基于 ARM 架构的 M1 并行微架构的分析

张怡桢

(南开大学 软件学院,天津 滨海新区 300457)
通讯作者: 张怡桢, E-mail: 2013747@mail.nankai.edu.cn

摘 要: 本文是一篇很基础的科普调研文章，从计算机 CPU 运行的最底层的指令集架构这一编译器与软件开发服务开始分析，到对负责具体实现指令集架构的硬件实现微架构的了解，再把握 Apple Silicon 芯片家族的脉络，然后，从以上的三个角度来分析基于 ARM 架构的 M1 并行微架构，同时与基于 X86 架构的 Intel 的 Skylake 与 AMD 的 Zen3 进行对比，最后，从成本兼容性与架构转换等角度分析 Apple 此次的 Mac 架构转换改革。
关键词: 指令集架构; 微架构; Apple Silicon；M1 芯片；并行微架构；Skylake；Zen3；架构转换
中图法分类号: TP311

Analysis of M1 parallel microarchitecture based on ARM architecture

ZHANG Yi-Zhen

(School of Software, Nankai University, Binhai New Area, Tianjin, 300457)
Corresponding author: Zhang Yizhen, E-mail: 2013747@mail.nankai.edu.cn

Abstract: This article is a very basic science research article, starting from the compiler and software development services of the lowest instruction set architecture running on the computer CPU, to the understanding of the hardware implementation microarchitecture responsible for the specific implementation of the instruction set architecture, and then grasping the context of the Apple Silicon chip family, and then, from the above three perspectives, analyze the M1 parallel microarchitecture based on the ARM architecture. At the same time, it is compared with Intel's Skylake and AMD's Zen3 based on X86 architecture, and finally, Apple's Mac architecture conversion reform is analyzed from the perspective of cost compatibility and architecture conversion.

Key words: instruction set architecture; Microarchitecture; Apple Silicon；M1 chip; Parallel microarchitecture; Skylake；Zen3；Schema transformation

Apple M1 是由苹果公司（Apple）研发的处理器芯片，于 2020 年 11 月 11 日在苹果新品发布会上发布，适用于部分 Mac、iPad 设备。Apple M1 的制程为 5 纳米，内置 8 核 CPU，集成 4 个高性能大核心以及 4 个高效能小核心；此外，Apple M1 还内置了 8 核 GPU（部分机型为 7 核）以及神经网络引擎。

M1 采用 ARM 指令集架构，不同于市面上流行于桌面级的 X86 指令集架构，ARM 指令集架构多用于移动端与嵌入式。

M1 的微架构为基于 ARM 的大核 Firestorm，小核 Icestorm。

M1 的 CPU 的工作流程是流水线特征的，其本质上是一个乱序执行，超标量，多流水线的 CPU，M1 采用统一内存架构，达到低延迟的目的。

Apple 对于生态的把握与控制权，让其可以在处理器研发时，在成本，性能，功耗这三点上找到相对高的平衡点。

从 X86 架构转到 ARM 架构时，软件的适配在架构的过渡中是非常重要的，Rosetta 2 是 M1 Mac 的一个关键转译工具。

1 指令集架构

指令集架构（英语：Instruction Set Architecture，缩写为 ISA），又称指令集或指令集体系，是计算机体系结构中与程序设计有关的部分，包含了基本数据类型，指令集，寄存器，寻址模式，存储体系，中断，异常处理以及外[I/O。指令集架构包含一系列的 opcode 即操作码（机器语言），以及由特定处理器执行的基本命令。

在计算机中，指示计算机硬件执行某种运算、处理功能的命令称为指令。指令是计算机运行的最小的功能单位，而硬件的作用是完成每条指令规定的功能。一台计算机上全部指令的集合，就是这台计算机的指令系统。指令系统也称指令集，是这台计算机全部功能的体现。从现阶段的主流体系结构来讲，指令集可分为复杂指令集（CISC）与精简指令集（RISC）。

目前市场上四大主流指令集为 X86、MIPS、ARM、RISC-V。下图源于 OFweek 电子工程网整理自公开资料，其中详细指出了各个主流指令集的运营者，特点与代表生产商。

芯片架构	运营者	特点	代表厂商
X86	Intel公司	功能强大，通用性、兼容性、与实用性强	Intel、AMD
ARM	Acorn公司	低功耗、低成本	苹果、谷歌、IBM、华为
RISC-V	RISC-V基金会	完全开源、架构简单、易于移植、模块化设计、完整的工具链	三星、英伟达、西部数据
MIPS	MIPS公司	简洁、优化、具有高度扩展性、包含大量的寄存器、指令数和字符、可视的管道延时时隙	龙芯中科

图 1: 各个主流指令集的运营者，特点与代表生产商

其中 ARM、MIPS、RISC-V 是基于 RISC 精简指令集，而 X86 则是基于 CISC 复杂指令集。

x86 架构是基于 8086 处理器执行的计算机语言指令集。早期的 x86（含 x64）是 cisc 的代表，后来的发展中逐步引入了 risc 的部分理念，将内部指令的实现大量模块化，准确来说是一个 cisc 外加 risc 部分技术的架构。

而 arm 是 risc 的典型代表，不过在发展过程中引入了部分复杂指令（完全没有复杂指令的话操作系统跑起来异常艰难），所以 arm 是一个 risc 基础外加 cisc。

在 PC 端，最主要的就是 X86 的处理器，而移动端就要属 ARM 的天下了。在英伟达要收购 ARM 的消息确认后，基于 ARM 的 CPU 设计公司担心未来架构授权问题，开源的 RISC-V 走向了 CPU 的舞台，成为了各家 IC 设计公司的新宠。

1.1 X86架构

X86 是微处理器执行的计算机语言指令集，指一个 Intel 通用计算机系列的标准编号缩写，也标识一套通用的计算机指令集合。1978 年 6 月 8 日，Intel 发布了新款 16 位微处理器 8086，也同时开创了一个新时代：X86 架构诞生了。

X86 指令集是美国 Intel 公司为其第一块 16 位 CPU（i8086）专门开发的，美国 IBM 公司 1981 年推出的世界第一台 PC 机中的 CPU﹣i8088（i8086 简化版）使用的也是 X86 指令。

随着 CPU 技术的不断发展，Intel 陆续研制出更新型的 i80386、i80486 直到今天的 Pentium 4 系列，但为了保证电脑能继续运行以往开发的各类应用程序以保护和继承丰富的软件资源，所以 Intel 公司所生产的所有 CPU 仍然继续使用 X86 指令集。

1.2 ARM架构

ARM 架构是一个 32 位精简指令集处理器架构，其广泛地使用在许多嵌入式系统设计。由于节能的特点，ARM 处理器非常适用于移动通讯领域，符合其主要设计目标为低耗电的特性。

如今，ARM 家族占了所有 32 位嵌入式处理器 75%的比例，使它成为占全世界最多数的 32 位架构之一。ARM 处理器可以在很多消费性电子产品上看到，从可携式装置到电脑外设甚至在导弹的弹载计算机等军用设施中都有它的存在。

ARM 和 X86 架构最显著的差别是使用的指令集不同。

序号	架构	特点
1	ARM	主要是面向 移动 、 低功耗领域 ，因此在设计上更偏重 节能 、 能效 方面
2	X86	主要面向 家用 、 商用 领域，在 性能 和 兼容性 方面做得更好

图 2:ARM 与 X86 的特点

1.3 RISC-V

RISC-V 架构是基于精简指令集计算（RISC）原理建立的开放指令集架构（ISA），RISC-V 是在指令集不断发展和成熟的基础上建立的全新指令。RISC-V 指令集完全开源，设计简单，易于移植 Unix 系统，模块化设计，完整工具链，同时有大量的开源实现和流片案例，得到很多芯片公司的认可。

RISC-V 架构的起步相对较晚，但发展很快。它可以根据具体场景选择适合指令集的指令集架构。基于 RISC-V 指令集架构可以设计服务器 CPU，家用电器 CPU，工控 CPU 和用在比指头小的传感器中的 CPU。

1.4 MIPS架构

MIPS 架构是一种采取精简指令集（RISC）的处理器架构，1981 年出现，由 MIPS 科技公司开发并授权，它是基于一种固定长度的定期编码指令集，并采用导入/存储（Load/Store）数据模型。经改进，这种架构可支持高级语言的优化执行。其算术和逻辑运算采用三个操作数的形式，允许编译器优化复杂的表达式。

如今基于该架构的芯片广泛被使用在许多电子产品、网络设备、个人娱乐装置与商业装置上。最早的 MIPS 架构是 32 位，最新的版本已经变成 64 位。

2 微架构

计算机架构(Computer Architecture)主要指的是指令集架构(Instruction Set Architecture)，而微架构(Micro Architecture)指的是集成电层面的架构。前者主要为编译器和软件开发服务，后者负责具体的硬件实现。

使用不同微架构的电脑可以共享一种指令集。例如，Intel 的 Pentium 和 AMD 的 AMD Athlon，两者几乎采用相同版本的 x86 指令集体系，但是两者在内部设计上有本质的区别。

微架构（英语：microarchitecture），也被叫做计算机组织，微架构使得指令集架构（ISA）可以在处理器上被执行。指令集架构可以在不同的微架构上执行。

下图为[Intel 80286]的微架构图:

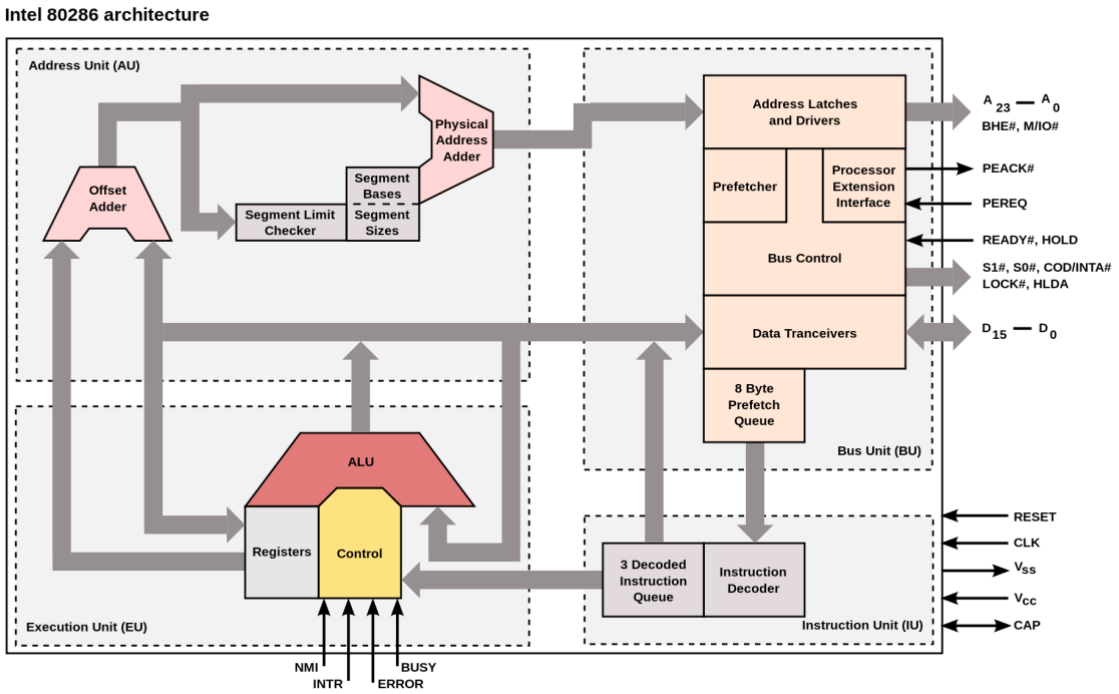


图 3: [Intel 80286]的微架构图

在 PC 端上，市面上典型的微架构生产公司有 Intel 与 AMD，Intel 公司在售的主流处理器系列有 Core，Pentium, Celeron 等系列，而 AMD 则有 K 系列与 Zen 系列等，都是基于 X86 指令集架构且属于 CISC。而在移动端与嵌入式上，则几乎是 ARM 指令集架构的天下。

2.1 AMD

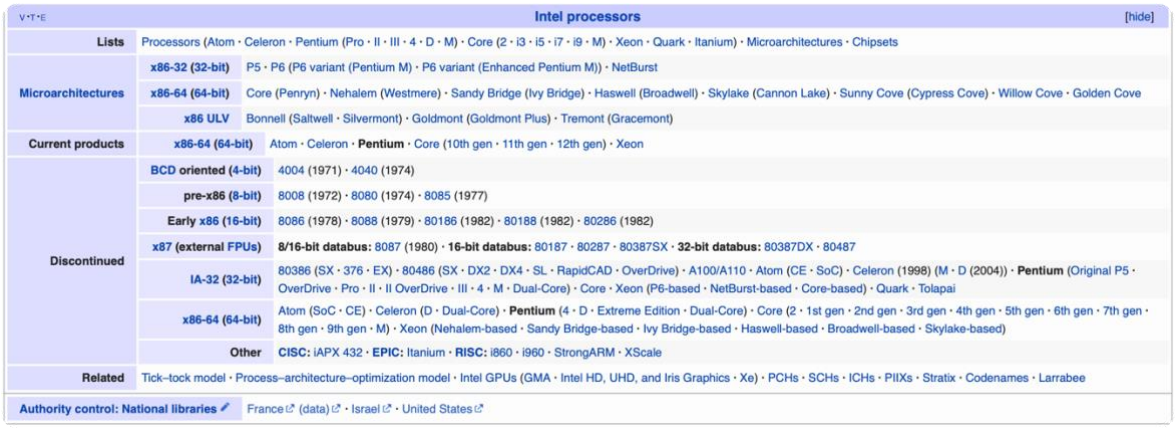
如图为 AMD CPU 的处理器列表，其中，可以看到 AMD 公司基于 ARM 指令集架构下的微架构仅有 K12，而 AMD 公司主打的 Zen 系列则是基于 X86-64 指令集架构的微架构。

v·t·e	AMD processors [hide]	
Lists	Processors (K5 · K6 · Athlon (XP · 64 · X2 · II) · Phenom · FX · Ryzen · Duron · Sempron · Turion · Opteron · Epyc) · APUs · Microarchitectures · Chipsets	
Microarchitectures	IA-32 (32-bit)	K5 · K6 · Athlon/K7
	x86-64 desktop	K8 · K9 · K10 (aka 10h) · 15h (Bulldozer · Piledriver · Steamroller · Excavator) · Zen (1st gen · Zen+ · Zen 2 · **Zen 3** · Zen 3+ · Zen 4 · Zen 5)
	x86-64 low-power	Bobcat (aka 14h) · 16h (Jaguar · Puma)
	ARM64	K12 (aka 12h)
Current products	x86-64 (64-bit)	Athlon · Ryzen · Threadripper · Epyc
Discontinued	Early x86 (16-bit)	Am286
	IA-32 (32-bit)	Am386 · Am486 · Am5x86 · K5 · K6 · K6-2 · K6-III · Athlon (XP · MP) · Duron · Geode
	x86-64 (64-bit)	Athlon 64 (X2 · II) · Sempron · Turion · Phenom (II) · FX · Opteron · A-series APUs
	Other	Am9080 · Am2900 (list) · Am29000 · Alchemy (MIPS32)
Italics indicates an upcoming architecture.		

图 4: AMD CPU 的处理器列表

2.2 Intel

一开始 Intel 公司便在 X86 指令集架构上陆续开发，由于生态的原因即为了保证电脑能继续运行以往开发的各类应用程序以保护和继承丰富的软件资源，所以 Intel 公司所生产的所有 CPU 仍然继续使用 X86 指令集。下图为 Intel 公司的处理器情况表。



v·t·e	Intel processors	[hide]
Lists	Processors (Atom · Celeron · Pentium (Pro · II · III · 4 · D · M) · Core (2 · i3 · i5 · i7 · i9 · M) · Xeon · Quark · Itanium) · Microarchitectures · Chipsets	
Microarchitectures	x86-32 (32-bit)	P5 · P6 (P6 variant (Pentium M) · P6 variant (Enhanced Pentium M)) · NetBurst
	x86-64 (64-bit)	Core (Penryn) · Nehalem (Westmere) · Sandy Bridge (Ivy Bridge) · Haswell (Broadwell) · Skylake (Cannon Lake) · Sunny Cove (Cypress Cove) · Willow Cove · Golden Cove
	x86 ULV	Bonnell (Saltwell · Silvermont) · Goldmont (Goldmont Plus) · Tremont (Gracemont)
Current products	x86-64 (64-bit)	Atom · Celeron · **Pentium** · Core (10th gen · 11th gen · 12th gen) · Xeon
Discontinued	BCD oriented (4-bit)	4004 (1971) · 4040 (1974)
	pre-x86 (8-bit)	8008 (1972) · 8080 (1974) · 8085 (1977)
	Early x86 (16-bit)	8086 (1978) · 8088 (1979) · 80186 (1982) · 80188 (1982) · 80286 (1982)
	x87 (external FPUs)	**8/16-bit databus:** 8087 (1980) · **16-bit databus:** 80187 · 80287 · 80387SX · **32-bit databus:** 80387DX · 80487
	IA-32 (32-bit)	80386 (SX · 376 · EX) · 80486 (SX · DX2 · DX4 · SL · RapidCAD · OverDrive) · A100/A110 · Atom (CE · SoC) · Celeron (1998) (M · D (2004)) · **Pentium** (Original P5 · OverDrive · Pro · II · II OverDrive · III · 4 · M · Dual-Core) · Core · Xeon (P6-based · NetBurst-based · Core-based) · Quark · Tolapai
	x86-64 (64-bit)	Atom (SoC · CE) · Celeron (D · Dual-Core) · **Pentium** (4 · D · Extreme Edition · Dual-Core) · Core (2 · 1st gen · 2nd gen · 3rd gen · 4th gen · 5th gen · 6th gen · 7th gen · 8th gen · 9th gen · M) · Xeon (Nehalem-based · Sandy Bridge-based · Ivy Bridge-based · Haswell-based · Broadwell-based · Skylake-based)
	Other	**CISC:** iAPX 432 · **EPIC:** Itanium · **RISC:** i860 · i960 · StrongARM · XScale
Related	Tick–tock model · Process–architecture–optimization model · Intel GPUs (GMA · Intel HD, UHD, and Iris Graphics · Xe) · PCHs · SCHs · ICHs · PIIXs · Stratix · Codenames · Larrabee	
Authority control: National libraries	France (data) · Israel · United States	

图 5: Intel CPU 的处理器列表

2.3 ARM

ARM 是 Advanced RISC Machine 的缩写，即进阶精简指令集机器。arm 更早称为 Acorn RISC Machine，是一个 32 位精简指令集（RISC）处理器架构。也有基于 ARM 设计的派生产品，主要产品包括 Marvell 的 XScale 架构和和德州仪器的 OMAP 系列。ARM 家族中 32 位嵌入式处理器占比达 75%，由于 ARM 的低功耗特性，被广泛反应于移动通信领域、便携式设备等领域。

1983 年 Acorn 电脑公司（Acorn Computers Ltd）开始开发一颗主要用于路由器的 Conexant ARM 处理器，由 Roger Wilson 和 Steve Furber 带领团队，着手开发一种新架构，类似进阶的 MOS Technology 6502 处理器。Acorn 有一大堆建构在 6502 架构上的电脑。该团队在 1985 年时开发出 ARM1 Sample 版，并于次年量产了 ARM2，ARM2 具有 32 位的数据总线、26 位的寻址空间，并提供 64 Mbyte 的寻址范围与 16 个 32-bit 的暂存器。

在 1980 年代晚期，苹果电脑开始与 Acorn 合作开发新版的 ARM 核心。1990 年将设计团队另组成一间名为安谋国际科技（Advanced RISC Machines Ltd.）的新公司,。1991 年首版 ARM6 出样，然后苹果电脑使用 ARM6 架构的 ARM 610 来当作他们 Apple Newton PDA 的基础。在 1994 年，Acorn 使用 ARM 610 做为他们 Risc PC 电脑内的 CPU。

ARM 是一家微处理器行业的知名企业，该企业设计了大量高性能、廉价、耗能低的 RISC （精简指令集）处理器，它只设计芯片而不生产。ARM 的经营模式在于出售其知识产权核（IP core），将技术授权给世界上许多著名的半导体、软件和 OEM 厂商，并提供技术服务。

ARM 的版本分为两类，一个是内核版本，一个处理器版本。内核版本也就是 ARM 架构，如 ARMv1、ARMv2、ARMv3、ARMv4、ARMv5、ARMv6、ARMv7、ARMv8 等。处理器版本也就是 ARM 处理器，如 ARM1、ARM9、ARM11、ARM Cortex-A（A7、A9、A15），ARM Cortex-M（M1、M3、M4）、ARM Cortex-R，这个也是我们通常意义上所指的 ARM 版本。

ARM 版本信息简化表如下表所示。

详细的 ARM 微架构发展见维基百科 List of ARM processors

arm 芯片历史

架构	处理器家族
ARMv1	ARM1
ARMv2	ARM2、ARM3
ARMv3	ARM6、ARM7
ARMv4	StrongARM、ARM7TDMI、ARM9TDMI
ARMv5	ARM7EJ、ARM9E、ARM10E、XScale
ARMv6	ARM11、ARM Cortex-M
ARMv7	ARM Cortex-A、ARM Cortex-M、ARM Cortex-R
ARMv8	Cortex-A35、Cortex-A50系列、Cortex-A72、Cortex-A73

图 6：arm 芯片家族

2.3.1 ARM7 系列

该系列主要针对某些简单的 32 位设备，作为目前较旧的一个系列，ARM7 处理器已经不建议继续在新品中使用。主要包括 ARM7TDMI-S（ARMv4T 架构）和 ARM7EJ-S（ARMv5TEJ 架构）。

2.3.2 ARM9 系列

主要针对嵌入式实时应用，主要包括 ARM926EJ-S、ARM946E-S 和 ARM968E-S。

2.3.3 ARM11 系列

主要应用在高可靠性和实时嵌入式应用领域，主要包括 ARM11MPCore、ARM1176、ARM1156、ARM1136。

2.3.4 Cortex-R 系列

Cortex-R，代表实时的意义（Real-Time），目标是实时任务处理，主要应用领域包括汽车、相机、工业、医学等。

该系列处理器主要包括 Cortex-R4、Cortex-R5、Cortex-R7、Cortex-R8、Cortex-R52、Cortex-A17。

2.3.5 Cortex-M 系列

Cortex-M，代表微处理器的意义（Microcontrollers），目标是最节能的嵌入式设备，主要应用领域包括汽车、能源网、医学、嵌入式、智能卡、智能设备。传感器融合、穿戴设备等。

该系列处理器主要包括 Cortex-M0、Cortex-M0+、Cortex-M3、Cortex-M4、Cortex-M7、Cortex-M23、Cortex-M33、Cortex-M35P。

2.3.6 Cortex-A 系列

Cortex-A，代表的是先进意义（Advanced），目标是以最佳功耗实现最高性能，主要应用领域包括汽车、工业、医学、调制解调器、存储等。Cortex-A 也是目前应用最广的处理器版本。

Cortex-A 处理器从高到低可排序为：Cortex-A73、Cortex-A72、Cortex-A57、Cortex-A53、Cortex-A35、Cortex-A32、Cortex-A17、Cortex-A15、Cortex-A7、Cortex-A9、Cortex-A8、Cortex-A5。

目前国产的 CPU 以及华为的手机麒麟手机芯片和海思芯片等都是基于 ARM V8 架构的，也是 cortex-A 系列。可以说在移动便携式领域设备，ARM 几乎全部覆盖。

3 Apple Silicon 芯片家族

除了 ARM 公司基于 ARM 内核架构版本设计的微架构，还有其他的公司基于 ARM 内核版本开发设计自

己的微架构，其中较为著名的是 Apple 公司也基于 ARM 指令集架构开发设计自己的微架构，在开发的微架构基础上，进而进行芯片设计或封装设计从而推出的“Apple Silicon”芯片家族。

Apple 芯片是一系列由 Apple Inc. 设计的芯片系统 (SoC) 和封装系统 (SiP) 处理器，其中应用于 iphone 上的 A 系列芯片，是 iphone 高性能以及畅销的主要原因，在 2020 年，基于 Apple Silicon 芯片家族的生态系统，Apple 推出了是适用于 PC 端的 M 系列芯片，实现了 ipad，iphone，macbook 三端的 ARM 指令集架构大一统，对软件开发而言，可以很省心的移植应用。

下图是 Apple 公司的 Apple Silicon 芯片家族：

v·t·e	Apple silicon	[hide]
A series	A4 · A5 · A5X · A6 · A6X · A7 · A8 · A8X · A9 · A9X · A10 Fusion · A10X Fusion · A11 Bionic · A12 Bionic · A12X / A12Z Bionic · A13 Bionic · A14 Bionic · A15 Bionic · A16 Bionic	
H series	H1 · H2	
M series	M1 / M1 Pro / M1 Max / M1 Ultra · M2	
S series	S1 / S1P · S2 · S3 · S4 · S5 · S6 · S7 · S8	
T series	T1 · T2	
W series	W1 · W2 · W3	
U Series	U1	

图 7: Apple Silicon 芯片家族

其中 A 系列最为著名，应用于 iphone 以及 ipad 端，“A”系列（适用于苹果）是 iPhone、某些 iPad 机型和 Apple TV 中使用的 SoC 系列。“A”系列芯片也用于停产的 iPod Touch 系列和原始 HomePod。它们集成了一个或多个基于 ARM 的处理核心（CPU）、图形处理单元（GPU）、缓存内存和其他必要的电子设备，以便在单个物理包中提供移动计算功能。

H 系列应用于 AirPods，AirPods Pro，AirPods Max 等耳机系列产品，Apple“H”系列（适用于耳机）是 SoC 系列，具有低功耗音频处理和无线连接，可用于耳机。

M 系列为应用在 Mac 端即 PC 设备上的自研芯片，也应用于 ipad 端，Apple“M”系列（适用于 Mac）是 2020 年 11 月或之后用于 Mac 计算机的芯片系统（SoC）系列，2021 年 4 月或之后用于 iPad Pro 平板电脑，以及 2022 年 3 月或更新机型用于 iPad Air 平板电脑。

S 系列应用在 Apple Watch 系列上，即穿戴电子表设备端，Apple "S" 系列是 Apple Watch 中使用的一系列系统包装 (SiP)。它使用定制的应用程序处理器，与用于无线连接的内存、存储和支持处理器、传感器和 I/O 一起在单个软件包中形成一台完整的计算机。它们由苹果设计，并由三星等合同制造商制造。

T 系列是一个安全芯片，主要功能是应用在处理与加密生物识别信息等，自 2016 年以来，T 系列芯片在基于英特尔的 MacBook 和 iMac 计算机上作为安全飞地运行。该芯片处理和加密生物识别信息（触控 ID），并充当麦克风和 FaceTime 高清摄像头的看门人，保护它们免受黑客攻击。该芯片运行 bridgeOS，据称是 watchOS 的变体。T 系列处理器的功能内置在 M 系列 CPU 中，从而结束了对 T 系列的需求。

U 系列则是应用在无线充电以及无线充电盒的控制芯片，Apple“U”系列是实现超宽带（UWB）无线电的软件包（SiP）中的一系列系统。

W 系列一开始的 W1 应用于初代 Airpods 用于保持蓝牙连接以及解码音频流，而后耳机系列使用 H 系列的芯片，从 W2 开始之后 W 系列主要应用在 Watch 系列，集成到了 S 系列的芯片中，该系列芯片使得 WiFi 速度提高，并用于蓝牙连接，Apple“W”系列（用于*无线*）是一系列用于蓝牙和 Wi-Fi 连接的 RF SoC。

4 M1 芯片

了解完指令集架构，微架构以及 Apple Silicon 芯片家族的细节之后，我们分别从指令集架构角度以及微架构角度来看 Apple 在 2020 年新推出的 M 系列芯片 M1。

4.1 指令集架构

从指令集架构上，M1 实现的是 ARMv8-A 指令集。2011 年 10 月，Armv8-A 代表了 ARM 架构的根本

性变化横空出世。它添加了一个可选的 64 位架构，名为“AArch64”，以及相关的新“A64”指令集。AArch64 提供了与 Armv7-A（32 位架构）的用户空间兼容性，其中称为“AArch32”和旧的 32 位指令集（现在称为“A32”）。Thumb 指令集被称为“T32”，没有 64 位对应指令。Armv8-A 允许 32 位应用程序在 64 位操作系统中执行，32 位操作系统由 64 位虚拟机管理程序控制。ARM 于 2012 年 10 月 30 日宣布了他们的 Cortex-A53 和 Cortex-A57 内核。苹果是第一个在消费品中发布 Armv8-A 兼容核心的（iPhone 5S 中的 Apple A7）。

当然，随着 ARMv8-A 系列的升级，Apple A7 到 Apple A10X 都是基于初代的 ARMv8-A，而 Apple A11 是基于 ARMv8.2-A，Apple A12，A12X 以及 A12Z 基于 ARMv8.3-A，A13 基于 ARMv8.4-A，A14，A15，A16 以及 M1 与新发布的 M2 则是 ARMv8.5-A。

大体上，从 2012-2022 年，Apple 在 A 系列以及 M 系列上采用的指令集架构都是 64 位系统的 ARMv8-A。

当然，由于功耗低等优势问题，基于 RISC 的 ARM 指令集架构占据了移动端与嵌入式的市场，而在 PC 端，很长的一段时间，Intel 与 AMD 的 X86 指令集桌面级处理器是 PC 端的主流，那么为什么在 Apple 发布 M1 芯片之前，我们鲜少能在市面上看到优秀的 ARM 类的桌面级处理器呢？

其实并不是没有出现，只是软件兼容性太差，早在 2012 年的时候，微软就发布了第一代的原生 Windows 操作系统——Windows RT，并将其用在了初代 Surface 平板上，这台平板搭载了英伟达 Tegra 3 SoC，基于 ARM 架构，这便是这一切的鼻祖。但是在当时，这款电脑的评价并不好，因为它性能太差，而且当时的 Windows RT 受限于 ARM 兼容性，几乎什么都运行不了，所以很快它就沉沦了。

所以其实在 ARM 架构上的 PC 不是没有可能，而是基于 ARM 的软件兼容性，以及操作系统的问题，而 Apple 公司拥有自己的 Mac OS 生态，以及强大的号召力，让各大软件厂商开始正对 ARM 架构的系统进行软件的适配。因此从 M1 发布之后，以及随着时间的流逝，各大厂商纷纷对 ARM 系统进行软件适配之后，基于 ARM 架构的 PC 由于功耗低，续航好的先天优势，在获得生态与软件上的支持后，或许也能在如今的 X86 占主流的桌面端市场上分到一份蛋糕。

4.2 微架构

其实对于 Arm 指令集的应用于 PC 端的微架构，并不是没有公司研究过，2019 年 10 月 2 日微软发布的的 surface pro x 便是 arm 指令集的 pc，却由于是 arm 架构的且微软没有做出合适的转译办法，导致很长一段时间，surface pro x 并不能兼容运行 64 位的 X86 软件，有着极其糟糕的生态与体验，也就是在 2020 年 M1 发布的前几天才刚刚兼容 64 位的 X86 软件，且依然有着许多限制， 微软和高通的合作定制的这款用着 A76 微架构的 SQ1 处理器，虽然具备了传统笔记本所不具备的高续航、永远在线等诸多新特性。但是这个处理器的性能也实在是谈不上优秀。在处理器天梯图里它甚至还赶不上 2018 年推出的苹果 A12 X。

而苹果的 M1 芯片却有着高性能与超高规格的微架构，M1 是一个超高规格版的 A14，与 A14 一样，M1 的大核是 Firestorm，小核是 Icestorm，但是不同的是，M1 有 4 个大核 4 个小核以及 8 核的 GPU，当然作为电脑处理器，M1 的频率相应有所提升。

由图可见，大核 Firestorm 的核心 IPC 大约是 SQ1 里面的 A76 大核的两倍，而小核的 IPC 则与之相仿，达成这种性能的关键就在于它恐怖的架构规模。

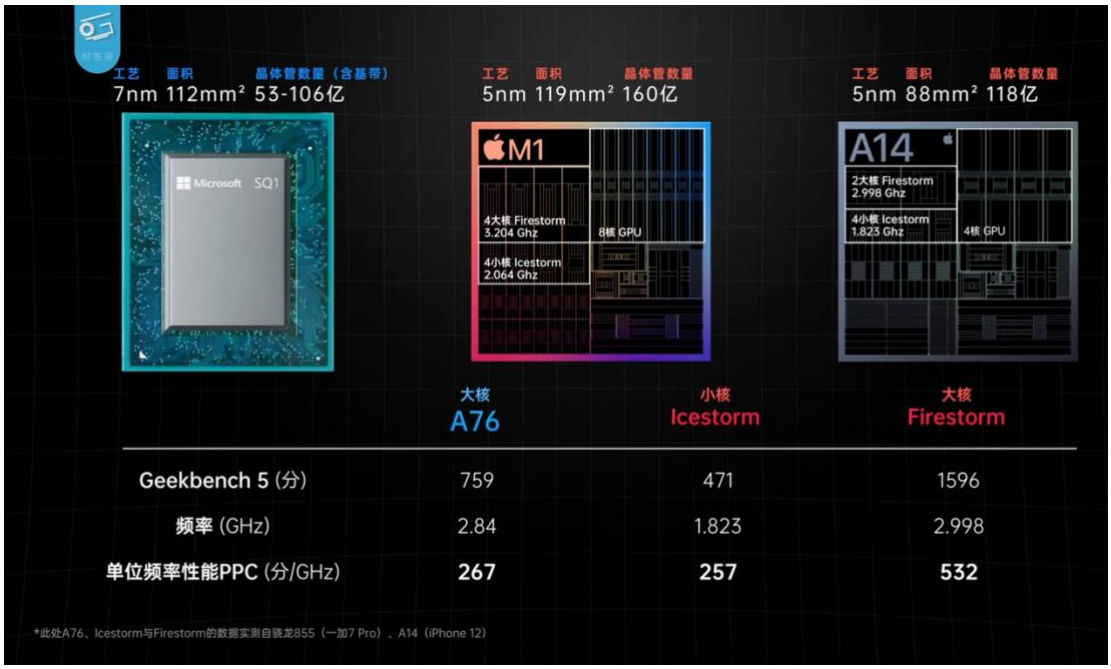


图 8:M1 的大小核分析与 A76 的对比

5 M1 的架构规模与并行分析

我们使用基于 X86 架构的 AMD 的 Zen3，以及 Intel 的 Skylake 微架构与 Apple 的 M1 的大核 Firestorm 进行架构对比，如图：

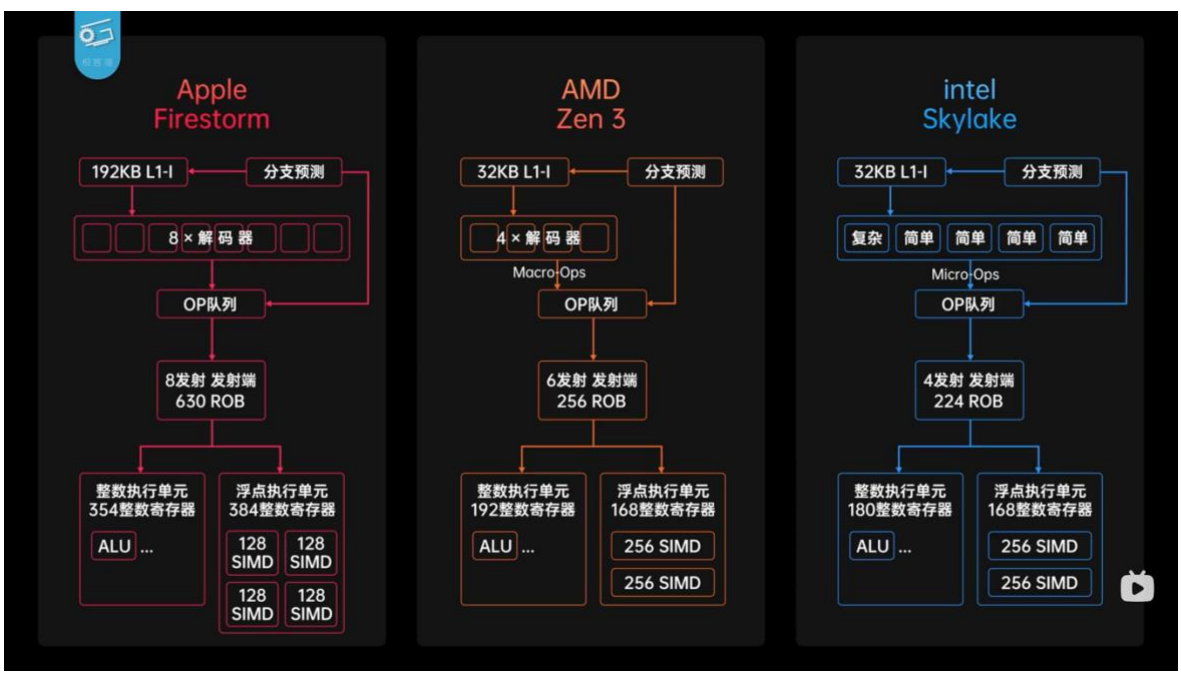


图 9：Firestorm，Zen3，Skylake 架构图

乍一看这三个的架构图都长得很像，因为不管是什么架构的 CPU ，他们的工作流程大体都是相同的，都是流水线形式的工作流程，要经过这五个阶段，分别是：取指令（IF丨Instruction Fetch），指令译码（ID丨Instruction Decode），执行指令（EXE丨Execute），访存取数（MEM丨Memory），结果写回（WB丨Writeback），所以他们的架构按着这五个运行阶段流程来设计的话，整体上可以看到的架构也是大同小异。

那这三种架构本质上都是乱序执行，超标量，多流水线的 CPU。我们可以按照处理器的工作步骤一点点来对比他们的架构。

5.1 IF（取指令）

在该阶段非常重要的是 L1 指令缓存，也就是暂时储存 CPU 指令的缓存。

从上图可以看出，对于 L1-1，M1 有 192KB 而 Zen3 和 Skylake 都只有 32KB。

处理器想要执行指令，就得先从一个地方加载指令。而 L1 指令缓存就是用来暂时保存这些指令以备后用的。从 M1 的 L1-1 缓存的大小可以看出，M1 在 JS 之类的高指令压力负载下，其表现较另外两个 32KB 的会非常好，同时这个大小的差别预示着 M1 可能是个非常宽的架构。

5.2 ID（指令译码）

指令解码器是用于将输入指令解码 CPU 为可执行指令的单元。即 CPU 拿到指令后，其内部的执行单元其实看不懂从内存中取出的指令。因为 CPU 内部有它自己的指令编码，所以咱们要对指令进行解码，把来自软件的指令翻译成 CPU 认识的指令。

在这个部分，有一个很重要的问题，也就是区分 ARM 架构与 X86 架构 CPU 的关键之处：就是关于 RICS 和 CISC 这两个指令集架构的处理器的区别。 RISC 处理器是指精简指令集处理器（Reduced Instruction Set Computer）， ARM 就属于此类，而 CISC 是指复杂指令集（Complex Instruction Set Computer），X86 就是此类的代表。

两者的界定是输入指令是复杂指令的就是 CISC，输入指令为精简指令的就是 RISC，CISC 有着复杂指令，结构复杂，种类繁多，长短不一，执行时间不恒定的特点，而 RISC 则有着精简指令，结构简单，种类精简，长短划一，执行时间恒定的特点。那么这是不是意味着 RISC 就会更快更简单呢？并不是如此。

现代的 RISC 处理器和 CISC 处理器最大的区别在于指令解码环节。

作为现代 CISC 处理器的代表，X86 接受的是 CISC 指令，但其实从 Intel 的奔腾 PRO 开始，处理器内部执行的其实已经是类 RISC 指令了。也就是 X86 处理器会通过解码器把这些复杂的指令（CISC）翻译成类似 RISC 的指令。

图 10:解码器示意图

Intel 把这些类 RISC 指令称作 Micro-Op。

而 AMD 从 K5 开始也引入了自己的类 RISC 指令集，叫做 RISC86，也就是 Macro-Op 然后再翻译成 Micro-Op，一条 Macro-Op 可以翻译成一个或多个 Micro-Op。

因此 Intel 与 AMD 的 CPU 解码设计上会有所差别。

由于这些差别，Intel 需要 N 个简单解码器加一个复杂解码器，而 AMD 则有 N 个解码器就可以了。

比如此处的 Zen3 需要 4 个解码器，而 Skylake 需要 4 个简单解码器与一个复杂解码器。

解码器的效率直接决定了后面的执行单元有多少活可以去做。

那么问题来了，M1 基于 ARM 处理器，而 ARM 处理器基于 RISC 指令集架构，为啥 CPU 还需要解码器呢？实际上 RISC 处理器接受的指令和执行单元执行的指令虽然差别很小，但依然有区别，所以也需要解码器来翻译。但显然这个解码工作简单得多，这就是 ARM 处理器能耗比优势的一个重要因素。

也就是对于 X86 与 ARM 架构的电脑来说，执行单元内部执行的其实都是类 RISC 指令，也就是说经过解码阶段后，X86 与 ARM 架构的 CPU 所要做的事情便趋近于一致。那么由此看来，基于不同指令集的处理器，最重要的区别便是 X86 接受的 CISC 指令在解码转化为类 RISC 命令时，解码器所做的工作量更大，能耗也更大，这么看来，基于 CISC 的 ARM 指令集架构在解码阶段有着先天的优势。

在解码工作量较 X86 更小的情况下，Apple 却在 M1 的大核心微架构 Firestorm 里面塞入了每周期解码 8 个指令的解码器，这个应该是目前处理器里规格里最大的解码器。

简单总结一下，现代的 X86 处理器和 ARM 处理器在经过解码阶段后，内部都会执行类 RISC 指令，因而接下来的运行阶段是高度类似的。而在解码部分，M1 具有 ARM 的先天优势，解码前后差异小，效率更高，同时解码器又做得更宽，所以有了更高的性能。

5.3 EXE（执行指令）

现代的应用于桌面端移动端的处理器，其架构大多具有乱序执行，超标量，多流水线的特性，这便是并行中很关键的特点。我们分析的三种处理器都是具有并行特性的。

5.3.1 多发射

AMD 的 X86 处理器实际执行的是 Macro-Ops，而 Zen3 每个周期可以发射 6 个 Macro-Ops 指令，于是我们就认为 Zen3 是一颗 6 发射的处理器。而这个指令发射数是目前 X86 处理器里最高的，再加上 Zen3 优秀的分支预测，消除了大部分的气泡，使得它达到了 X86 阵营目前的最高效率。

而对比 Intel 的 X86 处理器，其实际执行的是 Micro-Ops，Skylake 每个周期可以发射 4 个 Micro-Ops 指令。

图 11:发射数示意图

对于 Zen3 而言，其有 4 个解码器，每个周期可以发射 6 个 Macro-Ops 指令，Skylake 有 4 个简单解码器与 1 个复杂解码器即一共 5 个解码器，每个周期可以发射 4 个 Micro-Ops 指令，而对 M1 来说，其接收到的每一条指令将会转换，且仅会转换为一条指令，所以解码器的宽度就等于发射数。于是 Apple 制造了一个惊人的 8 发射处理器核心。作为对比，同为 ARM 架构的移动端处理器骁龙 888 的 Cortex-X1 是一个五发射处理器核心。

5.3.2 ROB（重排序缓冲区）

ROB（重排序缓冲区）对乱序执行的处理器而言至关重要。多发射的处理器每个周期会解码出多条的指令，对于没有依赖关系的指令，乱序执行的处理器可以在保证结果正确的情况下重新排列指令的顺序，让这些没有依赖关系的指令可以不用等待前面的指令执行完成就去执行。但乱序执行后的结果需要按原有顺序排好序才算执行完成。而 ROB 所做的事情就是将乱序执行后的结果进行排序。

ROB 里面保存着指令原有的顺序，经过乱序执行的指令，在执行完成之后，再按照 ROB 当中所存储的顺序写入寄存器，这样才算指令执行完毕。

所以往往 ROB 越大，意味着处理器的宽度越大且乱序度越高。M1 大概有 630 个条目的 ROB 作为对比，Zen3 有 256 个，Intel 的 Skylake 有 224 个，Sunny Cove 的 Ice Lake 有 352 个，而 ARM 的 Cortex-X1 有 224 个。

可以看出，M1 的处理器比起其他的处理器儿而言，其处理器的宽度非常大且乱序度很高。

5.4 执行单元（MEM&WB）

CPU 中的执行单元有两种，整数执行单元 ALU（算术逻辑单元）与浮点执行单元 FPU（浮点单元）。我们重点关注浮点执行单元。

5.4.1 SIMD

当你需要对多个元素执行相同的操作时，SIMD（单指令流多数据流）是一种获得更高性能的方法。这与矩阵运算密切相关。实际上，SIMD 指令通常用于加速矩阵乘法。SIMD 向量引擎是微处理器内核的一部分。在微处理器内部有一个指令解码器，它将对一条指令进行拆分，并决定激活什么功能单元。

X86 CPU 里有一个 SIMD 指令集叫做高级向量扩展指令集（Advanced Vector Extensions）即 AVX，是 X86 架构微处理器中的指令集，由英特尔在 2008 年 3 月提出。AVX 是在之前的 128 位扩展到 256 位的单指令多数据流。它解决了 X86 系列 CPU 在 decoding 上的不足。

X86 CPU 的 AVX 性能在如今十分重要。曾经 Zen2 相较于 Zen 或者 Zen+的巨大提升，很大程度就来源于 AVX 性能的提升。AVX 作为 SIMD（单指令流多数据流）指令集，十分依赖 SIMD 的宽度。而在 ARM 架构中有一种类似于 AVX 的指令集叫做 Neon。Neon 也是 SIMD 指令集。

图 12:浮点性能示意图

M1 的每个核心里有四个 128 位的引通道。这个规格和 Skylake 以及 Zen3 是同一水平。至少在处理器的架构层面上，这颗 ARM 处理器的浮点性能就已经站上了和 X86 最先进的架构平等的水平。

在 CPU 内部，会有 ALU、FPU 以及 SIMD 向量引擎，作为独立的部分，由指令解码器激活。而协处理器则是外置在微处理器内核上的。比如英特尔 8087，这是最早的协处理器芯片之一，它是一种物理上独立的芯片，旨在加快浮点计算的速度。 M1 有一个 Apple 独家的能够协同浮点单元和 NPU 计算的 AMX（矩阵协处理器）指令集，这让它的浮点性能更上一层楼，但浮点性能的发挥还和频率以及内存性能有关。

5.4.2 内存性能

接下来我们将分析 M1 的缓存与内存设计。缓存对于 CPU 的 PPC 有着巨大的帮助。Zen2 的桌面版和移动版的差距就是由于 L3 缓存导致的；COVES 的两个架构 Ice Lake 与改进版 Tiger Lake 更是几乎只有缓存容量上的区别。Apple 在 M1 的大核里塞进了 12 MB 的共享二级缓存。作为对比， A14 里面的大核的二级缓存是 8 MB，这个 12 MB 的二级缓存应该是现代 CPU 里最大的二级缓存设计，平均每核心 3 MB 的二级缓存远大于 Tiger Lake 或者是 Zen3 的二级缓存，更接近它们的三级缓存。在 Apple 以外的 CPU 上，见到这样的大二级缓存还是在酷睿 2 时代。

M1 的小核心则配置了 4 MB 的二级缓存，这个大小就更接近我们认知中的二级缓存。一般来说，二级缓存的容量越大，延迟就越难优化。但就容量来看， M1 的规格确实有点吓人。

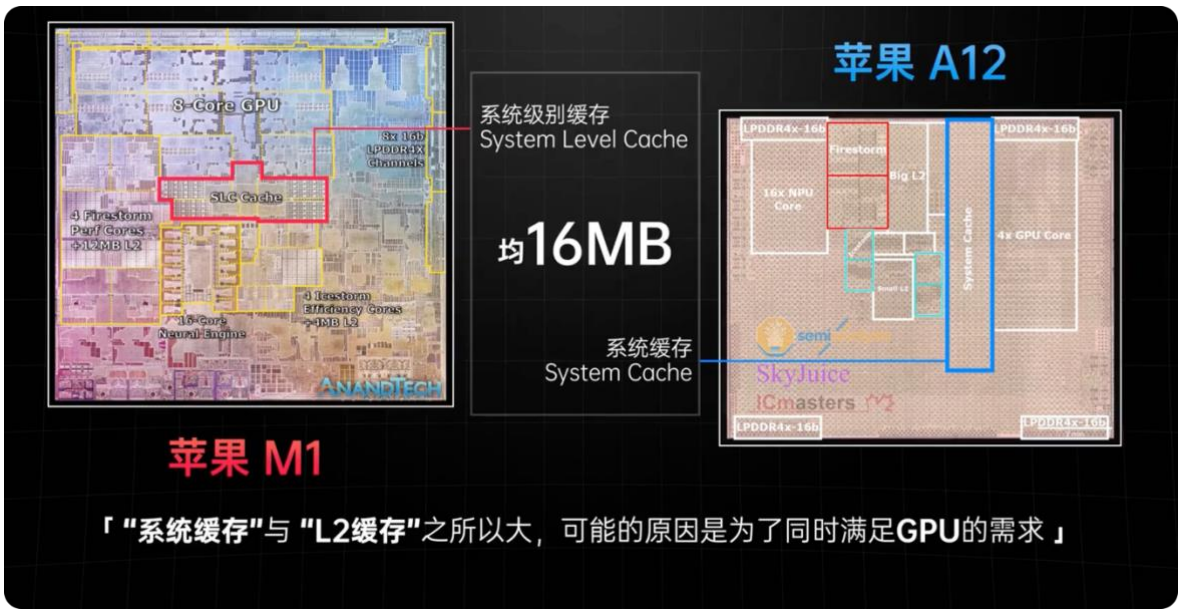


图 13:系统级别缓存示意图

除此之外， M1 还有一个类似三级缓存的系统级别缓存，这个缓存是 CPU，GPU，NPU 等等单元共享的。从官方的 keynote 公布的 DIE Shot 来看，这个系统级别缓存的部分应该和 A14 一样都是 16 MB，因为 GPU 也要用到这个缓存，所以 CPU 的大二级缓存应该也和这一点有关系。

在内存方面， M1 配备了 Apple 称之为统一内存架构的内存。实际上这并不是 Apple 第一次这样做，在 iPad pro 的 A12X 以及 A12Z，便已经应用了统一内存架构的内存。

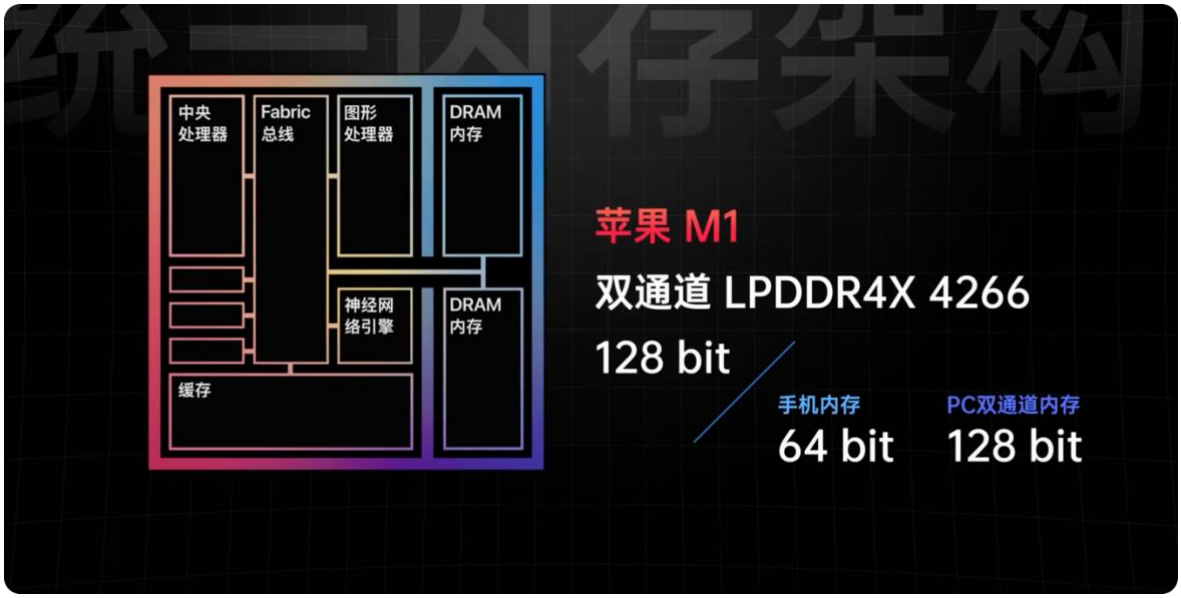


图 14:M1 架构图

这次的 M1，同样利用了统一内存架构，那 M1 集成在片上的内存是双通道的 LPDDR4X 4266，这里所说的双通道指的是 128 位，而不是手机上常说的 64 位，也就是说它的带宽相比 A14 是翻番的和内存配置比较好的 Intel 或者 AMD 的核显笔记本带宽近似。

计算机里有一个常识，距离越近延迟就越低。所以 Apple 这个统一内存架构的操作应该主要是为了降低延迟以及更好保证 GPU 的性能表现。

6 处理器设计平衡

	M1-FireStorm	Zen 3	Skylake
工艺	台积电 5 nm	台积电 7 nm	英特尔 14 nm
L1-I 指令缓存	192 KB	32 KB	32 KB
指令解码器	8	4	1大+5小
发射	8	6	4
ROB	630	256	224
整数寄存器	354	192	180
浮点寄存器	384	168	168
SIMD宽度	4×128 bit	2×256 bit	2×256 bit
L1-D (每核)	128 KB	32 KB	32 KB
L2 (每核)	3 MB（4核共享12MB）	0.5 MB	0.25 MB

图 15:微架构细节对比

简单总结一下 M1 的架构，这应该是目前最超大杯的处理器核心，即使和 Zen3 和 Skylake 相比依然十

分巨大，加上使用了目前桌面级 CPU 唯一的 5 纳米工艺，这颗处理器的性能表现十分优秀。

其实这么分析下来，基于 AMD 的 X86 Zen3 架构也非常优秀，似乎只是规格不如 M1 那么好，也许再做宽一点就能在 IPC 上打败 M1 了。可问题是，为什么其他厂商不去做像 M1 处理器一样宽的架构来提升 IPC 呢？这其实包含着几个方面的原因。

6.1 成本

Apple 的处理器不外卖，他卖的是笔记本、平板和手机，但其他处理器厂商卖的是处理器。不论是高通还是英特尔还是 AMD，这些公司赚取的是处理器那部分的利润，而 Apple 就可以省出这部分利润用来继续堆规格，但整机的成本却不会上涨。甚至因为 Mac 设备共享处理器架构，处理器的研发成本还可以得到 iPhone 和 iPad 的分摊。于是在处理器制造成本可能比英特尔或者说 AMD 更高的情况下，Apple 赚的反而更多了。

6.2 兼容性换性能

Apple 在 A11 的时代就在硬件层面终结了对 32 位应用的支持，而 macOS 也在去年的 10.15 版本里就去掉了对 32 位应用的支持。到如今 Apple 已经完成了全 64 位软件的准备。但是 X86 架构却无法脱离曾经的 32 位软件的捆绑，你是否能想象买回家的 Zen3 处理器完全不支持 32 位应用，如果真的这样，那么许多应用便完全不能使用。但 Apple 对生态的控制使得他们可以用一定的兼容性去换取更强的性能。而 X86 就得背着这种具有数十年历史的“X86 税”，在增加性能的同时继续保持对上古软件的兼容，但这种兼容性也是要占面积和费电的。

6.3 利于提高能耗比

M1 参考并在 A14 的基础上进行提升改进，而 A14 来自于手机，这意味着它要用更高的 IPC 去换取更低的频率和电压，这样才能够保证移动设备的续航。那对于 AMD 和英特尔来说，应用他们处理器的最低功耗设备是笔记本，而更高功耗的台式机并不需要考虑续航问题，所以他们可以用相对更高的频率获取相同的性能。虽然此时的能耗比会有所下降，但绝对性能上 4.5GHz- 5GHz 的 Zen3 也绝不会低于 3.2GHz 的 M1。可以说 M1 虽然是同频的性能之王，但未必是同面积性能之王。面积就是制造成本，这对于双核或者四核来说可能不会差那么大。但如果是 16 个甚至是 64 个核，处理器厂商的一个架构是要覆盖笔记本到服务器的，那么这件事就不得不考虑了。

6.4 处理器的设计

处理器的设计无非就是平衡功耗、性能和成本，功耗会因为频率和电压而大幅上升，成本会由于利润率的要求尽可能压低，所以反映在性能上就得找到一个平衡点。Apple 的特别之处就在于有着强大的生态，对于功耗，性能与成本这三点，它都有着绝对的控制权，于是就可以找到一个相对高的平衡点。因此如此高规模的宽架构的 M1 就因此诞生了。

6.5 架构的过渡

苹果宣布彻底放弃使用 X86 架构的 Intel 芯片，而是在 Mac 上搭载自己的基于 ARM 架构开发的芯片 M1，由于 M1 芯片底层架构和过去不同，由此带来的应用生态兼容性问题是首先需要解决的，为此苹果也开启了为期两年的 Mac 过渡计划。具体来说，苹果借助的是 macOS 11 Big Sur 系统以及其内置的 Universal 2、Rosetta 2 和 Virtualizaion 三种技术来解决问题。

Rosetta 2 是 M1 Mac 的一个杀手锏。但是事实上 Apple 不是第一次做处理器架构间的转换工具了。早在 2006 年，第一代 Rosetta 正是 Mac 从 Power PC 转向 Intel 架构时苹果推出的架构转换工具，当时是从 ARM 架构转换成 X86。

图 16:Rosetta 的两次应用

而这次的 Rosetta 2 是 Apple 从 X86 架构重新过渡回 ARM 架构的转换器。首先它的兼容性确实不错，不像 Microsoft 姗姗来迟的为自己推出的令人诟病的无法兼容 X86 软件的 ARM 架构的设备的 X86-64 支持，Apple 的 Rosetta 2，从一开始就可以运行 X86-64 位应用，且兼容性确实不错。基本上老 Mac 用户可以直接用恢复他们的备份到 M1 的 Mac，且绝大多数的软件运行都没啥问题。

但作为一个转译器，除了要保证兼容性，性能表现也十分重要。ARM 和 X86 的内存一致性模型不同，这会导致多线程软件运行的问题。苹果在 M1 里直接做了两版内存模型，在运行原生应用的时候，使用 ARM 的那套内存模型，而在运行 X86 转译应用的时候，就切换到 X86 的那套内存模型，这种软硬件结合带来了 Rosetta 2 的优秀表现。Rosetta 2 不仅是一个软件层面的转译器，更是需要配合 Apple 的处理器设计才实现了如今的表现。

此外，Rosetta 2 并不是一个动态转译器，而是一个静态转译器。Rosetta 2 并不是等到应用开启使用时，才一条指令一条指令地去翻译，而是在应用安装时就会对应用进行翻译，在用户使用应用前就给到一个针对 ARM 平台优化的版本。对于那些无法提前翻译的指令，才会在使用时候进行即时的翻译，在首次打开 X86 应用时，它会帮你默默转换完你的软件。然后之后打开的时候就会直接打开翻译后的代码。形象点说，动态翻译器就像是同声传译，而静态翻译就像是看翻译完的文章。这样一来性能损失会更小，也更容易保持软件的稳定。

基于这样的机制，Rosetta 2 能够帮助 X86 架构的应用在 M1 芯片的 Mac 上保持流畅快速的响应，从而获得很好的使用体验。当然，Rosetta 2 做翻译的方案毕竟只是权宜之计，不会一直存在，只是在开发者们将自己的应用全面转到 M1 芯片原生状态之前扮演过渡的作用。

References:

[1] https://baike.baidu.com/item/指令集/238130
[2] https://www.zhihu.com/question/423489755/answer/1622380842
[3] https://blog.csdn.net/leftfist/article/details/122488559
[4] https://zh.wikipedia.org/wiki/指令集架構#指令集的分类
[5] https://zh.wikipedia.org/wiki/微架構

[6] https://ark.intel.com/content/www/us/en/ark.html#@Processors

[7] https://blog.csdn.net/dunwin/article/details/84253928

[8] https://blog.csdn.net/qq_34160841/article/details/105611131

[9] https://en.wikipedia.org/wiki/List_of_ARM_processors

[10] https://en.wikipedia.org/wiki/ARM_architecture_family#64/32-bit_architecture

[11] https://en.wikipedia.org/wiki/Apple_M1

[12] https://en.wikipedia.org/wiki/Apple_silicon

[13] https://zhuanlan.zhihu.com/p/351072166

[14] https://www.bilibili.com/video/BV1Mo4y1Z7jb/?spm_id_from=333.1007.top_right_bar_window_history.content.click&vd_source=8db43b9c09e21e53f4e4e098be322d11