- 虽然很少有人设计处理器，但是许多人设计包含处理器的硬件系统。将处理器嵌入到现实世界的系统中，如汽车和家用电器，已经变得非常普通了。嵌入式系统的设计者必须了解处理器是如何工作的，因为这些系统通常在比桌面和基于服务器的系统更低抽象级别上进行设计和编程。
- 你的工作可能就是处理器设计。虽然生产处理器的公司很少，但是研究处理器的设计人员队伍已经非常巨大了，而且还在壮大。一个主要的处理器设计的各个方面大约涉及 1000 多人。

本章首先定义一个简单的指令集，作为我们处理器实现的运行示例。因为受 x86-64 指令集的启发，它被俗称为“x86”，所以我们称我们的指令集为“Y86-64”指令集。与 x86-64 相比，Y86-64 指令集的数据类型、指令和寻址方式都要少一些。它的字节级编码也比较简单，机器代码没有相应的 x86-64 代码紧凑，不过设计它的 CPU 译码逻辑也要简单一些。虽然 Y86-64 指令集很简单，它仍然足够完整，能让我们写一些处理整数的程序。设计一个实现 Y86-64 的处理器要求我们解决许多处理器设计者同样会面对的问题。

接下来会提供一些数字硬件设计的背景。我们会描述处理器中使用的基本构件块，以及它们如何连接起来和操作。这些介绍是建立在第 2 章对布尔代数和位级操作的讨论的基础上的。我们还将介绍一种描述硬件系统控制部分的简单语言，HCL(Hardware Control Language，硬件控制语言)。然后，用它来描述我们的处理器设计。即使你已经有了一些逻辑设计的背景知识，也应该读读这个部分以了解我们的特殊符号表示方法。

作为设计处理器的第一步，我们给出一个基于顺序操作、功能正确但是有点不实用的 Y86-64 处理器。这个处理器每个时钟周期执行一条完整的 Y86-64 指令。所以它的时钟必须足够慢，以允许在一个周期内完成所有的动作。这样一个处理器是可以实现的，但是它的性能远远低于同样的硬件应该能达到的性能。

以这个顺序设计为基础，我们进行一系列的改造，创建一个*流水线化的处理器*(pipelined processor)。这个处理器将每条指令的执行分解成五步，每个步骤由一个独立的硬件部分或*阶段*(stage)来处理。指令步经流水线的各个阶段，且每个时钟周期有一条新指令进入流水线。所以，处理器可以同时执行五条指令的不同阶段。为了使这个处理器保留 Y86-64 ISA 的顺序行为，就要求处理很多*冒险*或*冲突*(hazard)情况，冒险就是一条指令的位置或操作数依赖于其他仍在流水线中的指令。

我们设计了一些工具来研究和测试处理器设计。其中包括 Y86-64 的汇编器、在你的机器上运行 Y86-64 程序的模拟器，还有针对两个顺序处理器设计和一个流水线化处理器设计的模拟器。这些设计的控制逻辑用 HCL 符号表示的文件描述。通过编辑这些文件和重新编译模拟器，你可以改变和扩展模拟器行为。我们还提供许多练习，包括实现新的指令和修改机器处理指令的方式。还提供测试代码以帮助你评价修改的正确性。这些练习将极大地帮助你理解所有这些内容，也能使你更理解处理器设计者面临的许多不同的设计选择。

网络旁注 ARCH:VLOG 给出了用 Verilog 硬件描述语言描述的流水线化的 Y86-64 处理器。其中包括为基本的硬件构建块和整个的处理器结构创建模块。我们自动地将控制逻辑的 HCL 描述翻译成 Verilog。首先用我们的模拟器调试 HCL 描述，能消除很多在硬件设计中会出现的棘手的问题。给定一个 Verilog 描述，有商业和开源工具来支持模拟和*逻辑合成*(logic synthesis)，产生实际的微处理器电路设计。因此，虽然我们在此花费大部分精力创建系统的图形和文字描述，写软件的时候也会花费同样的精力，但是这些设计能够自动地合成，这表明我们确实在创建一个能够用硬件实现的系统。