**Chap1-概述** DMA（Direct Memory Access） ：直接存储器访问 DSP（Digital Signal Processor） ：数字信号处理器 PC（Program Counter） ：程序计数器

**1.2** **计算机系统的组成** **-** **冯.** **诺依曼结构的组成（五个部分）【掌握】**五个主要部分：存储器、运算器、控制器、输入设备、输出设备。存储器：主存储器（内存，ROM RAM）辅助存储器（外存，非易失性）,在 P267 中，有这样一句 话“片上 SRAM 也被成为主存储器”;运算器：ALU+寄存器;控制器：指令寄存器（IR） ，指令译码器（ID） ，操作控制器（OC） ;输入设备;输出设备（输入设备和输出设备统称为 I/O 设备，通过外部总线连接到适配器上，进而连接到内 部总线上） ;适配器;互联网络（总线） ：数据总线（Data Bus） 、地址总线（Address Bus） 、控制总线（Control Bus） 。**1.3** **计算机中数的表示方法** **-** **理解有符号数的表示方法，会求补码【掌握】**原码、反码、补码三者对于正数的 表示方式是完全一致的；对于负数 x：原码、反码最高位 1 反码其他取反，补码等于反码+1。【补码 & 原数】的映射关系：(10000000)b= -128；(10000001)b=-127……，(11111110)b=-2；(11111111)b=-1；(00000000)b=0；(01111111)b=127。

进制相关：2-B;8-O;16-H。 关于溢出： (对于有符号数)如果在加法中最高位发生了进位，则 CP=1；如果次高位发生了进位，则 CF=1。 当且仅当 CP 不等于 CF 时，才发生溢出。浮点数： ，其中 S 为符号位，M 为尾数，E 为阶

码。32-bit: len(S)=1, len(M)=23, len(E)=8；64-bit: len(S)=1, len(M)=52, len(E)=11。

BCD 码：二进制编码的十进制数，每个十进制数用4 位二进制数表示，称为 BCD 码（压缩 BCD 码：每个字节存储两个 BCD 码）例如： 14 对应 0001 0100

**Chap2-计算机系统的基本结构与工作原理** 2.1 计算机系统的基本结构与组成 - 微程序设计思想【理解】微程序设计思想：将指令的执行过程分解成一系列微操作，每个微操作对应一个微指令，微指令存储在控制存储器中，控制器按照 微指令的顺序执行微操作，从而完成指令的执行过程。CISC 指令集复杂，分为指令系统层-微体系结构层-数字逻辑子层。RISC 指令集精简，分为指令系统层-硬核层。Peripheral(外设）

**2.2** **模型机存储器子系统-存储器分级设计思想（兼顾速度、容量、成本）【理解】**一般情况下的存储架构（字扩展来理解,这里默认每个存储体有 8 位） ：若字长为8m 则分别有 m 个存储体;地址总线 n-bit，即 2n ，意味着每个存储体的 地址空间为2n;字节选择信号 m-bit，某一位有效意味着需要存取对应存储体的数据;总空间m\*2nB，总线宽度m+n bit. **小端和大端格式（基本概念）** ；字长与字的对齐【了解】 小端格式：低地址存放低位，高地址存放高位;大端格式： 低地址存放高位， 高地址存放低位(注意单个字节内，顺序永远是小端格式。事实上讨论字节内的端序是无意义的，一般我们以 byte 作为最小单位， 而不是 bit.存储器分级（从快到慢，从小到大） ：寄存器（Register） ：CPU 内部， 最快，最贵，最小;高速缓存（Cache）：CPU 内部，快，贵，小;一级缓存（单 CPU 内）两个/核（数据+程序） ;二级缓存（单 CPU 内） ;三级缓存（多核共享） ;主存储器（Main Memory） ：CPU 内部，慢，便宜，大;辅助存储器（Auxiliary Memory） ：CPU 外部，最慢，最便宜，最大.

**2.4** **模型机指令集和指令执行过程** **-** **模型机指令执行流程（结合汇编编程、指令翻译、寻址方式、流水线原理）【掌握】**模型机常用汇编指令（RISC 风格）

下方中 Rd 一般代表目的寄存器（destination register） ，Rs 代表源寄存器（source register）指令虽然很多，但是整体还是比较清晰的，可以分为以下几类：

1.二元算术指令：加法 ADD, 减法 SUB, 与 AND, 或 OR 等等。形式：ADD Rd Rs1 Rs2 其中 Rs2 也可以是某一个立即数 Imm 2.一元算术指令：取反 NOR 等等。形式：NOR Rd Rs1

3.读取写入：LDR 读取，STR 写入。形式：LDR Rd address, STR Rs address。 4.寄存器赋值：MOV。形式：MOV Rd Rs，其中 Rs 也可以是某一个立即数 Imm

5.有参数控制类（一般参数都是 address） 。形式：JX address，JNX address，JMP address，CALL address

有条件跳转：JX 和 JNX，其中 X 是某一个可用的条件标志位，如 Z（结果为 0） ，N（结果为负） ，C（进/借位） ，O（溢出）等等。 无条件跳转：JMP 调用子程序：CALL（RET 用于返回）

6.无参数控制类：过程返回 RET，停机 HLT 采用了流水线技术的 RISC 处理器（如 MIPS） 的所有指令执行时间相同（即：单周期处理器） ：指令周期 = 若干 CPU 周期（五级流水线：取指，指令译码，取操作数，执行，存操作数） CPU 周期 = 总线周期 = 若干 T 周期（一个 T 周期为一个脉冲，对应一个高电平+一个低电平）

**2.5** **计算机体系结构的改进** **-** **RISC** **与** **CISC** **各自特性与区别**【了解】RISC 指令的几个特点（与CISC 相比） ：1.指令定长：优点：指令译码简单，指令执行速度快，总线一次可以传输一条完整指令；缺点：指令数目有限，立即数长度受 限 2.采用加载/存储(Load/Store)体系架构：优点： 简化指令集，提高指令执行速度，仍能完成所有运算；缺点：无法像 CISC 直接采用内存地址作为参数，需要先加载到寄存器中，较麻烦 3.寻址方式简单，种类较少 4.指令集中的指 令数量较少 5.每条指令执行时间相同（即：单周期处理器）6.算术和逻辑运算指令普遍支持三操作数格式 7.只能对寄存器操作数进行算术和逻辑运算（Load/Store 的特性）8.程序代码量较大，执行复杂操作需要使用较多的简单指令（指 令集中的指令数量较少的特性）

**2.5-流水线基本原理，典型的三级、五级流水线划分，三种相关冲突及解决**【掌握】流水线的基本原理：将指令执行过程分为若干个阶段，每个阶段由一个专门的电路来完成，各个阶段之间采用流水线技术连接起来，使得每个阶段可以 并行执行，从而提高指令执行速度（原理：重叠使 IPC 大大增加） 。三级流水线：取指（Fetch） ，译码（Decode） ，执行（Execute） ；五级流水线：取指（Fetch） ，译码（Decode） ，取源操作数（Read），计算执行（Execute） ，回 写结果（Write Back） 。理想情况下，流水线的吞吐率是每个周期一个指令；事实上，n 级流水线需要额外插入 n-1 个流水线寄存器。流水线的三种相关冲突：（冒险）

1.资源相关（如总线冲突） 解决方案：哈佛结构（指令与数据用两套独立的总线） ，插入气泡（阻塞） 2.数据相关（后一条指令需要用前一条的结果） 解决方案：优化编译器，数据旁路（直接把结果传给需要的指令） ，插入气泡

3.控制相关（如条件跳转） 解决方案：转移延迟槽（在跳转指令后面插入一条指令，使得跳转指令后面的指令也能执行） ，动态转移预测（预测跳转的目标地址）

**Chap3-存储器系统** **3.2** **只读存储器** **-** **地址译码，字线、位线【理解】** ROM（Read Only Memory）：只读存储器，只能读取，不能写入【特殊情况可以写入】；掩模 ROM（Mask ROM）：只读（值由 mask 决定）；可编程 ROM（PROM，Programmable ROM） ：一次性可写（可以通过烧断熔丝来改写，但是不可逆） ；可擦除可编程 ROM（EPROM，Erasable Programmable ROM） ：可擦除可写（可以通过紫外线照射来擦除，每次擦除会擦除原有所有内容）； 电可擦除可编程 ROM（E²PROM， Electrically Erasable Programmable ROM）： 电可擦除可写（可以通过电压来擦除，每次擦除可以擦除指定内容）； 闪存 Flash：一种特殊的 EPROM，可以分块擦除，擦除速度比 EEPROM 快，但是寿命比 EEPROM 短。

地址译码：A1A0共同通过译码对应 W3-W0 中的某一个，每一根 Wi 线对应 4 个 bit，通过 d3-d0 输出。W3-W0 称为字线（Word Line）d3-d0 称为位线（Bit Line）（或数据线） ，A1A0称为地址线（Address Line） 。一个块有 22 × 22=16 个 bit **3.3** **随机存取存储器** **RAM**（Random Access Memory）SRAM（Static RAM） ：静态随机存取存储器，速度快，但是容量小，价格高 DRAM（Dynamic RAM） ：动态随机存取存储器，速度慢，但是容量大，价格低。

**3.4** **存储器与** **CPU** **的连接** **-** **地址空间与存储器连接，存储器的位扩展、字扩展【掌握】** eg 见右：

原理角度：存储器和 CPU 通过地址总线、数据总线、控制总线连接。 工程设计角度：存储器和 CPU 不在一起（CPU 在主板上） ，通过总线连接。

地址空间与存储器连接： 1.地址空间：CPU 可以访问的地址范围 2.地址总线宽度即 CPU 的地址空间大小 3.连接的线：不妨设 CPU 与 2N块 2M ×K 位的存储器芯片相连接： D0~Dk−1:（数据总线）双向传输数据用 A0~AM−1 :（地址总线）CPU 单向向 RAM 传输地址用 R/W#:（控制总线）CPU 单向向 RAM 传输，决定是读还是写

AM~AM+N−1 :地址总线）CPU 单向向 RAM 传输，决定选择是哪块存储器芯片（通过译码器译码，具体而言通过若干取反和与门实现 N 线变 1 线。总共地址总线数量为M+N

存储器拓展：①位拓展：将存储器的位数扩展为原来的2N 倍，即 2N个2M ×K 位的存储器芯片并联， 变成 2N ×2M ×K 位的存储器。实现： 简单将 D 线组合在一起即可，A 线, R/W#线, CS#线简单并联即可。②字拓展：将存储器的字长 扩展为原来的2N 倍，即 2N 个 2M ×K 位的存储器芯片并联，变成2M+N ×K 位 位的存储器。实现：将 D 线, R/W# 线, A 线低 M 位部分简单并联，A 线高 N 为部分通过 N→2N 译码器译码后连入 CS# 线。③复合拓展：先进行位拓展，位 拓展后作为一个整体再进行字拓展。

**3.5 高速缓冲- Cache 基本工作原理及作用【理解】**一个典型 Cache 行包含：缓存数据 512 位，标记信息 14 位，有效位 1 位，一致性控制位 1 位，替换控制位 2 位,共 530 位。多级 Cache：L1 Cache: 32~256 KB；L2 Cache: 512 KB（普通微机）>2MB

（某些服务器）；

**地址映射和转换：**

L3 Cache: 早期置于主板，现在置于 CPU ，多用于多个 CPU 之间共享数据.命中率 h=  ，L1 Cache 80% ；L1+L2+L3 Cache 95%

①全相联映射方式：主存中每个块都有可能在 Cache 中的任何一行中， 因此需要比较所有行的标记信息，效率低。查找方式：在块表中寻找块地址，如果找到则命中，否则未命中②直接相联映射方式：主存中每个块

只能映射到 Cache 中的某一行，因此只需要比较一行的标记信息，效率高。查找方式：比较 Cache 对应行上的标记信息，如果找到则命中，否则未命中③组相联映射方式：主存中每个块可以映射到 Cache 中的某一组，组内的行数为 N， 因此需要比较 N 行的标记信息，效率介于全相联和直接相联之间。分块方式： Cache 大小与主存每页的大小相同，Cache 的组大小和主存每组的大小相等， 即组外是直接映射方式，组内是全相联映射方式。Cache 2C = 2u ×2v 块 ；主 存 2M = 2s×2u ×2v块。查找方式：比较 Cache 对应组内的所有行的标记信息，如果找到则命中，否则未命中。**Cache 更新与替换策略:**①读取结构 1.贯穿读出：CPU 先访问 Cache ，未命中再访问主存 2. 旁路读出：CPU 同时访问 Cache 和 主存，Cache 的数据先传给 CPU，主存的数据后传给 CPU ，如果 Cache 命中则主存的数据被丢弃②写入更新策略 1. 写通方式：CPU 同时写入 Cache 和主存 2. 写回方式：CPU 直接写回 Cache ，在 Cache 被丢弃时若 Cache 中的数据被 修改则写回主存③替换策略 1. 随机替换：随机选取一行替换 2. 最不常用替换（ Least Frequecntly Used, LFU） ：替换一段时间内最少使用的行 3.先进先出替换（First In First Out, FIFO） ：替换最先进入的行 4.近期最少使用替换（ Least Recently Used, LRU） ：替换 CPU 最近最少使用的行。

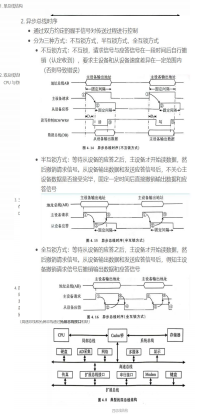
**Chap4-总线和接** **口** **4.1** **总线技术** **-** **总线操作与时序【理解】** 总线带宽 = 总线宽度 \* 时钟频率 ÷ 8（字节/秒）

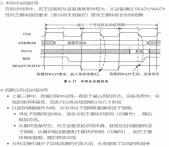
总线的分类： 1 按总线位置：按芯片内外： 片内总线，片间总线；按系统内外： 内总线（板级总线, 系统总线） ，外总线（I/O 总线,通信总线） 。2 按总线功能:数据总线：双向传输数据;地址总线：单向传输地址;控制总线：双向传输控 制信 3 按数据传输方式分类: 串行总线：一根线传输数据(单端：一根信号线，一根信号参考地线;差分：一对信号差分线);并行总线：多根线同时传输数据（时钟频率不能高，否则线间干扰） 。4 按时序控制方式分类； 同步总线：按照 相同的时间基准，规定的时钟周期进行数据传输；异步总线：收发双方信号，握手；半同步总线：平时按照同步总线工作，如果有一方跟不上则按照异步总线协调至同步。5 按时分复用方式分类：按照时间片，有时传输数据，有时传输 指令。

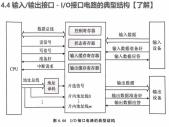
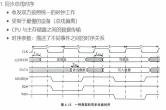
**总线仲裁：** **①**集中式仲裁: 串行仲裁 daisy chain 按照远近顺序仲裁;并行仲裁 按照优先级仲裁;混合仲裁 块内串行仲裁 块外并行仲裁

②分布式仲裁:主设备间自行协商,本质上是按照优先级，如果有优先级更高的设备在使用总线就等待，否则就使用总线（并发出总线请求）（发出请求的时候本质上不是“请求 ”， 而是“通知 ”）

**总线的结构**



**总线操作与时序**



**4.2** **片内总线** **AMBA** **-** **AHB** **数据传输过程，AHB“流水线** **”分离操作【理解】**AMBA（Advanced Microcontroller Bus Architecture）： 高级微控制器总线结构 ARM 公司提出的一 种片内总线标准.AMBA2 高级微控制器总线结构:1.AHB（Advanced High-performance Bus）： 高级高性能总线，主要用于连接处理器和存储器。2.ASB（Advanced System Bus）： 高级系统总线，主要用于连接处理器和外设 3.APB（Advanced Peripheral Bus）： 高级外设总线，主要用于连接外设。典型结构：基于一条高性能系统中枢（AHB / ASB） ， 连接多个低性能外设（APB） 。AHB / ASB 与 APB 之间通过桥接器（Bridge）（AHB-APB / ASB-APB）相连.

**AHB** **数据传输过程**首先，主设备通过 HBUSREQx 向仲裁器发起总线请求，仲裁器通过 HGRANTx 向主设备发出总线授权，随后主设备发送驱动地址信号和控制信号

1 单个数据简单传输（地址阶段，数据阶段）

i)HCLK 时钟信号 ii)地址阶段： HADDR 地址信号，控制信号 iii)数据阶段（可能存在多段） ：HWDATA（或 HRDATA）数据信号，HREADY 是否已准备好传输

2 单个数据简单传输中插入等待状态: 同 (1)，但是 HREADY 会有一段时间为低电平。在 HREADY 变成高电平之前，数据传输不开始

3 多个数据的传输: 类似流水线。在上一个数据的地址和控制信号发送完毕之后，下一个数据的地址和控制信号就可以发送了。当上一个数据传输完毕后，下一个数据的数据信 号就可以发送了（此时地址和控制信号才撤去）

4 流水线分离 i)如果从机不能在下一个时钟周期相应，可以通过 HRESP[1:0] 发出启动 SPLIT 传输的响应（此时仲裁器会将总线使用权让给其他主机） ii)从机做好数据传输准备后，通过 HSPLITx[15:0] 发出重新启动传输信号，主机根据优先级适当重新分配总线使用权

iii)HMASTER[3:0] 记录当前使用总线的主机编号，HSPLITx[15:0] 中应对对应的主机编号置 1（HMASTER 为一个 0 ~ 15 的二进制数，HSPLITx 的 16 位分别对应 16 个主机）

**AHB 仲裁过程**

1 主机通过 HBUSREQx 信号发出对总线的请求（若希望使用时锁定资源，需要同时发出 HCLOCKx） 2 仲裁器通过 HGRANTx 指示主机 x 获得总线使用权

3 若 HREADY 有效，仲裁器通过改变 HMASTER[3:0] 以指示当前获得总线使用权的主机号

**Chap5-ARM** **处理器体系结构和编程模型**

**5.1** **ARM** **体系结构与ARM** **处理器概述** **-** **微架构的概念、哈佛结构的特点以及与冯.** **诺依曼结构的区别【了解】**CA（计算 机体系结构） ISA(指令集体系结构) μarch（微架构）

指令集体系结构 ISA：描述软件如何使用硬件的一种规范和约定，是程序员眼中的概念结构和功能特征，具体地就是程 序员编程时能看到或者能用到的资源以及使用方式

微架构（Microarchitecture） ：对应 ISA 的硬件实现方式，即数字电路以何种方式来实现处理器的各种功能，包括运 算器，控制器，流水线，超标量和存储系统结构等内容，也就是计算机的组织和实现技术。同一个 ISA 可以通过不同微 架构来实现，但是只要基于同一个 ISA 即使使用了不同的微架构，也能在软件层面做到互相兼容。处理器的分类：①基 于 ISA：CISC，RISC 这一部分的区别可以在 [2.5](https://4tb.brealid.cn/2023/jiqian) 中找到②基于微架构：冯 ·诺依曼结构，哈佛结构③Cortex-A 是面向移 动计算、智能手机、数字电视、企业网络和服务器的高性能处理器；Cortex-R 是面向实时应用的处理器，聚焦于高性能

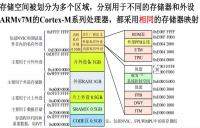
实时应用；Cortex-M 是面向嵌入式应用的处理器，低成本、低功耗、 小体积、 高性能。冯 ·诺依曼结构：指令和数据共用一条总线，存放在同一个存储器中，访问都是通过同一个

总线进行的。哈佛结构：指令和数据分开存放，指令和数据分开传输，指令和数据分开访问，指令和数据分开处理。 【缺点】成本增加；设计复杂，连接难度较大【优点】可以消除取指和取操作数之间的资源相关；指令数据分开

存储，指令和数据宽度可以不同；提高了存储器，总线，CPU 的利用率。

**5.2** **Cortex-M3/M4** **处理器结构** **-** **Cortex-M3/M4** **处理器的存储器映射及总线系统【掌握】**

Cortex-M3/M4 的共同技术特性： 内部有一条三级流水线；采用哈佛结构；32 位处理器（可以处理 8, 16, 32 位数据） ；多条总线、NVIC、调试和跟踪组件；没有 Cache 或 TCM， 没有协处理器接 口， 没有 MMU；Systick、双堆栈指针、特 权和非特权访问等级；可以选配 MPU，实现内存的分区保护；低功耗，低成本；具有丰富的开发调试工具。 **区别：**分别基于 ARMv7-M 和 ARMv7E-M 架构；Cortex-M4 增加了 DSP 运算指令；可选配单精度 FPU（浮点数处理单元），可在全 部计算完成之后再进行浮点数的舍入，减少舍入误差以提高 MAC 结果的精度；增加了支持 8 位和 16 位数据的 SIMD 指令，允许对多个数据同时进行并行处理；支持多个（包括 SIMD 在内）的饱和运算指令，避免在出现上溢出和下溢出 时计算结果出现较大畸变；支持单周期 16 位、双 16 位以及 32 位乘加（MAC）运算。

经典 ARM 处理的 7 种异常类型和 6 级优先级：优先级 1: 复位（Reset） ；优先级 2：数据中止（Data Abort） ；优先级 3：快速中断（Fast RQ） ；优先级 4：外部中断（IRQ） ；优先级 5：预取中止（Prefetch Abort)优先级 6：未 定义指令（Undefined）和软件中断（SWI）

**处理器系统**

Cortex-M3/M4 内核（包括 CPU、NVIC、Systick 以及可选的指令跟踪接 口；Cortex- M4 内核可选配 FPU） Cortex-M3/M4 处理器（Bus Matrix 总线交换矩阵 可选配 MPU 内存保护单元）

Cortex-M3/M4 处理器系统（WIC (Wake-up Interrupt Controller)电源控制和中断 CoreSight 调试架构）

**存储器管理特性：**Cortex-M3/M4 没有 Cache 或者 TCM ；4GB 线性地址空间（AHB-Lite 属于 32 位总线）

支持小端和大端（芯片制造商可能只选择其中一种配置类型） ；支持位带操作 Bit-Band Opertaions 内置写缓冲，可以提高程序的执行速度；可选 MPU ，支持 8 个可编程区域，可提高系统健壮性

所有基于 ARMv7-M 的处理器都支持非对准（非对齐）传送，但将增加总线操作次数

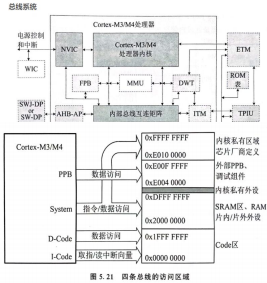
**存储器映射：**

4GB 地址空间分为不同区域

固定但统一的地址映射方案，有助于软件可移植性

程序代码既可以存放 CODE 区域，也可以存放在 AHB 总线所连接的 SRAM 中

**I-Code、D-Code、AHB** **系统总线、APB/PPB** **四套总线，还有** **DAP**





**注意：CPU** **通过内部总线互连矩阵直接访问内核私有外设，不经过四条总线**

①I-Code D-Code 总线：基于 AHB-Lite 总线协议的 32 位总线

0x0000 0000 ~ 0x1FFF FFFF 的 Code 区；I-Code 取指操作 D-Code 取数据操作 访问空间共用，物理上独立，彼此之间有一个仲裁器（解决冲突）；

Code 区的总线矩阵和总线复用器是两种不同的选件：总线矩阵：取指操作和 SRAM 的数据存取操作可以同时进行；总线复用器：对 Code 区的访问可以分时进行，不 再具有并行性，但能减少芯片电路数和芯片面积。

②System 总线：基于 AHB-Lite 总线协议的 32 位总线，也称为 AHB 总线；0x2000 0000 ~ 0xDFFF FFFF 和 0xE010 0000 ~ 0xFFFF FFFF 之间的数据传送。

③APB/PPB 总线（Private Peripheral Bus） ：连接 0xE004 0000 ~ 0xE00F FFFF 之 间的外部私有外设。

④调试访问端 口 DAP：连接内部调试访问端口 AHB-AP 和外部调试端 口 DP（SWJ-DP SW-DP 等等） ；AHB-AP 和内部总线互联矩阵之间有一条基于“增强型 APB 规格 ” 的 32 位总线。

Cortex-M3/M4 不能直接连接片外存储器，必须使用片外 RAM 控制器作为接 口。

嵌套向量中断控制器 NVIC (外部中断、NMI、 系统异常) 2、 中断向量表(异常/中断服务程序的入口地址、重定位 ;中断响应时，应由 I-Code 总线负责读

取中断向量， 而压栈操作理论上可由 D-Code 或者系统总线完成)3、 系统节拍定时器 SysTick(定时信号，计数器) CMS IS 微控制器软件接 口标准

中断向量表：地址偏移 = 异常类型（编号） × 0x04（即中断的进入地址） ;CMS IS 中断信号 = 异常类型（编号）-16

**异常处理特性：**

除了复位，NMI 之外其他所有异常 / 中断都可以被屏蔽；除了复位，NMI 和硬件错误之外，其他所有异常 / 中断都可以单独使能或进制

除了复位，NMI 和硬件错误具有固定的（高）优先级之外，其他所有异常/中断都具有多达 256 级可编程优先级；支持优先级的动态修改；支持向量中断/ 异常方式;向量表可以重定位在存储器中的其他区域；低中断处理延迟（零等待存储器系统， 中断处理延迟仅为 12 个时钟周期） ；可由软件触发

可以按照优先级进行中断屏蔽;进入中断时可以自动保存包括 PSR 在内的多个寄存器，异常返回时自动回复，无需另外编程 可选配唤醒中断控制器 WIC 支持睡眠（Sleep）和深度睡眠（Deep Sleep）

**5.3** **Cortex-M3/M4** **的编程模型** **-** **Cortex-M3/M4** **处理器** **2** **种操作状态，2** **种操作模式，2** **种访问等级（切换原理）【了解】**默认：Thumb 状态，特权线程 模式

①2 种操作状态——Thumb 状态和调试状态，调试器都可以访问系统存储器，包括位于处理器片内和片外的各种外设 Thumb 状态:1)执行 Thumb 指令程序代码 2)没有 ARM 状态（因为 Cortex-M 系列处理器不支持 ARM 指令集）

调试状态:1)处理器被暂停之后，例如通过调试器发布暂停命令 or 触发程序断点之后会进入 2)可以通过两种方式进入 1.调试器发起暂停请求 2.处理器中调试部件产生调试事件 3)调试状态下，调试器可以访问或修改处理器中寄存器数值

②2 种操作模式——操作模式又称为处理器工作模式 or 运行模式。Cortex-M3/M4 将经典 ARM 处理器的 7 中运行模式归并成以下两种操作模式

处理模式:1)即异常处理模式 2)该模式下执行的是异常/中断服务程序（ISR）3)相当于经典处理器中的5 种异常模式整合为一种模式 线程模式:除了处理模式之外的所有运行模式

③2 种访问等级——处理器处于线程模式时，特权访问等级有特殊寄存器 CONTROL 控制（特权模式下可以改变该寄存器以切换模式；非特权模式下无法改变该寄存器，只能借助异常机制才能切换模式这是一种最基本的安全模型，可以防

止用户程序对系统资源的非法访问） 1.特权访问等级：【处理模式】处理器处于处理模式下，正在执行的是异常/中断服务程序，特权访问等级可以访问处理器中的所有资源 【线程模式】特权线程模式 sys

2.非特权访问等级 1）【线程模式】非特权线程模式 usr 2）有几条指令无法执行（用法错误异常 Usage Fault）

3）不能访问大部分的内核私有区域， 以及某些特殊寄存器（如 NVIC 寄存器） 4） 系统中配置了 MPU 且划定某些区域智能特权等级访问（MemManage Fault 异常）

**5.3-** **Cortex-M3/M4** **处理器** **16** **个常规寄存器及程序状态寄存器** **PSR【掌握】**

**R0~R12** **通用寄存器** 1）按位：R0~R7：低位寄存器（Low Register） ；R8~R12： 高位寄存器（High Register）可用于 32 位指令和少数几个 16 位指令（如 MOV）

2）按子程序调用过程：R0~R3 用于子程序之间的参数传递；R4~R11 用于保存子程序的局部变量；R12 作为子程序调用的中间寄存器 3） 复位后初始值均未定义 **R13：堆栈指针** **SP**（Stack Pointer） **R14：链接寄存器** **LR**（Link Register） **R15：程序计数器** **PC**（Program Counter）

1）实际上有两个物理栈指针：MSP（Main Stack Pointer） ：主堆栈指针，用于处理器的异常处理 PSP（Process Stack Pointer） ：进程堆栈指针，用于线程模式下的进程堆栈2）对于一般程序而言，两个堆栈指针寄存器只有一个可见 **特殊寄存器**必须先通过 MSR/MRS 指令对其进行访问 MRS <reg>, <special\_reg> 读 special\_reg 到通用寄存器 reg ； MSR <special\_reg>, <reg>将 reg 内容写入 special\_reg 有以下这些：

**PSR** **程序状态寄存器：①**经典 ARM 处理器（旧，不使用） ：1）CPSR（Current Program Status Register）： 当前程序状态寄存器 2）SPSR（Saved Program Status Register） ：保存程序状态寄存器

②ARMv7 开始的： xPSR 代替了 CPSR，包括 APSR 应用程序状态寄存器；EPSR 执行程序状态寄存器； IPSR 中断程序状态寄存器 特殊寄存器还有异常/中断屏蔽寄存器 PRIMASK, FAULTMASK, BASEPRI 处理器控制寄存器 CONTROL SCB 系统控制块

**5.3-** **堆栈的原理，Cortex-M3/M4** **处理器的堆栈模型（满递减）及双堆栈结构** 地址增长方向+堆栈指针指示位置 满递增（FA，Full Ascending）栈：SP 指向最后压入的数据，且由低地址向高地址增长

满递减（FD，Full Descending）栈：SP 指向最后压入的数据，且由高地址向低地址增长 **Cortex-M** **系列只能使用满递减栈** 空递增（EA，Empty Ascending）栈：SP 指向最后压入的数据的下一个位置，且由低地址向高地址增长

空递减（ED，Empty Descending）栈：SP 指向最后压入的数据的下一个位置，且由高地址向低地址增长

**Cortex-M3/M4** **的双堆栈模式**分别服务于不同的操作模式和特权访问等级；本质上在一个栈里（SRAM 中） ，但是通过两个栈指针来实现； 高部分存储用户级（PSP 初值位于这一段的最高处 P:Process） ；低部分存储特权级（MSP 初值位于这一段的最高处 M:Main）

**5.4** **Cortex-M3/M4** **的处理器存储系统** **-** **位段（位带）** **优点：**操作代码更加简洁；C 语言中本身不支持，可以通过 #define 人为设置别名

位段和位段别名 1）一次仅能访问一个位 2）有两个预定义的存储器区域支持这种操作3）对位段区域的访问没有特殊指令，会被自动转换

SRAM 区域的最低 1MB 0x2000 0000 ~ 0x200F FFFF => 0x2200 0000 ~ 0x23FF FFFF；外设区域的最低 1MB 0x4000 0000 ~ 0x400F FFFF => 0x4200 0000 ~ 0x43FF FFFF

**5.5** **Cortex-M3/M4** **的异常处理** **-** **异常处理的基本过程，及异常优先级及优先级分组（概念）**

**基本流程：** **①**异常处理的接受②异常进入流程（多个寄存器和返回地址被压入当前使用的栈；更新内核寄存器和多个 NVIC 寄存器）③执行异常处理程序④异常返回 根据终端类型好在异常向量表中查询对应的异常处理程序入口地址，优先级高的异常对应可以打断优先级低的异常处理程序

**异常状态：** **①**非激活状态（ Inactive State） ：异常未被激活，处理器正常运行②激活状态（Active State） ：异常被激活，处理器进入异常处理模式，执行异常处理程序

③退出状态（Exit State） ：异常处理程序执行完毕，处理器退出异常处理模式，恢复到非激活状态④激活并挂起状态（Active and Pending） ：异常被激活，但是优先级不够，处理器仍然处于非激活状态

**异常优先级：**3 个固定的最高优先级 -3, -2, -1。256 个可编程优先级 之所以采用低位舍弃，是因为这样在不同的芯片设计下，优先级的大小关系不会发生变化（可移植性）

若芯片设计只实现了 3 位优先级，则 0x00, 0x20, 0x40, …, 0xC0, 0xE0 可用；若芯片设计只实现了 4 位优先级，则 0x00, 0x10, 0x20, …, 0xE0, 0xF0 可用；若只实现了 6 位优先级，则 0x00, 0x04, 0x08, …, 0xFC, 0xFF **优先级分组：**8 位优先级配置寄存器分为两个部分：分组(抢占）优先级和子优先级；子优先级只会在两个相同抢占优先级的中断同时产生的时候起作用。

**Chap6-基于** **ARM** **微处理器硬件与软件系统设计开发**

二元算术指令：加法 ADD, 减法 SUB, 与 AND, 或 OR 等等。一元算术指令：取反 NOR 等等。读取写入：LDR 读取，STR 写入

寄存器赋值：MOV。有参数控制类（一般参数都是 address） 无参数控制类：过程返回 RET，停机 HLT

拓展指令：MOVW, MOVT: 16 位立即数赋值 MOVW Rd, Imm16，MOVT Rd, Imm16 MOVW 用于赋值低 16 位，MOVT 用于赋值高 16 位 MVN：取反后赋值 MVN Rd, Rs、