

**计算机系统结构**



**使用MIPS指令实现冒泡排序法**

**姓 名 鄭毓恒**

**学 院 计算机学院**

**专 业 计算机科学与技术**

**班 级 2020211302**

**学 号 2020211262**

**任课教师 邝坚**

**2023年 4 月**

**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。

**实验原理**

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，要求数组长度不得小于10。在使用和不使用的定向功能的情况下，对比程序的执行结果和效率。

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。

然后，使用静态调度方法重排指令序列，减少相关，优化程序。在本次实验，静态调度方法减少数据冲突发生的次数，减少因为数据冲突而停顿的时钟周期数。在使用和不使用的定向功能的情况下，和未优化程序比较执行结果和效率。

**向量点积程序代码清单及注释说明**

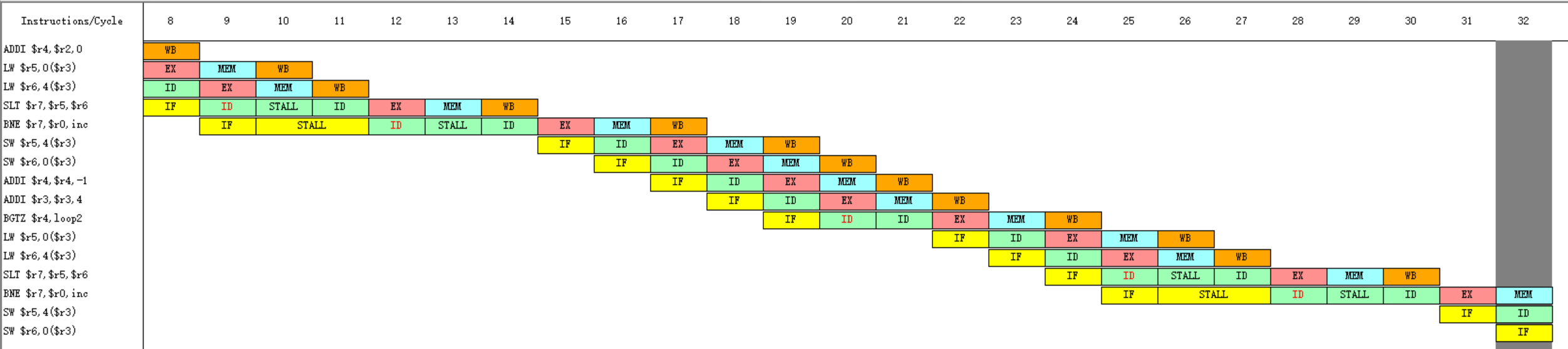
|  |
| --- |
| .text  main:  ADDIU $r1,$r0,array #数组地址  ADDI $r2,$r0,15 #外循环变量  loop1:  ADD $r3,$r0,$r1 #数组第一个元素  ADDI $r4,$r2,0 #内循环变量  loop2:  LW $r5,0($r3) #array[i]  LW $r6,4($r3) #array[i+1]  SLT $r7,$r5,$r6 #array[i]<array[i+1]？  BNEZ $r7,inc #如果array[i+1]>=array[i]，则跳转到inc  SW $r5,4($r3) #如果array[i+1]<array[i]，交换  SW $r6,0($r3)  inc:  ADDI $r4,$r4,-1 #内循环次数-1  ADDI $r3,$r3,4 #下一个元素开始内循环  BGTZ $r4,loop2  ADDI $r2,$r2,-1 #外循环次数-1  BGTZ $r2,loop1  TEQ $r0,$r0  .data  array:  .word 18,17,2,3,5,35,5,2,78,4,33,44,22,100,16,2 |

**优化后的程序代码清单**

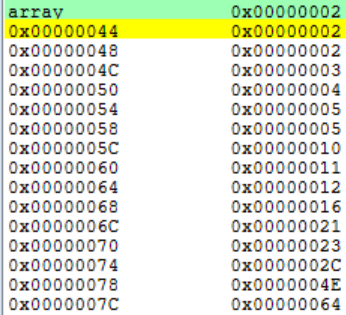
|  |
| --- |
| .text  main:  ADDIU $r1,$r0,array #数组地址  ADDI $r2,$r0,15 #外循环变量  loop1:  ADD $r3,$r0,$r1 #数组第一个元素  ADDI $r4,$r2,0 #内循环变量  loop2:  LW $r5,0($r3) #array[i]  LW $r6,4($r3) #array[i+1]  ADDI $r4,$r4,-1 #内循环次数-1  SLT $r7,$r5,$r6 #array[i]<array[i+1]？  BNEZ $r7,inc #如果array[i+1]>=array[i]，则跳转到inc  SW $r5,4($r3) #如果array[i+1]<array[i]，交换  SW $r6,0($r3)  inc:  ADDI $r3,$r3,4 #下一个元素开始内循环  BGTZ $r4,loop2  ADDI $r2,$r2,-1 #外循环次数-1  BGTZ $r2,loop1  TEQ $r0,$r0  .data  array:  .word 18,17,2,3,5,35,5,2,78,4,33,44,22,100,16,2 |

**未优化代码和优化代码性能分析比较结果**

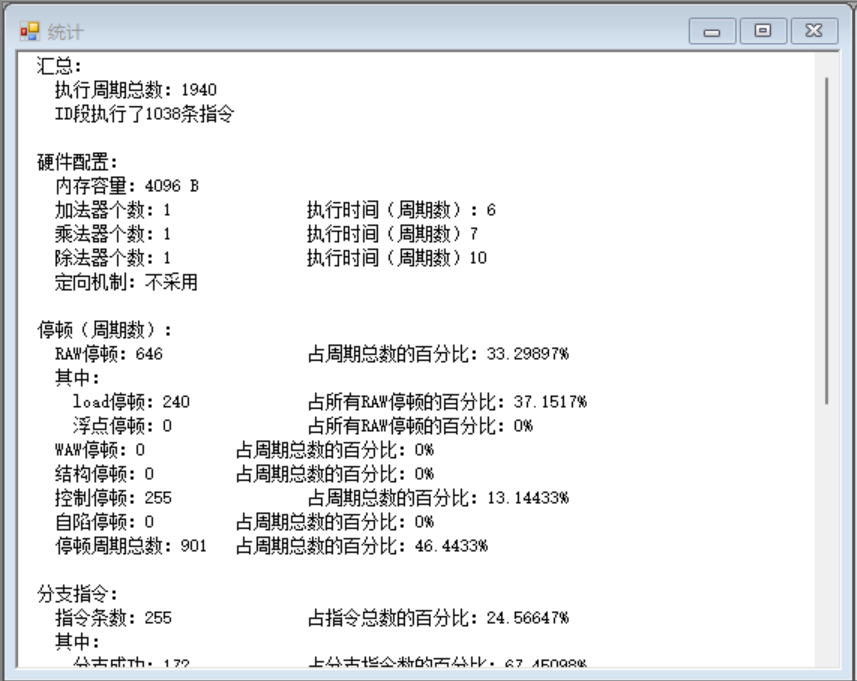
运行自己编写的程序，流水线的执行情况如下。



查看数据段里的array数组，每个元素从小到大排序，结果正确。



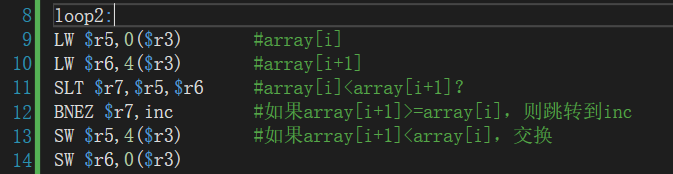
程序执行周期总数为1940，RAW停顿周期数为646，占周期总数的33.29897%。



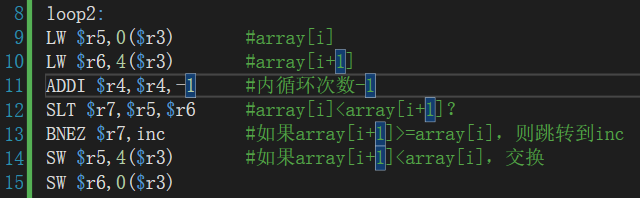
使用定向功能再次执行代码。程序执行周期总数降至1493，RAW停顿周期数为255，占周期总数的17.07971%。与没有使用定向功能时相比，效率提升了1.30倍。



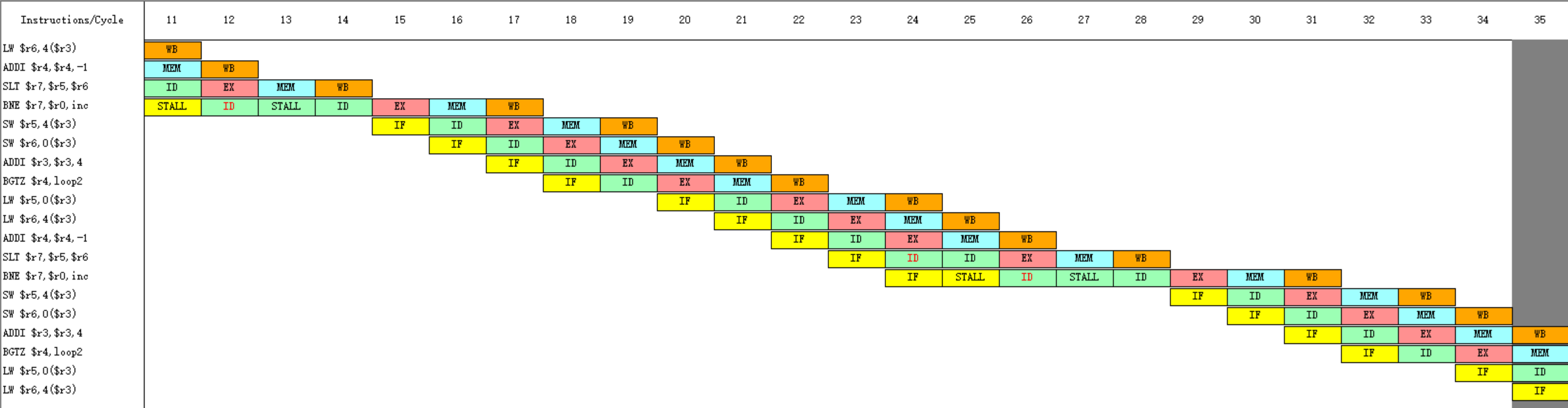
代码多处存在RAW相关指令，应从执行次数最多的内循环开始优化。第11行指令和第9、10行指令有RAW相关。第12行指令和第11行指令有RAW相关。



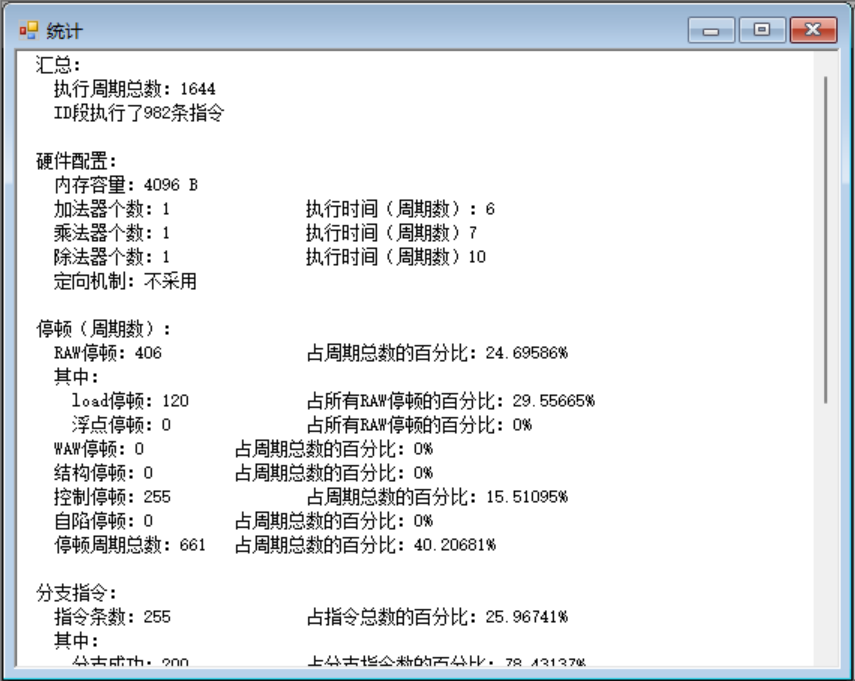
采用静态调度的方法，将其他指令穿插其中。由于能用于穿插的指令只有一条，只能放在第11行，尽可能减少RAW冲突。



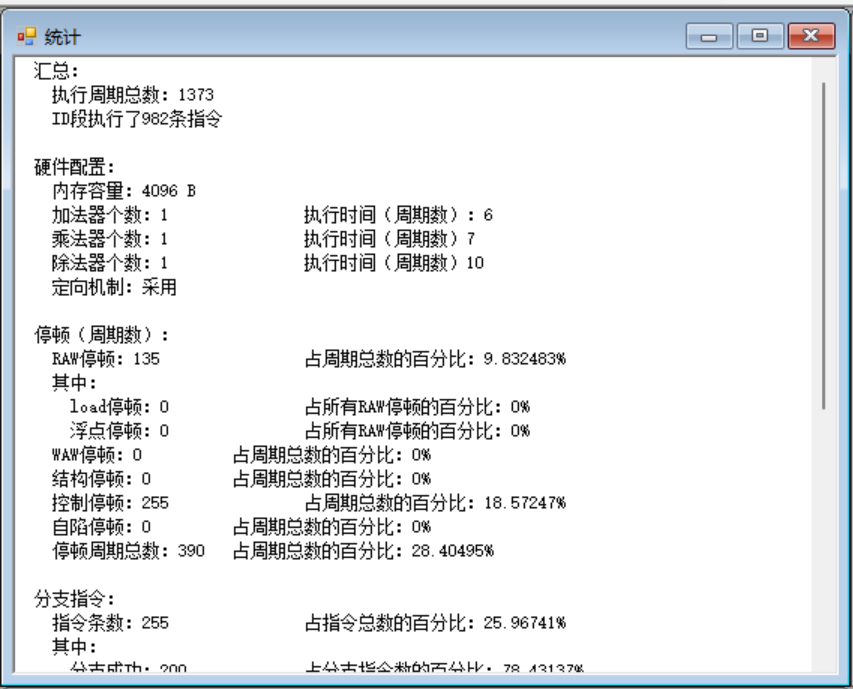
运行优化程序，流水线的执行情况如下。



程序执行周期总数为1644，RAW停顿周期数为406，占周期总数的24.69586%。比没有使用定向功能的未优化代码，效率提升了1.18倍。



使用定向功能再次执行代码。程序执行周期总数降至1373，RAW停顿周期数为135。比使用了定向功能的未优化代码，效率提升了1.09倍。



**实验总结**

通过编写汇编程序实现冒泡排序，增强了汇编语言编程能力，学习了如何使用静态调度优化代码。通过对比未优化和优化代码的性能，加深了我对数据冲突的理解，了解到定向功能和静态调度解决数据冲突的原理以及对提升CPU性能的帮助。