本 科 毕 业 论 文

课题名称：基于 **RISC-V** 存算一体芯片的编译器关键技术研究

学 员 姓 名：简泽鑫　　学　　号：202102001019

首次任职专业：无　　学历教育专业：计算机科学与技术（计算机系统）

命 题 学 院：计算机学院　　年　　级：2021 级

指 导 教 员：曾坤　　职　　称：副研究员

所 属 单 位：计算机学院微电子与微处理器研究所

国防科技大学教育训练部制

目　录

摘　要 i
ABSTRACT ii
第 1 章　引言 1
1.1　研究背景及意义 1
1.1.1　研究背景 1
1.1.2　研究意义 2
1.2　国内外研究现状 3
1.2.1　深度学习编译器 3
1.2.2　深度学习加速器 7
1.3　论文的主要研究工作 8
1.4　论文组织结构 8
第 2 章　主要技术基础 9
2.1　RISC-V 9
2.2　LLVM 编译器 9
2.2.1　LLVM 结构 10
2.2.2 11
2.3　计算图优化研究 11
2.4　本章小结 11
第 3 章　计算图算子自动调度器 12
第 4 章　基于 RISC-V 存算一体芯片的编译器后端设计 13
第 5 章　编译器测试与分析 14
第 6 章　总结与展望 15
6.1　本文的工作总结 15
6.2　未来的工作展望 15
致　谢 16

摘　要

随着 AI 技术逐渐渗透到各大应用场景，市场对算力的需求呈现爆发式增长。

因此本研究致力于解决上述难题，主要研究了

通过测试表明，本文实现的编译器能够将应用算子自动映射到具有不同 IP 设计的加速部件，根据不同芯片架构特征生成正确的指令流来协调各个计算部件，挖掘芯片内部的计算并行性。

关键词：深度神经网络；编译；调度器；存算一体

ABSTRACT

Abstract.

KEY WORDS: key word 1, key word 2, key word 3

第 1 章　引言

1.1　研究背景及意义

1.1.1　研究背景

随着人工智能算法复杂度呈指数级跃迁以及物联网终端设备产生的数据量突破 ZB 量级，传统的计算架构正面临前所未有的“双重困境”：一方面，受限于冯诺依曼架构中存储单元与计算单元的物理分离特性，数据在片外存储与运算核心之间的数据频繁迁移导致系统能效比急剧恶化。根据英特尔的研究显示，半导体工艺到了 7nm 时代，数据搬运功耗达到 35pJ/bit，占比达 63.7%。数据传输所导致的功耗损失越来越成为芯片发展的制约因素。这种“功耗墙”现象严重制约了 AI 芯片在边缘计算场景的部署能力。

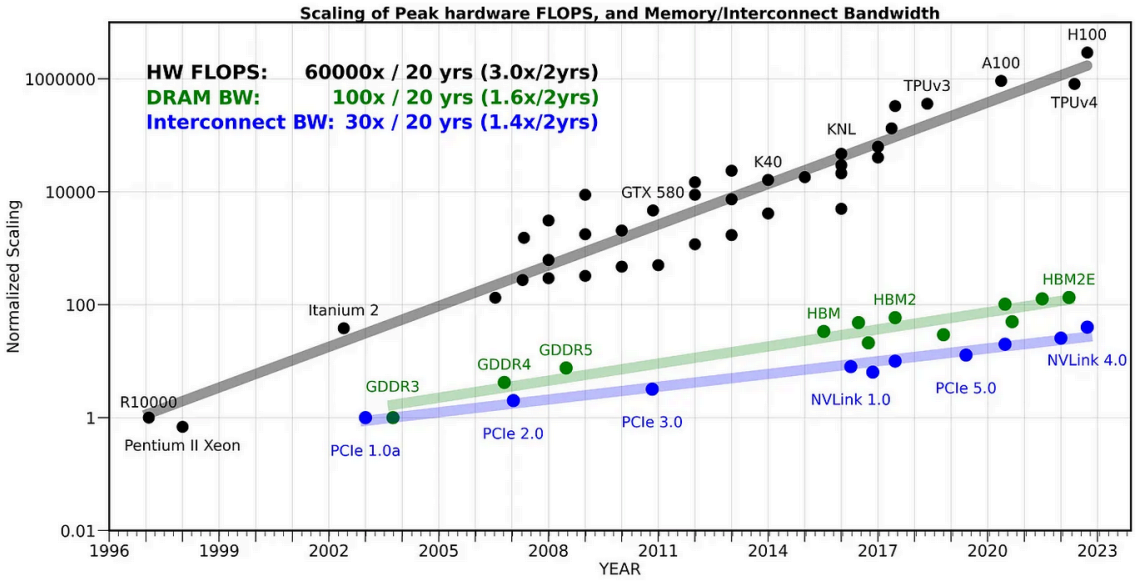

图 1.1　算力发展速度远超存储器

另一方面，随着半导体工艺逼近物理极限，单纯依靠工艺微缩带来的性能提升已显疲态。ITRS 路线图指出，传统架构下每代工艺节点的性能增益从 28nm 时代的 40% 骤降至 5nm 节点的 15% [2]，摩尔定律的失效迫使学界寻求架构层面的突破性创新。

在此严峻形势之下，存算一体（Compute-In-Memory，CIM）架构应运而生，为突破传统架构限制带来了全新的希望与解决方案。该架构的核心创新之处在于将计算功能巧妙地集成于存储单元之中，从根源上显著减少了数据传输的庞大体量，

从而成功突破了长期以来困扰业界的冯诺依曼瓶颈，实现了系统性能以及能效的大幅提升与飞跃，为计算架构领域开辟了全新的发展方向与路径。

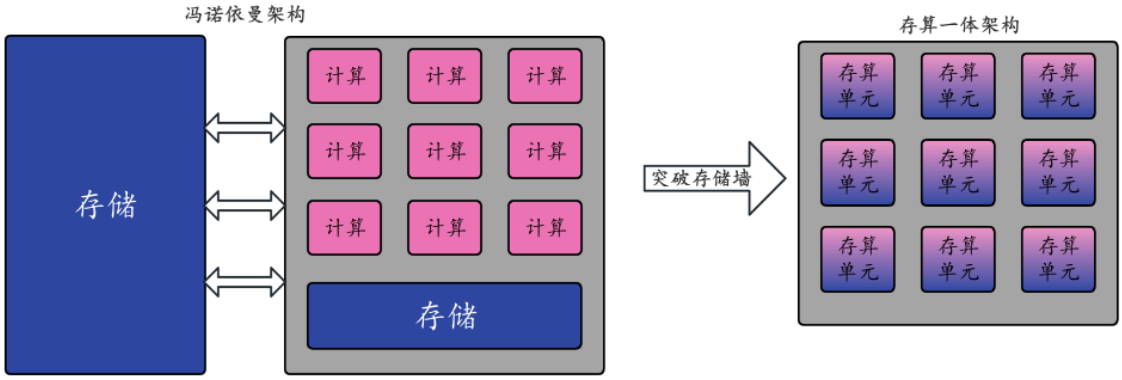

图 1.2　冯诺依曼 v.s. 存算一体架构

与此同时，RISC-V 架构以其开放包容、灵活可扩展的特性在众多指令集架构（Instruction Set Architecture，ISA）中脱颖而出。其完全开源的特质，搭配上模块化设计以及强大且卓越的可定制性优势，使其迅速成为构建存算一体芯片时备受青睐的首选架构方案。然而，不容忽视的是，基于 RISC-V 打造的存算芯片往往呈现出异构、碎片化的显著特点，这就给开发者带来了巨大的挑战与困扰。开发者在面向不同存算架构进行应用开发时，往往不得不针对每一种特定架构开发多个不同版本的应用程序，不仅极大地降低了开发效率，还使得应用部署过程变得异常艰难与繁琐。鉴于此，如何巧妙地实现软件操作（涵盖了计算过程、数据通信等关键环节）与硬件配置（诸如异构计算单元、存储层次结构等复杂要素）之间的深度解耦，从而使得 AI 应用开发能够摆脱对存算 IP 设计的高度依赖，已然成为破解当前“编程墙”困局的关键所在与核心突破口。

基于上述现状与需求，本课题将聚焦于面向 RISC-V 存算芯片的编译支持工作，针对性地对 LLVM 编译器进行深入修改与定制优化，使其能够有效支持存算指令的执行与处理。具体而言，我们将深入钻研 LLVM 编译器的架构体系以及内在工作原理，全面细致地分析 RISC-V 存算一体芯片的特性与需求，积极探索如何在 LLVM 编译框架中成功添加对 RISC-V CIM 架构的全面支持。这一过程涵盖了对指令集的合理扩展、内存模型的精准适配以及优化策略的科学调整等多个关键环节与技术要点。通过这些努力，我们旨在实现应用算子的自动化精准映射以及正确指令流的高效生成，进而更好地协调各计算部件之间的协作关系，深度挖掘芯片内部所蕴含的计算并行性优势，最终为 RISC-V 存算一体芯片构建起坚实可靠的编译支持体系，助力其在实际应用中发挥出卓越的性能表现。

1.1.2　研究意义

1.2 国内外研究现状

1.2.1 深度学习编译器

随着深度学习技术的不断成熟，深度学习编译器也得到了快速发展。在这一阶段，出现了许多具有代表性的深度学习编译器：

TensorFlow XLA（Accelerated Linear Algebra/加速线性代数）[1]：Google 于 2017 年开发的用于加快 TensorFlow 模型运行速度的编译器。其接收来自 PyTorch、TensorFlow 和 JAX 等 ML 框架的模型，在中间优化层级，XLA 包括整体模型优化，如简化代数表达式、优化内存数据布局和改进调度等等。但是 XLA 主要针对 TensorFlow 优化，对其他框架的支持可能需要额外的工作；同时，其主要面向 GPU 和谷歌的 TPU，其中间表示为深度学习算子级别的抽象，使其难以拓展到 RISC-V 存算一体加速器。

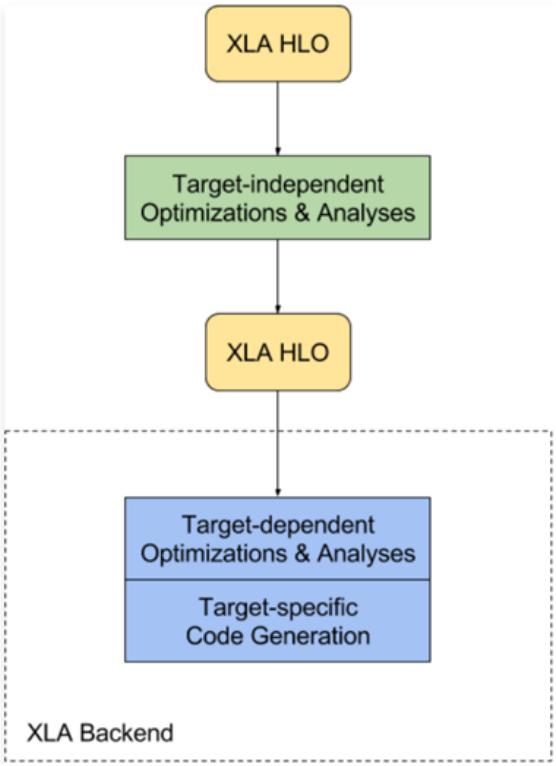

图 1.3 XLA 架构图

TVM（Tensor Virtual Machine）[2]：华盛顿大学陈天奇团队于 2018 年提出的面向人工智能异构加速器的编译器框架。其以 TensorFlow、PyTorch 或 ONNX 等 ML 框架导入模型，将模型编译为可链接对象模块，然后轻量级 TVM Runtime 可以用 C 语言的 API 来动态加载模型，也可以为 Python 和 Rust 等其他语言提供入口点。

在中间优化层级，其提出了 Relay IR[3] 和 Tensor IR[4] 两层中间表示来进行硬件无关（如常数折叠、算符融合等）、硬件相关（如计算模式识别与加速指令生成等）的优化。TVM 可以自动为多种硬件（包括 CPU、服务器 GPU、移动端 GPU 以及基于 FPGA 的加速器）来生成优化代码，支持端到端的学习优化，并且具备灵活的编译流程，但是 TVM 在面对新型加速器时，不但需要开发者根据芯片指令去扩展 TVM 中的 IR，还需要根据芯片的体系结构设计去添加定向优化策略，导致其扩展性较为有限。

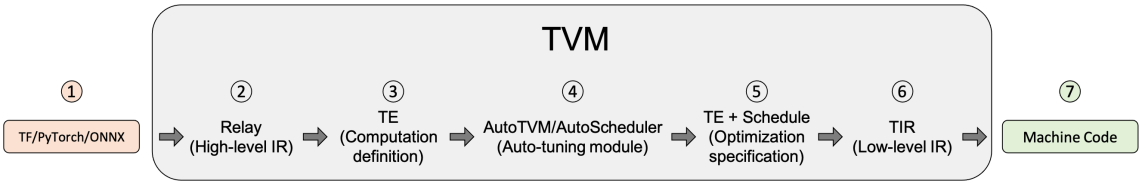

图 1.4　TVM 架构图

MindSpore AKG[5]：作为由华为主导开发并集成于其开源深度学习框架 MindSpore 中的深度学习编译器框架。AKG 可以接收来自 ML 框架的模型，生成针对特定硬件优化的内核。在中间优化层级，AKG 通过自动性能调优工具，自动生成优化的内核。同时 AKG 提供了自动化的调优过程，可以显著提高性能。然而，目前 AKG 主要针对华为的昇腾系列 AI 加速器和英伟达的 GPU 进行了优化支持，对于 RISC-V 存算一体异构芯片，其支持程度相对有限，适配性欠佳。

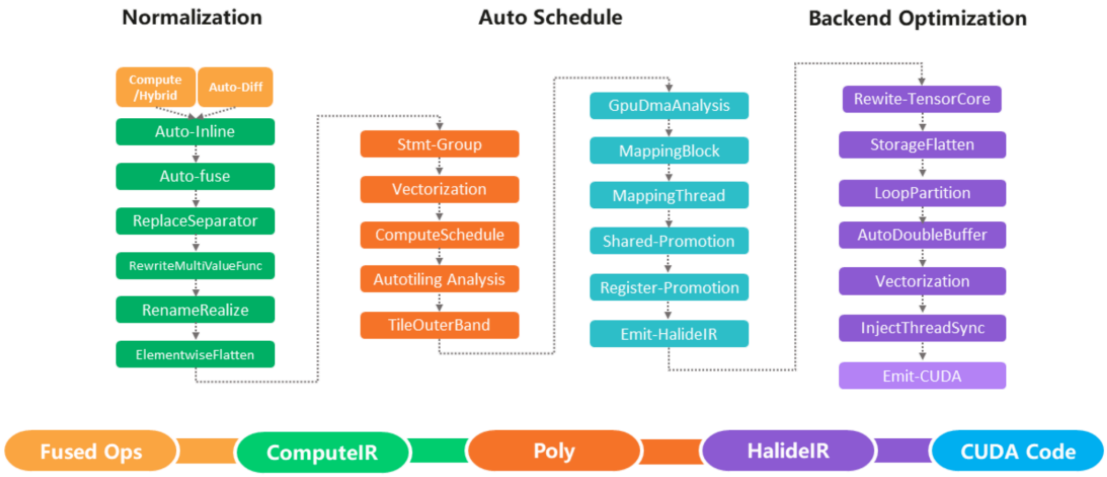

图 1.5　AKG 架构图

Triton[6]：OpenAI 于 2021 年推出的编译器，主要用于加速深度学习应用在 GPU 上执行效率。Triton 是一种 Python DSL，专门用于编写机器学习内核，支持 CPU、GPU 和 ASIC 等多种硬件平台，具备生成针对特定硬件优化内核的能力。在中间优化层级，Triton 编译器通过块级数据流分析技术，自动优化深度学习模型的执行过

程。不过，Triton 主要针对英伟达和 AMD 的 GPU 加速器进行优化，对于 RISC-V 存算一体异构芯片支持相对有限。

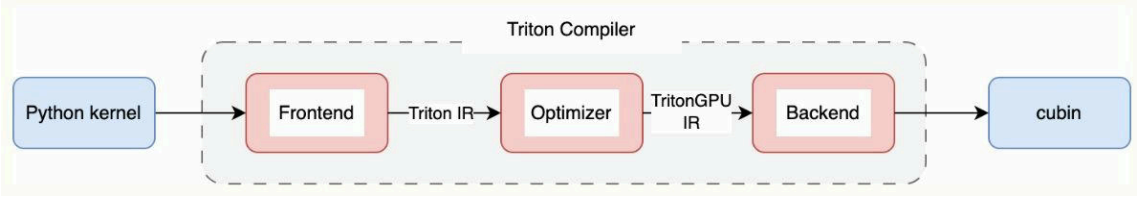

图 1.6　Triton 架构图

IREE[7]：Google 于 2019 年发布的一个开源的通用编译和运行时框架。通过输入高层次的机器学习模型，IREE 为各种硬件生成优化的可执行代码。在中间优化层级，IREE 利用 MLIR 进行多阶段优化，确保模型在目标平台上高效运行。IREE 提供了高性能的编译器后端，硬件抽象层允许轻松添加对新硬件的支持，但是 IREE 框架主要针对深度学习模型进行端到端的优化，而缺少统一编程模型。

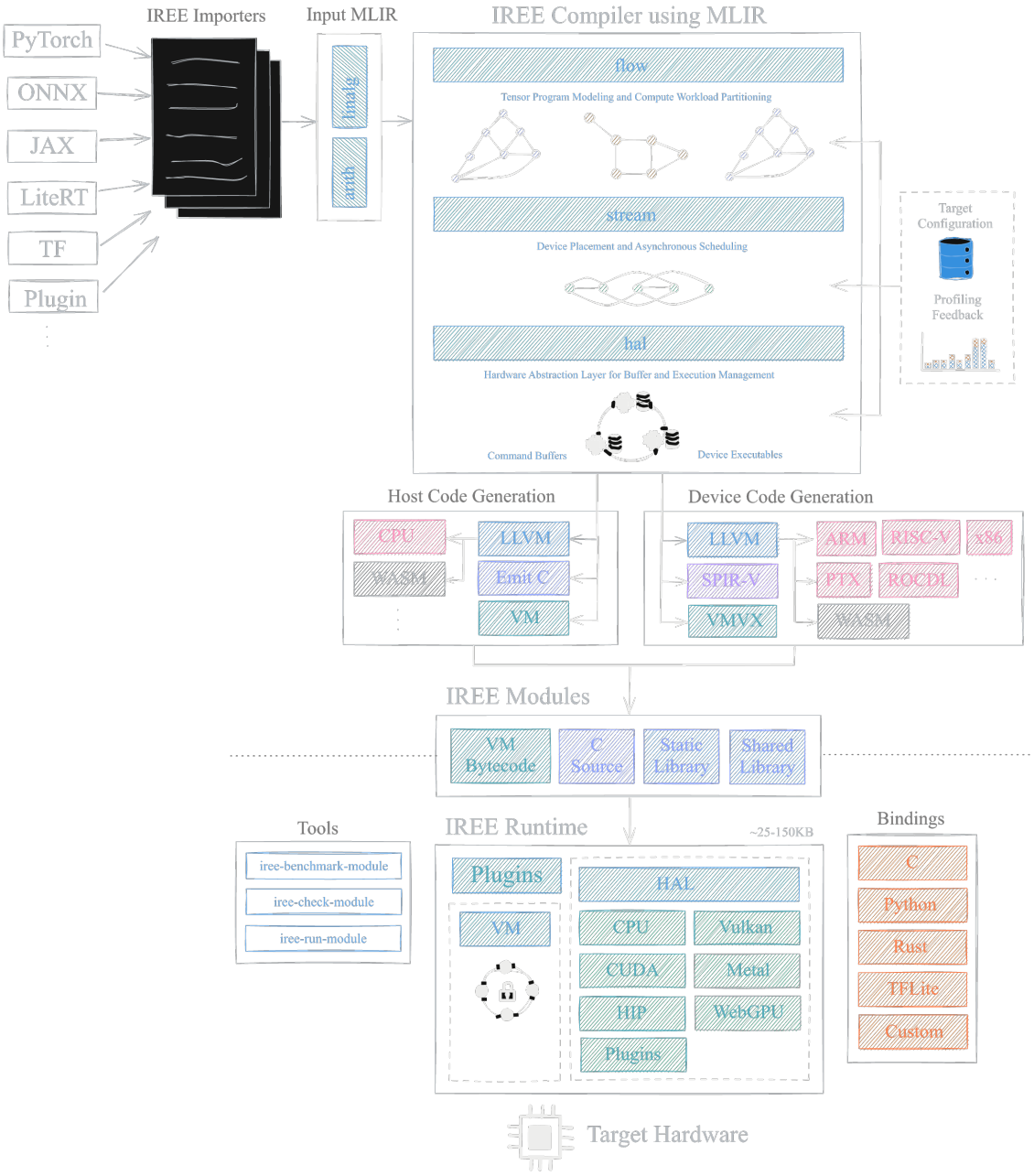

图 1.7　IREE 架构图

当下，众多深度学习编译器虽已面世，但依旧存在诸多局限性。例如，TVM 将算法定义与调度策略分离，而其调度策略需要手动编写或依赖现有模板调用，这对于缺乏编译优化专业知识的深度学习研究人员而言，使用难度较大。MindSpore AKG 能够自动生成优化的计算图，然而其优化重点在于为算子生成高性能 kernel，算子间的并行性尚未得到充分发掘等等。下表对上述深度学习编译器进行了对比总结。

表 1.1　国内外代表性工作总结

工作分类研究	核心思想	关键特征
TVM	将机器学习模型自动编译成可供不同硬件执行的机器语言	算子融合与图优化、量化技术、优化调度、Relay IR、代码生成和后端部署等
Triton	简化 GPU 上执行的复杂操作的开发，提供比 CUDA 更高的生产力	基于分块的编程范式、灵活的 DSL 以及自动性能调优。它允许用户编写高效的内核，同时不必关心底层硬件细节
XLA	将 TensorFlow 图编译成一系列专门为给定模型生成的计算内核，从而利用模型专属信息进行优化	操作融合、内存优化和专用内核生成
IREE	提供模块化和可扩展的编译器流水线，支持从高级中间表示到硬件特定执行的全流程	对不同硬件的兼容性、高效的内存管理以及对实时应用的支持
AKG	通过自动化的方式来探索不同的算法实现和调度策略，找到最优的执行方案	自动调优、多硬件支持和高性能内核生成

1.2.2　深度学习加速器

随着深度学习技术的广泛应用，为深度学习算法定制硬件加速器也成了学术界与工业界的研究热点，目前深度学习加速器主要朝两个方向发展。

其中一个是沿用传统的计算架构来提高硬件的加速性能，如 GPU、AISC、FPGA 等。寒武纪在 2014 年到 2016 年间陆续发表了 DIANNAO 系列论文[8]，提出了一系列全定制 AI 加速器的设计方案，使用多种深度学习加速算法；Google 于 2016 年提出一种以脉动阵列为计算核心加速矩阵运算的 AI 加速器 TPU[9]；同时 Yu-Hsin Chen 等人针对缓存与内存之间大量数据搬移问题设计了一种具有可重配置功能的深度学习加速器 Eyeriss[10]，主要通过行固定（Row Stationary，RS）等方法来降低数据搬运带来的延迟和能耗开销，之后又提出了一种用于紧凑神经网络模型的加速器 Eyeriss v2[11]；清华的 thinker 团队则提出了一种基于 CGRA 的可重构加速器[12]，该加速器可以通过对计算引擎单元阵列进行动态配置，实现以相同的硬件支持包括卷积在内的大多数神经网络运算。

加速器的另外一个发展方向是颠覆传统的冯诺依曼架构，

1.3 论文的主要研究工作

1.4 论文组织结构

第 2 章　主要技术基础

2.1　RISC-V

RISC-V 是一种 2010 年新兴的开源精简指令集架构。它的出现意图解决现有的指令集结构（如 x86、ARM、MIPS 等）的不合理设计。相较而言，其开源特性和模块化的架构保证了设计的灵活性和高效性，以满足各种不同应用场景。架构指令集方面，RISC-V 除标准功能设计指令外，包含实现多个不同功能的可选扩展指令。设计人员可以根据实际设计要求选择基础指令集和多个扩展指令集组合，并结合硬件平台组件扩展处理器的功能范围。

RISC-V 共有 5 种基础指令集[13]，指令空间涵盖不同位宽的指令格式，分别是弱内存次序指令集（RVWMO）、32 位整数指令集（RV32I）、32 位嵌入式整数指令集（RV32E）、64 位整数指令集（RV64I）、128 位整数指令集（RV128I）。在基础指令集的基础上 RISC-V 通过对指令集的架构设计的冗余指令进行分类，以提供扩展非标准架构指令的能力，为更专业的硬件提供设计余量。它为处理器设计中的特殊领域结构预留了指令编码空间，用户可以方便地扩展指令子集。如 图 8 所示，RISC-V 体系结构在 32/64 位指令中保留 4 组自定义指令类型，分别是 Custom-0、Custom-1、Custom-2/rv128、Custom-3/rv128。

inst[4:2] inst[6:5]	000	001	010	011	100	101	110	111 (>32b)
00	LOAD	LOAD-FP	*Custom-0*	MISC-MEM	OP-IMM	AUIPC	OP-IMM-32	48*b*
01	STORE	STORE-FP	*Custom-1*	AMO	OP	LUI	OP-32	64*b*
10	MADD	MSUB	NMSUB	NMADD	OP-FP	*reserved*	*custom-2/rv128*	48*b*
11	BRANCH	JALR	*reserved*	JAL	SYSTEM	*reserved*	*custom-3/rv128*	≥80*b*

图 2.8　RISC-V 指令集格式

根据 RISC-V 体系结构说明，用户自定义指令空间 custom-0 和 custom-1 被保留，不会用做标准扩展指令。而标记为 custom-2/rv128 和 custom-3/rv128 的操作码保留供未来 rv128 使用，标准扩展也会回避使用，以供用户进行指令扩展。

2.2　LLVM 编译器

LLVM[14] 是一个开源的编译器基础设施项目，它以“Low-Level Virtual Machine”的缩写命名，尽管名称中包含了“虚拟机”一词，但 LLVM 不仅仅是一个虚拟

机，而是一个综合的编译器工具链。LLVM 提供了一套通用的工具和库，用于开发编译器、优化器、代码生成器等。LLVM 的核心思想是基于中间表示（Intermediate Representation，IR），它定义了一种与机器和语言无关的中间代码表示形式。LLVM IR 是一种低级别的静态单赋值（Static Single Assignment，SSA）形式，它使用基本块和指令的层次结构来表示程序的结构和行为。

2.2.1 LLVM 结构

LLVM 框架主要由前端、中端、后端三大部分组成：

前端（Front End）阶段负责将高级编程语言（如 C、C++、Objective-C、Swift 等）的源代码转换为 LLVM 中间表示（LLVM IR）。这一过程涉及词法分析、语法分析、语义分析等操作，把高级语言的代码解析成编译器能够理解和处理的形式。

中端（Middle End）阶段主要对 LLVM IR 进行优化处理，目的是提高代码的质量和执行效率。优化操作包括但不限于消除无用代码、常量折叠、公共子表达式消除、循环优化等等。中端的优化是与目标硬件平台无关的，它只关注 LLVM IR 本身的优化，不涉及具体的机器指令生成。

后端（Back End）阶段将经过优化的 LLVM IR 转换为目标硬件平台能够执行的机器码。后端需要了解目标硬件的指令集架构、寄存器分配、内存布局等细节，根据这些信息将 LLVM IR 映射为相应的机器指令。同时，后端也会进行一些与硬件相关的优化，如指令调度、寄存器分配优化等，以充分发挥目标硬件的性能。LLVM 后端支持多种不同的硬件平台，包括 x86 架构的处理器、ARM 架构的处理器、PowerPC、MIPS、RISC-V 等，还包括一些新兴的专用硬件加速器。

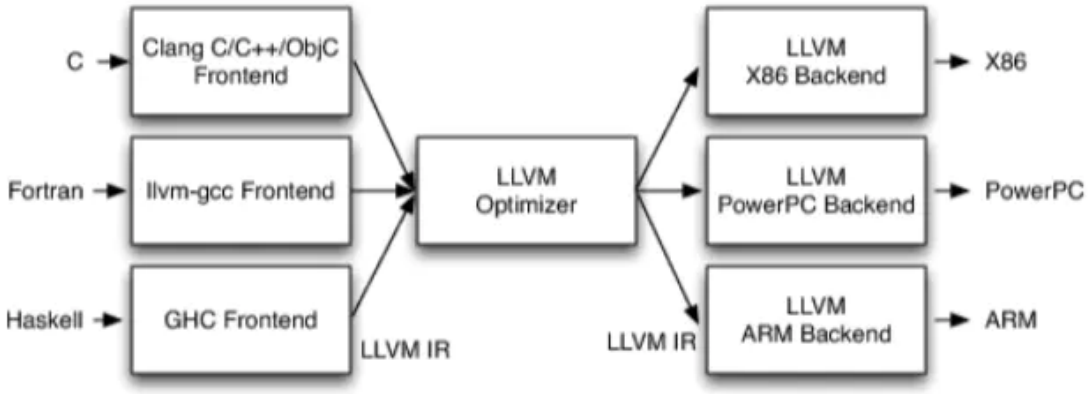

图 2.9　LLVM 编译器的结构

可以看到，若需引入新的编程语言，仅需开发相应的前端，让前端能够生成 LLVM IR 结构，就可以利用 LLVM 框架的相关优化。若要使编译器支持新型硬件设备，只需针对该硬件架构实现一个 LLVM 后端，将 LLVM 的中间表示（IR）转换为目标设备的机器码即可。

2.2.2

2.3 计算图优化研究

优先选用环保建材，减少对环境的污染。推广使用可再生能源，如太阳能、风能等，降低能源消耗。施工过程中应采取有效措施控制扬尘、噪音、废水等污染，最大限度减少对周边环境的影响。建立健全凌霄宝殿环境保护管理制度，明确责任主体，加强日常巡查和维护，及时发现和处理环境问题。推广使用节能环保设备，减少能源消耗和污染物排放。

注重生态修复与景观营造相结合，在凌霄宝殿周边建设生态绿地、湿地公园等，增加绿化面积，提升环境质量。选择适宜的植物种类，构建稳定的生态系统，发挥其净化空气、调节气候、美化环境等功能。建立完善的环境监测体系，定期对凌霄宝殿周边环境进行监测，评估环境质量变化趋势，为环境管理提供科学依据。加强环境保护宣传教育，提高天界居民的环保意识，鼓励公众积极参与凌霄宝殿环境保护工作。

Let a, b, and c be the side lengths of right-angled triangle. Then, we know that:

$$a^2 + b^2 = c^2 \tag{2.1}$$

Prove by induction:

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2} \tag{2.2}$$

2.4 本章小结

本章主要描述了论文所涉及的相关技术基础，首先介绍了深度学习相关的基本概念，如

第 3 章　计算图算子自动调度器

第 4 章　基于 RISC-V 存算一体芯片的编译器后端设计

第 5 章　编译器测试与分析

第 6 章　总结与展望

6.1　本文的工作总结

6.2　未来的工作展望

致　谢

时光荏苒，转眼间我的大学本科生活即将画上句号。回首这四年的点点滴滴，心中充满了无尽的感慨与思绪。在毕业论文完成之际，我愿将这四年的经历与感悟凝聚成文字，向求学路上给予我帮助的师长和亲友表达我最真挚的谢意。

师恩如海，深不可测。首先，我要特别感谢我的导师菩提教授。从初入大学时的懵懂无知，到如今能够独立完成毕业设计，菩老师始终是我前行路上的明灯。他不仅在学术上给予我悉心的指导，帮助我拓宽视野，提升能力，还在生活中给予我无微不至的关怀，让我感受到如家人般的温暖。在这次毕业设计的过程中，从选题到实验，从撰文到定稿，菩老师的全程指导让我受益匪浅。每一次对实验结果的精益求精，每一次对论文的反复修改，都让我深刻体会到菩老师在科研工作中的严谨态度和对学生的严格要求。在师门的四年时光里，菩老师不仅传授给我学术知识，更教会了我踏实、认真、负责、勤勉的品质，这些品质将伴随我一生，无论是在科研还是其他工作中，甚至在日常生活中。在此论文完成之际，我衷心感谢菩老师一路以来的教导、呵护与关怀。

参考文献

[1] SABNE A. XLA : Compiling Machine Learning for Peak Performance[Z]2020

[2] CHEN T, MOREAU T, JIANG Z, 等. TVM: An Automated End-to-End Optimizing Compiler for Deep Learning[C/OL]//13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)Carlsbad, CAUSENIX Association, 2018: 578-594. https://www.usenix.org/conference/osdi18/presentation/chen

[3] ROESCH J, LYUBOMIRSKY S, WEBER L, 等. Relay: a new IR for machine learning frameworks[C/OL]//Proceedings of the 2nd ACM SIGPLAN International Workshop on Machine Learning and Programming LanguagesPhiladelphia, PA, USAAssociation for Computing Machinery, 2018: 58-68. https://doi.org/10.1145/3211346.3211348. DOI:10.1145/3211346.3211348

[4] FENG S, HOU B, JIN H, 等. TensorIR: An Abstraction for Automatic Tensorized Program Optimization[C/OL]//Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2Vancouver, BC, CanadaAssociation for Computing Machinery, 2023: 804-817. https://doi.org/10.1145/3575693.3576933. DOI:10.1145/3575693.3576933

[5] ZHAO J, LI B, NIE W, 等. AKG: automatic kernel generation for neural processing units using polyhedral transformations[C]//PLDI 2021: Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation2021

[6] TILLET P, KUNG H T, COX D. Triton: an intermediate language and compiler for tiled neural network computations[C/OL]//Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming LanguagesPhoenix, AZ, USAAssociation for Computing Machinery, 2019: 10-19. https://doi.org/10.1145/3315508.3329973. DOI:10.1145/3315508.3329973

[7] iree[Z]https://github.com/iree-org/iree

[8] CHEN Y, CHEN T, XU Z, 等. DianNao family: energy-efficient hardware accelerators for machine learning[J/OL]Commun. ACM, 2016, 59(11): 105-112. https://doi.org/10.1145/2996864. DOI:10.1145/2996864

[9] JOUPPI N P, YOUNG C, PATIL N, 等. In-Datacenter Performance Analysis of a Tensor Processing Unit[C/OL]//Proceedings of the 44th Annual International Symposium on Computer ArchitectureToronto, ON, CanadaAssociation for Computing Machinery, 2017: 1-12. https://doi.org/10.1145/3079856.3080246. DOI:10.1145/3079856.3080246

[10] CHEN Y H, KRISHNA T, EMER J S, 等. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks[J/OL]IEEE Journal of Solid-State Circuits, 2017, 52(1): 127-138. DOI:10.1109/JSSC.2016.2616357

[11] CHEN Y H, YANG T J, EMER J, 等. Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices[EB/OL](2019). https://arxiv.org/abs/1807.07928

[12] YIN S, OUYANG P, TANG S, 等. A High Energy Efficient Reconfigurable Hybrid Neural Network Processor for Deep Learning Applications[J/OL]IEEE Journal of Solid-State Circuits, 2018, 53(4): 968-982. DOI:10.1109/JSSC.2017.2778281

[13] CUI E, LI T, WEI Q. RISC-V Instruction Set Architecture Extensions: A Survey[J/OL]IEEE Access, 2023, 11: 24696-24711. DOI:10.1109/ACCESS.2023.3246491

[14] LATTNER C, ADVE V. LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation[C]//Proceedings of the International Symposium on Code Generation and Optimization: Feedback-Directed and Runtime OptimizationPalo Alto, CaliforniaIEEE Computer Society, 2004: 75