第 2 章

嵌入式微处理器

本章对嵌入式微处理器进行概述，介绍 Arm 嵌入式微处理器、嵌入式微处理器分类和特点以及 Cortex-M3 嵌入式微处理器。

2.1 Arm 嵌入式微处理器简介

微课视频

Arm（Advanced RISC Machine）既是一个公司的名字，也是一类微处理器的通称，还可以认为是一种技术的名字。Arm 系列处理器是由英国 Arm 公司设计的，是全球最成功的 RISC 计算机。1990 年，Arm 公司从剑桥的 Acorn 独立出来并上市；1991 年，Arm 公司设计出全球第 1 款 RISC 处理器。从此以后，Arm 处理器被授权给众多半导体制造厂，成为低功耗和低成本的嵌入式应用的市场领导者。

Arm 公司是全球领先的半导体知识产权（Intellectual Property，IP）提供商，与一般的公司不同，Arm 公司既不生产芯片，也不销售芯片，而是设计出高性能、低功耗、低成本和高可靠性的 IP 内核，如 Arm7TDMI、Arm9TDMI、Arm10TDMI 等，授权给各半导体公司使用。半导体公司在授权付费使用 Arm 内核的基础上，根据自己公司的定位和各自不同的应用领域，添加适当的外围电路，从而形成自己的嵌入式微处理器或微控制器芯片产品。目前，绝大多数的半导体公司都使用 Arm 公司的授权，如 Intel、IBM、三星、德州仪器、飞思卡尔（Freescale）、恩智浦（NXP）、意法半导体等。这样既使 Arm 技术获得更多的第三方工具、硬件、软件的支持，又使整个系统成本降低，使产品更容易进入市场被消费者所接受，更具有竞争力。Arm 公司利用这种双赢的伙伴关系迅速成为全球性 RISC 微处理器标准的缔造者。

Arm 嵌入式处理器有着非常广泛的嵌入式系统支持，如 Windows CE、μC/OS-Ⅱ、μCLinux、VxWorks、μTenux 等。

2.1.1 Arm 处理器的特点

因为 Arm 处理器采用 RISC 结构，所以它具有 RISC 架构的一些经典特点，具体如下。

（1）体积小、功耗低、成本低、性能高。

（2）支持 Thumb（16 位）/Arm（32 位）双指令集，能很好地兼容 8 位 /16 位器件。

（3）大量使用寄存器，指令执行速度更快。
（4）大多数数据操作都在寄存器中完成。
（5）寻址方式灵活简单，执行效率高。
（6）内含嵌入式在线仿真器。
基于 Arm 处理器的上述特点，其被广泛应用于以下领域。
（1）为通信、消费电子、成像设备等产品提供可运行复杂操作系统的开放应用平台。
（2）在海量存储、汽车电子、工业控制和网络应用等领域，提供实时嵌入式应用。
（3）在军事、航天等领域，提供宽温、抗电磁干扰、耐腐蚀的复杂嵌入式应用。

2.1.2　Arm 体系结构的版本和系列

1. Arm 处理器的体系结构

Arm 体系结构是 CPU 产品所使用的一种体系结构，Arm 公司开发了一套拥有知识产权的 RISC 体系结构的指令集。每个 Arm 处理器都有一个特定的指令集架构，而一个特定的指令集架构又可以由多种处理器实现。

自从第 1 个 Arm 处理器芯片诞生至今，Arm 公司先后定义了 8 个 Arm 体系结构版本，分别命名为 V1 ～ V8；此外，还有基于这些体系结构的变种版本。V1 ～ V3 版本已经被淘汰，目前常用的是 V4 ～ V8 版本，每个版本均继承了前一个版本的基本设计，但性能有所提高或功能有所扩充，并且指令集向下兼容。

1）冯・诺依曼结构

冯・诺依曼结构也称为普林斯顿结构，是一种将程序指令存储器和数据存储器合并在一起的计算机设计概念结构。它描述的是一种实作通用图灵机的计算装置，以及一种相对于平行计算的序列式结构参考模型（Referential Model），如图 2-1 所示。

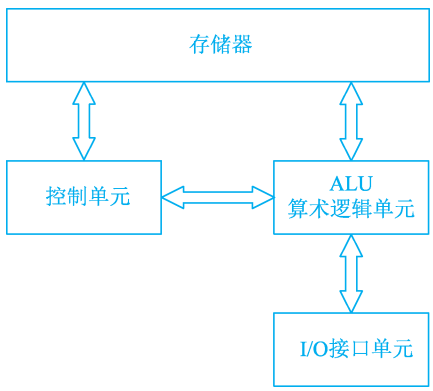

图 2-1　冯・诺依曼结构

冯・诺依曼结构隐约指导了将存储装置与中央处理器分开的概念，因此根据本结构设计出的计算机又称为存储程序型计算机。

冯・诺依曼结构处理器具有以下特点。
（1）必须有一个存储器。
（2）必须有一个控制器。
（3）必须有一个运算器，用于完成算术运算和逻辑运算。
（4）必须有输入和输出设备，用于进行人机通信。

2）哈佛结构

哈佛结构（Harvard Architecture）是一种将程序指令存储和数据存储分开的存储器结构。如图 2-2 所示，中央处理器首先到程序指令存储器中读取程序指令内容，解码后得到数据地址，再到相应的数据存储器中读取数据，并进行下

一步操作（通常是执行）。程序指令存储和数据存储分开，数据和指令的存储可以同时进行，可以使指令和数据有不同的数据宽度，如 Microchip 公司的 PIC16 芯片的程序指令是 14 位宽度，而数据是 8 位宽度。

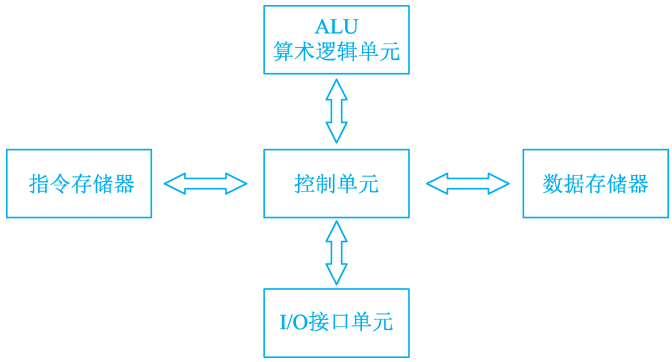

图 2-2　哈佛结构

与冯·诺依曼结构处理器比较，哈佛结构处理器具有两个明显的特点。

（1）使用两个独立的存储器模块，分别存储指令和数据，每个存储模块都不允许指令和数据并存。

（2）使用独立的两条总线，分别作为 CPU 与每个存储器之间的专用通信路径，而这两条总线之间毫无关联。

改进的哈佛结构的结构特点如下。

（1）使用两个独立的存储器模块，分别存储指令和数据，每个存储模块都不允许指令和数据并存，以便实现并行处理。

（2）具有一条独立的地址总线和一条独立的数据总线，利用公用地址总线访问两个存储模块（程序存储模块和数据存储模块），公用数据总线则被用来完成程序存储模块或数据存储模块与 CPU 之间的数据传输。

哈佛结构的微处理器通常具有较高的执行效率。其程序指令和数据指令是分开组织和存储的，执行时可以预先读取下一条指令。目前使用哈佛结构的中央处理器和微控制器有很多，除了上面提到的 Microchip 公司的 PIC 系列芯片，还有摩托罗拉公司的 MC68 系列、Zilog 公司的 Z8 系列、Atmel 公司的 AVR 系列和安谋（Arm）公司的 Arm9、Arm10 和 Arm11。Arm 有许多系列，如 Arm7、Arm9、Arm10E、XScale、Cortex 等，其中哈佛结构、冯·诺依曼结构都有，如控制领域最常用的 Arm7 系列是冯·诺依曼结构，而 Cortex-M3 系列是哈佛结构。

2. Arm 体系结构版本的变种

Arm 处理器在制造过程中的具体功能要求往往会与某个标准的 Arm 体系结构不完全一致，有可能根据实际需求增加或减少一些功能。因此，Arm 公司制定了标准，采用一些字

母后缀表明基于某个标准 Arm 体系结构版本的不同之处，这些字母称为 Arm 体系结构版本变量或变量后缀。带有变量后缀的 Arm 体系结构版本称为 Arm 体系结构版本变种。表 2-1 所示为 Arm 体系结构版本的变量后缀。

表 2-1　Arm 体系结构版本的变量后缀

变量后缀	描　　述
T	Thumb 指令集，Thumb 指令长度为 16 位，目前有两个版本：Thumb-1 用于 ArmV4 的 T 变种，Thumb-2 用于 ArmV5 以上的版本
D	含有 JTAG 调试，支持片上调试
M	内嵌硬件乘法器（Multiplier），提供用于进行长乘法操作的 Arm 指令，产生全 64 位结果
I	嵌入式 ICE，用于实现片上断点和调试点支持
E	增强型 DSP 指令，增加了新的 16 位数据乘法与乘加操作指令，加 / 减法指令可以实现饱和的带符号数的加 / 减法操作
J	Java 加速器 Jazelle，与一般的 Java 虚拟机相比，它将 Java 代码运行速度提高了 8 倍，而功耗降低了 80%
F	向量浮点单元
S	可综合版本

3. Arm 处理器的命名规则

一般 Arm 处理器内核都有一个规范的名称，该名称概括地表明了内核的体系结构和功能特性。

Arm 产品名称通常以 Arm [x][y][z][T][D][M][I][E][J][F][-S] 形式出现。

所有命名以 Arm 字符开头，后面是若干描述参数，每个参数并不是必需的。后缀的含义已在表 2-1 中列出，前 3 个参数的含义如下。

（1）[x] 表示系列号，是共享相同硬件特性的一组处理器的具体实现，如 Arm7TDMI、Arm740T 和 Arm720T 都属于 Arm7 系列。

（2）[y] 表示内存存储管理和保护单元，如 Arm72、Arm92。

（3）[z] 表示含有高速缓存，如 Arm720、Arm940。

另外，还有一些附加的要点。

（1）Arm7TDMI 之后的所有 Arm 内核，即使 Arm 标志后没有包含 TDMI 字符，也都默认包含了 TDMI 的功能特性。

（2）TAG 是由 IEEE 1149.1 标准测试访问端口和边界扫描结构来描述的，它是 Arm 用来发送和接收处理器内核与测试仪器之间调试信息的一系列协议。

（3）嵌入式 ICE 宏单元是建立在处理器内部用来设置断点和观察点的调试硬件。

（4）可综合版本，意味着处理器内核是以源代码形式提供的。这种源代码形式可被编译为一种易于电子设计自动化（Electronic Design Automation，EDA）工具使用的形式。

（5）自 2005 年以后，ArmV7 体系结构的命名方式有所改变，名称用 Arm Cortex 开头，

随后使用附加字母 -A、-R 或 -M 表示该处理器内核所适合使用的领域，再加上一个数字表示处理器在该领域的产品序号，如本书主要介绍的 Arm Cortex-M3 系列处理器。

2.1.3 Arm 的 RISC 结构特性

Arm 内核采用精简指令集计算机（RISC）体系结构，它是一个小门数的计算机，指令集和相关的译码机制比复杂指令集计算机（Complex Instruction Set Computer，CISC）要简单得多，其目标就是设计出一套能在高时钟频率下单周期执行、简单而有效的指令集。RISC 的设计重点在于降低处理器中指令执行部件的硬件复杂度，这是因为软件比硬件更容易提供更大的灵活性和更高的智能化，因此 Arm 具备了非常典型的 RISC 结构特性。

（1）具有大量的通用寄存器。

（2）通过装载/保存（Load/Store）结构使用独立的 load 和 store 指令完成数据在寄存器和外部存储器之间的传输，处理器只处理寄存器中的数据，从而可以避免多次访问存储器。

（3）寻址方式非常简单，所有装载 / 保存的地址都只由寄存器内容和指令域决定。

（4）使用统一和固定长度的指令格式。

此外，Arm 体系结构还具有以下特性。

（1）每条数据处理指令都可以同时包含算术逻辑单元（Arithmetic Logic Unit，ALU）的运算和移位处理，以实现对 ALU 和移位器的最大利用。

（2）使用地址自动增加和自动减少的寻址方式优化程序中的循环处理。

（3）load/store 指令可以批量传输数据，从而实现最大数据吞吐量。

（4）大多数 Arm 指令是可“条件执行”的，也就是说，只有当某个特定条件满足时指令才会被执行。通过使用条件执行，可以减少指令的数目，从而提高程序的执行效率和代码密度。

这些在基本 RISC 结构上增强的特性使 Arm 处理器在高性能、低代码规模、低功耗和小的硅片尺寸方面取得良好的平衡。

从 1985 年 Arm1 诞生至今，Arm 指令集体系结构发生了巨大的改变，还在不断地完善和发展。为了清楚地表达每个 Arm 应用实例所使用的指令集，Arm 公司定义了 7 种主要的 Arm 指令集体系结构版本，以版本号 V1 ～ V7 表示。

2.1.4 Arm 处理器系列

Arm 十几年如一日地开发新的处理器内核和系统功能块，其功能不断进化，处理水平持续提高。根据功能 / 性能指标和应用方向，开发出多个内核，实现了处理器内核的系列化。下面按照内核体系分别介绍 Arm 处理器的产品。

1. Arm7 系列

Arm7 内核采用冯·诺依曼体系结构，数据和指令使用同一条总线。内核有一条 3 级流水线，执行 Arm V4 指令集。

Arm7 系列处理器主要应用于对功耗和成本要求比较苛刻的消费类产品。其最高主频可以到达 130MIPS（MIPS 指每秒执行的百万条指令数）。

Arm7 系列处理器主要具有以下特点。

（1）具有 32 位 RISC 处理器。

（2）最高主频达 130MIPS。

（3）功耗低。

（4）代码密度高，兼容 16 位微处理器。

（5）开发工具多，EDA 仿真模型多。

（6）调试机制完善。

（7）提供 0.25μm、0.18μm 及 0.13μm 的生产工艺。

（8）代码与 Arm9、Arm9E 以及 Arm10E 系列兼容。

Arm7 系列包含 Arm7EJ-S、Arm7TDMI、Arm7TDMI-S、Arm720T，它们属于低端的 Arm 微处理器核，在工业控制器、MP3 播放器、喷墨打印机、调制解调器以及早期的移动通信设备等产品中使用。典型的 Arm7 系列微处理器芯片有三星公司的 S3C44B0、恩智浦公司的 LPC2131 和 Atmel 公司的 AT91SAM7S/256 等。

2. Cortex-A8 处理器

Cortex-A8 是 Arm 公司开发的基于 Arm V7 架构的首款应用级处理器，同时也是 Arm 所开发的同类处理器中性能最好、能效最高的处理器。从 600MHz 开始到 1GHz 以上的运算能力使 Cortex-A8 能够轻易胜任那些要求功耗小于 300mW 的、耗电量最优化的移动电话器件以及那些要求有 2000MIPS 执行速度的、性能最优化的消费者产品的应用。

Cortex-A8 是 Arm 公司首个超量处理器，其特色是运用了可增加代码密度和加强性能的技术、可支持多媒体以及信号处理能力的 NEONTM 技术以及能够支持 Java 和其他字节码语言（Byte-Code Language）的提前和即时编译的 Jazelle 运行时编译目标代码（Runtime Compilation Target，RCT）技术。

Arm 最新的 Artisan Advantage-CE 库以其先进的泄漏控制技术使 Cortex-A8 处理器实现了优异的速度和能效。

Cortex-A8 具有多种先进的功能特性，它是一个有序、双行、超标量的处理器内核，具有 13 级整数运算流水线、10 级 NEON 媒体运算流水线、可对等待状态进行编程的专用的二级缓存以及基于历史的全局分支预测；在功耗最优化的同时，实现了 2.00MIPS/MHz 的性能。Cortex-A8 完全兼容 Arm V7 架构，采用 Thumb-2 指令集，带有为媒体数据处理优化的 NEON 信号处理能力、Jazelle RC Java 加速技术，并采用了 TrustZong 技术保障数据的安全性。Cortex-A8 带有经过优化的一级缓存，还集成了二级缓存。众多先进的技术使其适用于家电

以及电子行业等各种高端的应用领域。

3. Arm9/9E 系列

Arm9 系列发布于 1997 年，由于采用了 5 级指令流水线，Arm9 处理器能够运行在比 Arm7 更高的时钟频率上，改善了处理器的整体性能；存储器系统根据哈佛体系结构（程序和数据空间独立的体系结构）重新设计，区分了数据总线和指令总线。Arm9 系列的第 1 个处理器是 Arm920T，包含独立的数据指令 Cache 和 MMU。该处理器能够应用在要求有虚拟存储器支持的操作系统上。Arm922T 是 Arm920T 的变种，只有一半大小的数据指令 Cache。Arm940T 包含一个更小的数据指令 Cache 和一个内存保护单元（Memory Protection Unit，MPU），它是针对不要求运行操作系统的应用而设计的。Arm920T、Arm940T 都执行 V4T 架构指令。Arm9 系列的下一个处理器是基于 Arm9E-S 内核的。这个内核是 Arm9 内核带有 E 扩展的一个可综合版本，有 Arm946E-S 和 Arm966E-S 两个变种，两者都执行 V5TE 架构指令，它们也支持可选的嵌入式跟踪宏单元，支持开发者实时跟踪处理器上指令和数据的执行。当调试对时间敏感的程序段时，这种方法非常重要。

Arm9 系列包含 Arm922T、Arm926EJ-S、Arm940T、Arm946E-S、Arm966E-S 等多种类型的微处理器核。典型的 Arm9 系列微处理器芯片有三星公司的 S3C2410、S3C2440 以及恩智浦公司的 LPC2900 和 Atmel 公司的 AT91RM9200 等。

4. Arm10 系列

Arm10 发布于 1999 年，具有高性能、低功耗的特点。它将 Arm9 的流水线扩展到 6 级，也支持可选的向量浮点单元（Vector Float Point，VFP），对 Arm10 的流水线加入了第 7 段。VFP 明显增强了浮点运算性能，并与 IEEE 754.1985 浮点标准兼容。Arm10E 系列处理器采用了新的节能模式，提供了 64 位的 Load/Store 体系，支持包括向量操作的满足 IEEE 754 标准的浮点运算协处理器，系统集成更加方便，拥有完整的硬件和软件开发工具。Arm10E 系列包括 Arm1020E、Arm1022E 和 Arm1026EJ-S 这 3 种类型。

5. Arm11 系列

Arm1136J-S 发布于 2003 年，是针对高性能和高能效而设计的。Arm1136J-S 是第 1 个执行 Arm V6 架构指令的处理器。它集成了一条具有独立的 Load/Store 体系和算术流水线的 8 级流水线。ArmV6 指令包含了针对多媒体处理的单指令流多数据流扩展，采用特殊的设计改善视频处理能力。Arm11 系列包含 Arm1136J-S、Arm1136JF-S、Arm1156T2（F）-S、Arm1176JZ（F）-S、Arm11 MPCore 等。典型的 Arm11 系列微处理器芯片有三星公司的 S3C6410 和飞思卡尔公司的 i.MX35 系列等。

6. SecurCore 系列

SecurCore 系列处理器提供了基于高性能的 32 位 RISC 技术的安全解决方案。SecurCore 系列处理器除了具有体积小、功耗低、代码密度高等特点外，还具有以下特有的特点。

（1）支持 Arm 指令集和 Thumb 指令集，以提高代码密度和系统性能。

（2）采用软内核技术以提供最大限度的灵活性，可以防止外部对其进行扫描探测。

（3）提供了安全特性，可以抵制攻击。

（4）提供面向智能卡和低成本的 MPU。

（5）可以集成用户自己的安全特性和其他协处理器。

SecurCore 系列包含 SC100、SC110、SC200 和 SC210 这 4 种类型。

7. Cortex 系列

2006 年 Arm 公司推出了基于 ArmV7 架构的 Cortex 系列标准体系架构，从而满足各种技术的不同性能要求。Cortex 系列明确地分为 A、R、M 这 3 个系列。

Arm Cortex-A 系列处理器主要用于具有高计算要求、运行丰富的操作系统及提供交互媒体和图形体验的应用领域，如智能手机、平板电脑、汽车娱乐系统、数字电视等。这类应用所需处理器都运行在很高的时钟频率（超过 1GHz）上，支持 Linux、Android、Windows 和移动操作系统等完整操作系统，具有满足操作系统需要的内存管理单元。

Arm Cortex-R 系列处理器属于面向实时应用的高性能处理器系列，主要用于硬盘控制器、汽车传动系统、无线通信的基带控制、大容量存储控制器等深层嵌入式实时应用。多数实时处理器不支持 MMU，不过通常具有 MPU、Cache 和其他针对工业应用设计的存储器功能。实时处理器运行在比较高的时钟频率（如 200MHz ～ 1GHz），响应延迟非常低。虽然实时处理器不能运行完整版本的 Linux 和 Windows 操作系统，但是支持大量的实时操作系统（RTOS）。

Arm Cortex-M 系列处理器主要针对低成本和功耗敏感的应用，如智能测量、人机接口设备、汽车和工业控制系统、家用电器、消费性产品和医疗器械。其巨大的性价比优势已经对传统的 8 位和 16 位单片机市场构成实质性的威胁和冲击。以本书将着重介绍的 STM32F10x 系列为例，其低配置型号的价格只有 10 元人民币左右，却同样具有 32 位处理器的强大性能，无论在性能上还是在功耗上，相对于传统的单片机都具有极大的优势。

2.1.5　Arm Cortex-M 处理器

Arm Cortex-M 处理器家族更多地集中在低性能端，但是这些处理器相比于许多传统微控制器性能仍然更强大。例如，Cortex-M4 和 Cortex-M7 处理器应用在许多高性能的微控制器产品中，最大的时钟频率可以达到 400MHz。表 2-2 所示为 Arm Cortex-M 处理器家族。

表 2-2　Arm Cortex-M 处理器家族

处 理 器	描　　述
Cortex-M0	面向低成本、超低功耗的微控制器和深度嵌入式应用的非常小的处理器
Cortex-M0+	针对小型嵌入式系统的最高能效的处理器，与 Cortex-M0 处理器的尺寸和编程模式接近，但是具有扩展功能，如单周期 I/O 接口和向量表重定位功能

续表

处 理 器	描　　述
Cortex-M1	针对 FPGA 设计优化的小处理器，利用 FPGA 上的存储器块实现了紧耦合内存（TCM），和 Cortex-M0 有相同的指令集
Cortex-M3	针对低功耗微控制器设计的处理器，面积小但是性能强劲，支持可快速处理复杂任务的丰富指令集。具有硬件除法器和乘加指令（MAC），并且支持全面的调试和跟踪功能，使软件开发者可以快速地开发他们的应用
Cortex-M4	不但具备 Cortex-M3 的所有功能，并且扩展了面向数字信号处理的指令集，如单指令多数据指令和更快的单周期 MAC 操作。此外，还有一个可选的支持 IEEE 754 浮点标准的单精度浮点运算单元
Cortex-M7	针对高端微控制器和数据处理密集的应用开发的高性能处理器。具备 Cortex-M4 支持的所有指令功能，扩展支持双精度浮点运算，并且具备扩展的存储器功能，如 Cache 和紧耦合存储器
Cortex-M23	面向超低功耗、低成本应用设计的小尺寸处理器，和 Cortex-M0 相似，但是支持各种增强的指令集和系统层面的功能特性，还支持 TrustZone 安全扩展
Cortex-M33	主流的处理器设计，与之前的 Cortex-M3 和 Cortex-M4 处理器类似，但系统设计更灵活，能耗比更高效，性能更高；还支持 TrustZone 安全扩展


微课视频

相比于老的 Arm 处理器（如 Arm7TDMI、Arm9），Cortex-M 处理器有一个非常不同的架构，例如：

（1）仅支持 Arm Thumb 指令，已扩展到同时支持 16 位和 32 位指令 Thumb-2 版本；

（2）内置的嵌套向量中断控制负责中断处理，自动处理中断优先级、中断屏蔽、中断嵌套和系统异常。

2.2　嵌入式处理器的分类和特点

处理器分为通用处理器与嵌入式处理器两类。通用处理器以 x86 体系架构的产品为代表，基本被 Intel 和 AMD 两家公司垄断。通用处理器追求更快的计算速度、更大的数据吞吐率，有 8 位处理器、16 位处理器、32 位处理器和 64 位处理器。

在嵌入式应用领域应用较多的还是各种嵌入式处理器。嵌入式处理器是嵌入式系统的核心，是控制、辅助系统运行的硬件单元。根据现状，嵌入式处理器可以分为嵌入式微处理器、嵌入式微控制器、嵌入式 DSP 和嵌入式 SoC。因为嵌入式系统有应用针对性的特点，不同系统对处理器的要求千差万别，因此嵌入式处理器种类繁多。据不完全统计，全世界嵌入式处理器的种类已经超过 1000 种，流行的体系架构有 30 多个。现在几乎每个半导体制造商都生产嵌入式处理器，越来越多的公司有自己的处理器设计部门。

1. 嵌入式微处理器

嵌入式微处理器处理能力较强、可扩展性好、寻址范围大、支持各种灵活设计，且不限于某个具体的应用领域。嵌入式微处理器是 32 位以上的处理器，具有体积小、重量轻、成本

低、可靠性高的优点，在功能、价格、功耗、芯片封装、温度适应性、电磁兼容方面更适合嵌入式系统应用要求。嵌入式微处理器目前主要有 Arm、MIPS、PowerPC、xScale、ColdFire 系列等。

2. 嵌入式微控制器

嵌入式微控制器（MCU）又称为单片机，在嵌入式设备中有着极其广泛的应用。嵌入式微控制器芯片内部集成了 ROM、可擦可编程只读存储器（Erasable Programmable Read-Only Memory，EPROM）、RAM、总线、总线逻辑、定时 / 计数器、看门狗、I/O、串行口、脉宽调制输出、ADC、DAC、Flash RAM、EEPROM 等各种必要功能和外设。和嵌入式微处理器相比，嵌入式微控制器最大的特点是单片化，体积大大减小，从而使功耗和成本下降，可靠性提高。嵌入式微控制器的片上外设资源丰富，适合嵌入式系统工业控制的应用领域。嵌入式微控制器从 20 世纪 70 年代末出现至今，出现了很多种类，比较有代表性的嵌入式微控制器产品有 Cortex-M、8051、AVR、PIC、MSP430、C166、STM8 系列等。

3. 嵌入式 DSP

嵌入式数字信号处理器（Embedded Digital Signal Processor，EDSP）又称为嵌入式 DSP，是专门用于信号处理的嵌入式处理器，它在系统结构和指令算法方面经过特殊设计，具有很高的编译效率和指令执行速度。嵌入式 DSP 内部采用程序和数据分开的哈佛结构，具有专门的硬件乘法器，广泛采用流水线操作，提供特殊的数字信号处理指令，可以快速实现各种数字信号处理算法。在数字化时代，数字信号处理是一门应用广泛的技术，如数字滤波、快速傅里叶变换（Fast Fourier Transform，FFT）、谱分析、语音编码、视频编码、数据编码、雷达目标提取等。传统微处理器在进行这类计算操作时的性能较低，而嵌入式 DSP 的系统结构和指令系统针对数字信号处理进行了特殊设计，因而在执行相关操作时具有很高的效率。比较有代表性的嵌入式 DSP 产品是 Texas Instruments 公司的 TMS320 系列和 Analog Devices 公司的 ADSP 系列。

4. 嵌入式 SoC

针对嵌入式系统的某一类特定的应用对嵌入式系统的性能、功能、接口有相似的要求的特点，用大规模集成电路技术将某一类应用需要的大多数模块集成在一枚芯片上，从而在芯片上实现一个嵌入式系统大部分核心功能的处理器就是嵌入式 SoC。

嵌入式 SoC 把微处理器和特定应用中常用的模块集成在一枚芯片上，应用时往往只需要在 SoC 外部扩充内存、接口驱动、一些分立元件及供电电路，就可以构成一套实用的系统，极大地降低了系统设计的难度，还有利于减小电路板面积、降低系统成本、提高系统可靠性。嵌入式 SoC 是嵌入式处理器的一个重要发展趋势。

5. 嵌入式处理器的特点

在分类的基础上，描述一款嵌入式处理器通常包括以下方面。

（1）内核。内核是一个处理器的核心，它影响处理器的性能和开发环境。通常，内核包

括内部结构和指令集。而内部结构又包括运算和控制单元、总线、存储管理单元及异常管理单元等。

（2）片内存储资源。高性能处理器通常片内集成高速 RAM，以提高程序执行速度。一些 MCU 和 SoC 内置 RoM 或 Flash ROM，以简化系统设计，提高相关处理器应用的方便性。

（3）外设。嵌入式处理器不可缺少外设，如中断控制器、定时器、直接存储器访问（Direct Memory Access，DMA）控制器等，还包括通信、人机交互、信号 I/O 等接口。

（4）电源。嵌入式处理器的电源电气指标通常包括处理器正常工作和耐受的电压范围，以及工作所需的最大电流。

（5）封装形式。封装形式包括处理器的尺寸、外形和引脚方式等。

嵌入式处理器是嵌入式系统的核心。为了满足嵌入式系统实时性强、功耗低、体积小、可靠性高的要求，嵌入式处理器具有以下特点。

（1）速度快。实时应用要求处理器必须具有高处理速度，以保证在限定的时间内完成从数据获取、分析处理到控制输出的整个过程。

（2）功耗低。电池续航能力是手机等移动设备的一项重要的性能指标。电子气表、电子水表、电子锁等产品要求电池续航时间达一年甚至更长的时间。进一步地，嵌入式处理器不仅要求低功耗，还需具有管理外设功耗的能力。

（3）接口丰富，I/O 能力强。手机、个人数字助理（Personal Digital Assistant，PDA）以及个人媒体播放器（Personal Media Player，PMP）等设备要求系统具有液晶显示、扬声器等输出设备，支持键盘、手写笔等输入设备及 Wi-Fi、蓝牙、USB 等通信能力。这类产品要求嵌入式处理器能够集成多种接口，满足系统丰富的功能需求。

（4）可靠性高。不同于通用计算机，嵌入式系统经常工作在无人值守的环境中，一旦系统出错难以得到及时纠正。因此，嵌入式处理器常采用看门狗（Watchdog）电路等技术提高可靠性。

（5）生命周期长。一些嵌入式系统的应用需求比较稳定，长时间内不发生变化。另外，稳定成熟的嵌入式处理器不仅可以保证产品质量的稳定性，而且具有较低的价格。因此，嵌入式处理器通常具有较长的生命周期。例如，Intel 公司于 1980 年推出的 8 位微控制器 8051，至今仍然是全球流行的产品。

（6）产品系列化。为了缩短产品的开发周期和上市时间，嵌入式处理器产品呈现出系列化、家族化的特征。通常，同一系列不同型号处理器采用相同的架构，其内部组成和接口有所区别。产品系列化保证了软件的兼容性，提高了软件升级和移植的方便性。

通常的处理器分类方法可以用于对嵌入式微处理器进行分类。例如，可以依据嵌入式处理器指令集的特点、处理器字长、内部总线结构和功能特点等进行分类。

2.3　Cortex-M3 嵌入式微处理器

2.3.1　Arm 概述

Arm 包括 Arm1 ～ Arm11 与 Cortex，其中被广泛应用的是 Arm7、Arm9、Arm11 以及 Cortex 系列。

Arm 的设计具有典型的精简指令系统(RISC)风格。Arm 的体系架构已经经历了 6 个版本，版本号分别为 V1 ～ V6，每个版本各有特色，定位也各有不同，彼此之间不能简单地相互替代。其中，Arm9、Arm10 对应的是 V5 架构，Arm11 对应的是发表于 2001 年的 V6 架构，时钟频率为 350 ～ 500MHz，最高可达 1GHz。

Cortex 是 Arm 的全新一代处理器内核，它在本质上是 Arm V7 架构的实现，它完全有别于 Arm 的其他内核，是全新开发的。按照 3 类典型的嵌入式系统应用，即高性能、微控制器、实时类，Cortex 又分为 3 个系列，即 Cortex-A、Cortex-M、Cortex-R。而 STM32 就属于 Cortex-M 系列。

Cortex-M 旨在提供一种高性能、低成本的微处理器平台，以满足最小存储器、小引脚数和低功耗的需求，同时兼顾卓越的计算性能和出色的中断管理能力。目前典型的、使用最为广泛的是 Cortex-M0、Cortex-M3、Cortex-M4。


微课视频

与 MCS-51 单片机采用的哈佛结构不同，Cortex-M 采用的是冯 • 诺依曼结构，即程序存储器和数据存储器不分开，统一编址。

Arm 在 1990 年成立，最初的名字是 Advanced RISC Machines Ltd.，当时它由 3 家公司——苹果电脑公司、Acorn 电脑公司以及 VLSI 技术（公司）合资成立。1991 年，Arm 推出了 Arm6 处理器家族，VLSI 则是第 1 个制造 Arm 芯片的公司。后来，TI、NEC、Sharp、ST 等公司陆续都获取了 Arm 授权，使 Arm 处理器应用在手机、硬盘控制器、掌上计算机、家庭娱乐系统以及其他消费电子产品中。

Arm 过去称作高级精简指令集机器（Advanced RISC Machine，更早称作 Acorn RISC Machine），是一个 32 位精简指令集（RISC）处理器架构，其广泛地使用在许多嵌入式系统设计中。

Arm 公司是一家出售技术知识产权的公司。所谓的技术知识产权，类似于卖房屋的结构设计图，至于要怎样修改，哪边开窗户，以及要怎样加盖其他的花园，就由买了设计图的厂商自己决定。而有了设计图，当然还要有实现设计图的厂商，这些就是 Arm 架构的授权客户群。Arm 公司本身并不靠自有的设计制造或出售 CPU，而是将处理器架构授权给有兴趣的厂家。许多半导体公司持有 Arm 授权，Intel、TI、Qualcomm、华为、中兴、Atmel、Broadcom、Cirrus Logic、恩智浦半导体（于 2006 年从飞利浦独立出来）、富士通、英特尔、IBM、NVIDIA、新唐科技（Nuvoton Technology）、英飞凌、任天堂、OKI 电气工业、三星电子、Sharp、STMicroelectronics 和 VLSI 等公司均拥有各种不同形式的 Arm 授权。Arm 公

司与获得授权的半导体公司的关系如图 2-3 所示。

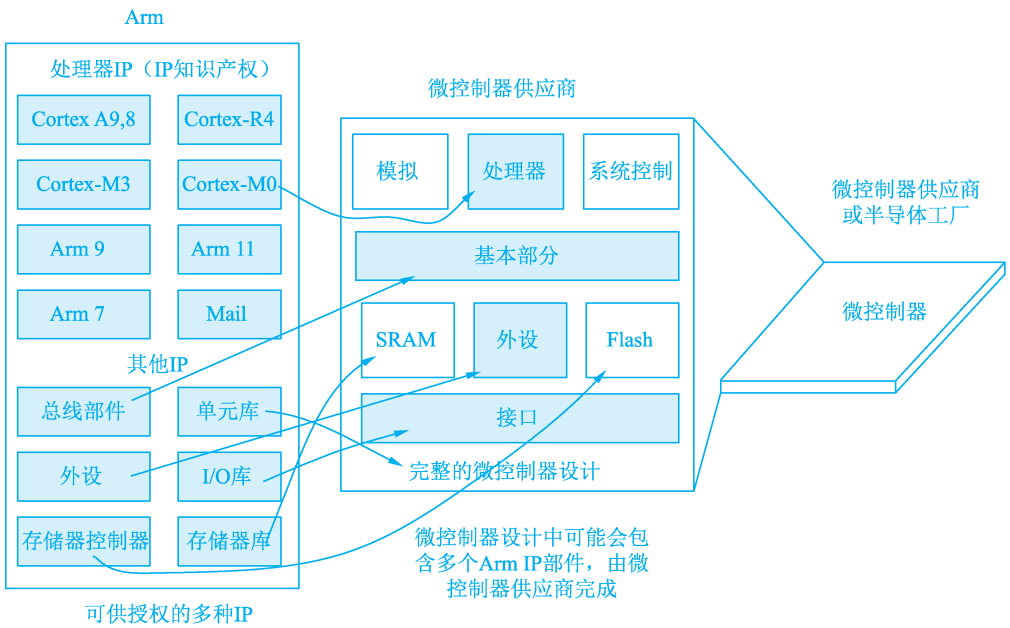

图 2-3　在微控制器中使用 Arm 授权

Cortex-M3 是首款基于 Arm V7-M 架构的 32 位处理器，具有功耗低、逻辑门数较少、中断延迟短、调试成本低等诸多优点。与 8 位 /16 位设备相比，Arm Cortex-M3 32 位 RISC 处理器提供了更高的代码效率。它整合了多种技术，减少了使用内存，并在极小的 RISC 内核上提供低功耗和高性能。Arm Cortex-M3 内核降低了编程难度，集高性能、低功耗、低成本于一体。

Arm Cortex-M3 处理器不仅使用了先进的哈佛结构，执行 32 位的 Thumb-2 指令集，同时包含高效的系统外设——嵌套向量中断控制器（Nested Vectored Interrupt Controller，NVIC）和 Arbiter 总线，还实现了 Tail-Chaining 中断技术，可在实际应用中缩短 70% 的中断处理时间。

Arm Cortex-M3 内部还具有多个调试组件，用于在硬件水平上支持调试操作，如指令断点、数据观察点等。另外，为了支持更高级的调试，还有其他可选组件，包括指令跟踪和多种类型的调试接口。下面对 Arm Cortex-M3 处理器的性能进行详细介绍。

1. 高性能

（1）许多指令都是单周期的，包括乘法相关指令，并且在整体性能上，Cortex-M3 优于绝大多数其他的架构。

（2）采用哈佛结构，指令总线和数据总线被分开，取指令和访问可以并行操作。

（3）Thumb-2 指令集无须进行 32 位 Arm 状态和 16 位 Thumb 状态的切换，极大地简化了软件开发和代码维护，也缩短了产品开发周期。

（4）Thumb-2 指令集为编程带来了更大的灵活性，提高了 Cortex-M3 的代码密度，进而减少了存储器的需求。

（5）取指令都按 32 位处理，同一周期最多可以取出两条指令，留下了更多的带宽给数据传输。

（6）Cortex-M3 的设计允许单片机高频运行，即使在相同的速度下运行，Cortex-M3 的每指令周期数（Cycles Per Instruction，CPI）也更低，于是同样的频率下可以做更多的工作；另外，也使同一个应用在 Cortex-M3 上需要更低的主频。

2. 先进的中断处理功能

（1）内置的嵌套向量中断控制器支持多达 240 条外部中断输入。向量化的中断功能极大地缩短了中断延迟，因为不再需要软件判断中断源。中断的嵌套也是在硬件水平上实现的，不需要软件代码实现。

（2）Cortex-M3 在进入异常服务例程时，因为自动压栈了 R0 ～ R3、R12、LR、PSR 和 PC 寄存器，同时在返回时自动弹出它们，因此加速了中断的响应，也无需汇编语言代码。

（3）NVIC 支持对每路中断设置不同的优先级，使中断管理极富弹性。

3. 低功耗

（1）Cortex-M3 需要的逻辑门数少，自然对功耗的要求就低（低于 0.19mW/MHz）。

（2）在内核水平上支持节能模式（SLEEPING 和 SLEEPDEEP 位）。通过使用等待中断指令（WFI）和等待事件指令（WFE），内核可以进入睡眠模式，并且以不同的方式唤醒。

（3）Cortex-M3 的设计是全静态的、同步的、可综合的。任何低功耗的或标准的半导体工艺均可放心使用。

4. 系统特性

（1）系统支持“位寻址”操作和字节不变大端模式，并且支持非对齐的数据访问。

（2）拥有先进的 Fault 处理机制，支持多种类型的异常和 Fault，使故障诊断更容易。

（3）通过引入 Banked 堆栈指针机制，将系统程序使用的堆栈和用户程序使用的堆栈划清界限。如果再配上可选的 MPU，处理器就能彻底满足对软件健壮性和可靠性有严格要求的应用。

5. 调试支持

（1）在支持传统的 JTAG 的基础上，还支持串行线调试接口。

（2）基于 CoreSight 调试解决方案，使处理器运行期间也能访问处理器状态和存储器内容。

（3）内建了对多达 6 个断点和 4 个数据观察点的支持。

（4）可以选配一个 ETM 模块，用于指令跟踪。

（5）在调试方面还加入了 Fault 状态寄存器、新的 Fault 异常，以及闪存修补（Patch）操作等新特性，使调试大幅简化。

（6）可选 ITM 模块，测试代码可以通过它输出调试信息，而且使用方便。

CPU 寄存器是处理器内部主要用于暂存运算数据、运算中间结果的存储单元，这种寄存器称为通用寄存器；也有一些寄存器用于暂存处理器的状态、控制信息、一些特殊的指针等工作信息，这种寄存器称为特别功能寄存器。对于嵌入式开发，常见的寄存器又分为针对内核的寄存器和针对硬件外设的寄存器。

嵌入式开发者不必关心处理器内部电路的具体实现方式，只须关注这个处理器的应用特性，即如何编程使用 Arm 处理器进行运算处理，以及该处理器芯片引脚的信号特性（如引脚属性、时序、交流特性、直流特性等）。对于应用编程设计用户，处理器可抽象为物理寄存器，用户通过对这些物理寄存器的设置完成程序设计，进而进行数据处理。

当前很多设计开发和学习过程中，寄存器并不是完全可见的。在 PC 程序开发中，程序功能的实现更多依赖于系统提供的 API 和其他一些封装控件，开发者无须接触寄存器。

本书介绍的 STM32 开发中，将采用基于标准外设库的开发模式，可以使开发者不用深入了解底层硬件细节（包括寄存器）就可以灵活规范地使用每个外设。但是，对于嵌入式开发和学习者，寄存器的学习还是必要的，因为它是深入开发和研究的基础，只有掌握这些底层的相关知识才能提升嵌入式学习的水平。

Arm Cortex-M3 有通用寄存器 R0 ～ R15 以及一些特殊功能寄存器。R0 ～ R12 是“通用的”，但是绝大多数 16 位的指令只能使用 R0 ～ R7（低位寄存器组），而 32 位的 Thumb-2 指令则可以访问所有通用寄存器（包括低位寄存器组和高位寄存器组）。特殊功能寄存器必须通过专门的指令访问。

2.3.2 CISC 和 RISC

Arm 公司在经典处理器 Arm11 以后的产品都改用 Cortex 命名，主要分成 A、R 和 M 3 类，旨在为各种不同的市场提供服务，A 系列处理器面向尖端的基于虚拟内存的操作系统和用户应用，R 系列处理器针对实时系统，M 系列处理器针对微控制器。

指令的强弱是 CPU 的重要指标，指令集是提高处理器效率的最有效工具之一。从现阶段的主流体系架构来看，指令集可分为复杂指令集（CISC）和精简指令集（RISC）两部分。

CISC 是一种为了便于编程和提高存储器访问效率的芯片设计体系。在 20 世纪 90 年代中期之前，大多数的处理器都采用 CISC 体系，包括 Intel 的 80x86 和 Motorola 的 68K 系列等，即通常所说的 x86 架构就属于 CISC 体系。随着 CISC 处理器的发展和编译器的流行，一方面指令集越来越复杂，另一方面编译器却很少使用这么多复杂的指令集。而且，如此多的复杂指令，CPU 难以对每条指令都作出优化，甚至部分复杂指令本身耗费的时间反而更多，这就是著名的“8020 定律”，即在所有指令集中，只有 20%的指令常用，而 80%的指令基本上很少用。

20 世纪 80 年代，RISC 开始出现，它的优势在于将计算机中最常用的 20%的指令集中优化，而剩下的不常用的 80%的指令，则采用拆分为常用指令集的组合等方式运行。RISC 的关键技术在于流水线操作，在一个时钟周期内完成多条指令，而超流水线及超标量技术在芯片设计中已普遍被使用。RISC 体系多用于非 x86 阵营高性能处理器 CPU，如 Arm、

MIPS、PowerPC、RISC-V 等。

1. CISC 机器

CISC 体系的指令特征为使用微代码，计算机性能的提高往往是通过提高硬件的复杂性获得的。随着集成电路技术，特别是超大规模集成电路（Very Large Scale Integration Circuit，VLSI）技术的迅速发展，为了软件编程方便和提高程序的运行速度，硬件工程师采用的办法是不断增加可实现复杂功能的指令和多种灵活的编址方式，甚至某些指令可支持高级语言语句归类后的复杂操作，因此硬件越来越复杂，造价也越来越高。为实现复杂操作，CISC 处理器除了向程序员提供类似各种寄存器和机器指令功能，还通过存储于 ROM 中的微代码实现其极强的功能，指令集直接在微代码存储器（比主存储器的速度快很多）中执行。庞大的指令集可以减少编程所需要的代码行数，减轻程序员的负担。

优点：指令丰富，功能强大，寻址方式灵活，能够有效缩短新指令的微代码设计时间，允许设计师实现 CISC 体系机器的向上兼容。

缺点：指令集及芯片的设计比上一代产品更复杂，不同的指令需要不同的时钟周期来完成，执行较慢的指令，将影响整台机器的执行效率。

2. RISC 机器

RISC 体系的特征是包含简单、基本的指令，这些指令可以组合成复杂指令。每条指令的长度都是相同的，可以在一个单独操作中完成。大多数指令都可以在一个机器周期内完成，并且允许处理器在同一时间执行一系列的指令。

优点：在使用相同的芯片技术和相同运行时钟下，RISC 系统的运行速度是 CISC 系统的 2 ～ 4 倍。由于 RISC 处理器的指令集是精简的，它的存储管理单元、浮点单元等都能设计在同一枚芯片上。RISC 处理器比相对应的 CISC 处理器设计更简单，所需要的时间将变得更短，并可以比 CISC 处理器应用更多先进的技术，开发更快的下一代处理器。

缺点：多指令的操作令程序开发者必须小心地选用合适的编译器，而且编写的代码量会变得非常大。另外，RISC 处理器需要更快的存储器，并将其集成于处理器内部，如一级缓存（L1 Cache）。

3. RISC 和 CISC 的比较

综合 RISC 和 CISC 的特点，分析两者之间的区别，具体如下。

（1）指令系统：RISC 设计者把主要精力放在那些经常使用的指令上，尽量使它们具有简单、高效的特点。对于不常用的功能，常通过组合指令来完成。因此，在 RISC 机器上实现特殊功能时，效率可能较低。但可以利用流水技术和超标量技术加以改进和弥补。而 CISC 机器的指令系统比较丰富，有专用指令完成特定的功能，因此处理特殊任务效率较高。

（2）存储器操作：RISC 对存储器的操作有限制，使控制简单化；而 CISC 机器的存储器操作指令多，且操作直接。

（3）程序：RISC 汇编语言程序一般需要较大的内存空间，实现特殊功能时程序复杂，不易设计；而 CISC 汇编语言程序编程相对简单，科学计算及复杂操作的程序设计相对容易，

效率较高。

（4）CPU：RISC 的 CPU 包含较少的单元电路，因而面积小、功耗低；CISC 的 CPU 包含丰富的电路单元，因而功能强、面积大、功耗大。

（5）设计周期：RISC 处理器结构简单，布局紧凑，设计周期短，且易于采用最新技术；CISC 处理器结构复杂，设计周期长。

（6）用户使用：RISC 处理器结构简单，指令规整，性能容易把握，易学易用；CISC 处理器结构复杂，功能强大，实现特殊功能容易。

（7）应用范围：由于 RISC 指令系统的确定与特定的应用领域有关，故 RISC 机器更适用于专用机，而 CISC 机器更适用于通用机。

2.3.3 Arm 架构的演变

1985 年以来，Arm 陆续发布了多个 Arm 内核架构版本，从 Arm V4 架构开始的 Arm 架构发展历程如图 2-4 所示。

目前，Arm 体系结构已经经历了 6 个版本。从 V6 版本开始，各个版本都在实际中获得了应用，还有一些变种，如支持 Thumb 指令集的 T 变种、长乘法指令（M）变种、Arm 媒体功能扩展（SIMI）变种、支持 Java 语言的 J 变种和增强功能的 E 变种等。例如，Arm7TDMI 表示该处理器支持 Thumb 指令集（T）、片上 Debug（D）、内嵌硬件乘法器（M）、嵌入式 ICE（I）。

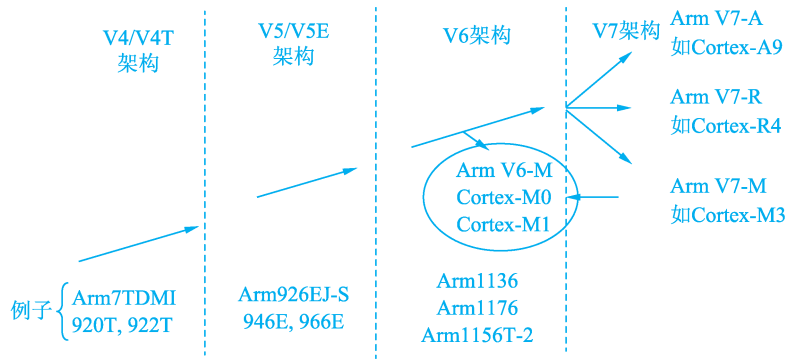

图 2-4 Arm 架构发展历程

常见 Arm 处理器的演变过程如图 2-5 所示。

每个系列都有其子集的架构，如用于 Arm V6-M 系列（所使用的 Cortex-M0/M0+/M1）的一个子集 Arm V7-M 架构（支持较少的指令）。

Cortex 是 Arm 的新一代处理器内核，本质上是 Arm V7 架构的实现。与以前的向下兼容、逐步升级策略不同，Cortex 系列处理器是全新开发的。正是由于 Cortex 放弃了向前兼容，老版本的程序必须经过移植才能在 Cortex 处理器上运行，因此对软件和支持环境提出了更高的要求。

从 Cortex 系列的核心开始，存在 3 种系列：应用系列（Cortex-A 系列，适用于需要运

行复杂应用程序的场合）、实时控制系列（Cortex-R 系列，适用于实时性要求较高的应用场合）和微控制器系列（Cortex-M 系列，适用于要求高性能、低成本的应用场合）。许多厂商提供的 Cortex-M 系列芯片内集成了大量 Flash 存储器（数十到数百千字节）、ADC、USART、SPI、I2C、DAC、CAN、USB、定时器等组件，在实际工程使用中非常方便，深受广大工程师的欢迎。

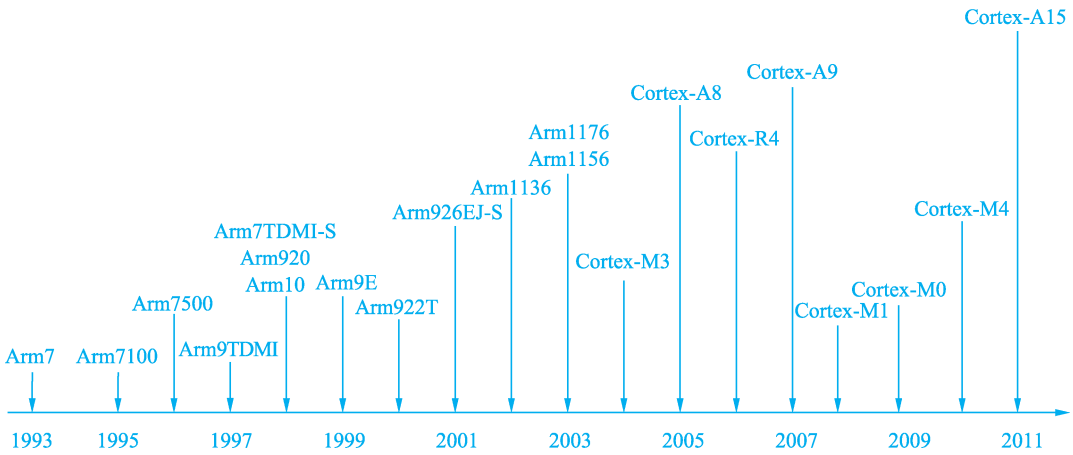

图 2-5　常见 Arm 处理器的演变过程

各种架构的 Arm 应用领域如图 2-6 所示。

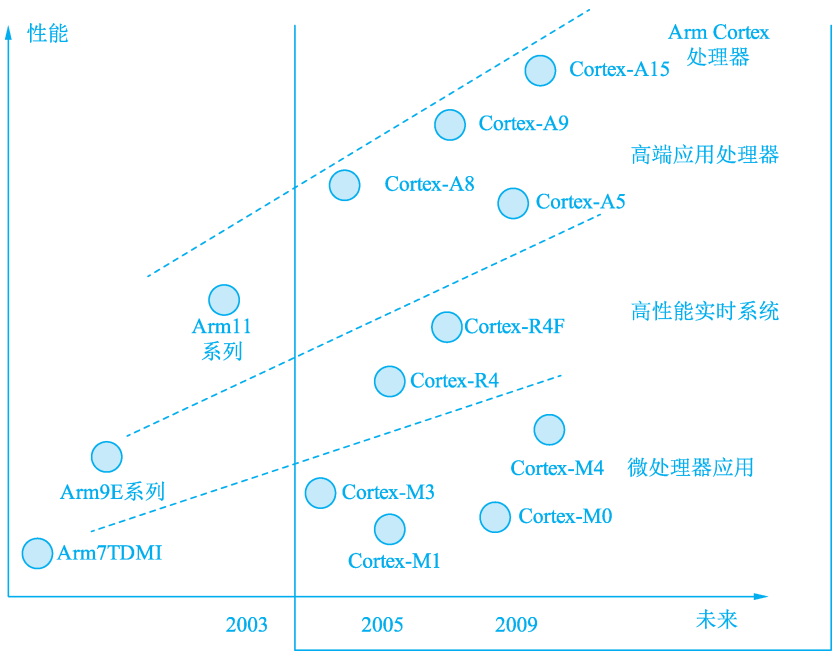

图 2-6　各种架构的 Arm 应用领域

在 Arm 公司发出的 Cortex 内核中，Cortex-M3 授权数量最多。在诸多获得 Cortex-M3 内核授权的公司中，意法半导体公司是较早在市场上推出基于 Cortex-M3 内核微控制器的厂商，STM32F1 系列是其典型的产品系列。本书后续介绍的 Arm Cortex-M3 是诸多 Arm 内核架构中的一种，并以基于该内核的意法半导体公司的 STM32F103ZT6 微控制器为背景进行原理介绍，以 STM32F103ZT6 微控制器开发为背景进行应用实例讲解。

2.3.4 Arm 体系结构与特点

从学习人脑的过程也会了解到，看微处理器是多角度的。从外观角度看，它是一枚有着丰富引脚的芯片，个头一般比较大，比较方正。再进一步看其组成结构，就是计算单元 + 存储单元 + 总线 + 外部接口的架构。细化一些，计算单元中会有 ALU 和寄存器组。ALU 是由组合逻辑构成的，有与门，有非门；寄存器是由时序电路构成的，有逻辑，有时钟。再细化一些，与门就是一个逻辑单元。

如图 2-7 所示，任何微处理器都至少由内核、存储器、总线、I/O 构成。Arm 公司的芯片特点是内核部分都是统一的，由 Arm 设计，但是对于其他部分，各个芯片制造商可以有自己的设计。有的甚至包含一些外设在里面。

Cortex-M3 处理器内核是微处理器的中央处理单元（CPU）。完整的基于 Cortex-M3 的 MCU 还需要很多其他组件。芯片制造商得到 Cortex-M3 处理器内核的使用授权后，就可以把 Cortex-M3 内核用在自己的硅片设计中，添加存储器、外设、I/O 以及其他功能块。不同厂家设计出的微处理器会有不同的配置，包括存储器容量、类型、外设等都各具特色。本书主讲处理器内核本身。如果想要了解某个具体型号的处理器，还需要查阅相关厂家提供的文档。

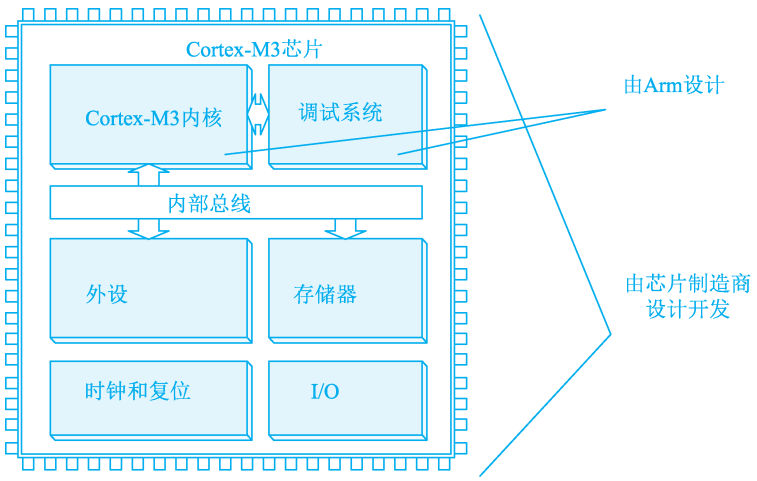

图 2-7　微处理器内核示例

如果把微处理器内核更加详细地画出来，可以看到处理器内核中包含中断控制器、取指单元、指令解码器、寄存器组、算术逻辑单元（ALU）、存储器接口、跟踪接口等，如图 2-8 所示。如果将总线细分下去，可以分为指令总线和数据总线，并且这两种总线之间带有存储器保护单元。这两种总线从内核的存储器接口接到总线网络上，再与指令存储器、存储器系统和外设等连接在一起。存储器也可细分为指令存储器和其他存储器。外设可以分为私有外设和其他外设等。

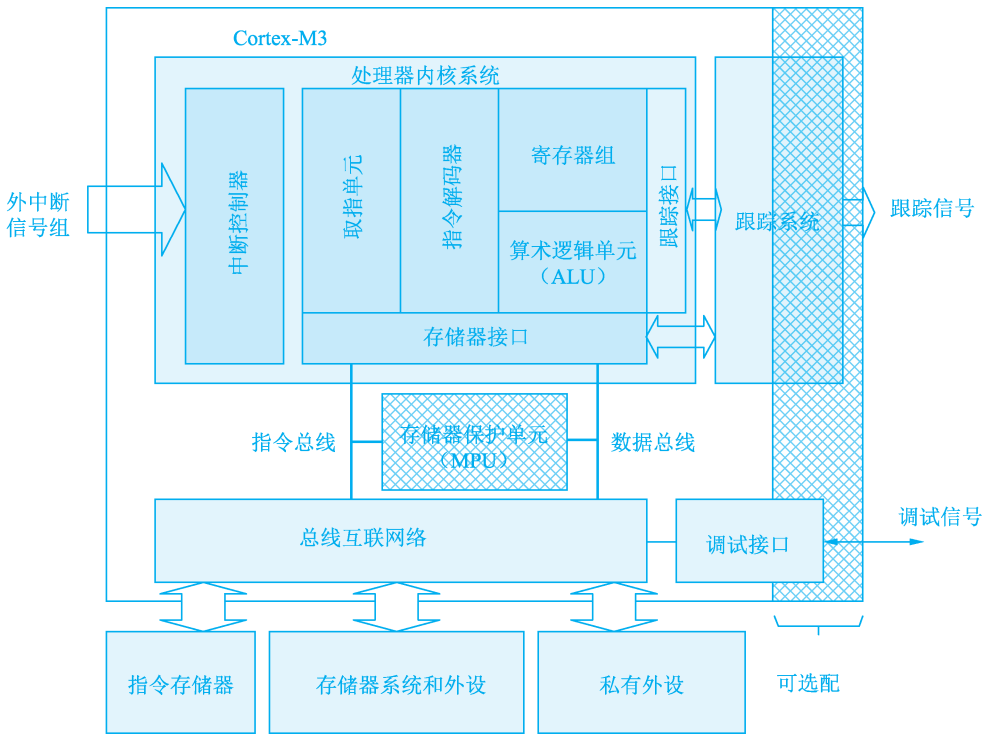

图 2-8　微处理器内核的进一步细化

如果从编程的角度来看微处理器内核，则看到的主要就是一些寄存器和地址，如图 2-9 所示。对于 CPU，编程就是使用指令对这些寄存器进行设置和操作；对于内存，编程就是对地址的内容进行操作；对于总线和 I/O 等，主要的操作包括初始化和读写操作，都是针对不同的寄存器进行设计和操作。另外，有两部分值得一提，一部分是计数器，另一部分是“看门狗”。在编程中，计数器是需要特别关注的，因为计数器一般会产生中断，所以对于计数器的操作，除了初始化以外，还要编写相应的中断处理程序。“看门狗”是为了防止程序跑飞，可以是硬件的，也可以是软件的。对于硬件“看门狗”，需要设置初始状态和阈值；对于软件“看门狗”，则需要用软件实现具体功能，并通过软中断机制产生异常，改变 CPU 的模式。如果是专门的数模转换接口，那么编程也是针对其寄存器进行操作，从而完成数模转换。对于串口编程，也就是对它的寄存器进行编程，其中还会包含具体的串口协议。

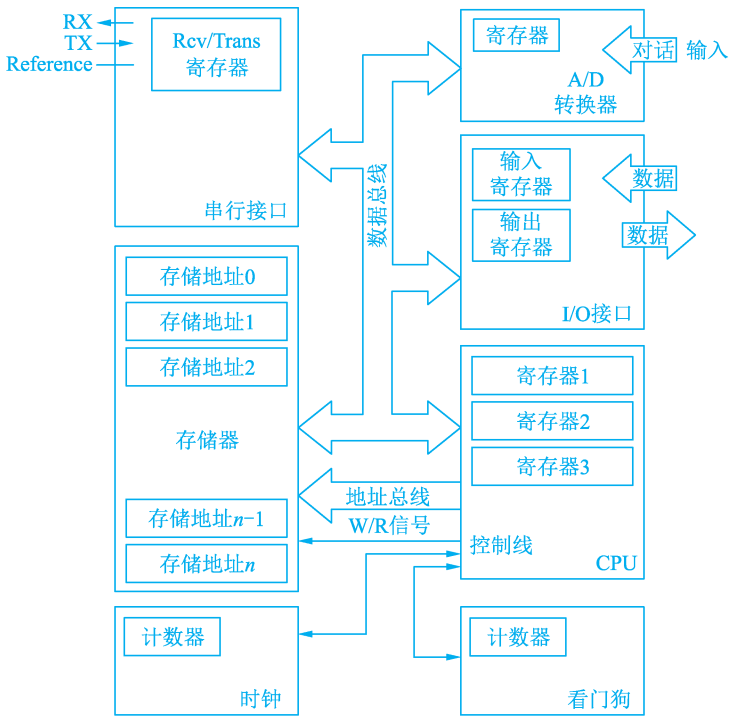

图 2-9　从编程的角度看到的微处理器内核

2.3.5　Cortex-M 系列处理器

Cortex-M 系列处理器应用主要集中在低性能端领域，但是这些处理器相比于传统处理器（如 8051 处理器、AVR 处理器等）性能仍然更强大，不仅具备强大的控制功能、丰富的片上外设、灵活的调试手段，一些处理器还具备一定的 DSP 运算能力（如 Cortex-M4 处理器和 Cortex-M7 处理器），这使其在综合信号处理和控制领域也具备较大的竞争力。

1. Cortex-M 系列处理器的特征

Cortex-M 系列处理器的特征如下。

（1）RISC 处理器内核：具有高性能 32 位 CPU、确定性的运算、低延迟 3 阶段管道，可达 1.25DMIPS/MHz[①]。

（2）Thumb-2 指令集：16/32 位指令的最佳混合，代码大小小于 8 位设备，对性能没有负面影响，提供最佳的代码密度。

（3）低功耗模式：集成的睡眠状态支持、多电源域、基于架构的软件控制。

（4）嵌套向量中断控制器（NVIC）：低延迟、低抖动中断响应，不需要汇编编程，以纯

① DMIPS 为 Dhrystone Million Instructions Executed Per Second 的缩写，主要用于衡量整数计算能力。

C 语言编写中断服务例程，能完成出色的中断处理。

（5）工具和 RTOS 支持：广泛的第三方工具支持、Cortex 微控制器软件接口标准（Cortex Microcontroller Software Interface Standard，CMSIS）、最大限度地增加软件成果重用。

（6）CoreSight 调试和跟踪：JTAG 或 2 针串行线调试（SWD）连接，支持多处理器，支持实时跟踪。此外，Cortex-M 系列处理器还提供了一个可选的内存保护单元（MPU），提供低成本的调试 / 追踪功能和集成的休眠状态，以增加灵活性。

Cortex-M0、Cortex-M0+、Cortex-M3、Cortex-M4、Cortex-M7 处理器之间有很多的相似之处，例如：

（1）基本编程模型；

（2）嵌套向量中断控制器（NVIC）的中断响应管理；

（3）架构设计的休眠模式，包括睡眠模式和深度睡眠模式；

（4）操作系统支持特性；

（5）调试功能。

2. Cortex-M3 指令集

Cortex-M3 处理器是基于 Arm V7-M 架构的处理器，支持更丰富的指令集，包括许多 32 位指令，这些指令可以高效地使用高位寄存器。另外，Cortex-M3 处理器还支持：

（1）查表跳转指令和条件执行（使用 IT 指令）；

（2）硬件除法指令；

（3）乘加指令（MAC 指令）；

（4）各种位操作指令。

更丰富的指令集通过以下几种途径增强性能：32 位 Thumb 指令支持了更大范围的立即数、跳转偏移和内存数据范围的地址偏移；支持基本的 DSP 操作（如支持若干条需要多个时钟周期执行的 MAC 指令，还有饱和运算指令）；这些 32 位指令允许用单个指令对多个数据一起做桶形移位操作。但是，支持更丰富的指令导致了更高的成本和功耗。

3. Cortex-M4 指令集

Cortex-M4 处理器在很多地方和 Cortex-M3 处理器相同，如流水线、编程模型等。Cortex-M4 处理器支持 Cortex-M3 处理器的所有功能，并额外支持各种面向 DSP 应用的指令，如 SIMD（Single Instruction Multiple Data）指令、饱和运算指令、一系列单周期 MAC 指令（Cortex-M3 处理器只支持有限条数的 MAC 指令，并且是多周期执行的）和可选的单精度浮点运算指令。

Cortex-M4 处理器的 SIMD 操作可以并行处理 2 个 16 位数据和 4 个 8 位数据。在某些 DSP 运算中，使用 SIMD 指令可以加速计算 16 位和 8 位数据，因为这些运算可以并行处理。但是，在一般的编程中，C 编译器并不能充分利用 SIMD 运算能力，这是 Cortex-M3 处理器和 Cortex-M4 处理器典型 Benchmark 分数差不多的原因。然而，Cortex-M4 处理器的内部数据通路和 Cortex-M3 处理器的内部数据通路不同，在某些情况下，Cortex-M4 处理器可以处

理得更快（如单周期 MAC 指令可以在一个周期中写回到两个寄存器）。

2.3.6 Cortex-M3 处理器的主要特性

Cortex-M3 是 Arm 公司在 Arm V7 架构的基础上设计出来的一款新型芯片内核。相比于其他 Arm 系列微控制器，Cortex-M3 内核拥有以下优势和特点。

1. 三级流水线和分支预测

现代处理器中，大多数都采用了指令预存及流水线技术提高处理器的指令运行速度。执行指令的过程中，如果遇到了分支指令，由于执行的顺序也许会发生改变，指令预存队列和流水线中的一些指令就可能作废，需要重新取相应的地址，这样会使流水线出现“断流现象”，处理器的性能会受到影响。尤其在 C 语言程序中，分支指令的比例可能达到 10%～20%，这对于处理器来说无疑是一件很恐怖的事情。因此，现代高性能的流水线处理器都会针对一些分支预测的部件，在处理器从存储器预取指令的过程中，当遇到分支指令时，处理器能自动预测跳转是否会发生，然后才从预测的方向进行相应的取值，从而让流水线能连续地执行指令，保证它的性能。

2. 哈佛结构

哈佛结构的处理器采用独立的数据总线和指令总线，处理器可以同时进行指令和数据的读写操作，使处理器的运行速度得以提高。

3. 内置嵌套向量中断控制器

Cortex-M3 首次在内核部分采用了嵌套向量中断控制器，即 NVIC。也正是采用了中断嵌套的方式，Cortex-M3 能将中断延迟缩短到 12 个时钟周期（一般，Arm7 需要 24～42 个时钟周期）。Cortex-M3 不仅采用了 NVIC 技术，还采用了尾链技术，从而使中断响应时间缩短到 6 个时钟周期。

4. 支持位绑定操作

在 Cortex-M3 内核出现之前，Arm 内核是不支持位操作的，而是要用逻辑与、逻辑或的操作方式屏蔽对其他位的影响。这带来的结果是指令的增加和处理时间的增加。Cortex-M3 采用了位绑定的方式让位操作成为可能。

5. 支持串行调试（SWD）

一般的 Arm 处理器采用的都是 JTAG 调试接口，但是 JTAG 接口占用的芯片 I/O 端口过多，这对于一些引脚少的处理器来说很浪费资源。Cortex-M3 在原来的 JTAG 接口的基础上增加了 SWD 模式，只需要两个 I/O 端口即可完成仿真，节约了调试占用的引脚。

6. 支持低功耗模式

Cortex-M3 内核在原来只有运行 / 停止的模式上增加了睡眠模式，使 Cortex-M3 的运行功耗也很低。

7. 拥有高效的 Thumb-2 16/32 位混合指令集

原有的 Arm7、Arm9 等内核使用的都是不同的指令，如 32 位的 Arm 指令和 16 位的 Thumb 指令。Cortex-M3 使用了更高效的 Thumb-2 指令实现接近 Thumb 指令的代码尺寸，达到 Arm 编码的运行性能。Thumb-2 是一种高效的、紧凑的新一代指令集。

8. 32 位硬件除法和单周期乘法

Cortex-M3 内核加入了 32 位的除法指令，弥补了一些除法密集运用导致性能不好的问题。同时，Cortex-M3 内核也改进了乘法运算的部件，使 32 位乘 32 位的乘法在运行时间上缩短到了一个时钟周期。

9. 支持存储器非对齐模式访问

Cortex-M3 内核的 MCU 一般用的内部寄存器都是 32 位编址。如果处理器只能采用对齐的访问模式，那么有些数据就必须被分配，占用一个 32 位的存储单元，这是一种浪费。为了解决这个问题，Cortex-M3 内核采用了支持非对齐模式的访问方式，从而提高了存储器的利用率。

10. 内部定义了统一的存储器映射

在 Arm7、Arm9 等内核中没有定义存储器的映射，不同的芯片厂商需要自己定义存储器的映射，这使得芯片厂商之间存在不统一的现象，给程序的移植带来了麻烦。Cortex-M3 则采用了统一的存储器映射的分配，使存储器映射得到统一。

11. 极高的性价比

Cortex-M3 内核的 MCU 相对于其他的 Arm 系列的 MCU 性价比高很多。

2.3.7　Cortex-M3 处理器结构

Arm Cortex-M3 处理器是新一代的 32 位处理器，是一个高性能、低成本的开发平台，适用于微控制器、工业控制系统以及无线网络传感器等应用场合，其特点如下。

（1）性能丰富成本低。专门针对微控制器应用特点而开发的 32 位 MCU，具有高性能、低成本、易应用等优点。

（2）低功耗。把睡眠模式与状态保留功能结合在一起，确保 Cortex-M3 处理器既可提供低能耗，又不影响高运行性能。

（3）可配置性强。Cortex-M3 的 NVIC 功能提高了设计的可配置性，提供了多达 240 个具有单独优先级、动态重设优先级功能和集成系统时钟的系统中断。

（4）丰富的连接。功能和性能兼顾的良好组合，使基于 Cortex-M3 的设备可以有效处理多个 I/O 通道和协议标准。

Cortex-M3 处理器结构如图 2-10 所示。

Cortex-M3 处理器结构中各部分的解释和功能如下。

（1）嵌套向量中断控制器（NVIC）：负责中断控制。该控制器和内核是紧耦合的，提供

可屏蔽、可嵌套、动态优先级的中断管理。

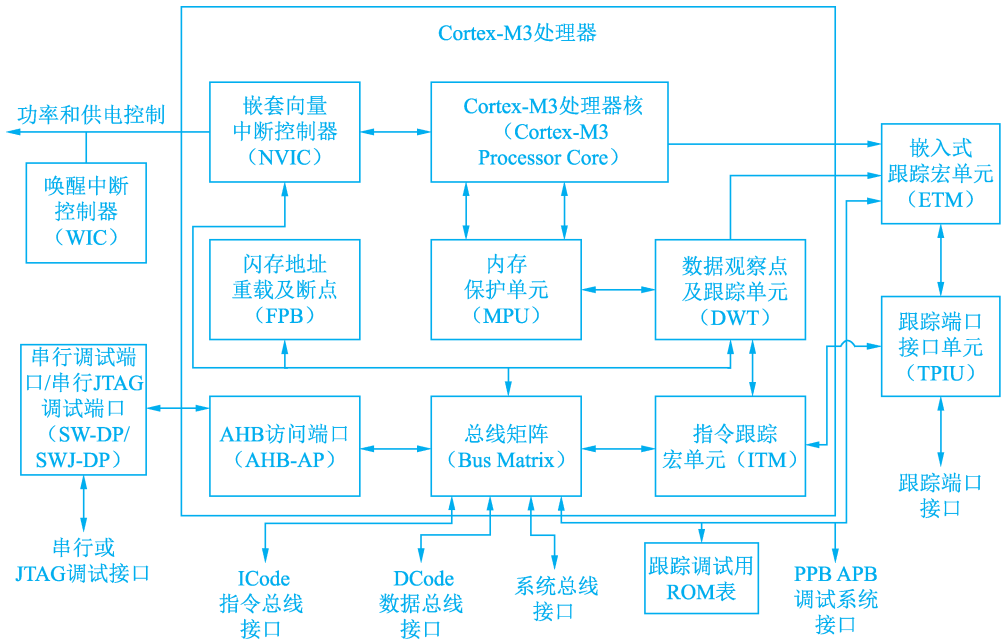

图 2-10　Cortex-M3 处理器结构

（2）Cortex-M3 处理器核（Cortex-M3 Processor Core）：Cortex-M3 处理器核是处理器的核心所在。

（3）闪存地址重载及断点（FPB）：实现硬件断点以及代码空间到系统空间的映射。

（4）内存保护单元（MPU）：实施存储器的保护，能够在系统或程序出现异常而非正常地访问不应该访问的存储空间时，通过触发异常中断而达到提高系统可靠性的目的。STM32 系统并没有使用该单元。

（5）数据观察点及跟踪单元（DWT）：调试中用于数据观察功能。

（6）AHB 访问端口（AHB-AP）：高速总线 AHB 访问端口将 SW/SWJ 端口的命令转换为 AHB 的命令传输。

（7）总线矩阵（Bus Matrix）：Cortex-M3 总线矩阵，CPU 内部的总线通过总线矩阵连接到外部的 ICode、DCode 及系统总线。

（8）指令跟踪宏单元（ITM）：可以产生时间戳数据包并插入跟踪数据流中，用于帮助调试器求出各事件的发生时间。

（9）唤醒中断控制器（WIC）：可以使处理器和 NVIC 处于低功耗睡眠模式。

（10）嵌入式跟踪宏单元（ETM）：调试中用于处理指令跟踪。

（11）串行调试端口 / 串行 JTAG 调试端口（SW-DP/SWJ-DP）：串行调试的端口。

（12）跟踪端口接口单元（TPIU）：跟踪端口的接口单元。用于向外部跟踪捕获硬件发

送调试信息的接口单元，作为来自 ITM 和 ETM 的 Cortex-M3 内核跟踪数据与片外跟踪端口之间的桥接。

Cortex-M3 是 32 位处理器核，地址线、数据线都是 32 位的。Cortex-M3 采用了哈佛结构，这种并行结构将程序指令和数据分开进行存储，其优点是在一个机器周期内处理器可以并行获得执行字和操作数，提高了执行速度。简单理解一下，指令存储器和其他数据存储器采用不同的总线（ICode 和 DCode 总线），可并行取得指令和数据。

2.3.8　存储器系统

1. 存储器系统的功能

Cortex-M3 存储器系统的功能与传统的 Arm 架构相比，有了明显的改变。

（1）存储器映射是预定义的，并且还规定好了哪个位置使用哪条总线。

（2）Cortex-M3 存储器系统支持“位带”（Bit-Band）操作。通过它，实现了对单一位的操作。

（3）Cortex-M3 存储器系统支持非对齐访问和互斥访问。

（4）Cortex-M3 存储器系统支持大端配置和小端配置。

2. 存储器映射

Cortex-M3 只有一个单一固定的存储器映射，极大地方便了软件在各种 Cortex-M3 单片机间的移植。举个简单的例子，各款 Cortex-M3 单片机的 NVIC 和 MPU 都在相同的位置布设寄存器，使它们变得与具体器件无关。存储空间的一些位置用于调试组件等私有外设，这个地址段被称为私有外设区。私有外设区的组件包括以下几种。

（1）闪存地址重载及断点单元（FPB）。

（2）数据观察点及跟踪单元（DWT）。

（3）指令跟踪宏单元（ITM）。

（4）嵌入式跟踪宏单元（ETM）。

（5）跟踪端口接口单元（TPIU）。

（6）ROM 表。

Cortex-M3 的地址空间是 4GB，程序可以在代码区、内部 SRAM 区以及 RAM 区执行。但是，因为指令总线与数据总线是分开的，最理想的是把程序放到代码区，从而使取指令和数据访问各自独立进行。

内部 SRAM 区的大小是 512MB，用于让芯片制造商连接片上的 SRAM，这个区通过系统总线来访问。在这个区的下部，有一个 1MB 的区间，称为位带区。该位带区还有一个对应的 32MB 的位带别名区，容纳了 8M 个“位变量”。位带区对应的是最低的 1MB 地址范围，而位带别名区中的每个字对应位带区的 1 位。位带操作只适用于数据访问，不适用于取指令。

地址空间的另一个 512MB 范围由片上外设的寄存器使用，这个区中也有一个 32MB 的位带别名区，以便于快捷地访问外设寄存器，用法与内部 SRAM 区中的位带别名区相同。

例如，可以方便地访问各种控制位和状态位。需要注意的是，外设区内不允许执行指令。

还有两个 1GB 的范围，分别用于连接外部 RAM 和外部设备，它们之中没有位带区。两者的区别在于外部 RAM 区允许执行指令，而外部设备区则不允许。

最后还剩下 0.5GB 的隐秘地带，Cortex-M3 内核的核心就在这里面，包括系统级组件、内部私有外设总线、外部私有外设总线，以及由提供者定义的系统外设。

其中，私有外设总线有以下两条。

（1）AHB 私有外设总线，只用于 Cortex-M3 内部的 AHB 外设，它们是 NVIC、FPB、DWT 和 ITM。

（2）APB 私有外设总线，既用于 Cortex-M3 内部的 APB 设备，也用于外部设备（这里的“外部”是相对内核而言）。Cortex-M3 允许器件制造商再添加一些片上 APB 外设到 APB 私有总线上，它们通过 APB 接口来访问。

NVIC 所处的区域叫作系统控制空间（System Control Space，SCS），在 SCS 中除了 NVIC 外，还有 SysTick、MPU 以及代码调试控制所用的寄存器。

3. 存储器的各种访问属性

Cortex-M3 除了为存储器做映射，还为存储器的访问规定了 4 种属性：可否缓冲（Bufferable）、可否缓存（Cacheable）、可否执行（Executable）、可否共享（Shareable）。

如果配备了 MPU，则可以通过它配置不同的存储区，并且覆盖默认的访问属性。Cortex-M3 片内没有配备缓存，也没有缓存控制器，但是允许在外部添加缓存。通常，如果提供了外部内存，芯片制造商还要附加一个缓存控制器。它可以根据可否缓存的设置管理对片内和片外 RAM 的访问操作。地址空用可以通过另一种方式分为 8 个 512MB 等份。

（1）代码区（0x0000 0000 ～ 0x1FFF FFFF）。该区是可以执行指令的，缓存属性为 WT（写通，White Through），即不可以缓存。该区也允许布设数据存储器，在该区的数据操作是通过数据总线接口完成的（读数据使用 DCode，写数据使用 System），且在该区的写操作是缓冲的。

（2）SRAM 区（0x2000 0000 ～ 0x3FFF FFFF）。该区用于片内 SRAM，写操作是缓冲的。并且，可以选择 WB-WA（Write Back-Write Allocated）缓存属性。该区也可以执行指令，允许把代码复制到内存中执行，常用于固件升级等维护工作。

（3）片上外设区（0x4000 0000 ～ 0x5FFF FFFF）。该区用于片上外设，因此是不可缓冲的，也不可以在该区执行指令（这也称为 Execute Never，简写为 XN，Arm 的参考手册大量使用此术语）。

（4）外部 RAM 区的前半段（0x6000 0000 ～ 0x7FFF FFFF）。该区可用于布设片上 RAM 或片外 RAM，可缓存（缓存属性为 WB-WA），并且可以执行指令。

（5）外部 RAM 区的后半段（0x8000 0000 ～ 0x9FFF FFFF）。除了不可缓冲（WT）外，与前半段相同。

（6）外部外设区的前半段（0xA000 0000 ～ 0xBFFF FFFF）。该区用于多核系统中的共

享内存（需要严格按顺序操作，即不可缓冲）。该区也是不可执行区。

（7）外部外设区的后半段（0xC000 0000 ～ 0xDFFF FFFF）。目前与前半段的功能完全一致。

（8）系统区（0xE000 0000 ～ 0xFFFF FFFF）。该区是私有外设和供应商指定功能区，不可执行代码。系统区涉及很多关键部位，因此访问都是严格序列化的（不可缓存，不可缓冲）。而供应商指定功能区则是可以缓存和缓冲的。

4. 存储器的默认访问许可

Cortex-M3 有一个默认的存储访问许可，它能防止用户代码访问系统控制存储空间，保护 NVIC、MPU 等关键部件，默认访问许可在以下条件下生效。

（1）没有配备 MPU。

（2）配备了 MPU，但是 MPU 被禁止。

如果启用了 MPU，则 MPU 可以在地址空间中划出若干区，并为不同的区规定不同的访问许可权。

5. 位带操作

支持了位带操作后，可以使用普通的 load/store 指令对单一的比特进行读写。在 Cortex-M3 中，有两个区实现了位带，一个是 SRAM 区的最低 1MB，另一个则是片内外设区的最低 1MB。这两个位带中的地址除了可以像普通的 RAM 一样使用外，它们还都有自己的位带别名区，位带别名区把每比特膨胀成一个 32 位的字。当通过位带别名区访问这些字时，就可以达到访问原始比特的目的。

在位带中，每比特都映射到位带别名区的一个字，这是只有最低有效位（Least Significant Bit，LSB）有效的字。当一个别名地址被访问时，会先把该地址变换为位带地址。

（1）对于读操作，读取位带地址中的一个字，再把需要的位右移到 LSB，并将 LSB 返回。

（2）对于写操作，把需要写的位左移到对应的位序号处，然后执行一个原子（不可分割）“读改写”过程。

支持位带操作的两个内存区的范围是 0x2000 0000 ～ 0x200F FFFF（SRAM 区的最低 1MB）和 0x4000 0000 ～ 0x400F FFFF（片上外设区的最低 1MB）。