实验 3 使用 MIPS 指令实现求两个数组的点积

张梓良
2021212484

日期：2024 年 5 月 4 日

1 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

2 实验平台

指令级和流水线操作级模拟器 MIPSsim。

3 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。

4 实验步骤及分析

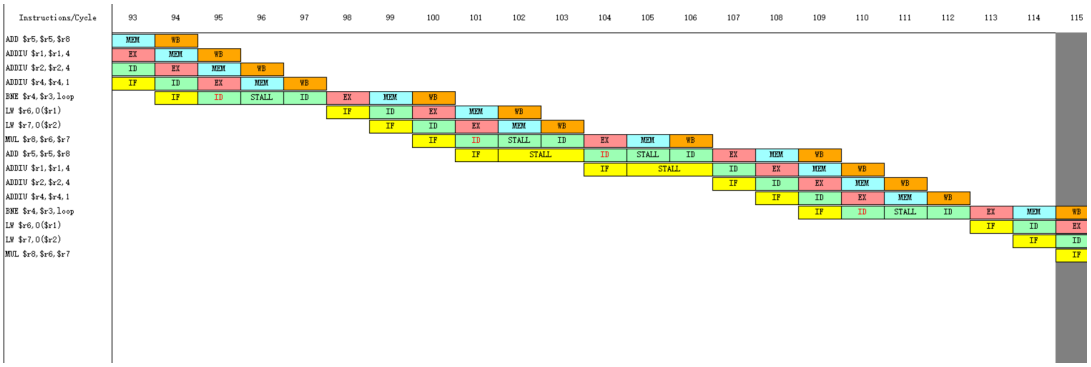
自行编写一个计算两个向量点积的汇编程序：

```
.text
main:
ADDIU  $r1,$r0,a # 取向量a的地址
ADDIU  $r2,$r0,b # 取向量b的地址
ADDIU  $r3,$r0,n # 取n的地址
BGEZAL  $r0, naive_prod # 调用naive_prod
NOP  # 无操作
TEQ $r0,$r0

naive_prod:
LW     $r3, 0($r3) # 取n的值
ADD    $r4, $r0, $r0 # 初始化 i = 0
ADD    $r5, $r0, $r0 # 初始化 ans = 0
loop:
LW     $r6, 0($r1) # 取a[i]
LW     $r7, 0($r2) # 取b[i]
MUL    $r8, $r6, $r7 # 计算a[i]*b[i]
ADD    $r5, $r5, $r8 # ans += a[i]*b[i]
ADDIU  $r1, $r1, 4 # 取a[i+1]的地址
ADDIU  $r2, $r2, 4 # 取b[i+1]的地址
ADDIU  $r4, $r4, 1 # i++
BNE    $r4, $r3, loop # i != n, 跳转到loop
JR     $r31 # 返回

.data
a:
.word 1,2,3,4,5,6,7,8,9,10,11,12
b:
.word 1,2,3,4,5,6,7,8,9,10,11,12
n:
.word 12
```

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: 193
    ID段执行了106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: 72   占周期总数的百分比: 37.3057%
    其中:
```

```
    load停顿：24    占所有RAW停顿的百分比：33.33333%
    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：14    占周期总数的百分比：7.253886%
  自陷停顿：0    占周期总数的百分比：0%
  停顿周期总数：86  占周期总数的百分比：44.55959%

分支指令：
  指令条数：13    占指令总数的百分比：12.26415%
  其中：
    分支成功：13    占分支指令数的百分比：100%
    分支失败：1    占分支指令数的百分比：7.692307%

load/store指令：
  指令条数：26    占指令总数的百分比：24.5283%
  其中：
    load：26    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：0.9433962%
```



其中主要是：

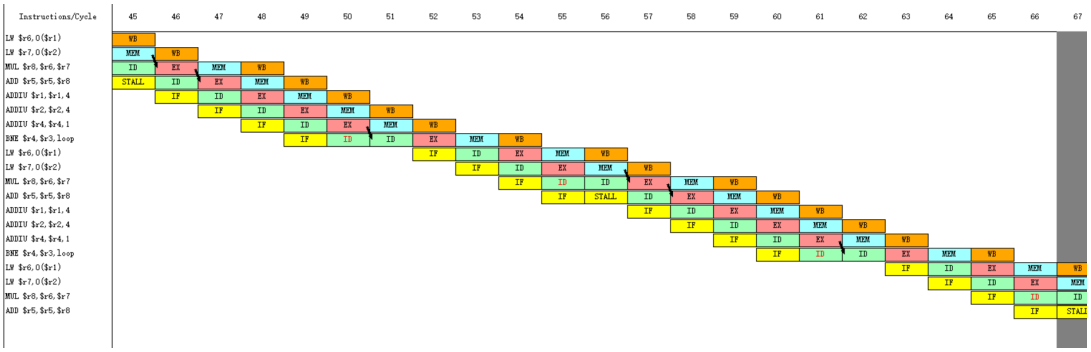
1. LW $r6, 0($r1)、LW $r7, 0($r2)和MUL $r8, $r6, $r7
2. 以及MUL $r8, $r6, $r7和ADD $r5, $r5, $r8

造成的数据冲突。

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: 145
  ID段执行了106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: 24   占周期总数的百分比: 16.55172%
  其中:
    load停顿: 12    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: 14    占周期总数的百分比: 9.655172%
  自陷停顿: 0   占周期总数的百分比: 0%
  停顿周期总数: 38  占周期总数的百分比: 26.2069%

分支指令:
  指令条数: 13    占指令总数的百分比: 12.26415%
  其中:
    分支成功: 13    占分支指令数的百分比: 100%
    分支失败: 1   占分支指令数的百分比: 7.692307%

load/store指令:
  指令条数: 26    占指令总数的百分比: 24.5283%
  其中:
    load: 26    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: 0.9433962%
```



定向功能消除了部分数据冲突，性能提升了 **193/145=1.33** 倍。

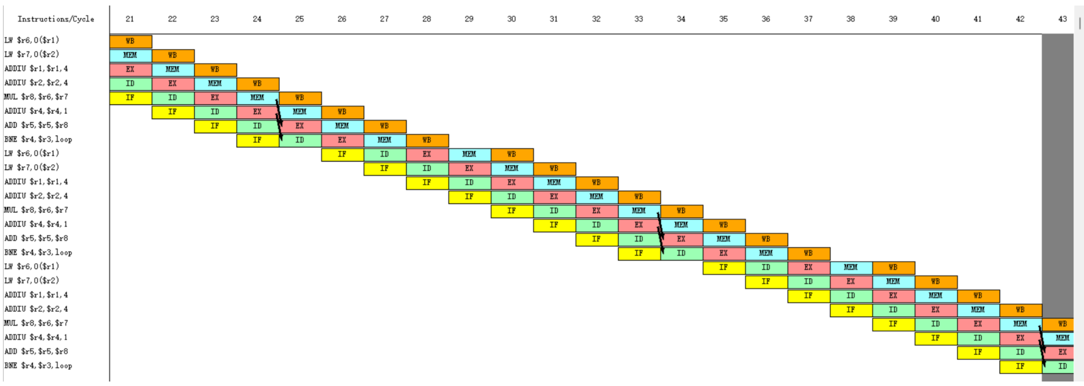
采用静态调度方法重排指令序列，在上述造成数据冲突的指令之间插入无关指令形成延迟槽，消除数据冲突，优化程序。

```
.text
main:
ADDIU  $r1,$r0,a # 取向量a的地址
ADDIU  $r2,$r0,b # 取向量b的地址
ADDIU  $r3,$r0,n # 取n的地址
BGEZAL  $r0, naive_prod # 调用naive_prod
NOP  # 无操作
TEQ $r0,$r0

naive_prod:
LW     $r3, 0($r3) # 取n的值
ADD    $r4, $r0, $r0 # 初始化 i = 0
ADD    $r5, $r0, $r0 # 初始化 ans = 0
loop:
LW     $r6, 0($r1) # 取a[i]
LW     $r7, 0($r2) # 取b[i]
ADDIU  $r1, $r1, 4 # 取a[i+1]的地址
ADDIU  $r2, $r2, 4 # 取b[i+1]的地址
MUL    $r8, $r6, $r7 # 计算a[i]*b[i]
ADDIU  $r4, $r4, 1 # i++
ADD    $r5, $r5, $r8 # ans += a[i]*b[i]
BNE    $r4, $r3, loop # i != n, 跳转到loop
JR     $r31 # 返回

.data
a:
.word 1,2,3,4,5,6,7,8,9,10,11,12
b:
.word 1,2,3,4,5,6,7,8,9,10,11,12
n:
.word 12
```

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: 121
  ID段执行了106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: 14    占周期总数的百分比: 11.57025%
  自陷停顿: 0   占周期总数的百分比: 0%
  停顿周期总数: 14  占周期总数的百分比: 11.57025%

分支指令:
  指令条数: 13    占指令总数的百分比: 12.26415%
  其中:
    分支成功: 13    占分支指令数的百分比: 100%
    分支失败: 1   占分支指令数的百分比: 7.692307%

load/store指令:
  指令条数: 26    占指令总数的百分比: 24.5283%
  其中:
    load: 26    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: 0.9433962%
```

消除了全部数据冲突，性能提升了 **145/121=1.20** 倍。


Instructions/Cycle
21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
LW $r6,0($r1)
LW $r7,0($r2)
ADDIU $r1,$r1,4
ADDIU $r2,$r2,4
MUL $r8,$r6,$r7
ADDIU $r4,$r4,1
ADD $r5,$r5,$r8
BNE $r4,$r3,loop
LW $r6,0($r1)
LW $r7,0($r2)
ADDIU $r1,$r1,4
ADDIU $r2,$r2,4
MUL $r8,$r6,$r7
ADDIU $r4,$r4,1
ADD $r5,$r5,$r8
BNE $r4,$r3,loop
LW $r6,0($r1)
LW $r7,0($r2)
ADDIU $r1,$r1,4
ADDIU $r2,$r2,4
MUL $r8,$r6,$r7
ADDIU $r4,$r4,1
ADD $r5,$r5,$r8
BNE $r4,$r3,loop