北 京 邮 电 大 学
计 算 机 系 统 结 构 实 验 报 告

实验名称：	实验 2：流水线及流水线中的冲突
班　　级：	2016211301
学　　号：	2016211134
姓　　名：	李智盛
指导教师：	邝坚
实验日期：	2019 年 5 月 22 日

0. 实验目的

（1）加深对计算机流水线基本概念的理解。

（2）理解 MIPS 结构如何用 5 段流水线来实现，理解各段的功能和基本操作。

（3）加深对数据冲突和资源冲突的理解，理解这两类冲突对 CPU 性能的影响。

（4）进一步理解解决数据冲突的方法，掌握如何应用定向技术来减少数据冲突引起的停顿。

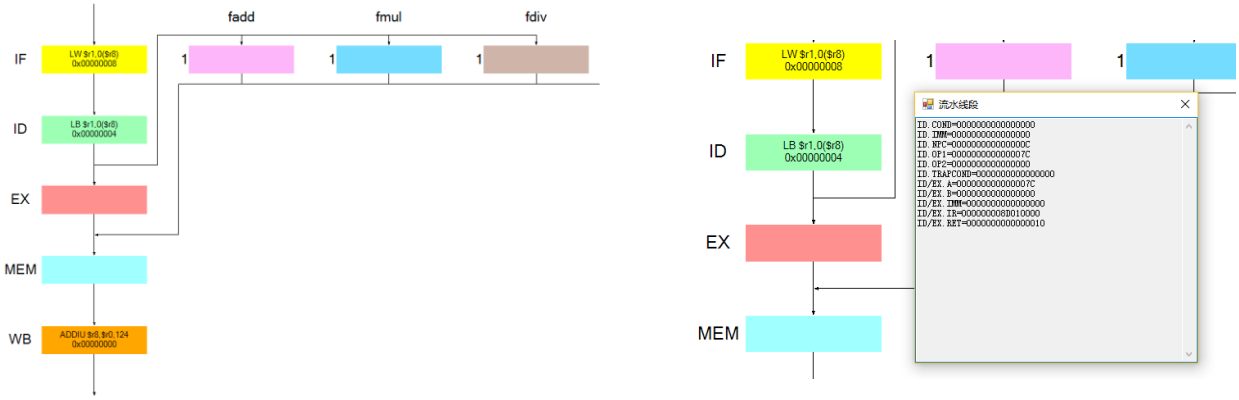
1. 实验平台

指令级和流水线操作级模拟器 MIPSsim

2. 实验内容和步骤

（1）启动 MIPSsim。

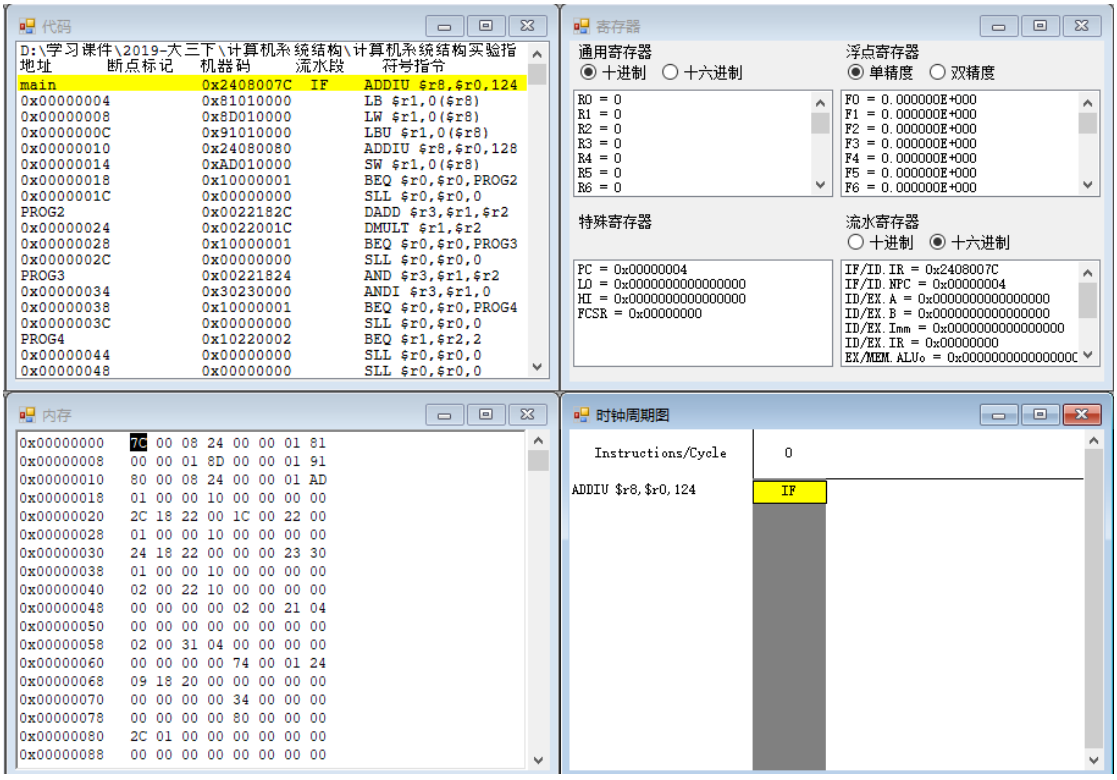
（2）进一步理解流水线窗口中各段的功能，掌握各流水寄存器的含义。（鼠标双击各段，即可看到各流水寄存器的内容）



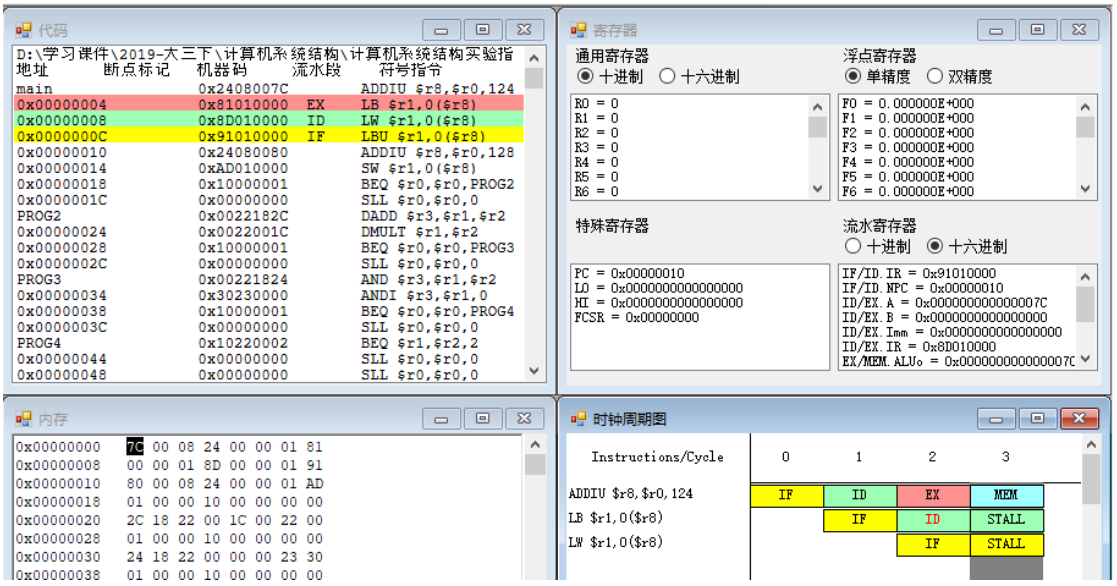
（3）载入一个样例程序（在本模拟器所在文件夹下的“样例程序”文件夹中），然后分别以单步执行一个周期、执行多个周期、连续执行、设置断点等方式运行程序，观察程序的执行

情况，观察 CPU 中寄存器和存储器内容的变化，特别是流水寄存器内容的变化。

- 单步执行

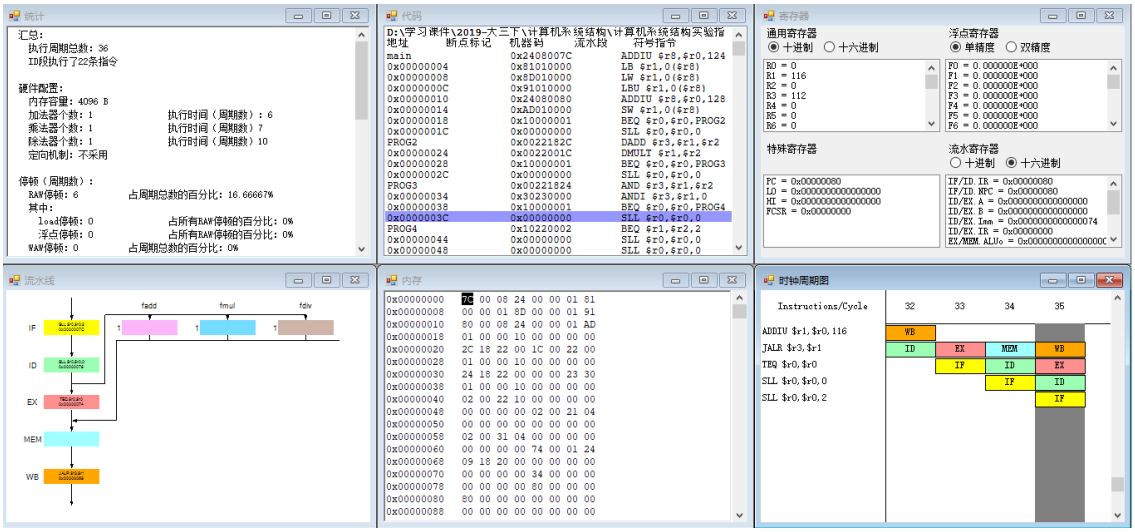


- 执行多个周期

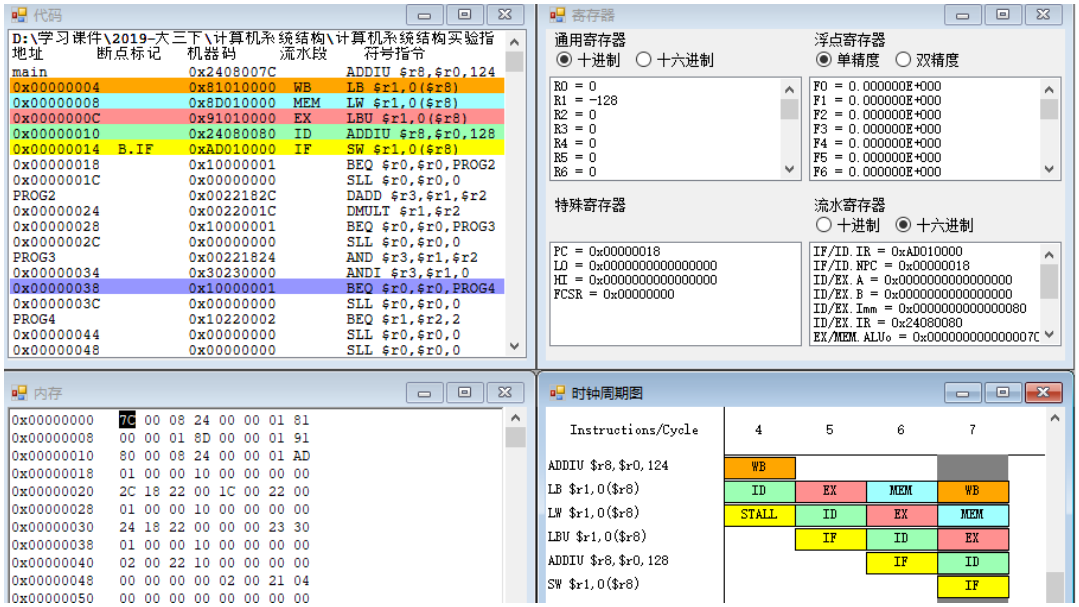


- 连续执行

  连续执行下将执行完整个程序



- 设置断点



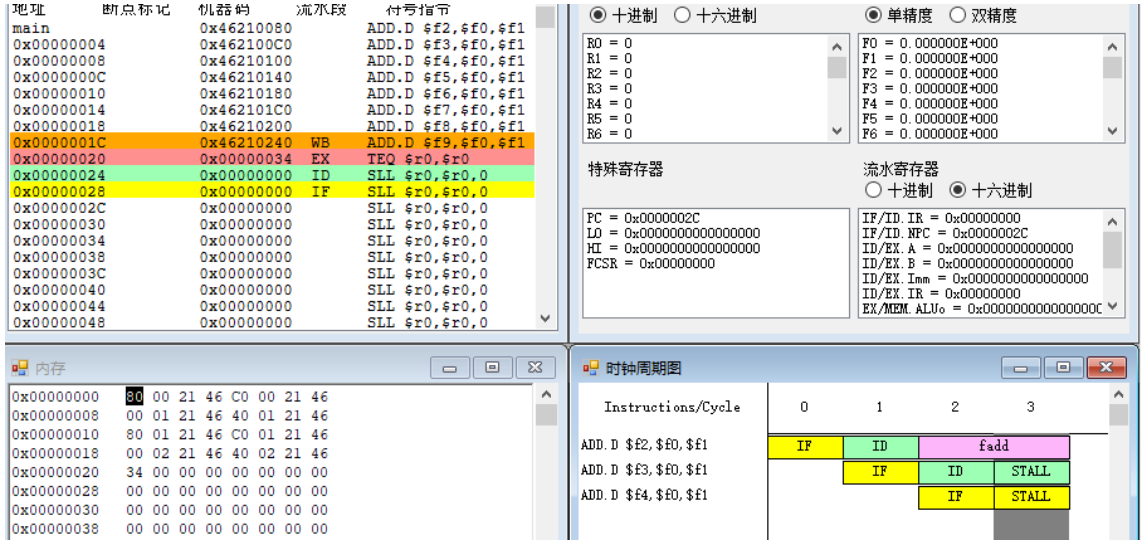
(4) 选择配置菜单中的“流水方式”选项，使模拟器工作于流水方式下。

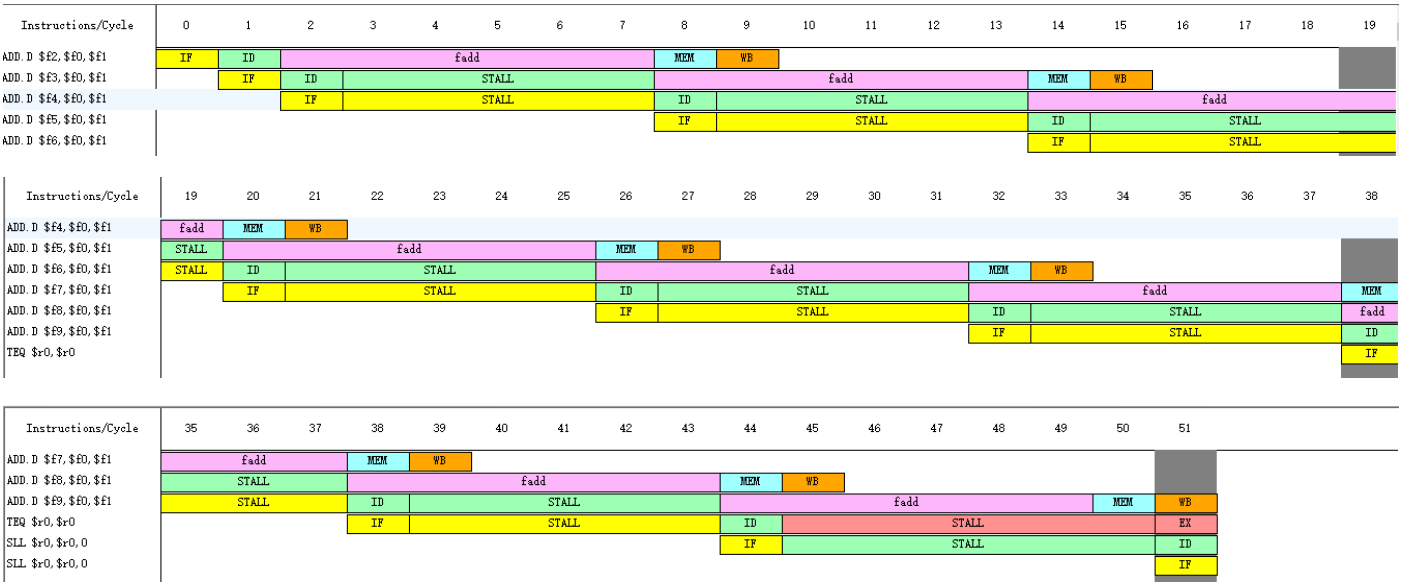
(5) 观察程序在流水方式下的执行情况。

(6) 观察和分析结构冲突对 CPU 性能的影响，步骤如下:

1) 加载 structure_hz.s（在模拟器所在文件夹下的“样例程序”文件夹中）。

2) 执行该程序，找出存在结构冲突的指令对以及导致结构冲突的部件。





存在结构冲突的指令：所有相邻的 ADD 计算指令间都存在结构冲突

导致结构冲突的部件：EX 段的 Fadd 浮点数加法器只有一个，上一条指令未执行完时，后续指令必须等待

3) 记录由结构冲突引起的停顿周期数，计算停顿周期数占总执行周期数的百分比。

汇总：

执行周期总数：52

ID段执行了10条指令

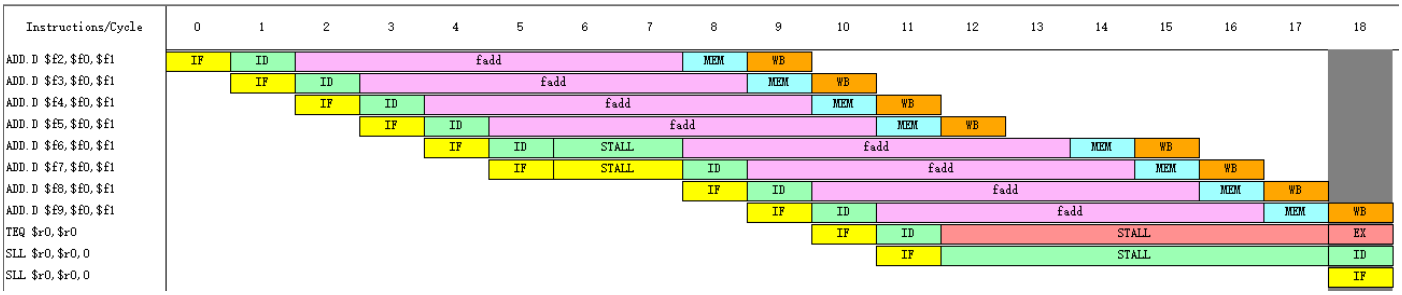
硬件配置：

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：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：35　　占周期总数的百分比：67.30769%
控制停顿：0　　占周期总数的百分比：0%
自陷停顿：6　　占周期总数的百分比：11.53846%
停顿周期总数：41　　占周期总数的百分比：78.84615%

4) 把浮点加法器的个数改为 4 个。

5) 再重复 1-3 的步骤。



浮点数加法器增加至 4 个，此时在流水方式下执行时，前 4 条指令都不会发生结构冲突而导致停顿，直至第五条指令到来，Fadd 加法器达到饱和，此时必须停顿等待之前的其他指令执行完毕后，释放出空闲的 Fadd。

6) 分析结构冲突对 CPU 性能的影响，讨论解决结构冲突的方法。

```
汇总：
  执行周期总数：52
  ID段执行了10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：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：35                占周期总数的百分比：67.30769%
  控制停顿：0          占周期总数的百分比：0%
  自陷停顿：6          占周期总数的百分比：11.53846%
  停顿周期总数：41     占周期总数的百分比：78.84615%
```

```
汇总：
  执行周期总数：9
  ID段执行了6条指令

硬件配置：
  内存容量：4096 B
  加法器个数：4                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：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：2         占周期总数的百分比：22.22222%
  控制停顿：0          占周期总数的百分比：0%
  自陷停顿：0          占周期总数的百分比：0%
  停顿周期总数：2      占周期总数的百分比：22.22222%
```

左边为浮点数加法器数量为 1 时的执行结果；右边为浮点数加法器数量为 4 时的执行结果

可以看出增加资源 Fadd 的数量能够有效的减少因为结构冲突导致的结构停顿，从而提高执行效率

解决结构冲突的方法：

- 增加资源数量
- 在有结构冲突的地方，将流水线停顿一定的时钟周期，等待资源的释放

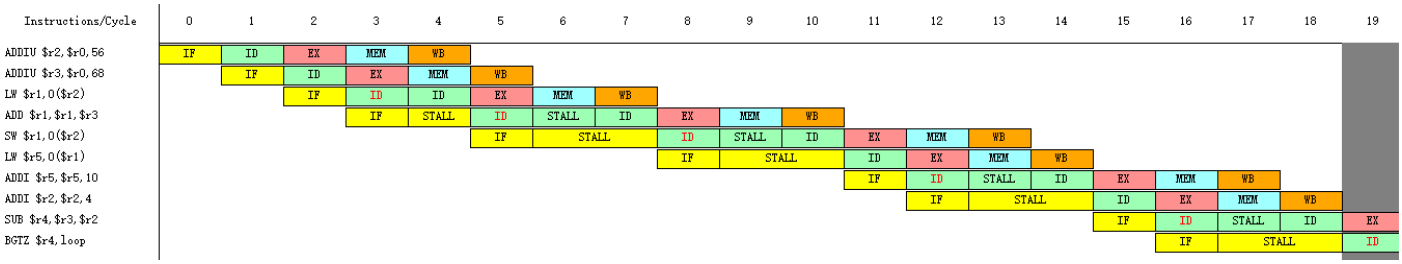
（7）观察数据冲突并用定向技术来减少停顿，步骤如下：

1) 全部复位。

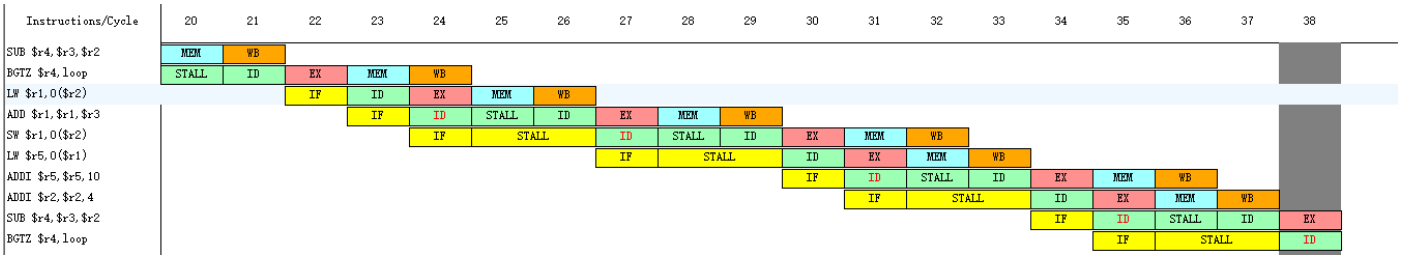
2) 加载 data_hz.s（在模拟器所在文件夹下的“样例程序”文件夹中）。

3) 关闭定向功能（在“配置”菜单下选择取消“定向”）。

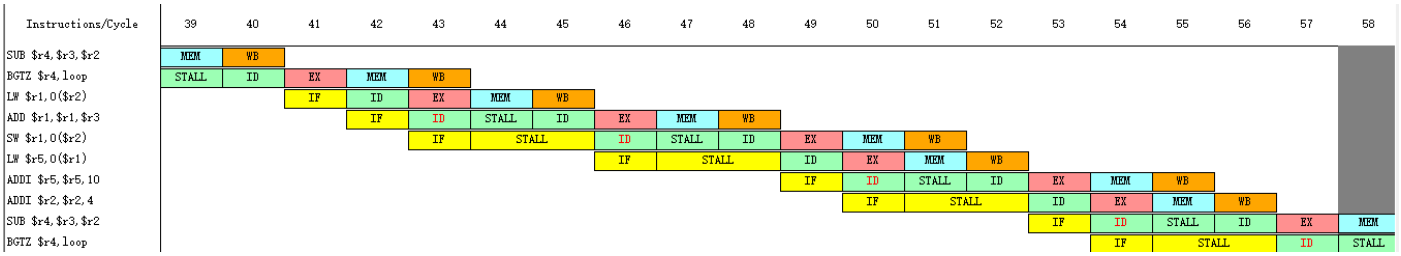
4) 用单步执行一个周期的方式执行该程序，观察时钟周期图，列出什么时刻发生了 RAW 冲突。



时刻：3，5，6，8，9，12 ，13，16，17，19，



时刻：20，24，25，27，28，31，32，35，36，38



时刻：39，43，44，46，47，50，51，54，55，57，58

5) 记录数据冲突引起的停顿周期数以及程序执行的总时钟周期数，计算停顿时钟周期数占总执行周期数的百分比。

汇总：

执行周期总数：65

ID段执行了29条指令

停顿（周期数）：

RAW停顿：31　　占周期总数的百分比：47.69231%

其中：

load停顿：12　　占所有RAW停顿的百分比：38.70968%

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：3　　占周期总数的百分比：4.615385%

自陷停顿：1　　占周期总数的百分比：1.538462%

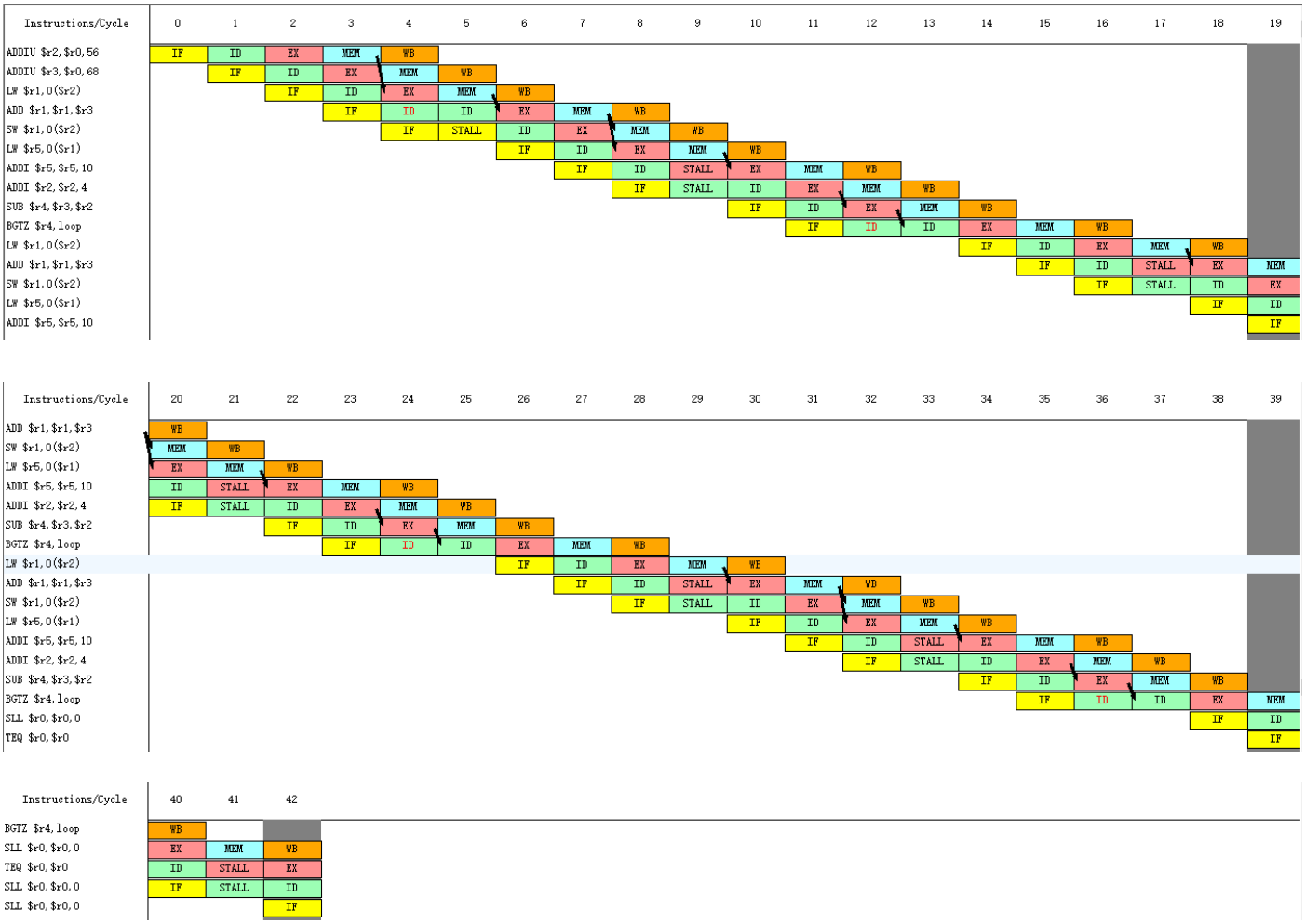
停顿周期总数：35　　占周期总数的百分比：53.84615%

6) 复位 CPU。

7) 打开定向功能。

8）用单步执行一个周期的方式执行该程序，查看时钟周期图，列出什么时刻发生了 RAW 冲突，并与步骤 3）的结果比较。

时钟周期图为：



RAW 停顿时刻：4，9，12，17，21，24，29，33，36

9) 记录数据冲突引起的停顿周期数以及程序执行的总周期数。计算采用定向以后性能比原来提高多少

左边为未使用定向，右边使用了定向功能：

汇总：

执行周期总数：65

ID段执行了29条指令

停顿（周期数）：

RAW停顿：31　　占周期总数的百分比：47.69231%

其中：

load停顿：12　　占所有RAW停顿的百分比：38.70968%

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：3　　占周期总数的百分比：4.615385%

自陷停顿：1　　占周期总数的百分比：1.538462%

停顿周期总数：35　　占周期总数的百分比：53.84615%

汇总：

执行周期总数：43

ID段执行了29条指令

停顿（周期数）：

RAW停顿：9　　占周期总数的百分比：20.93023%

其中：

load停顿：6　　占所有RAW停顿的百分比：66.66666%

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：3　　占周期总数的百分比：6.976744%

自陷停顿：1　　占周期总数的百分比：2.325581%

停顿周期总数：13占周期总数的百分比：30.23256%

总体效率提高为原来的 151.3%