北京郵電大學

实验报告

题目：使用 MIPS 指令实现求两个数组的点积

班　　级：2019211306

学　　号：2019211397

姓　　名：毛子恒

学　　院：计算机学院

2022 年 4 月 30 日

一、 实验目的

（1）通过实验熟悉实验 1 和实验 2 的内容。
（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。
（4）了解对代码进行优化的方法。

二、 实验内容

（1）自行编写一个计算两个向量点积的汇编程序，该程序要求可以实现求两个向量点积计算后的结果向量的点积：假设有两个 n 维向量 a、b，则 a 与 b 的点积为：

$$a \cdot b = \sum_{i=1}^{n} a_i\, b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

两个向量元素使用数组进行数据存储，要求向量的维度不得小于 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，使用 TEQ $r0 $r0 结束程序。

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）自行编写一个计算两个向量点积的汇编程序:

```
.text
main:
ADDIU   $r1, $r0, a # 取 a 地址
ADDIU   $r2, $r0, b # 取 b 地址
ADDIU   $r3, $r0, n # 取 n 地址
LW      $r3, 0($r3) # 取 n
ADDIU   $r4, $r0, 0 # 初始化 ans=0
loop:
LW      $r5, 0($r1) # 取 ai
LW      $r6, 0($r2) # 取 bi
MUL     $r7, $r5, $r6   # ai*bi
ADD     $r4, $r4, $r7   # ans+=ai*bi
ADDI    $r1, $r1, 4 # 取下一个 ai 地址
ADDI    $r2, $r2, 4 # 取下一个 bi 地址
ADDI    $r3, $r3, -1    # n--
BGTZ    $r3, loop   # 循环
TEQ     $r0, $r0

.data
a:
.word 1,2,3,4,5,6,7,8,9,10
b:
.word 1,2,3,4,5,6,7,8,9,10
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：160

ID段执行了87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：62　　占周期总数的百分比：38.75%

其中：

load停顿：20　　占所有RAW停顿的百分比：32.25806%

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：10　　占周期总数的百分比：6.25%

自陷停顿：0　　占周期总数的百分比：0%

停顿周期总数：72　　占周期总数的百分比：45%

分支指令：

指令条数：10　　占指令总数的百分比：11.49425%

其中：

分支成功：9　　占分支指令数的百分比：90%

分支失败：1　　占分支指令数的百分比：10%

load/store指令：

指令条数：21　　占指令总数的百分比：24.13793%

其中：

load：21　　占load/store指令数的百分比：100%

store：0　　占load/store指令数的百分比：0%

浮点指令：

指令条数：0　　占指令总数的百分比：0%

其中：

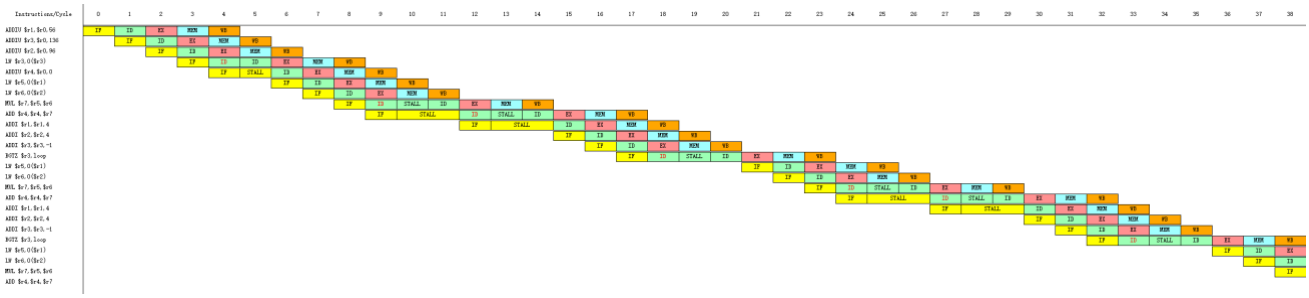
加法：0　　占浮点指令数的百分比：0%

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：1.149425%



其中主要是 **ADDIU** $r2, $r0, b 和 **LW** $r3, 0($r3)指令，以及 **LW** $r5, 0($r1)、**LW** $r6, 0($r2)和 **MUL** $r7, $r5, $r6、**ADD** $r4, $r4, $r7 指令的数据冲突。

(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：118

ID段执行了87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：20　　占周期总数的百分比：16.94915%

其中：

load停顿：10　　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：10　　占周期总数的百分比：8.474576%

自陷停顿：0　　占周期总数的百分比：0%

停顿周期总数：30　　占周期总数的百分比：25.42373%

分支指令：

指令条数：10　　占指令总数的百分比：11.49425%

其中：

分支成功：9　　占分支指令数的百分比：90%

分支失败：1　　占分支指令数的百分比：10%

load/store指令：

指令条数：21　　占指令总数的百分比：24.13793%

其中：

load：21　　占load/store指令数的百分比：100%

store：0　　占load/store指令数的百分比：0%

浮点指令：

指令条数：0　　占指令总数的百分比：0%

其中：

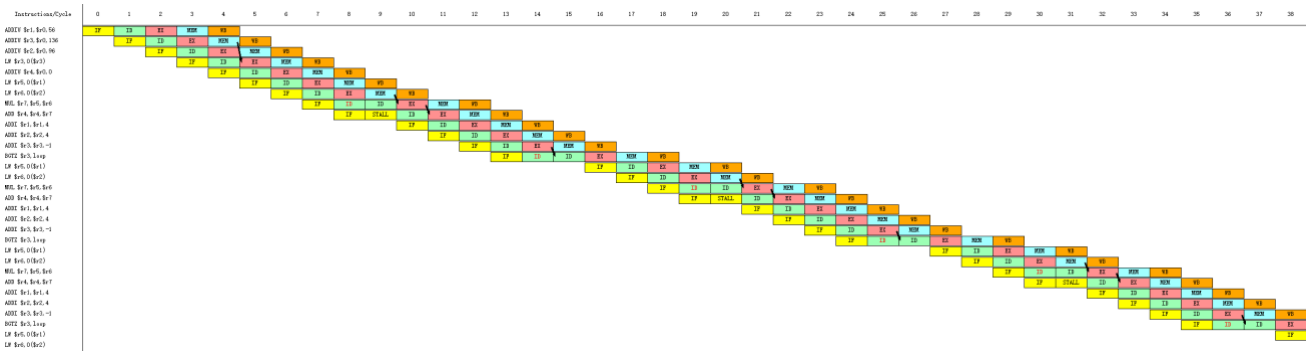
加法：0　　占浮点指令数的百分比：0%

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：1.149425%



定向功能消除了部分数据冲突，性能提升了 160/118=1.36 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   $r1, $r0, a # 取 a 地址
ADDIU   $r3, $r0, n # 取 n 地址
ADDIU   $r2, $r0, b # 取 b 地址
ADDIU   $r4, $r0, 0 # 初始化 ans=0
LW      $r3, 0($r3) # 取 n
loop:
LW      $r5, 0($r1) # 取 ai
LW      $r6, 0($r2) # 取 bi
ADDI    $r1, $r1, 4 # 取下一个 ai 地址
ADDI    $r2, $r2, 4 # 取下一个 bi 地址
MUL     $r7, $r5, $r6   # ai*bi
ADDI    $r3, $r3, -1    # n--
ADD     $r4, $r4, $r7   # ans+=ai*bi
BGTZ    $r3, loop   # 循环
TEQ     $r0, $r0

.data
a:
.word 1,2,3,4,5,6,7,8,9,10
b:
.word 1,2,3,4,5,6,7,8,9,10
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：98

ID段执行了87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：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：0　　占周期总数的百分比：0%

控制停顿：10　　占周期总数的百分比：10.20408%

自陷停顿：0　　占周期总数的百分比：0%

停顿周期总数：10　　占周期总数的百分比：10.20408%

分支指令：

指令条数：10　　占指令总数的百分比：11.49425%

其中：

分支成功：9　　占分支指令数的百分比：90%

分支失败：1　　占分支指令数的百分比：10%

load/store指令：

指令条数：21　　占指令总数的百分比：24.13793%

其中：

load：21　　占load/store指令数的百分比：100%

store：0　　占load/store指令数的百分比：0%

浮点指令：

指令条数：0　　占指令总数的百分比：0%

其中：

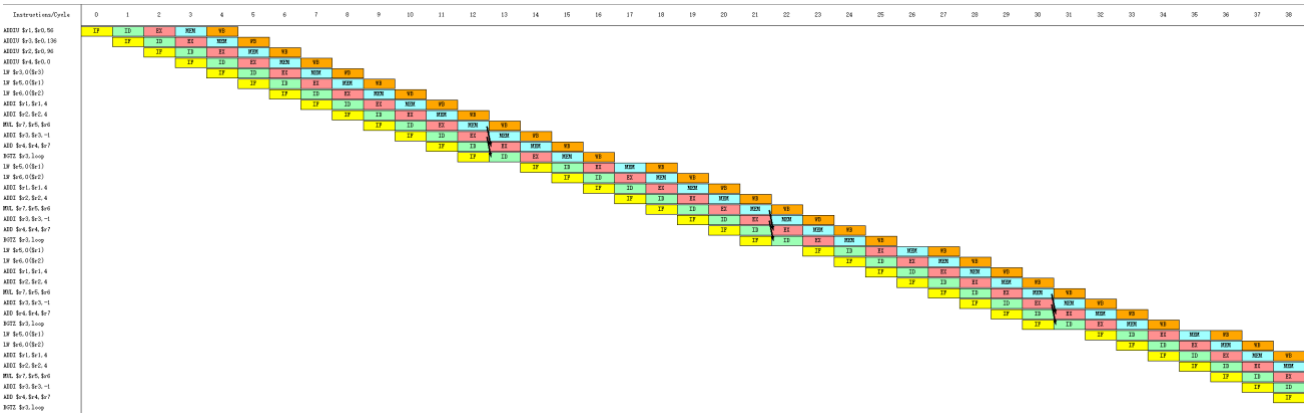
加法：0　　占浮点指令数的百分比：0%

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：1.149425%



消除了全部数据冲突，性能提升了 118/98=1.20 倍。