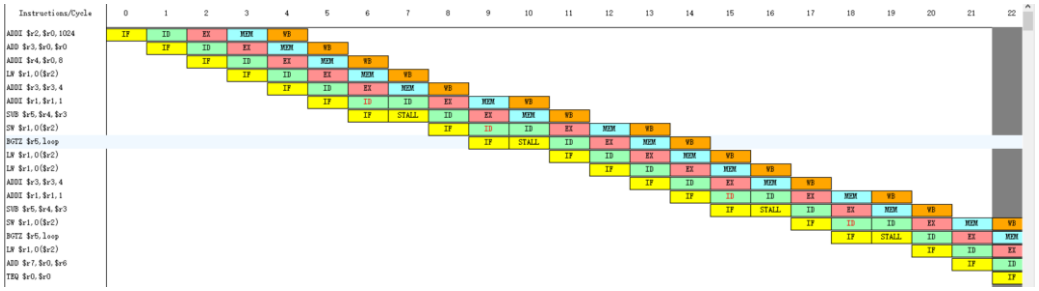
```
汇总：
  执行周期总数：26
  ID段执行了20条指令

硬件配置：
  内存容量：4096 B
  加法器个数：1                执行时间（周期数）：6
  乘法器个数：1                执行时间（周期数）7
  除法器个数：1                执行时间（周期数）10
  定向机制：不采用

停顿（周期数）：
  RAW停顿：4          占周期总数的百分比：15.38461%
  其中：
    load停顿：2              占所有RAW停顿的百分比：50%
    浮点停顿：0              占所有RAW停顿的百分比：0%
  WAW停顿：0          占周期总数的百分比：0%
  结构停顿：0          占周期总数的百分比：0%
  控制停顿：0          占周期总数的百分比：0%
  自RAW停顿：1         占周期总数的百分比：3.8461546
```

2.4 比较时钟周期图

由下面两图对比可知，分支延迟槽减少了分支指令执行时的等待时间和分支之后指令的等待时间。



a.无分支延迟槽的指令周期图



b.增加分支延迟槽后的时钟周期图

2.5 性能比较

加速比：S = 38 / 26 = 1.46

当延迟槽中的指令是有用的指令时，延迟槽能较少延迟等待，提高 CPU 性能。

3. 补充实验 1

代码如下：

```
1   .text
2   main:
3   ADDI $r2, $r0, 1024
4   ADDI $r3, $r0, 16
5   loop:
6   L.D  $f1, 0($r2)
7   ADD.D $f1,$f1,$f1
8   S.D  $f1, 0($r2)
9   ADDI $r2, $r2, 4
10  ADDI $r3, $r3, -1
11  BGTZ $r3, loop
12  ADD $r7, $r0, $r6
13  TEQ $r0, $r0
```

其中 9 10 行的 ADDI 指令由于均操作存储器，产生结构冲突；10 11 行的两条指令一个要写入寄存器 r3，另一个要读寄存器 r3，产生数据冲突；6 7 行均操作寄存器 f1，产生数据冲突。

```
汇总:
  执行周期总数: 134
  ID段执行了84条指令

硬件配置:
  内存容量: 4096 B
  加法器个数: 2                执行时间（周期数）: 3
  乘法器个数: 2                执行时间（周期数）3
  除法器个数: 2                执行时间（周期数）3
  定向机制: 不采用

停顿（周期数）:
  RAW停顿: 32          占周期总数的百分比: 23.8806%
  其中:
    load停顿: 0                占所有RAW停顿的百分比: 0%
    浮点停顿: 0                占所有RAW停顿的百分比: 0%
  WAW停顿: 0           占周期总数的百分比: 0%
  结构停顿: 16                 占周期总数的百分比: 11.9403%
  控制停顿: 0          占周期总数的百分比: 0%
  自陷停顿: 1          占周期总数的百分比: 0.7462686%
```

4. 补充实验 2

   对之前的代码进行循环展开、分支延迟、寄存器换名和指令调度后，得到的代码如下：

3.2.s

```
.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) # 寄存器换名 指令调度 循环展开

ADD.D $f1,$f1,$f1
ADD.D $f10,$f10,$f10

L.D  $f11, 8($r2)
L.D  $f12, 12($r2)

S.D  $f1, 0($r2)
S.D  $f10, 4($r2)

ADD.D $f11,$f11,$f11
ADD.D $f12,$f12,$f12

S.D  $f11, 8($r2)
S.D  $f12, 12($r2)

BGTZ $r3, loop
ADDI $r2, $r2, 16

ADD $r7, $r0, $r6
TEQ $r0, $r0
```

执行总周期数：

```
汇总:
  执行周期总数: 49
  ID段执行了33条指令

硬件配置:
  内存容量: 4096 B
  加法器个数: 2                执行时间（周期数）: 3
  乘法器个数: 2                执行时间（周期数）3
  除法器个数: 2                执行时间（周期数）3
  定向机制: 不采用

停顿（周期数）:
  RAW停顿: 2          占周期总数的百分比: 4.081633%
  其中:
    load停顿: 0               占所有RAW停顿的百分比: 0%
    浮点停顿: 0               占所有RAW停顿的百分比: 0%
  WAW停顿: 0          占周期总数的百分比: 0%
  结构停顿: 12                占周期总数的百分比: 24.4898%
  控制停顿: 0         占周期总数的百分比: 0%
  自陷停顿: 1         占周期总数的百分比: 2.040816%
```

性能比较：S = 134 / 49 = 2.73

5. 当定向技术打开和关闭时结果是否有差异
   有差异，如图时开启定向时的执行效果，可以看出，总周期减少了 2，RAW 停顿也减少了。

```
汇总:
  执行周期总数: 47
  ID段执行了33条指令

硬件配置:
  内存容量: 4096 B
  加法器个数: 2                执行时间（周期数）: 3
  乘法器个数: 2                执行时间（周期数）3
  除法器个数: 2                执行时间（周期数）3
  定向机制: 采用

停顿（周期数）:
  RAW停顿: 0          占周期总数的百分比: 0%
  其中:
    load停顿: 0               占所有RAW停顿的百分比: 0%
    浮点停顿: 0               占所有RAW停顿的百分比: 0%
  WAW停顿: 0          占周期总数的百分比: 0%
  结构停顿: 12                占周期总数的百分比: 25.53192%
  控制停顿: 0         占周期总数的百分比: 0%
  自陷停顿: 1         占周期总数的百分比: 2.12766%
```

6. Stall 是否越少越好
   是的，Stall 越少，停顿越少，执行速度越快，但是必要的 Stall 不能减少，且 stall 降低到一定数量后再降低的效果不明显。

总结	1. 循环展开加上寄存器换名的方法能有效地减少数据冲突和结构冲突 2. 进行指令调度时，要注意原本指令执行的效果，不能打乱寄存器的存取顺序，修改后虽然能避免数据冲突，但会导致结果是错误的。