**计算机系统结构研究报告**

——RISC-V技术发展和性能特点

课号：42036804(周四7.8节) 组号：第12小组

作者信息与任务分配比例：

2253744林觉凯(40%)：

jklin7918@gmail.com；引言、RISC-V指令集设计和总结部分及其PPT

2253909张宏晔(30%)：

zhy454@tongji.edu.cn；RISC-V的架构设计及其PPT

2253922丁宇轩(30%)：

2253922@tongji.edu.cn；RISC-V性能分析与优化及其PPT

**摘要：**随着嵌入式系统、物联网和高性能计算等领域的发展，对灵活、可扩展且高效的指令集架构的需求不断增加。现有的商业指令集架构(如ARM和x86)虽然功能强大，但其封闭性和高昂的授权费用限制了其一定发展。加州大学伯克利分校在2010年启动了RISC-V项目，目的是为学术研究和教学提供一个开放的、可自由使用和修改的指令集架构。本文围绕RISC-V技术发展和性能特点，分别介绍RISC-V指令集设计、架构设计和性能对比结果，旨在深入了解RISC-V这一处理器架构体系，同时在学习的过程中巩固对课内知识的掌握。

**一、引言**

**1.1.RISC-V简介**

随着物联网、人工智能和高性能计算等新兴领域的发展，传统ISA(指令集架构，Instruction Set Architecture)面临新的挑战。传统ISA比如ARM和X86架构虽然在计算领域具有显著的优势和广泛的应用，但是由于其架构的复杂性(指令集复杂且庞大、高性能处理器复杂)、开发和使用成本较高、开放性与自由低(限制了开发者的自由和灵活性)和模块化和可扩展性等不足之处，使得计算机领域急需要一种开源、灵活和自由的指令集架构。

于是在2010年，加州大学伯克利分校的一个研究团队在设计一款CPU、选择项目选架构的时候，发现像ARM、MIPS、和X86等指令集不仅结构复杂，还难以获取(比如X86授权难以获取，ARM价格昂贵，还涉及到一些法律问题)，于是该研究团队最终决定设计一套全新的指令集。经过三个多月的开发工作，RISC-V指令集就这样诞生了。

RISC-V处理器在过去十来年间通过其开源、灵活和高效的设计，已经在全球计算领域取得了显著的发展。随着技术的不断进步和社区的持续努力，RISC-V有望在更多领域实现突破，成为计算领域的重要支柱。

**1.2.研究目的和意义**

本次研究报告主要分析了RISC-V处理器的指令集设计，学习了其基础指令、扩展指令和权限模式，同时从微架构(比如流水线设计、缓存架构、分支预测等)和多核和并行处理等多个方面介绍RISC-V的架构设计，最后通过与ARM、X86等经典架构进行对比，对RISC-V处理器的性能做出分析，得出最后的结论。

研究RISC-V处理器的原理和性能，有助于我们理解当前一个完全开源的指令集架构，RISC-V的简洁和模块化设计使其易于我们初学者理解、实现和扩展。本次研究将巩固我们在书上学到的理论知识并且学会分工协作和互相学习。

**二、RISC-V指令集设计**

**2.1.RISC-V基础指令集**

RISC-V基础指令集设计简洁、模块化，涵盖了一系列必需的指令，以确保处理器的基本功能，同时允许根据需要进行扩展。RISC-V基本指令集有RV32I、RV64I、RV32E、RV128I, 分别代表了32位整数指令集、64位整数指令集、32位嵌入式整数指令集、128位整数指令集。其中，RV32I和RV64I是最主要的两种, 分别针对32位和64位工作环境而设计。

**(一)RV32I指令集**

RV32I是RISC-V指令集架构的基础，可以说是RISC-V指令集中最重要的指令。它包含了一组核心指令，确保处理器能够执行基本的计算任务。

如图2.1，RV32I指令集使用32个通用寄存器(x寄存器)和一个额外的非特权寄存器(pc寄存器)。寄存器的位宽XLEN由指令集设计的位数决定，比如RV32I的位数是32位，RV64I的位数是64位，RV128I的位数是128位，因此它们在结构上相似。pc寄存器用户可见，它保存了当前指令的地址。

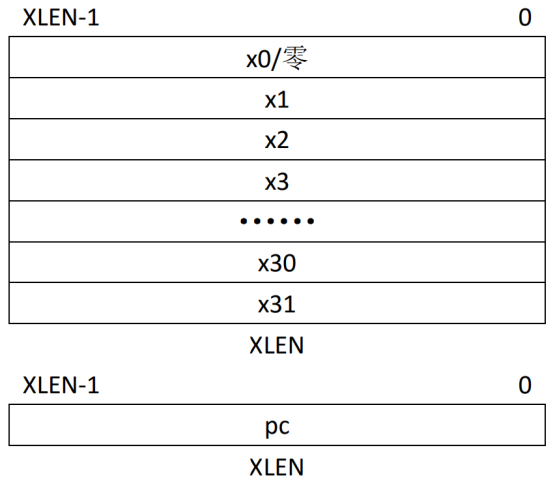
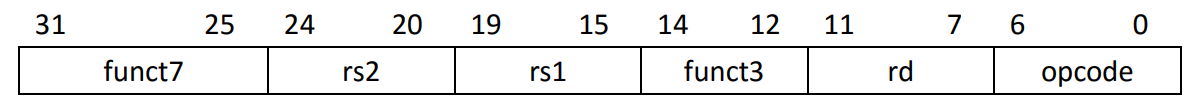


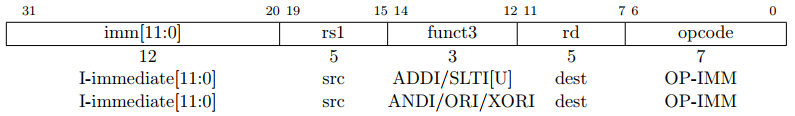
图2.1 RV32I使用的寄存器

RV32I指令集有六种基本的指令格式，分别是R类、I类、S类、U类、B类和J类指令格式，分别用于寄存器-寄存器操作、短立即数-load操作、store操作、长立即数操作、条件跳转操作和无条件跳转操作。固定长度为32位。

32位的R类指令由6个字段组成，opcode是操作码，分布在第0-6bit位上，占7位；rd指的是目标寄存器的地址，分布在第7-11bit位上，占5位；rs1和rs2代表着两个源操作数寄存器，分别占5位，rs1占第15-19bit位，rs2占第20-24bit位；funct3和funct7代表两个操作字段，其中funct3占了3bit，在指令格式的第12-14bit位上；funct7占了7bit，在指令格式的第25-31bit位上。

图2.2 RV32I中R类指令格式

32位的I类型指令由5个字段组成，其中rs1、funct3、rd和opcode与上述的R类指令相同，由于I类指令为有关短立即数-load的操作，所以需要存放12位的立即数——imm[11:0]，它在指令格式的20-31bit位上。

图2.2 RV32I中I类指令格式

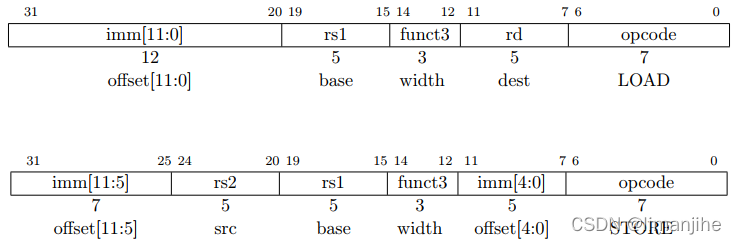
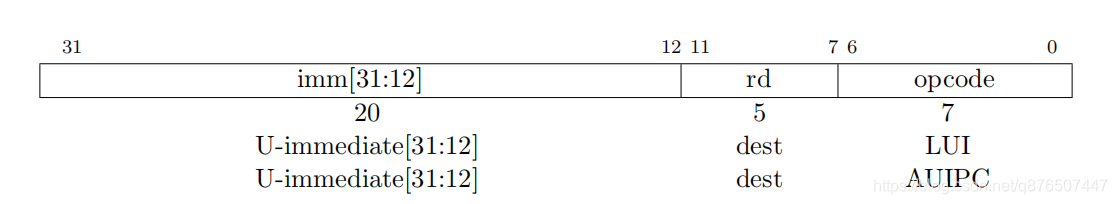
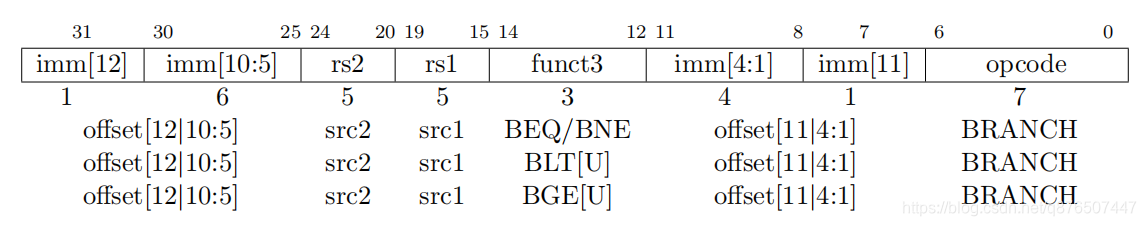
32位的S类型指令由5个字段组成：其中opcode、rs1、rs2和funct3与上述的R类指令相同。RV32I指令集中的Store指令用于将数据从寄存器存储到内存中。因而将立即数拆分为分成两个：存放前5位的立即数——imm[4:0],在指令格式的7-11bit位上；存放后7位立即数——imm[11:5]，在指令格式的25-31bit位上。将立即数高七位和低五位地存放，可以避免了为每个指令都分配一个固定长度的立即数字段，从而节省了指令编码的位数。

图2.3 RV32I中S类指令格式

32位的U类型指令常用于立即数扩展。imm[31:12]是一个20位的立即数字段，它被用来扩展PC的高20位。这rd 是目标寄存器的编号，它指定了结果应该存储在哪个寄存器中。在RV32I中，有32个通用寄存器，因此需要5位来表示这些寄存器的编号。opcode是操作码字段，用于指定具体的操作。

图2.4 RV32I中U类指令格式

B类指令用于条件跳转，我们把重点放在跳转条件和分析上。funct3为功能码字段，用于指定具体的条件分支操作。funct3为BEQ和BNE时分别表示当rs1和rs2相等和不相等时，跳转否则就不跳；funct3为BLT和BLTU是有符号数比较和无符号数比较，当rs1<rs2时候就跳，否则不跳；Funct3为BGE和BGEU是有符号数比较和无符号数比较，当rs1>=rs2时候就跳，否则不跳。

图2.5 RV32I中B类指令格式

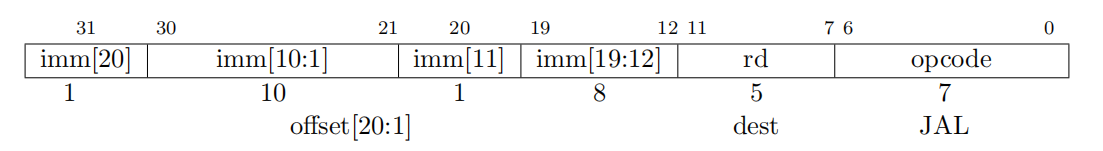
J类指令用于非条件跳转，我们上课中也学习到，跳转指令需要跳转并连接(Jump and Link)。opcode操作码中的操作最常见的便是“JAL”。JAL将跳转指令后面指令的地址(pc+4)保存到寄存器rd中，形成跳转目标地址。

图2.6 RV32I中J类指令格式

**(二)其他基础指令集**

RV32E、RV64I和RV128I也是RISC-V的基础指令集，它们都是RV32I的变体，因此与上述介绍的RV32I差别不大。

RV32E使用与RV32I相同的指令集编码，它是RV32I为嵌入式系统而设计的简化版本。RV32E针对对资源有限的嵌入式系统设计。它通常包含较少的通用寄存器和简化的指令集，以降低硬件成本和功耗。

RV32E和RV32I的差别就是在通用寄存器的数量上，在RV32I中，总共有32个32位宽的通用寄存器，而RV32E只支持16个32位宽的通用寄存器。它包含较少的通用寄存器和简化的指令集，以降低硬件成本和功耗。

RV64I、RV128I与RV32I指令集的区别并不大，分别适用于64位和128位架构的处理器，它们具有更大的通用寄存器和地址空间，可用于处理需要更大内存空间和更高精度计算的应用场景，支持更多的指令集扩展。

**2.2.RISC-V扩展指令集**

RISC-V架构是一个模块化的指令集架构，允许通过添加扩展指令集来增强其功能。这些扩展指令集提供了额外的指令，以满足不同应用领域的需求。现有的RISC-V扩展指令集主要有二十多种，重要的扩展指令集有如下四种。

乘法拓展(M扩展)，它提供了对整数和浮点数的乘法和除法运算的支持。M扩展引入了一组用于执行整数乘法的指令，例如 MUL(乘法)、MULH(乘法高位、MULHSU(有符号-无符号乘法)、MULHU(无符号-无符号乘法)等。这些指令用于在整数数据之间执行乘法运算，并且可以处理不同精度的整数数据，包括标准整数、扩展整数和固定点数等。M扩展还引入了一组用于执行整数除法的指令，例如 DIV(除法)和 REM(求余)。这些指令用于执行整数之间的除法运算，并且可以处理有符号和无符号整数数据。

原子拓展(A扩展)，A扩展引入了一组原子操作指令，用于在多线程并发环境中确保对共享内存的安全访问。比如加载-存储原子操作指令，Load-Reserved指令用于从内存中加载数据到寄存器，并将内存地址标记为“保留”，表示该地址正在被当前线程使用；Store-Conditional指令用于将数据从寄存器存储回内存，并在存储时检查内存地址是否仍然处于“保留”状态。

单精度浮点扩展(F扩展)，F扩展提供了一组指令用于执行单精度(32位)浮点数的运算。比如加载和存储单精度浮点数、单精度浮点数的算数运算和单精度浮点数的比较运算等等。这些指令允许RISC-V处理器高效地执行单精度浮点运算，并提供了丰富的操作类型以支持各种计算需求。

双精度浮点扩展(D扩展)，D扩展提供了一组指令用于执行双精度(64位)浮点数的运算。比如加载和存储双精度浮点数、双精度浮点数的算数运算和双精度浮点数的比较运算等等。些指令允许RISC-V处理器高效地执行双精度浮点运算，这对于需要高精度和高性能浮点计算的应用场景具有重要意义。

**2.3.RISC-V权限模式**

**(一)RISC-V的三种特权模式**

根据RISC-V特权架构规定，RISC-V的指令集必须工作在一种确定的权限模式下。根据RISC-V权限规范, 目前共有以下4种权限模式: 机器模式(Machine，M模式)、用户模式(User，U模式)、管理模式(Supervisor，S模式) 、监视模式(Hypervisor,H模式)。其中监视模式是基于管理模式的虚拟化扩展，暂处于草案状态，我们目前不考虑，仅探讨前三种特权模式。

用户模式是为了运行用户应用程序而设计。它的权限有限制访问关键系统资源和指令，以确保系统的安全性和稳定性。用户模式下的程序无法直接访问硬件或关键系统资源，需通过系统调用与操作系统进行交互。用户模式的编码为00。

管理模式是为了操作系统内核或其他特权级别的软件运行而设计。它的权限有能够访问用户模式无法访问的硬件资源和指令。通常用于操作系统的核心功能，包括内存管理、设备控制和任务调度。管理模式程序可以直接访问物理内存并管理用户进程的虚拟内存。管理模式的编码为01。

机器模式是为了最低级别的硬件控制和初始化运行环境而设计。它拥有最高权限，可以访问所有的硬件资源和指令。通常用于引导加载程序、硬件驱动程序和其他需要完全访问硬件的功能。机器模式程序负责系统的启动、硬件初始化和模式切换。机器模式的编码为11。

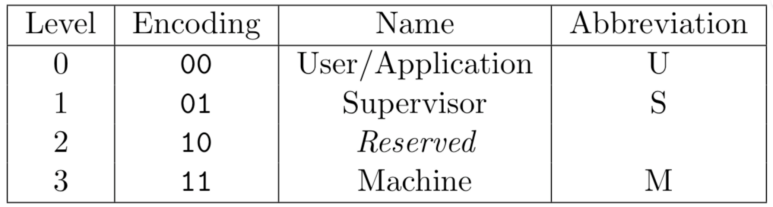


图2.5 RISC-V中三种特权模式

**(二)RISC-V的特权模式的转化**

机器模式是RISC-V指令集架构中最高级别的权限模式，在这种模式下具有执行任何机器操作的权限，也是在系统设计中必须实现的一个工作模式，提供裸机执行环境。用户模式是特权架构里最低级别的权限模式，通常用于执行来自用户等外部环境的不可信操作，实现用户和系统之间的隔离。管理模式具有比用户模式更高的权限，它可以设置相应组件来实现操作系统的使用，调度多个应用程序。只有在响应异常或异常返回时才会发生特权级的切换。

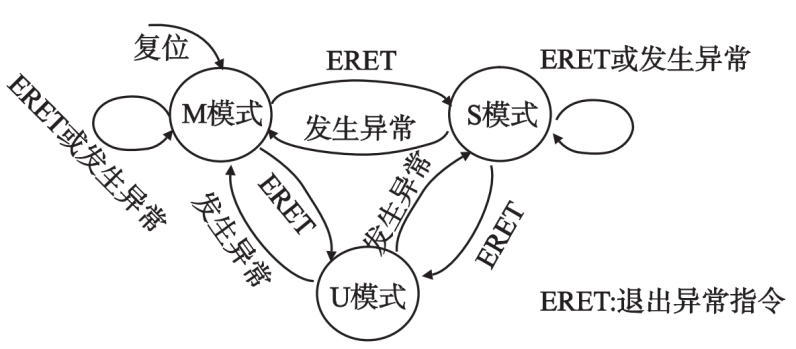


图2.6 RISC-V中三种特权模式的转化条件

同时，我们也可以使用特定的模式转化的指令来实现RISC-V中的特权模式转化。mret是从机器模式返回到先前的模式的指令，当操作系统内核完成了必要的特权操作后，它可以使用mret指令返回到先前的模式，通常是用户模式和管理模式；sret是从管理模式返回到先前的模式的指令。当内核完成了特权操作后，它可以使用sret指令返回到先前的模式，通常是用户模式；uret是从用户模式返回到先前的模式的指令。当应用程序执行需要特权的操作时，它可能会切换到更高特权级别的模式，完成操作后再使用uret指令返回到用户模式。

**三、RISC-V的架构设计**

**3.1.RISC-V的流水线设计**

**(一)RISC-V应用流水线的优点**

流水线是一种在执行过程中将多条指令互相重叠，从而并行执行指令，提高处理器吞吐量，从而提高处理器运行速度的技术。RISC-V指令集本身的设计非常契合流水线技术，这可以从以下三点看出：

1.所有RISC-V指令都具有相同的长度(32位)，且每个指令都是定长的。这个约束使得取指和译码的操作相当容易。

2.RISC-V指令集只有少数的几种指令格式，且指令中的源寄存器和目标寄存器都位于指令中相同的位置。

3.RISC-V中采用的是加载、存储架构，对存储器的访问只出现在load和store指令中。

这意味着我们可以在执行阶段计算内存地址并且在下个周期再访问存储器。

**(二)RISC-V流水线中的冲突**

和其他流水线一样，RISC-V流水线也存在结构冲突、数据冲突和控制冲突。

RISC-V指令集解决结构冲突是很简单的。得益于其简洁的指令结构设计，仅有译码阶段和写回阶段存在结构冲突的隐患，因为其同时使用了通用寄存器组。但是，通过简单地将寄存器组的读口和写口分离即可解决这一问题。

RISC-V指令集解决数据冲突和控制冲突的方式较为常规，大致有硬件阻塞、软件NOP、旁路、分支预测等技术。

**3.2.RISC-V的处理器设计**

**(一)处理器设计概要**

处理器的设计要考虑到许多问题：

1.该处理器所支持的指令集：由于RISC-V能支持扩展指令，所以支持的指令集也会影响处理器的细节设计。

2.组成指令执行通路的组件：如ALU、PC、REG、Cache、Mem等。

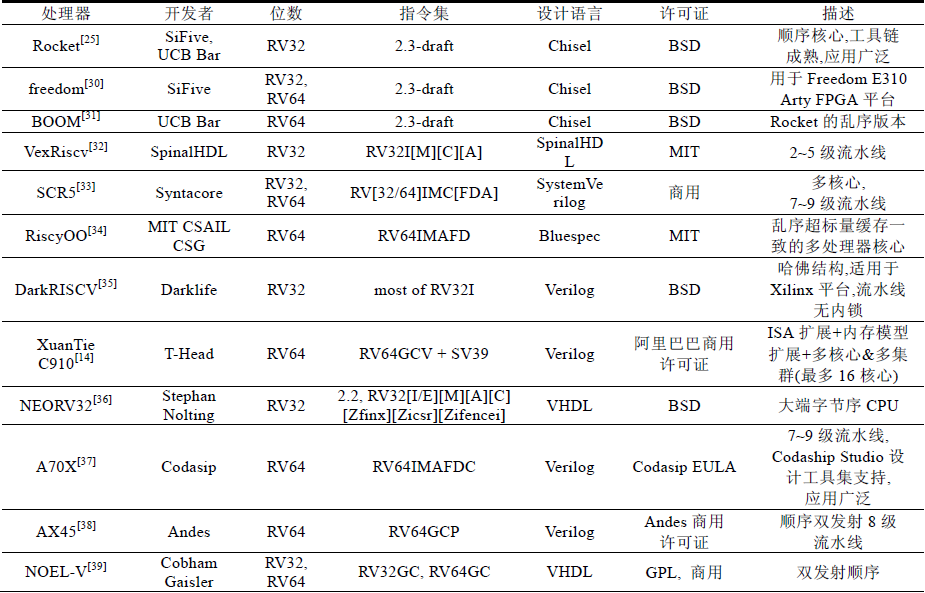
3.指令的处理方式：包括控制信号、时钟周期、流水线、任务调度等问题。

4.多核心处理器：需要额外考虑核心结构、核间通信等设计。

RISC-V社区在2021年的统计中，已经公开发布并提交到社区的RISC-V处理器共计107种。由于RISC-V的模块化可扩展设计，基于其设计出的处理器也各具特色。接下来分别介绍通用RISC-V处理器和嵌入式RISC-V处理器种较有代表性的一些处理器。

**(二)通用RISC-V处理器**

Rocket、freedom、Boom等处理器支持更全面的指令集，这使得他们能对接各类系统，拥有最为广泛的适用性；A70X自带工具集，能方便开发。SCR5、RiscyOO、XuanTie910能在性能方面上取得更大优势；DarkRISCV、NEORV32、AX45、NEOL-V等则更加专注于特定结构，能够相应地满足各自特殊的设计需求。

表3.1 12种典型的通用RISC-V处理器

**(三)嵌入式RISC-V处理器**

嵌入式处理器的设计关键在于对有限资源的利用能力，或对于资源受限环境的适应性。

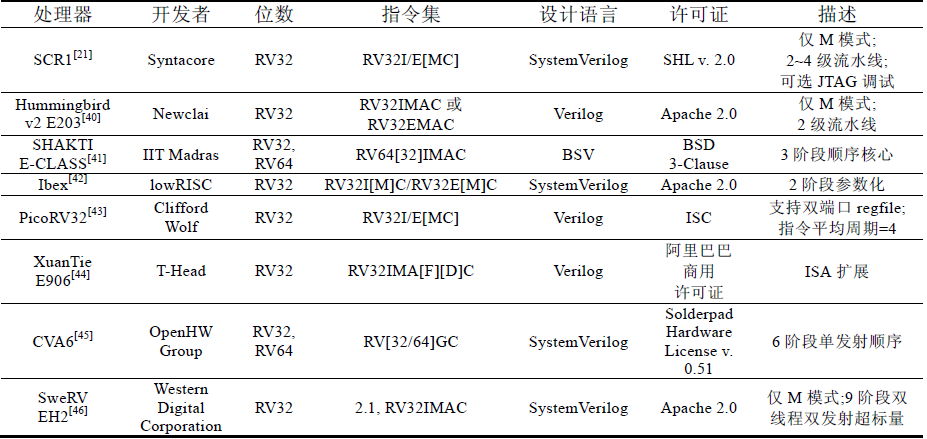


表1.2 8种典型的嵌入式RISC-V处理器

SCR1和SweRV EH2仅含M模式，可提供最基本的处理能力，相对适应性最强；Hummingbirdv2 E203、SHAKTI E-CLASS、Ibex可适用于常规嵌入式环境、PicoRV32、XuanTie E906、CVA6处理能力更高，能够承担有更高能力要求的环境中的数据处理任务。

**3.3.RISC-V的多处理器系统**

多处理器系统比起单处理器系统能显著提高性能，但也需要关心处理器核心增加带来的核间管理和通信问题，关注数据的并行或并发处理能力。具体来说，多处理器系统中可能存在如下挑战：

1.存储一致性与缓存共享问题。

2.RTL电路大幅增长带来的复杂性问题。

3.组件依赖性问题。

4.通信延迟问题。

**(一)存储一致性问题和RTL电路问题**

存储一致性是指在读操作时应该返回对该存储单元最近一次写操作的值，多核处理器可能会导致执行程序时的不确定性，造成存储一致性问题。

对于存储一致性问题和RTL电路问题，2017年，Manerkar等人介绍了一种用于缩小微架构/RTL存储一致性模型验证差距的方法和工具——RTLCheck。其给定了一组关于存储一致性模型行为的微架构公理、一个用于弥补RTL设计与用户之间差距的映射，并自动生成SystemVerilog断言，来验证其是否满足指定测试程序的微架构规范。

**(二)组件依赖性问题**

对于组件依赖性问题，2020年Kruth等人提出了一个模块化、高效的原子单元(ATUN)架构，它可以在不同的内存结构层次中灵活实施，使任何级别的内存层次结构都可以实现可扩展的原子操作(AMO)。

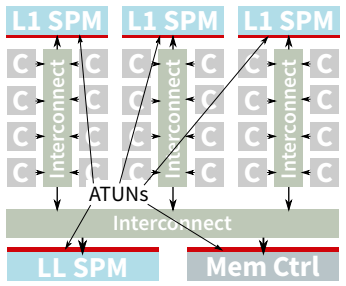


图3.2 在基于SPM的存储器层次结构中，利用 ATUN(红色横条)实现 AMO。

**(三)通信延迟问题**

通信是连接系统各组件、各成分之间的信息交换过程，通信延迟将推迟目标获得所需信息的时间，从而增大其空闲等待时间，造成总体用时延长、目标利用率降低、或者能量空耗。

Morais等人在2019年提出了一种将任务调度硬件加速器与通用CPU紧密集成的体系结构，以减少通信延迟及运行开销，从而提高与任务调度并行的应用程序的性能。研究团队开发了硬件加速的轻量级任务调度运行时环境Phentos，允许任务调度软件通过自定义指令与硬件任务调度器和CPU直接进行交互，以最大限度减少硬件任务调度器和CPU之间的通信开销。

图3.3 以 RocketChip 等效周期为单位的多个平台的任务调度寿命开销。

**四、RISC-V性能分析与优化**

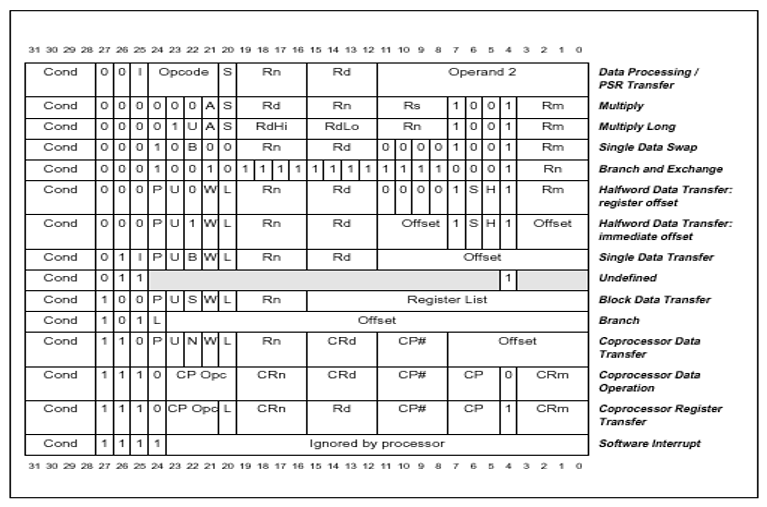
**4.1.性能比较分析**

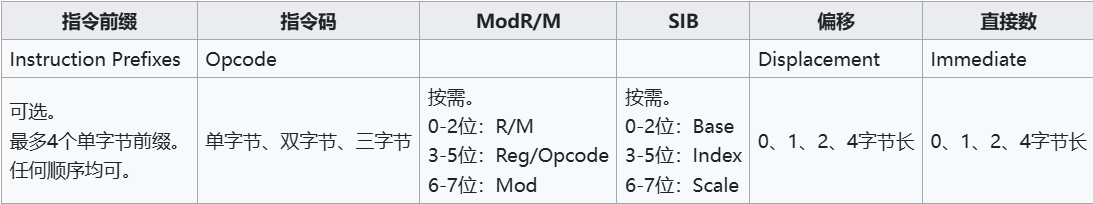
**(一)指令集编码**

RISC-V是精简指令集架构，具有简洁清晰的设计，采用固定长度的指令格式。RISC-V的指令格式通常由32位或64位组成，指令长度一致，便于解析和执行。RISC-V还提供了扩展指令集可以根据应用需求灵活扩展。

ARM也是精简指令集架构，具有较为复杂的指令格式和丰富的指令集。ARM 指令格式包括条件码、操作码、寄存器标识符等多个字段，长度通常为32 位或64位。ARM 还提供了Thumb指令集，在armv7中长度为16位和32位，牺牲功能性，提供了更高密度的指令编码，以节省存储空间和功耗。

x86是复杂指令集架构，具有非常复杂的指令格式和丰富的指令集。x86指令格式通常包括操作码、寄存器标识符、内存地址等多个字段，具有变长指令，指令长度可能是若干字节。x86 庞大的指令集支持复杂的内存寻址模式和操作。

 图4.1 ARM指令编码格式

图4.2 x86指令编码格式

综上所述，RISC-V具有相比之下比较精简的指令集，减少了处理器设计和开发的复杂度，可以更好地进行优化，减少指令的执行时间和功耗，从而提高处理器的性能和效率。精简的指令集可以提供更快的解码速度，相比于需要解码才能知道指令长度的边长指令集而言。精简指令集的指令功能更加基本，更加原子，容易实现指令级并行。另一方面，精简指令集对编译技术提出了更高的要求，同时也给了编译器更多程序优化的机会，相比于复杂指令集向编译器隐藏了复杂指令的实现细节，其更难再被优化。

**(二)流水线**

RISC-V通常采用经典的五级流水线设计，包括取指(IF)、译码(ID)、执行(EX)、访存(MEM)和写回(WB)五个阶段，设计简洁明了，便于实现和优化。由于RISC-V的开源，众多开发者正在以其为基础尝试开发更高性能，或有特殊优点的处理器，包括SiFive的商用系列，Berkeley Out-of-Order Machine (BOOM)项目，Alibaba T-Head的芯片，加州大学伯克利分校的Rocket Chip等。

ARM 处理器的流水线设计根据具体的处理器系列和型号而有所不同。ARM7处理器核使用了典型三级流水线的冯·诺伊曼结构，具有基本的 FETCH-DECODE-EXECUTE(F&E)循环；ARM9系列则采用了基于五级流水线的哈佛结构，包括取指、解码、执行、LS1、LS2阶段；ARM10 ARM11引入更多流水线段数；之后的Cortex系列采用了ARMv7 v8架构，已经与之前的落后的架构不同，引入了超标量流水线，乱序执行单元等先进技术，但是仍然考虑了向下兼容，尽管其严格不如x86，保留了一定的灵活性。随着流水线级数增加，吞吐量和效率会增加。

x86的流水线设计由于其指令集复杂度更加复杂，但是基本模式还是译码、执行和退出三个阶段。i486首次使用流水线技术，采用了五级流水线；Pentium处理器添加了一条独立的超标量流水线用来并行运行一些较简单的指令，比如定点算术，而且该流水线能更快的进行该运算；Pentium Pro采用了诸多新特性以提高性能，包括乱序执行的部件以及猜测执行，流水线扩展到了12级；Pentium4引入了超线程技术，加入了第二套前端部件(指取指，译码，寄存器重命名等模块)，充分利用乱序执行部件的性能，在操作系统看来是两个处理器；之后更新的处理器也使用了复杂先进的技术。

综上所述，RISC-V处理器的流水线设计简单，流水线利用率好，其轻量级低功耗小尺寸特点和开源使其更多地用于嵌入式处理器和特定设备。ARM也由于其轻量级常常用于移动设备或嵌入式，比如物联网设备等。x86则由于其复杂性和个人电脑市场提出的性能需求，经过多年发展和优化，使用的复杂流水线技术同样有一定的优势。同时，由于RISC-V的开源和非商业化，吸引了全球范围内的大量开发者和公司参与，带来了高度的灵活性和多样化的发展路径，与ARM和x86需要考虑向下兼容的商业化需求不同。

**4.2.RISC-V性能优化技术**

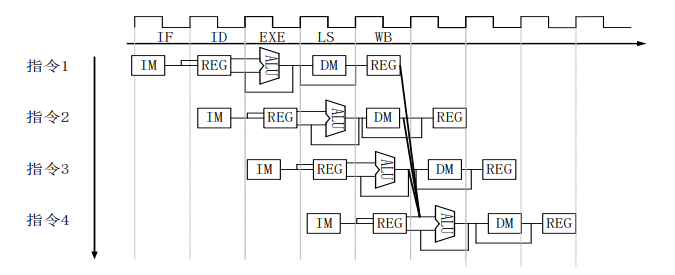
这一部分通过浏览有关文献研究，列举了一些RISC-V性能优化的技术。

**(一)分支跳转指令**

通常在流水线中，取指阶段以及译码阶段无法对有条件跳转指令做出判断，决定是否跳转，这会产生较长时间延迟，影响处理器性能。将分支指令的处理提前到译码阶段，可以降低处理分支指令产生的延迟对后续指令的影响。

**(二)流水线冒险处理**

流水线冒险包括结构冒险、数据冒险、控制冒险。RISC-V经典五级流水线会产生的数据冲突只可能是写后读(RAW)冲突，也叫做真相关。发生真相关的指令必须等到前一条指令执行完成才能进行后续操作。可以采用数据重定向技术解决此问题。通过数据旁路，将前一条指令产生的中间数据直接定向到后面数据的ALU输入端。

图4.3 使用数据旁路解决数据冲突

乱序执行技术能够在不改变程序结果的前提下，以不同于程序原始顺序的方式执行指令，这种方法通过重排指令能够避开数据冒险，从而增加处理器性能。是x86架构的核心技术，RISC-V使用这种技术的有Berkeley Out-of-Order Machine (BOOM)，加州大学伯克利分校开发的一款基于RISC-V的乱序执行处理器，旨在研究高性能处理器的设计；以及SiFive U系列高端处理器等商用RISC-V处理器。然而实现这种技术需要的成本也会增加，对于用于嵌入式系统的轻量级RISC-V乱序执行并不常见。

**(二)流语义寄存器**

在冯诺依曼体系的计算机系统中，指令与数据存放在一起，这导致取指令和取数据不能同时进行，CPU在执行运算前后,都需要额外的时间等待数据完成存取,而不能一直处于工作状态，从而导致冯诺依曼瓶颈。针对此问题，2021年，Schuiki等人提出了流语义寄存器拓展，关键思想是允许在任何指令中对加载和存储进行编码，以便进行常规数据访问的指令序列，而不是显式的加载/存储指令。通过在寄存器文件中截获对某些寄存器的访问，并将这些访问从内核路由到内存系统，将特定寄存器读取和写入的操作将从处理器中转移出来。

**(三)多线程缓存优化**

多核处理器中，随着芯片上核心数量的增加，会出现“缓存行乒乓”问题，当多个CPU共享同一缓存行中的变量时，不同CPU对该变量的频繁读写会导致其他CPU的缓存行频繁失效。2019年，Dogan等人提出移动计算模型(MC)来加速线程同步。在 MC模型中，共享数据在逻辑上固定到称为服务线程的专用线程。工作线程执行应用程序代码并调用请求以更新服务线程上的共享数据。该模型能够解决缓存行乒乓的问题。

**五、总结与思考**

RISC-V作为一种开源的指令集架构，为计算机架构领域带来了新的活力和无限可能。它的基础指令构成了处理器执行基本任务所需的最小集合，确保了基本的计算功能和控制流操作；扩展指令增加特定功能和优化性能，以满足不同应用需求；三种特权模式提供了分层的安全机制，确保系统的稳定性和安全性。RISC-V架构设计简介，处理器设计通过灵活的ISA选择、优化的微架构设计、关键组件的高效实现和实现了高效、灵活和可扩展的处理器架构。RISC-V具有相比ARM、X86有着比较精简的指令集，减少了处理器设计和开发的复杂度；通过一系列技术优化性能。RISC-V凭借其开放性、灵活性和强大的技术基础，在未来将成为计算机技术发展的重要驱动力，带来更多创新和变革。

**六、参考文献**

[1]刘畅,武延军,吴敬征,赵琛.RISC-V指令集架构研究综述[J].软件学报,2021,32(12):3992-4024.

[2] F. Schuiki, F. Zaruba, T. Hoefler and L. Benini, "Stream Semantic Registers: A Lightweight RISC-V ISA Extension Achieving Full Compute Utilization in Single-Issue Cores," in IEEE Transactions on Computers, vol. 70, no. 2, pp. 212-227, 1 Feb. 2021, doi: 10.1109/TC.2020.2987314.

[3] Halit Dogan, Masab Ahmad, Brian Kahne, and Omer Khan. 2019. Accelerating Synchronization Using Moving Compute to Data Model at 1,000-core Multicore Scale. ACM Trans. Archit. Code Optim. 16, 1, Article 4 (February 2019), 27 pages. https://doi.org/10.1145/3300208

[4] [基于RISC-V架构-五级流水线CPU - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/453232311)

[5] [多处理机系统——存储一致性模型(consistency)-CSDN博客](https://blog.csdn.net/qq_39815222/article/details/107029271)

[6] 李金凤,于德明,郭瑞华等.基于RISC-V指令集的处理器及其运行环境设计[J]南方论坛,辽宁沈阳2023.8:34-39

[7] 《基于FPGA与RISC-V的嵌入式系统设计》

[8] [基于RISC-V的CPU设计入门——流水线概览|SunnyChen的小窝](https://www.sunnychen.top/archives/rvintropipeline)

[9] 闫润,黄立波,成元虎,杨凌,兰孟桥,张京等.ＲISC-V特权架构配置的硬件实现影响研究[A].国防科技大学计算机学院.2024.4:1019-1024

[10] RISC-V指令集手册[riscv-spec-v2.1中文版.pdf (whycan.com)](http://file.whycan.com/files/members/7090/riscv-spec-v2.1%e4%b8%ad%e6%96%87%e7%89%88.pdf)

[11] 吴安龙 支持浮点扩展指令的RISC-V处理器的研究[A] 东华大学 2023.5

[12] [适合新手的RISC-V指令详解 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/578816042)