Lab 2 报告

学号 2022K8009929011
姓名 王泽黎
箱子号 16

一、实验任务

本次实验任务分为三部分：

Exp7：不考虑相关冲突处理的简单流水线 CPU

Exp8：阻塞技术解决相关引发的冲突

Exp9：前递技术解决相关引发的冲突

二、实验设计

（一）总体设计思路

（1）五级流水线：

① 流水线阶段划分：

IF（Instruction Fetch)：指令获取阶段，从指令存储器中读取指令。

ID（Instruction Decode)：指令解码阶段，解析指令并读取寄存器文件中的操作数。

EX（Execute)：执行阶段，进行算术逻辑运算或地址计算。

MEM（Memory Access)：访存阶段，访问数据存储器进行读写操作。

WB（Write Back)：写回阶段，将计算结果写回寄存器文件。

② 在每个阶段之间设置寄存器，用于存储当前阶段的输出，作为下一个阶段的输入。

③ 通过设置流水线控制信号，控制数据和信号在流水级之间的传递。

（2）阻塞与前递：

① 数据前递：

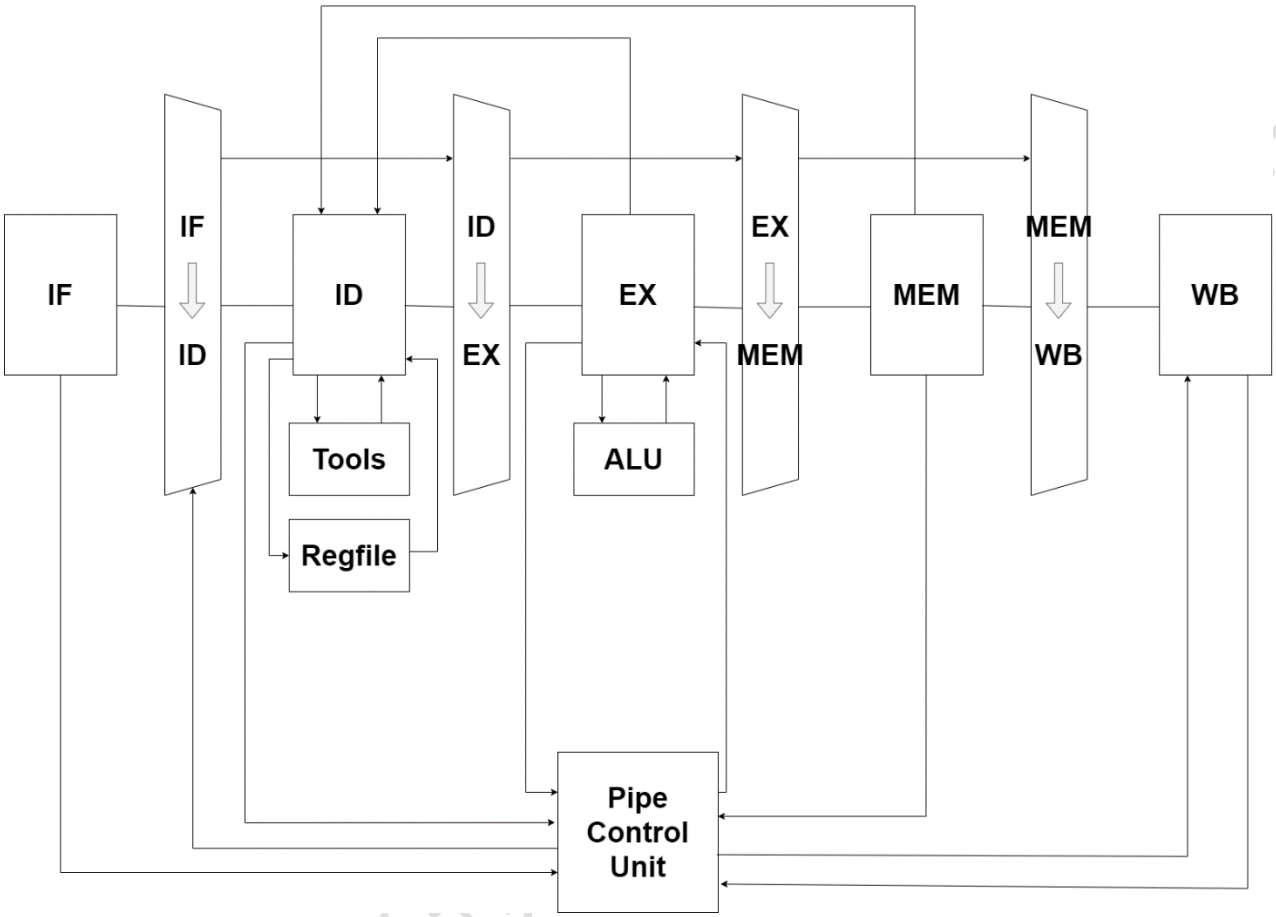
当指令之间存在数据依赖时，通过前递机制将数据直接从后续阶段传递到前面的阶段，避免等待数据写回寄存器文件。通过比较寄存器地址，判断是否需要前递，并选择合适的数据源（ALU 结果、访存结果等）。

② 阻塞机制：

当指令之间存在数据依赖且无法通过前递解决时，需要阻塞流水线，等待数据准备好。通过流

水线控制信号，阻止流水线向前推进。

（3）处理器结构设计框图：



三、实验过程

（一）Exp8 与 Exp9 性能比较

由于 Exp9 实际已完成阻塞和前递设计，单独在 Exp8 实现没有前递的阻塞功能就显得十分冗余，因此笔者并未使用两份代码进行仿真调试，进而对比性能。

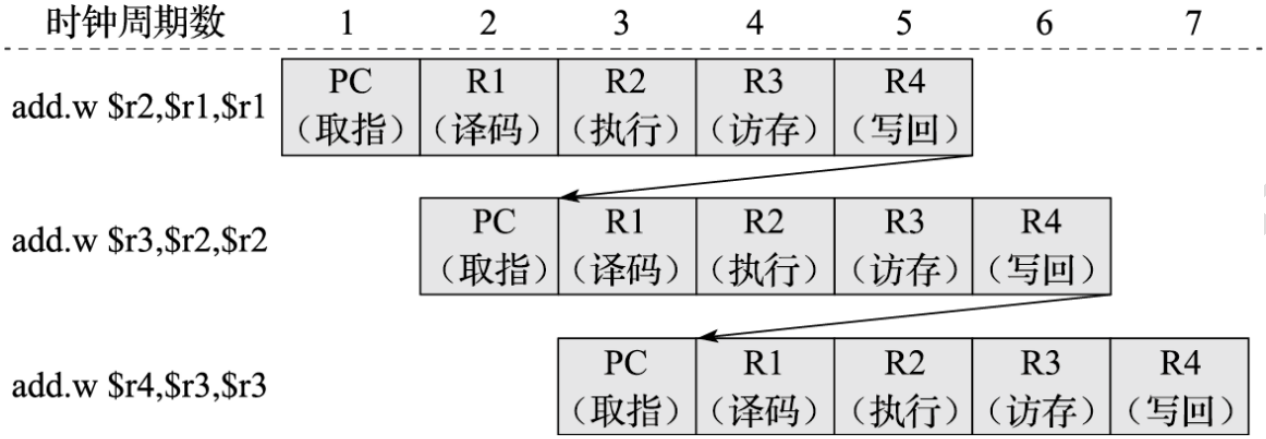
不过此处可以通过分析特定情况下流水线时空图说明同一段代码在 Exp8 和 Exp9 设计上运行时的性能差异。

对于如下指令序列：

```
add.w $r2, $r1, $r1
add.w $r3, $r2, $r2
ld.w $r4, $r3, 0
add.w $r5, $r4, $r4
```

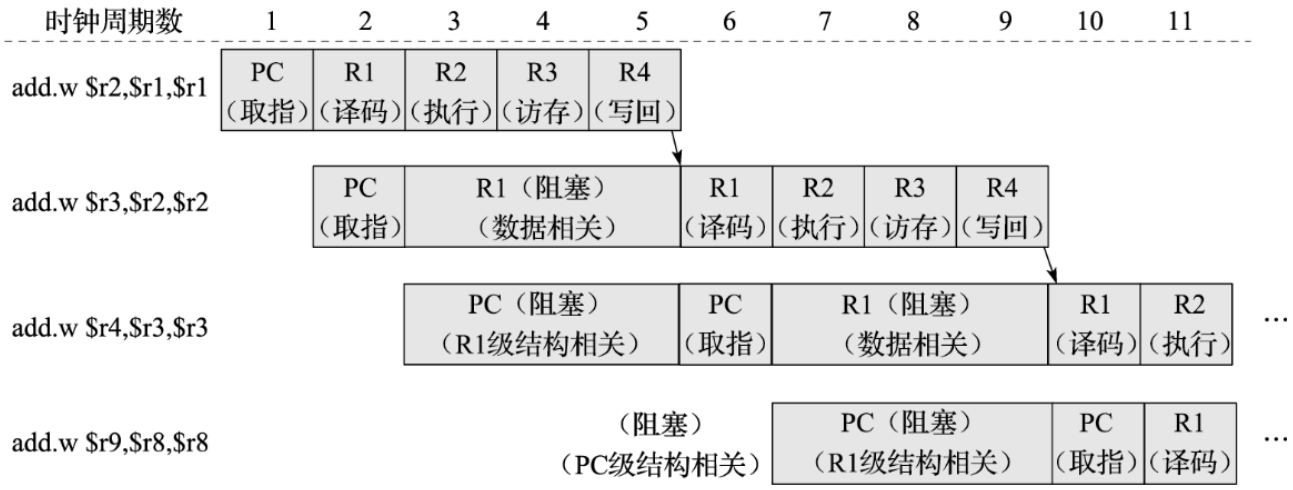
其中第 1、2 条指令间，第 2、3 条指令间，第 3、4 条指令间存在 RAW 相关。

其在五级流水线处理器中执行的流水线时空图如下所示：

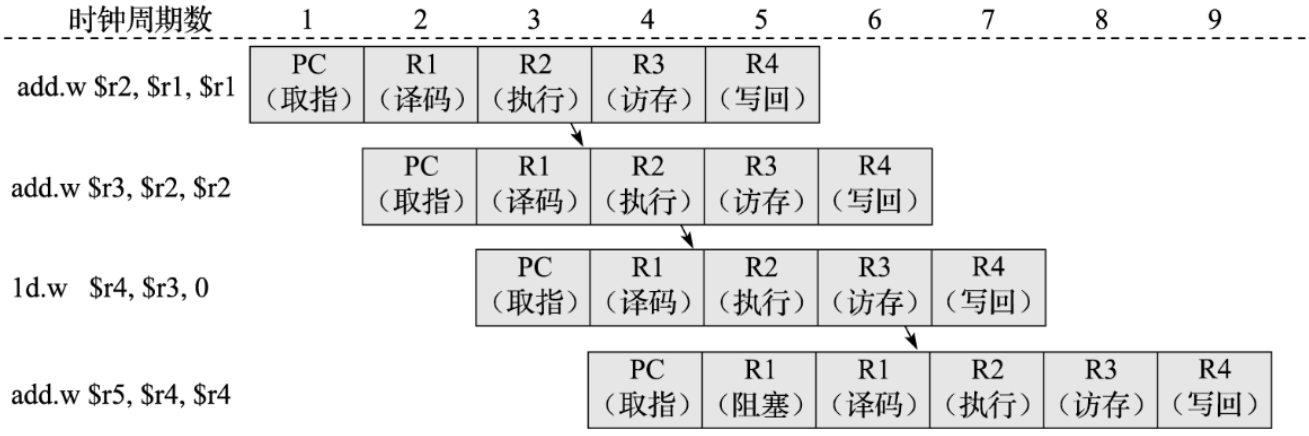


可见从第 1 条指令的写回阶段指向第 2 条指令的译码阶段的箭头以及从第 2 条指令的写回阶段指向第 3 条指令的译码阶段的箭头都表示 RAW 相关会引起冲突。

如果仅采用阻塞解决冲突，则有流水线时空图如下所示：



加入前递技术之后，则有流水线时空图如下所示：



由图可知，加入前递技术之后，执行这 4 条指令的性能有大幅提高。

（二）错误记录

1、错误 1：流水线控制信号异常

（1）错误现象

仿真调试跳过了大部分测试内容，直接 PASS，但在 FPGA 测试出现错误。

（2）分析定位过程

观察仿真波形发现第一条跳转指令未能成功跳转，进而发现由流水线控制信号控制的跳转信号异常。

（3）错误原因

流水线控制信号在每一个流水级更新时，更新位数错位，导致信号异常

（4）修正效果

将信号更新位数纠正，则实现正常流水线处理器功能

2、错误 2：数据前递异常

（1）错误现象

仿真调试中 ALU 相关数据前递正常，而访存数据前递异常。

（2）分析定位过程

观察仿真波形发现执行一条 add.w 指令时，相关寄存器涉及 add.w 和 ld.w ，其中前者结果前递正常，而后者数据前递异常。

（3）错误原因

译码阶段访存数据前递控制信号误写为 EX 阶段而非 MEM 阶段信号，进而导致错误。

（4）修正效果

将信号纠正，则实现正常前递功能

四、实验总结

本次实验加深了我对 LoongArch 架构下流水线处理器的理解，了解并实现了流水线处理器中的阻塞和前递技术。