微机原理与接口技术

第十六讲 总复习

考核方式

期末考试（60%）+实验（20%）+
出勤、课堂测验及作业（平时）（20%）
期末考试：开卷
题型：填空、简答和编程
时间：
2022-12-26　　13:20-15:20
考试会议室：另行通知

第1章 微型计算机概述

第一代
4位和低档8位机
Intel 4004/8008

（1971-1973）主要应用于各种袖珍计算器、家电、交通灯控制等简单控制领域

第二代
中高档8位机
8080/8085、Z80

（1974-1978）广泛用于数据处理、工业控制智能仪器仪表及家电等各个领域

第三代
16位机
Intel 8086、80286

（1978-1981）在家用游戏机和早期的个人计算机领域得到广泛应用

第四代
32位机
80386、80486

（1984-1999）这段时期的微处理器得到了飞速的发展

第五代
64位机和双核
Intel Intanium、Pentium D、Pentium EE

（2000年以后）

特点：

1、速度越来越快。

2、容量越来越大。

3、功能越来越强。

微型计算机指令系统：

按照指令的执行方式和指令集的复杂程度来划分可以分为两类：

(1)复杂指令集（Complex Instruction Set Computer , CISC）

(2) 精简指令集（Reduce Instruction Set Computer , RISC）

(1)复杂指令集（CISC）结构和技术特征

在计算机发展初期，软件编程方式比较落后，为软件编程方便和提高程序的运行速度，不断增加可实现复杂功能的指令和灵活的寻址方式。为实现程序兼容，同一系列的新机器的指令系统只能扩充而不能减少，也使指令系统愈加复杂。

CISC指令集的处理器主要特点：指令的长度不同，指令译码步骤比较复杂，指令集庞大。

随着技术的发展CISC指令集的执行效率和计算机硬件资源利用率低下等问题越来越突出。

(2)精简指令集（RISC）结构和技术特征

RISC指令集从如何使计算机的结构更加合理，如何提高处理器的运行速度出发，优选使用频率高的简单指令，避免复杂指令；将指令长度固定、指令格式和寻址方式种类减少；CPU的控制功能改由硬件逻辑电路实现，取代CISC结构中用软件程序实现控制功能方式。从而提高执行效率。

RISC具有以下特点：指令集有限、简单；单周期指令；大量使用寄存器

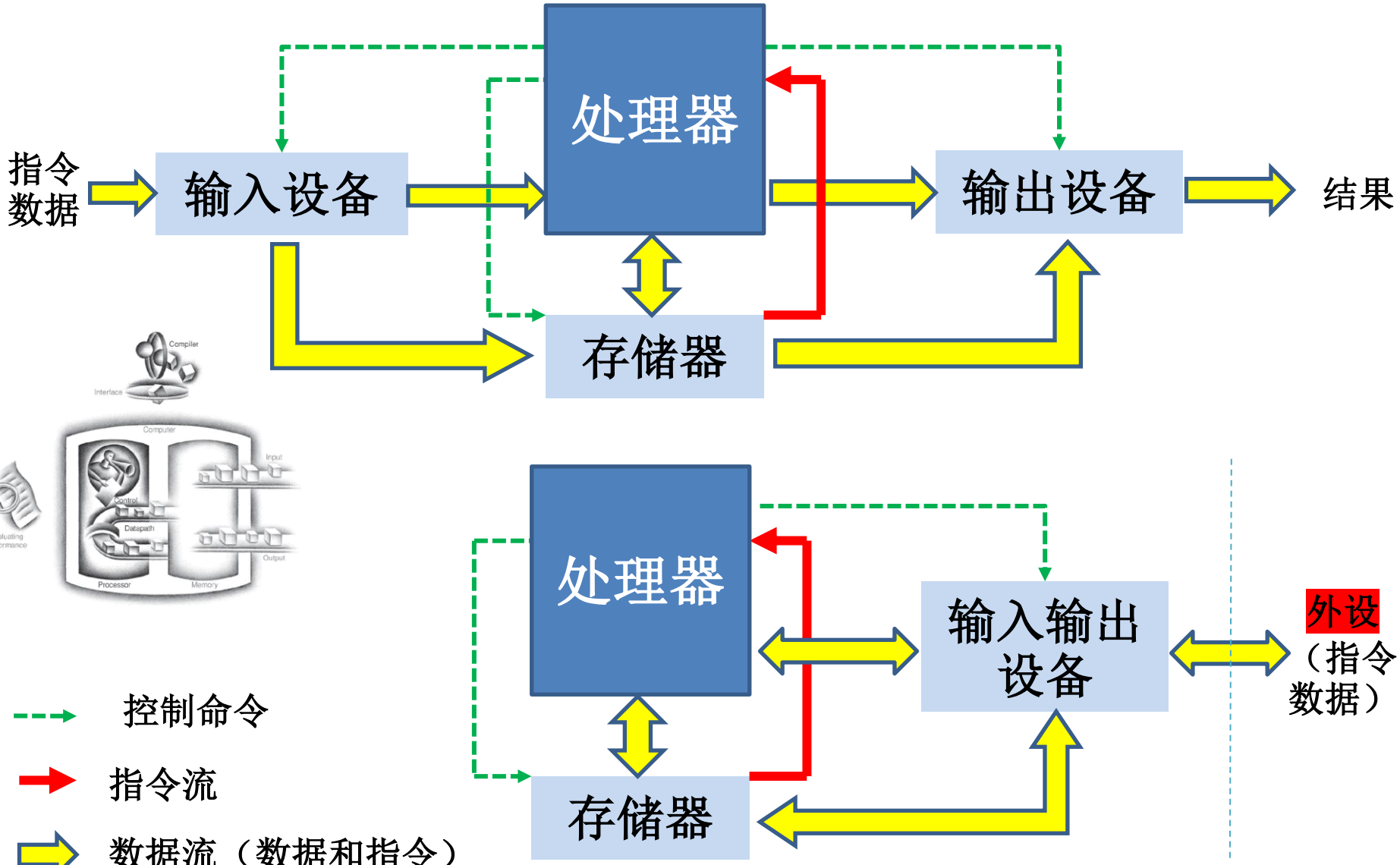
1940年数学家冯·诺依曼首先提出计算机体系结构，其基本设计思想为：

①以二进制形式表示指令和数据。

②程序和数据事先存放在存储器中，计算机在工作时能够高速的从存储器中取出指令加以执行。

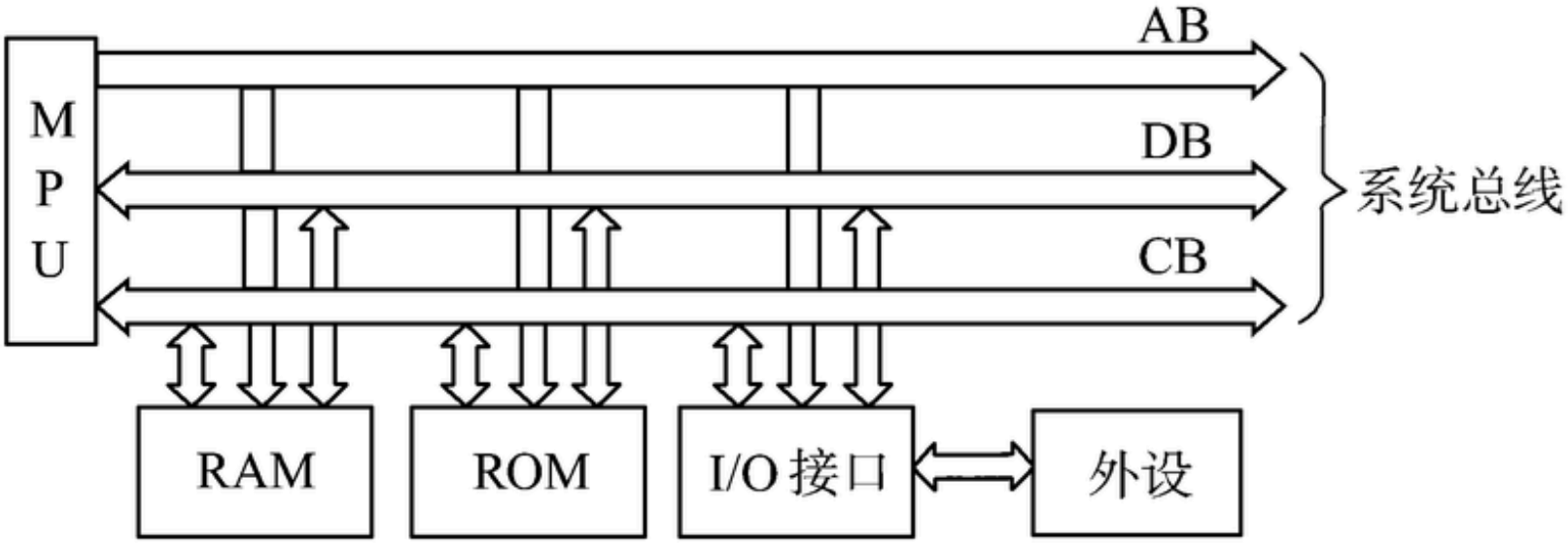
③由运算器、控制器、存储器、输入设备和输出设备等五大部件组成计算机系统。

到目前为止，部分微型计算机仍沿用冯·诺依曼的体系结构。


处理器
指令
数据
输入设备
输出设备
结果
存储器
Compiler
Interface
Computer
Input
Control
Datapath
Output
Processor
Memory
Evaluating performance
处理器
输入输出
设备
外设
（指令
数据）
存储器
控制命令
指令流
数据流（数据和指令）

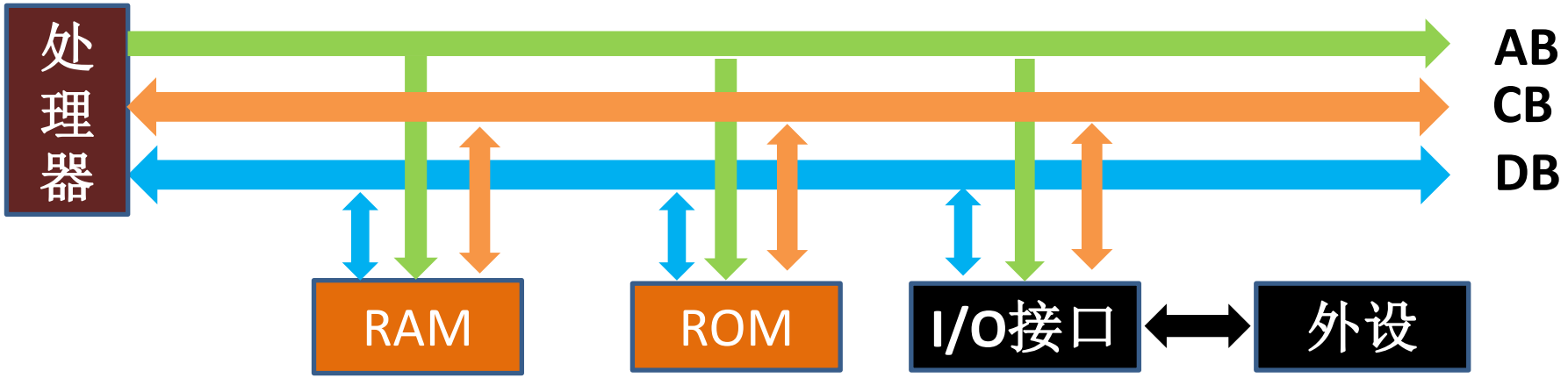
第1章 微型计算机概述

主机与外部设备之间的信息交换是通过输入/输出接口来进行的。输入/输出接口简称I/O接口，由图可以看出，**接口在这里起着主机与外部设备之间数据通信的“桥梁”的作用**。



- 接口（*interface*）：是介于微处理器和外设之间的一种缓冲电路,是*CPU*与外部设备交换信息交换的中转站

系统（内）总线分类：



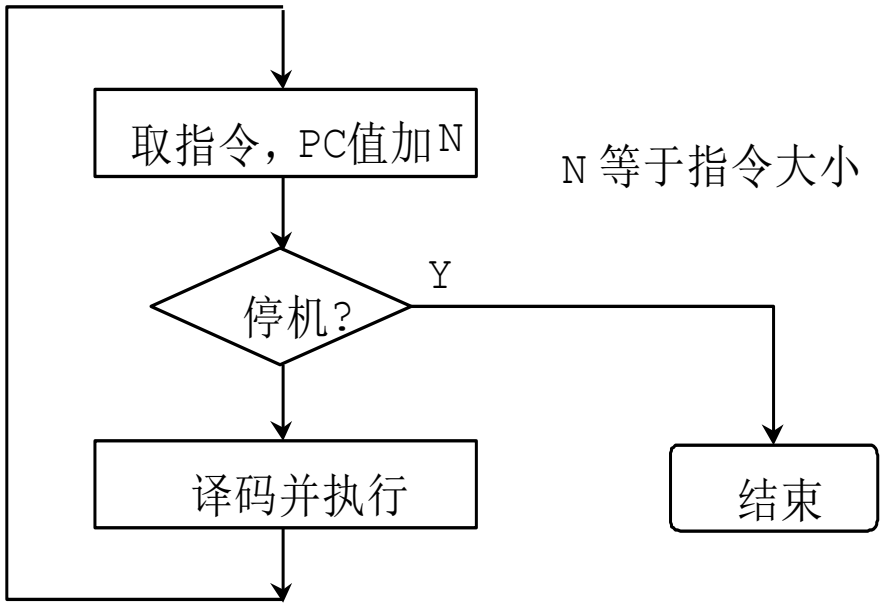
按传输信息类型

- 数据总线DB　双向传输，位宽通常等于字长
- 地址总线AB　由CPU向设备单向传输，数据线宽度决定地址寻址范围
- 控制总线CB　有往CPU传输的，有往设备传输的

中断请求、总线请求、设备状态信号等

总线控制授权、读写控制、设备选择控制等

- 存储程序工作原理



存储程序工作原理：**先存储，再执行**。即先把程序和数据送到具有记忆功能的存储器中保存，然后将程序第一条指令的地址送到程序计数器（PC），控制器依次取出存储器中的指令，集合相应的数据，执行每条指令，直至结束。

（1）指令的概念

- 指令是发送到CPU的命令，指示CPU执行一个特定的处理，如数据搬移、对数据进行算术运算等
- 指令通常包含操作码和操作数两部分
  - 操作码指明要完成的操作功能，如加、减、数据传送、移位等；
  - 操作数指明上述规定操作的数据或获取该数据所存放地址的方法。
- 指令通常有两种形式
  - 机器指令：包含操作码和操作数，由0和1组成，CPU能直接解析、执行并完成一定功能，

    如1011 0000 0101 1100
  - 汇编指令：与机器指令功能一一对应，由具有一定含义的字符串、寄存器名和立即数组成，

    如：MOV R1,#5CH

(2) 程序执行过程

- 程序由完成一定功能的多条指令组成
- 指令执行的过程分为N阶段，当N=3时：

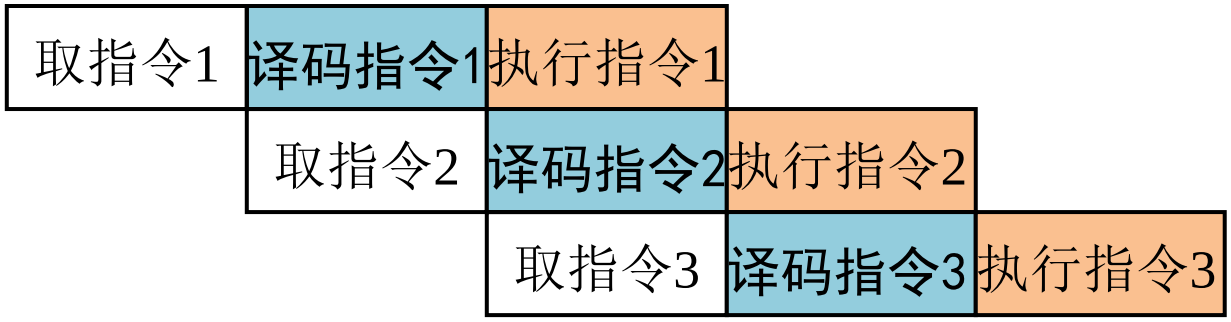
- 取指令：从存储器读取指令(fetch)。
- 指令译码：CPU的控制器解析指令(decode)。
- 执行指令：各个控制信号控制CPU内部和外部电路完成对应操作功能(execute)。

- 程序运行的过程，实际上就是周而复始地完成这3个阶段操作的过程，直至遇到停机指令时才结束整个机器的运行。早期计算机的指令执行是一个串行过程：取指令、译码指令和执行指令。

取指令1	译码指令1	执行指令1	取指令2	译码指令2	执行指令2

（3）指令流水线工作原理

- 指令流水线类似于工厂的装配线。一个产品有若干个部件，不同的部件在装配线上的不同阶段同时装配，不同部件的装配在时间上具有重叠性。同样的，完整执行一条指令可分为多个阶段，由不同的部件来同时完成指令执行的不同阶段。
- 有了流水线结构，不同指令的取指、分析、执行3个阶段可并行处理。下图是一个典型的三级流水的工作流程，通过三级流水提高执行效率。



- CPU时序是CPU在执行指令时所需控制信号的时间顺序。
- 时钟周期：是CPU工作的最小时间单位。CPU完成任何操作需要的时间，均是时钟周期的整数倍。
- 指令周期：一条指令从其代码被从内存单元中取出，到操作执行完毕所用的时间。
- 总线周期：把CPU通过总线与内存、外设或者内部某部件之间，进行一次数据交换所进行的操作所需要的时间。

计算机中信息的表示

1、 计算机中整数的几种二进制表示方法

（假设以下讨论都是使用n位二进制数）

- 无符号数

$$Value(B) = b_{n-1}\times 2^{n-1} + \dots + b_1\times 2^1 + b_0\times 2^0$$

- 有符号数（sign-and-magnitude）

最高位表示符号位：0表示正（+），1表示负（-）。

- 反码

最高位表示符号位：0表示正（+），1表示负（-），数值位取反。

- 补码

最高位表示符号位：0表示正（+），1表示负（-），数值位取反+1。

计算机中信息的表示

2、 计算机中小数的表示法

对于32位计算机，

方法一:表示的数的范围：

整数：-2^{31} ～$2^{31}-1$（小数点在最右侧）；

小数：-1 ～2^{-31} （小数点在符号位右侧）。

Problems:

这种表示方法还无法满足很多科学和工程计算需求，希望用32位二进制数，能够表示更大的整数和更小的分数（小数）。

计算机中小数的表示法

方法二：采用科学记数法表示小数：$F \times 2^E$。因为科学记数法中小数点位置是变化的，所以称为浮点（float），表示的数称为浮点（float-Point）数。

这样的数需要表示：F，E和符号位。

IEEE 754 Floating-Point Standard 32(64) bits:

32位单精度浮点数

$N = -1^S \times 1.\text{fraction} \times 2^{\text{exponent}-127}, 1 \le \text{exponent} \le 254$ 规约

$N = -1^S \times 0.\text{fraction} \times 2^{-126}, \text{exponent} = 0$ 非规约

计算机中信息的表示

浮点数例子：

IEEE单精度浮点数：

- 符号：1 – 表示负数.
- 指数：01111110 = 126 (decimal).
- 尾数：0.10000000000… = 0.5 (decimal).

则：　　$\text{Value} = -1.5 \times 2^{(126-127)} = -1.5 \times 2^{-1} = -0.75$.

计算机中信息的表示

3. 大小端分配

- Big-Endian就是数据的高字节位放在内存的低地址端，低位字节排放在内存的高地址端。

  数据高位存放低地址端

- Little-Endian就是数据的高位字节排放在内存的高地址端，低位字节排放在内存的低地址端。

  数据高位存放高地址端

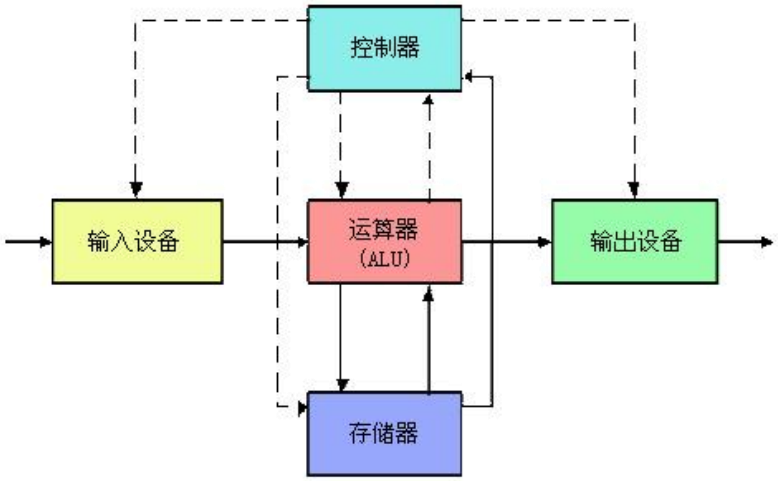
- 比特序：

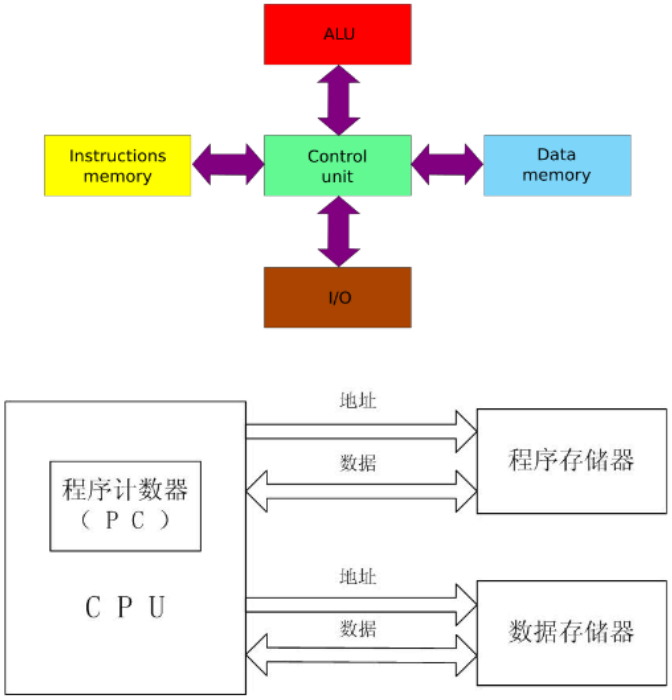
  字节比特序：LSB（Least Significant Bit），

  MSB (Most Significant Bit)

ARM微处理器主要特点如下：

(1) ARMv7系列以前的处理器采用冯·诺依曼体系结构，Cortex M3、M4（F）采用哈佛体系结构，M7是改进后的哈佛体系结构（a variant of the Harvard Architecture）





ARM微处理器主要特点如下：

(2) ARM采用RISC指令，硬件实现有限且最常用的指令，大部分复杂的操作则利用成熟的编译技术简化为多条简单指令。

(3) 定义了多种处理器工作模式，提高了处理器工作效率。

(4) 大量使用寄存器。ARM 处理器中使用大量寄存器来防止大量内存数据交互。寄存器充当所有操作的本地存储器，存储数据和地址。

(5) 具有灵活方便的接口。

Cortex-M4/M7主要特点如下：

(1)基于哈佛架构，取指令和数据访问可以同时进行。

(2) M4三级/M7六级流水线设计。

(3) 32位寻址，可支持4GB存储器空间。

(4)基于ARM AMBA技术的片上接口，支持高吞吐量的流水线总线操作。

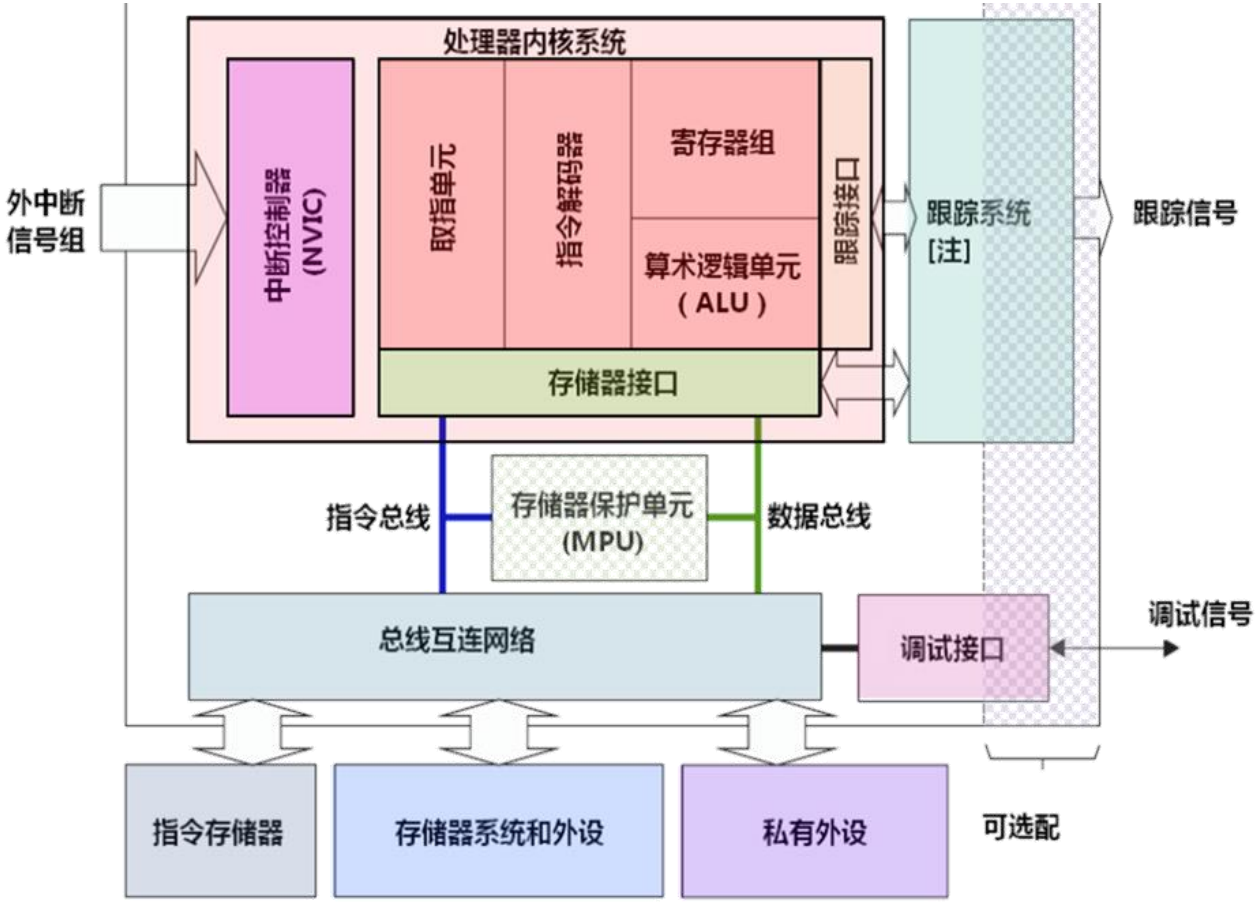
(5)内嵌的嵌套向量中断控制负责中断处理，自动处理中断优先级、中断屏蔽、中断嵌套和系统异常处理。

Cortex-M4/M7主要特点如下（续）：

(6)支持内存保护单元（ MPU），提供内存保护，提高安全性。

(7)通过位段特性支持两个特定存储器区域中的位数据访问。

(8)从经典ARM微处理器仅支持ARM、Thumb指令，扩展到可同时支持16位和32位指令的Thumb-2版本。

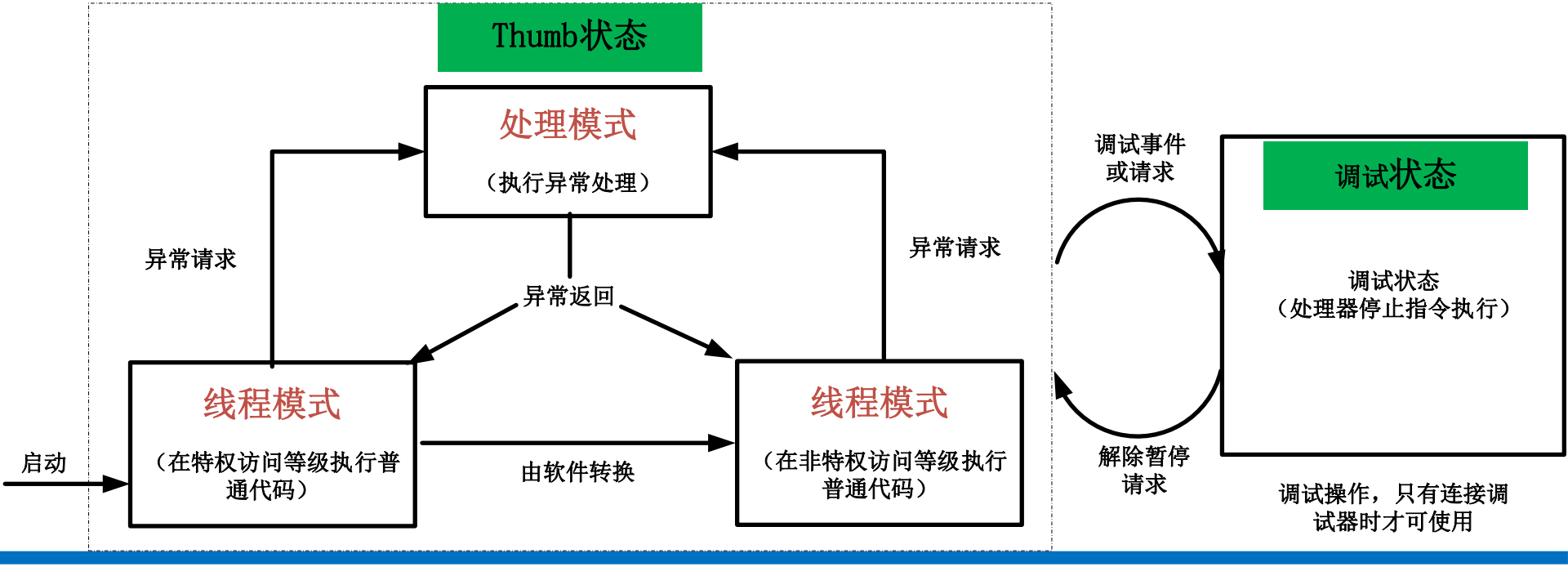


ARMv7_M处理器内部：

- 寄存器
- ALU
- 存储器
- 总线

ARMv7-M架构微处理器工作状态和模式

ARMv7-M架构微处理器有两种工作状态和两种模式，即Thumb状态和调试状态；处理模式和线程模式。处理器在线程模式时可以分特权和非特权访问等级。



ARMv7-M架构微处理器工作状态和模式（续）

1. 工作状态

第一种为调试状态：

调试状态仅用于调试操作，可以通过两种方式进入调试状态：调试器发起的暂停请求，或处理器中的调试部件产生的调试事件。在此状态下，调试器可以访问或修改处理器寄存的数值。

第二种为Thumb状态：

当处理器执行程序代码(Thumb指令)时，系统处于Thumb状态。

无论在Thumb状态还是调试状态，调试器都可以访问处理器内外的外设和系统存储器。

ARMv7-M架构微处理器工作状态和模式（续）

2. 工作模式

ARMv7-M架构微处理器有两种工作模式：线程模式和处理模式。

线程模式(thread mode)：在复位或异常返回时进入该模式。ARMv7-M架构微处理器处于线程模式时既能使用特权级，也可以使用用户级（非特权）代码。

处理模式(handler mode)：执行中断服务程序等异常处理。在处理模式下，处理器具有特权访问等级。

通过软件可以将处理器从特权线程模式切换到非特权线程模式，但是无法从非特权线程模式切换到特权线程模式。如果要进行这种切换的话，处理器需要借助于异常机制。

ARMv7-M架构微处理器在启动后默认处于线程模式以及Thumb状态。

ARMv7-M架构微处理器工作状态和模式（续）

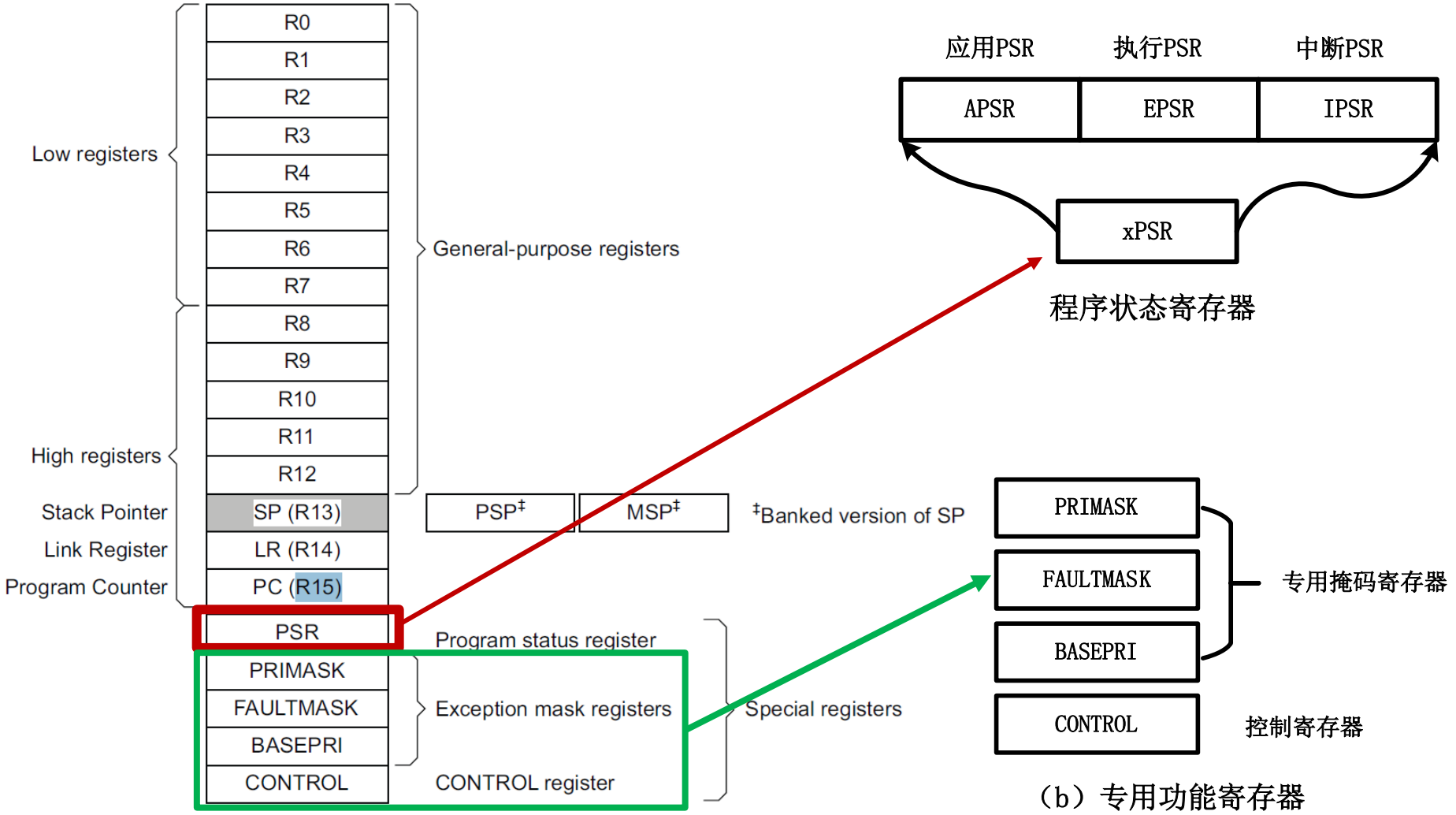
3. 访问级别

线程模式是 ARMv7-M 架构处理器中执行应用程序的基本模式，在该模式下支持特权和非特权访问，由代码决定应用程序处于特权访问还是非特权访问方式。如果操作系统同时支持特权访问和非特权访问，应用程序一般以非特权访问方式运行，允许操作系统分配系统资源供应用程序专用或共享使用，并针对其他进程和任务提供一定程度的保护。

系统复位后进入特权访问的线程模式，可以由软件切换至非特权模式。无论线程模式下处于特权访问还是非特权访问级别，都可以通过执行 SVC 指令触发Supervisor Call异常（SVC通常用于在操作系统上请求特权操作或访问系统资源），请求处理系统访问控制和系统资源访问。

所有异常处理程序都在特权访问下运行（此时处理模式）。

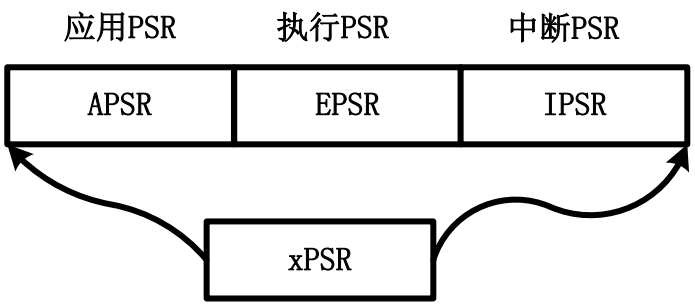
ARMv7-M架构处理器的寄存器组



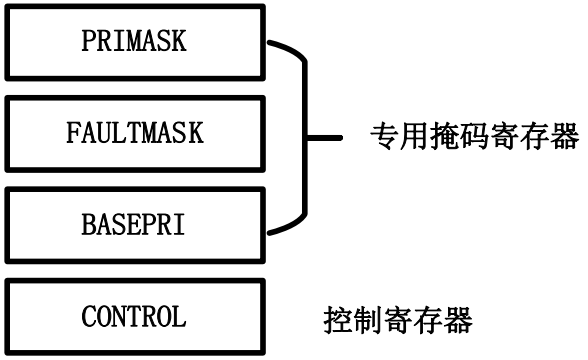
2. 专用功能寄存器

除了通用寄存器组中的寄存器外，处理器中还存在多个专用功能寄存器。这些专用功能寄存器表示处理器状态，定义了操作状态和中断/异常屏蔽。

专用功能寄存器分为程序状态寄存器、专用掩码寄存器和控制寄存器3类。



程序状态寄存器



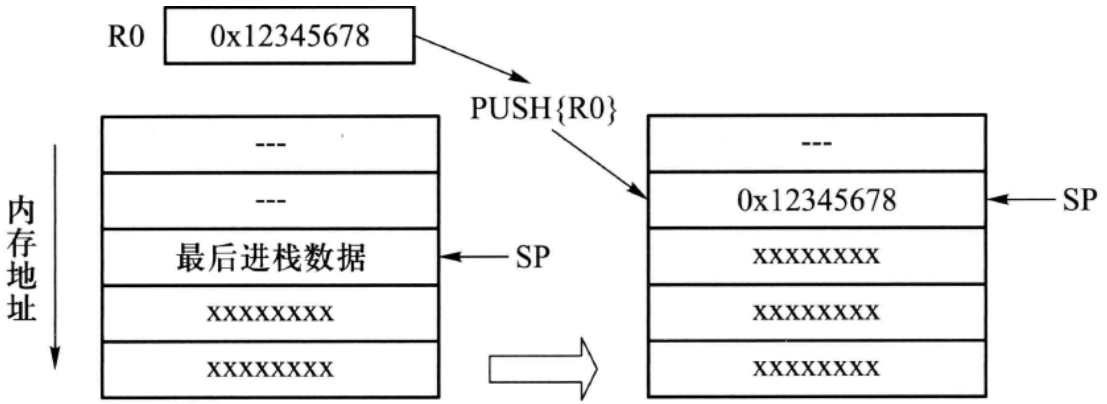
专用功能寄存器

3.栈内存操作

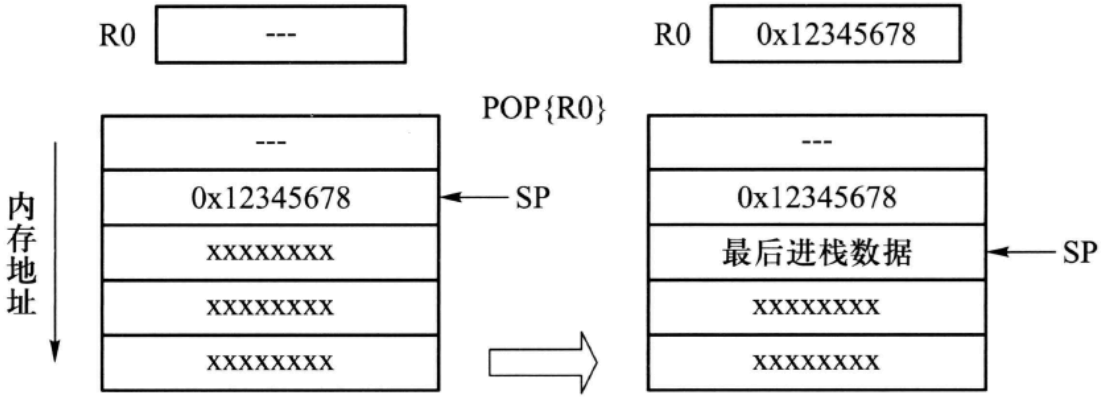
ARMv7-M微处理器使用的是“满递减”栈。

执行PUSH指令时，处理器首先减小堆栈指针的值，然后将数据存储在堆栈指针所指的存储器位置。

执行POP指令时，堆栈指针指向的存储器位置的数据被读出，然后堆栈指针的值自动增加。



(a) 执行进栈PUSH指令



(b) 执行出栈POP指令

图 2-3-6 执行 PUSH 和 POP 指令

1）Thumb-2指令语法格式：

```
<opcode> {<cond>} {S}   <Rd> ,<Rn>{,<operand2>}
```

其中<>号内的项是必须的，{}号内的项是可选的。各项的说明如下：

opcode：指令助记符；

cond：执行条件；

S：是否影响APSR寄存器的值；

Rd：目标寄存器；

Rn：第1个操作数的寄存器；

operand2：第2个操作数；

- 寻址方式：

  如何根据指令中给出的地址信息寻找操作数的物理地址。

- 操作数可能在：
  - 指令中直接给出：立即数
  - 寄存器
  - 内存单元

8种基本寻址方式。

1.立即寻址；

2. 寄存器寻址；

3.寄存器移位寻址；

4.寄存器间接寻址；

5.基址变址寻址；

6.多寄存器寻址；

7.堆栈寻址；

8.相对寻址。

Thumb-2指令指令种类

1. 数据传送指令
2. 存储器访问指令
3. 算术运算指令
4. 逻辑运算指令
5. 移位和循环指令
6. 符号扩展指令
7. 字节调序指令
8. 位域处理指令
9. 比较和测试指令
10. 子程序调用与无条件转移指令
11. 伪指令

一段汇编语言程序一般由机器指令、汇编指示命令和宏命令组成。

机器指令（Machine Instructions）是CPU能直接识别并执行的指令，包含操作码和操作数，表现形式为二进制编码。

汇编指示命令（Assembler Directives）在源程序中是为了完成汇编程序作各种准备，仅在汇编过程中起作用，一旦汇编结束，汇编指示命令即完成使命。

宏命令是一段独立的程序代码，是通过汇编指示命令定义的一段代码，汇编时用原汇编程序取代（展开），宏命令结束其使命。

第4章 ARM汇编程序设计基础

- 符号定义指示命令
- 数据定义指示命令
- 汇编控制指示命令
- 其他指示命令

基本的AAPCS（Procedure Call Standard for the Arm Architecture）

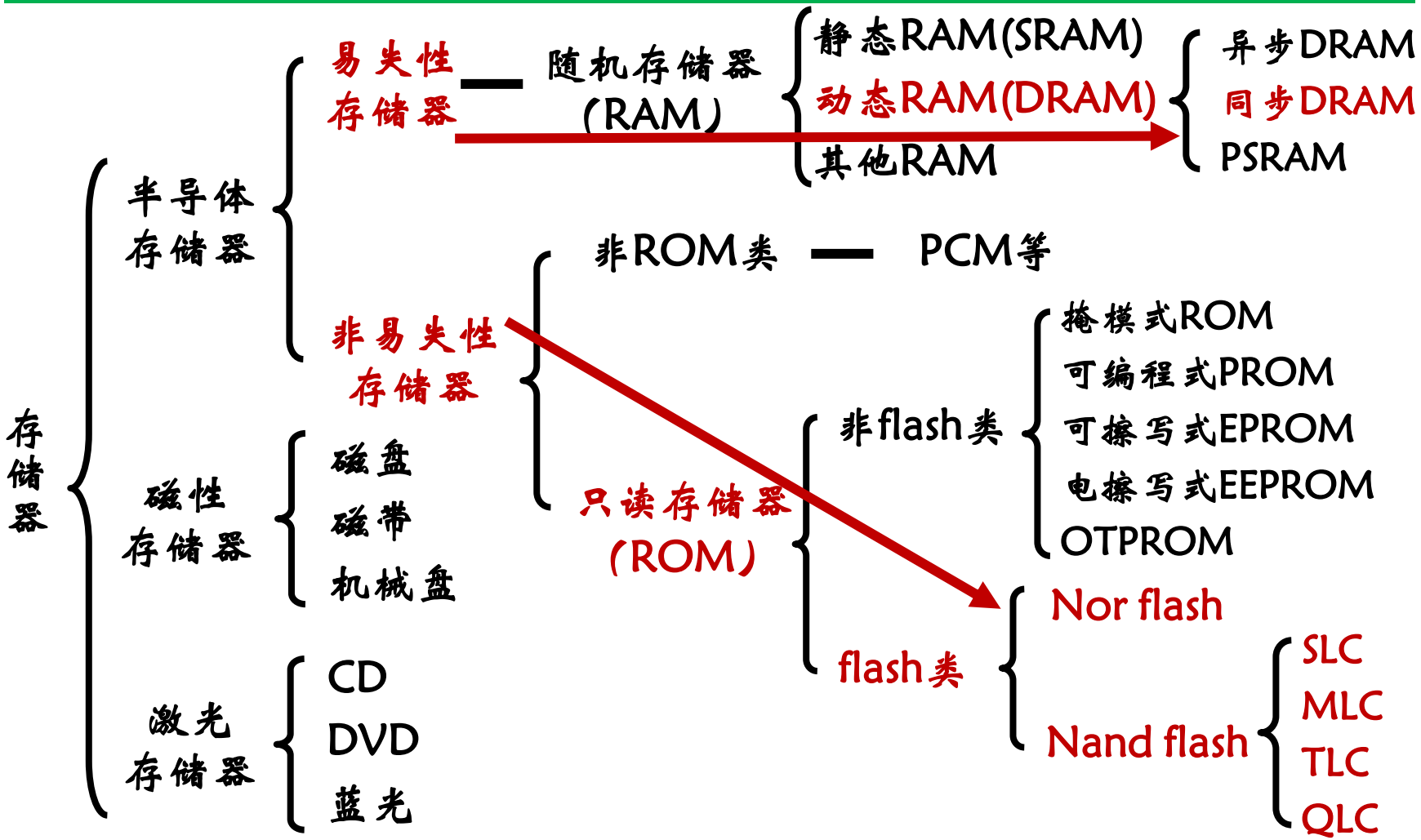
基本的AAPCS规定了在混合编程时子程序调用的一些基本规则，主要包括寄存器的使用、堆栈的使用、参数传递和子程序结果的返回等方面的规则。

1. 寄存器的使用规则
2. 堆栈使用规则
3. 参数传递规则
4. 子程序返回规则

第4章 ARM汇编程序设计基础

- C与汇编的相互调用
- 嵌入汇编

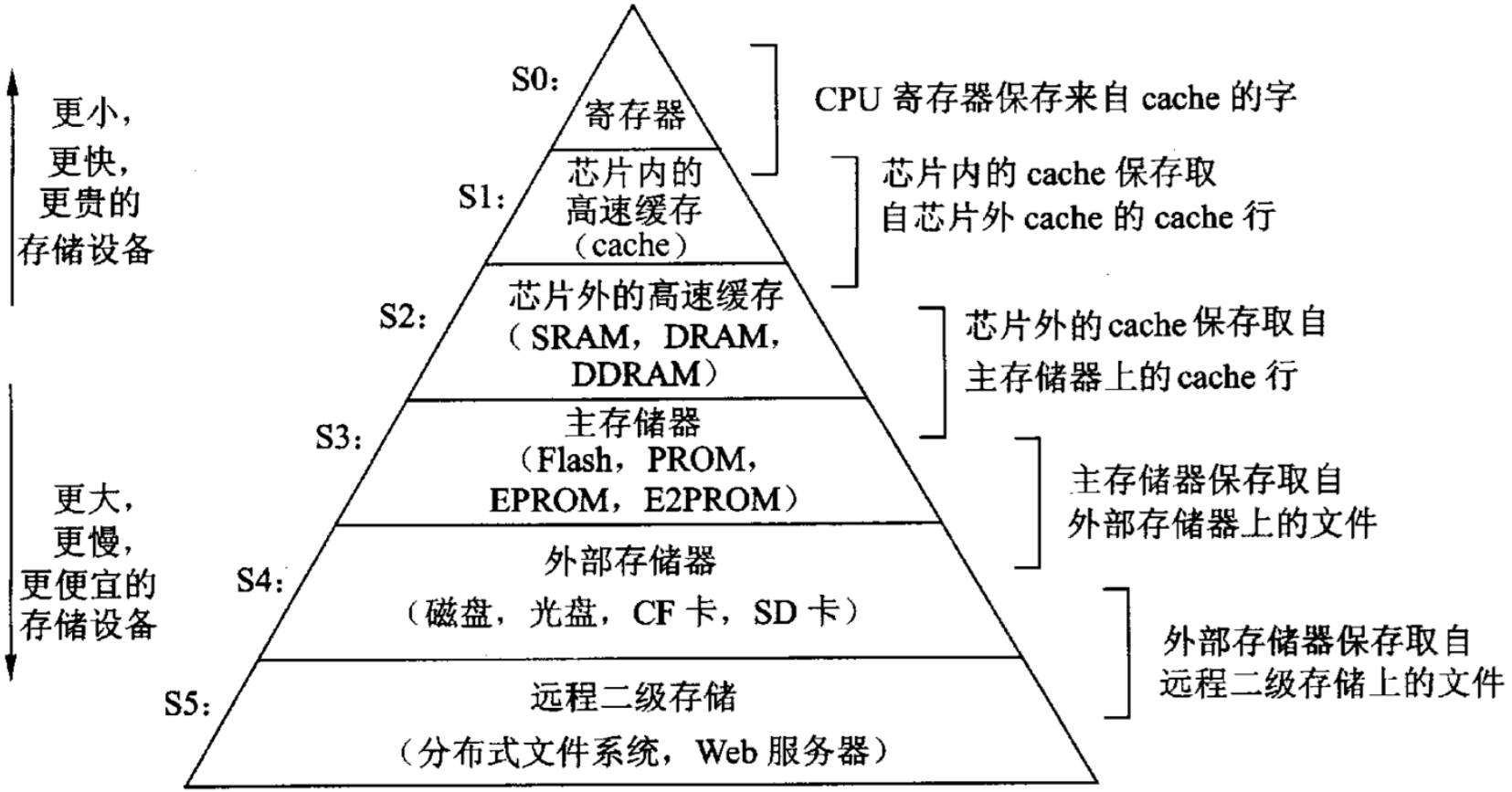
存储器的分类：


存储器
半导体存储器
易失性存储器
随机存储器（RAM）
静态RAM(SRAM)
动态RAM(DRAM)
其他RAM
异步DRAM
同步DRAM
PSRAM
非易失性存储器
非ROM类
PCM等
只读存储器（ROM）
非flash类
掩模式ROM
可编程式PROM
可擦写式EPROM
电擦写式EEPROM
OTPROM
flash类
Nor flash
Nand flash
SLC
MLC
TLC
QLC
磁性存储器
磁盘
磁带
机械盘
激光存储器
CD
DVD
蓝光

存储器技术指标：

- **存储容量**
  - 字节数或比特数
  - 1M比特 = 128K×8bit = 64K×16bit = 32K×32bit
- **存取速度**
  - 访问时间T_A是指启动一次存储器操作（读或写）到完成该操作所需要的时间，取决于存储介质的物理特性和寻址部件的结构
  - 存取周期T_M是指在存储器连续的读写过程中一次完整的存取操作所需的时间，T_M大于T_A
- **数据总线带宽**
- **端口逻辑电平和驱动能力**
- **工作寿命、可靠性、功耗、体积、价格**等各项指标

存储器的层次结构：

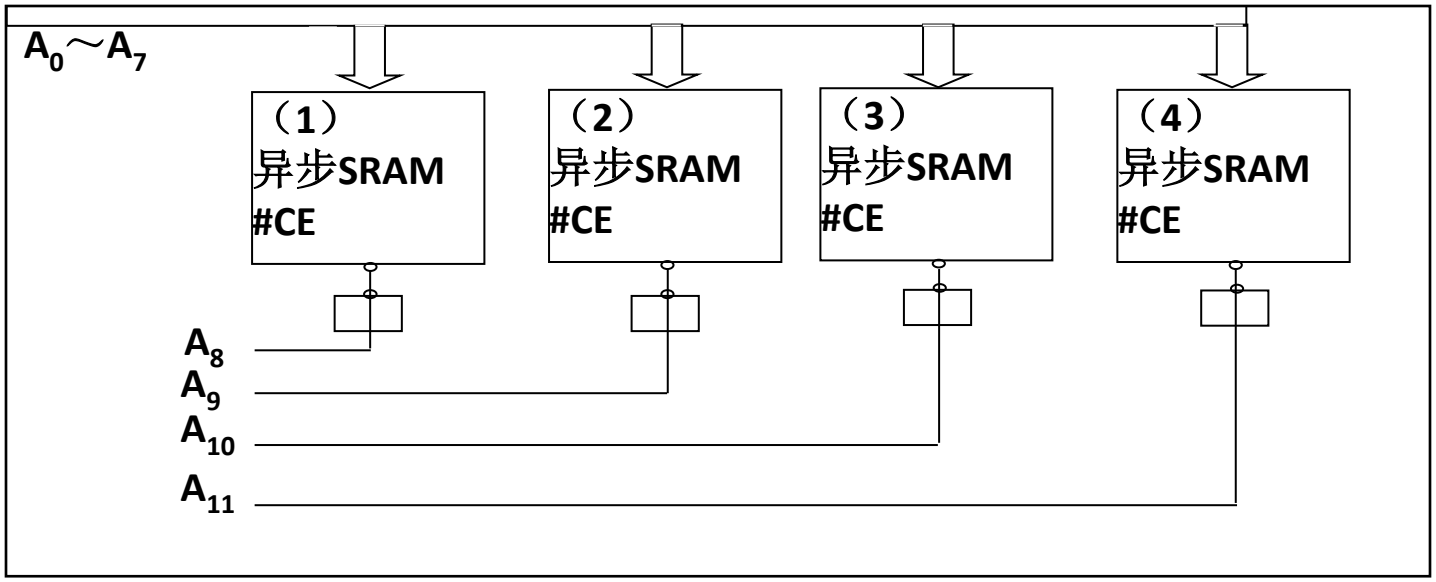

更小，
更快，
更贵的
存储设备
更大，
更慢，
更便宜的
存储设备
S0：
寄存器
S1：
芯片内的
高速缓存
（cache）
S2：
芯片外的高速缓存
（SRAM，DRAM，
DDRAM）
S3：
主存储器
（Flash，PROM，
EPROM，E2PROM）
S4：
外部存储器
（磁盘，光盘，CF 卡，SD 卡）
S5：
远程二级存储
（分布式文件系统，Web 服务器）
CPU 寄存器保存来自 cache 的字
芯片内的 cache 保存取
自芯片外 cache 的 cache 行
芯片外的 cache 保存取自
主存储器上的 cache 行
主存储器保存取自
外部存储器上的文件
外部存储器保存取自
远程二级存储上的文件

高速缓冲存储器(Cache):

- **CACHE的作用**：在原有的内存和CPU之间增加了更高速的CACHE，把当前正在执行的指令地址附近的一部分指令或当前正在使用的数据可以从内存中调入CACHE中，以提高系统速度；
- 引入CACHE的**依据**：程序的局部性原理，也就是在某一段时间内频繁访问某一局部的存储器地址，而对此以外的地址则很少访问的现象；
- **命中率**： CPU访问CACHE，找到信息的百分比；高速缓存器的设计目标是使CPU访问尽可能在CACHE中进行；

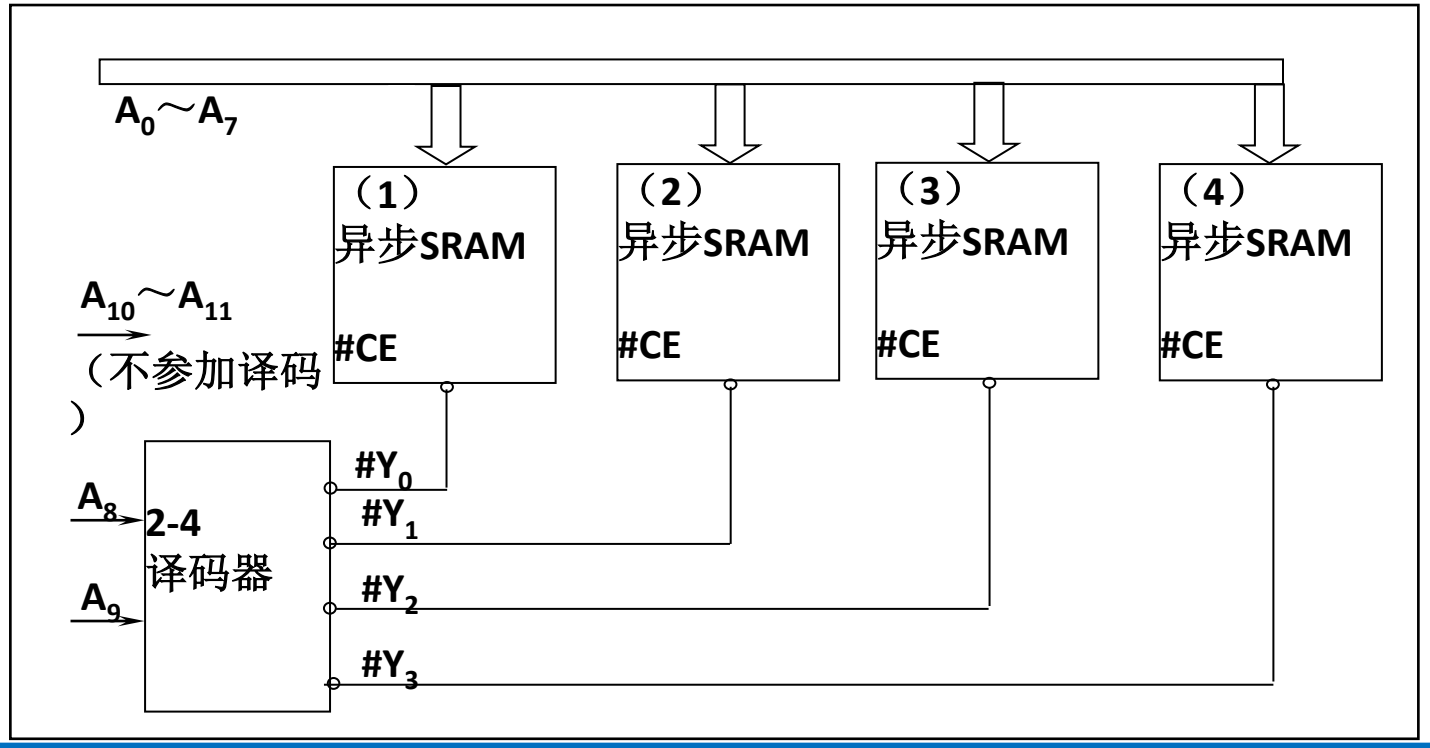
存储器的接口设计：

- 地址译码的方法：
  - 线选法（优缺点）



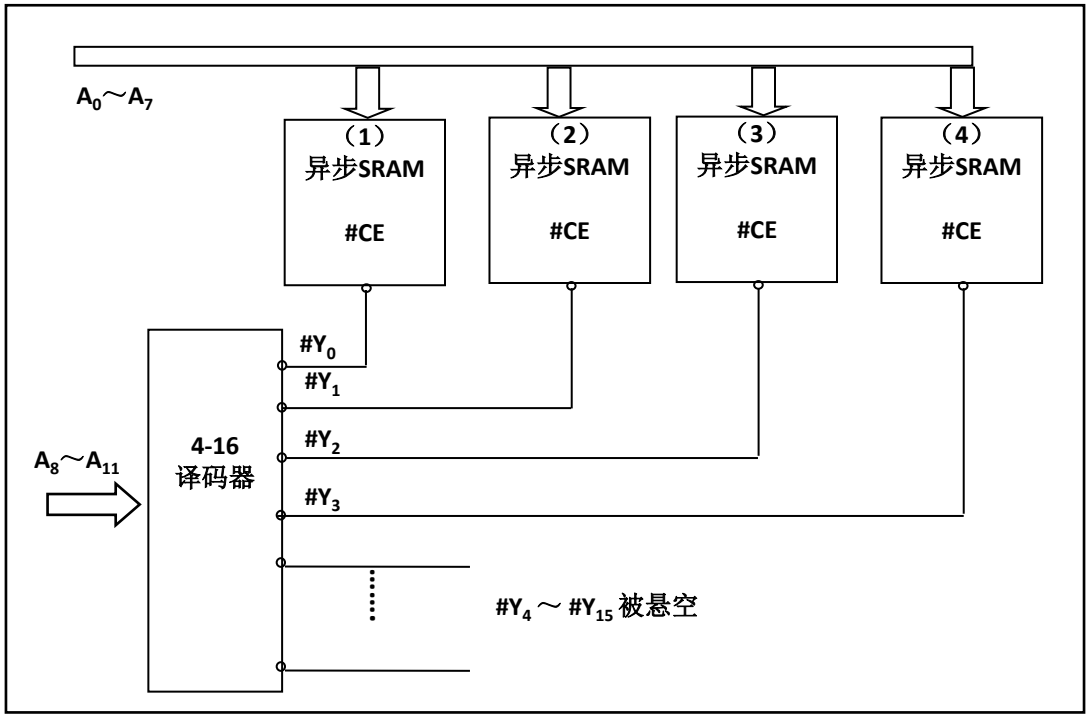
存储器的接口设计（续）：

- **地址译码**的方法：
  - **部分译码**法 （优缺点）



存储器的接口设计（续）：

- 地址译码的方法：
  - 全译码法 （优缺点）



CPU与外设传送数据方式：

一、无条件传送方式

- 在传送信息时，已知外设是“准备好”的；所以CPU在和外设交换数据时，不用查询外设的状态，直接进行输入/输出；
- 一般用于：
  - 外设是简单设备，例如：开关，发光二极管等；
  - 外设工作速度足够快
  - 两次数据传送的间隔时间足够长
- 接口电路中一般有输入缓冲器/输出锁存器（数据端口），由地址译码的输出信号来选通它们。

二、查询传送方式

- 这种方式，CPU要遵循“先查询，后传送”的原则，保证只有在外设已经是在“准备好”状态，才开始传送数据；
- 接口电路中至少需要两个端口：状态端口和数据端口

- 查询式传送的一般流程：
  - 先从状态口读入状态字；
  - 如果状态是“准备好”，开始传送；
  - 如果状态是“没有准备好”，则继续查询，直到“准备好”，开始传送；

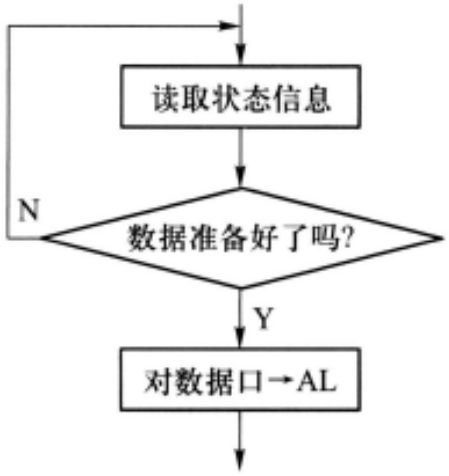


图 8-1-5 查询式输入程序流程图

二、查询传送方式

- 在编写查询式传送程序时，要先确定两个问题：
  - 状态信号的位置：需要规定状态信号是在寄存器中的第几位；
  - 状态信号的有效电平：即是高电平表示准备好，还是低电平表示外设准备好；
- 优点：CPU和外设之间可以很好地配合工作；
- 缺点：CPU要长期地查询外设的状态，查询实际上就是一种等待；CPU长期的等待会影响CPU的工作效率；

三、中断传送方式

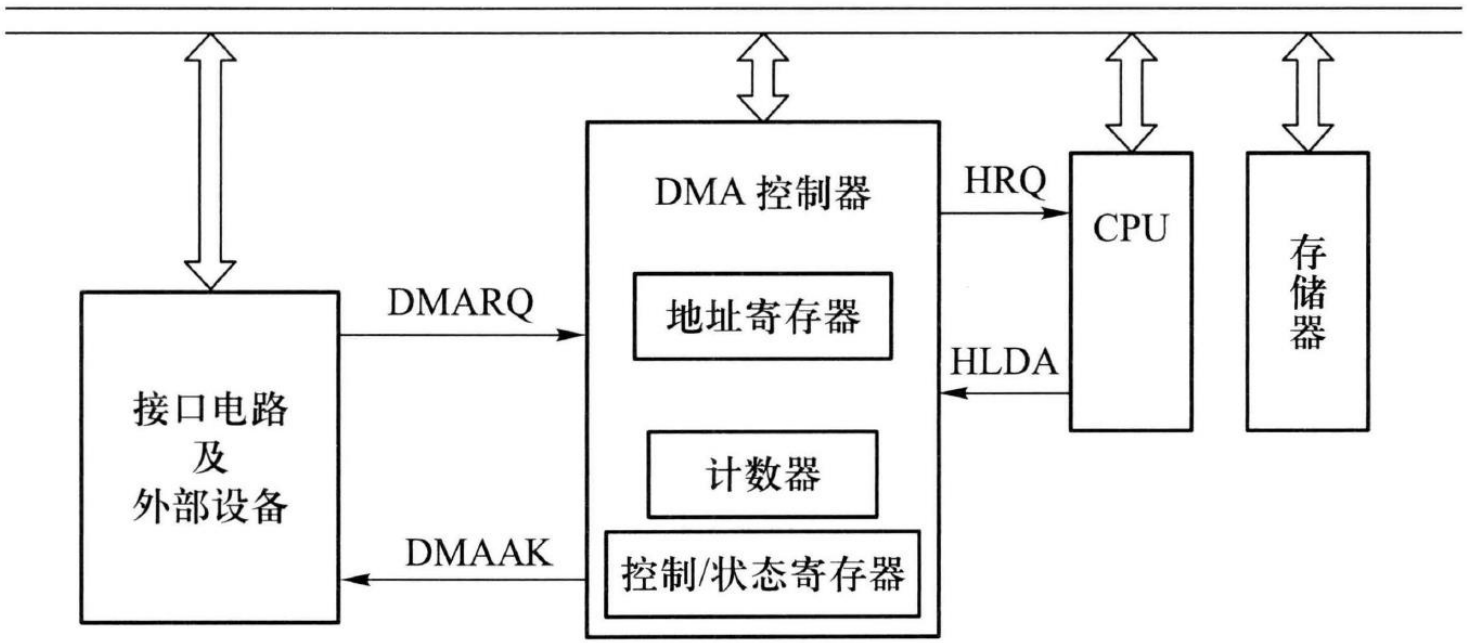
- 当外设要输入/输出时（即外设已把待输入数据存放在数据输入寄存器，或输出时外设已把上一个数据输出，输出寄存器已空），向CPU发中断请求；
- CPU收到中断请求后，中断当前的工作，为外设服务，服务结束(输入或输出)后，继续原来的工作；
- 中断方式允许CPU与外设(甚至多个外设)同时工作，克服了查询方式的缺点；

四、直接存储器存取方式（DMA）

- 中断方式仍是CPU通过程序来传送，每次传送一字节；为了要传送这一字节需要将保护断点，保护现场等多条指令执行一遍；
- 对一个高速I/O设备，成组交换数据的情况，中断方式显得不合适；
- DMA方式是由硬件(DMA控制器)来控制数据从外设到存储器的直接传送，而不通过CPU；
- DMA方式下，要求CPU让出这些总线，即要求CPU相应的引脚输出为高阻状态，系统总线由DMA控制器接管这些，产生相应的总线信号；

四、直接存储器存取方式（DMA）

- DMA 传送的工作原理：



四、直接存储器存取方式（DMA）

- DMA传送的流程：
  - CPU通过指令将要传送数据块的长度、在内存的首地址等信息写入DMA控制器。
  - 外部设备通过DMA控制器向CPU发出DMA请求。
  - CPU接收DMA请求，暂停执行的程序，放弃对总线控制，由DMA控制，存储器的数据线和外设的数据线经由DMA控制器直接连通。
  - DMA控制器发出地址信息，能对存储器寻址及修改地址指针；
  - DMA控制器发出存储器和外设的读或写控制信号。
  - 数据块长度减一，重复上面进程，直到传送完毕。
  - 发出DMA结束信号，使CPU恢复正常工作状态。

中断和异常：

中断：是指计算机的CPU暂时中止它正在执行的程序，转去执行请求中断的那个外设或事件的中断服务（处理）程序，待处理完后，又返回到被中止了的程序。

异常：是在程序执行过程中如果发生了意外事件。例如：处理器内核产生复位、存取失败、总线错误、遇到未定义指令时，正常执行的程序被暂停，转到相应的处理程序执行。

ARM架构处理器中**中断通常被看作是一种异常**。因为，**中断**对内核来说是**意外事件**，请求信号通常来自外部；**异常**通常是在执行指令或存取中产生。

第7章 中断与异常

中断系统：计算机所具有的上述功能，称为中断功能。为了实现中断功能而设置的各种硬件和软件统称为中断系统。

中断源：能够导致CPU产生中断的来源就成为中断源。

中断优先级：不同的中断可以设置不同的优先等级。

中断嵌套：高优先级中断可以打断低优先级中断。

对于CPU来说中断源有两类：**硬中断**和**软中断**。

硬中断也称为外中断，是由外部的电路在CPU的引脚上产生的中断请求。CPU的硬中断源可以不止一个，取决于CPU有多少可以接受硬中断请求的引脚。

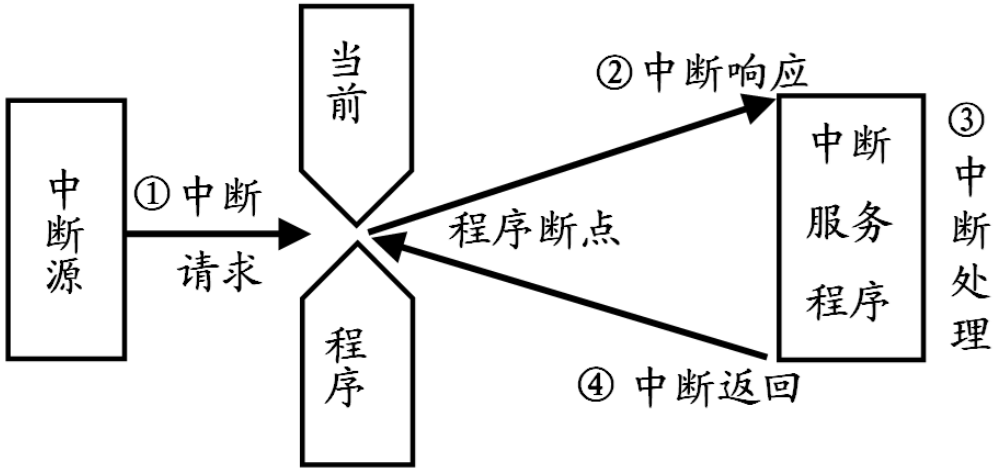
软中断是在CPU执行程序过程中产生的中断请求，不是由外部电路产生。例如，在执行指令时，出现了除法的除数为0的情况，也就是出现了除法溢出。这时，指令无法继续执行，可以通过软中断，调用专门的服务程序来妥善处理除法溢出。CPU的软中断源也不止一个。

中断处理过程：

中断是打断正常运行程序，处理中断事件的过程。这些事情有的是用户做，有的是CPU做。

处理中断的过程一般分为以下几个阶段：

- 中断初始化
- 中断请求
- 中断响应
- 中断服务
- 中断返回



异常处理过程：

1. 初始化过程
2. 异常申请过程
3. 异常响应过程
4. 异常服务
5. 异常返回

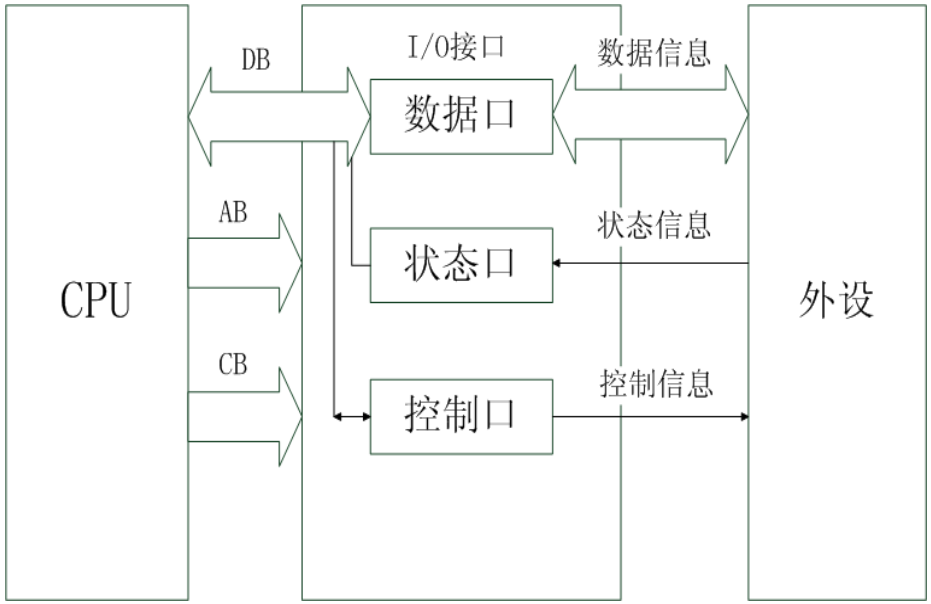
具体内容见12讲ppt

CPU与I/O设备之间的信息：

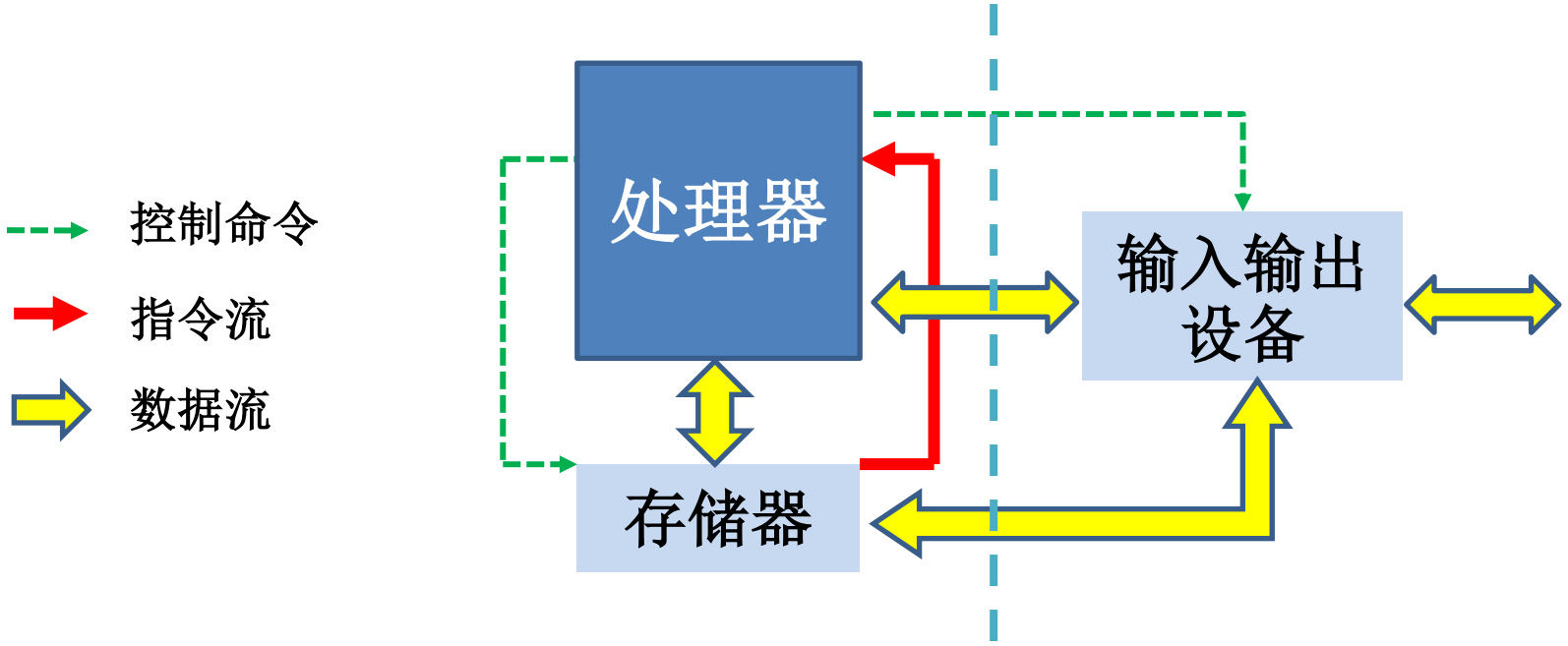
- CPU与I/O设备之间的信息通常包括：

  数据信息，状态信息和控制信

1. 数据信息：数字量、模拟量、开关量
2. 状态信息：状态信息一般是外设通过接口送到CPU的；主要是反映外设的工作状态；
3. 控制信息：控制信息一般是CPU通过接口电路传送给外部设备的，主要用来控制外部设备的动作；



接口与端口定义：



- I/O接口（interface）： 是把外部设备与微处理器连接起来实现数据传送的控制电路称为I/O接口电路；是一种缓冲电路，是CPU与外部设备交换信息交换的中转站。

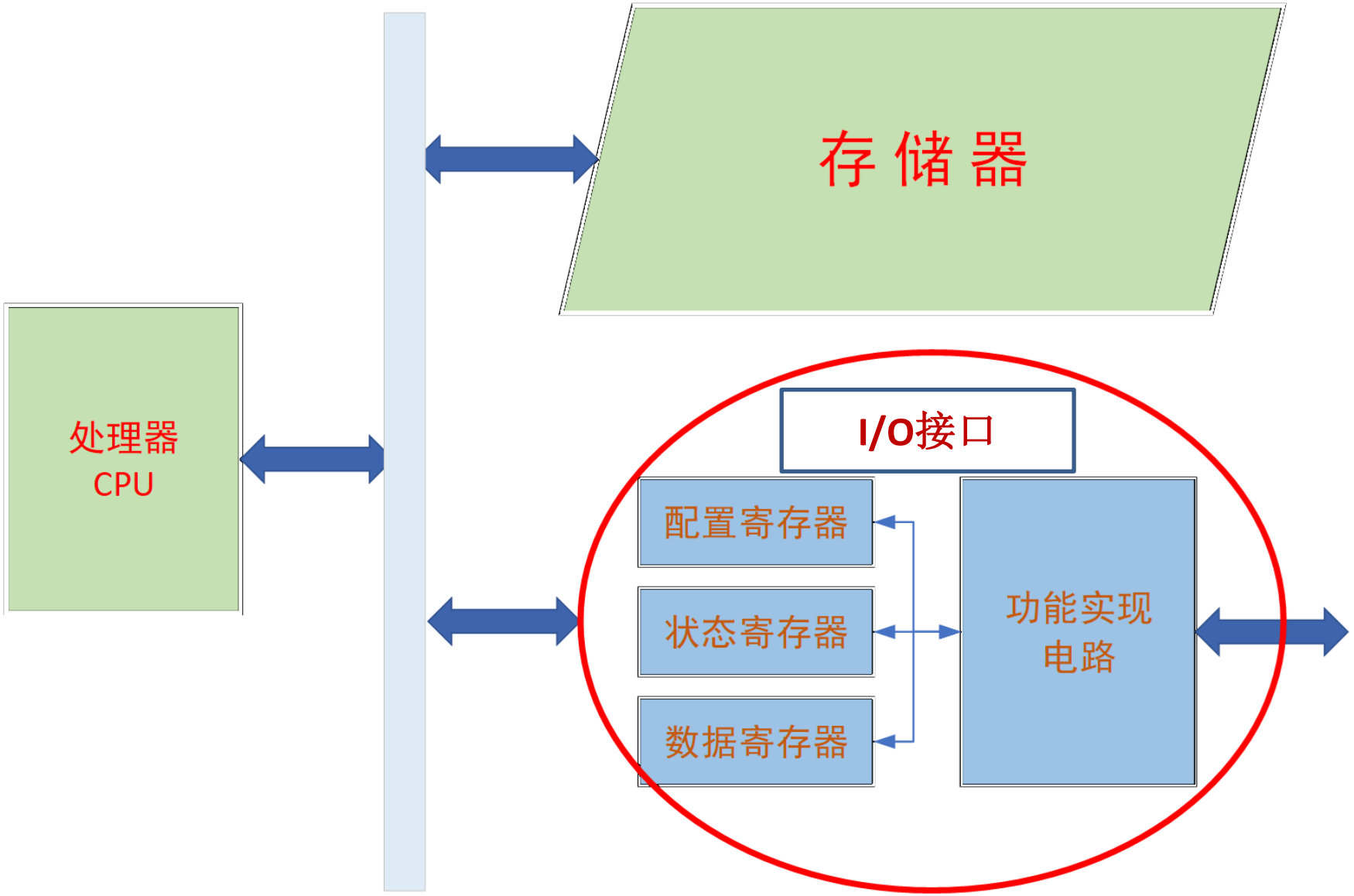
接口与端口定义：

在接口部件中都包含一组寄存器；CPU和外设交换信息时，将根据三种不同的信息，对不同的寄存器进行读写；

这些寄存器就称为端口（PORT）；一个外设往往需要几个端口地址；不同的端口中存放不同的信息；

端口主要有三类，状态口，命令口和数据口

CPU需要访问外设时，寻址的是端口，而不是笼统的外设；


存 储 器
处理器
CPU
I/O接口
配置寄存器
状态寄存器
数据寄存器
功能实现
电路

GPIO：

GPIO引脚的工作模式：

(1) 输入模式（上拉 / 下拉 / 浮空）

(2) 输出模式（推挽 / 开漏、上拉 / 下拉 / 浮空）

(3) 复用功能（推挽 / 开湍、上拉 / 下拉 / 浮空）

(4) 模拟输入输出模式

Timer 7种工作模式：

1. 定时中断
2. 输入捕获
3. 输出比较
4. 脉冲宽度调制（PWM）输出
5. 脉冲宽度调制（PWM）输入
6. 单脉冲OPM模式
7. 强制输出模式

A / D转换器的主要参数：

(1) 转换精度：ADC 的实际输出接近理想输出的精确程度。通常用数字量的最低有效位(LSB) 来表示。

(2) 转换率：完成一次A/D 转换所需要的时间的倒数。

例如完成一次A/D转换需要的时间是100ns，那么转换率为10MHz，有时也标为10MSPS，即每秒转换1000 万次。

(3) 分辨率：

表明了能够分辨最小的量化信号的能力，通常用位数来表示。对于一个实现N位二进制转换的ADC来说，它能分辨的最小量化信号的能力为2^N位，所以它的分辨率为2^N。

数模转换器和模数转换器

ADC工作模式：

1) 单次转换模式

ADC 执行一次转换。

2) 连续转换模式

ADC 结束一个转换后立即启动一个新的转换。

3) 模拟看门狗

如果ADC转换的模拟电压低于阈值下限或高于阈值上限,则模拟看门狗状态位会置1。

4) 扫描模式

ADC可以在此模式下扫描一组选中的模拟通道。为组中的每个通道都执行一次转换。每次转换结束后，会自动转换该组中的下一个通道。

5) 不连续采样模式

该模式可用于转换含有n个转换的短序列，该短序列是上述扫描模式中选择的转换序列的一部分。出现外部触发时，将启动接下来n个转换，直到序列中的所有转换均完成为止。

谢谢!