## CHAP2:

**冯·诺依曼结构的计算机系统的五大组成部分**：运算器、控制器、存储器、输入设备、输出设备
**微程序设计思想**：一个机器指令编写一个微程序，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
“ADD R1，R2，R3：1.取指 2.指令译码 3.R2->加法器 4.R3->加法器 5.加法器完成相加 6.结果->R1 7.更新状态寄存器 8.PC=PC+n
**控制器：**微程序控制器和硬连线控制器
**指令周期：**开始取指令到完成指令操作的时间，一个指令周期分为多个 CPU/机器/总线周期，一个总线周期包含多个 T/时钟周期
**指令总结**：指令执行时，指令操作码送到 IR，经 ID 译码后 OC 产生操作控制信号。指令的地址码送到地址形成部件，生成地址信号。CPU 中的所有数据都在数据通道中传送。指令执行过程属于多级串行作业，始终有一部分部件处于空闲状态，部件的利用率不高。模型机属于冯·诺依曼结构，串行作业方式的取指和存取操作数在时间上相互错开，不会出现冲突。如果改用流水线方式，前后指令的取指和存取操作数可能同时发生。冯·诺依曼结构将造成总线竞争，故不太适合流水线模式
**CISC：**指令数量多、功能丰富、可实现复杂操作 1. ）指令长度不一 2.非 Load/Store 体系，算术和逻辑运算指令的操作数可以是存储器数 3. MOVE 操作 4.两操作数 5. 指令功能强大、寻址方式多样、程序简洁 6.流水线和超标量很难在 CISC 上实现
**RISC：**采用硬连线方式实现控制器，采用精简指令集指令简单、长度一致、执行时间相同，这些特点使其易于引入流水线和超标量等可大幅度提高处理器性能的并行处理技术，CISC 处理器一条复杂指令就能实现的操作，RISC 处理器需要使用多条简单指令才能完成 1. 寻址方式简单，种类较少 2. 指令集中的指令数量较少 3. Load/Store 体系结构；4. 每条指令长度一致，执行时间相同 5. 面向寄存器的编程思想（早期的 CISC 属于面向累加器）6. 算术和逻辑运算指令普遍支持三操作数 7. 只能对寄存器操作数进行算术和逻辑运算
**流水线**：三级：取指->译码->执行 五级：取指->译码->取操作数->执行->回写
**流水线寄存器：**将功能部件按指令操作步骤顺序进行排列部署，前后部件之间增加缓冲寄存器，构成指令处理流水线前后两个部件经过缓冲寄存器隔离后，可以相对独立地并行工作
**流水线冲突：**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）（使用 BTB（转移目标缓冲器），收集和存储了近期所有转移指令的有关信息，并按照查找表的形式进行组织）
**超标量技术：**拥有标量多条流水线，通过空间并行方式提高处理能力。（1.将多条指令分发到多条流水线上，同时执行）（2.分发前需配对检查，不同流水线上的指令不相干）（3.可以使用不同类型的流水线，如整数与浮点数）。
**线程（Thread）**：能独立执行的代码最基本单位。
**多线程技术：**为进一步减少“窝工”，添加少量部件对流水线升级改造，使处理器在同一时间可执行两个线程，称为同时多线程（Simultaneous Multi-Treading）技术。
**MIPS:**：单字长定点指令平均执行速度,它不能客观反映计算机的运算速 度，因为不同的处理器的指令能力不同，换句话说完成同一指令动作，可能消耗的指令数量不一样。

## CHAP3:

**.存储器**：半导体/磁介质/光存储器三种（半导体存储器是一种以半导体电路作为存储媒体的存储器，按照功能分为 RAM 与 ROM，优点：体积小/速度快/密度高/接口容易；缺点：RAM 为易失性存储器，EPROM 编程次数有限）（磁介质：使用磁性介质的磁极化存储信息。早期为主存。现在为外设，需接口电路支持。非易失性存储器，容量大，存取速度越来越高，体积不断减小。主要的外设，应用广泛）（光：使用反射光强记录，只读，可记录）
**存储体系架构：**二级（主存+辅存）四级（寄存器 + Cache + 主存 + 辅存（联机、脱机外存））
**接口标准**：计算机常用储存接口 IDE/SCSI/STAT/SAS 嵌入式设备常用（含有控制器与存储单元）SD/eMMC/UFS
**存储器性能**：存储容量，存取时间，存取周期，数据传送速率（带宽）
**ROM：**（ROM 早已不是“只读”，仍称为 ROM 只是一种习惯只读，是相对于 CPU 的读写控制电平及其控制逻辑而言）：1. Mask ROM（掩膜 ROM）的存储数据由专门设计的掩模板决定，为固化数据，用户不能修改 2.PROM：出厂时全写了 1，用户可写入 0，一次可写 3.EPROM 用紫外线擦除，一次全部擦除 4.EEPROM 电压擦除，可以以字节为单位擦除，时间远远小于 PROM 5.FLAHSH：按块可擦除，（1、Flash 没有选通管 T，因此集成度高，容量大，但稳定性和擦写次数都不如 E2PROM。2、E2PROM 可按位擦除，Flash 只能按块（或按页或行）擦除。）
**NOR flash**：可以按位随机读取，读效率高，故可用于存放程序；写入操作之前要先擦除，而擦除只能按“块”进行，故写入和擦除速度较慢。各个基本存储单元 Bit Line 是并联的，可按位随机读取
**NAND flash**：也需要先擦除再写入，读、写、擦出均需要以块为单位。各个基本存储单元 Bit Line 是串联的，随机读取速度慢且不能按字节随机编程
**SRAM**：双稳态触发 不需刷新 集成度低 速度快 容量小
**DRAM**：电容 需要刷新 集成度高 速度慢 容量大
**存储器扩展**：字扩展/位扩展/复合扩展
**内存条**：扩展性/互换性/灵活性/可维护性：
**Cache**:为了解决 CPU 和主存储器的速度差距问题而设计的以 SRAM 为基础的存储器。（根据程序访问的时空局部性，把经常访问的代码和数据保存到高速缓冲存储器（Cache)中，把不常访问的代码和数据保存到大容量的相对低速 DRAM 中，尽量减少 CPU 访问 DRAM 的概率，在保证系统性能的前提下，降低存储器系统的实现代价。）
**局限性原理**：时间/空间局限性 命中率 h=Nc/（Nc+Nm）（影响因素：Cache 容量、存储单元组数目和组大小、地址映射方案、联想比较策略、数据替换算法、写操作处理方法、程序本身特性等。）
**地址映射与转换：**
1.全：主存中的块可映像到 Cache 的任意位置，主存地址到 cache 地址转化通过查找块表实现， Cache 利用率高。缺点：比较和替换策略都需要硬件实现，电路复杂，只适用于小容量 Cache。访问相关存储器时，每次都要与全部内容比较，速度低，成本高。
2.直接：将主存空间按照 Cache 的大小划分为若干页，也称为区，页内分块。主存储器中一块只能映射到 Cache 的一个特定的块中。优点：地址映射方式简单，数据访问时，只需检查页号是否相等即可，因而可以得到比较快的访问速度，硬件设备简单。缺点：替换操作频繁。主存每个块在 Cache 中只有一个对应位置，若另一个块也要调入该位置，将会发生冲突，即使 Cache 的其它块位置空闲，也不能接受它，导致命中率低。
3.组：Cache 划分成大小相等的组，将总块数为 2C 的 Cache 分成 2u 组，每组 2v 块。主存容量是 Cache 容量的整数倍，将总块数为 2M 的主存划分为 2s 页，每页 2u 块，即主存每页的大小与 Cache 的组数相等。 优点：块冲突概率比较低，块利用率大幅度提高，块失效率明显降低。缺点：实现难度和造价要比直接映射方式高。
**更新**：贯穿读出（Look Through）旁路读出（Look Aside）写通方式（Write Through）写回方式（Write Back）
**替换**：随机（Random）替换策略最不经常使用（Least Frequently Used，LFU）替换策略先进先出（FIFO）替换策略 近期最少使用（Least Recently Used，LRU）替换策略
**虚拟存储器**：解决应用程序太大以至于内存容纳不下该程序
**虚存与 cache 比较：**相同点 1.两者都有利于提高存储系统的性能。2. 均基于程序局部性原理。用到的信息放到较快的存储器中
不同点 1.虚拟存储器未命中的性能损失要远大于 Cache 系统中未命中的损失 2. Cache 应对主存与 CPU 的速度差异，虚存解决内存容量问题 3. CPU 与 Cache 和主存之间均有直接访问通路，而虚存所依赖辅存与 CPU 之间不存在直接的数据通路 4.Cache 的管理完全由硬件完成，对系统程序员和应用程序员均透明。而虚存管理由软件（操作系统）和硬件共同完成，虚存对实现存储管理的程序员是不透明的。
**虚存地址变换：**段（主存按段分配，主存与辅存间信息传送单位是不定长的段）/页（主存按页分配，主存与辅存间信息传送单位是定长的页。）/段页（程序分为若干个段，每一个段再划分成若干页，以页为单位离散分配主存）虚地址由段号 S、虚页号 P、页内偏移 D 组成。

## CHAP4:

**总线特性（共享、分时）** **主要性能**：频率、宽度、带宽、同步方式、总线复用、信号线数、控制方式、寻址能力（地址总线位数与所寻存储器空间大小）、定时协议（为使源与目的同步，需要有信息传送的时间协议有同步、异步、半同步）、负载能力 **特性**：机械（总线部件之间的可靠连接）、电气（电气上的正确连接）、功能、规程特性（正确传输信息）
**分类：（片内、芯间总线）**（内：ISA、PCI、PCIe 外：SCSI、USB）（AB、CB 用于传输完成各项操作所需要的控制信号，协调计算机不同部件有序化地使用数据总线和地址总线、DB）（串行、并行）（异步、同步）（复用、非复用）
**结构**：单（结构简单，但影响了 CPU 与存储器的数据存取速度）、双（面向存储器（面向存储器的双总线结构信息传送效率较高。缺点：CPU 与 I/O 接口都要访问存储器时，会产生冲突）与 CPU（提高了微机系统信息传送的速率和效率。但外部设备与主存储器之间没有直接的通路，它们之间的信息交换必须通过 CPU 才能进行中转，会降低了 CPU 的工作效率））、三（任一时间只有一个能用，DMA 与主存总线不能同时对主存访问）、

四：PCI
**总线仲裁**（用于有多个主从机时）：集中式（串行、并行、串并混合）（将控制逻辑集中在一处）和分布式（将控制逻辑分布在各个设备上）
**总线周期四个阶段：**1.请求与仲裁 2.寻址 3.数据传输 4.结束
**1.同步时序：**收发按照统一时钟工作，受制于最慢的设备，模块间配合简单 **2.异步（不互锁、半互锁、全互锁）**：异步总线允许各模块速度的不一致性，提高了模块的适应性，没有公用的时钟，主从模块之间采用应答方式进行联络和协调工作。应答交互次数过多速度会慢 **3.半同步**：在同步时序中，若主设备和从设备速度差异较大，从设备通过 READY/WAIT# 线向主模块提出要求（表示尚未准备好）请求主模块延长时钟周期（提高了适应性，但是还是慢）**4.周期分裂：**将一个传输周期分为寻址与数据传输两个子周期，（减少了总线资源的无效占用，从而提高了总线的利用率）
AHB（主机、从机、仲裁器、译码器）• 地址和控制选择器负责将主机发出的地址和控制信号连接至从机。• 数据选择器负责主机和从机之间的数据信号连接。• 仲裁器在不同的总线主机之间进行总线使用权仲裁，并决定当前得到授权的主机能将它的地址和控制信号连通到哪个从机。• 译码器控制读数据选择器，读数据选择器负责将从机信号连接到对应的主机上。
**AHB 支持突发传输**、分裂式操作、单周期总线主机移交、单一时钟沿操作、非三态的实现、可扩展至更宽的数据总线架构
**流水线机制（地址与数据信息交叠）：**2 级流水线：第 n 次传输的地址在第 n-1 次传输时被驱动到了地址总线上。“驱动地址”和“驱动数据”两个操作构成 2 级流水线操作。从机不响应这个流水线会被打断。HSPLITx[15:0] 发出启动 SPLIT 传输的信号。仲裁器检测到 HSPLITx 后，知道从机当前不进行传输，则可以把总线的使用权出让给其他主机。当从机做好接收数据准备后，通过 HSPLITx[15:0]发出重新启动传输的信号，仲裁器许可后继续刚才挂起的传输操作。
**突发传输**：SINGLE 单次传输，INCR 地址顺序递增，WRAP 地址循环递增（4 个 4 个递增，在 16 字节发生回转），后面得数字表示突发长度，HSIZE 的值表示传输宽度为 8*2^x bit
**AHB 三通道：**地址、读写数据； AXI 五通道：读写地址、读写数据、写响应
**PCI 并行/PCIe 串行/USB 串行。串行还有 SAS，SATA**
**I/O 接口：**1.数据缓冲解决速度不匹配 2.设置电平转换电路解决电平不一致问题 3. 设置信息转换逻辑满足各自格式要求 4. 设置时序控制电路同步 CPU 和外设的工作 5. 提供地址译码电路 6. 提供 I/O 控制、读/写控制及中断控制等逻辑
**.I/O 端口**：指 I/O 接口中的各类寄存器，数据端口（用来存放 CPU 与外设需要交换的数据，长度一般为 1~2 字节。数据口主要起数据缓冲作用）/状态端口（指示外设的当前状态。）/命令，控制端口（用来存放 CPU 向接口发出的各种命令和控制字，控制接口或设备的动作。）
**.I/O 端口编址**：**1.内存映像编址**（系统中每个 I/O 端口都看作 1 个存储单元，并与存储单元统一编址，所有访存指令均可用来访问 I/O 端口，不用设置专门的 I/O 指令。）优点：(1)对 I/O 口的操作与对存储器的操作完全相同，无须专用的 I/O 指令；(2)外设数目或 I/O 寄存器数目几乎不受指令限制； (3)CPU 读/写控制逻辑较简单缺点：(1)占用了存储器的一部分地址空间；(2)增了地址译码电路的复杂性 **2.I/O 端口独立编址**（对系统中的 I/O 端口地址单独进行编址，不占用存储空间；使用专门的 IN/OUT 指令来访问 I/O 端口。）优点：(1)I/O 端口地址不占用存储器地址空间；(2)I/O 端口地址译码较简单，寻址速度较快；(3)使用专用 I/O 指令和存储器访问指令有明显区别，可使程序编制得清晰 缺点：(1)专用 I/O 指令类型少，远不如存储器访问指令丰富；(2)CPU 提供存储器读/写、I/O 端口读/写两组控制信号
**数据传送方式**：1.无条件传送 2.查询传送 3.中断传送 4.DMA（直接存储器访问）传送（又分为单次/数据块/请求传送），不能嵌套
DMA 完全依靠硬件，无程序不能处理复杂事件，无法对 IO 口寻址，所以有多个 DMA 需求时 DMAC 设置多个通道。
**并行接口**（无握手信号/有握手信号）：GPIO 是可编程并行接口
**串行接口**：有比特率和波特率（一般是相等的）
**串行优点**：（1）不仅传输距离远，而且串行数据传送速率比并行数据传送速率更快，因为串行通信的通信时钟频率比并行通信更容易提高； （2）抗干扰能力强，通信费用低； （3）传输线既传数据，又传联络信息。 缺点：需要串/并转换，对数据格式有要求。
IIC：串行同步
SPI：双全工同步总线，四根信号线，只有工作与空闲两个状态

## CHAP5:

**哈佛结构**:指令和数据分开存放,分开传输、访问、处理（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【缺点】成本增加、设计复杂,连接难度较大）
**Cortex-M3/4** :基于 ARM(E)-v7,三级流水线,哈佛结构。（32 位处理器,本身不含存储器,可选 MPU,没有 cache 与虚存,支持特权非特权访问的等级,M4 支持选配 MPU 与 DSP 扩展）
**处理器内核**:1.CPU(ALU,CU 和寄存器阵列),NVIC(嵌套向量中断控制器),SysTick(倒计时计数器产生系统定时中断),FPU(不支持双精度浮点数转换,M4 有专门浮点运算与数据传送指令,FPU 与 CPU 共用取指部件,译码-执行时两者并行)2.处理器：总线交换举证（基于 AHB 的交换网络），MPU（内存保护单元，把存储空间分成 8 个可互相重叠区域）3.WIC（在处理器处于深度睡眠下使能，有 NMI 或 IRQ 时唤醒系统），调试组件（包括 ETM，嵌入式跟踪宏单元 ITM，指令跟踪宏单元 DWT，数据观察点和跟踪单元 FPB： Flash 地址重载和断点单元 TPIU：，跟踪端口接口单元 ROM：存放调试组件的配置信息）coresight 调试框架（处理器内部的调试访问接口（AP）外部调试端口（DP）合称调试访问端口（DAP））
**存储器**：4GB 线性地址空间（AHB-Lite 属于 32 位总线，可通过接口连接 16.8 位的存储器）支持小端和大端（可能只选择其中一种配置类型）支持位带操作 Bit-Band Opertaions（对某一个 bit 取值/赋值）写缓冲，提高程序执行速度 MPU，内存的分区保护（可选）非对准传送（但额外增加总线传送次数）
**异常与中断处理**：除复位和 NMI 外，其它异常/中断都可以被屏蔽；除复位、NMI 和硬件错误外，其它异常/中断都可以单独使能或禁止；除了复位、NMI 和硬件错误具有固定的（高）优先级之外，其它所有异常/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（Cortex-M0/M0+无此特性）；可以按照优先级对中断进行屏蔽；中断响应时自动给出中断/异常处理程序入口地址；向量表可以重定位在存储器中的其他区域；进入中断/异常时可自动保存相关寄存器，异常返回时自动恢复； 低中断处理延迟；中断和多个异常可由软件触发；可选配唤醒中断控制器 WIC
**中断向量表**：存放异常/中断服务程序的入口地址，第一位是 MSP 的初始值，起始地址默认位于存储器空间最开始位置（中断响应时，若中断类型号为 n，只需计算 n×4（左移两位）即可
得到对应的中断向量在中断向量表中存放地址，可以快速获取中断服务程序的入口地址；为了加快中断处理速度，在中断响应时，读取中断向量以及中断服务程序的取指操作应与保护断点的寄存器压栈操作同时进行，处于 CODE 区由 I-CODE 总线读取而中断响应时的压栈操作理论上可由 D-Code 或者系统总线完成，以充分发挥哈佛结构的优势。但是为了减少一片 CODE 区的 SRAM，把数据都存放在由系统总线负责管理的 SRAM 区，因此压栈操作只能由系统总线完成）
**操作模式与状态**：Cortex-M3/M4 支持 16 位和 32 位指令并存的 Thumb-2 技术，不再有 ARM 和 Thumb 两种工作状态之分（状态：Thumb 状态和调试状态）（模式：处理模式与线程模式）（线程模式分特权和非特权）（有几条指令只有特权访问等级才能执行，使用这些指令将引起用法错误（Usage Fault）异常；非特权访问等级代码不能访问某些特殊寄存器，如 NVIC 内部的各种寄存器；非特权访问等级代码不能访问内核私有区域以及由 MPU 设定的保护区域否则有 MemManage Fault）（在特权模式下可以对 Control 寄存器进行写来切换非特权，非特权只能依靠异常机制才能转换特权）（无论是特权还是非特权线程模式，在异常响应时都进入处理模式，并具有特权访问等级）（线程模式可以切换使用独立的进程栈指针 PSP，使应用任务的栈空间和操作系统的主栈空间独立，提高系统健壮性和可靠性）
**通用寄存器**（复位后初始值均为定义 R0~R12 常规：R0~R7：低位，R8~R12：高位，R0~R3 子程序之间的参数传递 R4~R11 保存子程序的局部变量 R12 作为子程序调用的中间寄存器）（特殊：R13：堆栈指针 SP；R14：链接寄存器 LR；R15：程序计数器 PC）（R13 采用双字对齐，第三位地址为 0，R14、15 半字对齐，最后一位默认 0 但置 1 表示处于 Thumb 状态）
**特殊寄存器：**（没有映射到内存空间，只能利用 MSR（写）/ MRS（读）指令访问）（PRIMASK 置 1，屏蔽屏蔽除复位、NMI 和硬件错误以外所有的（优先级数值大于 0 的）系统异常和外部中断；FAULTMASK 当最低位被置位后，硬件错误异常也被屏蔽，相当于把异常/中断的优先级门槛提高到“-1；BASEPR 可伸缩设计宽度取决于中断优先级数量，中断优先级不少于 8 级，此时 BASEPRI[7:5]这 3 位用于设置中断屏蔽；如果设计了 32 级中断，BASEPRI 的宽度应为 5 位，若 BASEPRI[7:3] = 0b1 0000，将屏蔽优先级数值大于等于 0b1 0000 的所有中断；设置 256 级中断 BASEPRI[7:0]有效，全为 0 都不屏蔽，改为使用 PRIMASK）
**堆栈**：双字对齐，保护断点，现场，实现主程序与函数子程序的参数传递，存储局部变量，Cortex-M 系列处理器只能使用满递减（FD）（SP 指向最后压入的数据，且由高地址向低地址生长）双堆栈结构 MSP 上电后在中断向量表开头取出完成初始化，PSP 初始化需要另外编程（堆栈空间一般位于系统主存储器 SRAM 区（也可以放置在 CODE 区中通过 D-Code 总线的连接的 SRAM 中））
**位段操作**：对字节中的某一位直接进行修改，需要映射地址 0x2000 0000 的 8bit 对应 0x2200 0000-0x2200 001C，一个地址对应 32 个，1bit 对应 4 个从第 0 位数起。支持位带操作区域：1.SRAM 最低 1MB 区域（0x2000 0000-0x200F FFFF）2.外设区域最低 1MB(0x4000 0000-0x400F FFFF).
**向量表重定位（用于在运行时修改向量表）：**向量表重定位使用 VTOR（Vector Table Offset Register，向量表偏移寄存器）指示向量表的位置。VTOR 中保存向量表相对于存储器的起始地址（0x0000 0000）的偏移量。1.制器启动时，启动 ROM 中的 Boot loader 首先执行。在跳转到 Flash 中的用户程序前，设置 VTOR 指向用户 Flash 存储器的开始处，从而向量表切换为用户 Flash 中的向量表。2. 应用程序是从外部加载到 RAM 中执行。此时，存储在片上存储器的启动程序需要初始化相关硬件，把外部设备中的应用程序复制到 RAM，然后更新 VTOR，最后执行已加载至 RAM 的程序。
请求->挂起->激活： 挂起状态、挂起状态清除、进入激活状态、清除激活状态(线程模式 处理模式 线程模式)
中断使能有两个寄存器（用于使能和禁止）：防止使能或禁止时影响到其他中断的状态