2024 年春季学期

体系结构实验报告

姓名：	蒲海博
班级：	2104102
学号：	2021211041

一、 静态 5 级流水线原理

静态 5 级流水线是计算机处理器中常见的一种设计。它是指将指令执行过程分为五个连续的阶段，每个阶段负责处理指令执行过程的不同部分，从而实现指令的并行执行。这五个阶段通常是：

1. 取指（Instruction Fetch）：从内存中获取下一条指令的地址，并将指令送入流水线。
2. 译码（Instruction Decode）：对取出的指令进行解码，确定执行的操作类型以及操作数。
3. 执行（Execute）：执行指令所指定的操作，可能涉及算术逻辑运算、数据传输等操作。
4. 访存（Memory Access）：如果指令需要访问内存，则在这个阶段进行内存读取或写入操作。
5. 写回（Write Back）：将执行阶段得到的结果写回到相应的寄存器或内存中。

在静态 5 级流水线中，每个指令都会依次经过这五个阶段，并且每个阶段都会同时处理不同指令的不同部分，从而实现指令的并行执行。这种流水线设计可以提高处理器的效率，使得多条指令可以在同一时间段内被处理，从而提高了处理器的吞吐量。

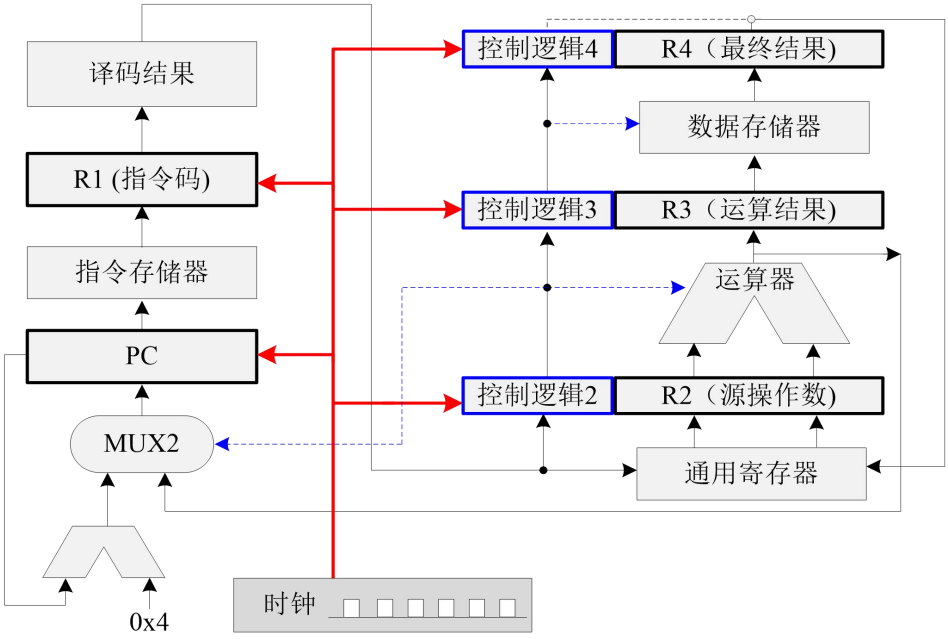


图 1-1 流水线 cpu 结构示意图

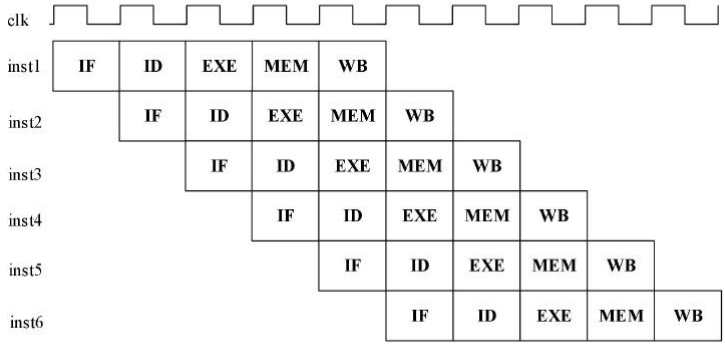


图 1-2 五级流水时空图

二、 静态 5 级流水线的相关问题

在流水线中，如果某指令的某个阶段必须等到它前面另一条指令的某个阶段后才能开始，则这两条指令存在相关。通常有结构相关、控制相关、数据相关三种类型。相关的指令要隔开足够远，否则后面的指令就必须等待，这造成了流水线阻塞。

1. 名相关：如果两条指令使用相同的名，但是它们之间并没有数据流动，则称这两条指令存在名相关。

2.控制相关：与 PC 有关的相关，通常是指由于分支指令（如条件分支或无条件分支）的执行结果还未确定而导致的冲突。当指令流中出现分支指令时，流水线可能会出现控制相关。常见的控制相关问题包括分支延迟和分支预测错误。

分支延迟（Branch Delay）：由于分支指令的执行需要一定的时间，流水线中后续的指令可能已经进入流水线，导致分支指令后面的指令无效，需要清空（Flush）流水线中的指令并重新开始执行。

分支预测错误（Branch Misprediction）：当流水线中的分支指令的执行结果与预测的分支方向不符时，之前预测的分支路径上的指令就会被清空，流水线需要重新开始执行。为了解决分支相关问题，现代处理器通常采用分支预测技术，通过预测分支的执行路径来减少分支相关带来的停顿。

3.数据相关：后面的指令使用前面指令的结果，通常分为三大类：

写后读（Write-After-Read，WAR）：当一个指令在写入一个寄存器或内存后，紧接着另一个指令需要读取该寄存器或内存中的数据，就会发生 WAR 相关。

写后写（Write-After-Write，WAW）：当一个指令在写入一个寄存器或内存后，紧接着另一个指令也需要写入同一个寄存器或内存，就会发生 WAW 相关。

读后写（Read-After-Write，RAW）：当一个指令在写入一个寄存器或内存后，紧接着另一个指令需要读取该寄存器或内存中的数据，就会发生 RAW 相关。

三、 静态 5 级流水线相关问题的解决办法

由于相关的存在，使得指令流中的下一条指令不能在指定的时钟周期执行，产生了流水线冲突。流水线冲突有 3 种类型：

1．结构冲突

在流水线处理机中，为了能够使各种组合的指令都能顺利地重叠执行，需要对功能部件进行流水或重复设置资源。如果某种指令组合因为资源冲突而不能正常执行，则称该处理机有结构冲突。

解决办法：

插入暂停周期（“流水线气泡”或“气泡”）；设置相互独立的指令存储器和数据存储器，或设置相互独立的指令 Cache 和数据 Cache。

2．数据冲突

当相关的指令靠得足够近时，它们在流水线中的重叠执行或者重新排序会改变指令读/写操作数的顺序，使之不同于它们串行执行时的顺序，则发生了数据冲突。

根据指令读访问和写访问的顺序，可以将数据冲突分为 3 种类型。

（1）写后读冲突（RAW）：在 i 写入之前，j 先去读，j 读出的内容是错的。

（2）写后写冲突（WRW）：在 i 写入之前，j 先写，最后写入的结果是 i 的。

（3）读后写冲突（WAR）：在 i 读之前，j 先写，i 读出内容错误。

解决办法：

（1）通过定向技术减少数据冲突引起的停顿，在计算结果尚未出来之前，后面等待使用该结果的指令并不真正立即需要该计算结果，如果能够将该计算结果从其产生的地方直接送到其他指令需要它的地方，那么就可以避免停顿；

（2）依靠编译器解决数据冲突，让编译器重新组织指令顺序来消除冲突，这种技术称为指令调度或流水线调度。

3．控制冲突：

流水线遇到分支指令和其它会改变 PC 值的指令所引起的冲突。

解决办法：

（1）在流水线中尽早判断出分支转移是否成功；

（2）尽早计算出分支目标地址。

四、 实验现象及结果分析

用于测试的汇编代码及编译后 16 进制机器码

addiu $2,$1,1　　24220001

addiu $3,$1,2　　24230002

addiu $4,$3,3　　24640003

and　$5,$4,$4　　00842824

流水线阻塞：

时空图

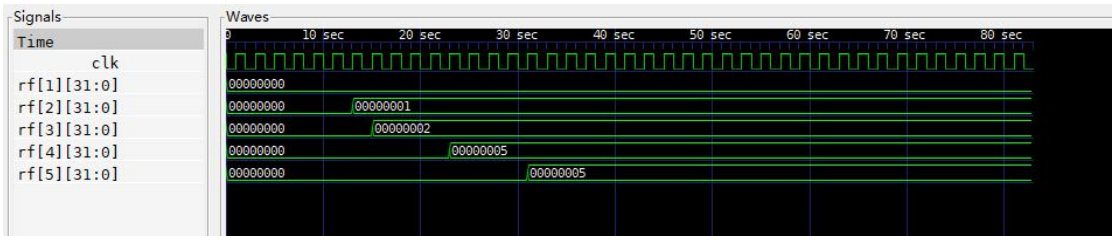
分析

1、 addiu $2,$1,1：无阻塞，1 和 1 号寄存器的值相加存入 2 号寄存器；

2、 addiu $3,$1,2：无阻塞，2 和 1 号寄存器的值相加存入 3 号寄存器；

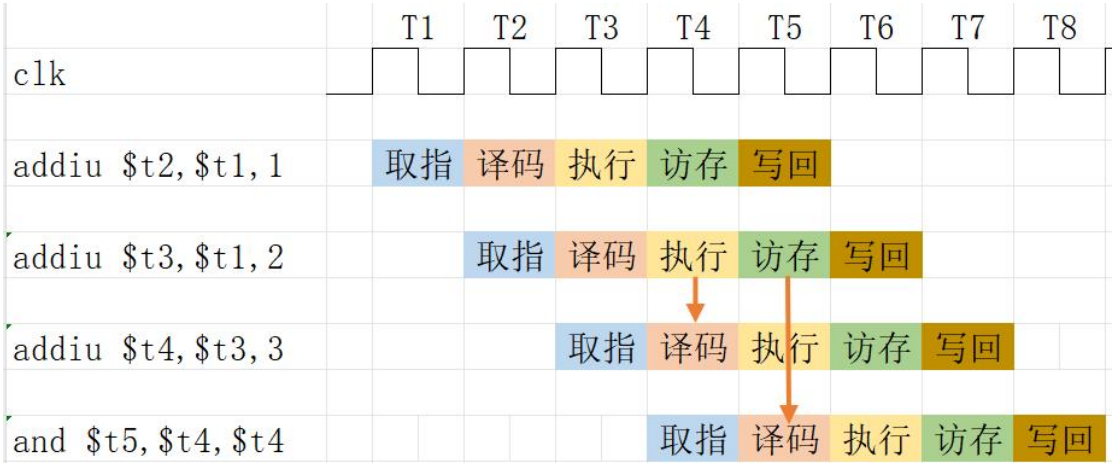
3、 addiu $4,$3,3：3 和 3 号寄存器的值相加存入 3 号寄存器。因为 3 需要等待第二条指令执行的结果写回 3 号寄存器，所以在译码阶段阻塞，直到第二条指令写回的下一拍才顺利译码；

4、 and $5 ,$4,$4：4 号寄存器的值与 4 号寄存器的值做与操作，存到 5 号寄存器中。由于第三条指令被阻塞在译码阶段，因此该指令前三拍被阻塞在取指阶段，直到第四拍才成功取指；又由于译码阶段需要从 4 号寄存器取数，而需要等待第三条指令将结果写回 4 号寄存器，因此在译码阶段又阻塞了三拍，之后才顺利往下执行；



流水线前递：

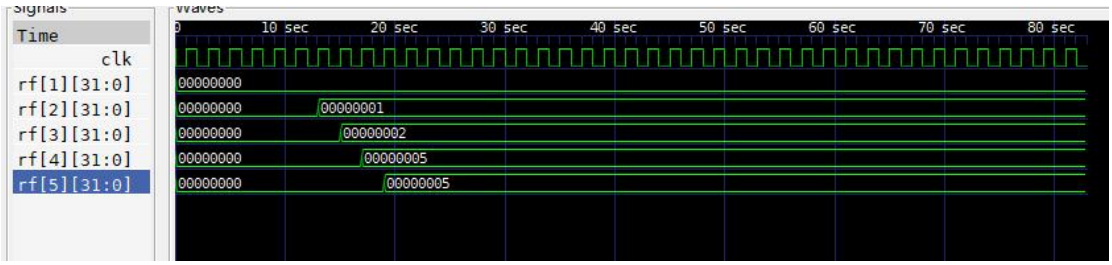
时空图



分析

1、第二条指令在执行阶段结束后，数字 2 和 1 号寄存器运算的结果从 ALU 和 DM 之间的流水寄存器直接送给第三条指令的 ALU 输入，无需等待写回 3 号寄存器再读出。属于 es 段数据前递。因此下一拍顺利进行执行阶段；

2、第二条指令在访存阶段结束后，DM 和 Reg 之间的流水寄存器的值直接送入第四条指令的 ALU 输入，属于 ms 段数据前递。因此第四条指令没有在译码阶段阻塞；



仿真图

五、 实验心得

通过本次五级流水实验，我对从体系结构课上学习的内容有了进一步的认识，并学会了通过观察仿真结果来分析程序。

实验过程中通过绘制时空图，我更深入地了解了流水线如何将指令分段执行，并在时间上交错，以使功能部件能够重叠运行，实现并行执行多条指令的目的。

通过设计五级流水线处理器，我深入理解了计算机的指令执行过程。从取指阶段到写回阶段，每个阶段都有其特定的功能和任务，而这些任务的协同工作使得指令能够被高效地执行。这种理解不仅帮助我掌握了计算机体系结构的基础知识，也提升了我对计算机工作原理的整体把握能力。学习了数据前递、流水线阻塞技术，对原本课程中所学的抽象的流水线有了具象化的感受。

经过本次实验，我对 MIPS 流水线有了更深入的了解，在实验过程中，我遇到了各种各样的挑战和困难，但通过不断地思考和尝试，我最终克服了这些困难，完成了任务。综上所述，五级流水线实验是一次非常有意义的学习经历。它不仅让我更深入地了解了计算机体系结构和硬件设计，还提升了我的问题解决能力和团队合作能力。这些收获将对我未来的学习和工作产生积极的影响。