《计算机系统结构》

使用 MIPS 指令实现两个数组的点积

学院：计算机学院（国家示范性软件学院）

班级：2018211314

姓名：李志毅

学号：2018211582

实验三 使用 MIPS 指令实现求两个数组的点积

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、实验环境

指令级和和流水线操作级模拟器 MIPSsim

三、实验原理

采用静态调度方法重排指令的顺序，从而使得 RAW 等冲突所导致的空操作得以利用，减少因为 RAW 等冲突而占用的无用周期，可以通过将不想关指令前移等方式来进行静态调度优化。

四、向量点积程序代码清单及注释说明

```
.text
main:
ADDIU $r1,$r0,array1 #array1 段地址
ADDIU $r2,$r0,array2 #array2 段地址
ADDIU $r3,$r0,10 #向量长度 10
ADDIU $r4,$r0,0 #保存最终结果
```

```
loop:

LW $r5,0($r1)

LW $r6,0($r2)

MUL $r7,$r5,$r6   #相乘

ADD $r4,$r4,$r7   #r4 存放的是点积结果

ADDI $r1,$r1,4    #下一个数据

ADDI $r2,$r2,4    #下一个数据

ADDI $r3,$r3,-1   #递减

BGTZ $r3,loop     #循环判断

TEQ $r0,$r0



.data

array1: .word 1,2,3,4,5,6,7,8,9

array2: .word 1,2,3,4,5,6,7,8,9
```

五、优化后的代码清单

```
    .text

    main:

    ADDIU $r1,$r0,array1 #array1 段地址

    ADDIU $r2,$r0,array2 #array2 段地址

    ADDIU $r3,$r0,10     #向量长度 10

    ADDIU $r4,$r0,0      #保存最终结果

    loop:
```

```
LW $r5,0($r1)

LW $r6,0($r2)

ADDI $r1,$r1,4 #优化点 1

ADDI $r2,$r2,4 #优化点 2

MUL $r7,$r5,$r6 #相乘

ADDI $r3,$r3,-1 #优化点 3

ADD $r4,$r4,$r7   #r4 存放的是点积结果

BGTZ $r3,loop #循环判断

TEQ $r0,$r0



.data

array1: .word 1,2,3,4,5,6,7,8,9

array2: .word 1,2,3,4,5,6,7,8,9
```

六、实验步骤

1. 自动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：

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

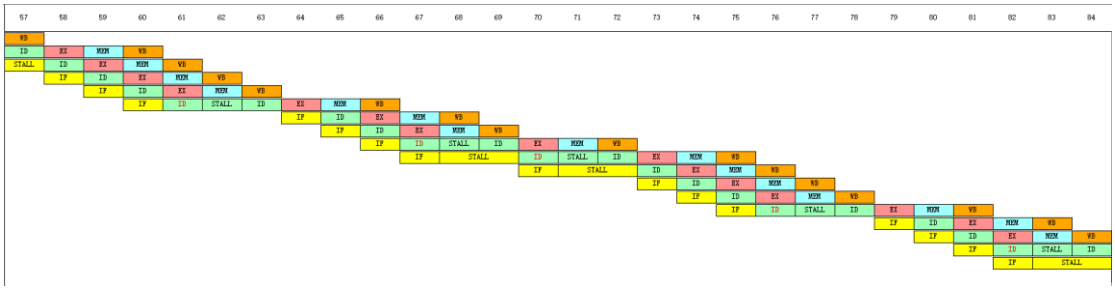
两个向量元素使用数组进行数组存储，要求向量的维度不得小于 10

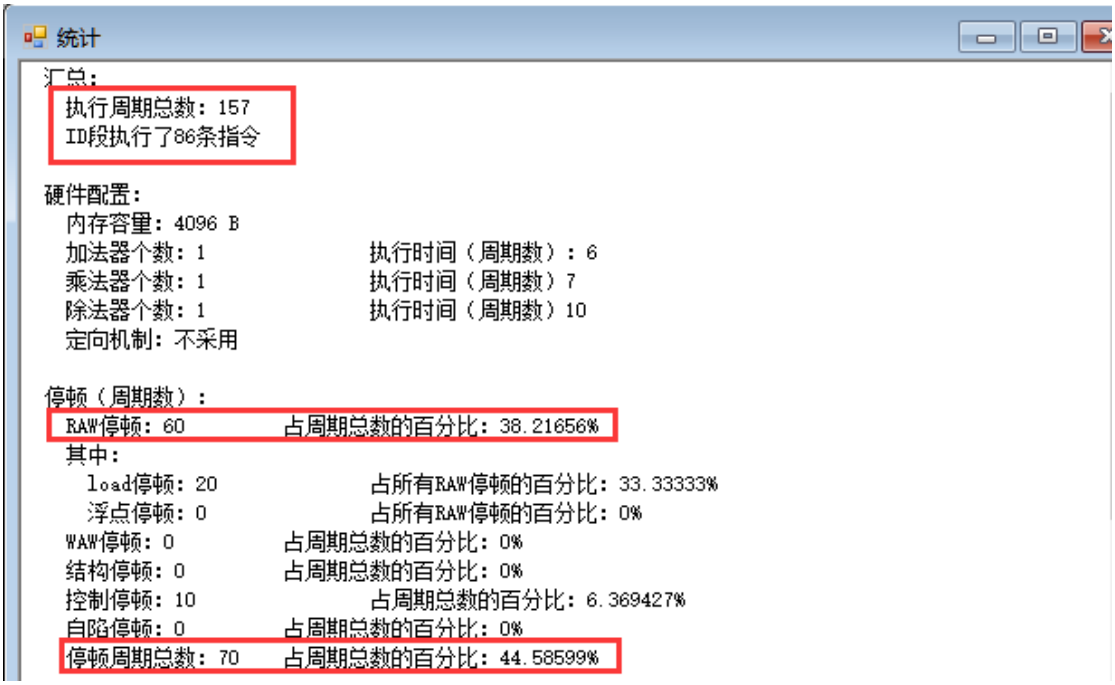
源代码见附录

2. 启动 MIPSsim

3. 载入自己编写的程序，观察流水线输出结果

流水线的部分执行情况：

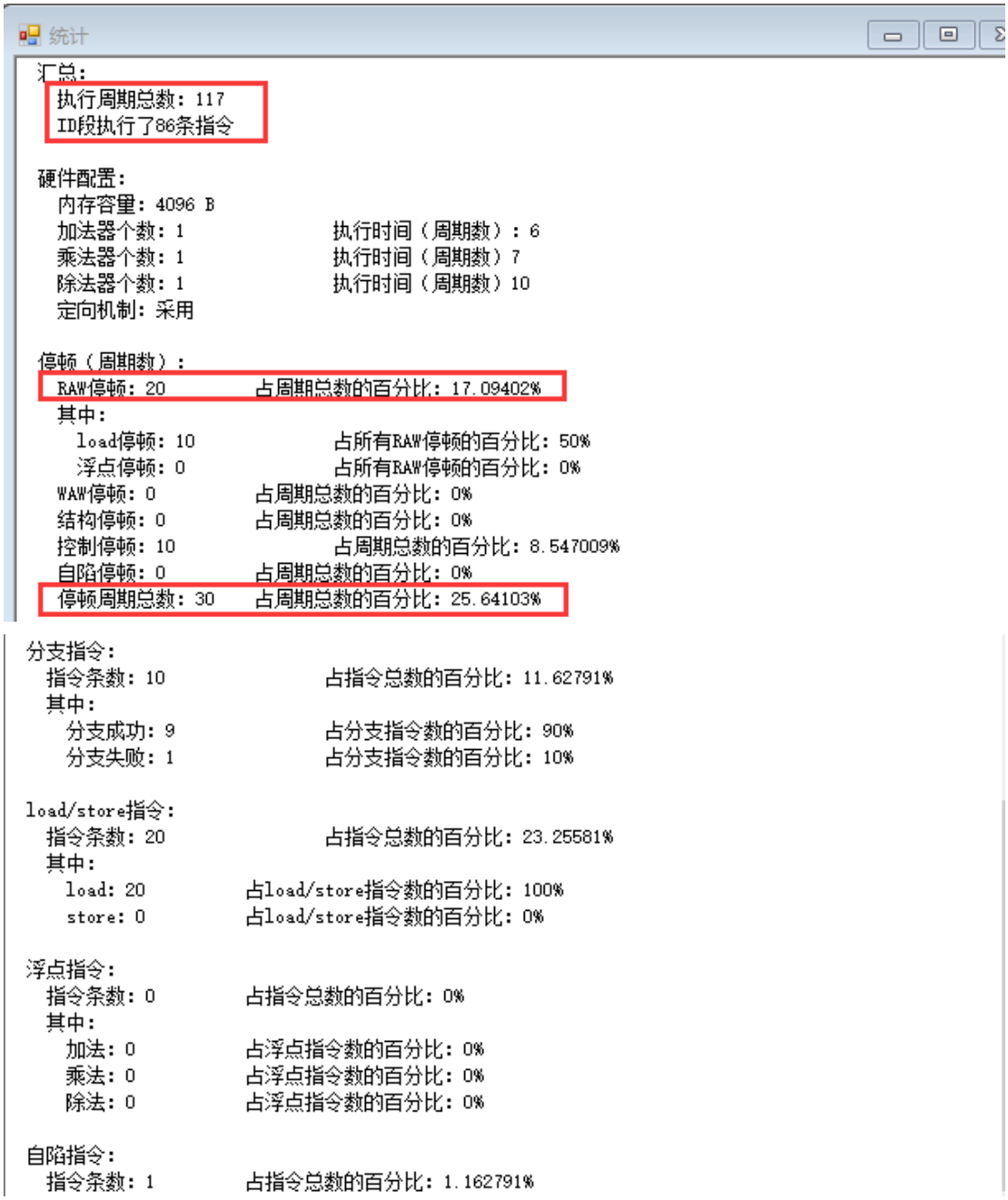




程序计算的是(1,2,3,4,5,6,7,8,9,10)和(1,2,3,4,5,6,7,8,9,10)的结果

程序执行 **157** 个周期，其中 RAW 冲突占用 **60** 个周期，占比 **38.21656%**

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。



可以看到，采用定向技术后，执行总周期数变为 **117**，其中 RAW 停顿 **20** 个周期，占比为 **17.09405%**，执行的效率变为原来的 157/117=**1.34** 倍

5. 采用静态调度方法重排指令序列，减少相关，优化程序

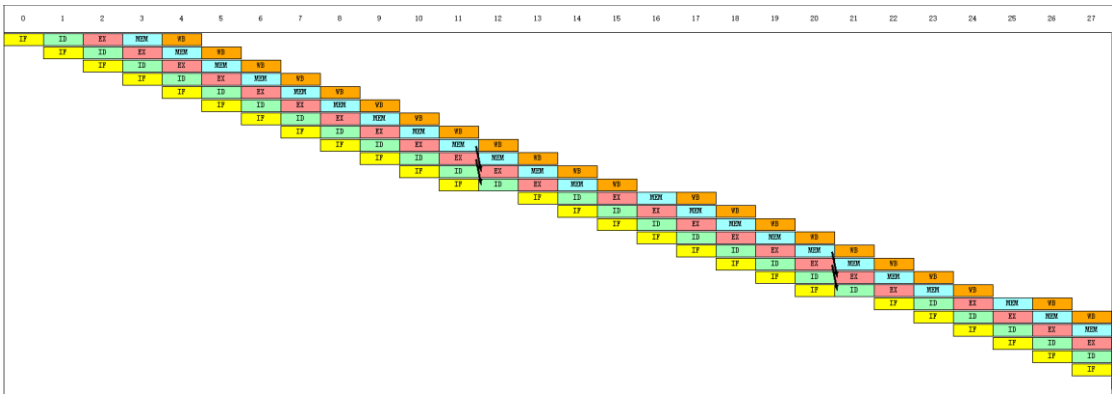
```
loop            0x8C250000      LW $r5,0($r1)
0x00000014      0x8C460000      LW $r6,0($r2)
0x00000018      0x70A63802      MUL $r7,$r5,$r6
0x0000001C      0x00872020      ADD $r4,$r4,$r7
0x00000020      0x20210004      ADDI $r1,$r1,4
```

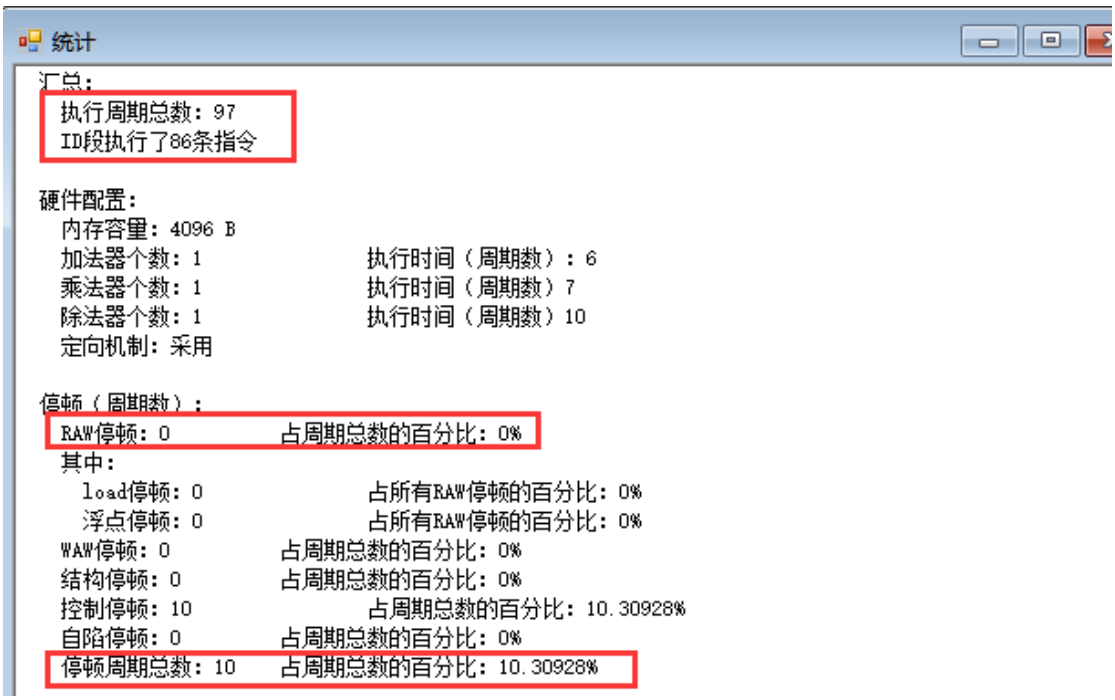
可以看到，LW $r5,0($r1)指令和 LW $r6,0($r2)指令和 MUL $r7,$r5,$r6 存在

读后写相关，MUL $r7,$r5,$r6 指令和 ADD $r4,$r4,$r7 存在读后写相关，可以采用静态调度的方法，将三个 ADDI 指令穿插在其中

```
loop            0x8C250000      LW $r5,0($r1)
0x00000014      0x8C460000      LW $r6,0($r2)
0x00000018      0x20210004      ADDI $r1,$r1,4
0x0000001C      0x20420004      ADDI $r2,$r2,4
0x00000020      0x70A63802      MUL $r7,$r5,$r6
0x00000024      0x2063FFFF      ADDI $r3,$r3,-1
0x00000028      0x00872020      ADD $r4,$r4,$r7
0x0000002C      0x1C60FFF8      BGTZ $r3,loop
0x00000030      0x00000034      TEQ $r0,$r0
```

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。





可以看到采用静态调度的方法优化程序，并使用定向功能后，执行总周期数为 **97**，RAW 停顿为 **0**，效率变为原来的 117/97=**1.206** 倍。

七、实验问题与心得

本次实验使用指令级和流水线操作级模拟器 MIPSsim 分析了数组点积程序优化的过程，加深了我对于计算机流水线基本概念的理解，理解了 MIPS 结构是如何使用 5 段流水线来实现的，理解了各段的功能和基本操作，加深了我对数据冲突和结构冲突的理解，以及采用定向技术解决数据冲突带来的好处和性能的提升，进一步掌握了解决数据冲突的方法，掌握了如何应用定向技术来减少数据冲突引起的停顿，增强汇编语言编程能力，了解了对代码进行优化的方法。