Risc-v 开发

Risc-v 基础知识

北京飞利信科技股份有限公司

2018 年 5 月

RISC-V 动态跟踪及历史发展情况

1 Risc-v 产生背景

Risc-v 是加州大学伯克利分校（UC Berkeley）设计并发布的一种开源指令集架构，其目标是成为指令集架构领域的 Linux，应用覆盖 IOT（Internet of Things）设备、桌面计算机、高性能计算机等众多领域。其产生是因为加州大学伯克利分校的研究人员在研究指令集架构的过程中，发现当前指令集架构存在如下问题：

① 绝大多数指令集架构都是受专利保护的，比如：x86、MIPS、Alpha，使用这些架构需要授权，限制了竞争的同时也扼制了创新。

② 当前的指令集架构都比较复杂，不适合学术研究，而且很多复杂性是因为一些不合理的设计或者背负历史包袱所带来的。

③ 当前的指令集架构都是针对某一领域的，比如：x86 主要是面向服务器、ARM 主要是面向移动终端，为此对应的指令集架构针对该领域做了大量的领域特定优化，缺乏一个统一的架构可以适用多个领域。

④ 商业的指令集架构容易受企业发展状况的影响，比如：Alpha 架构就随着 DEC 公司的被收购而几近消失。

⑤ 当前已有的各种指令集架构不便于针对特定的应用进行自定义扩展。

为此，加州大学伯克利分校的研究人员设计一种新的指令级架构，并以 BSD 授权的方式开源，希望借此可以有更多创新的处理器产生，有更多的处理器开源，并以此降低电子产品成本。

在第 4 届 Risc-v 专题研讨会上宣布成立了 Risc-v 基金会，吸纳了众多实力雄厚的商业公司和知名研究机构，其中包括中国科学院。可以预见，Risc-v 即将进入一个快速发展的阶段，应该会在以下几个方面有突破进展：

① 有若干成熟的、可商业化的、采用 Risc-v 架构的芯片问世，并得到大规模应用；

② 性能逼近主流桌面处理器；

③ 主流处理器与采用 Risc-v 架构的开源处理器组成的异构系统；

④ 移植到 Risc-v 架构的操作系统更加稳定可靠；

⑤ 采用上百个简单 Risc-v 核的多核并行计算；

⑥ 计算机教学中采用 Risc-v 作为范例教学；

⑦ 调试功能得到进一步加强。

对于国内而言，Risc-v 提供了一个很好的参考，可以用来实现自主可控的处理器。

2 Risc-v 概述

定义

RISC-V 是一种新的指令集架构（简称 ISA，instruction set architecture），旨在支持计算机体系结构的研究，定义 RISC-V 的目的包括：

- 将 ISA 打造成完全开放的指令集架构，免费提供给学术界和工业界。
- 能够适合直接在硬件上实现，而不仅仅是适用于模拟或者二进制翻译。
- 避免对某一种微体系结构风格（例如微编码、按序、去耦合、乱序等）或者实现技术（例如全定制、ASIC、FPGA）“过度体系结构化（over-architecting）”的 ISA，但是也能够高效地利用任何一种技术实现。
- 包含一个小的基本整数 ISA（可以作为一个定制的加速器的基础或者作为教学用途）和多个可选的标准扩展的 ISA，可以支持通用的软件开发。
- ISA 支持丰富的用户级 ISA 扩展和各种特殊的变种。
- 对应用程序、操作系统内核、硬件实现的 32 位、64 位地址空间变种。
- ISA 支持高度并行的多核、众核实现，包括异构多处理器等。
- 可选的变长指令，支持扩展可用的指令编码空间、支持一个可选的密集指令编码，以提高性能、静态代码大小和能耗效率。
- 一个可完全虚拟化的 ISA，以简化虚拟机监督管理器（Hypervisor）的开发。
- ISA 支持新的管理员级（supervisor-level）和虚拟机监督管理级（hypervisor-level）ISA 设计。

Risc-v 基本设计

RISC-V 是一个典型三操作数、加载-存储形式的 RISC 架构，包括 3 个基本指令集和 6 个扩展指令集，如下表所列。

指令集类型	名　称	指令数	说　明
基本指令集	RV32I	47	整数指令，包含：算术、分支、访存。32 位寻址空间，32 个 32 位寄存器
	RV32E	47	指令与 RV32I 一样，只是寄存器数量变为 16 个，用于嵌入式环境
	RV64I	59	整数指令，64 位寻址空间，32 个 64 位寄存器
	RV128I	71	整数指令，128 位寻址空间，32 个 128 位寄存器
扩展指令集	M	8	包含 4 条乘法、2 条除法、2 条余数操作指令
	A	11	包含原子操作指令，比如：读-修改-写，比较-交换等
	F	26	包含单精度浮点指令
	D	26	包含双精度浮点指令
	Q	26	包含四倍精度浮点指令
	C	46	压缩指令集，其中的指令长度是 16 位，主要目的是减少代码大小

RISC-V 指令集被设计成一个丰富的可定制化的指令集。基本整数指令集体系结构 ISA 能够通过添加一个或者多个可选的指令集扩展来进行功能的增强，但是基本整数集本身不允许被扩展。将 RISC-V 指令集扩展分为标准扩展和非标准扩展。标准扩展一般都是有用的，并且与其它的标准扩展并不冲突。非标准扩展是高度特殊化的，并可能与其它的标准扩展或者非标准扩展冲突。指令集扩展根据基本整数指令集宽度不同，可能有轻微的功能差异。基本指令集的名称后缀都是 I，表示 Integer，任何一款采用 RISC-V 架构的处理器都要实现一个基本指令集，根据需要，可以实现多种扩展指令集，例如：如果实现了 RV32IM 表示实现了 32 位基本指令集和乘法除法扩展指令集。如果实现了 RV32IMAFD，那么可以使用 RV32G 来表示实现了通用标量处理器指令集。

常见扩展指令集有：

（1）“M”扩展，用于整数的乘法和除法的运算，其中增加了对保存在整数寄存器中的值进行乘法和除法的指令；

（2）“A”扩展，用于实现原子性操作，其中增加了对存储器进行原子的读、修改和写操作的指令，以支持处理器间的同步；

（3）“F/D/Q”扩展，用于实现单精度/双精度/四倍精度的浮点运算；

（4）“C”扩展，压缩指令的实现。

（5）一个基本整数内核加上这四个标准扩展（“IMAFD”），被缩写为“G”，它提供了一个通用的标量指令集。

指令长度编码

基本 RISC-V ISA 具有 32 位固定长度指令，并且必须在 32 位边界对齐。然而，标准 RISC-V 编码模式被设计成支持变长指令的扩展，在这个扩展中，每条指令长度可以是 16 位指令包裹（parcel）长度的整数倍，并且这些指令包裹必须在 16 位边界对齐。

下图展示了标准 RISC-V 指令长度编码约定。所有基本 ISA 中的 32 位指令的最低 2 位被设置为 11。可选的压缩 16 位指令集扩展中的指令，最低 2 位被设置为 00、01 或者 10。超过 32 位的标准指令集扩展，在低位有额外的位被设置为 1，48 位、64 位长度约定如下图所示。指令长度在 80 位到 176 位之间的长度信息，被编码到一个 3 位的字段[14:12]中，给出了 16 位字的数量，加上最开始的 5×16 位字。位[14:12]编码为 111，保留给未来更长的指令编码。

		xxxxxxxxxxxxxxaa	16 位（aa≠11）
••• xxxx	xxxxxxxxxxxxxxxx	xxxxxxxxxxbbb11	32 位（bbb≠111）
••• xxxx	xxxxxxxxxxxxxxxx	xxxxxxxxxx011111	48 位
••• xxxx	xxxxxxxxxxxxxxxx	xxxxxxxxx0111111	64 位
••• xxxx	xxxxxxxxxxxxxxxx	xnnnxxxxx1111111	（80+16*nnn）位，nnnn≠1111
••• xxxx	xxxxxxxxxxxxxxxx	x111xxxxx1111111	保留给≥192 位
字节地址：	基址+4	基址+2	基址

图 1.1 RISC-V 指令长度编码

基本 RISC-V ISA 具有一个小端存储器系统，但是非标准变种可以提供大端或者双端存储器系统。指令被保存在存储器中，每个 16 位包裹以实现的端字节顺序，被保存到一个存储器半字中。包含一条指令的包裹，被保存到递增的半字地址，其中最低寻址的包裹保存着指令规范中最低位的二进制值，也就是说，指令总是按照一系列包裹的小端顺序保存的，而不管存储器的端字节顺序。图中的代码序列，将把一条 32 位指令正确地保存到存储器中，而不管存储器的端字节顺序。

```
// 将x2中的32位指令，保存到x3指向的存储器
sh          x2, 0(x3)     // 将指令的低半部分保存到第一个包裹中
srli        x2, x2, 16    // 将高位移动到低位，覆盖x2
sh          x2, 2(x3)     // 将高位保存到第二个包裹中
```

图 1.2 将 32 位指令从寄存器保存到存储器的推荐代码序列。在大端、小端存储器系统中都能正确工作，当使用变长指令集扩展时，可避免出现非对齐访问。

异常、自陷和终端

异常（exception）认为是在运行时出现了一个与当前 RISC-V 线程中的一条指令相关的非正常的情况。

自陷（trap）认为是在一个 RISC-V 线程中出现了一个异常的情况，导致将控制同步传输到自陷处理函数。自陷处理函数通常是在一个更高特权环境中执行的。

中断（interrupt）认为是在当前 RISC-V 线程外异步出现了一个事件。如果出现了一个必须处理的中断，将会选择某条指令来接收中断异常，然后顺序地产生一个自陷。

异常是否和如何转变为自陷的，依赖于执行环境，虽然预期是绝大多数环境在一个异常被触发时（signaled），采取一个精确的自陷（除了浮点异常，在标准浮点扩展中，并不会产生自陷）。

参考文献： riscv-spec-v2.2