上海开放处理器产业
创新中心

《RISC-V导论：设计与实践》

第一模块

Chapter 1: RISC-V前世今生

- CPU与CPU ISA的发展历史、Trade offs
- 从CISC到RISC，再到Berkely RISC
- RISC-V，重点讲开放生态、灵活性、与先进计算的融合

Chapter 2: RISC-V的指令级系统

- ISA基本指令功能
- 指令编码、汇编语言
- 指令设计及扩展

Chapter 3: RISC-V的数据通路设计

- 基础指令的数据通路
- 单周期、perf./cpi、流水线
- 非理想流水线问题与解决方法

完成实验一的设计与材料准备

- RISC-V简易流水处理器设计实战

Chapter 1

▲教学目标

- ▶了解RISC发展简史与发展趋势，掌握指令集定义、特点、需求与相关概念，了解我国处理器设计发展情况

▲教学内容

- ▶RISC vs CISC
- ▶RISC-V为什么诞生与发展历程
- ▶“开源”与“开放”的区别

▲学成效果

- ▲掌握指令集相关基础概念与定义，了解常见指令集，了解指令集发展趋势，能够区别“开源”与“开放”

Chapter 1

▲参考文献

- Patterson, David, and Andrew Waterman. *The RISC-V Reader: an open architecture Atlas*. Strawberry Canyon, 2017.
- Saidova, Jasmina. “RISC-V Architecture and its Role in the Near Future.” *Journal of Advanced Scientific Research (ISSN: 0976-9595)* 5.9 (2024).
- 胡振波等，手把手教你设计 CPU—RISC-V 处理器，人民邮电出版社，2018

▲研究课题

- RISC-V指令集扩展方法与硬件设计
- 指令集对于处理器性能、能效、设计代价的影响

Chapter 2

- 教学目标（Teaching goals）
  - 基础目标为掌握RISC-V指令集基础知识，了解计算机基本工作模型；
  - 中阶目标实现举一反三，给定指令集手册，可以对其他指令集手册中内容进行解读；
  - 高阶目标为设计自定义指令集，实现定制化功能拓展，并能在模拟器中实现相应功能；
- 教学内容 (Contents)
  - 建议教学从RV32I基础指令内容及架构规范及指令编码出发，可选教学内容为指令集拓展及RISC-V指令集发展前沿；

Chapter 2

- 学习内容 (Topics)
  - 了解RISC-V指令集基础
    - RISC-V处理器基本设置
    - RISC-V基础指令的功能
    - RISC-V基础指令的编码
    - RISC-V指令拓展
- 学成效果 (Learning Outcomes)
  - 能够回答以下问题
    - RISC-V指令集架构除指令集外还规范了RISC-V处理器的哪些设置？
    - 如何将一段C语言代码转换为等效的RISC-V基础指令代码？
    - RISC-V基础指令编码设计中有哪些巧思？为什么要这样设计编码？
    - RISC-V基础指令有哪些重要的指令拓展？它们都实现了什么功能？
  - 了解并熟练使用RISC-V汇编语言编程；了解RISC-V指令与其机器码之间的转换

Chapter 2

▲参考文献：

- ▶1、RISC-V foundation. RISC-V Technical Specifications (Updated May 12 2025) RISC-V Technical Specifications. Accessed June 5 2025.
- ▶2、Patterson, David A. and Hennessy, John L. Computer Organization and Design RISC-V Edition: The Hardware Software Interface. Morgan Kaufmann Publishers Inc.. 2017. ISBN:978-0-12-812275-4.

▲ 研究课题

- ▶ 1、RISC-V的自定义指令拓展及指令设计

Chapter 3

▲教学目标（Teaching goals）

- ▶教学目标：掌握RISC-V基础指令在数据通路中的执行逻辑，理解流水线设计原理及其性能影响因素。

▲教学内容 (Contents)

- ▶以ld/sd（访存）、add/sub/and/or（ALU运算）、beq（分支）三类指令为例，解析单周期/流水线数据通路结构、控制信号生成机制、流水线冒险（结构/数据/控制）的成因与缓解策略。建议教学从RISC-V出发

◆学成效果：

- ▶了解：微架构核心组件（PC、寄存器堆、ALU、存储器接口）的功能与互连关系；
- ▶理解：指令在数据通路中的分阶段执行过程（取指、译码、执行、访存、写回）；
- ▶实现：绘制支持ld/sd/add/beq的5级流水线数据通路图，标注关键控制信号；
- ▶评估：量化流水线冒险对CPI（Cycle Per Instruction）的影响，对比阻塞（Stall）与转发（Forwarding）策略的性能差异。

Chapter 3

▲参考文献

[1] 胡振波.手把手教你设计CPU. RISC-V处理器篇[M].人民邮电出版社,2018.

[2] 姚永斌.超标量处理器设计[M].清华大学出版社,2014.

[3] Harris S, Harris D. Digital Design and Computer Architecture, RISC-V Edition[M]. Morgan Kaufmann, 2021.

[4] Hennessy J L, Patterson D A. Computer architecture: a quantitative approach[M]. Elsevier, 2011.

[5] 魏继增,郭炜.计算机系统设计. 上册, 基于FPGA的RISC处理器设计与实现[M].电子工业出版社,2019.

◆建议实验

- ▶基于E203中的处理器设计实践
- ▶蜂鸟E203中的数据冒险处理

第二模块 基于RISC-V的SoC硬件架构

Chapter 4 RISC-V SoC体系架构

- SoC基本概念及组成
  - SoC的基本概念
  - SoC的基本架构组成
- SoC总线
  - SoC总线的基本概念
  - SoC总线的类别
- SoC常见IP综述
  - 中断控制器、实时时钟RTC、Timer、Watchdog、PMC、DMAC
- 以蜂鸟E203为核心的开源SoC框架
  - HBirdv2 SoC简介

第二模块 基于RISC-V的SoC硬件架构

Chapter 5 RISC-V处理器中断和异常

- 中断和异常的基本概念
  - 同步异常、异步异常的类别与基本概念
- RISC-V处理器异常
  - RISC-V处理器异常处理机制
  - RISC-V架构的异常相关寄存器
- RISC-V处理器中断
  - RISC-V架构中断定义与分类
  - 中断控制器
  - 中断屏蔽、中断等待、中断优先级与中断嵌套
- 以蜂鸟E203开源RISC-V核心为例，介绍RISC-V处理器异常处理与中断实现方式
  - 蜂鸟E203异常处理的实现
  - 蜂鸟E203调试机制的实现

第二模块 基于RISC-V的SoC硬件架构

Chapter 6 RISC-V SoC 架构超低功耗及高性能设计

- 超低功耗架构设计方法
  - 低功耗设计的重要性
  - RISC-V架构低功耗机制和硬件实现
  - 蜂鸟E203的低功耗机制
  - 超低功耗架构设计案例：PULP介绍
- 高性能架构设计方法
  - 高性能架构设计的主流与前沿方法
  - 超高性能架构设计案例：Tenstorrent
  - 超高性能架构设计案例：算能SG2044芯片

第二模块 基于RISC-V的SoC硬件架构

实验 基于蜂鸟E203的SoC实验

- 实验一：基于蜂鸟E203的内外部中断实验
  - 实验目的：了解E203 SoC的基本架构与中断处理过程
  - 实验内容：计时器中断与软件中断、外部中断
  - 实验评估：基于FPGA平台完成E203 SoC中断的上板验证
- 实验二：基于蜂鸟E203及DMA的SoC集成实验
  - 实验目的：掌握卷积计算过程与DMA的工作机制与挂载方法
  - 实验内容：DMA挂载、二维卷积算子、DMA驱动与算子调用的软件代码编写
  - 实验评估：基于FPGA平台进行上板、软件编译烧录，串口输出完成验证

第二模块 基于RISC-V的SoC硬件架构

Chapter 4 RISC-V SoC体系架构

- 教学目标：引导学生建立系统级芯片（SoC）的宏观认知，理解SoC作为一种集成电路设计方法的动因、优势与挑战。学生将掌握SoC的核心组成部分，包括处理器内核、片上总线、存储器层次结构以及各类IP核，并以开源的蜂鸟E203 SoC为例，深入理解各组件如何协同工作，构建一个完整的系统。本章旨在培养学生分析和解构复杂芯片系统的能力，并为后续的SoC设计与实践打下坚实基础。
- 教学内容
  - SoC基本概念及组成： 介绍SoC（片上系统）的定义、本质及其为应对快速迭代和高性能需求而产生的设计背景 。阐述SoC在高性能计算、智能手机、车载电子等领域的广泛应用。分析SoC设计在性能、功耗、成本、可靠性等方面的优势及其在升级灵活性上的不足。引入Chiplet作为后摩尔时代SoC设计的新型异构集成方案。讲解SoC由硬件（芯片）、软件和集成三部分构成的系统解决方案，并介绍其基础架构，包括处理器、存储器、协处理器及外设接口等。
  - SoC总线： 讲解总线作为SoC内部互连共享硬件机制的基本概念及其关键设计要素 。介绍业界主流的AMBA总线标准（如AHB、APB、AXI） 和蜂鸟E203 SoC内部使用的ICB总线协议 ， 分析其架构、特点与适用场景。
  - SoC常见IP综述： 详细介绍SoC中各类关键IP核的功能与作用，包括负责中断管理的中断控制器（如RISC-V PLIC、CLINT） ，提供计时与日历功能的实时时钟（RTC） ，用于时间管理和系统可靠性保障的定时器（Timer）与看门狗（Watchdog） ，负责电源与功耗管理的PMC，以及实现高效数据传输的直接存储器访问控制器（DMAC）。
  - 以蜂鸟E203为核心的开源SoC框架： 以国产开源HBirdv2 SoC项目为例，介绍其以蜂鸟E203处理器核为中心的整体架构 。展示蜂鸟E203处理器核的内部结构 及其集成的丰富外设IP，如CLINT 、GPIO 、SPI 、I2C 、UART 、PWM 、WDT 、RTC 和PMU ， 帮助学生建立具体、完整的SoC系统认知。

第二模块 基于RISC-V的SoC硬件架构

▲学成效果：

- 了解：了解SoC的定义、发展背景、产业链模式及典型应用场景 。知晓SoC的基本组成部分，包括处理器、总线、常见IP核（如中断控制器、DMA、Timer等）的功能 。熟悉蜂鸟E203 SoC的整体架构和主要外设 。
- 理解： 理解SoC作为一种设计方法的核心优势（如高性能、低功耗、低成本）与固有局限性 。掌握AMBA、ICB等片上总线的工作原理及在系统中的作用 。明晰存储金字塔结构的设计思想 。理解蜂鸟E203核与其外设如何通过总线协同工作 。
- 实现： 能够识别一个给定SoC系统框图中的主要功能模块（CPU、总线、内存、外设），并初步分析其数据流路径。能够根据功能需求，为简单的应用场景选择合适的外设IP。
- 评估： 能够初步评估一个SoC架构设计的合理性。能够对比不同总线协议（如AHB与APB）的适用场景 。能够评价蜂鸟E203作为一个开源SoC平台在教学和研究中的价值。

第二模块 基于RISC-V的SoC硬件架构

Chapter 5 RISC-V处理器中断和异常

- 教学目标： 使学生系统掌握RISC-V架构中异常与中断处理的核心机制。通过对相关概念、特权级CSR寄存器以及处理流程的深入学习，学生将能够区分不同类型的异常与中断，并理解其触发与响应过程。结合蜂鸟E203处理器的具体实现，本章旨在帮助学生将理论知识与硬件实践相结合，理解从架构规范到具体芯片实现的完整逻辑，培养学生设计和调试底层系统软件的能力。
- 教学内容
  - 中断和异常的基本概念： 介绍广义异常的整体概念，并根据触发原因将其区分为同步异常和异步异常，为后续学习建立清晰的分类框架。V扩展基本模型：介绍中断寄存器的基本结构，可存储不同类型和数量的元素，并支持多种加载方式；阐述7个非特权CSR的功能。
  - RISC-V处理器异常： 详细阐述RISC-V架构定义的异常处理机制，包括进入异常时处理器如何跳转至mtvec指定的地址，并由硬件自动更新mcause、mepc、mtval、mstatus等CSR寄存器；以及退出异常时如何使用mret指令恢复程序流。强调上下文的保存与恢复需由软件完成。
  - RISC-V处理器中断： 介绍RISC-V定义的几类主要中断：外部中断 、计时器中断 、软件中断 和调试中断 。讲解用于管理多源外部中断的平台级中断控制器（PLIC） 和处理核本地中断的核心本地中断器（CLINT） 。介绍通过mie寄存器 进行中断屏蔽和通过mip寄存器 查询中断等待状态的方法。讨论中断的默认优先级及软件实现中断嵌套的策略。
  - 以蜂鸟E203开源RISC-V核心为例： 展示蜂鸟E203处理器对RISC-V异常与中断机制的具体实现 ，包括其支持的异常类型与编号、中断信号如何接入处理器交付模块 。介绍其SoC中的CLINT和PLIC模块的功能与寄存器映射 。详细讲解蜂鸟E203基于JTAG和调试模块（DTM）实现的交互式调试机制，以及ebreak、dret等相关指令的作用 。

第二模块 基于RISC-V的SoC硬件架构

▲学成效果：

- 了解：了解异常与中断的基本概念和分类。知晓RISC-V架构定义的四种中断类型（外部、计时器、软件、调试） 。熟悉PLIC和CLINT中断控制器的基本功能 。了解蜂鸟E203的调试系统组成 。
- 理解： 理解RISC-V架构的异常处理流程，掌握mtvec, mcause, mepc, mstatus等关键CSR寄存器在异常处理中的作用。理解中断屏蔽（mie） 、中断等待（mip） 、中断优先级和中断嵌套的核心原理。
- 实现： 能够根据mcause寄存器的值判断异常或中断的来源。能够编写简单的中断服务程序，在程序中正确读写相关CSR寄存器以完成中断处理。能够配置蜂鸟E203的PLIC或CLINT以使能和响应一个中断。
- 评估： 能够分析在特定场景下（如嵌入式实时系统）中断处理机制对系统响应时间的影响。能够评价蜂鸟E203中断处理实现的合规性与特点 。

第二模块 基于RISC-V的SoC硬件架构

Chapter 6 RISC-V SoC 架构超低功耗及高性能设计

- 教学目标： 使学生认识到功耗和性能是现代SoC设计的两个关键且常相互制约的维度。本章旨在引导学生掌握在不同设计层次（从系统级到物理级）进行低功耗设计的核心技术与方法，并了解追求极致性能的前沿架构思想。通过分析PULP、Tenstorrent、算能SG2044等先进的RISC-V SoC案例，学生将获得对业界真实世界设计理念的洞察，培养其在设计中平衡与优化多重目标的能力。
- 教学内容
  - 超低功耗架构设计方法： 阐述低功耗设计对于便携式设备和数据中心等场景的重要性 。系统介绍从系统级到物理级不同层次的低功耗设计技术，如软硬件协同、时钟门控、多电压、电源门控、DVFS等，并分析其效果与设计复杂度 。以蜂鸟E203为例，讲解其在系统、处理器、单元等多个层面应用的具体低功耗机制（如wfi指令、电源域划分、时钟门控等）。介绍PULP平台作为超低功耗架构设计的典型案例。
  - 高性能架构设计方法： 介绍提升SoC性能的主流方法，涵盖微架构优化、内存子系统加速和多核扩展等。探讨Chiplet异构集成等前沿设计趋势。以两个业界领先的RISC-V芯片为例进行深度案例分析：
    - Tenstorrent： 剖析其为AI计算设计的众核（Many-core）架构，讲解其由多个RISC-V CPU控制的Tensix计算核心、创新的Packet-based片上网络（NoC）以及灵活的并行化软件栈，展示其如何实现高计算利用率 。
    - 算能SG2044： 介绍这款面向服务器级应用的高性能64核RISC-V处理器，讲解其多核集群、大容量缓存、对RISC-V向量（Vector）扩展的支持，以及与自研TPU的异构融合架构，展示RISC-V在高算力场景的应用潜力 。RISC-V处理器中断： 介绍RISC-V定义的几类主要中断：外部中断 、计时器中断 、软件中断 和调试中断 。讲解用于管理多源外部中断的平台级中断控制器（PLIC） 和处理核本地中断的核心本地中断器（CLINT） 。介绍通过mie寄存器 进行中断屏蔽和通过mip寄存器 查询中断等待状态的方法。讨论中断的默认优先级及软件实现中断嵌套的策略。

第二模块 基于RISC-V的SoC硬件架构

▲学成效果：

- 了解：了解低功耗与高性能设计的市场需求与重要性 。知晓各设计层次的关键低功耗技术（如时钟门控、电源门控）和高性能架构（如多核、异构计算、Chiplet）。熟悉PULP、Tenstorrent、算能SG2044等案例的设计亮点 。
- 理解： 理解不同低功耗技术（如时钟门控与电源门控）在功耗节省效果和实现代价上的差异 。理解Tenstorrent的并行计算与片上网络设计哲学，以及算能SG2044通过多核与异构融合实现高性能的路径 。
- 实现： 能够识别设计中可应用时钟门控等基础低功耗技术的点。能够针对一个计算密集型应用，从架构层面提出初步的加速思路（例如，使用专用协处理器）。
- 评估：能够评估不同低功耗技术对系统特定指标（如面积、唤醒延迟）的影响。能够对比分析Tenstorrent和算能SG2044在架构设计上的不同取舍及其面向的应用场景。能够对未来SoC的性能与功耗发展趋势进行合理分析。

第二模块 基于RISC-V的SoC硬件架构

实验 基于蜂鸟E203的SoC实验

- 教学目标： 本系列实验旨在将前面章节的理论知识付诸实践，让学生亲手完成一个RISC-V SoC从硬件实现到软件驱动的完整流程。通过在FPGA平台上部署蜂鸟E203 SoC，并进行中断处理和DMA集成实验，学生将掌握SoC系统集成、外设驱动编写和软硬件协同调试的关键技能。实验旨在锻炼学生的动手能力和解决实际工程问题的能力，加深对SoC工作原理的理解。
- 教学内容
  - 实验一：基于蜂鸟E203的内外部中断实验： 以掌握E203 SoC中断处理过程为目的 。实验内容分为两部分：一是通过配置SoC中的CLINT模块，学习计时器中断和软件中断的产生与处理 ；二是通过配置GPIO和PLIC模块，学习外部中断的触发、响应与处理 。实验流程包括将E203 SoC移植到DDR200T FPGA开发板 ，使用Vivado进行硬件综合与比特流生成并烧录 ，最后在芯来IDE中编写C语言中断服务程序进行编译和上板验证 。
  - 实验二：基于蜂鸟E203及DMA的SoC集成实验： 以掌握DMA工作机制及SoC IP集成为目的。实验要求学生理解DMA不占用CPU资源进行数据搬运的原理 ，并以二维卷积计算为例 。实验内容包括：将提供的DMA控制器IP挂载至E203的SoC总线上 ；编写二维卷积计算的C语言算子 ；编写DMA驱动软件，配置DMA源/目的地址和传输长度，以实现内存数据的高效搬运 ；最后在FPGA平台上进行软硬件联合验证，通过串口输出验证数据搬运与计算的正确性 。

第二模块 基于RISC-V的SoC硬件架构

▲学成效果：

- 了解：了解在FPGA上验证SoC设计的基本流程，熟悉Vivado和Nuclei Studio等开发工具的使用。知晓蜂鸟E203 SoC的中断和DMA模块的寄存器接口。
- 理解： 理解内部中断（Timer/Software）和外部中断（GPIO）在硬件触发和软件处理上的区别。深刻理解DMA的工作原理，以及它如何通过与CPU并行工作来提升系统效率 。
- 实现： 能够独立完成E203 SoC在FPGA开发板上的部署、软件编译和下载。能够编写C代码来配置中断控制器并实现中断服务函数。能够将一个第三方IP（DMA）集成到SoC系统中，并编写驱动程序来控制该IP完成指定任务。
- 评估： 能够通过实验现象（如串口打印、LED闪烁）判断中断程序是否正确执行。能够分析和对比CPU直接搬运数据与通过DMA搬运数据在代码实现和预期性能上的差异。具备初步的软硬件协同调试能力。

上海开放处理器产业
创新中心

《RISC-V导论：设计与实践》

第三模块 嵌入式系统软件开发

第三模块内容

- 第7章 RISC-V 编程
  - **RISC-V 汇编语言程序设计**
  - **RISC-V GNU 工具链**
  - **集成开发环境 Nuclei Studio**
- 第8章 嵌入式操作系统FreeRTOS移植与示例运行
  - FreeRTOS 的概念
  - FreeRTOS移植
  - FreeRTOS运行案例

- 第9章嵌入式软件运行
  - 高级语言翻译
  - 程序的运行
  - 案例分析：HelloWorld程序运行过程综述
  - 进阶：嵌入式软件行为的安全可靠保障
- 第10章 嵌入式系统应用案例
  - PWM应用开发
  - SPI接口应用
  - 嵌入式系统实际应用案例
- 实验三 I2C温度传感器测试

上海开放处理器产业
创新中心

《RISC-V导论：设计与实践》

第七章 RISC-V编程

学习目标

▲学习内容 (Topics)

▶掌握RISC-V汇编语言编程（必学）

- 理解 RISC-V 汇编语言的基本结构、指令集与伪指令功能（如数据定义、符号声明、段定义等）；
- 能编写简单的 RISC-V 汇编程序，实现数据处理、流程控制（分支 / 循环）等基础逻辑；
- 掌握汇编程序与 C/C++ 语言的混合编程方法（嵌入汇编代码或调用 C 函数）。

▶熟悉 RISC-V GNU 工具链的使用（必学）

- 了解 RISC-V GNU 工具链的核心组件（GCC 编译器、Binutils 工具、GDB 调试器等）；
- 能使用工具链完成汇编程序的编译、链接、调试及格式转换（如.elf转.bin）；
- 掌握通过命令行参数配置架构（-march）、ABI（-mabi）等编译选项。

▶掌握集成开发环境 Nuclei Studio 的操作（拓展）

- 了解 Nuclei Studio 的功能特点（代码编辑、编译调试、可视化界面等）；
- 能通过 Nuclei Studio 创建、编译和调试 RISC-V 项目，熟悉其工程管理流程。

学习目标

▲学成效果 (Learning Outcomes)

▶知识层面

- 理解 RISC-V 汇编语言的语法规则和编程模型，区分伪指令与真实指令的差异；
- 掌握 RISC-V 工具链的全流程使用，从源代码到可执行文件的完整构建过程；
- 了解 Nuclei Studio 在 RISC-V 开发中的应用场景，对比命令行工具与 IDE 的优缺点。

▶技能层面

- 能独立编写包含数据操作、分支判断、循环结构的 RISC-V 汇编程序；
- 能通过 GDB 调试汇编程序，定位运行时错误；
- 能在实际项目中灵活运用汇编与 C/C++ 混合编程，优化关键代码性能。

▶实践层面

- 通过案例分析（如汇编程序示例），提升解决实际问题的能力；
- 熟悉开源工具链的生态，为后续 RISC-V 架构开发（如操作系统、驱动程序）奠定基础。

研究课题和参考文献

▲研究课题

▶RISC-V 伪指令的实现原理

- 分析不同伪指令在汇编阶段的转换逻辑，对比不同工具链，如 GCC 与 LLVM的处理差异

▶RISC-V 汇编与 C 语言混合编程优化

- 探讨如何通过内联汇编提升 C 程序关键函数的执行效率，如数学运算、位操作，并测试性能差异

▶基于 RISC-V 的嵌入式系统开发

- 使用 RISC-V GNU 工具链和 Nuclei Studio 开发一个简单的嵌入式程序，如 LED 控制、串口通信，实现硬件与软件的协同调试

▶RISC-V 向量指令集（RVV）的汇编编程

- 结合 RVV 指令集特性，尝试编写向量运算的汇编程序，如矩阵乘法、FFT 变换，利用工具链的向量化优化选项

1. 《RISC-V CPU 工程与实践》
2. 《嵌入式系统原理与开发——基于 RISC-V 和 Linux 系统》
3. Nuclei Studio

第七章 RISC-V 编程

7.1 RISC-V 汇编语言程序设计

- 7.1.1 汇编语言概述
- 7.1.2 RISC-V 汇编程序概述
- 7.1.3 RISC-V 汇编伪指令
- 7.1.4 RISC-V 汇编程序伪操作
- 7.1.5 RISC-V 汇编程序示例
- 7.1.6 在C/C++程序中嵌入汇编程序
- 7.1.7 在汇编程序中调用C/C++语言中的函数
- 7.1.8 总结

第七章 RISC-V 编程

上海开放处理器产业
创新中心

《RISC-V导论：设计与实践》

第八章 FreeRTOS 移植与示例运行

学习目标

- ▲学习内容 (Topics)
  - ▶ 结合FreeRTOS，了解实时操作系统（必学）
    - RTOS的基本概念
    - RTOS和Baremetal程序开发的不同
    - RTOS程序开发的方法
  - ▶ 掌握FreeRTOS的移植（必学）
    - 了解FreeRTOS源代码目录的作用
    - 了解硬件抽象层、编译器调整和验证等各个步骤在RTOS移植过程中的作用
    - 掌握移植各步骤的实现方法
- ▲学成效果 (Learning Outcomes)
  - ▶ 能够回答以下问题
    - FreeRTOS的移植需要做哪些工作
    - 具备移植FreeRTOS的基本知识，完成RISC-V处理器的FreeRTOS移植
    - 理解FreeRTOS操作系统的运行模式
  - ▶ 了解保障嵌入式软件正确性的模型检测方法

研究课题和参考文献

- FreeRTOS在不同规格和配置的RISC-V 处理器移植的不同实现方案
- FreeRTOS在不同规格和配置的RISC-V 处理器移植的不同方案的性能评测

1. Amazon. FreeRTOS Documentation
2. Nuclei. GitHub - riscv-mcu/e203_hbirdv2: The Ultra-Low Power RISC-V Core
3. 胡振波，手把手教你RISC-V CPU（下），人民邮电出版社，2021年9月

第八章 FreeRTOS 的概念

- RTOS与Baremetal
- RTOS介绍
- FreeRTOS基础知识
- FreeRTOS移植介绍
- FreeRTOS在GD32VF103中的移植
- FreeRTOS移植后的运行

上海开放处理器产业
创新中心

《RISC-V导论：设计与实践》

第九章-嵌入式系统软件运行

学习目标

- 学习内容 (Topics)
  - 结合蜂鸟SoC，了解硬件/软件接口（必学）
    - 计算机是一个软硬件协同工作的整体
    - 高级语言程序到RISC-V机器指令的翻译过程
  - 围绕FreeRTOS，理解操作系统为程序的运行提供的服务（必学）
    - 计算机运行程序前的准备工作
    - 一个程序运行所得到的操作系统支持
  - 开发安全可靠的嵌入式软件（拓展）
- 学成效果 (Learning Outcomes)
  - 能够回答以下问题
    - 理解计算机将C或Python等高级语言程序翻译成机器指令的过程
    - 理解操作系统帮助下RISC-V机器语言程序的执行过程
    - 能够评估C语言在蜂鸟SoC上的各种性能优化方案
  - 了解保障嵌入式软件正确性的模型检测方法

研究课题和参考文献

- 针对蜂鸟E230的编译器优化
- 在资源受限的RISC-V SoC上实现自适应控制算法
- 验证控制算法的安全性和可靠性
- 高级RISC-V计算机系统中多核任务分配对系统实时性能的影响

1. Amazon. FreeRTOS Documentation
2. Atif, Muhammad, Groote, Jan Friso. Understanding Behaviour of Distributed Systems UsingmCRL2. Springer, 2023
3. Nuclei. GitHub - riscv-mcu/e203_hbirdv2: The Ultra-Low Power RISC-V Core
4. Samek, Miro Practical UML Statecharts in C/C++ 2nd ed. Taylor and Francis. 2009
5. Wang, Jiacun. Real-time embedded systems. Wiley 2017

上海开放处理器产业
创新中心

《RISC-V导论：设计与实践》

第10章 嵌入式系统应用案例

学习目标

- 学习内容 (Topics)
  - MCU硬件接口模块
    - 硬件接口功能介绍
    - 接口初始化与控制方法
  - 嵌入式系统开发示例
    - 学习如何用软件程序控制硬件接口工作
- 学成效果 (Learning Outcomes)
  - 了解MCU常见外设接口的构造与工作原理
  - 掌握硬件接口编程控制方法
  - 了解嵌入式系统在实际工程中的应用

研究课题和参考文献

- 针对蜂鸟E230的接口开发
- 嵌入式MCU在工业控制领域的应用
- 嵌入式MCU在移动产品领域的应用
- 嵌入式MCU在新能源领域的应用

1. 胡振波，手把手教你RISC-V CPU（下），人民邮电出版社，2021年9月
2. 王剑等，嵌入式系统原理与开发-基于RISC-V与linux系统，清华大学出版社，2024年6月

第四部分：智算时代的RISC-V

Chapter 11:智慧计算的发展趋势

- 摩尔定律的兴衰
  - 摩尔定律的内容与辉煌历程
  - 摩尔定律面临的瓶颈：物理极限与经济成本
- 算力需求与供给的矛盾
  - 算力需求的爆发：AI 与大模型的推动及其他领域需求
  - 算力供给的困境：晶体管密度增速放缓及 “剪刀差” 影响
- 超越摩尔的计算革新
  - 异构计算：多元算力协同的架构与应用
  - 架构革新：Chiplet、存算一体等技术原理与优势
- 冯・诺依曼瓶颈及应对策略
  - 冯・诺依曼瓶颈的核心问题：存储墙与功耗墙
  - 应对策略：DSA、新型架构等解决方案
- 软件新挑战与开源生态
  - 硬件革新下的软件挑战：编程、资源管理等方面
  - 开源生态的支撑：Linux 系统、OpenCL 等的作用

第四部分：智算时代的RISC-V

Chapter 12: V 扩展指令集

- V 扩展指令集概述
  - 向量化计算原理：与传统标量计算的对比
  - RISC-V V 扩展的设计目标与特点
- V 扩展基本模型
  - 向量寄存器：结构、类型及分段加载机制
  - 非特权 CSR：各寄存器的功能与作用
  - 向量寄存器的灵活性：可变长度与掩码操作
- V 扩展核心指令集
  - 指令类型分类：加载 / 存储、算术运算等指令详解
  - 指令示例与应用场景：以矩阵运算等为例
- V 扩展的应用与未来
  - 应用领域：高性能计算、嵌入式等领域的应用
  - 未来发展：标准制定与软件生态完善方向

第四部分：智算时代的RISC-V

Chapter 13: Matrix 与 P/K/VK 扩展指令集

Matrix 扩展指令集

- Matrix 扩展概述：设计目的与性能亮点
- 矩阵寄存器与 CSR：寄存器结构与控制状态寄存器
- 指令集：配置、乘法、加载 / 存储等指令介绍
- AME 与 IME：附加与集成矩阵扩展的对比

P 扩展指令集

- 设计动机与应用场景：小型处理器的需求与适用场景
- 技术细节与指令表：SIMD 处理及各类指令操作
- 实现架构与标准化：硬件实现与标准化进展

K 扩展指令集

- 设计动机与应用场景：密码学需求及应用领域
- 指令集与设计原则：各类密码指令及设计原则
- 实现架构与标准化：实现方式与标准化状态

VK 扩展指令集

- 设计动机与性能优势：高端应用需求与性能提升
- 指令集与架构实现：向量密码指令及架构设计
- 标准化与产业项目：标准化进展与相关项目

第四部分：智算时代的RISC-V

Chapter 14: RISC-V 扩展指令集实验

- 矩阵运算案例实验
  - 实验需求与原理：矩阵运算要求与指令集扩展原理
  - 实验步骤：定义指令、C 语言实现及 NICE 协处理器实现
  - 实验评估：对比软件与硬件实现的性能差异
- Yolo 目标检测加速实验
  - 实验背景与架构原理：AI 检测需求与系统架构
  - 协处理器设计：模型部署、数据流及加速器架构
  - 主处理器设计：自定义指令及相关功能实现
  - 实验验证与拓展：仿真、下载验证及 Nuclei Wizard 应用
- RISC-V 自定义扩展：Vortex - GPGPU
  - 实验背景与架构原理：RISC-V 在 GPU 领域的机遇与系统架构
  - 硬件架构与指令扩展：Core 单元架构及线程相关指令
  - 内联汇编与 CSR 扩展：自定义指令调用及 CSR 监控
  - Tensor core 设计与扩展：Tensor core 的设计与扩展方式

第四模块

Chapter 11:智慧计算的发展趋势

- 教学目标：引导学生知悉智慧计算领域的发展脉络，理解摩尔定律演进与算力供需矛盾产生的内在逻辑，了解异构计算、多元架构协同等前沿技术的原理与应用场景。同时，培养学生分析硬件革新对软件带来的挑战及应对策略的能力，突出 RISC-V 在智慧计算中的创新价值与实践应用，提升对智慧计算技术发展趋势的前瞻性认知与技术评估能力。

- 教学内容
  - 摩尔定律的兴衰与算力供需矛盾：介绍摩尔定律从辉煌到面临瓶颈的过程，包括物理极限和经济成本的制约；阐述AI与大模型发展带来的算力需求指数式增长，以及算力供给增速放缓形成的“剪刀差”，分析其在技术和经济层面的影响。
  - 各类计算架构的特点与应用：详解RISC-V与DSA结合在领域专用计算中的应用；分析CPU、GPU/GPGPU在并行计算中的核心特点、实现方式、应用场景和局限性，以及NVIDIA GPGPU与RISC-V结合的情况；阐述异构计算中多元算力协同的模式和优势，以及Chiplet异构集成和存算一体架构对计算能效的提升。
  - 智慧计算时代的软件挑战与应对：论述硬件革新给软件带来的挑战；介绍软件发展的未来方向，如软件定义硬件；强调Linux系统作为开源基石在智慧计算中的重要性，以及OpenCL、OpenGL、Vulkan在跨平台并行编程中的作用和优势。
  - RISC-V 在智慧计算中的崛起与创新：阐述RISC-V开放标准的特性和价值；介绍RISC-V在多个领域的应用场景和案例，以及其向量（V）、矩阵（Matrix）扩展和自定义扩展在提升计算性能方面的作用和优势。
  - 展示RISC-V相关企业的智算系统案例

第四模块

▲学成效果：

- 了解：了解摩尔定律兴衰历程、算力供需矛盾现状，知晓 CPU、GPU/GPGPU 等各类计算架构的特点与应用场景，熟悉智慧计算时代软件面临的挑战，掌握 RISC-V 开放标准特性。
- 理解：理解异构计算与架构革新的技术路径，包括 Chiplet 异构集成、存算一体芯片的优势，以及 RISC-V 与 DSA 结合的意义；明晰 Linux 系统及跨平台并行编程工具在智慧计算中的重要作用。
- 实现：能够结合具体场景，实现对不同计算架构适用性的初步判断，能够运用 RISC-V 相关知识分析其在特定领域应用的可行性，能够针对硬件革新带来的软件挑战提出基础的应对思路。
- 评估：能够评估算力供需矛盾对技术和经济层面的影响，能够评价 RISC-V 在智慧计算领域的创新价值与发展潜力，能够对智慧计算技术未来发展趋势进行合理的分析与预测 。

参考文献及研究课题

▲参考文献：

- 1、Yu Z, Liang S, Ma T, et al. Cambricon-llm: A chiplet-based hybrid architecture for on-device inference of 70b llm[C]//2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2024: 1474-1488.
- 2、Wang S, Wang X, Xu Z, et al. Optimizing cnn computation using risc-v custom instruction sets for edge platforms[J]. IEEE Transactions on Computers, 2024, 73(5): 1371-1384.
- 3、Syu J H, Lin J C W, Srivastava G, et al. A comprehensive survey on artificial intelligence empowered edge computing on consumer electronics[J]. IEEE Transactions on Consumer Electronics, 2023, 69(4): 1023-1034.

▲研究课题

- 1、基于 DSA 的视觉 Transformer 模型加速设计与实现
- 2、CPU-NPU 异构架构下的模型压缩协同优化策略
- 3、基于 GPU-FPGA 异构的实时视频处理任务卸载与优化

第四模块

Chapter 12:V 扩展指令集

- 教学目标：本章聚焦 RISC-V V 扩展技术，旨在让学生系统掌握 RVV 指令集的设计理念、核心架构与指令体系，深入理解向量化计算的优势及实现原理。通过理论讲解与实例分析结合，使学生熟练运用 V 扩展指令进行数据处理与运算，清晰把握其在多领域的应用价值与未来发展趋势，培养学生将 RVV 技术应用于实际问题解决的能力，提升对前沿计算技术的探索与创新思维。
- 教学内容
  - RISC-V V扩展概述：介绍向量化计算相较于传统标量计算的优势，以简单加法和神经网络计算为例说明其提升性能的原理；阐述RVV的核心设计目标，包括灵活性与可扩展性、高效性与通用性、硬件简化与代码兼容性，以及与传统SIMD的对比优势。
  - V扩展基本模型：介绍向量寄存器的基本结构，可存储不同类型和数量的元素，并支持多种加载方式；阐述7个非特权CSR的功能；强调向量寄存器的灵活性，如可变长度向量可根据运行时动态配置，向量化掩码能提升计算效率。
  - V扩展核心指令集：介绍V扩展指令类型丰富；说明各类型指令的子类和功能，通过大量示例程序展示如何使用这些指令进行数据处理、运算、掩码操作、向量排列和特殊计算，涵盖整数、浮点、定点数等多种数据类型。
  - V扩展的应用与未来：阐述V扩展在高性能计算、嵌入式设备与边缘计算、软件性能优化等领域的应用；探讨V扩展的未来发展方向，包括标准制定的推进，以及在智能驾驶、区块链与加密、新能源与工业物联网等新兴领域的应用潜力，强调软件生态完善的重要性，如编译器、标准库和仿真验证工具的发展。

第四模块

▲学成效果：

- 了解：了解 RISC-V V 扩展的设计初衷、向量寄存器基本结构及 7 个非特权 CSR 功能；知晓 V 扩展指令类型划分，以及其在高性能计算、嵌入式设备等领域的应用场景；了解 RVV 未来在新兴领域的应用潜力和软件生态发展方向。
- 理解：理解向量化计算相较于标量计算的性能提升原理，掌握 RVV 灵活性、可扩展性等核心设计目标的实现逻辑；理解 V 扩展指令各类型功能及与传统 SIMD 的差异；理解可变长度向量和向量化掩码的工作机制及其对计算效率的影响。
- 实现：能够实现基于 RISC-V V 扩展指令集编写简单数据处理、算术运算程序，熟练运用向量配置与控制、加载 / 存储等指令完成基础计算任务；能针对不同应用场景，合理选择 V 扩展指令优化软件性能。
- 评估：能够评估 RISC-V V 扩展在不同计算场景中的适用性与优势；对 RVV 未来标准制定、应用拓展方向及软件生态完善的可行性进行分析与判断；能根据实际需求，评估采用 V 扩展技术解决计算问题的潜在价值与局限性。

参考文献及研究课题

参考文献：

- 1、 Wang C, Fang C, Wu X, et al. SPEED: A Scalable RISC-V Vector Processor Enabling Efficient Multiprecision DNN Inference[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2024.
- 2、 Guthmuller E, Fuguet C, Bocco A, et al. Xvpfloat: RISC-V ISA extension for variable extended precision floating point computation[J]. IEEE Transactions on Computers, 2024.
- 3、 Perotti M, Cavalcante M, Ottaviano A, et al. Yun: An open-source, 64-bit RISC-V-based vector processor with multi-precision integer and floating-point support in 65-nm CMOS[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2023, 70(10): 3732-3736.

研究课题

- 1、基于 RISC-V V 扩展指令集的稀疏矩阵向量乘法性能优化
- 2、RISC-V V 扩展指令集在图像滤波算法中的高效实现
- 3、基于 RISC-V V 扩展指令集的 FFT 算法硬件加速设计

第四模块

Chapter 13: Matrix与P/K/VK扩展指令集

- 教学目标：通过本章教学使学生全面掌握各扩展指令集的设计理念、技术特性、应用场景与实现架构。结合实际案例分析与理论推导，帮助学生理解不同扩展指令集在满足特定计算需求时的优势与局限，培养学生运用这些扩展指令集解决实际计算问题的能力，同时深化学生对 RISC-V 开放工具链与分层设计哲学的认知，提升学生对前沿计算技术的探索与创新能力。
- 教学内容
  - Matrix扩展指令集：介绍其采用与向量扩展解耦的架构，定义了相关参数、矩阵寄存器和CSR；详细阐述指令格式、指令类型及其功能，还提及AME与IME两种附加矩阵扩展的特点和区别。
  - P扩展指令集：P扩展是为小型处理器提供的一种硬件友好、面积和功耗可控的SIMD支持方式，旨在满足其并行计算需求，应用于图像处理、数字信号处理、机器学习推理等场景；介绍其技术细节；列举P扩展指令表中的各类指令，说明其实现架构和优势，以及标准化状态、商业实现、开源项目和工具链支持情况。
  - K扩展指令集：K扩展是为解决传统指令集执行密码学操作效率低的问题而设计，应用于TLS/SSL协议加速、区块链签名验证、安全启动机制等场景；介绍其设计原则；详细阐述各类K扩展指令的功能和适用场景，说明其实现架构、标准化状态、商业实现、开源项目和模拟器支持情况。
  - VK扩展指令集：VK扩展结合V扩展和K扩展，面向高端应用，旨在用向量化方法加速密码运算，具有吞吐量提升和能耗效率比提升的优势；介绍其技术细节，包括基于向量寄存器的操作、支持的算法和特性；详细阐述各类VK扩展指令的功能和约束，说明其架构实现、标准化状态、实验性实现和开源探索情况。
  - 开源工具链与生态系统及设计哲学：介绍开源工具链对这些扩展指令集的支持情况；阐述不同类型扩展指令集的设计哲学，基础计算类扩展追求效率，安全敏感操作采用协处理器实现物理隔离，高性能专用扩展通过向量架构实现吞吐量突破，这种分层实现方式平衡了灵活性与性能。。

第四模块

- 学成效果
  - 了解：了解 Matrix、P、K、VK 等扩展指令集的设计初衷、应用场景、标准化状态、商业实现及开源项目支持情况；知晓开源工具链对各扩展指令集的支持方式；了解不同类型扩展指令集的分层设计哲学。
  - 理解：理解 Matrix 扩展与向量扩展解耦架构的优势，P 扩展满足小型处理器并行计算需求的原理，K 扩展提升密码学操作效率的设计原则，以及 VK 扩展融合向量化与密码运算的技术逻辑；理解 AME 与 IME 两种附加矩阵扩展的区别，P 扩展定长 SIMD 处理机制，K 扩展各类密码指令功能，VK 扩展指令的操作约束与特性。
  - 实现：能够根据具体计算任务需求，选择合适的 RISC-V 扩展指令集并设计基础解决方案；实现基于各扩展指令集编写简单的矩阵运算、图像处理、密码学操作等程序；能够运用开源工具链对包含扩展指令集的程序进行编译、仿真与调试。
  - 评估：能够评估不同扩展指令集在特定应用场景中的适用性与性能表现，分析其优势与局限性；对各扩展指令集的未来发展趋势、标准化进程及生态完善方向进行合理判断；能够根据实际需求，评估采用 RISC-V 扩展指令集解决问题的技术可行性与经济成本效益 。

参考文献及研究课题

参考文献：

- 1、Kim H, Ye G, Wang N, et al. Exploiting intel® advanced matrix extensions (amx) for large language model inference[J]. IEEE Computer Architecture Letters, 2024.
- 2、Ali M, Aliagha E, Elnashar M, et al. RV-VP 2: Unlocking the Potential of RISC-V Packed-SIMD for Embedded Processing[C]//International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation. Cham: Springer Nature Switzerland, 2024: 59-71.
- 3、Zhao Y, Xie R, Xin G, et al. A high-performance domain-specific processor with matrix extension of RISC-V for module-LWE applications[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2022, 69(7): 2871-2884.

研究课题

- 1、基于 RISC-V Matrix 扩展指令集的边缘 AI 推理性能优化
- 2、P扩展指令在 H.264 视频编码帧内预测的优化实现
- 3、基于 K扩展指令的 AES-256-GCM 模式硬件加速设计

第四模块

Chapter 14: RISC-V 扩展指令集实验

- 教学目标：通过矩阵运算、Yolo 目标检测加速、RISC-V 自定义扩展（Vortex - GPGPU）三大实验教学，引导学生深入理解指令集扩展对提升计算性能与能效的核心作用，掌握 RISC-V 在不同应用场景下的定制化开发方法。结合理论讲解与实践操作，使学生熟悉从实验设计、架构搭建到代码实现、结果验证的完整流程，培养学生运用 RISC-V 开放指令集解决实际计算问题、优化系统性能的能力，以及对新兴计算架构的探索与创新思维。
- 教学内容
  - 矩阵运算案例实验：以3x3矩阵运算为实验需求，对比C语言实现和使用自定义指令实现的差异，引出通过指令集扩展提升性能与能效的实验目的；介绍NICE指令格式及扩展步骤；分别展示C语言实现和NICE协处理器实现的代码及计算逻辑，通过对比不同优化级别下常规软件实现和采用NICE协处理器的硬件实现在指令数和时钟周期数上的差异，得出NICE协处理器可带来性能提升且矩阵越大性能提升越明显的结论。
  - Yolo目标检测加速实验：基于AI目标检测的大量并行计算和边缘嵌入式部署需求，结合RISC-V开放指令集和DSA架构优化的特点，阐述实验背景；介绍加速器系统整体架构，包括主处理器与协处理器的分工交互；讲解yolov3-tiny模型、非对称量化方法、数据流设计、加速器架构以及NICE指令和指令集扩展；展示主处理器设计中的自定义指令、NICEcore解码、ACR读写实现、封装内联函数和矩形框计算；通过NucleiStudio联合Vivado仿真验证和下载验证两种方式对实验进行验证，并介绍Nuclei Wizard应用在实验拓展中的作用。
  - RISC-V自定义扩展（ Vortex - GPGPU ）：针对高性能计算公司对指令集架构闭源的现状，说明RISC-V开放指令集在GPU设计领域打破垄断的潜力，阐述实验背景；介绍CPU + Vortex异构系统架构中Host与Device的分工交互，以及基于RISC-V扩展的CORE Vortex硬件架构；详细讲解Vortex指令扩展中的线程模型、线程发散与同步相关指令及其硬件实现结构，以及指令扩展步骤和内联汇编调用方式；介绍Tensor core设计与扩展的方法和思路。

第四模块

学成效果

- 了解：了解矩阵运算、Yolo 目标检测、GPU 设计等实验的背景需求；知晓 NICE 指令格式、Yolo 加速器系统架构、CPU + Vortex 异构系统架构的基本组成；了解 NucleiStudio、Vivado 等实验工具及 Nuclei Wizard 应用的功能；熟悉 Vortex 指令扩展中的线程模型、CSR 功能及 Tensor core 设计思路。
- 理解：理解通过指令集扩展提升矩阵运算性能与能效的原理，掌握NICE 指令的工作机制；理解 Yolo 目标检测加速器的数据流设计、乘加计算优化等架构设计逻辑，以及非对称量化方法的作用；理解 Vortex 指令扩展中线程发散与同步指令的硬件实现原理，明白 RISC-V 开放指令集在 GPU 设计领域打破垄断的优势所在。
- 实现：能够实现基于 NICE 协处理器完成 3x3 矩阵运算程序的编写与优化，并对比分析与 C 语言实现的性能差异；能够基于 RISC-V 开放指令集和 DSA 架构，完成 Yolo 目标检测加速器的设计与开发，通过仿真验证和下载验证实现模型加速。
- 评估：能够评估不同指令集扩展方案在特定计算任务中的性能提升效果与能效优势；对 Yolo 目标检测加速器架构设计的合理性、可扩展性进行分析判断；能够评估 RISC-V 自定义扩展（Vortex - GPGPU）在高性能计算领域的应用潜力与技术可行性，为相关计算系统的优化与创新提供参考依据。