龙腾指令集参考手册

Rev 0.01

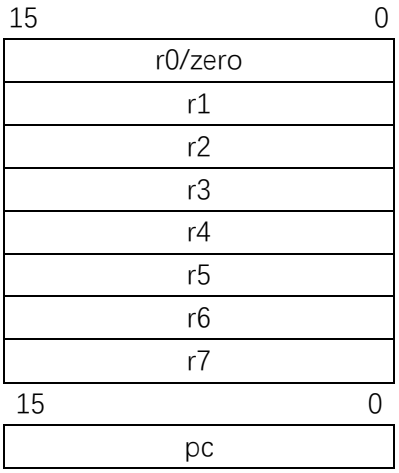
概览

龙腾（英文：Dragonfly）是一个全新的，拥有自主知识产权的 16 位 RISC 指令集架构。龙腾架构的设计目标是简单易用，精简高效，适合在各种平台上实现。龙腾指令集主要面向电子爱好者，计算机组成原理初学者以及广大初高中学生，但由于其结构简单，易于实现的优势，未来也可以使用在嵌入式、物联网和边缘计算领域。龙腾指令集的主要特点包括：

- 16 位定长指令，四种指令编码格式，指令集可扩展
- 28 条指令超精简 RISC 核心
- 16 位数据，Load/Store 架构
- 8 个通用寄存器（包括一个零寄存器）指令和寄存器完全正交，每个寄存器都可以完成所有功能
- 16 位指令地址总线，可直接访问 128KiB 指令存储器
- 16 位数据地址总线，附加三位选择位，最多可以访问 1MiB 数据存储器。
- 指令编码参考 MIPS 和 RISC-V，方便学习

1.1 寄存器组

龙腾指令集共有八个通用寄存器，r0-r7，通用寄存器全部为 16 位宽。其中 r0 是硬件零寄存器，对此寄存器的读取将永远返回 0。任何通用寄存器可以用作算术运算指令的操作数或目标寄存器，也可以用作跳转指令的基地址寄存器或者 Load/Store 指令的地址寄存器和数据寄存器。龙腾指令集还有一个专用寄存器，程序计数器（pc）。程序计数器是 16 位宽，并永远指向下一条指令的地址。



1.2 指令编码

龙腾指令集架构共有四种基础指令集编码：R 类（寄存器-寄存器指令），I 类（寄存器-立即数指令），B 类（有条件跳转指令）和 U 类（高位立即数指令）。四种编码都是 16 位长，在内存中对齐存储。所有跳转指令都按字寻址，即每次寻址 16 位内存单元，也可以看作跳转指令的地址末尾默认是 0。四种指令格式编码如下：

15 14	13 11	10 8	7 5	4 2	1 0	
funct2	rs1	rs2	rd	funct3	opcode	R 类指令
imm[0:4]		rs2	rd	funct3	opcode	I 类指令
imm[4:3]	rs1	rs2	imm[2:0]	funct3	opcode	B 类指令
imm[15:8]			rd	imm[5:7]	opcode	U 类指令

基础指令集架构

龙腾基础指令集架构是一个自主可控的，拥有完全自主知识产权的指令集，共有 28 条指令，分为四大类：寄存器-寄存器算术逻辑指令；寄存器-立即数算术逻辑指令；高位立即数加载指令；控制和访存指令。

2.1 寄存器-寄存器算术逻辑指令

龙腾指令集的寄存器-寄存器算术逻辑指令都是 R 类指令，共有 8 条，分别为加（ADD），减（SUB），与（AND），或（OR），异或（XOR），算术左移（SLA），算术右移（SRA）和逻辑右移（SRL）。rs1 指定第一个操作数，rs2 指定第二个操作数，rd 指定结果写回寄存器。在减法指令中，rs1 指定被减数，rs2 指定减数；在位移指令中，rs2 指定被移位的数，rs1 指定位移的位数。opcode 为 00，funct2 恒为 00，funct3 选择 8 个操作中的一个，具体对应关系如下表：

000	SLA	算术左移
001	SRA	算术右移
010	SUB	减
011	ADD	加
100	XOR	异或
101	OR	或
110	AND	与
111	SRL	逻辑右移

需要注意的一点是，16 位全是 0 的指令会被解码为一个寄存器-寄存器算术逻辑指令，其意义为将 r0（零寄存器）的内容算术左移 r0（也就是 0）位，结果写回 r0，该指令执行完成以后没有任何效果，除了将 pc（以及相关的性能计数器）加一，即龙腾架构的全 0 指令是 NOP 指令。

2.2 寄存器-立即数算术逻辑指令

龙腾指令集的寄存器-寄存器算术逻辑指令都是 I 类指令，功能与寄存器-寄存器算术逻辑指令相同，rs2 和 rd，funct3 的功能也相同，但是 opcode 为 01。主要的区别在于 rs1 和 funct2 被替换为 5 位立即数 imm[4:0]，该立即数在参与计算之前会被位扩展至 16 位，然后和 rs2 指定的寄存器中的数值进行计算。

2.3 高位立即数加载指令

高位立即数加载指令，即 LUI 指令，是唯一的 U 类指令，其 opcode 为 10。该指令将一个 11 位的立即数加载到由 rd 指定的目标寄存器的高 11 位，由一个 LUI 指令加上一个 ADD 指令即可将任何 16 位立即数加载至通用寄存器中。

2.4 控制和访存指令

龙腾架构的控制和访存指令的 opcode 为 11，控制和访存指令可以进一步划分为有条件跳转指令，无条件跳转指令，访存指令和权限控制指令。有条件跳转（分支）指令是 B 类指令，无条件跳转指令是 I 类，访存指令和权限控制指令的格式都是 R 类，但是对指令编码的使用和寄存器-寄存器算术逻辑指令有细微的不同。

2.4.1 有条件跳转指令

有条件跳转指令有六条，都是 B 类指令，分别是若小于则分支（BLT），无符号若小于则分支（BLTU），若大于则分支（BGT），无符号若大于则分支（BGTU），若等于则分支（BEQ），若不等于则分支（BNE）。有条件跳转指令比较由 rs1 和 rs2 指定的寄存器的内容，并根据结果决定是否跳转到一个相对于 pc 的地址。偏移量由一个 5 位立即数表示，所以有条件跳转指令的范围是当前指令的前后 16 条指令。有条件跳转指令的 funct3 值如下：

000	BLT	若小于则分支
001	BLTU	无符号若小于则分支
010	BGT	若大于则分支
011	BGTU	无符号若大于则分支
100	BEQ	若等于则分支
101	BNE	若不等于则分支

2.4.2 无条件跳转指令

无条件跳转指令只有一条，跳转并链接寄存器（JALR），是 I 类指令。无条件跳转指令将 pc 设置为由 rs2 指定的寄存器的内容，并将指向下一条指令的 pc 存储到由 rd 指定的寄存器中。由 JALR 指令可以构建 CALL RET JMP 等常见无条件调用/返回/跳转指令。无条件跳转的 funct3 为 110。

2.4.3 访存指令

访存指令有两条，分别为加载指令（LOAD）和存储指令（STORE）。加载指令将由 rs2 指定的寄存器中的内容作为地址的存储器数据加载到由 rd 指定的寄存器中。存储指令将由 rs1 指定的寄存器的内容存储到由 rs2 指定的寄存器作为地址的存储器单元中。访存指令的 funct3 为 111，加载指令的 funct2 为 00，存储指令的 funct2 为 01。存储和加载指令各有三位空闲位，称为设置（OPT）位，这三位的功能由具体实现者决定，比如可以用来直接充当地址位的扩展，或者用来选择读写的地址空间，或者用来选择加载 8/16 位数据，选择按字还是字节寻址，是否对齐等等。

2.4.4 权限控制指令

权限控制指令有两条，分别为系统调用指令（SYSCALL）和中断指令（BREAK）其中系统调用指令用来调用系统提供的功能，而中断指令用来进行代码调试，两种指令都将控制权移交给执行环境，两者的主要区别在于中断指令只应使用在调试环境中，正式的程序不应该包含中断指令，除非用来标记常规情况下不可到达的，程序无法处理的错误。权限控制指令的 funct3 为 111，系统调用指令的 funct2 为 10，中断指令的 funct2 为 11，rs1，rs2 和 rd 的作用由具体实现者决定。需要注意到，全 1 指令会被解码为一条 BREAK 指令，这样一来就方便了调试，因为 EEPROM 清空以后默认为 FF（全 1），所以只要程序执行了空存储器（很有可能是跑飞的结果），就可以被调试器捕获。

修改和扩展指令集架构

因为龙腾指令集是拥有自主知识产权的指令集，任何人都可以对龙腾指令集进行扩展和修改，不需要担心收到律师函。同时，由于龙腾指令集的核心目的之一是便于在各种平台上实现，所以龙腾指令集并不规定一个必须兼容的“基本指令集”。即使是前文提到的“基础指令集”，也不是必须兼容的，只是起到一个指导性作用。扩展龙腾指令集的方式主要有以下几类：

3.1 寄存器-寄存器指令扩展

龙腾指令集的寄存器-寄存器指令空间有一些未定义指令，即 funct2 不为 00 的空间，这个空间可以用来加入其他的寄存器-寄存器操作指令，如乘除指令。但是，由于寄存器-立即数指令使用 funct2 段作为立即数的第 4、5 位，所以这部分扩展的指令只能是寄存器-寄存器指令而没有对应的寄存器-立即数指令。

3.2 立即数位移指令扩展

寄存器-立即数指令的编码空间有一小块未定义指令：位移量（立即数）最高位为 1 的指令。由于位移量总是非负且不大于 15 的，所以这部分指令可以用来作为扩展指令的空间。然而，这个扩展空间较为狭小，而且只有三条指令有这个空间。

3.3 暗示（hint）指令扩展

暗示（hint）指令是 RISC 架构中常用的用来提高性能的手段，通过无效果的指令来暗示处理器执行一些优化操作，相对低性能的处理器可以直接将 hint 指令当成 NOP 指令执行。龙腾指令集的暗示指令空间主要集中在目标寄存器为 r0 的算术逻辑指令上。而且，由于龙腾指令集并不强制要求对基础指令集的兼容性，所以龙腾指令集的暗示扩展可以更加复杂，甚至包含会改变处理器状态（比如寄存器值）的“有效”指令，虽然这种操作是不被推荐的。

3.4 系统和中断指令扩展

龙腾指令集的系统调用指令和中断指令都有大量的未定义空间，因为这两个指令本身就是为了实现者定义而存在的。这两个空间可以用来放置协处理器指令，其优势在于如果协处理器不存在，则这些指令会自然被系统或者调试器捕获。然而，不建议在中断指令的空间实现任何不是代表“非正常状态”的指令。

3.5 内存模型扩展

龙腾指令集并不规定一个指定的内存模型，实现者可以自行决定对龙腾指令集的实现是普林斯顿架构还是哈佛架构，按字寻址或者按字节寻址（数据）。这种自由度的典型代表是加载和存储指令中的设置（OPT）位。同时，实现者可以自行为龙腾添加 MMU，以扩展龙腾指令集的寻址空间。

3.6 修改龙腾指令集

龙腾指令集是可以修改的，修改过后的指令集不一定要和龙腾基础指令集兼容。比如，一个可能的修改是简化位移指令为只位移一位，这样一来指令集的实现会大大简化。然而，为了避免造成疑惑，如果实现者对指令集进行了影响兼容性的修改，则高度推荐实现者详细阐明其修改。

3.7 龙腾标准执行环境

龙腾指令集虽然不规定一个必须兼容的基本指令集，但是，规定了一套标准化的环境，即龙腾标准执行环境。不完全兼容龙腾基础指令集的实现，仍然可以被称为“基于龙腾”的处理器实现，但是如果不兼容龙腾标准执行环境的实现，无论是硬件还是软件，都不能称为兼容龙腾标准执行环境。为龙腾标准执行环境编写的程序，可以不加修改地在所有兼容该环境的实现上运行。关于该执行环境的详细信息，请参考《龙腾标准执行环境参考手册》。注：该手册还在编写当中