北京郵電大學

实验报告

题目：使用 MIPS 指令实现冒泡排序法

班　　级：2019211306

学　　号：2019211397

姓　　名：毛子恒

学　　院：计算机学院

2022 年 5 月 1 日

一、 实验目的

（1）掌握静态调度方法。
（2）增强汇编语言编程能力。
（3）学会使用模拟器中的定向功能进行优化。

二、 实验内容

（1）自行编写一个实现冒泡排序的汇编程序，该程序要求可以实现对一维整数数组进行冒泡排序。冒泡排序算法的运作如下:
①比较相邻的元素。如果第一个比第二个大，就交换他们两个。
②对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。在这一点，最后的元素应该会是最大的数。
③针对所有的元素重复以上的步骤，除了最后一个。
④持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。
要求数组长度不得小于 10。
（2）启动 MIPSsim。
（3）载入自己编写的程序，观察流水线输出结果。
（4）使用定向功能再次执行代码，与刚才执行结果进行比较，观察执行效率的不同。
（5）采用静态调度方法重排指令序列，减少相关，优化程序。
（6）对优化后的程序使用定向功能执行，与刚才执行结果进行比较，观察执行效率的不同。
注意：不要使用浮点指令及浮点寄存器。整数减勿使用 SUB 指令，请使用 DSUB 指令代替。

三、 实验平台和环境

指令级和流水线操作级模拟器 MIPSsim。

四、 实验步骤及实验分析

（1）自行编写一个实现冒泡排序的汇编程序:

```
.text
main:
ADDIU   $r1, $r0, a # 取a地址
ADDIU   $r2, $r0, n # 取n地址
LW      $r2, 0($r2) # 取n
SLL     $r2, $r2, 2 # n<<=2
ADD     $r2, $r2, $r1   # 取a[n]地址
loop1:
ADDI    $r2, $r2, -4 # n--
BEQ     $r2, $r1, end   # n==1 -> end
ADDIU   $r3, $r1, 0 # j=0，取a[j]地址
loop2:
LW      $r4, 0($r3) # 取a[j]
LW      $r5, 4($r3) # 取a[j+1]
SLT     $r6, $r5, $r4   # a[j+1]<a[j]?
BEQ     $r6, $r0, iter  # a[j+1]>=a[j] -> iter
SW      $r5, 0($r3) # 存a[j+1]
SW      $r4, 4($r3) # 存a[j]
iter:
ADDI    $r3, $r3, 4 # j++
BNE     $r3, $r2, loop2 # j!=n -> loop2
BEQ     $r0, $r0, loop1 # -> loop1
end:
TEQ     $r0, $r0

.data
a:
.word 10,9,8,7,6,5,4,3,2,1
n:
.word 10
```

(3) 载入自己编写的程序，观察流水线输出结果。

汇总：

执行周期总数：831

ID段执行了405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：316　　占周期总数的百分比：38.02647%

其中：

load停顿：92　　占所有RAW停顿的百分比：29.11392%

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：109　　占周期总数的百分比：13.11673%

自陷停顿：0　　占周期总数的百分比：0%

停顿周期总数：425　占周期总数的百分比：51.1432%

分支指令：

指令条数：109　　占指令总数的百分比：26.91358%

其中：

分支成功：46　　占分支指令数的百分比：42.20184%

分支失败：63　　占分支指令数的百分比：57.79816%

load/store指令：

指令条数：181　　占指令总数的百分比：44.69136%

其中：

load：91　　占load/store指令数的百分比：50.27624%

store：90　　占load/store指令数的百分比：49.72376%

浮点指令：

指令条数：0　　占指令总数的百分比：0%

其中：

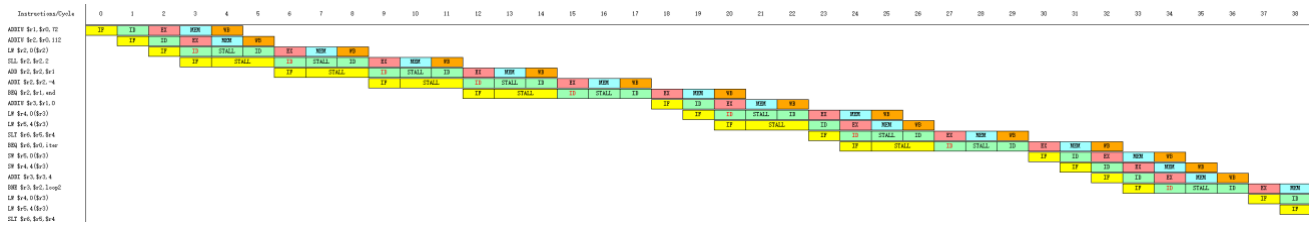
加法：0　　占浮点指令数的百分比：0%

乘法：0　　占浮点指令数的百分比：0%

除法：0　　占浮点指令数的百分比：0%

自陷指令：

指令条数：1　　占指令总数的百分比：0.2469136%



其中有相当一部分指令都发生了 RAW 冲突。

(4) 使用定向功能再次执行代码，与刚才执行结果进行比较，观察执行效率的不同。

汇总：

执行周期总数：661

ID段执行了405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：采用

停顿（周期数）：

RAW停顿：146　　占周期总数的百分比：22.08775%

其中：

load停顿：46　　占所有RAW停顿的百分比：31.50685%

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：109　　占周期总数的百分比：16.49017%

自陷停顿：0　　占周期总数的百分比：0%

停顿周期总数：255　占周期总数的百分比：38.57791%

分支指令：

指令条数：109　　占指令总数的百分比：26.91358%

其中：

分支成功：46　　占分支指令数的百分比：42.20184%

分支失败：63　　占分支指令数的百分比：57.79816%

load/store指令：

指令条数：181　　占指令总数的百分比：44.69136%

其中：

load：91　　占load/store指令数的百分比：50.27624%

store：90　　占load/store指令数的百分比：49.72376%

浮点指令：

指令条数：0　　占指令总数的百分比：0%

其中：

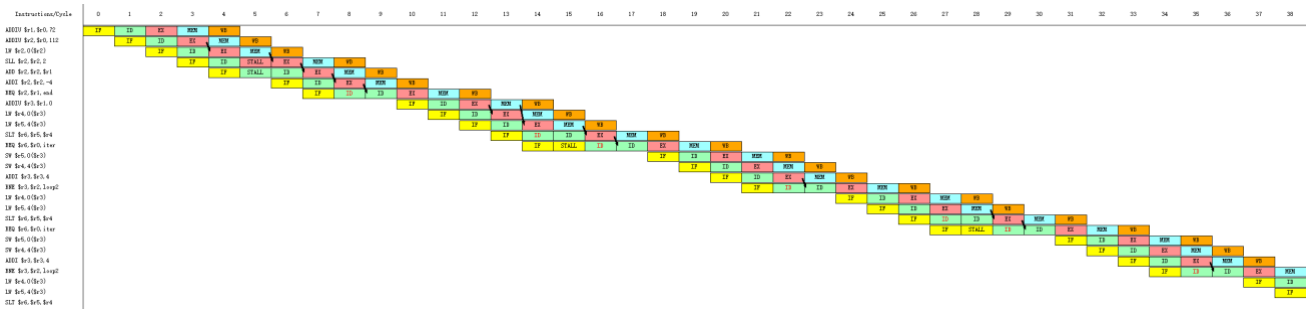
加法：0　　占浮点指令数的百分比：0%

乘法：0　　占浮点指令数的百分比：0%

除法：0　　占浮点指令数的百分比：0%

自陷指令：

指令条数：1　　占指令总数的百分比：0.2469136%



定向功能消除了部分数据冲突，但是当相邻的指令发生 RAW 冲突时，仍然会停顿一个周期。性能提升了 831/661=1.26 倍。

（5）采用静态调度方法重排指令序列，减少相关，优化程序。

```
.text
main:
ADDIU   $r2, $r0, n # 取 n 地址
ADDIU   $r1, $r0, a # 取 a 地址
LW      $r2, 0($r2) # 取 n
SLL     $r2, $r2, 2 # n<<=2
ADD     $r2, $r2, $r1   # 取 a[n]地址
loop1:
ADDI    $r2, $r2, -4 # n--
ADDIU   $r3, $r1, 0 # j=0，取 a[j]地址
BEQ     $r2, $r1, end   # n==1 -> end
loop2:
LW      $r4, 0($r3) # 取 a[j]
LW      $r5, 4($r3) # 取 a[j+1]
SLT     $r6, $r5, $r4   # a[j+1]<a[j]?
ADDI    $r3, $r3, 4 # j++
BEQ     $r6, $r0, iter  # a[j+1]>=a[j] -> iter
SW      $r5, -4($r3)    # 存 a[j+1]
SW      $r4, 0($r3) # 存 a[j]
iter:
BNE     $r3, $r2, loop2 # j!=n -> loop2
BEQ     $r0, $r0, loop1 # -> loop1
end:
TEQ     $r0, $r0

.data
a:
.word 10,9,8,7,6,5,4,3,2,1
n:
.word 10
```

(6) 对优化后的程序使用定向功能执行，与刚才执行结果进行比较，观察执行效率的不同。

汇总：

执行周期总数：562

ID段执行了406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：采用

停顿（周期数）：

RAW停顿：46　　占周期总数的百分比：8.185054%

其中：

load停顿：46　　占所有RAW停顿的百分比：100%

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：109　　占周期总数的百分比：19.39502%

自陷停顿：0　　占周期总数的百分比：0%

停顿周期总数：155　占周期总数的百分比：27.58007%

分支指令：

指令条数：109　　占指令总数的百分比：26.84729%

其中：

分支成功：46　　占分支指令数的百分比：42.20184%

分支失败：63　　占分支指令数的百分比：57.79816%

load/store指令：

指令条数：181　　占指令总数的百分比：44.58128%

其中：

load：91　　占load/store指令数的百分比：50.27624%

store：90　　占load/store指令数的百分比：49.72376%

浮点指令：

指令条数：0　　占指令总数的百分比：0%

其中：

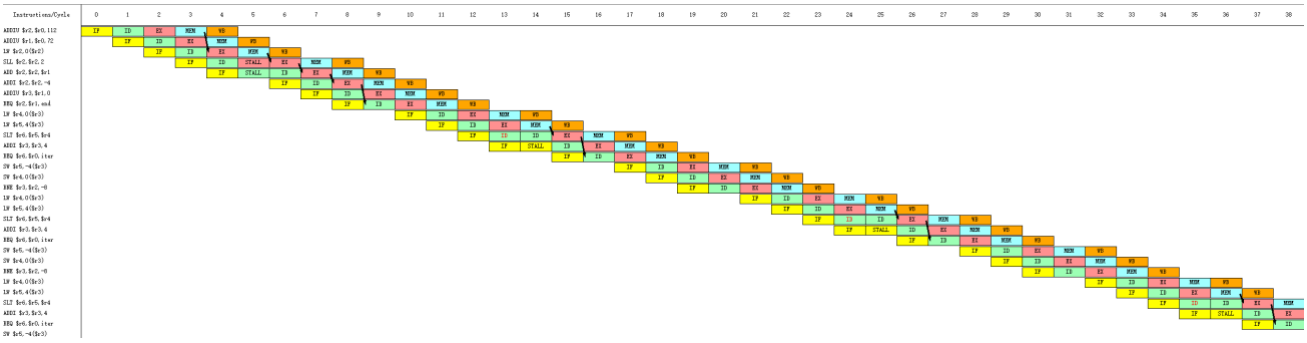
加法：0　　占浮点指令数的百分比：0%

乘法：0　　占浮点指令数的百分比：0%

除法：0　　占浮点指令数的百分比：0%

自陷指令：

指令条数：1　　占指令总数的百分比：0.2463054%



消除了 **LW** `$r5, 4($r3)` 和 **SLT** `$r6, $r5, $r4` 以外的所有 RAW 冲突，性能提升了 661/562=1.18 倍。