第 1 章 概述

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

存储器，运算器，控制器，输入设备，输出设备。

1 存储器：（1）主要功能：存储程序和数据，

（2）分为主存储器和辅助存储器。

主存：读写速度相对较快，价格高，但容量受限、掉电后存储的信息消失

辅存：读写速度慢，价格低，但容量大、具有非易失性

2 运算器：（1）主要功能：完成各种数据运算和处理，ALU 负责实现算术或逻辑运算，寄存器是运算器内部的高速存储单元，访问速度最快

（2）主要构成：算术逻辑单元 ALU 和寄存器阵列

3 控制器：计算机的指挥控制中心

（1）主要功能： 根据指令对计算机各部件进行操控，协调各部件有序工作

（2）主要构成：指令寄存器 IR（Instruction Register），指令译码器 ID（Instruction Decoder），操作控制器 OC（Operation Controller）。

4 输入设备：将信息进行编码后输入计算机

5 输出设备：向外界输出计算机处理后的结果

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

三种编码表示方式：原码、反码和补码。原码：最高位符号位，其余为绝对值。反码：负数为数值取反。补码：负数为数值取反加一。

第 2 章 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

每条指令的执行过程都可以分解为一系列的微操作

1 微操作特点：可由简单电路实现，可被多个指令复用。

2 两种实现方式：（1）微程序控制器

a. 基本概念：指令执行过程看作多个微操作序贯执行完成的。

指令（微程序）=> 微指令=> 微操作=> 编码形成微操作码

对每个微操作进行编码，形成微操作码，微操作码可由简单电路产生微操作控制信号。执行顺序控制位：指示后续微操作的执行顺序。

微操作码+执行顺序控制位 = 微指令

指令：一段由若干微指令编排而成微程序。

所有指令对应的微程序都存放在控制存储器 CM 中，指令执行时，对应的（微程序的）微指令从 CM 中（逐条）读出，其中微操作码经过译码产生微操作控制信号。

b. 工作原理和过程

计算机指令分为操作码和操作数地址两部分

其中，操作码由指令译码器译码，译码结果是该指令对应的微程序在 CM 中的首地址（微程序的入口地址），该首地址经微地址译码器译码后，从 CM 中读出第一条微指令，其中微操作码部分送往微操作码译码器进行译码，生成相应的控制信号以实现规定的微操作。执行顺序控制位送往微地址形成电路，生成下一条微指令的微地址（微指令地址），不断重复上述过程，直到这段微程序全部执行完

c. 特点：硬件电路简单，可支持复杂指令，速度慢

（2）硬连线控制器

也称为组合逻辑控制器，最早采用的控制器设计方法把控制器看作专门产生固定时序的控制信号的逻辑电路，以使用元件少和速度快作为设计目标。因指令功能的多样性和差异性，导致所实现的控制器逻辑电路复杂、规模庞大，并且一旦形成就无法变更，除非重新设计和重新布线。

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

对存储器的要求：速度快、容量大、成本低

破局方法：分级存储体系结构

使用外存满足大容量、低成本和非易失的要求，使用 DRAM 型内存，兼顾容量、速度和成本

使用高速缓存 cache，减少 CPU 访问内存的开销。cache 位于 CPU 与内存之间，为 SRAM 型小容量快速存储器，用于存放 CPU 最近使用过或者可能要使用的指令和数据。

小端和大端格式（基本概念）了解

小端格式：高位高地址，低位低地址

地址 m m+1 m+2 m+3 内容 11 22 33 44（字 44332211）

大端格式：高位低地址，低位高地址

地址 m m+1 m+2 m+3 内容 44 33 22 11

字长与字的对齐 了解

16 位机，字起始地址为 2 的倍数，32 位机，字起始地址为 4 的倍数

2.3 模型机 CPU 子系统

（1）构成

1 运算器

a. 算术逻辑单元 ALU（Arithmetic Logical Unit）负责运算，也是数据传送的一条重要途径

组成：带有先行进位功能的全加器（简称加法器）、移位寄存器以及相应的控制逻辑

b. 累加器 ACC（Accumulator） 提供需要送入 ALU 的操作数，存储 ALU 的计算结果

c. 暂存寄存器 暂时存放需要送入 ALU 的操作数，但不存放计算结果；暂存寄存器是透明的，程序员不可见。

d. 标志寄存器 FR（Flag Register） 标志寄存器 FR：分为状态（条件码）位和控制位

状态位：记录 ALU 运算后的状态或者特征；控制位：对 CPU 的某些行为进行控制和管理

2 控制器 整个 CPU 的指挥控制中心

①功能和作用

根据指令中的操作码和时序信号，产生各种控制信号，对系统各个部件的工作过程进行控制，指挥和协调整个计算机有序地工作

②控制器的主要构成

指令寄存器 IR（Instruction Register）：临时存放从内存或者 Cache 中取出的下一条待执行指令，其输出作为指令译码器的输入。

指令译码器 ID（Instruction Decoder）：计算机能且只能执行“指令”；指令由操作码和地址码两部分构成；指令译码器只对操作码进行译码，分析和识别指令应该执行什么样的操作。

操作控制器 OC（Operation Controller）：根据指令译码器的译码结果，产生所需的各种控

制信号并发送到相关部件，控制这些部件完成规定的操作。操作控制器内部包括时序脉冲发生器、控制信号发生器、启停电路和复位逻辑等。

有种观点认为还包括程序计数器 PC（ProgramCounter）： 存放下一条待执行指令在内存中的地址。

（2）微操作

每条指令的执行过程都可以分解为一系列的微操作

微操作特点：可由简单电路实现；可被多个指令复用

两种实现方式：微程序控制器和硬连线控制器

a. 微程序控制器

计算机指令分为操作码和操作数地址两部分

特点：硬件电路简单，可支持复杂指令，速度慢

过程：①操作码由指令译码器译码，译码结果是该指令对应的微程序在 CM 中的首地址（微程序的入口地址）（所有指令对应的微程序都存放在控制存储器 CM 中）

②该首地址经微地址译码器译码后，从 CM 中读出第一条微指令

③其中微操作码部分送往微操作码译码器进行译码，生成相应的控制信号以实现规定的微操作

④执行顺序控制位送往微地址形成电路，生成下一条微指令的微地址

⑤不断重复上述过程，直到这段微程序全部执行完毕

b. 硬连线控制器

把控制器看作专门产生固定时序的控制信号的逻辑电路，以使用元件少和速度快作为设计目标。

特点：速度快，电路复杂，不支持复杂指令。调试和改动困难，一度被微程序取代。近年因 RISC 的兴起和 VLSI 的进步，又重新焕发青春，再度兴起

（3）寄存器阵列

也称为寄存器组、寄存器堆和寄存器文件。CPU 内部的若干高速存储单元，每个都有编号或

名称。CPU 与寄存器之间的数据交换传送速度最快。寄存器数量有限。分为专用寄存器和通用寄存器两大类。

（4）地址和数据缓冲器：CPU 内部总线与系统总线之间的接口

（5） 数据通道：数据是在运算器、寄存器阵列和系统总线接口之间通过内部总线进行传送，所以这几个部件也被称为数据通道（data-path）

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

（1）概念

a. 指令：分为微指令、机器指令和宏指令。微指令：微程序级的命令；机器指令：简称指令是 CPU 能识别和直接执行的一条二进制编码序列，包括操作码和操作数两部分；宏指令：由若干条机器指令组成的软件指令。

b. 指令系统：一台计算机中所有指令的集合

c. 汇编指令：助记符 to 操作码，标号和符号 to 指令和操作数地址

d. 指令周期：开始取值到完成指令操作的时间

一个指令周期分为若干个 CPU 周期（也称为机器周期），一个 CPU 周期等于一次取指时间，也称为总线周期。

一个总线周期包括若干个 T 周期，也称为时钟周期，T 周期或时钟周期是处理器最基本的时间单位。

（2）指令执行流程

用汇编语言编写的程序称为汇编语言源程序，简称汇编程序。汇编后的机器指令顺序存放，若指令长度为 4 字节，后一条指令地址等于前一条地址加 4（PC 的单位为字节）。

①PC 内容 0x20000000 送至地址缓冲器/驱动器，地址总线的输出经地址译码器译码，寻址内存单元

②PC 值自动加 4（假设 PC 内容的单位是字节），指向下一条指令的存放地址（何时修改 PC

有不同的策略 ）

③ OC 发读信号，将“E3 A0 06 FF”读出到数据总线；

④ 由于是取指操作，数据总线上的数据被装入 IR

⑤ ID 对操作码译码，OC 产生相应的控制信号

⑥ 第一条指令源操作数是立即数（取指时能从指令编码中立即得到的数），被装入 R0 寄存器后指令执行完毕

2.5 计算机体系结构的改进 RISC 与 CISC 各自特性与区别 了解

（1）CISC

指令长度不一，非 Load/Store 体系（运算不必用寄存器），MOVE 操作（寄存器间和寄存器与存储器之间复制传送），两操作数，指令功能强大、寻址方式多样、程序简洁，CISC 处理器采用微程序控制器

性能问题：完成一条指令需要从控制 ROM 中顺序读出多条微指令，需要多个在时间上序贯执行的微操作，这种在时间上的串行作业模式将影响指令的执行

解决思路：1. 提高处理器的工作时钟频率，加快微操作的节奏，但是增加时钟频率受到半导体材料物理特性的限制，并且难以消除由此产生的功耗和发热问题.

2. 使用流水线和超标量等技术，让多条指令在时间上并行执行.但是囿于 CISC 体系结构的特点，流水线和超标量的设计和实现遭遇了很多困难

（2）RISC

寻址方式简单，种类较少指令集中的指令数量较少；Load/Store 体系结构；每条指令长度一致，执行时间相同；面向寄存器的编程思想（早期的 CISC 属于面向累加器）；算术和逻辑运算指令普遍支持三操作数；只能对寄存器操作数进行算术和逻辑运算；程序代码量较大，因为执行复杂操作需要使用较多的简单指令。

特点：摒弃微程序设计思想，采用硬连线方式实现控制器；

为了减少硬件实现难度，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

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

（1）流水线技术

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。多条指令可以在流水线上以时间重叠方式序贯执行

五级流水线：Fetch 取指 Decode 译码 Read 取操作数 Execute 执行 Writeback 回写

三种相关冲突：

a. 资源相关，也称为结构相关：多条指令在同一个周期内争用同一个公用部件

解决：1 后面一条指令等待一个节拍再启动，2 采用哈佛结构

b. 数据相关：后一条指令执行需要使用前一条指令的结果，例如

写后写 WAW 后一条指令试图在前一条指令写前写指令

读后写 WAR 后一条指令试图在前一条指令写前读指令

解决：1 定向推送（前递），前一条指令执行结果通过专用通道直接推送给下一条，减少一个流水线周期，可减少数据相关。2 优化编译器，对前后指令进行检查，调整执行顺序

c. 控制相关：遇到转移指令时，后续已进入流水线的指令都应清空。

减少转移代价的方法：1 对于无条件转移指令，增加电路，在译码阶段提前计算转移目标地址 2 转移预测技术

（2）转移预测技术

转移延迟槽（branch delay slot）：转移指令 Ij 后面的一个时间片。无论是否转移，位于转移延迟槽的指令总是会被执行。可根据预测结果选择合适的指令“装入”转移延迟

槽

动态转移预测：根据转移指令过去的行为进行预测

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。BTB 不能太大，一般为 1024 个表项，其内容包括：转移指令 Ij 的地址（查找表索引）；Ij 转移可能性的量化结果（2bit 权值）；转移目标指令 Ik 的地址。

每条指令在取指时，处理器根据其地址在 BTB 中进行快速搜索，若有记录则表明这是转移指令，再根据其“档案” 进行相应处理，最后根据这条指令的实际行为对其进行打分，修正其在 BTB 中的“档案”记录。

2.8 计算机性能评测 了解

第 3 章 存储器系统

3.2 只读存储器 地址译码，字线、位线 理解

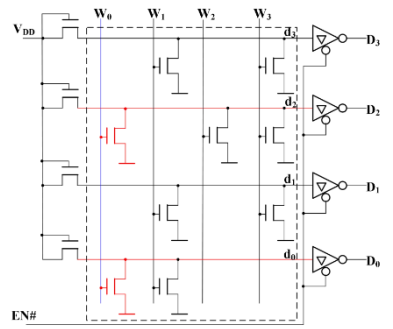
（1）地址译码：ROM 电路结构包括存储矩阵、地址译码器和输出缓冲器三个组成部分。

存储矩阵由许多存储单元排列而成。存储单元以用二极管构成，也可以用双极型三极管或 MOS 管构成。每个单元能存放一位二值代码（0 或 1）。每一个或一组存储单元有一个对应的地址代码。

地址译码器的作用是将输入的地址代码译成相应的控制信号，利用这个控制信号从存储矩阵中将指定的单元选出，并把其中的数据送到输出缓冲器。

三态输出缓冲器的作用有两个，一是能提高存储器的带负载能力，二是实现对输出状态的三态控制，以使与系统的总线连接与隔离。

（2）字线、位线：



如 4 位数据输出，4×4 位的 MOS 管，单译码结构。地址线 A1、A0， 译码后输出 4 条选择线 W3~W0，用于选中 4 个单元的某一个（ 每个单元 4 位输出） 。存储矩阵由 MOS 门组成， W3~W0 任何一根线上给出高电平信号时， d3~d0 会输出一个 4 位二值代码。将每个输出代码称为一个“字”，并将 W3~W0 称为字线（选择作用），将 d3~d0 称为位线（或数据线）。输出端的缓冲器用来提高带负载能力，并将输出的高、低电平变换为标准的逻辑电平。同时， 通过给定 EN#信号实现对输出的三态控制，将数据反相输出。

3.3 随机存取存储器 了解

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

位扩展：在存储器芯片字数不变的前提下，进行数据的位数扩展

字扩展：在存储器芯片的位数满足的前提下，进行字数扩展

复合扩展：在位长和字数均不足时，采用复合扩展方式。先位扩展再字扩展

3.5 高速缓冲 Cache 基本工作原理及作用（仅描述概念即可） 理解

（1）程序访问的局部性原理：两种局部性：时间局部性和空间局部性

时间局部性：最近访问的信息很可能再次被访问。

空间局部性：最近访问信息的邻近信息可能被访问。

（2）思想

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。

（3）实现

Cache 设置在 CPU 与主存储器之间，通常采用存取速度快并且无需刷新的 SRAM 来实现。

在主存和 CPU 之间设置 Cache 后，如果当前正在执行的程序和数据存放在 Cache 中，则当程序运行时不必再从主存储器读取指令和数据， 而只需访问 Cache 即可。

（4）组织方式

Cache 的基本单元称为行或字块(Line，Cache line)

32 位地址分为：Tag（比较 Tag） Index Blockoffset Byteoffset（一般两位）

数据字段：保存从主存单元复制过来的数据，单位是块。每个块的大小为 4~128 字节，典型的大小为 32 或 64 字节。

标志字段（tag）：保存数据字段在主存中的地址信息，又称为地址标记寄存器，记为 Tag。

有效位字段（valid）：标识区块和 Tag 是否有效。

一致性控制位字段：指示区块数据是否被 CPU 更新但并未写回至主存。

替换控制位字段：向替换算法指示区块状态。

主存是以字块为单位映像/复制到 Cache 中。

（5）Cache 管理

当 Cache 已经用满，但主存还需将新的字块调入 Cache 时，就会执行一次 Cache 字块的替换。

当程序对 Cache 字块执行写入时，需保证 Cache 字块和内存字块的一致性，通常的有两种写入方式：

写返回：先写 Cache 字块，待 Cache 字块被替换出去时再一次性写入内存字块

写穿透/写直达：在写 Cache 字块的同时也写入内存字块

（6）Cache 性能 – 命中率（Hit Rate）

任一时刻 CPU 能从 Cache 中获取数据的几率称为命中率（Hit Rate）

（7）Cache 的地址映射与转换

地址映射，由相联存储器(Associative Memory)的块表(Block Table)实现，（地址自动转换）

必须记住的映像关系

全相联映射（Fully associative mapping）：完全随意的对应 （没有 index）

直接相联映射(Direct mapping)：一对多的硬性对应

多路组相联映射(Multi-way set associative mapping):多对多有限随意对应

（8）Cache 更新与替换策略

贯穿读出（Look Through）

CPU 对主存的所有数据请求都首先送到 Cache，在其中查找。如果命中，则切断 CPU 对主存的请求，并将数据送出，不命中，则将数据请求传给主存。

优缺点：优点：降低了 CPU 对主存的请求次数；

缺点：延迟了 CPU 对主存的访问时间。

旁路读出（Look Aside）

CPU 同时向 Cache 和主存发出数据请求。由于 Cache 速度更快，如果命中，则 Cache 在将数据回送给 CPU 的同时，还来得及中断 CPU 对主存的请求，不命中，则 Cache 不做任何动作，由 CPU 直接访问主存。

优缺点：优点：没有时间延迟；

缺点：每次 CPU 都存在主存访问，从而占用一部分总线时间。

（9）Cache 写入更新策略

写通方式（Write Through）

任一从 CPU 发出的写信号送到 Cache 的同时，也写入主存，以保证主存的数据能同步地更新。

优缺点：优点：操作简单，可靠性高；

缺点：写速度是主存写速度，由于主存的慢速，降低了系统的写速度并占用了总线时间，没有发挥 Cache 高速访问优势。

写回方式（Write Back）

更新数据只写到 Cache，而主存中的数据不变。在 Cache 中设置“修改标志位”，供每次 Cache 的数据更新时判断，以写入主存相应的单元中

优缺点：优点：克服写通方式的弊病，减少了对主存的访问次数

缺点：有 Cache 与主存数据不一致的隐患，控制也较复杂

（10）Cache 替换策略

随机（Random）替换

优点：方法最简单，易硬件实现，速度快

缺点：被换出的数据可能马上就需要再次使用，增加了映射装入次数，降低命中率和效率

先进先出（FIFO）

根据进入 Cache 的先后次序来替换，先调入的 Cache 块被首先替换

优点：不需随时记录各个块的使用情况，易实现，且系统开销小

缺点：一些需要经常使用的程序块可能会被调入的新块替换

最不经常使用（Least Frequently Used，LFU）

将一段时间内被访问次数最少的块替换出去。每块设置一计数器，从 0 开始计数，每访问一次，被访块的计数器就增 1。需要替换时，将计数值最小的块换出，同时将所有块的计数器清零

优点：方法较简单，较易硬件实现

缺点：统计的是各块两次替换间的访问次数，不能严格反映近期被访问情况。新调入的块很容易被替换出去

近期最少使用（Least Recently Used，LRU）

将 CPU 近期最少使用的块作为被替换的块。需要随时记录 Cache 中各块的使用情况，以便确定哪个块是近期最少使用的块。为每个块设置一个“未访问次数计数器”，每次 Cache 命中时，命中块的计数器清 0，其它各块的计数器加 1。每当有新块调入时，将计数值最大的块替换出去。

优点：确保新加入的块保留，还可把频繁调用后不再需要的数据淘汰掉，提高 Cache 利用率和命中率。硬件实现并不困难

缺点：无

3.6 虚拟存储器 了解

虚拟存储器的两大特点：①允许用户程序使用比实际主存空间大得多的空间；②每次访存都要进行虚实地址转换

虚拟存储器提供了三个重要的能力：①高效使用主存，将主存看成一个磁盘地址空间的高速缓存，只保存活动区域，并根据需要在磁盘和主存之间来回传送数据；②为每个进程提供一致的地址空间，从而简化了存储器管理；③保护了每个进程的地址空间不被其他进程破坏

虚拟存储器解决了三个根本需求：①确保可以运行存储空间需求比实际主存容量大的应用程序②确保可执行程序被装载后占用的内存空间是连续的 ③确保同时加载多个程序的时候不会造成内存地址冲突。

虚拟地址和物理地址之间需要映射：虚存的地址变换，分三种：段式，页式，段页式

段式虚拟存储器：段式存储管理：把主存按段分配的存储管理方式主存-辅存间信息传送单位是不定长的段

页式虚拟存储器：页式存储管理，把主存按页分配的存储管理方式主存-辅存间信息传送单位是定长的页

段页式虚拟存储器：先将用户程序按程序的逻辑关系分为若干个段，并为每一个段赋予一个段名，再将每一个段划分成若干页，以页为单位调入调出。但可以按段实现共享和保护

段页式虚拟存储器充分利用段式和页式两种虚拟存储器在管理主存和辅存空间的优点，提高了主存利用率

第 4 章 总线和接口

4.1 总线技术

总线的五种分类方式，主要是 DB、AB、CB 理解

总线按位置分类

片内总线(In Chip Bus)：位于芯片(CPU 或其他的处理器)的内部，连接 CPU 内部的各个部件

芯间总线(Chip Bus)：也称为芯片总线(Component-Level Bus)或者局部总线(Local Bus)，连接 CPU 和外围芯片

内总线(Internal Bus)：又称为板级总线(Board-Level Bus)或系统总线(System Bus)用于系统内部各高速模块之间的互连；例如 ISA、EISA、PCI 等

外总线(External Bus)：又称 I/O 总线或通信总线(Communication Bus) ，用于计算机之间，或者计算机与外设之间的互连；例如 SCSI、USB 等

总线按功能分类

地址总线（AB，Address Bus）：一般为单向传送总线，信号通常从 CPU 发出，送往总线上所连接的各个模块或器件；地址信号线用来指定数据总线上数据的去向与来源；地址总线宽度决定了最大存储器空间寻址范围；另外，地址总线也可用于 I/O 端口的寻址。

数据总线（DB，Data Bus）：用于传送数据信息。通常为双向三态形式；数据总线的位数（宽度）是计算机系统的一个重要指标。通常与微处理器的字长相一致。

控制总线（CB，Control Bus）：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。

控制信号的双向特性体现：CPU 送往存储器和 I/O 接口电路的；如，读/写、片选、中断响应等信号；其它部件反馈给 CPU 的；如，中断请求、复位、总线请求、设备就绪等信号

涉及到不同的电路单元时，有些控制信号仅连接一个外设模块，有些同时连接多个模块有些控制信号线为单向传送，有些为双向传送；有些控制信号线甚至在不同时间段有不同的功能定义。

一般来说，总线信号线中，除电源线、地线、数据总线和地址总线外的所有信号线都归纳为控制总线。

总线按数据传输方式分类

串行传输：需传输的比特串一个接一个地在一条信道上传输

并行传输：比特以成组的方式在两条或更多的并列信道上进行传输

总线按时序控制方式分类

异步传输：双方有各自独立的定时时钟

同步传输：双方需要采用统一的时钟

总线按时分复用方式分类

非复用：每条信号线的功能恒定，缺点是总线上的信号线数量较多

复用：某些信号线在不同时段传输不同类型的信号，旨在减少信号线的数量；区分信号线上到底传输的是什么信号（通信协议的约定/增加专用信号线加以标识）

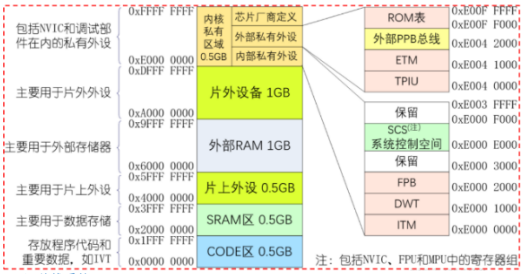
总线周期的四个阶段 掌握

（1）总线操作与总线周期

通过总线进行信息交换的过程，称为总线操作

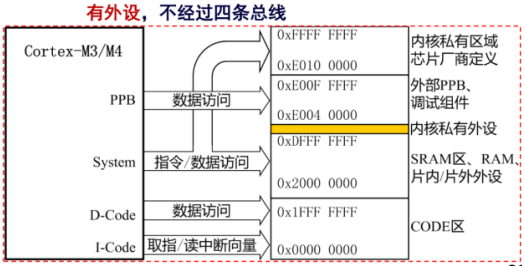
总线设备完成一次完整信息交换的时间，称为总线周期（或总线传输周期）
总线时序是指，总线操作过程中，总线上各信号之间在时间顺序上的配合关系
（2）总线周期的 4 个阶段
请求及仲裁（Request and Arbitration）阶段：主模块请求，仲裁机构决定把下一个总线传输周期分给哪一个请求源。
寻址（Addressing）阶段：取得总线使用权的主模块，通过总线发出本次要访问的从模块（存储器地址或 I/O 端口）地址及有关命令，通知参与传输的从模块开始启动。
数据传输（Data Transferring）阶段：主模块和从模块进行数据传输，数据由源模块发出，经数据总线到达目的模块。
结束阶段：主模块、从模块的有关信息均从总线上撤销，让出总线，以便下一个总线传输周期其他模块能够使用总线。
常见集中式仲裁、分布式仲裁方法的原理及不同方法的优缺点　　了解
总线的使用权分配即总线判优控制，也称为总线使用权仲裁。
当有多个主设备同时申请总线时，按一定的优先等级顺序，判定哪个主设备能优先使用总线。
（1）集中式：将控制逻辑（即总线仲裁器 arbitrator）集中在一处 ，分为串行、并行、串并行混合式
串行仲裁：串行仲裁又称为“菊花链” （Daisy Chainning）仲裁：越靠近控制器的模块，优先级越高；这种延迟与模块数量成正比，所以判优速度较 链形优先级存在传播延迟慢，一般只能接少量的（几个）模块；一个模块有故障就会造成整条“链”失效 链形结构可靠性低。结构简单，造价较低。
并行仲裁：又称为“独立请求式仲裁” 。每个主设备都有独立的 BR 和 BG 信号线，并分别接到仲裁器上。判优速度快，且与模块数无关。所需“请求线”和“允许线”较多，N 个模块需要 2N 条。
串并行混合式仲裁：
先并行。所有主设备的总线请求首先经过总线判决器（BR1 或 BR2），总线判决器决定是 BR1 所连主设备；还是 BR2 所连主设备获得总线控制权。再按串行方式。来决定 BR1 上的设备是主设备 2 还是 4、BR2 上的 设备是主设备 1 还是 3 应该获得总线控制权。
兼具串行和并行仲裁优点，既有较好的灵活性、可扩充性又可容纳较多的设备而结构也不会过于复杂，且有较快的响应速度。
（2）分布式：将控制逻辑分散在（与总线连接的）各个部件或设备上，由各个节点竞争使用权 自举分布式仲裁
使用多个请求线，不需要中心裁决器，每个设备独立地决定自己是否是最高优先级请求者
原理：分为申请期和裁决期。在申请期，需要请求总线控制权的设备在各自对应的总线请求线上送出请求信号。在裁决期，每个设备将有关请求线上的合成信号取回分析，以确定自己能否拥有总线控制权
总线时序 理解
总线时序是指，总线操作过程中，总线上各信号之间在时间顺序上的配合关系。
（1）分类
a. 同步总线时序
时标通常由 CPU 的总线控制部件发出，送到总线上的所有部件；也可以由每个部件各自的时序发生器发出，但是必须由总线控制部件发出的时钟信号对它们进行同步
优点：模块间的配合简单一致
缺点：主从模块之间的时间配合属强制性同步，必须按速度最慢的部件来设计公共时钟
b. 异步总线时序
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，给设计者带来更多的选择余地
异步总线中，系统没有公用的时钟，主从模块之间通信时，采用应答方式（又称握手方式）
根据问答信号之间的关系，分成不互锁方式，半互锁方式，全互锁方式。
不互锁方式：读：①主设备将拟访问的地址信号送上地址总线，同时发出主设备请求信号，并且延迟 D1 后发出固定宽度的读命令；
②从设备收到主设备请求之后，将所需数据读出并送往数据总线，同时发出从设备应答信号；
③主设备在读信号后沿，将从设备输出的数据读入主设备内部寄存器。
写：①主设备将拟写入的单元地址以及需要写入的数据驱动至相应的总线，同时发出主设备请求信号，并且延迟 D2 后发出固定宽度的写命令；
②从设备收到主设备请求后准备接收数据，并在完成准备工作之后发出应答信号；
③从设备利用主设备的写信号后沿，将总线上的数据锁入内部寄存器。
半互锁方式：①主设备发请求信号并等待从设备的应答；
②从设备收到请求后输出数据，然后作出应答；
③主设备收到从设备的应答后开始读数据，数据接收完毕才撤销请求信号。
半互锁方式与不互锁方式的主要区别在于：主设备发出请求信号之后，必须等待从设备应答后才启动读数据操作，只有收到数据后才撤销请求。
全互锁方式：①主设备发出读请求；
②从设备送出数据并发出应答；
③主设备收到从设备的应答后开始读数据，并在数据读取结束之后撤销读请求；
④从设备得知主设备撤销请求后，才停止驱动数据总线并撤销应答。
全互锁方式与不互锁方式和半互锁方式对比，全互锁方式的主从双方都在确认对方状态之后才开始下一步操作，各个动作之间环环相扣，传输可靠性最高。
c. 半同步总线时序
半同步总线是对同步总线的一种优化，对于大多数速度较快的传送对象，均按照同步方式定时。对于系统所连接的少数速度较慢的设备，增加一条 Ready/wait 状态信号线，当慢速设备被访问时，可以利用这条信号线请求主模块延长传送周期
d. 周期分裂式总线时序
分离式总线的思想：将一个传输周期（或总线周期）分解为两个子周期。在第一个子周期（寻址子周期）中，主模块 A 获得总线使用权后，将命令、地址、A 模块的编号等其它信息发到系统总线上，由相关的从模块 B 接收下来。然后 A 模块放弃总线，供其它模块使用。在
第二个子周期（数据传输子周期）中，B 模块根据所收到的命令，经过一系列的内部操作，将 A 模块所需的数据准备好，然后由 B 模块申请总线使用权，一旦获准，B 模块将 A 模块的编号和所需数据、B 模块的地址等信息送到总线上，供 A 模块接收。
将一个总线读周期或写周期分解为两个分离的子周期：寻址子周期、数据传送子周期也把上述寻址子周期称作地址阶段；把数据传送子周期称作数据阶段。
（3）关于同步和异步通信方式的几点讨论
传输速度：同步：如果从设备太慢，就无法满足时序要求；异步：应答过程的交互次数越多，速度越慢。
可靠性：最可靠的方式：异步全互锁，每步操作“环环相扣”。
4.2 片内总线 AMBA AHB 数据传输过程， AHB“流水线”分离操作 理解
2 个阶段：地址阶段、数据阶段
流水线机制：地址信息和数据信息交叠（overlapping）的操作方式，被称作流水线机制
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这两个操作构成 2 级流水线操作。从机因某种原因不能及时响应时，这个流水线就会被打断。
SPLIT 思想（即周期分裂式时序）
周期分裂式时序：地址阶段和数据阶段可以被分离。
从机不能及时响应时，发送控制信号 HSPLITx 通知仲裁器。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，通过控制信号 HSPLITx 发出重新启动传输的信号，仲裁器根据挂起操作主机的优先级决定何时再次分配总线使用权当主机获得总线使用权后，重新发送地址、控制等信息，继续刚才挂起的传输操作
插入等待周期的数据传输
半同步式时序
若从机在数据阶段的第一个时钟周期没能准备好，则需要把 HREADY 拉为低电平，从而插入了等待周期
AHB 的突发传输
突发传输就是，一次传输过程传输一个数据块而不是单个数据。突发的意思是 burst，可以理解成爆发，很多数据需要运输
总线允许信号与主机号的生成
由于采用集中式的译码器，每个主机在需要的时候随即驱动自身的地址信号，无须等待总线允许信号 HGRANTx。集中式的译码器根据各个主机的总线使用请求产生总线允许信号 HGRANTx，并在仲裁器的控制下生成主机号 HMASTER[3:0]，指示地址和控制多路选择器把主机的地址总线与对应从机的地址总线连接。
4.3 系统总线和外部总线 USB、 PCIe 了解
4.4 输入/输出接口 I/O 接口电路的典型结构 了解
（1）IO 接口的结构
a. 数据端口
b. 状态端口：准备就绪位(Ready)忙碌位(Busy)错误位(Error)
c. 命令/控制端口：常见的命令信息有启动、停止、允许中断
（2）I/O 接口的分类
按数据传输方式：串行接口；并行接口
按时序控制方式分类：同步接口；异步接口
按主机访问 I/O 设备的控制方式分类：程序查询接口；无条件查询；中断接口直接存储器访问（DMA）接口

数据流程方向：单工；半双工；全双工
（3）I/O 接口的功能
设置数据缓冲解决速度不匹配问题
设置电平转换电路解决电平不一致问题
设置信息转换逻辑满足各自格式要求
设置时序控制电路同步 CPU 和外设的工作
提供地址译码电路
提供 I/O 控制、读/写控制及中断控制等逻辑
（4）I/O 端口的缓冲器和锁存器
输入需要经过缓冲器：
缓冲器是一种三态元件，当缓冲器使能时，可以将外设的数据同步到数据总线上，而没有使能时则输出高阻态，从而不影响总线上的数据。外设数据的保持时间相对于 CPU 的处理时间而言长的多，在不需要从当前端口读取输入时，输入数据不能影响系统总线的正常使用，因此需要经过缓冲器。
输出需要经过锁存器：
锁存器可以在输入数据消失后继续保持之前的输出不变。CPU 速度很快，而物理外设的速度比较慢，因此需要锁存器保持电路输出端的数据
（5）I/O 端口的编址方法
I/O 端口和存储器统一编址，也称存储器映像的 I/ O（Memory Mapped I/O）方式
存储器映像：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。
I/O 端口和存储器分开单独编址，也称 I/O 映像的 I/O（I/O Mapped I/O）方式
I/O 映像：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
对比
存储器映像：优点：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。外设数目或 I/O 寄存器数目几乎不受指令限制 CPU 读/写控制逻辑较简单
缺点：占用了存储器的一部分地址空间增加了地址译码电路的复杂性
I/O 映像：优点：I/O 端口地址不占用存储器地址空间 I/O 端口地址译码较简单，寻址速度较快；使用专用 I/O 指令和存储器访问指令有明显区别，可使编程清晰。
缺点：专用 I/O 指令类型少，远不如存储器访问指令丰富；CPU 提供存储器读/写、I/O 端口读/写两组控制信号。
第 5 章 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
计算机体系结构 CA（Computer Architecture）
CA 是对计算机系统的设计思想、逻辑特征、原理特征、结构特征和功能特征的一种抽象。
包括：指令集体系结构 ISA（Instruction Set Architecture）和微架构 μarch（Microarchitecture）以及硬件实现（Hardware Implementation）
指令集体系结构 ISA
ISA，描述软件如何使用硬件的一种规范和约定。描述处理器指令及其功能、组织方式的规范称为指令集体系架构 ISA。
微架构 μarch 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
哈佛结构
哈佛结构属于一种并行体系结构。程序和数据存储在不同的存储空间中，每个存储器独立编址，使用两套总线单独访问。可提高数据吞吐率，消除流水线上取指和取操作操作的资源相关，处理器性能高于冯·诺依曼结构。哈佛结构较为复杂，实现难度较大，早期较少采用。
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
（1）存储器映射
所有基于 ARMv7M 的 Cortex-M 系列处理器，都采用相同的存储器映射关系方式，有助于提高设备之间的软件可移植性和代码可重用性。
分区
内核私有区域 0.5GB；片外设备 1GB；外部 RAM 区 1GB；片上外设区 0.5GB；SRAM 区 0.5GB；
CODE 区 0.5GB。

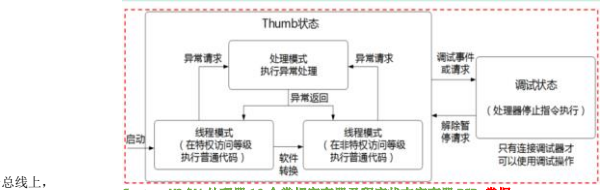


（2）总线系统
核心：基于 AHB 总线协议的内部总线互连矩阵
构成：I-Code 总线。
D-Code 总线。I-Code 总线与 D-Code 在物理上彼此独立，但两者之间有一个仲裁器，当 I-Code 和 D-Code 同时访问同一区域时，D-CODE 优先
System。基于 AHB-Lite 规范的 32 位系统总线，也被称为 AHB 总线
APB/PPB，32 位 APB 总线
CPU 通过内部总线互连矩阵直接访问内核私有外设，不经过四条总线

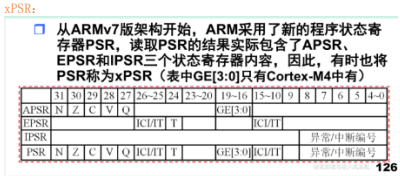
注意：CPU通过内部总线互连矩阵直接访问内核私有外设，不经过四条总线

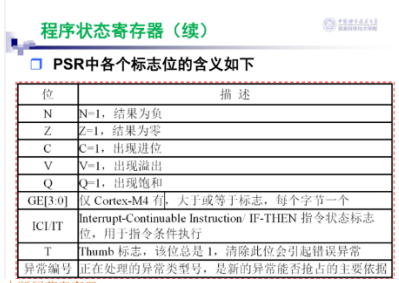


（3）其他总线
调试访问端口（DAP）基于“增强型 APB”总线规范的 32 位总线。主要用于连接处理器内部的调试访问接口 AP 与外部调试端口 DP，如 SWJ-DP 和 SW-DP。
5.3 Cortex-M3/M4 的编程模型 Cortex-M3/M4 处理器 2 种操作状态，2 种操作模式，2 种访问等级（切换原理） 理解
（1）操作状态（运行状态）
Thumb 状态：Thumb 状态，执行 Thumb 指令的状态。由于 Cortex-M 系列处理器不支持 ARM 指令集，所以没有 ARM 状态。（thumb 状态：arm 执行 16 位指令的状态，即 16 位状态，ARM 状态：arm 处理器工作于 32 位指令的状态，所有指令均为 32 位）
调试状态：调试状态，当处理器被暂停后就会进入调试状态 同时会停止指令执行。例如通过调试器或触发断点后。
（2）操作模式（运行模式）
处理模式：处理模式（Handler mode），类似于经典处理器中的异常模式。执行的是中断服务程序（ISR），此时处理器具有特权访问等级
线程模式：线程模式（Thread mode），除处理模式以外的模式，分为特权线程模式（类似于经典处理器 sys(system)模式）和非特权线程模式（类似于经典处理器 usr(user)模式）
（3） 特权与非特权的转换
特权线程模式切换到非特权线程模式：系统启动后处于特权线程模式，在此模式下，可以通过对特殊寄存器 CONTROL 的写操作，将处理器从特权线程模式切换到非特权线程模式。
非特权线程模式切换到特权线程模式：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。非特权线程模式可以利用这种机制，在处理模式中修改 CONTROL 寄存器，从异常返回时变成特权访问等级。



Cortex-M3/M4 处理器 16 个常规寄存器及程序状态寄存器 PSR 掌握
AAPCS（ARM Architecture Procedure CallStandard）规范：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：作为子程序调用中间寄存器。
（1）通用寄存器：R0~R12：
R0~R7：8 个低位寄存器，因受指令编码空间限制，许多 16 位 Thumb 指令只能访问低位寄存器。
R8~R12：5 个高位寄存器，可用于 32 位指令和少数几个 16 位指令（如 MOV 指令）。系统复位之后，R0~R12 的初始值均未定义。
（2）栈指针：R13
Cortex-M3/M4 处理器采用双堆栈设计，有两个物理上的栈指针，也就是有两个 R13 寄存器，一个是主栈指针 MSP，另一个是进程栈指针 PSP。对于一般程序而言，两个栈指针只有一个可见。
系统复位之后，PSP 的初值未定义；而 MSP 的初值存放在整个存储空间最开始（中断向量表）处第一个字中，系统初始化时，需要将其取出并对 MSP 进行赋值。
栈指针的选择是通过特殊寄存器 CONTROL 设定的 MSP 为默认栈指针，在系统复位后或处理器处于处理模式时，处理器使用 MSP；PSP 只能用于线程模式。
（3）链接寄存器： R14（又称 LR）
用于保存函数或子程序调用时的返回地址，在函数或子程序结束时，LR 中的数值用于调用返回。在异常处理时，LR 中将自动保存返回地址 EXC_RETURN，异常处理结束用于异常/中断返回。如果是嵌套调用，调用时需将 LR 中的数值压栈保存。
（4）程序计数器：R15（又称 PC）
①读 PC 时返回的是当前指令地址加 4，这是因为流水线的特性以及与 ARM7T 系列处理器兼容的需要
②对 PC 的写操作可以使用 MOVE 指令以及数据处理指令来实现，以实现程序的跳转
③需要注意的是，无论使用跳转指令还是直接写 PC 寄存器，写入值必须是奇数，确保其最低位是“1”，以表示其处于 Thumb 状态，否则将被认为试图转入 ARM 模式，从而导致出现错误异常
（5）特殊寄存器
xPSR：





位	描 述
N	N=1，结果为负
Z	Z=1，结果为零
C	C=1，出现进位
V	V=1，出现溢出
Q	Q=1，出现饱和
GE[3:0]	仅 Cortex-M4 有，大于或等于标志，每个字节一个
ICI/IT	Interrupt-Continuable Instruction/ IF-THEN 指令状态标志位，用于指令条件执行
T	Thumb 标志，该位总是 1，清除此位会引起错误异常
异常编号	正在处理的异常类型号，是新的异常能否抢占的主要依据

中断屏蔽寄存器：
PRIMASK
当最低位被置位（写入 1）后，将屏蔽除复位、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，类似于 x86 系统中的关中断（IF=0），以便处理紧急事务
当紧急事务处理完毕，要及时将 PRIMASK 的最低位进行复位，类似于 x86 系统的“开中断”
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”
常用于执行负责错误处理（包括硬件错误）的中断服务程序
与 PRIMASK 不同的是，FAULTMASK 无需主动清理
BASEPRI
BASEPRI 寄存器的最低 8 位采用了“可伸缩”设计，具体宽度取决于芯片制造商实际设计的中断优先级数量
写数不为零时，屏蔽优先级数值大于等于值的所有中断
屏蔽大于等于 0 的中断时，需要使用 PRIMASK 写数为 0 时，不屏蔽
中断屏蔽寄存器属于特殊寄存器，只有特权访问等级才可以进行读写访问
(6)CONTROL 寄存器
用于选择线程模式的特权访问等级以及栈指针
nPRIV：设置线程模式的特权访问等级
该位为 0/1，处理器进入特权线程模式/非特权线程模式
SPSEL：选择线程模式中的堆栈指针
当该位为 0 时，线程模式使用主栈指针 MSP；当该位为 1 时，线程模式使用进程栈指针 PSP；在处理模式下，该位始终为 0，并且忽略对其的写操作。
FPCA：配置 FPU 的 Cortex-M4 才有此位
当发生异常时，若该位为 1，浮点单元中的寄存器内容被压栈保存。执行浮点指令时 FPCA 位自动置位，在异常处理程序入口处该位被硬件自动清除。
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 理解
堆栈是一种特殊的数据结构，一种只能在一端进行插入和删除操作的线型表。
堆栈的数据存取操作按照后进先出(LIFO)的原则，并通过堆栈指针指示当前的操作位置。
（1）堆栈的作用
①保护断点，以便在异常/中断返回后，处理器能够从断点处继续运行。保护断点即，在异常/中断响应时，保存被中断程序的下一条指令的地址，以及处理器状态寄存器的内容。
②保存现场，以便当异常返回或者函数或子程序处理结束时，可以恢复现场。异常/中断服务程序，或者正在执行的函数或者子程序，如果需要使用某些寄存器，可以使用堆栈保存这些原来的内容。
③实现主程序与函数或者子程序之间的参数传递。函数或者过程调用有多种参数传递方式，堆栈传递是一种最安全的方式，并且对参数数量几乎没有限制。
④用于存储局部变量
（2）堆栈类型
①按照堆栈区在存储器中的地址增长方向
递增栈（Ascending Stack）：向堆栈写入数据时，堆栈区由低地址向高地址生长。
递减栈（Descending Stack）：向堆栈写入数据时，堆栈区由高地址向低地址生长。
②按照堆栈指针 SP 所指示的位置
满堆栈（Full Stack）：堆栈指针 SP 始终指向栈顶元素，也就是指向堆栈最后一个已使用的地址。
空堆栈（Empty Stack）：SP 始终指向下一个将要放入元素的位置，也就是指向堆栈的第一个没有使用的地址或者空位置。
（3）4 种基本堆栈类型
满递增（FA）： SP 指向最后压入的数据，且由低地址向高地址生长
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
空递增（EA）： SP 指向下一个可用空位置，且由低地址向高地址生长
空递减（ED）： SP 指向下一个可用空位置，且由高地址向低地址生长
Cortex-M 系列处理器只能使用满递减（FD）类型
（4）双堆栈
Cortex-M3/M4 的双堆栈设计有 MSP 和 PSP 两个堆栈指针，分别服务于不同的操作模式和特权访问等级。（MSP 主堆栈指针，PSP 进程堆栈指针）
CONTROL 寄存器中 nPRIV 和 SPSEL 的不同组合，两个堆栈共有 4 种场景，其中前三种比较常见。

nPRIV	SPSEL	应用场景
0	0	无操作系统的简单应用，特权访问等级+主堆栈（MSP）
0	1	有操作系统的应用。当前执行的任务是具有特权访问等级的线程模式，选择使用进程栈。主栈用于操作系统内核以及处理模式。
1	1	有操作系统的应用。当前执行的任务是非特权线程模式，只能使用进程栈。主栈用于操作系统内核以及处理模式。
1	0	这种情形只出现在处理模式，使用主栈和MSP，但是只有非特权访问等级。线程模式不会出现这种情况。

如果使用双堆栈，应通过 MPU 在 SRAM 中建两个区域。一个定义为特权级，其中一部分用作主栈存储区；另一个定义为非特权级，其中一部分用于进程栈。
注意：Cortex-M3/M4 的堆栈是满递减类型，因此两个栈指针的初始值应该是两个区域的最大地址。如果采用双字对齐，栈顶应位于双字边界上，MSP 和 PSP 最低 3 位为 000

5.4 Cortex-M 处理器存储系统 位段（位带）操作 理解

位段（也称位带）操作：一次存储器操作只访问一个位
Cortex-M3/M4 在 SRAM 区和片上外设区，各有一个位段区和位段别名区
SRAM 区域的最低 1MB（0x2000 0000 ~ 0x200F FFFF）
外设区域的最低 1MB（0x4000 0000 ~ 0x400F FFFF）
位段区域中特定存储单元的某一位，映射为一个位段的别名地址（一个字）
一个字的 32 个比特被映射到 32 个位段别名地址（ 32 个字）
位段操作的其他优点：操作的原子性：操作过程不会被其它事务打断；简化转移决断过程
C 程序实现位段操作：C 编译器本身不支持位段操作但是可以编程实现位段操作
举例：
0x4000 0000 第 0 位->0x4200 0000
0x4000 0000 第 3 位->0x4200 000C
0x4000 1000 第 3 位->0x4202 000C
0x20 就是十进制的 32，意思是扩展 32 倍大小 0x1000 * 0x20 + 2x4200 0000 + 0x0C

5.5 Cortex-M 处理器的异常处理 异常处理的基本过程，及异常优先级及优先级分组（概念） 了解

（1）异常类型（Exception types）

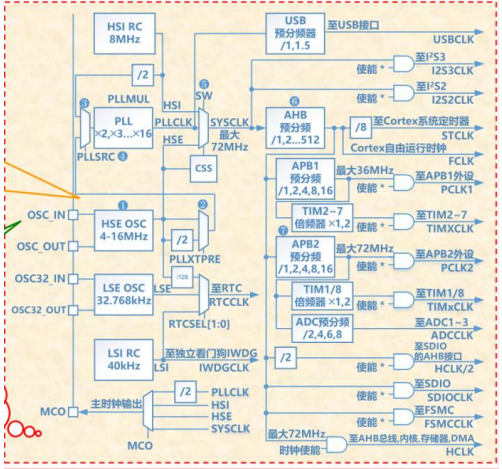
编号	类型	优先级	简介
0	N/A	N/A	没有异常在运行
1	复位	-3（最高）	复位
2	NMI	-2	来自 NMI 引脚，一般由看门狗或者掉电监测单元 BOD 产生
3	HardFault 硬件错误	-1	如果相应的异常处理未使能，所有错误都可能引发此异常
4	MemManage 错误	可编程	访问内存的行为违法了 MPU 定义的规则
5	总线错误	可编程	AHB 收到从总线的错误响应，如指令预取和数据读写被终止
6	用法错误	可编程	无效指令或试图访问未配置的部件，如 M3/M4 没有的协处理器
7~10	保留		
11	SVC	可编程	有 OS 时，应用程序可藉此调用系统服务（类似于 DOS 调用）
12	调试监视	可编程	使用基于软件的调试时，断点和数据观察点等调试事件的异常
13	保留		
14	PendSV	可编程	可挂起（缓期执行）的 SVC，常用于多任务 OS 的上下文切换
15	SYSTICK	可编程	系统节拍定时器产生的周期性异常，例如任务之间的切换定时
16	IRQ#0	可编程	由片上外设或者外设中断源产生
17	IRQ#1	可编程	
...	...	可编程	
255	IRQ#239	可编程	

此页PPT在 5-2-5 出现过

（2）异常状态（Exception states）
非激活状态 Inactive：异常既不在激活状态也不在挂起状态
挂起状态 Pending：异常源发出了服务请求，正在等待处理
激活状态 Active：正在接受处理器服务但未结束的异常（如果某异常处理程序被更高优先级的异常服务打断，则两个异常均处于激活状态）
激活并挂起状态：Active and pending：异常正在接受处理器服务，而相同异常源又产生
（3）异常处理程序（Exception handlers）
（4）异常向量表（Vector table）255 个
（5）异常的优先级（Exception priorities）
（6）中断优先级分组（Interrupt priority grouping）
Cortex-M3/M4 中，每一个中断都有一个 8 位的中断
优先级的减少通过去除优先级配置寄存器的最低位（LSB）实现
优先级寄存器 3~8 位又分为两部分（以 8 位为例）：分组优先级（group priority），又称为抢占优先级（preempt priority），子优先级（subpriority）
（7）异常流程（Exception entry and return）
a.异常请求的接受
处理器接受请求的条件：处理器处于运行状态；异常处于使能状态；异常的优先级高于当前等级异常没有被屏蔽（如没有设置 PRIMASK）。
注意，如果异常处理程序中出现了 SVC 指令，而该异常的优先级不低于 SVC 的优先级，就会触发硬件错误，从而进入硬件错误的处理程序。
b.异常进入流程
异常进入流程包括如下操作：
①多个寄存器的值和返回地址被压入当前使用的栈。若处理器处于线程模式且正在使用进程栈指针 PSP，则 PSP 指向的堆栈区域就会用于该压栈过程，否则就会使用主栈指针 MSP 指向的堆栈区域。
②从向量表中取出异常向量
③取出异常处理程序中的指令
④更新多个 NVIC 寄存器（后续介绍）和内核寄存器（PSR、LR、PC 及 SP）
加快中断执行速度—压栈顺序
R0~R3、R12，LR、PC（返回地址）和 PSR 共 8 个寄存器被压栈（注意：压栈顺序和栈帧结构不同）。如果需要压栈保存 FPU 状态，则共有 26 字，如果使能双字栈对齐，可能还会修改已入栈的 PSR[9]。
压栈时为了尽快更新 PC，首先压栈的是 PC（返回地址）和 PSR，出栈时为了尽快恢复处理器状态和返回主程序，出栈时也应该先出栈 PSR 和 PC，咋办？
前述栈指针“…会在异常处理开始前自动调整”
c.执行异常处理程序
进入异常处理程序内部后，处理器进入处理模式，并运行于特权访问等级，栈操作使用 MSP。此过程中如果有更高优先级的异常产生，处理器会接受新的中断，当前正在执行的处理被更高优先级的处理抢占而进入挂起状态，此即异常嵌套。若执行过程中产生的其他异常具有相同或更低的优先级，新产生的异常就会进入挂起状态，待当前异常处理完成后才可能被处理。程序代码执行的返回指令会引起 EXC_RETURN 数值被加载到程序计数器中（PC），并触发异常返回机制。
d.异常返回
EXC_RETURN 写入 PC 时，就会触发异常返回流程。异常返回机制被触发后，进入异常期间被压入栈中的寄存器数值会被恢复到寄存器组中，因而多个 NVIC 寄存器和处理器内核中的寄存器（如 PSR、SP 和 CONTROL）都会被更新。
e.EXC_RETURN
返回至处理模式，使用 MSP 恢复非浮点状态信息：0xFFFFFFF1
返回至线程模式，使用 MSP 恢复非浮点状态信息：0xFFFFFFF9
返回至线程模式，使用 PSP 恢复非浮点状态信息：0xFFFFFFFD
返回至处理模式，使用 MSP 恢复浮点状态信息：0xFFFFFFE1
返回至线程模式，使用 MSP 恢复浮点状态信息：0xFFFFFFE9
返回至线程模式，使用 PSP 恢复浮点状态信息：0xFFFFFFED

第 6 章 ARM 指令系统 能看懂给出的指令语法及功能说明 了解

第 8 章 基于 ARM 微处理器硬件与软件系统设计开发
8.1 嵌入式系统设计与开发综述 嵌入式系统的交叉开发环境 了解
8.2 嵌入式系统开发过程 嵌入式系统开发过程各阶段 理解
8.4 ARM 微处理器最小硬件系统 微处理器最小硬件系统概念 了解
STM32 时钟树的基本概念、功能、作用、意义、特点等 理解



（1）概念
输入 外部晶振 HSE，可选为 2~16MHZ
②—第一个分频器 PLLXTPRE 可选 1 分频/2 分频
③—时钟源选择，开关 PLLSRC 可选其输出为：外部高速时钟 HSE 或内部高速时钟 HSI
④—锁相环 PLL：具有倍频功能(2~16)。经过 PLL 的时钟称为 PLLCLK；若设 9 倍频，即从 8MHz 的 HSE 变为 72MHz
⑤—开关 SW。经过 SW 后即系统时钟 SYSCLK。SW 可选 SYSCLK 时钟源为以下之一：HIS，PLLCLK，HSE。
⑥—AHB 预分频器 分频系数为 1/2/4/8/16/64/128/256/512
⑦—APB2 预分频器 分频系数为 1/2/4/8/16。若为 1，则高速外设 APB2(PCLK2)为 72MHz(AHB
输出为 72MHz 时)

时钟系统是由振荡器（信号源）、定时唤醒器、分频器等组成的电路。常用的信号源有晶体振荡器和 RC 振荡器。时钟是嵌入式系统的脉搏，处理器内核在时钟驱动下完成指令执行，状态变换等动作。外设部件在时钟的驱动下完成各种工作，比如串口的数据发送，A/D 转化，定时器计数等。因此时钟对于计算机系统是至关重要的，通常时钟系统出现问题也是致命的，比如振荡器不起振，振荡不稳，停振等。
（2）时钟树
时钟树从左至右，相关时钟依次可分为 3 种：输入时钟、系统时钟和由系统时钟分频所得其他时钟。
a.输入时钟
从时钟频率分：高速时钟和低速时钟
从芯片角度分：内部时钟(片内时钟)和外部时钟源(片外时钟)
因此，结合频率及内外：
高速外部时钟 HSE：①→②→③→④→⑤得 SYSCLK 。
高速内部时钟 HIS：片内 RC 振荡器产生；不稳定；上电开始作为初始系统时钟；8MHz。
低速外部时钟 LSE：外部晶振，提供给实时时钟：32kHz。
低速内部时钟 LSI：片内 RC 振荡器产生；提供给实时时钟和看门狗：40kHz
锁相环倍频输出 PLL：输入源可选 HSI/2、HSE 或 HSE/2。倍频 2~16 倍；输出最大 72MHz
b.系统时钟 SYSCLK
由 SW 根据用户设置，选择以下 3 个中的一路输出而得 PLLCLK；HSE；HSI
专门提供引脚 MCO(主时钟输出)；以实时检测时钟系统是否运行正常。通过软件编程，选择以下 4 个中的一路在 MCO 上输出 SYSCLK；PLLCLK；HSE；HSI。
c.由系统时钟分频所得其他时钟：即 SYSCLK 经过 AHB 预分频器输出
HCLK：AHB 时钟(通常，预分频系数为 1，常为 72MHz)。最高 72MHz；为内核、存储器和 DMA 时钟信号
FCLK：内核“自由运行”时钟；与 HCLK 互相同步；最高 72MHz；HCLK 停止时仍能继续运行，保证内核睡眠时也能采样到中断和跟踪休眠事件
PCLK1：外设时钟；再经 APB1 预分频器(系数常为 2)后得到；最高 36MHz(常为 36MHz)。为 APB1 总线上(低速)外设时钟(如需使用外设，须先开启其时钟)
PCLK2：类似 PCLK1。外设时钟；最高 72MHz(常为 72MHz)；为 APB2 总线上(高速)外设时钟
SDIOCLK：SDIO 外设时钟
FSMCCLK：可变静态存储控制器时钟
STCLK：系统时间定时器 SYSTICK 的外部时钟源；AHB 输出再经过 8 分频后得到；等于 HCLK/8
TIMXCLK：定时器 2~7 内部时钟源：PCLK1 经过倍频(*1 或*2，由 APB1 分频系数是否为 1
判断得出)所得
TIMxCLK：定时器 1/8 内部时钟源：PCLK2 经过倍频(*1 或*2，由 APB2 分频系数是否为 1
判断得出)所得
ADCCLK：ADC1~ADC3 时钟；PCLK2 经过 ADC 预分频器(/2,4,6,8)所得

8.6 ARM 中的 GPIO 给定库函数时 GPIO 的基本输入输出编程；引脚复用功能 掌握
8.7 定时器（基本和通用）的 3 种计数模式，普通输入捕获、PWM 输入捕获、比较输出、PWM 输出的基本原理 掌握
给定库函数时定时器的基本功能编程，包括硬件连线、相关 GPIO 口及定时器的初始化配置、精确延时的实现、结合中断的综合应用 掌握
8.8 中断控制器 NVIC 的基本概念及特性，中断优先级、向量表、服务函数、设置过程等几个重要概念 掌握
给定库函数时 EXTI 及 NVIC 的基本功能编程，包括硬件连线、软件配置（初始化）、简单 ISR 的编写 掌握
8.9 USART 给定库函数时 USART 简单数据收发功能编程，包括硬件连线、相关部件初始化配置、数据收发操作 掌握
8.10 SPI 与 I2C SPI、 I2C 接口原理（大致传输过程） 了解
（1）SPI（串行外设接口(Serial Peripheral Interface)）
需要至少 4 根线，事实上 3 根也可以（单向传输时）
MISO（Master Input Slave Output），主设备数据输入，从设备数据输出
MOSI（Master Output Slave Input），主设备数据输出，从设备数据输入
SCLK（Serial Clock），时钟信号，由主设备产生
CS（Chip Select），从设备使能信号，由主设备控制
SPI 上可以挂载一个主设备和多个从设备。任何时刻，一个主设备只与一个从设备进行通信，通信的从设备 CS 为低电平（有效）。在不使用 CS 信号时，SPI 上只能有一个主设备与一个从设备。SPI 的缺点是没有应答机制，传输过程全都由主设备进行控制，数据传输成功与否没法直接验证。
（2）I2C（集成电路总线（Inter-Integrated Circuit））
一条串行数据线 SDA，一条串行时钟线 SCL
特点：同步方式、主从协议、半双工
I2C 总线对发生在 SDA 信号线上的总线竞争进行仲裁，原理为：
在检测到总线空闲(SCL 和 SDA 均为高电平)后，拟使用总线的主机向 SDA 信号线发送数据(高电平或低电平)，每一位数据发送后随即检测 SDA 信号线电平是否与自身发送电平一致，若电平不符则竞争失败，自动关闭其输出。
I2C 总线标准对同类型的器件规定了一个固有的地址和可编程的地址，采用软件寻址方式，实现对每一个器件的访问。总线上每一个器件的地址必须是唯一的。

快速辨析（选择填空）
计算机发展历程：电子管阶段；晶体管时代；集成电路时代；LSI & VLSI 时代；ULSI & GSI 时代
摩尔定律： 晶体管的大小将以指数速率变小，在价格基本不变时，芯片上集成的晶体管数目每年将增加一倍
计算机软件：按特定顺序组织的计算机数据和指令的集合
分级存储体系结构：外存（主要为磁介质的机械硬盘、Flash 固态硬盘 SSD）
主存储器（内存）：主要由 DRAM、ROM 组成
高速缓存 Cache：SRAM
寄存器：触发器
微控制器的实现方式：
微操作：每条指令的执行过程都可以分解为一系列的微操作
两种实现方式：微程序控制器 CISC：硬连线控制器 RISC
计算机各部件按功能划分为两大阵营：CU 和 EU；
CU 就是控制器，负责指令译码，生成相应控制信号
EU 负责指令执行，如生成地址、读取和传送数据、计算和处理数据、存储结果、更新 PSR 和 PC
指令
微指令：微程序级的命令
机器指令：简称指令。CPU 能识别和直接执行的一条二进制编码序列；包括操作码和操作数两部分。
宏指令：由若干条机器指令组成的软件指令
存储器类型
（1）半导体存储器
只读存储器 ROM：掩模 ROM；可编程 ROM(PROM)；
可擦除可编程 ROM(EPROM)：UVEPROM；EEPROM；Flash Memory（包括 NAND 和 NOR）
NAND：NAND-flash 存储器具有容量较大，改写速度快等优点，适用于大量数据的存储
NOR：NOR 的特点是芯片内执行（XIP，eXecute In Place），这样应用程序可以直接在 flash 闪存内运行，不必再把代码读到系统 RAM 中。NOR 的传输效率很高，在 1~4MB 的小容量时具有很高的成本效益，但是其很低的写入和擦除速度大大影响了它的性能。
随机存储器 RAM：静态存储器 SRAM；动态存储器 DRAM
（2）磁介质存储器：磁盘（硬盘、软盘）；磁带；利用磁性介质的磁极化来存储信息
（3）光存储器：只读型光盘；可记录型光盘；反射光强度代表 0 和 1
存储器性能指标：
最重要的指标是存储器的容量和存取速度：存储器中存储单元的总数，常称为该存储器的存储容量
存取时间（访问时间）TA：存取时间称存储器访问时间，是指从启动一次存储器操作到完成该操作所经历的时间
存取周期 TM：存取周期是指连续启动两次独立的存储器操作（如连续两次读操作）所需间隔的最小时间
数据传送速率（频宽）BM：数据传送速率，指单位时间内能够传送的信息量
体积与功耗：便携式微机，其便携性能和续航时间尤为重要，因而对体积、功耗非常敏感
可靠性：采用平均故障间隔时间 MTBF 衡量，即两次故障之间的平均时间间隔
DRAM 和 SRAM
SRAM：用双稳态触发器（锁存器）存储信息；速度快（双极型<5ns， MOS 型几十~几百 ns ），不需刷新，外围电路比较简单，但集成度低（存储容量小，约 1Mbit/片），功耗大。SRAM 被广泛地用作高速缓冲存储器 Cache
DRAM：DRAM 是靠 MOS 电路中栅极电容存储信息，电容上的电荷会逐渐泄漏。需要定时充电以维持存储内容不丢失（称为动态刷新）；集成度高（存储容量大，可达 1Gbit/片以上），功耗低；但速度慢，约为 SRAM 的一半，且需要刷新。
地址空间：计算机中地址总线 AB 的宽度决定了存储器空间的最大寻址范围，常把这个寻址范围称为地址空间
为什么采用 Cache？
存储器的访问速度低是制约计算机系统性能的关键因素
解决方法：
1-在存储器访问时，通过指令等待，以牺牲 CPU 速度性能为代价
2-采用速度更高的静态存储器 SRAM 成本过高（对比 DRAM）
3-采用高速缓冲存储器 Cache 在 CPU 与 DRAM 之间建立一个（以 SRAM）构成的缓冲存储器
总线的两个基本特性
共享：当多个部件连接在同一组总线上，各部件之间相互交换的信息都可以通过这组总线传送
分时：是指任意时刻只能有一个设备向总线发送信息
总线主要性能指标
总线频率：总线宽度：总线带宽 带宽（MB/s）=总线宽度/8×总线频率；
同步方式：同步总线 or 异步总线
总线复用：信号线数
总线控制方式：并发工作、自动配置、仲裁方式、逻辑方式、计数方式
寻址能力：指地址总线的位数及所能直接寻址的存储器空间大小
总线的定时协议：为使源与目的同步，需要有信息传送的时间协议 分为同步总线定时、异步总线定时、半同步总线定时
负载能力：指总线上最多能连接的器件数
饱和和溢出
饱和：当数据超过所能表示的最大数据范围时，将其置为所能表示的最大（或最小）允许值。可以减小数据的畸变
溢出：当数据超过所能表示的最大数据范围时，将超出范围的高位数据丢弃。会产生较大畸变。
三种复位方式
系统复位：电源复位；备份区域复位
引脚功能复用：
是指将片内的不同功能资源分配到同一引脚，通过编程分别将不同的功能引出。由于实际应用中很少会用到器件全部资源，通过引脚复用可以大大减少引脚数量，从而节省成本、降低装焊难度。
MPU 和 MMU
MPU：MPU 负责将内存空间进行分区域的访问权限管理，适合要求对处理器时间有明确要求的实时系统
MMU：MMU 除了分区域访问权限管理以外，主要还提供了内存的分页管理和虚拟地址到物理地址的转换，适合多用户系统。
ISA 和 uarch
ISA：是指令集体系架构，是描述处理器指令及其功能、组织方式的规范，包括指令系统和寄存器组模型两部分，是介于硬件和软件之间的中间抽象层。
微架构：是 ISA 的具体硬件实现方式，即数字电路以何种方式实现处理器的各种功能。
转移相关的概念
转移目标指令：是控制转移指令的目标指令。当满足转移指令的条件时，程序将跳转到转移目标指令处执行。
转移代价：当程序发生转移时，需要排空流水线，造成流水线断流。这引起的流水线周期延迟称为转移代价。
转移延迟槽：是转移指令后面的一个时间片，无论是否发生转移，其中的指令总是会被执行。
（具体什么指令被放入转移延迟槽取决于编译器，编译器会找到一个无论是否转移都会执行的指令放进去。如果找不到，就需要放一个空指令，此时就会产生转移代价）
转移目标缓冲器（BTB）：它收集和存储近期所有转移目标指令的地址、转移可能性权值和转移目标指令的地址，并按照查找表的形式组织，为动态转移预测提供信息。
嵌入式操作系统的特点：实时性可靠性可裁剪
嵌入式系统的特点：嵌入性；专用性；计算机系统
判断补码是否溢出：若记符号位向前进位为 CP，次高位向前进位为 CF，当且仅当 CP 异或 CF =1 时，结果发生溢出
中断的状态：请求—挂起—激活：
出现中断请求之后，如果没有得到服务，就一直被挂起。即使中断源因某种原因撤销了请求，仍然会被处理。
解决方法：在编写 ISR 时，应先读取中断源相关状态，若的确需要服务，继续执行 ISR；否则退出
向量表重定位机制
原因：向量表默认位置位于 CODE 区最开始处，MCU 制造商在此区域一般配置的是存放启动代码的 Flash 或者是 ROM 型存储器。在有些 MCU 中，包含 Bootloader 的 ROM 就位于 CODE 区的最开始位置，而且没有使用存储器重定位特性或者存储器别名。两种情况都需要迁移中断向量表
实现：在 Cortex-M3/M4 处理器所集成的 NVIC 中，有一个名为 VTOR（Vector Table Offset Register，地址为 0xOE000 ED08）的寄存器，修改 VTOR 的值就能实现中断向量表的重定位起始地址必须能够被大于等于（中断向量数×4）的最小 2 的整数次幂整除假设一共 n 个中断，设 m=4n，设 k 是大于 m 的最小二的幂，那么起始地址是 k 的倍数。
中断、异常、外部中断及 EXTI
中断指，系统停止当前正在运行的程序转到其他服务
异常指，所有能打断正常执行的事件，但常指由于 CPU 本身故障、程序故障或请求服务
等引起的错误，异常包含中断(即中断是异常的子集)，异常与中断都有硬件支持
STM32F103 异常系统：16 个系统异常(也称,内核中断/异常，编号 0~15，优先级为-3 到 6)
和 60 个“外部中断”(M3 内核的定义，编号 16 以上，此时，不是指 EXTI 中断，而是所有中断)
外部中断 EXTI 概念要小得多，特指，EXTI 中断。NVIC 支持 19 个 EXTI 中断/事件请求(即 19 条外部中断线
中断处理
中断的整个过程分为 3 个阶段：中断响应、中断处理和中断返回
中断响应：CPU 确定要响应某中断源后，根据中断类型码去查找中断向量表中对应的表项，获得中断服务子程序的入口地址，接下来保护现场和断点（将标志寄存器和断点地址等信息压入堆栈），随后中断向量被装入 PC 寄存器，在下一个指令周期即进入中断服务子程序。中断处理：就是执行中断服务子程序的过程
中断返回：该过程由 CPU 内部硬件电路自动完成 包括断点恢复和现场恢复(从堆栈中恢复)
其他问题：中断优先级：中断嵌套：中断屏蔽。
AHB

AHB 系统构成：主机：从机：仲裁器：译码器。
AHB 数据传输：流水线：突发传输：流水线分离
微嵌易错题：

3.20 某计算机按字节编址，其主存容量为 1MB，Cache 容量为 16KB，Cache 和主存之间交换的块大小为 64B，采用直接相联映射方式。

（1）Cache 共有多少个字块（Cache line）？

（2）主存地址为 02021H 的单元装入 Cache 后对应的 Cache 地址是？

（3）主存地址为 02021H 的单元装入 Cache 后存放在 Cache 中的第几字块中（Cache 起始字块为第 0 字块）？

（1）$\frac{16\times2^{10}}{64}=2^8=256$

（2）主存页号：$\frac{1\times2^{20}}{16\times2^{10}}=2^6$，即页号 T 占 6 位；块号占 8 位；块内地址占 6 位

02021H=0000 0010 0000 0010 0001b

对应 Cache 内地址位 10 0000 0010 0001b=2021H

3) 1000 0000b=128

3.21 某计算机按字节编址，其主存容量为 1MB，Cache 容量为 16KB，Cache 和主存之间交换的块大小为 64B，采用 8 路组相联映射方式。

（1）主存地址中页号 s、页内块号 u、块内地址 W 各占多少位？

（2）主存地址为 02021H 的单元装入 Cache 后存放在 Cache 中的第几组（起始组为第 0 组）？

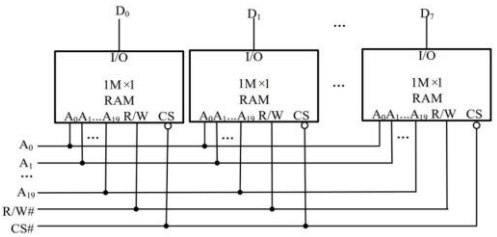
（3）Cacheline 对应的 Tag 字段占用多少位？

（1）$\frac{16\times2^{10}}{64\times8}=2^5$，即 32 个组，主存每页有 32 块，W 占 6 位，页内块号 u 占 5 位，页号 s 占 20-6-5=9 位

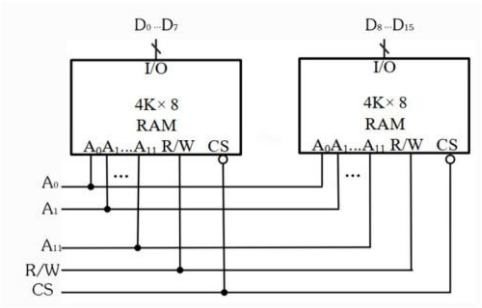
（2）02021H=0000 0010 0000 0010 0001b，页内块号为 0 0000，即组号为 0，

（3）9 位

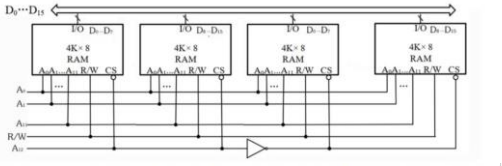
3.23 试用 1M×1 位的芯片构成 1M×8 位的存储器。



3.24 试用 4K×8 位的芯片构成 4K×16 位的存储器。



3.27 设计一个用 4 片 4K×8 位的芯片构成 8K×16 位的存储器。



5.17 在 Cortex-M3/M4 中，寄存器 R0~R12 有何异同？如果这些寄存器都是空闲的，你觉得首先使用哪些？为什么？

R0~R7 低位寄存器(许多 16 位的 thumb 指令只能访问低位寄存器)

R8~R12 高位寄存器(可用于 32 位指 令和少数几个 16 位指令)

同：都是通用寄存器。

异: 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 作为子程序调用中间寄存器。R0~R7 是 8 个低位寄存器，R8~R12 是 5 个高位寄存器。

优先低位寄存器，低位使用较频繁，优先低位可以降低功耗。可保证大多数指令可访问且速度快。

5.22 某基于 Cortex-M4 的 SOC 芯片共有 64 级外部中断，BASEPRI 寄存器的宽度共有几位？如果想屏蔽所有优先级大于 16 的中断，请写出对 BASEPRI 寄存器进行设置的汇编指令。如果想屏蔽所有优先级大于 0 的中断，又该如何设置？

BASEPRI:7:2 共 6 位

```
MOV R0, #0b010001 //16
MOV BASEPRI, R0

MOV R0,#0b000001 //0
MOV PRIMASK,R0
```