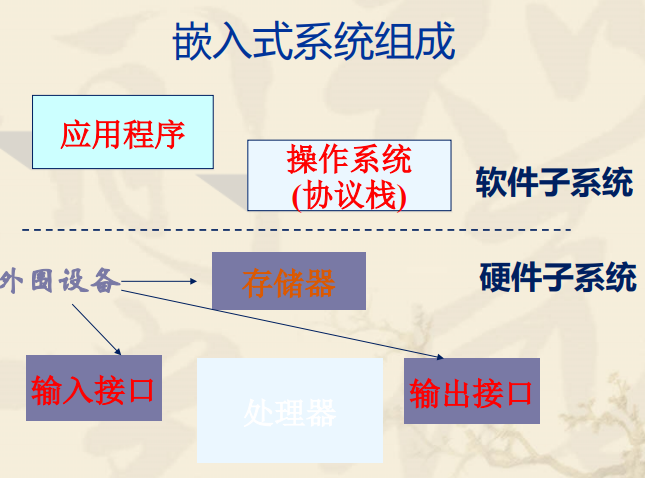
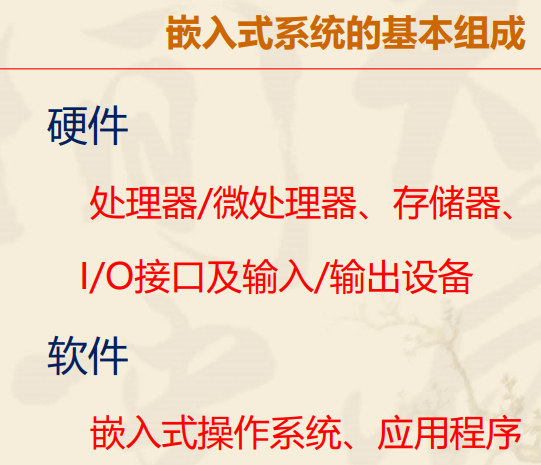
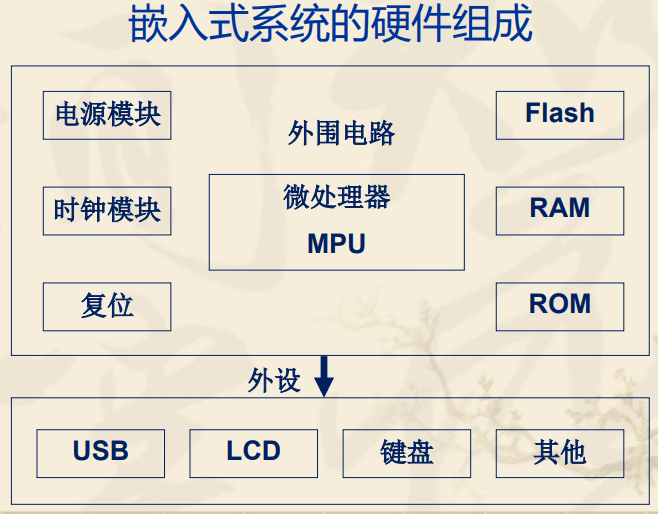
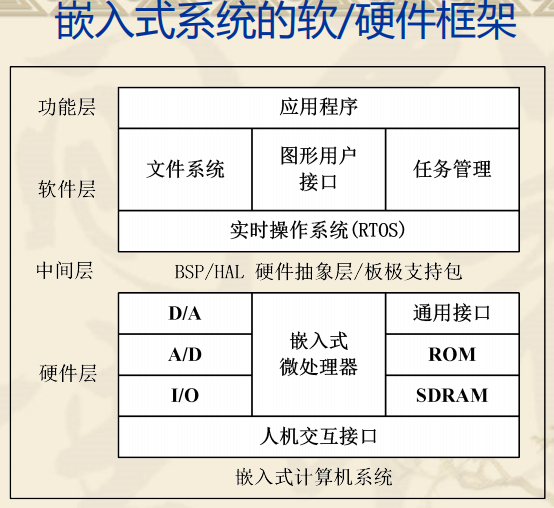
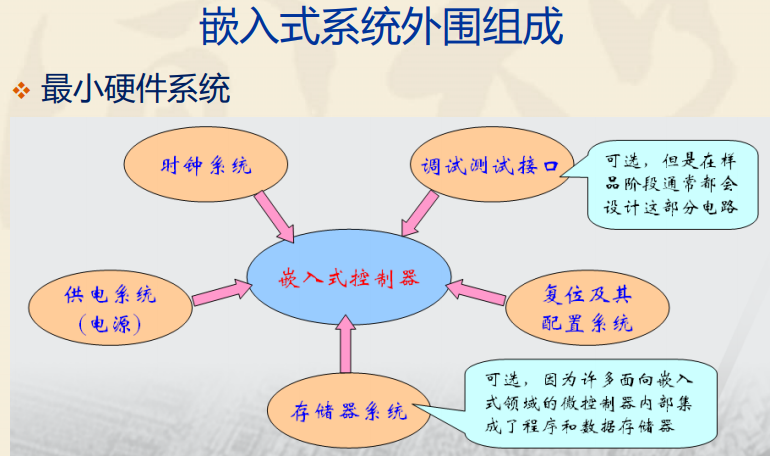
**第一章**

**（1）嵌入式系统基本组成**







**(2)嵌入式系统中，微处理器控制I/O端口或部件的数据传送方式，外部通讯两种方式，结合串口工作方式**

答：2种,中断方式和程序查询方式

程序查询方式是由微处理器周期性地执行一段查询程序来读取 I/O 端口或部件

中状态寄存器的内容，并判断其状态，从而使微处理器与 I/O 端口或部件在进行

数据、命令传送时保持同步。程序查询方式下效率非常低，因为微处理器要花费

大量时间测试 I/O 端口或部件的状态。并且 I/O 端口或部件的数据也不能得到实

时地处理。

中断方式是 I/O 端口或部件在完成了一个 I/O 操作后，产生一个信号给微处理

器，这个信号叫做“中断请求”，微处理器响应这个请求信号，停止其当前的程

序操作，而转向对该 I/O 端口或部件进行新的读/写操作，特点是实时性能好。

**(3)什么是、为何采用交叉开发 概念，涉及两纲**

交叉编译是指在一个平台上编译生成在另一个平台上运行的可执行程序，因为目标平台不具备直接运行开发环境的条件，嵌入式系统应用软件一般在宿主机上开发，在目标机上运行，因此需要交叉开发环境

交叉开发是指在通用电脑上把程序编写、编译、调试好，再下载到嵌入式产品中去运行。

**为何采用**：嵌入式系统是计算机专用的系统。由于嵌入式系统硬件上的特殊性，

一般不能安装发行版的 Linux 系统。例如 flash 储存空间很小，没有足够的空间

安装，或者处理器很特殊，没有发行版的 linux 可用。所以需要专门为特定的目

标版指定 linux 操作系统，这必然需要相应的开发环境，于是人们想到了交叉开

发模式。

简略版：由于嵌入式系统资源匮乏，一般不能像 PC 一样安装本地编译器和调试

器，不能在本地编写、编译和调试自身运行的程序，而需借助其它系统如 PC 来

完成这些工作。

**(4)嵌入式硬件设计几个阶段及主要目标**



**硬件设计阶段：**

1. **系统方案分析与设计（处理器和外围器件的选择）**

根据系统所要完成的功能，选择合适的处理器和 外围器件，完成系统的功能框图设计

1. **PCB设计与仿真**

需在EDA仿真设计平台下，设计系统原理图及PCB，并对 PCB板上的信号完整性、EMI等进行仿真，根据仿真结果来对PCB进行合理的布局布线调整，完成PCB的设计

1. **PCB的加工与测试**

对加工完成的PCB进行器件焊接、调试和测试， 完成整个系统硬件的设计

**第二章**

**（1）arm微处理器支持的数据类型，多少位的**

**ARM处理器支持下列数据类型：**

字节 8位， 半字16位（必须分配为占用两个字节）， 字32位（必须分配为占用4字节）

§ 所有数据操作，例如 ADD，都以字为单位；

§ 装载和保存指令可以对字节、半字和字进行操作，当装载字节或半字时自动

实现零扩展或符号扩展；

§ ARM 指令的长度刚好是 1 个字（分配为占用 4 个字节），Thumb 指令的长度刚

好是半字（占用 2 个字节）

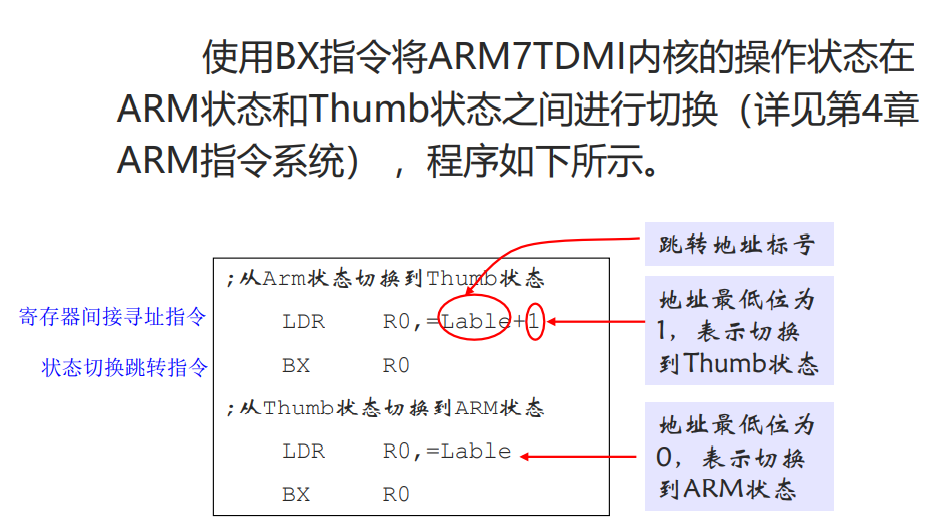
**（2）处理器状态切换 到底几种：**

**有两种操作状态：**

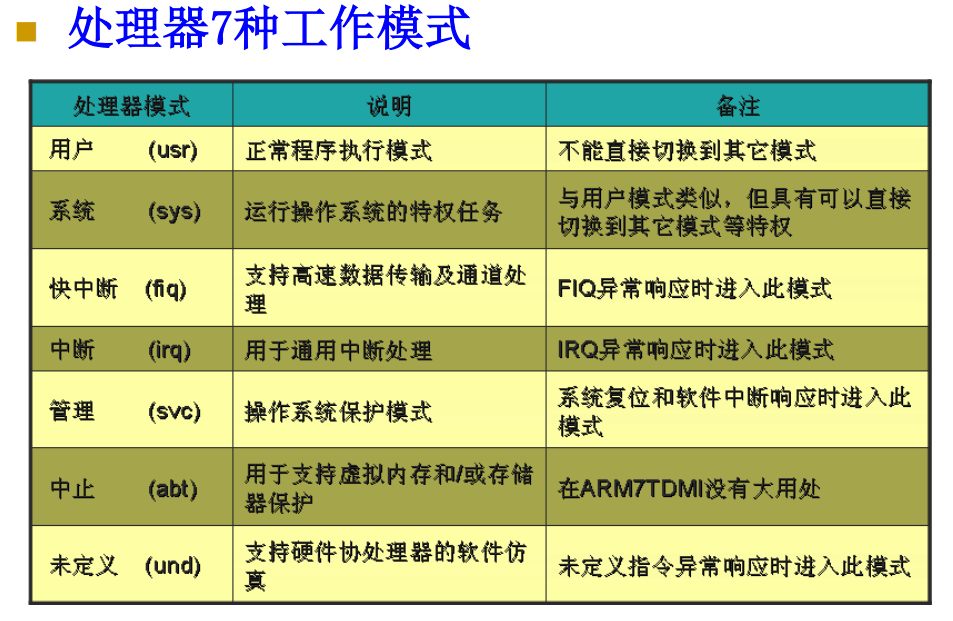
**ARM状态：**32位，这种状态下执行的是字方式的ARM指令；

**Thumb状态：**16位，这种状态下执行半字方式的ARM指令。

**注意：两个状态之间的切换并不影响处理器模式或寄存器内容。**



**（3）工作模式7种：（填空）**

****

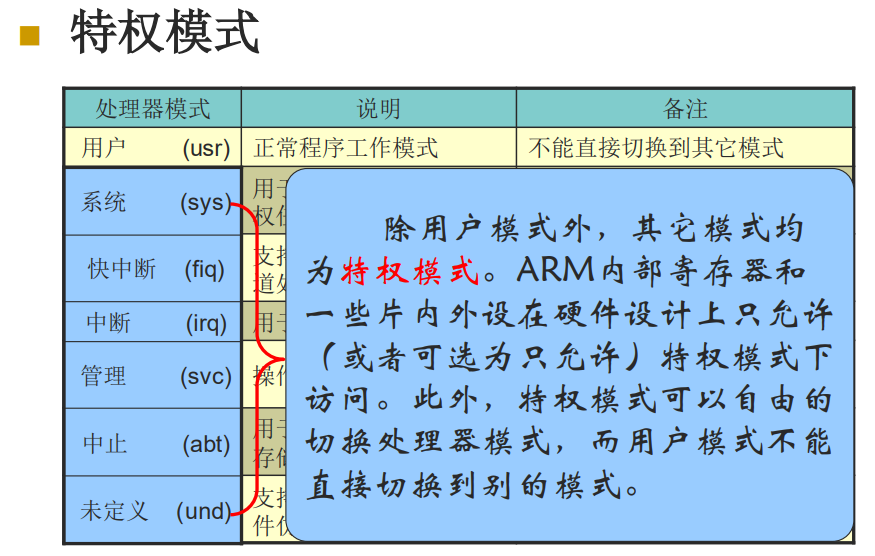
**用户模式：**

**（4）特权模式：**

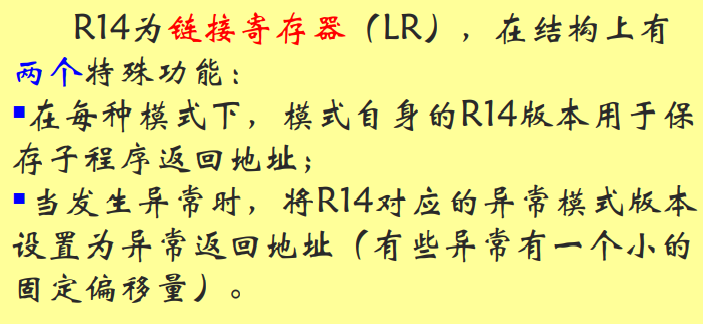
除用户模式外，其他模式均为特权模式。ARM 内部寄存器和一些片

内外设在硬件设计上只允许特权模式下访问。特权模式可自由切换处理器模

式，用户模式不能直接切换到别的模式。

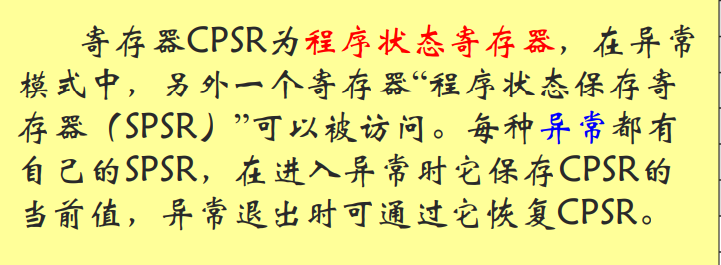


**（5）LR寄存器的作用：**

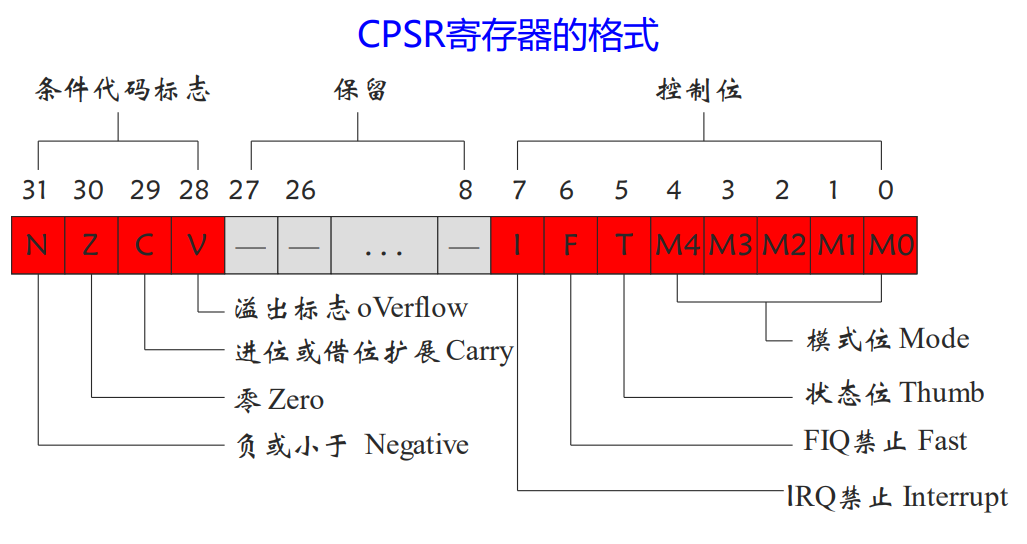


**（6）cpsr和spsr寄存器是什么，全称**

ARM7TDMI内核包含1个CPSR和5个供异常处理程序使用 的SPSR



CPSR反映了当前处理器的状态，其包含： 4个条件代码标志（负(N)、零(Z)、进位(C)和溢出(V) ）； 2个中断禁止位，分别控制一种类型的中断； 5个对当前处理器模式进行编码的位； 1个用于指示当前执行指令(ARM还是Thumb)的位。



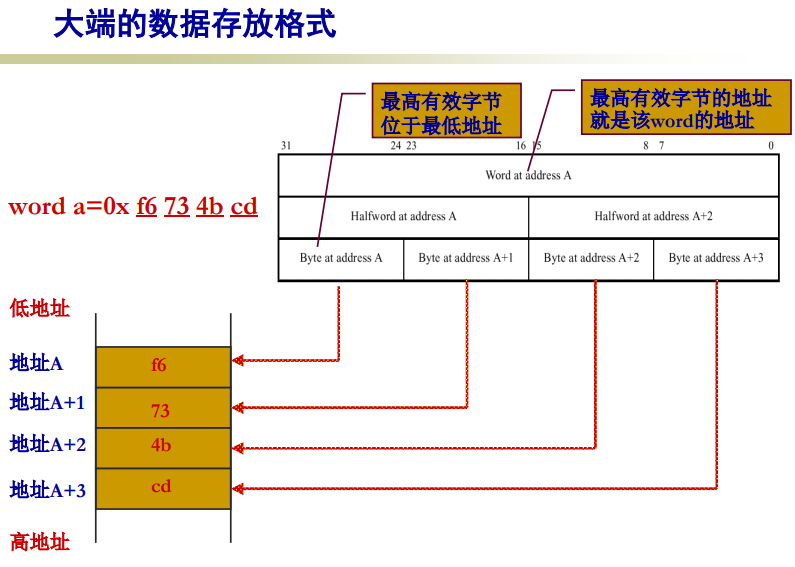
每个异常模式还带有一个程序状态保存寄存器 （SPSR），它用于保存在异常发生之前的CPSR。 CPSR和SPSR通过特殊指令（MRS、MSR）进行 访问。

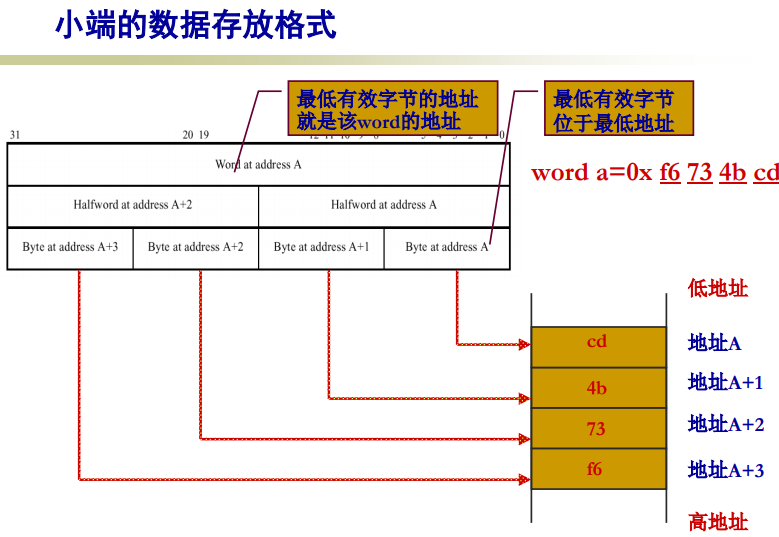
**(7)大端格式和小端格式 存储时：都按字节存储，计算或者存储**

ARM7TDMI处理器可以将存储器中的字以下列格式存储：

大端格式（Big-endian） 小端格式（Little-endian），默认小端







**如果实际的存储器格式与芯片的存储器格式 不符时，只有以字为单位的数据存取才正确，否则 将出现不可预期的结果。**

**(8)异常进入和退出异常 了解一下**

**异常：**是由内部或者外部原因引起的，当异常发生时CPU将暂停执行当前指令，自动到指定的向量地址读取指令并且执行。

**进入异常** ：在异常发生后，ARM7TDMI内核会作以下工作：

1、将CPSR复制到适当的SPSR中；

2、将CPSR模式位强制设置为与异常类型相对应的值；

3、强制PC从相关的异常向量处取指。

**退出异常：**当异常结束时，异常处理程序必须：

1、将LR（ R14）中的值减去偏移量后存入PC，偏移量根据异常的类型而有所不同；

2、将SPSR的值复制回CPSR；

3、清零在入口置位的中断禁止标志。

注：恢复CPSR的动作会将T、F和I位自动恢复为异常发生前的值。

**（9）复位的操作**

当nRESET信号再次变为高电平时，ARM处理器执行 下列操作：

1、强制CPSR中的M[4:0]变为b10011（管理模式）；

2、置位CPSR中的I和F位；

3、清零CPSR中的T位；

4、强制PC从地址0x00开始对下一条指令进行取指；

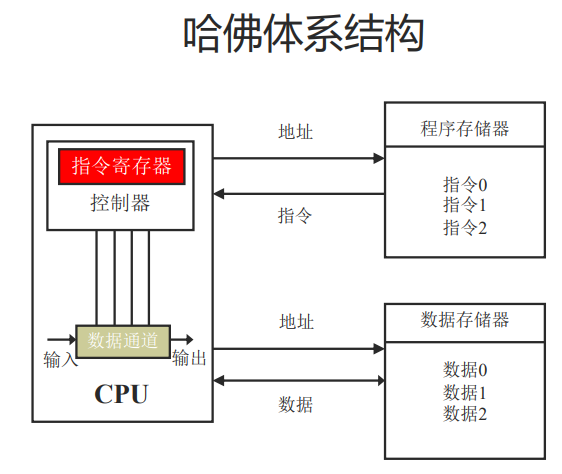
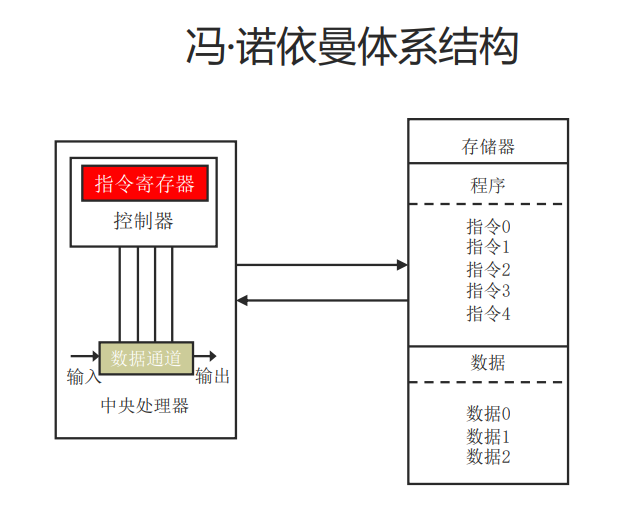
5、返回到ARM状态并恢复执行

**第三章**

**CISC和RISC的全称，区别**



**冯诺依曼和哈佛结构体系**



1. **流水线是什么，作用是什么，要总结出来，是为了提高什么**

流水线(Pipeline)技术：几个指令可以并行执行

指令流水线：是为提高处理器执行指令的效率，把一条指令的操作分成多个细小的步骤，每个步骤由专门的电路完成的方式。

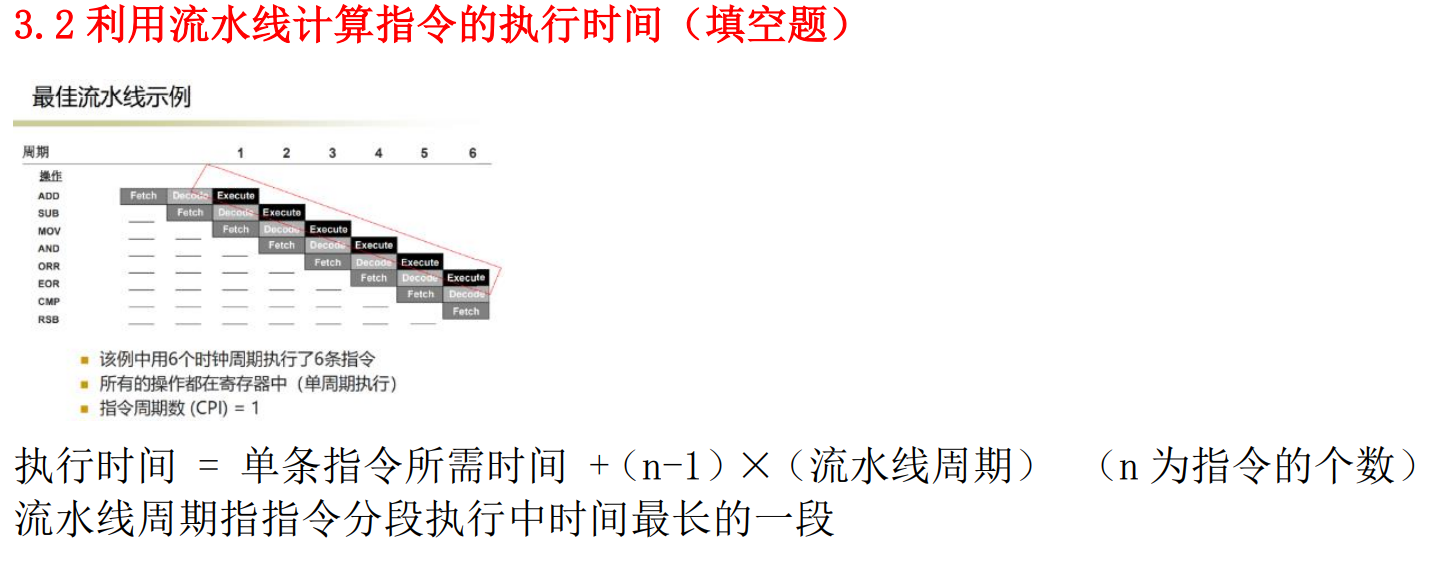
流水线的执行顺序：取指令->译码->执行

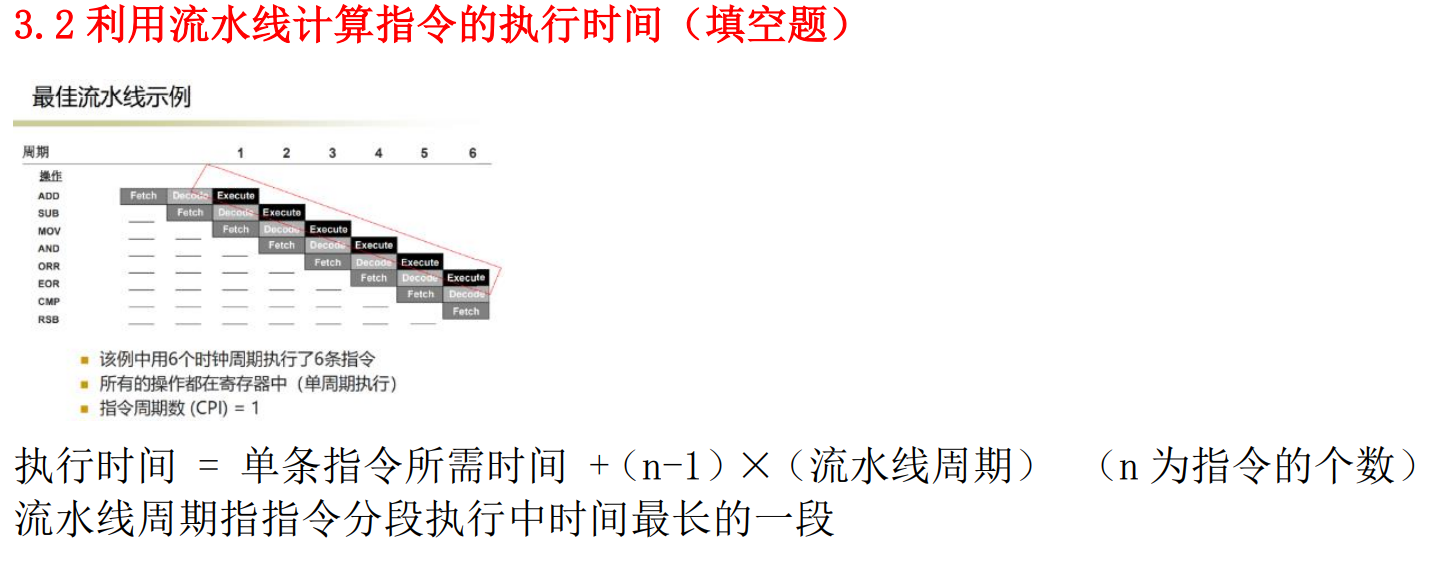
作用：1，提高了CPU的运行效率 ，2内部信息流要求通畅流动



1. **利用流水线计算标准指令执行所需要的时间，填空题**

**指令周期数（CPI）=时钟周期数/指令数**





1. **并行串行，基本概念了解是什么**

并行通信：将数据的各位用多条数据线同时进行传送，外加地址线和通信控制 线。优点是传输速率高，缺点是长距离传输成本高，可靠性差，只适用于近距离 传输。 串行通信：将数据逐位按顺序在一条传输线上传送。优点是传输线少，长距离传 送时成本低，缺点是传输速率低。

1. **同步异步基本概念**

根据数据传输方式的不同，可将串行通信分为同步通信和异步通信。

同步传输时发送方发送时钟，接收方的时钟和发送方是同一个时钟；

异步传输时发送方不需发送时钟，并不要求接收方的时钟与发送方完全一样。

1. **I2C总线，SPI总线物理组成上的特点**

**IIC：**一种用于内部 IC 控制的简单的双向两线串行总线，最高速率 100Kbps，25 英尺，最多可支持 40 个设备。

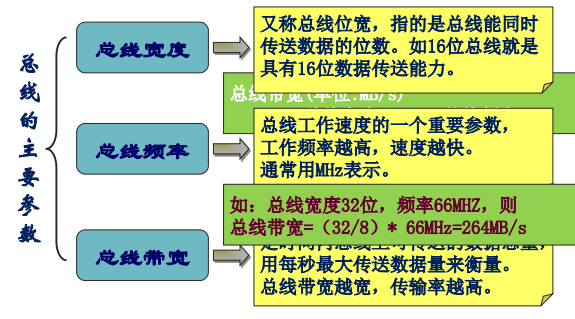
**SPI：**串行外设接口(Serial Peripheral Interface)，是一种高速的，全双工， 同步的通信总线。

**异同：**（SDA 数据线,SCL 时钟线） IIC 总线不是全双工（半双工），2 根线 SCL、SDA。SPI 总线实现全双工，4 根线。 IIC 总线是多主机总线，通过 SDA 上的地址信息来锁定从设备。SPI 总线只有一 个主设备，主设备通过 CS 片选来确定从设备 IIC 总线传输速度在 100kbps-4Mbps。SPI 总线传输速度更快，可以达到 30Mbps 以上。 IIC 总线和 SPI 总线时钟都是由主设备产生，并且只在数据传输时发出时钟 IIC 总线读写时序比较固定统一，SPI 不同从设备读写时序差别比较大

按照冯·诺依曼体系结构思想，计算机的硬件是由 CPU、存储器和I/O设备三部分组成的。

总线是把 CPU与存储器、I/O设备相连接的信息通道，但总线并不仅仅指的是一束信号线，而应包含相应的通信协议。

按照使用场合的不同，总线分成：芯片级总线 （CPU总线）、板卡级总线（内总线）和系统级 总线（外总线）。



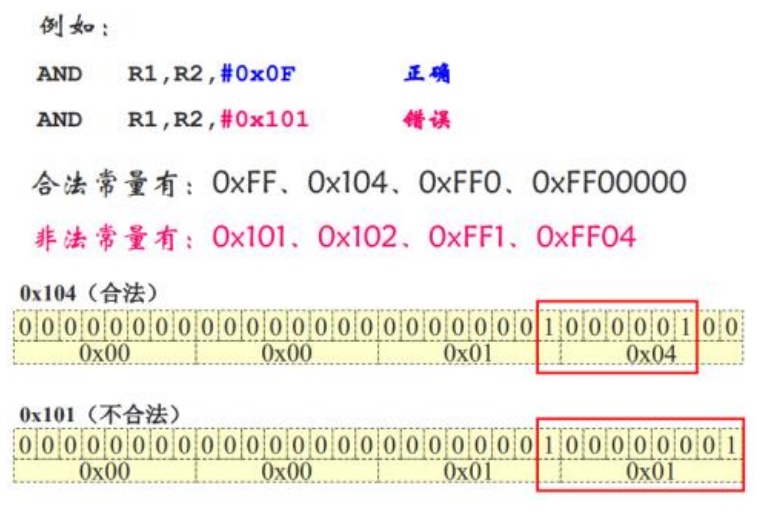
**第四章 重点**

**（1）第二操作数，遵循的原则，分析原因**

在算术指令中，可以将第二操作数在运算之前移位（如：LDREQ，R0,[R1,R2,LSR #16]!）。移位通过组合电路完成，而不需时钟脉冲的作用，不影响指令执行时间。 每条指令由：操作码域、条件码域、条件码设置域、目标操作数、第一操作数和 第二操作数组成。



**1.#immed\_8r（常数表达式）遵循的原则：**该常数必须能对应 8 位位图，即一个 8 位的常数通过循环右移偶数位得到，而不是一个任意数。（12 位）



**原因：**指令编码时，第二操作数要用 12 位来表示 32 位数，因此采用 8 位存放数 据，4 位存放位移量的方式进行编码。 2.寄存器方式下，即为寄存器的数值。 3.寄存器移位方式，将寄存器移位的结果作为操作数，Rm 的值保持不变。

**（2）常用指令的理解，：存储器访问指令ldr，str使用过程中特别是基址寻址指令是怎么做的，寄存器里值是怎么变化的，基址寻址偏移量**

**（3）and逻辑运算指令，其他的不管**

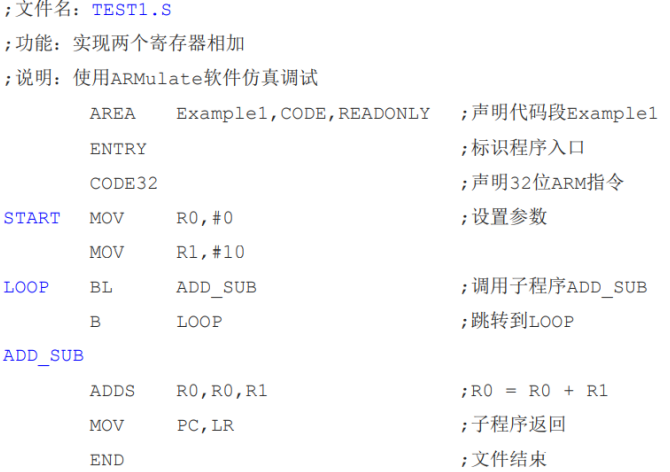
**（4）简单的arm汇编指令编写**

**（5）汇编程序基本格式 如程序顶头不顶头**

**（6）数据传送和处理指令：mov传送 add加 sub减**

**（7）分支转移指令 b bl**

**（由上可猜出课上源程序）**

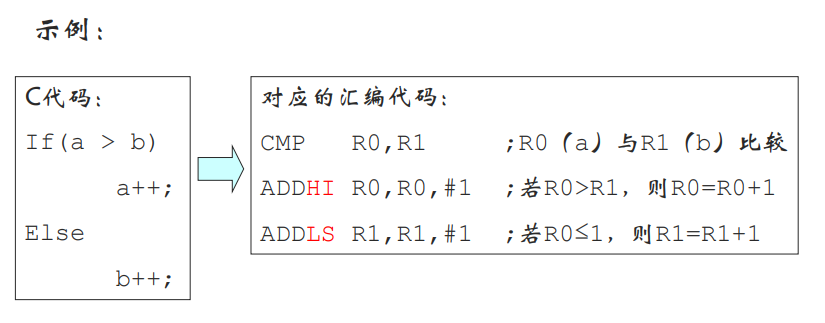


**（8）条件码的使用 ne Hi ls**









**（9）带s的指令**

条件代码标志大多数“数值处理指令”可以选择是否影响条件代码标志位。通常如果指令带S后缀，则该指令的执行会影响条件代码标志；但有一些指令的执行总是会影响 条件代码标志。 N、 Z、 C和V位都是条件代码标志。算术操作、逻 辑操作、MSR或者LDM指令可以对这些位进行设置。 所有ARM指令都可按条件来执行，而Thumb指令中只 有分支指令可按条件执行。

各标志位的含义如下：

**N** 运算结果的最高位反映在该标志位。对于有符 号二进制补码，结果为负数时N=1，结果为正数或 零时N=0；

**Z** 指令结果为0时Z=1（通常表示比较结果“相 等”），否则Z=0；

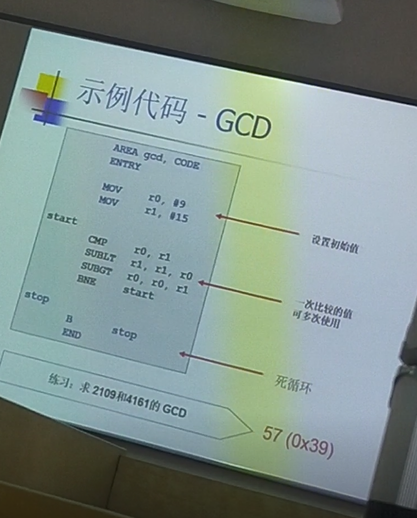
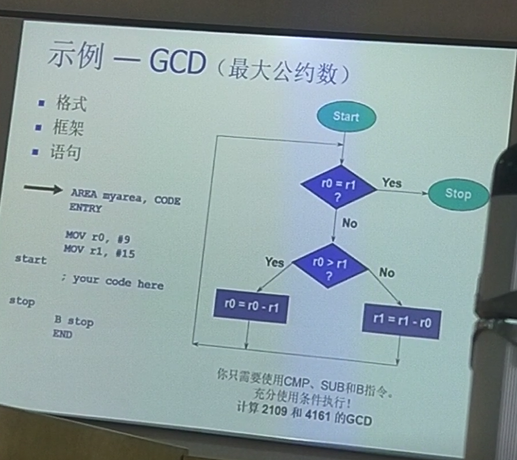
**C** 当进行加法运算(包括CMN指令)，并且最高位产生 进位时C=1，否则C=0。当进行减法运算(包括CMP 指 令)，并且最高位产生借位时C=0，否则C=1。对于结合 移位操作的非加法/减法指令，C为从最高位最后移出的 值，其它指令C通常不变；

**V** 当进行加法/减法运算，并且发生有符号溢出时V=1， 否则V=0，其它指令V通常不变。

**（10）注释，标号的使用**

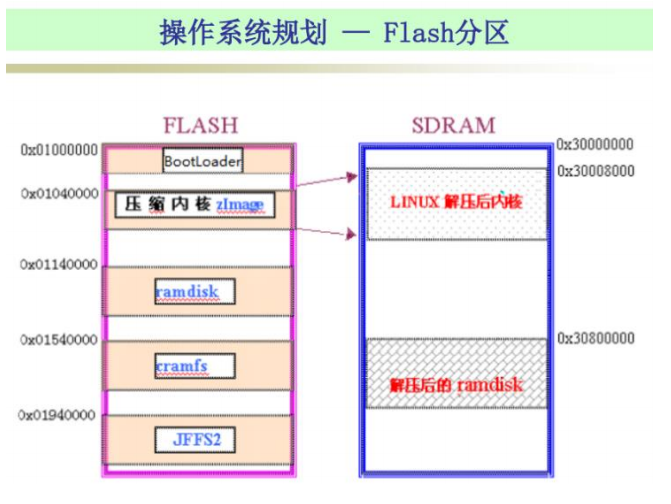
**（11）杂项指示符 area entry end**

**（12）汇编程序，框架必须写，顶头不顶头，不是汇编调c ，不写扣分 最好要写注释，**

****

**第五章**

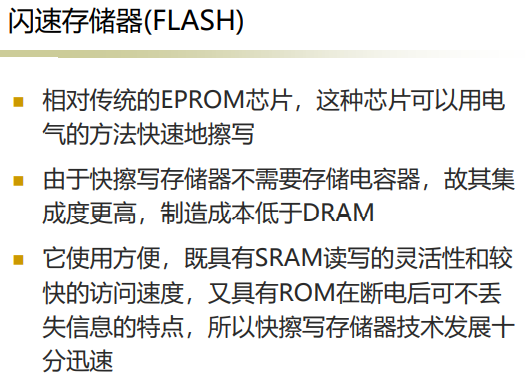
**（1）linux操作系统如何规划，bootload、、、flash分区**



Uboot 分区、kernel 分区、rootfs 分区，后者需要前者提供功能支持，前者先启 动，后者再启动。 Bootloader 在运行过程中虽具有初始化系统和执行用户输入的命令等作用，位于 Flash 最前端。但它最根本的功能就是为了启动 Linux 内核 RootFileSystem 是 Linux 系统的核心组成部分，它可以作为 Linux 系统中文件和数据的存储区域，通常它还包括系统配置文件和运行应用软件所需要的库。

**(2)有个图，flash从技术角度分为几种**

**2种，NORflash和NANDflash**





NandFlash强调降低每比特的成本，更高的性 能，并且像磁盘一样可以通过接口轻松升级。NandFlash具有容量大、回写速度快、芯片面 积小等特点，主要用于外存。

NOR Flash具有随机存储速度快、电压低、功 耗低、稳定性高等特点，主要用于主存。

**嵌入式系统启动时，引导代码、操作系统的运行和应用程序的加载的两种架构（跟flash有关），启动方式有几种**

嵌入式系统在启动时，引导代码、操作系统的运行和应用程序的加载主要有两种架构，一种是直接从 Nor Flash 启动的架构，另一种是直接从 Nand Flash 启动的架构。

**从 Nor Flash 启动**:

Nor Flash 具有芯片内执行（XIP eXecute In Place）的特点，在嵌入式系统中 常做为存放启动代码的首选。从 Nor Flash 启动的架构又可细分为只使用 Nor Flash 的启动架构和 Nor Flash 与 Nand Flash 配合使用的启动架构。 单独使用 Nor Flash：在该架构中，引导代码、操作系统和应用代码共存于同一 块 Nor Flash 中。系统上电后，引导代码首先在 Nor Flash 中执行，然后把操作 系统和应用代码加载到速度更高的 SDRAM 中运行。另一种可行的架构是，在 Nor Flash 中执行引导代码和操作系统，而只将应用代码加载到 SDRAM 中执行。

**从 Nand Flash 启动:** S3C2410 支持从 Nand Flash 启动的模式，它的工作原理是将 NandFlash 中存储 的前 4KB 代码装入一个称为 Steppingstone（BootSRAM）的地址中，然后开始执 行该段引导代码，从而完成对操作系统和应用程序的加载。 Nor Flash 和 Nand Flash 配合使用 Nor Flash（2M 或 4M）中存放启动代码和操作系统，而 Nand Flash 中存放应用 代码，根据存放的应用代码量的大小可以对 Nand Flash 容量做出相应的改变。 系统上电后，引导代码直接在 Nor Flash 中执行，把 Nand Flash 中的操作系统 和应用代码加载到速度更高的 SDRAM 中执行。也可以在 Nor Flash 中执行引导代 码和操作系统，而只将 Nand Flash 中的应用代码加载到 SDRAM 中执行。该架构 是当前嵌入式系统中运用广泛的启动架构之一。

**第六章**

**（1）linux操作系统中，设备分为哪几类**

3 类：字符设备、块设备、网络设备



**（2）驱动程序的功能是什么，与应用程序的关系是什么**

**功能：**从驱动开发人员看，**驱动程序是直接操控硬件的软件**，主要完成以下功能：

初始化和释放设备；

直接读写硬件寄存器来控制硬件；

实现内核与硬件之间的数据交换；

操作设备缓冲区；

操作输入、输出设备，如键盘、打印机等；

实现应用程序与设备之间的数据交换；

检测和处理设备出现的错误。

**关系：**从应用程序看，**驱动程序应为应用程序提供访问硬件设备的编程接口**

主要提供以下功能：

应用程序通过驱动程序安全有效地访问硬件；

驱动程序隐藏底层细节，从而提高应用软件的可移植性和可复用性；

驱动程序文件节点可方便地提供访问权限控制。

**（3）linux内核主要组成 5个大的组成**

内核主要由五大块组成：**进程调度、内存管理、虚拟文件系统、网络接口、进程间通信。**

1、内存管理：主要完成的是如何合理有效地管理整个系统的物理内存，同时快 速响应内核各个子系统对内存分配的请求。Linux 内存管理支持虚拟内存，而多 余出的这部分内存就是通过磁盘申请得到的，平时系统只把当前运行的程序块保 留在内存中，其他程序块则保留在磁盘中。在内存紧缺时，内存管理负责在磁盘 和内存间交换程序块。

2、进程管理：主要控制系统进程对 CPU 的访问。当需要某个进程运行时，由进 程调度器根据基于优先级的调度算法启动新的进程。：Linux 支持多任务运行， 那么在一个单 CPU 上支持多任务就是由进程调度管理来实现的。

3、进程间通信：主要用于控制不同进程之间在用户空间的同步、数据共享和交换。由于不用的用户进程拥有不同的进程空间，因此进程间的通信要借助于内核 的中转来实现。 一般情况下，当一个进程等待硬件操作完成时，会被挂起。当硬件操作完成，进 程被恢复执行，而协调这个过程的就是进程间的通信机制。

4、虚拟文件系统：用一个通用的文件模型表示了各种不同的文件系统，这个文 件模型屏蔽了很多具体文件系统的差异，使 Linux 内核支持很多不同的文件系统。这个文件系统可以分为逻辑文件系统和设备驱动程序：逻辑文件系统指 Linux 所支持的文件系统，设备驱动程序指为每一种硬件控制器所编写的设备驱 动程序模块。

5、网络接口 网络接口提供了对各种网络标准的实现和各种网络硬件的支持。网络接口一般分 为网络协议和网络驱动程序。网络协议部分负责实现每一种可能的网络传输协议。 网络设备驱动程序则主要负责与硬件设备进行通信，每一种可能的网络硬件设备 都有相应的设备驱动程序。

**（4）为什么将linux作为嵌入式操作系统，它的优势是什么，简答题简单答**

Linux 做嵌入式的优势：

1，Linux 是开放源代码的，不存在黑箱技术，遍布全球的众多 Linux 爱好者又是 Linux 开发者的强大技术支持；

2，Linux 的内核小、效率高，内核的更新速度很快,Linux 是可以定制的，其系统内核最小只有约 134KB。

3，Linux 是免费的 OS，在价格上极具竞争力。 Linux 还有着嵌入式操作系统所需要的很多特色，突出的就是 Linux 适应于多种 CPU 和多种硬件平台，是一个跨平台的系统。

**（5）什么是交叉开发、交叉编译？交叉编译的两个组成部分（嵌入式系统，PC机）书名是？**

在一种计算机环境中运行的编译程序，能编译出在另外一种环境下运行的代码， 我们就称这种编译器支持交叉编译。这个编译过程就叫交叉编译。简单地说，就 是在一个平台上生成另一个平台上的可执行代码，同一个体系结构可以运行不同 的操作系统，同样，同一个操作系统也可以在不同的体系结构上运行。这里需要 注意的是所谓平台，实际上包含两个概念：体系结构（Architecture）、操作系统（OperatingSystem）。同一个体系结构可以运行不同的操作系统；同样，同一 个操作系统也可以在不同的体系结构上运行。

**交叉开发：** 首先在通用计算机上编写程序，然后通过交叉编译生成目标平台上可以运行的二 进制代码格式，最后再下载到目标平台上的特定位置上运行。

**交叉编译：** 在宿主机平台上使用某种特定的交叉编译器，为某种与宿主机不同平台的目标系 统编译程序，得到的程序在目标系统上运行而非在宿主机本地运行。

**组成部分：** 宿主机（Host）是一台通用计算机（如 PC 机或者工作站），它通过串口或者以太网接口与目标机通信。 目标机（Target）一般在嵌入式应用软件开发和调试期间使用，用来区别与嵌入 式系统通信的宿主机。

**第七章**

**（1）进程，线程有何区别**

**进程**是可并发执行的程序在一个数据集合的运行过程。它是操作系统动态执行的基 本单元，是系统进行资源分配的独立单位。

**线程**是进程的一个实体,是CPU 调度和分派的基本单位,它是比进程更小的能独立运行的基本单位。

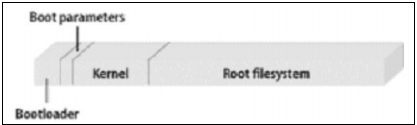
**区别**：线程自己基本上不拥有系统资源,只拥有一点在运行中必不可少的资源(如程序计数器,一组寄存器和栈),但是它可与同属一个进程的其他的线程共享进程 所拥有的全部资源。一个线程可以创建和撤销另一个线程；同一个进程中的多个 线程之间可以并发执行。 进程与线程的关系：通常在一个进程中可以包含若干个线程，它们可以利用进程所拥有的资源。在引入线程的操作系统中，通常都是把进程作为分配资源的基本单位，而把线程作为独立运行和独立调度的基本单位。由于线程比进程更小，基本上不拥有系统资源，故对它的调度所付出的开销就会小得多，能更高效的提高 系统内多个程序间并发执行的程度。

**（2）多线程与忙等待，什么情况下出现忙等待，该怎么做**

线程是程序中一个单一的顺序控制流程。在单个程序中同时运行多个线程完成 不同的工作，称为**多线程。** **忙等待：**处理线程同步时，需要等待某种资源，线程可以在一个循环中不断检查， 直到条件满足退出循环。可以采用阻塞等待的方式进行处理，这样进程在等待被唤醒的过程中不会占用 CPU 内存。

**（3）BootLoader及其工作的两个阶段，第一阶段怎么做，第二阶段怎么做**

简单地说，Boot Loader 就是在操作系统内核运行之前运行的一段小程序。通 过这段小程序，我们可以初始化硬件设备、建立内存空间的映射图，从而将系统 的软硬件环境带到一个合适的状态，以便为最终调用操作系统内核准备好正确的 环境。 不同的目标板需要不同的 BootLoader 支持。

****

**两个阶段：** 依赖于 CPU 体系结构的代码，比如设备初始化代码等，通常都放在阶段 1 中，而 且通常都用汇编语言来实现，以达到短小精悍的目的。而阶段 2 则通常用 C 语 言来实现，这样可以实现一些复杂的功能，而且代码会具有更好的可读性和可移 植性。

**Boot Loader 的阶段 1 通常包括以下步骤：**

①硬件设备初始化。为加载 Boot Loader 的阶段 2 准备 RAM 空间。

②拷贝 Boot Loader 的阶段 2 到 RAM 空间中。

③设置好堆栈。跳转到阶段 2 的 C 入口点。

**Boot Loader 的阶段 2 通常包括以下步骤：**

①初始化本阶段要使用到的硬件设备，串口。

②检测系统内存映射(memory map)。

③将 kernel 映像和根文件系统映像从 Flash 读到 RAM 空间中。

④为内核设置启动参数。

⑤调用内核。

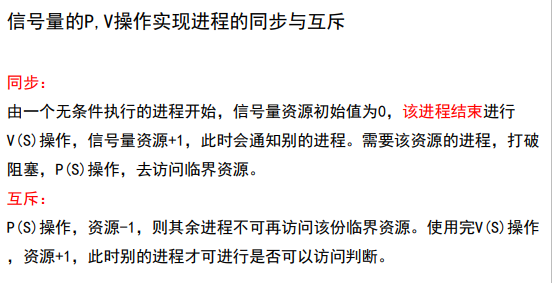
**实验：汇编程序**

**驱动程序 ：十三个里面 最基本的那个，调一个就行，基本组建啥的**

**线程实验 ：线程创建，结束，同步等，以互斥量、信号量为重点**

**（a卷必考）区分开进程，A卷线程，B卷进程**

**同步的概念方法 互斥量 信号量**



Linux进程之间的通信——信号量

**本质上，**信号量是一个计数器，它用来记录对某个资源（如共享内存） 的存取状况。

一般说来，为了获得共享资源，进程需要执行下列操作：

（1）测试控制该资源的信号量；

（2）若此信号量的值为正，则允许进行使用该资源，进程将进号量减1 ；

（3）若此信号量为0，则该资源目前不可用，进程进入睡眠状态，直至 信号量值大于0，进程被唤醒，转入步骤（1）；

（4）当进程不再使用一个信号量控制的资源时，信号量值加1，如果此 时有进程正在睡眠等待此信号量，则唤醒此进程。

**代码都是搭框架，四五行核心代码，只用汇编程序**