

**计算机系统结构**



**指令调度与延迟分支**

**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**

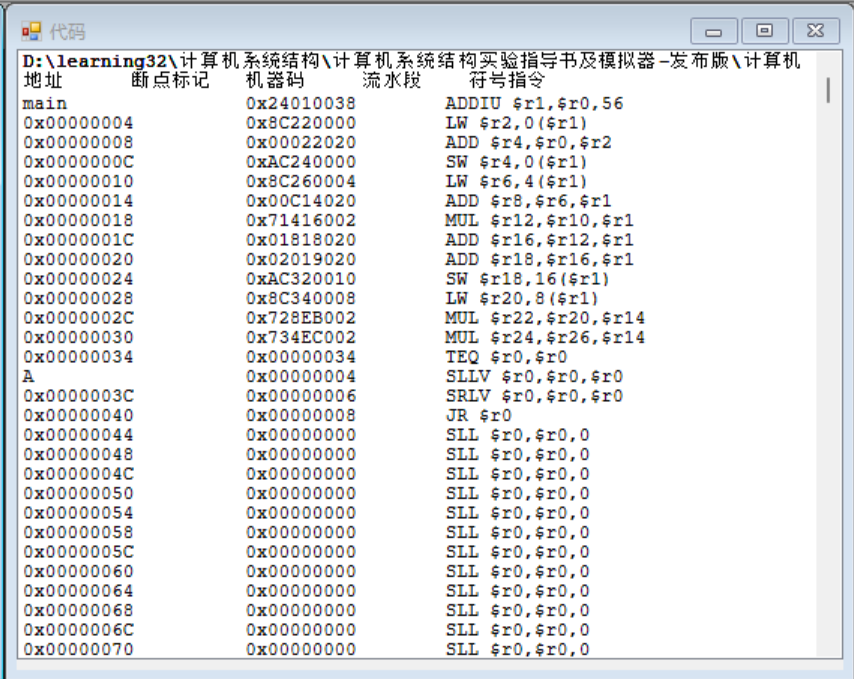
1. 加深对指令调度技术的理解。
2. 加深对延迟分支技术的理解。
3. 熟练掌握用指令调度技术解决流水线中的数据冲突的方法。
4. 进一步理解指令调度技术对 CPU 性能的改进。
5. 进一步理解延迟分支技术对 CPU 性能的改进。

**实验环境**

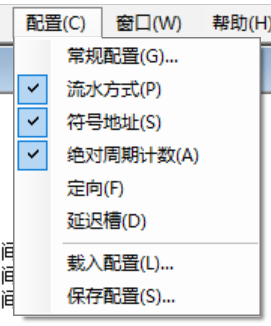
指令级和流水线操作级模拟器MIPSsim.

**实验内容和步骤**

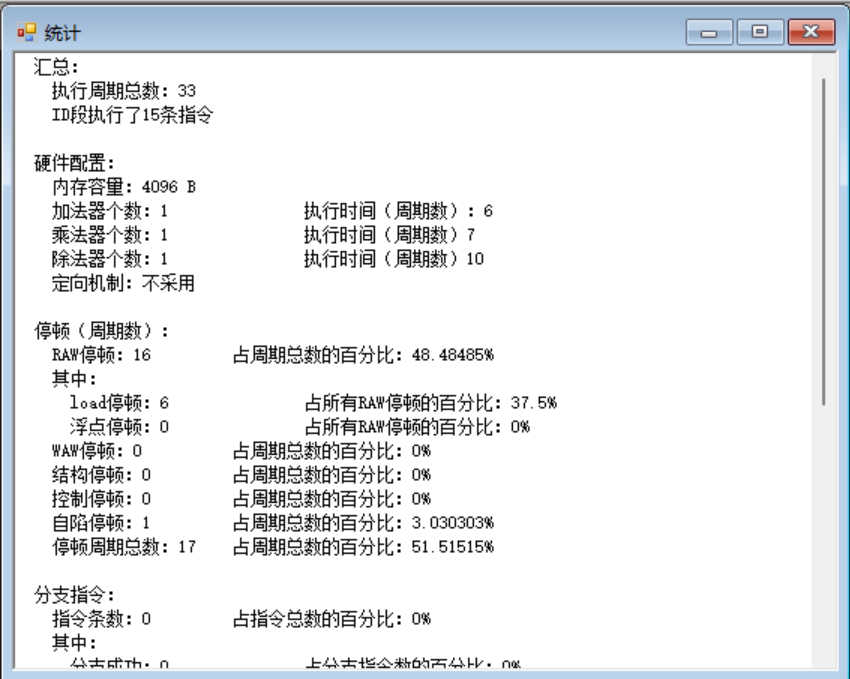
载入程序schedule.s



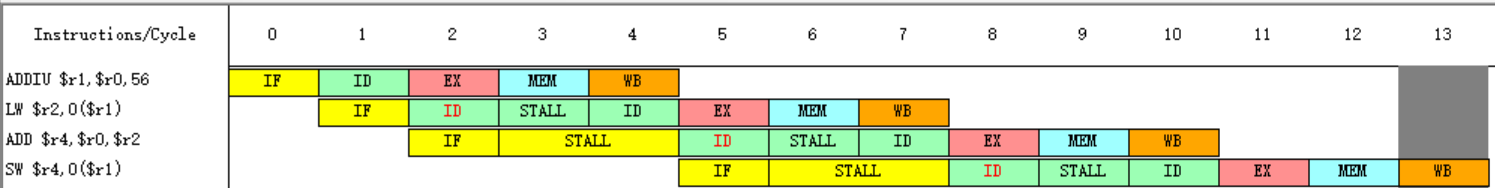
开启流水方式、关闭定向。



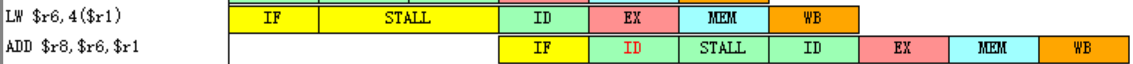
程序执行周期总数是33，停顿周期总数是17，其中有16个RAW停顿和1个自陷停顿。



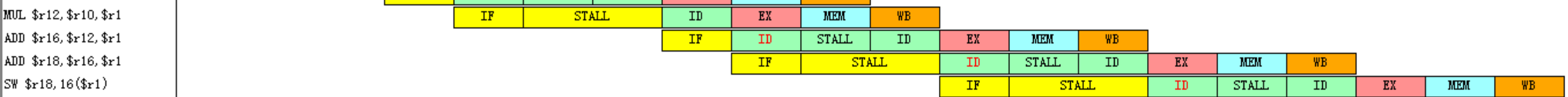
通过查看流水线时钟周期图，找到了以下存在冲突的指令。



|  |
| --- |
| ADDIU $r1,$r0,A  LW $r2,0($r1)  ADD $r4,$r0,$r2  SW $r4,0($r1) |



|  |
| --- |
| LW $r6,4($r1)  ADD $r8,$r6,$r1 |



|  |
| --- |
| MUL $r12,$r10,$r1  ADD $r16,$r12,$r1  ADD $r18,$r16,$r1  SW $r18,16($r1) |

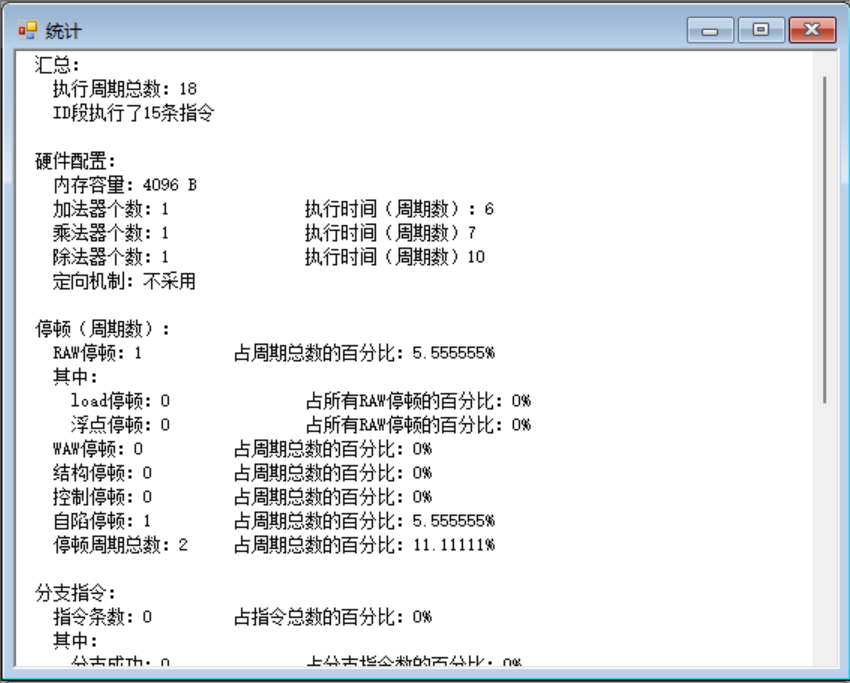


|  |
| --- |
| LW $r20,8($r1)  MUL $r22,$r20,$r14 |

通过静态调度，消除冲突。优化后的代码after-schedule.s内容如下。

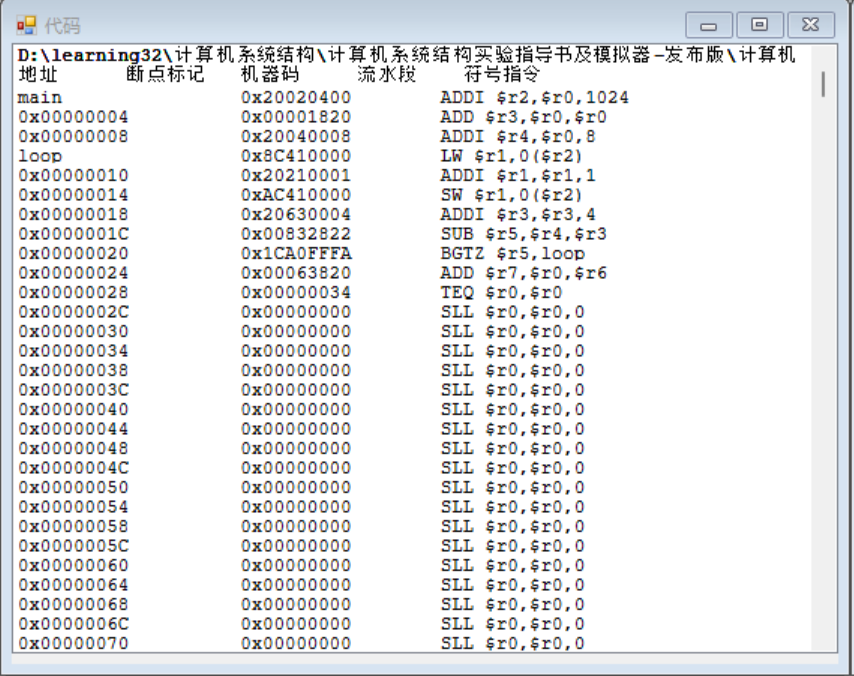
|  |
| --- |
| .text  main:  ADDIU $r1,$r0,A  MUL $r24,$r26,$r14  MUL $r12,$r10,$r1  LW $r2,0($r1)  LW $r6,4($r1)  ADD $r16,$r12,$r1  ADD $r4,$r0,$r2  LW $r20,8($r1)  ADD $r18,$r16,$r1  SW $r4,0($r1)  ADD $r8,$r6,$r1  MUL $r22,$r20,$r14  SW $r18,16($r1)  TEQ $r0,$r0  .data  A:  .word 4,6,8 |

程序执行周期总数是18，停顿周期总数是2，其中有1个RAW停顿和1个自陷停顿。

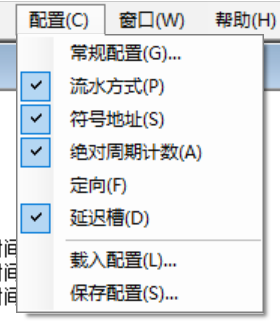


程序执行周期从33降至18，停顿周期从17降至2。可见经过静态调度，对存在RAW冲突的代码进行重排后，因RAW冲突而产生停顿的数目大大减少，显著提升了CPU性能。

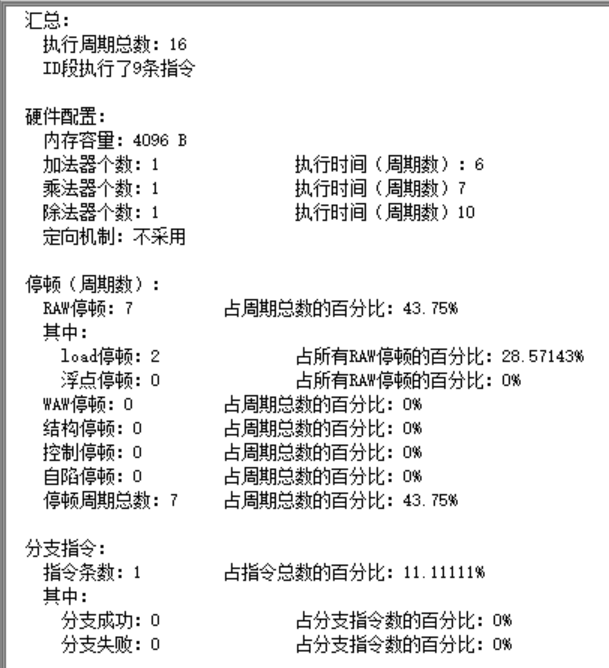
载入branch.s样例程序。



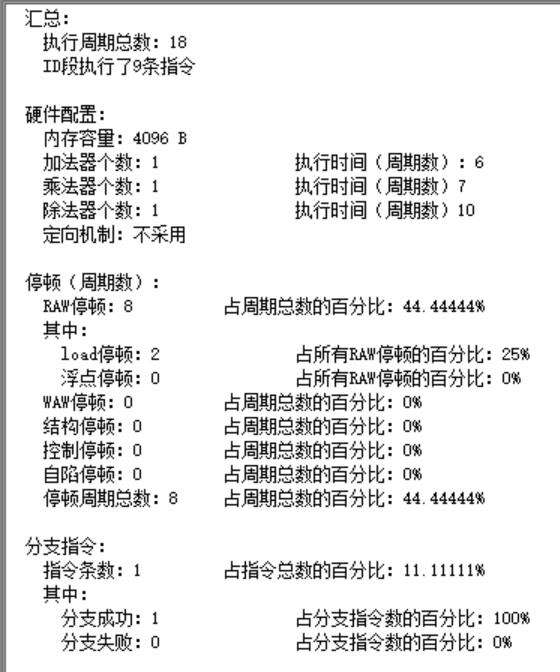
关闭延迟分支。



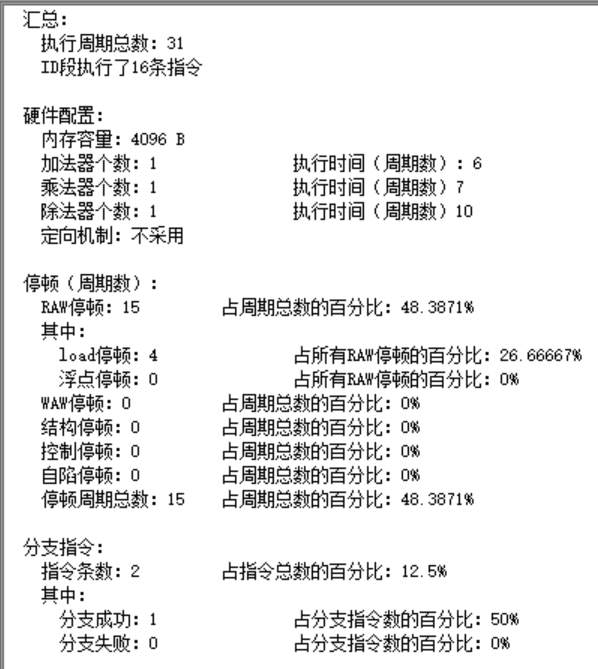
第16个周期发生第一个分支。



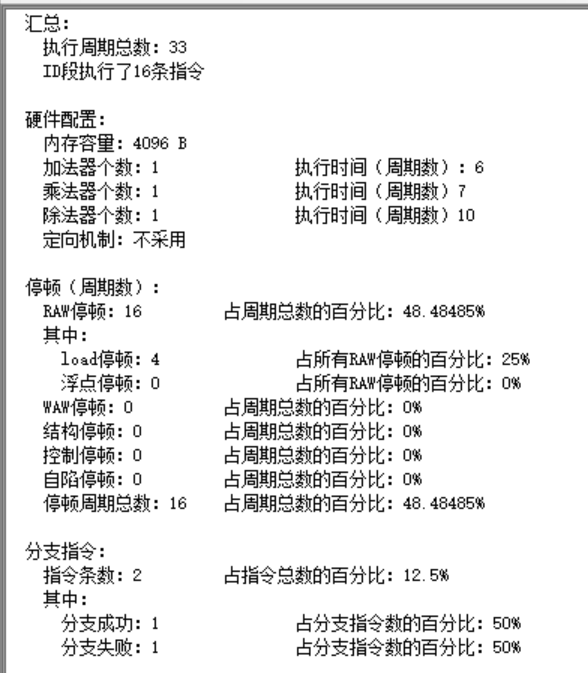
在第18个周期可知第一次分支成功。



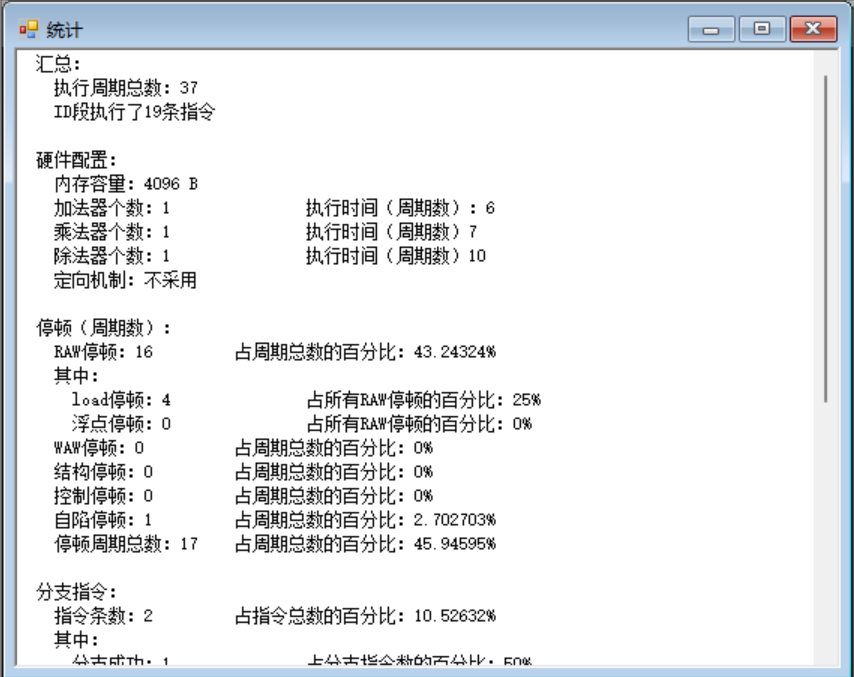
第31个周期发生第二次分支。

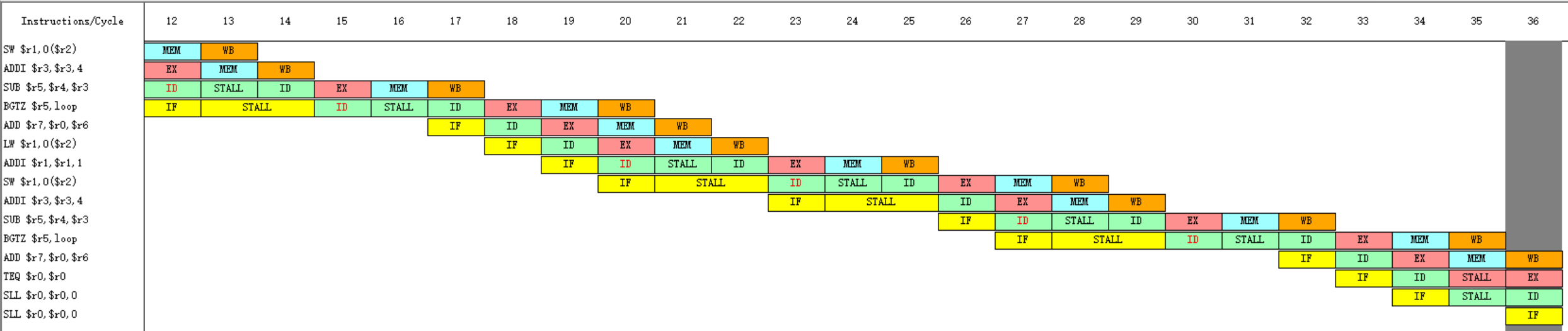


在第33个周期可知第二次分支失败。



程序周期总数37，RAW停顿周期有16个，占周期总数的43.24324%。总共发生两次分支，第一次成功和第二次失败。

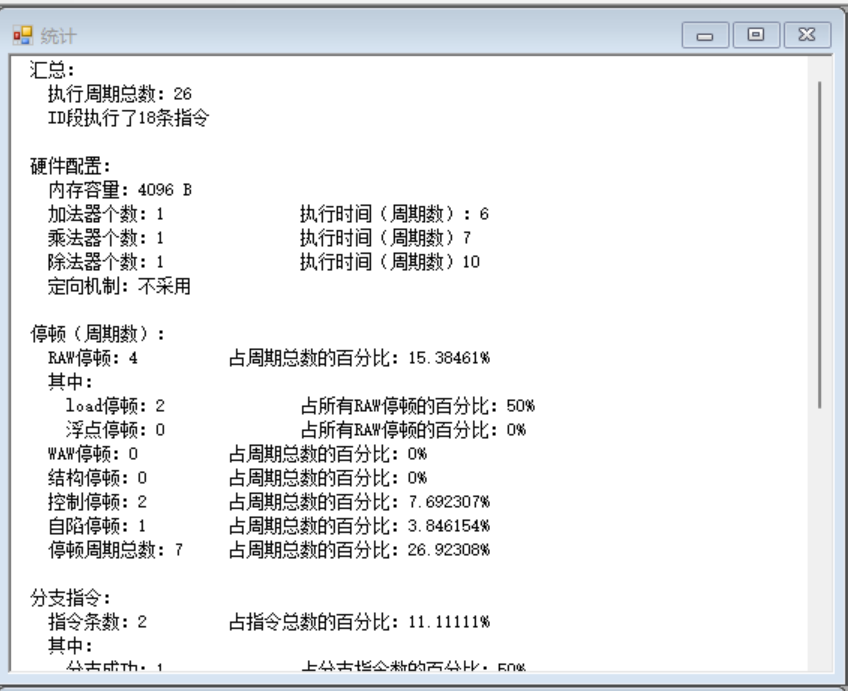


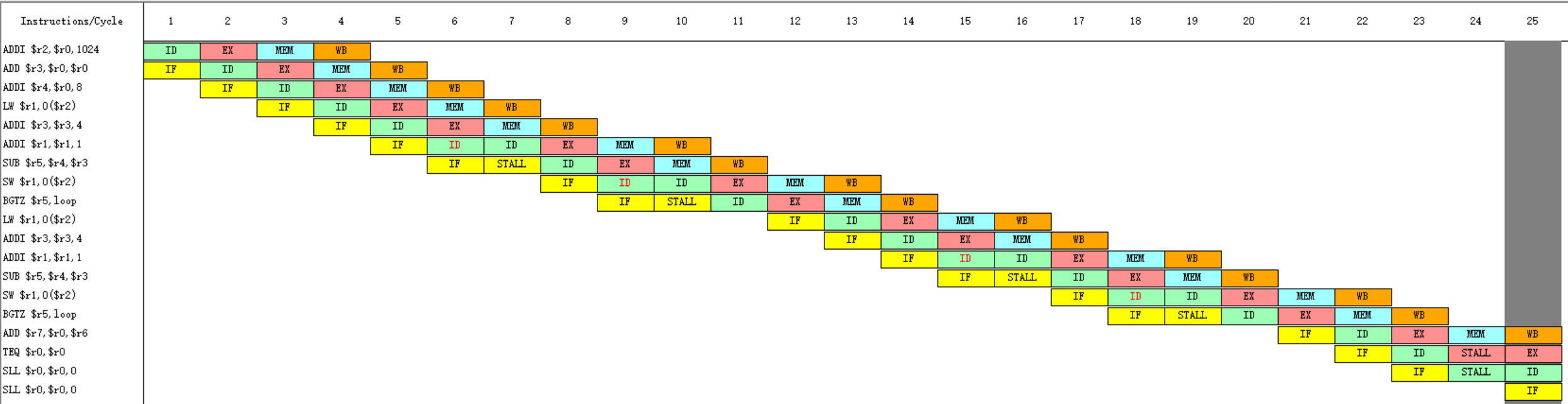


假设延迟槽为一个，对branch.s进行指令调度，调度结果为delayed-branch.s。

|  |
| --- |
| .text  main:  ADDI $r2,$r0,1024  ADD $r3,$r0,$r0  ADDI $r4,$r0,8  loop:  LW $r1,0($r2)  ADDI $r3,$r3,4  ADDI $r1,$r1,1  SUB $r5,$r4,$r3  SW $r1,0($r2)  BGTZ $r5,loop  ADD $r7,$r0,$r6  TEQ $r0,$r0 |

打开延迟分支功能运行程序。程序执行周期总数是26，RAW停顿周期数是4，占周期总数的15.36461%。





采用了延迟分支后，程序执行周期总数从37降至26，效率提高了1.42倍。在使用延迟槽后，指令在运行到跳转指令时，不会出现延迟等待，提高CPU的性能。

**实验总结**

通过运行并分析样例程序的结果，然后进行指令调度和延迟分支的优化，理解两种方法对CPU性能的提升。通过对样例程序的指令调度，更熟悉掌握用指令调度技术解决流水线中的数据冲突的方法。