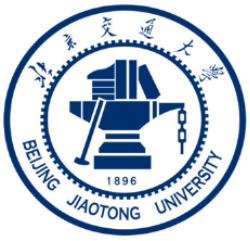



北京交通大學

BEIJING JIAOTONG UNIVERSITY

# 《计算机体系结构》

# 实验报告

| | |
|---|---|
| 实验名称: | 实验 5_分支预测实验报告 |
| 学　号: | |
| 姓　名: | |
| 学　院: | 计算机与信息技术学院 |
| 日　期: | 2022 年 11 月 22 日 |

# 目录

# 1. 理解转移延迟槽(branch slot)的作用

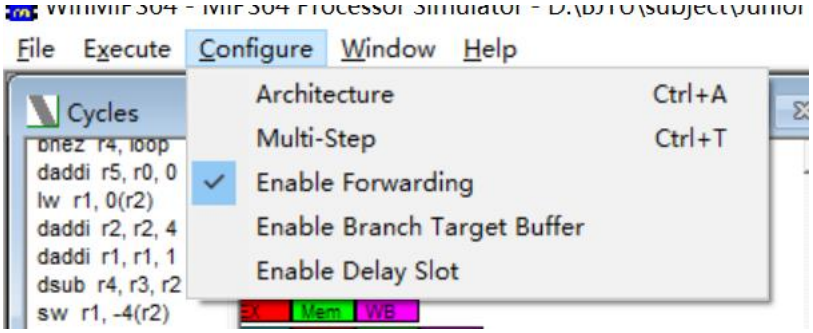
① 将 WinMIPS64 设置成允许 Enable Forwarding 和不允许 Enable Branch Delay Slot，运行 incre1.s，记录运行结果。运行结果包括 Data 窗口和 Statistics 窗口的结果。

② 考虑带一个延迟槽的情况下，对 incre1.s 进行调度。请仔细考虑，教材上给出的三种策略中，哪种策略适合这个程序。将 WinMIPS64 设置成允许 Enable Forwarding 和允许 Enable Branch Delay Slot，运行你调度后的程序并记录运行结果。运行结果包括 Data 窗口和 Statistics 窗口的结果。

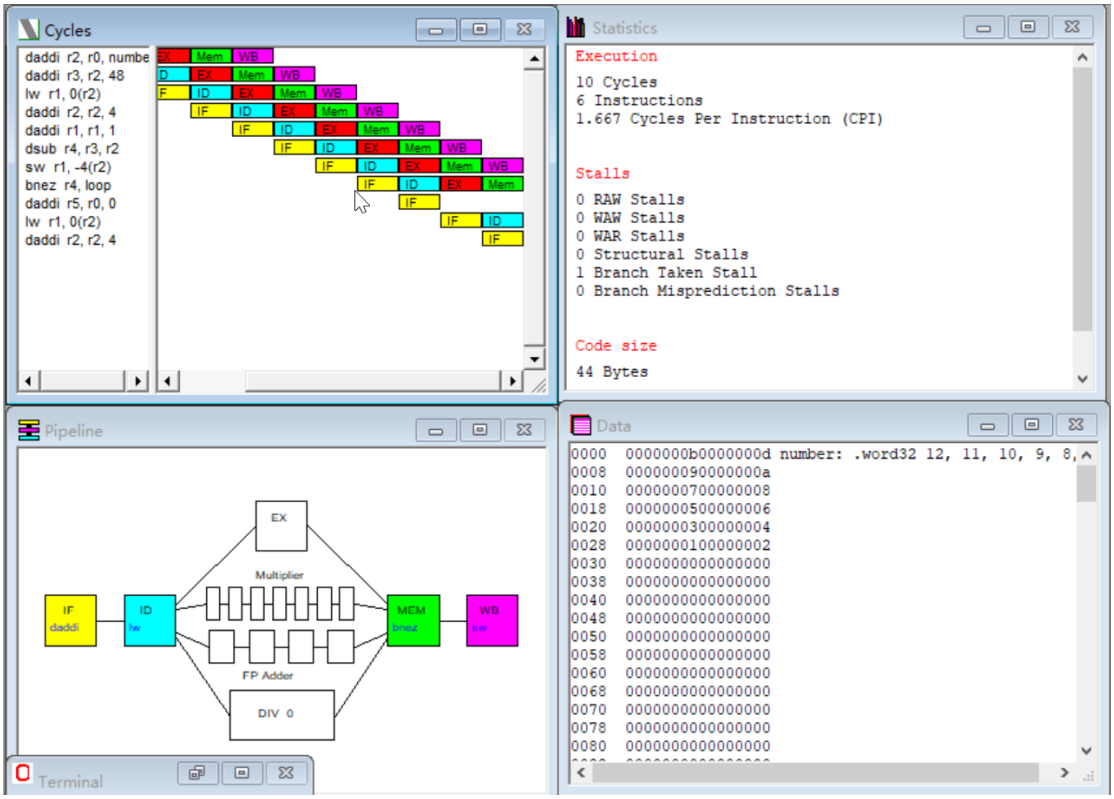
③ 比较①和②两种情况下的运行结果。首先要确定两种情况下 Data 窗口中的运行结果是一致的。其次，比较两种情况下 Statistics 窗口的性能结果，分析一下为什么调度后运行周期减少了。

---

① 把程序设置为

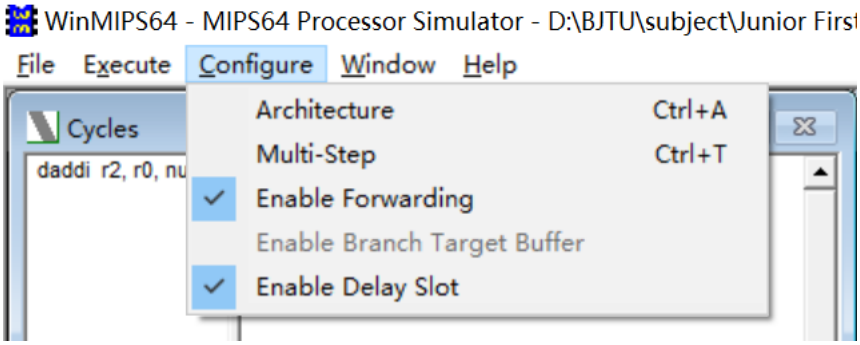



图表 1-1 允许 Enable Forwarding 和不允许 Enable Branch Delay Slot 进行运行。



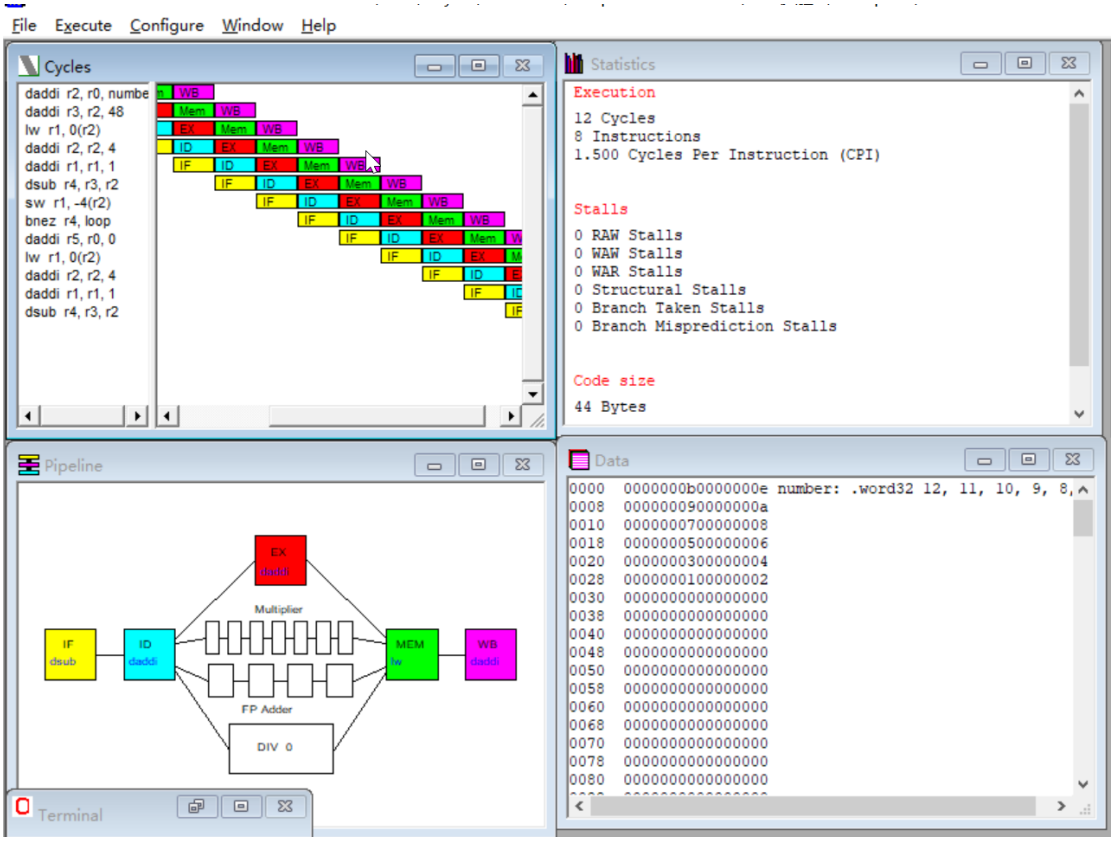

图表 1-2 运行结果

② 再按实验要求如下：



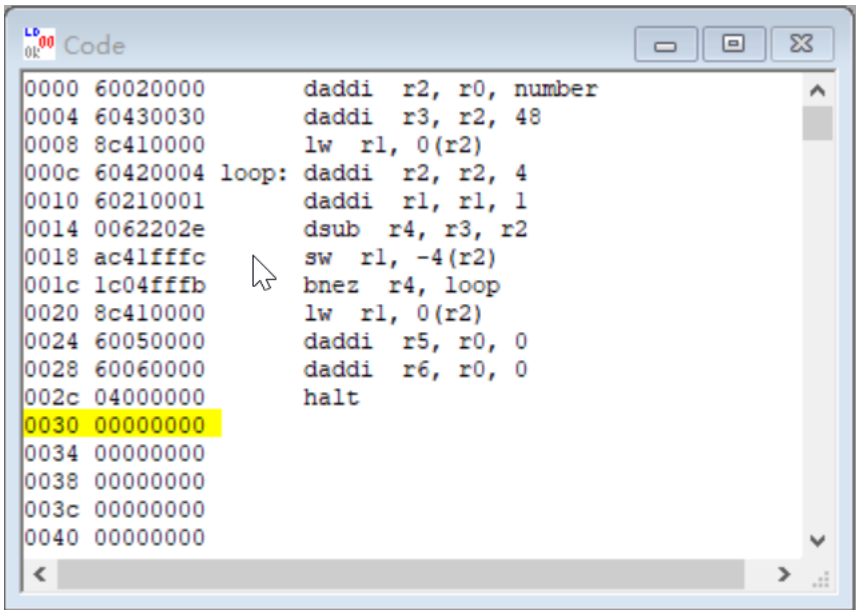

图表 1-3 允许 Enable Forwarding 和允许 Enable Branch Delay Slot

运行结果：




图表 1-4 有 delay slot 运行结果

在课本给出的三种策略中，明显是“拷贝自对象”的命令放到 delay slot 中比较好，最适合。



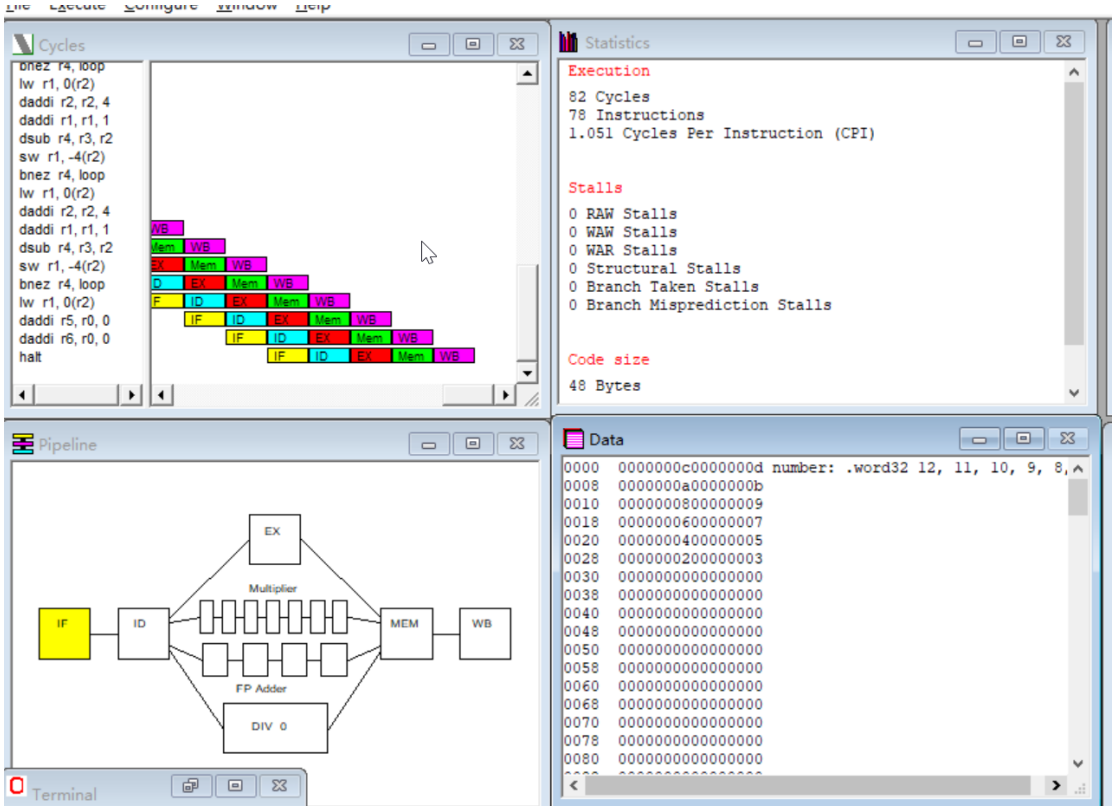

图表 1-5 **“拷贝自对象”的** delay slot

这样就可以避免 delay slot 中的指令被浪费。同时，“来自以前”的方法只能调整 sw 的位置，但是这样又会引入 raw；“转移不成功”的方法又减少的指令周期数有限，只有在最后一次跳出循环时，才能做有效的减少。**所以选择了“拷贝自对象”。改变了 lw 的位置。**

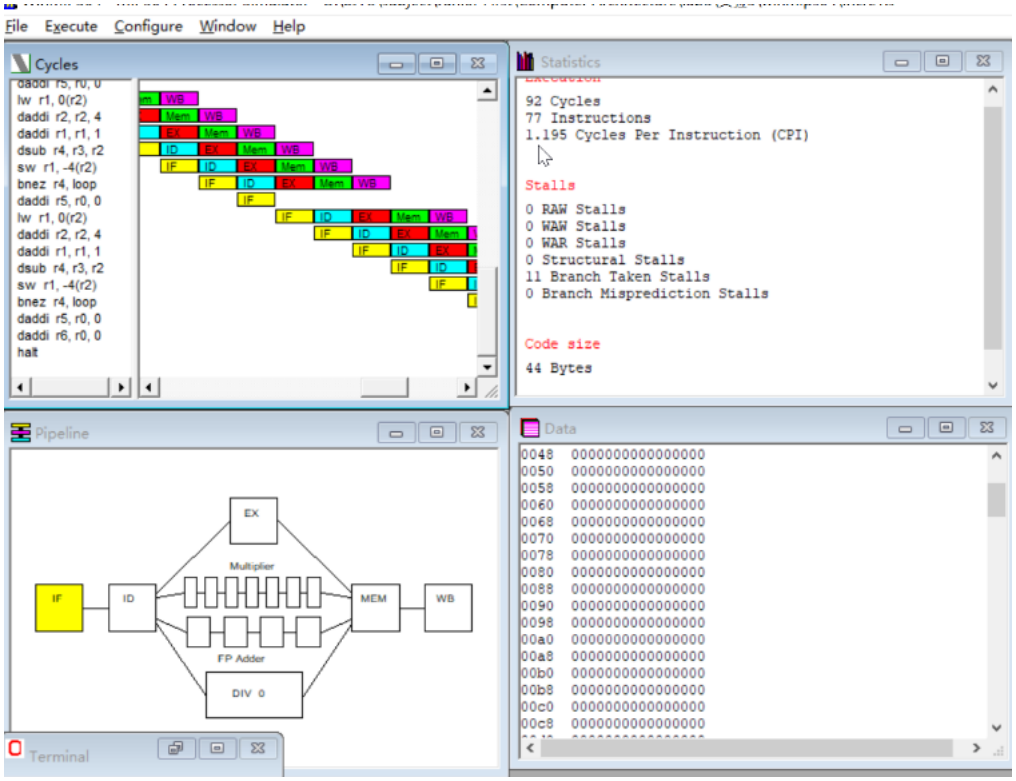
**③ 运行周期减少的解释：**

实验 运行对比见下图表 1-6 调用之后开启 delay slot 的结果和图表 1-7 调度之前没有使用 delay slot 的结果。可以看出使用调度之后，整个程序运行的周期数为 **82**，之前的为 **92**，大大减少了。

我们通过“拷贝自对象”的调度策略，把 delay slot 中一定要执行的命令，变成了 lw 命令，而在 loop 中的 lw 提前到 loop 上边。这样就可以有效的减少周期数。




图表 1-6 调用之后开启 delay slot 的结果




图表 1-7 调度之前没有使用 delay slot 的结果

# 2. 理解分支目标缓冲器(BTB)的作用

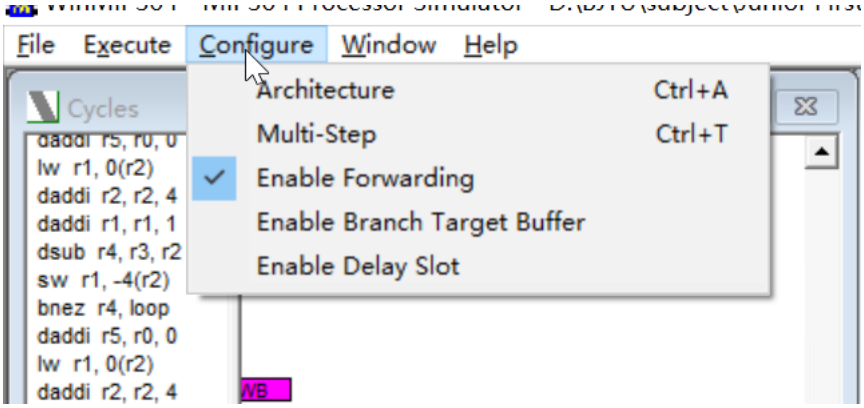
① 将 WinMIPS64 设置成允许 Enable forwarding 和不允许 Enable Branch Target buffer，运行 incre1.s 程序，记录 Statistics 窗口的统计结果。

② 将 WinMIPS64 设置成允许 Enable forwarding 和允许 Enable Branch Target buffer，运行 incre1.s 程序，记录 Statistics 窗口的统计结果。

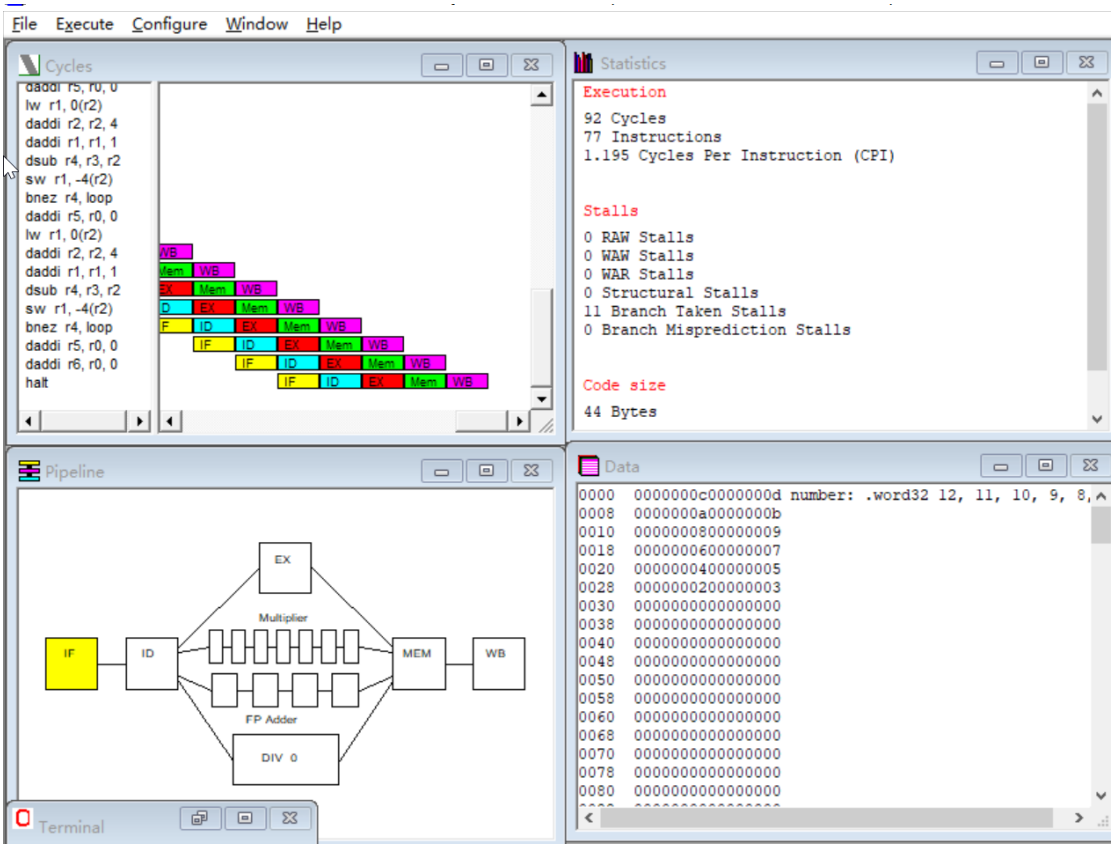
③ 比较①和②两种情况下的运行结果，解释为什么允许 BTB 就减少了运行周期数。为便于解释，建议按教材中图 3.49 的 BTB 表的形式，给出在程序运行过程中 BTB 的变化情况。

---

① 按照要求做如下设置：

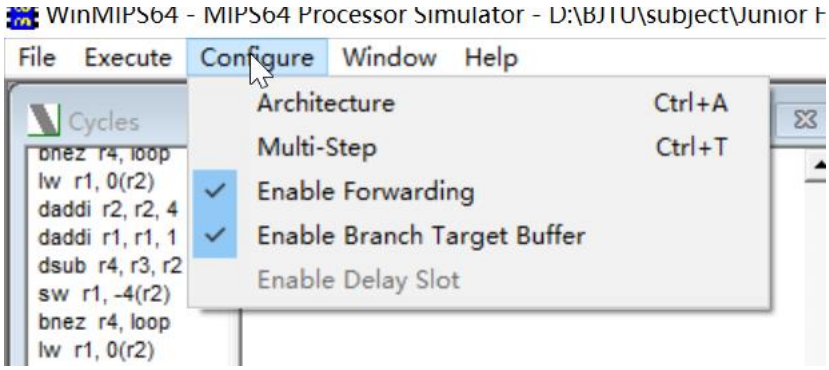



图表 2-1 允许 Enable forwarding 和不允许 Enable Branch Target buffer

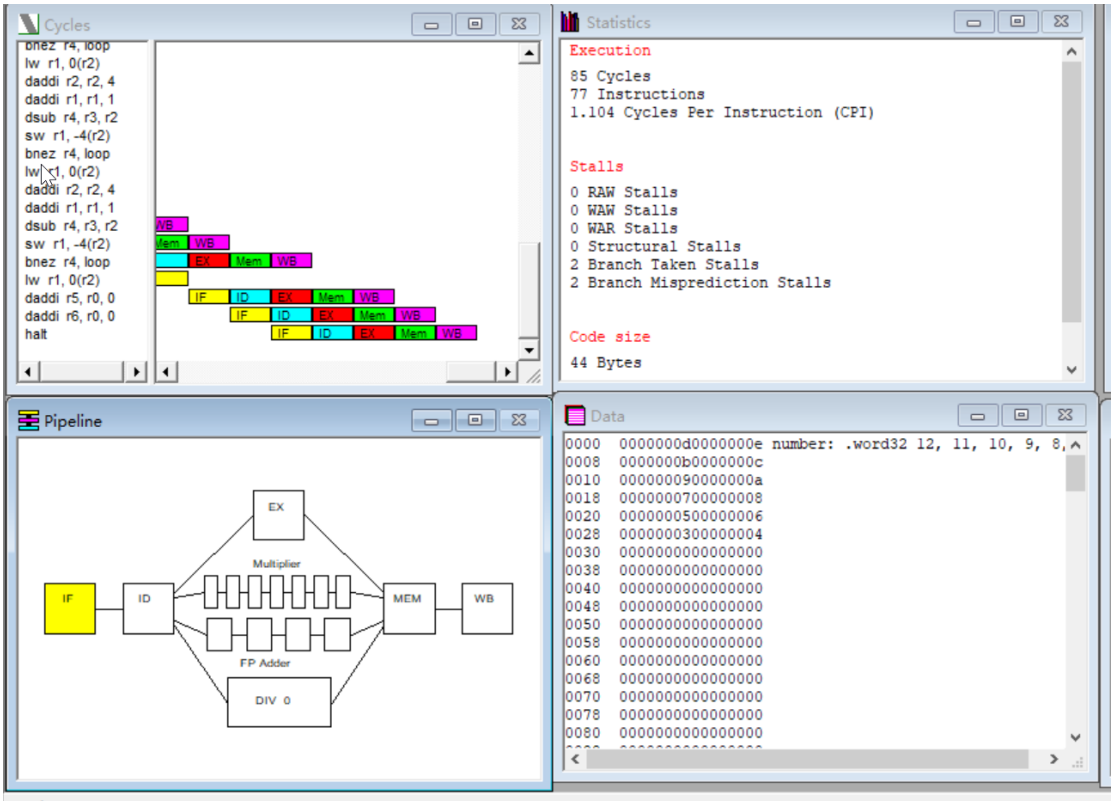



图表 2-2 允许 Enable forwarding 和不允许 Enable Branch Target buffer 结果

② 按要求做如下设置




图表 2-3 允许 Enable forwarding 和允许 Enable Branch Target buffer

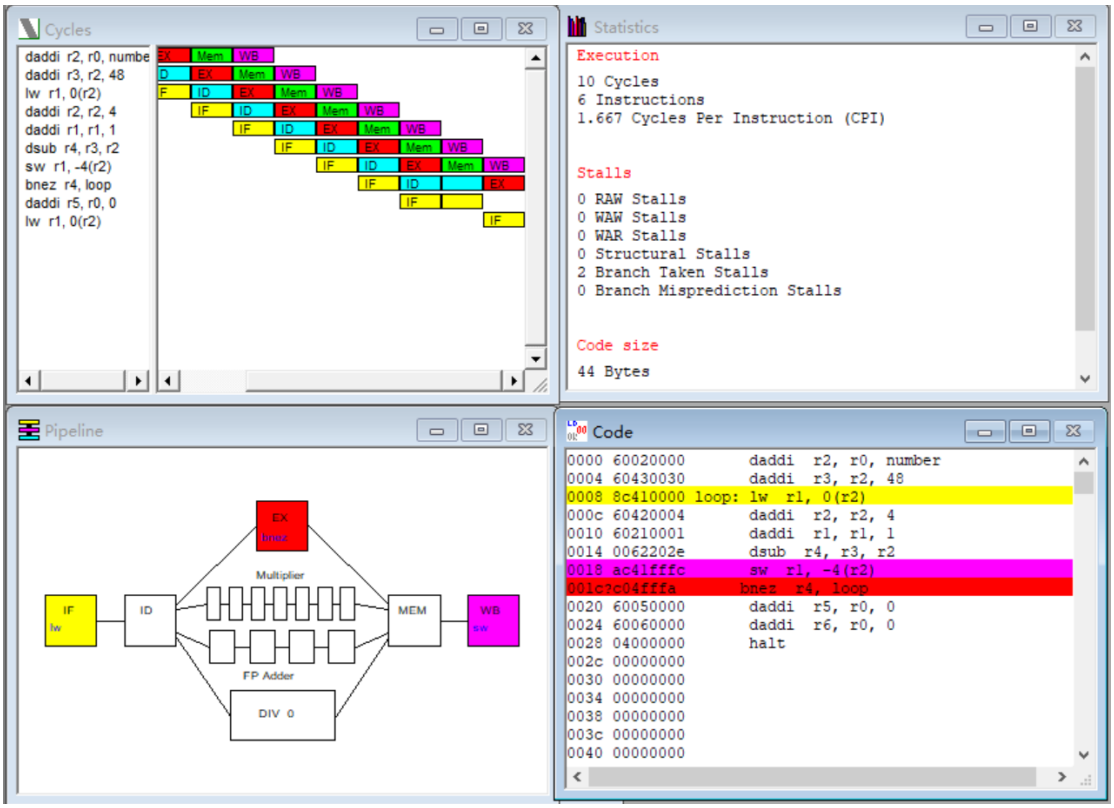



图表 2-4 允许 Enable forwarding 和允许 Enable Branch Target buffer 结果

### ③ BTB 减少运行周期数的解释：

BTB（Branch Target Buffer）采用两部分硬件结构进行动态的预测分支指令的跳转，他们是“地址标识”和“预测的分支目标地址”。这样的表格记录这执行过的分支指令跳转的地址，下次再遇到相同的指令的时候既可以直接跳到上次该条指令执行的地址进行。从而消除 stall，减少运行周期数。

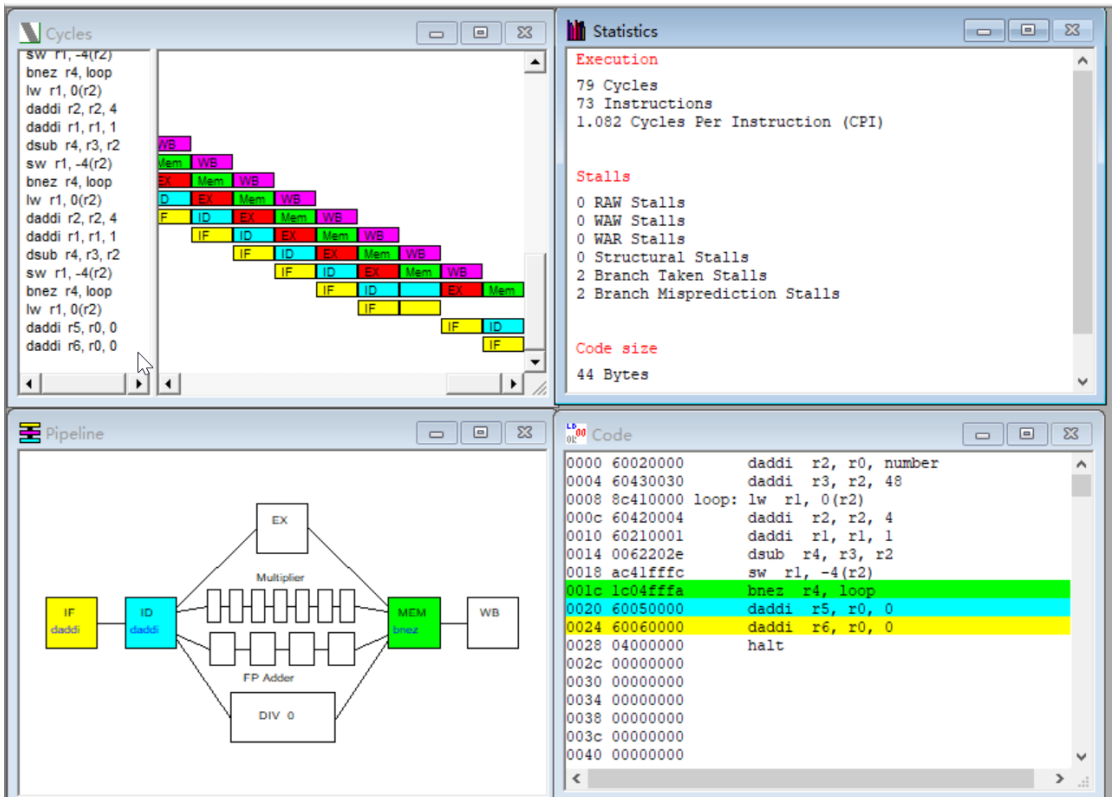
第一次遇到 bnz 指令时，BTB 中没有这个指令，于是按照基本的预测不跳转进行，之后 Stall，然后写入 BTB 表中，这时 BTB 记录了 BNZ 跳转的地址，即 LOOP。这样在下边循环中，除最后一次都可以预测正确。




**图表** 2-5 **第一次** stall

第二次 stall 是在出循环时，由于按照 BTB 中的地址，即预测为跳转，所以在 BNZ 的 ID 阶段后发现出错，于是 misprediction stall。修改 BTB 中的跳转地址。

修改地址需要 1 个周期，所以最后的结果是 taken stall 为 2， misprediction stall 也是 2。




图表 2-6 第二次 stall