ISA 既有 RISC 指令集的属性，也有 CISC 指令集的属性。然后，将不同指令组织放到五个阶段中处理，在此，根据被执行的指令的不同，每个阶段中的操作也不相同。据此，我们构造了 SEQ 处理器，其中每个时钟周期执行一条指令，它会通过所有五个阶段。

流水线化通过让不同的阶段并行操作，改进了系统的吞吐量性能。在任意一个给定的时刻，多条指令被不同的阶段处理。在引入这种并行性的过程中，我们必须非常小心，以提供与程序的顺序执行相同的程序级行为。通过重新调整 SEQ 各个部分的顺序，引入流水线，我们得到 SEQ+，接着添加流水线寄存器，创建出 PIPE－流水线。然后，添加了转发逻辑，加速了将结果从一条指令发送到另一条指令，从而提高了流水线的性能。有几种特殊情况需要额外的流水线控制逻辑来暂停或取消一些流水线阶段。

我们的设计中包括了一些基本的异常处理机制，在此，保证只有到异常指令之前的指令会影响程序员可见的状态。实现完整的异常处理远比此更具挑战性。在采用了更深流水线和更多并行性的系统中，要想正确处理异常就更加复杂了。

在本章中，我们学习了有关处理器设计的几个重要经验：

- *管理复杂性是首要问题。*想要优化使用硬件资源，在最小的成本下获得最大的性能。为了实现这个目的，我们创建了一个非常简单而一致的框架，来处理所有不同的指令类型。有了这个框架，就能够在处理不同指令类型的逻辑中共享硬件单元。
- *我们不需要直接实现 ISA。*ISA 的直接实现意味着一个顺序的设计。为了获得更高的性能，我们想运用硬件能力以同时执行许多操作，这就导致要使用流水线化的设计。通过仔细的设计和分析，我们能够处理各种流水线冒险，因此运行一个程序的整体效果，同用 ISA 模型获得的效果完全一致。
- *硬件设计人员必须非常谨慎小心。*一旦芯片被制造出来，就几乎不可能改正任何错误了。一开始就使设计正确是非常重要的。这就意味着要仔细地分析各种指令类型和组合，甚至于那些看上去没有意义的情况，例如弹出值到栈指针。必须用系统的模拟测试程序彻底地测试设计。在开发 PIPE 的控制逻辑中，我们的设计有个细微的错误，只有通过对控制组合的仔细而系统的分析才能发现。

网络旁注 ARCH:HCL　Y86-64 处理器的 HCL 描述

本章已经介绍几个简单的逻辑设计，以及 Y86-64 处理器 SEQ 和 PIPE 的控制逻辑的部分 HCL 代码。我们提供了 HCL 语言的文档和这两个处理器的控制逻辑的完整 HCL 描述。这些描述每个都只需要 5～7 页 HCL 代码，完整地研究它们是很值得的。

Y86-64 模拟器

本章的实验资料包括 SEQ 和 PIPE 处理器的模拟器。每个模拟器都有两个版本：

- GUI(图形用户界面)版本在图形窗口中显示内存、程序代码以及处理器状态。它提供了一种方式简便地查看指令如何通过处理器。控制面板还允许你交互式地重启动、单步或运行模拟器。
- 文本版本运行的是相同的模拟器，但是它显示信息的唯一方式是打印到终端上。对调试来讲，这个版本不是很有用，但是它允许处理器的自动测试。

这些模拟器的控制逻辑是通过将逻辑块的 HCL 声明翻译成 C 代码产生的。然后，编译这些代码并与模拟代码的其他部分进行链接。这样的结合使得你可以用这些模拟器测试原始设计的各种变种。提供的测试脚本，它们全面地测试各种指令以及各种冒险的可能性。

参考文献说明

对于那些有兴趣更多地学习逻辑设计的人来说，Katz 的逻辑设计教科书[58]是标准的入门教材，它强调了硬件描述语言的使用。Hennessy 和 Patterson 的计算机体系结构教科书[46]覆盖了处理器设计的广泛内容，包括这里讲述的简单流水线，还有并行执行更多指令的更高级的处理器。Shriver 和 Smith [101]详细介绍了 AMD 制造的与 Intel 兼容的 IA32 处理器。