

**实验三**

**流水线中的相关**

学科：计算机体系结构

学期：2024-2025 第一学期

编制日期：2024 年 11 月 11 日

编制人：江家玮

学号：22281188

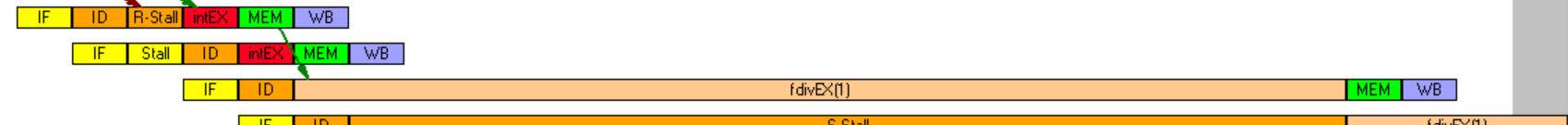
班级：计科2204

**实验一**

**1）结合你的运行结果说明程序段1中存在结构相关导致的暂停吗？什么原因导致的结构相关？**

**第一个结构相关：**

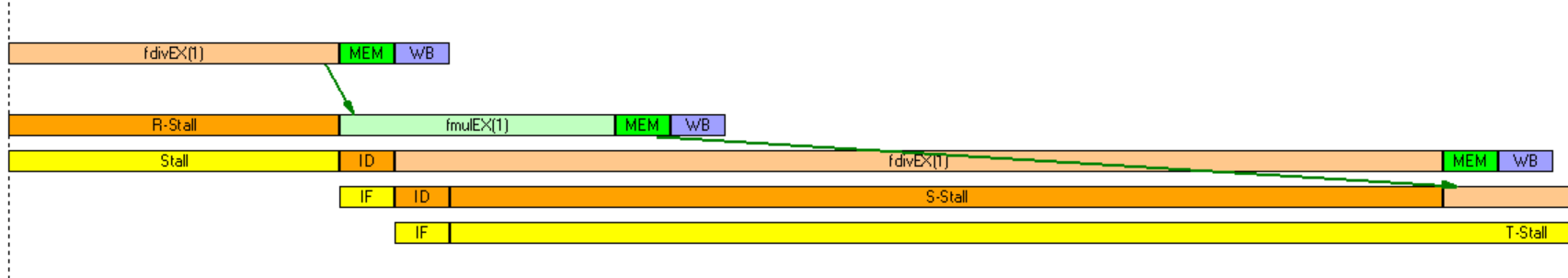
由于除法运算单元未实现流水线化，在前一个除法指令执行完毕之前，无法开始执行第二个除法指令。这是由硬件结构的限制引起的暂停。

****

**图1-1 第一个结构相关流水展示**

**第二个结构相关：**

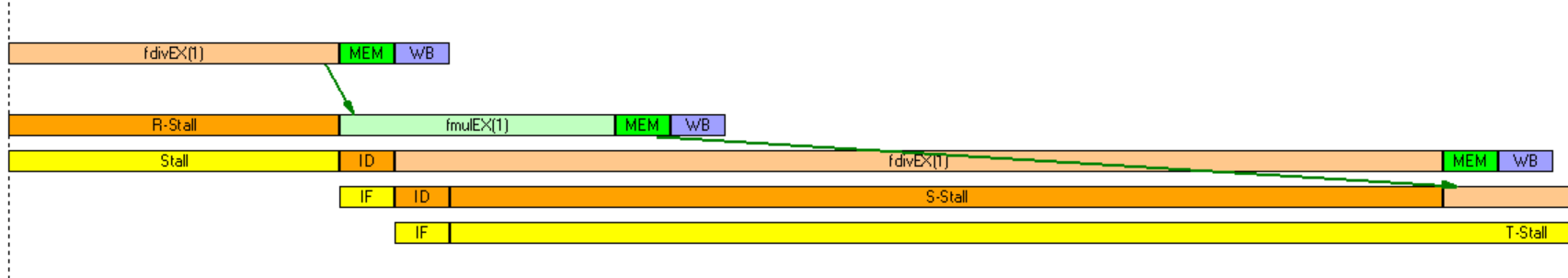
由于除法单元未实现流水线化，因此在前一个除法指令计算完成并退出执行阶段之前，后续的除法指令必须等待，无法并行执行。



**图1-6 第二个结构相关流水展示**

**第三个结构相关：**

由于除法运算单元未实现流水线化，在前一个除法指令执行完毕之前，无法开始执行第二个除法指令。这是由硬件结构的限制引起的暂停。



**图1-7 第一个结构相关流水展示**

##### **相关分析：**

结构相关是指多个指令需要同时访问相同的硬件资源（如ALU、寄存器、内存等），但这些资源在同一时刻只能被一个指令使用，从而导致指令无法同时执行，造成暂停。**结构相关的原因**是硬件资源的不足或资源共享引起的。例如，如果除法单元只有一个，并且没有流水线化处理，那么多个除法指令就无法并行执行，从而造成了暂停。

在程序段1中，除法操作（divf）在多条指令中频繁出现，且这些除法指令都需要共享除法单元（假设硬件中只有一个除法单元）。如果硬件中没有足够的资源或没有流水线化设计，那么在执行某一条除法指令时，其他的除法指令将无法执行，直到前一条除法指令完成。这种等待状态就表现为结构相关的暂停。

出现原因为除法单元没有实现流水化，刚好程序中有多处连续使用除法指令，导致出现结构相关。

**2）Why would a designer sometimes allow structural hazards?**

答：尽管通过对比整数单元、浮点和整数乘法器、浮点加法器等功能单元，浮点和整数除法器可以进行流水化设计，以便多个指令并行进入执行阶段（EX段），但浮点除法操作的周期需求较高。在本次实验中，除法操作的EX计算部分需要耗费19个周期，意味着流水化设计会分为19个周期进行，同时增加大量站间寄存器和数据转移的开销。考虑到除法指令在整体指令集中的占比较低，且结构相关仅影响性能而不影响程序结果，保持除法单元不进行流水化反而可能是更高效的选择。

此外，设计者选择允许结构相关的原因还包括：

**1. 简化硬件设计：**为了避免增加过多的硬件资源（如多个ALU或功能单元），设计者有时会选择简化硬件设计，容忍一定的结构相关。在一些嵌入式系统或低功耗设计中，允许结构相关可能会更节省成本和功耗。

**2. 性能权衡：**尽管结构相关可能导致性能下降，但这种权衡有时是可以接受的。设计者可能会更重视降低系统复杂度、减少功耗或确保设计制造的简化，而不是完全消除结构相关带来的影响。

**3. 流水线设计限制：**在一些设计中，尤其是老旧系统或固定架构，增加更多资源或优化以消除结构相关可能需要大幅度修改流水线设计。在这种情况下，设计者可能会选择在性能下降可接受的范围内容忍一定的结构相关。

**4. 成本与效益：**为了消除结构相关，增加更多的功能单元可能会带来效益递减。如果结构相关的发生频率较低，并且不会显著影响整体性能，设计者可能会认为增加硬件资源的成本不值得。

综上所述，允许结构相关是一种设计选择，通常基于简化硬件设计、权衡性能与其他因素、或是在成本与效益之间做出的平衡。

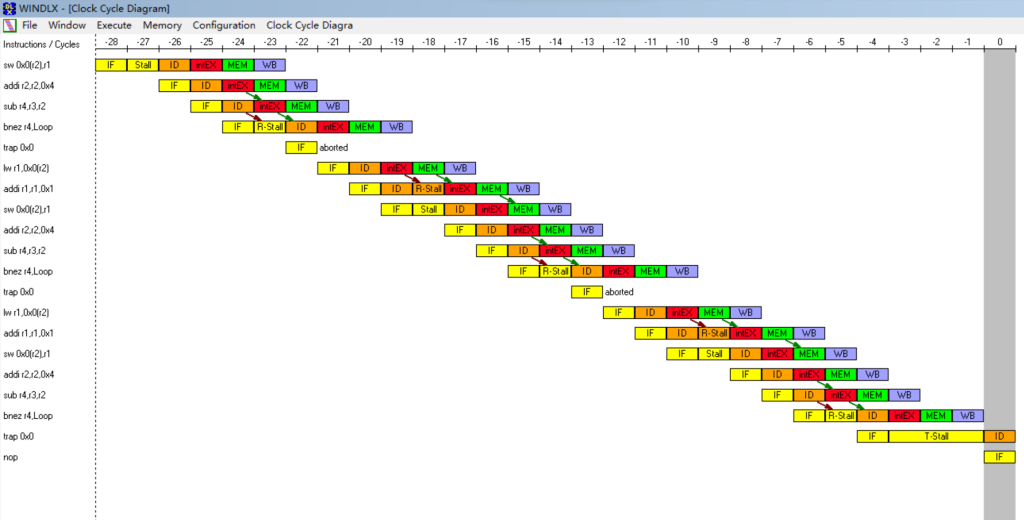
**实验二**

1. **观察分析使用定向技术与不使用定向技术的情况下，该程序段的执行周期数。画出在使用定向技术的情况下，一个循环的流水线时空图，并根据一个循环的时空图推算出整个程序的周期数。对比是否与模拟器中的clock cycle diagram以及程序中统计出的周期数一致。**

答：在使用了定向技术的前提下，整个程序段执行的周期数如图2-1所示。在使用了定向技术的前提下，程序总共执行了95个周期。

****

**图2-1 定向技术下的程序段执行结果**

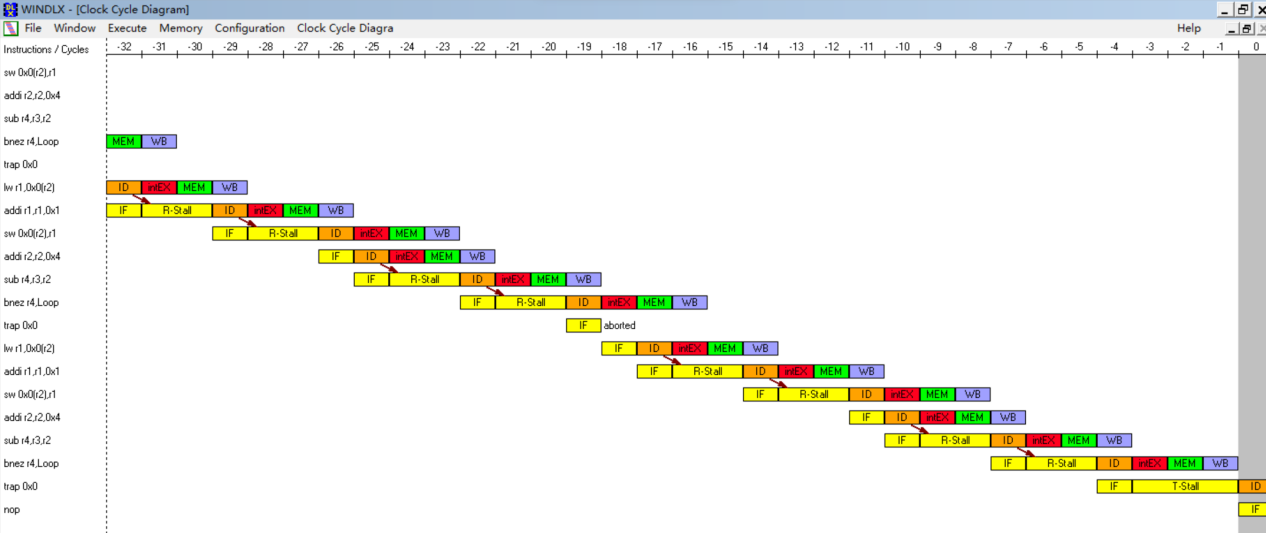


**图2-2 定向技术下的程序段流水展示**

在不使用定向技术的前提下，整个程序段执行的周期数如图2-3所示：在不使用定向技术的情况下，程序执行了155个周期。

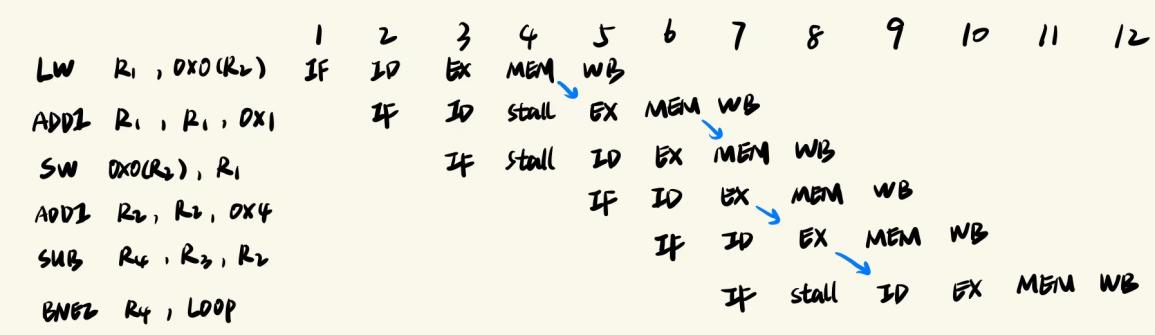
****

**图2-3 非定向技术下的程序段执行结果**

****

**图2-4 非定向技术下的程序段流水展示**

画出在使用定向技术的情况下一个循环的流水线时空图：



一次循环的执行需要12个周期。由于流水线采用预测失败的转移策略，因此两个循环之间的公共部分占用了3个周期。程序中循环部分执行了10次，此外还包含一个 trap 0 指令，同时在开头对 R3 进行了初始化，因此总的周期数为：1 + 12 + 9 \* 9 + 1 = 95个周期，与statistics中的说明一致。

1. **一次循环中的暂停周期数是多少，引起这些暂停的原因是什么？一次循环中有哪些相关，引起这些相关的原因是什么？**

答：一次循环中有两个暂停周期。通过观察流水线时空图可以看出，整形流水线包含6条指令，理论上无暂停情况下应为5 + 1 \* 5 = 10个周期，而实际执行时间为12个周期，因此存在2个周期的暂停。

第一个暂停原因：在第四周期出现数据相关问题。前一个指令是 load 指令，在 MEM 阶段才能得到 R1 的值，因此后续指令需要等待一个周期，才能在 EX 阶段通过前递获取所需数据。

第二个暂停原因：第九周期同样由于Data Hazard导致。经过将转移目标计算和条件测试前移到ID阶段的改进后，转移指令在 ID 阶段就需要知道 R4 的数值，但前一条指令在 EX 阶段才完成对R4的计算，因此需等待一个周期，才能通过前递将R4值传递给转移指令的 ID 阶段。

综上，一次循环中产生了Data Hazard和Control Hazard问题。Data Hazard源于流水线中的指令重叠执行，可能导致读写操作次序改变，数值结果与非流水线情况下的执行有所不同。而Control Hazar则由转移指令引起的 PC 值变化所导致。

1. **为了减少控制相关带来的暂停，我们学习了多种方法，该模拟器中使用的是哪种方法？该方法的思路是什么？在什么情况下可得到改进？**

答：在模拟器中显然使用了预测分支转移失败的策略。可以观察到，在十次循环中，每次执行条件转移指令后，流水线都会预先读取 trap 0 指令，确认转移成功后再取其他指令。

该方法的原理是：流水线正常推进，如果分支转移成功，则将分支后的指令置为空操作（取消执行），并从分支目标开始取指执行；若分支不成功，则继续按顺序执行。此外，取消的指令不会对数据产生影响，因为被取消的指令不会进入 WB 阶段。

这种方法在指令不转移的情况下可提升性能，而在指令转移时至少不会降低性能，因为转移条件和转移地址的获取本身就需要一个周期。

1. **In order to enable the forwarding paths in the above pipeline, what are the key observations needed to be implemented?**

要在上述流水线中启用forwarding paths，需要实现：

**1. 数据依赖性检测：**前递路径的关键在于能够检测指令之间是否存在数据相关，尤其是需要识别“读取后写入”型的依赖关系。具体来说：当一个指令需要使用某个寄存器的值，而该寄存器的值正由前一条指令在流水线的后续阶段（例如EX阶段或MEM阶段）生成时，系统应能在流水线中自动将数据从前一条指令传递给当前指令。

**2. 识别数据相关的类型：**需要明确三种主要的相关类型：

**RAW：**最常见的数据相关类型，即当前指令 需要读取的寄存器被前一条指令修改。这是最需要前递的情况。

**WAR：**当前指令修改一个寄存器，而前一条 指令正在读取该寄存器。

**WAW：**两条指令都写入同一个寄存器，通 常对流水线的影响较小。

**3. 判断前递的时机：**要检查指令中哪些操作在前一个阶段已经完成，哪些操 作尚未完成。对于存在数据相关的指令，前递路径需要确保将数据在正确的 时机（such as EX阶段）传递给依赖该数据的指令。

**4. 控制前递路径的开关：**在实现前递的硬件设计中，流水线的控制逻辑必须能够根据指令类型和阶段信息动态决定是否启用前递。例如，对于一个ADD指令，系统需要检查其输入寄存器是否在前一个ADD指令的EX/MEM/WB中生成。

**5. 监测指令之间的延迟**：前递路径必须考虑不同阶段之间的延迟。例如，如果一个指令的数据是在MEM阶段产生的，而另一个指令依赖于该数据，则必须确保前递路径在该时刻提供数据，而不是等到WB。

实现前递路径需要设计一个有效的数据相关检测机制，并在流水线中提供合适的路径来传递数据，避免不必要的暂停和等待。

**实验三**

**1）延迟槽中调度的是哪条指令？为什么选择该指令？调度后的程序与调度前的程序周期数相差多少？改进是如何体现的？**

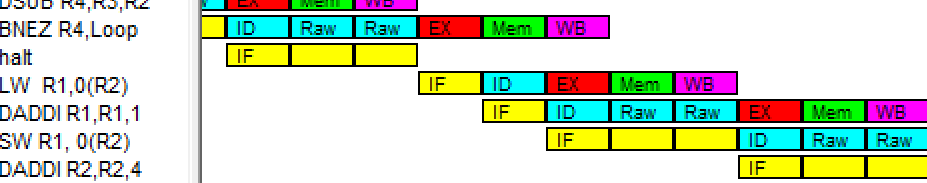
**答：**延迟槽中调度的是取数指令LW R1,0(R2); (load R1 from address 0+R2)。

选择该指令的原因：在程序中，主题的指令只有六条，其中的五条指令之间有着明确的先后关系：取指，计算，存入内存，计算，另外一条指令是转移指令。分析可知，只有第一条取数值的指令可以移动到延迟槽而不影响结果。并且还要在循环之前留一个load指令，用于为第一个循环取数。

调度后的程序比调度前减少了八个指令周期。

根据cycles分析可知：

改进前LW在上条指令的EX阶段才开始执行；



改进后LW在上条指令的ID阶段开始执行；



1. **观察流水线时空图，该模拟器采用的是改进前设计（page3-11，figure3.4）还是改进后设计（page3-38，figure3.22），为什么？改进后的设计是如何体现性能提升的？**

答：我认为是**改进后设计，**改进前设计特点，分支计算和条件判断在EX完成，因此分支指令必须等待EX阶段结束后才能确定是否跳转，这种设计会导致大量的Control Hazards。此外，由于缺乏针对数据相关问题的Forwarding Paths，只能通过插入暂停周期来解决相关问题，从而进一步降低了流水线的效率。

**改进后设计**特点：分支目标地址计算和条件判断被前移至ID，从而显著减少了分支指令引发的控制相关暂停。此外，通过包含Forwarding Paths，能够直接解决数据相关问题，有效降低暂停周期，进一步提升流水线的效率。

通过分析，从时空图来看，该模拟器采用的是**改进后的设计**，因为：

1. 分支指令的控制相关暂停周期显著减少，表明分支计算已前移到译码阶段。
2. 时空图中没有显示Load-Use Data Hazard引起的多周期暂停，表明模拟器支持前递路径。
3. BNEZ指令在ID阶段完成分支计算，从而大幅减少控制相关的停顿。

**改进后的设计如何体现性能提升？**

1. 减少Control Hazard带来的暂停：分支目标地址和条件判断前移到ID阶段，使得分支指令的跳转决策更早进行，减少了分支延迟。同时使用Delayed Branch技术，可以将延迟槽填充为有意义的指令，而不是简单地插入NOP，从而提高了流水线利用率。
2. Data Hazard优化：增加Forwarding Paths，允许指令间直接传递数据。例如，LW指令加载的数据可以在MEM阶段通过前递路径供后续指令使用，而无需等待WB阶段。
3. 改进后的流水线时空图表现：

分支相关：BNEZ指令只需一次分支延迟，而非改进前的多次暂停。

数据相关：LOAD指令后没有出现因数据相关而导致的多周期停顿。

**具体的性能提升体现：**

在时空图中，程序循环的执行周期减少至一个较小的固定周期数，分支跳转的开销显著降低。前递路径的有效利用减少数据相关导致的暂停。分支计算前移至ID阶段，减少了控制相关带来的分支预测延迟。延迟槽的使用提升了分支指令后一个周期的有效利用。