第十一章 RISC-V 未来的可选扩展

对于复杂性，傻子无视它，实用主义者忍受它，有的人有时能规避它，而天才则解决它。

——Alan Perlis, 1982

Alan Perlis（1922-1900），因为在高级编程语言和编译器领域的贡献而成为第一位图灵奖得主（1966）。1958 年参与设计了 ALGOL，对后来所有的命令式语言，包括 C 和 Java 都有巨大影响。

以 RISC-V 为基础，至少可以有 8 种可选的扩展。

11.1 “B”标准扩展：位操作

B 扩展提供位操作，包括插入、提取和测试位字段（insert, extract, and test bit fields），旋转（rotations），漏斗位移（funnel shifts），位置换和字节置换（bit and byte permutations），计算前导 0 和尾随 0（count leading and trailing zeros）和计算置位数（count bits set）等。

11.2 “E”标准扩展：嵌入式

为了降低对低端核心的开销，这个扩展少了 16 个寄存器。正是因为 RV32E，被保存寄存器和临时寄存器都是在 0-15 号和 16-31 号这两部分之间分开的（图 3.2[link]）。

11.3 “H”特权态架构扩展：支持管理程序（Hypervisor）

H 特权架构扩展加入了管理程序模式和基于内存页的二级地址翻译机制，提高在同一台计算机上运行多个操作系统的效率。

11.4 “J”标准扩展：动态翻译语言

有许多常用的语言使用了动态翻译，比如 Java 和 Javascript。这些语言的动态检查和垃圾回收可以得到 ISA 的支持。（J 表示即时（Just-In-Time）编译。）

11.5 “L”标准扩展：十进制浮点

L 扩展的目的是支持 IEEE 754-2008 标准规定的十进制浮点算术运算。二进制数的问题在于无法表示出一些常用的十进制小数，如 0.1。RV32L 使得计算基数可以和输入输出的基数相同。

11.6 “N”标准扩展：用户态中断

N 扩展允许用户态程序发生中断和例外后，直接进入用户态的处理程序，不触发外层运行环境响应。用户态中断主要用于支持存在 M 模式和 U 模式的安全嵌入式系统（见第 10 章）。不过，它也能支持类 Unix 操作系统中的用户态中断。当在 Unix 环境中使用时，传统的信号处理机制依然保留，而用户态中断可以用来做未来的扩展，产生诸如垃圾回收屏障（garbage collection barriers）、整数溢出（integer overflow）、浮点陷入（floating-point traps）等用户态事件。

11.7 “P”标准扩展：封装的单指令多数据（Packed-SIMD）指令

P 扩展细分了现有的寄存器架构，提供更小数据类型上的并行计算。封装的单指令多数据指令代表了一种合理复用现有宽数据通路的设计。不过，如果有额外的资源来进行并行计算，第 8[link]章的向量架构通常是更好的选择，设计者更应使用 RVV 扩展。

11.8 “Q”标准扩展：四精度浮点

Q 扩展增加了符合 IEEE 754-2008 标准的 128 位的四精度浮点指令。扩展后的浮点寄存器可以存储一个单精度、双精度或者四精度的浮点数。四精度浮点扩展要求 RV64IFD。

11.9 结束语

简化，简化。

——亨利·大卫·梭罗（Henry David Thoreau），19 世纪著名作家，1854

RISC-V 具有开放、标准的扩展方式，可能意味着可以在指令集最终确定之前得到反馈和争论，使得进一步的修改为时未晚。理想情况下，一小部分成员将先把一个提议实现出来，然后再提交通过，而 FPGA 上让实现的过程变得很容易。通过 RISC-V 基础委员会提交指令扩展所需工作量比较适中，他们将努力控制 ISA 变动的速度，至少不会像 x86-32 那样有太快的变化（见第[1]章的图 1.2[link]）。另外别忘了，不管有多少扩展被应用了，这一章提到的这些东西都是可选的。

我们希望 RISC-V 可以在保持简洁和高效的同时适应技术需求的发展。如果 RISC-V 成功了，它将成为以往增量式 ISA 上的一次革命性突破。