嵌入式复习指北

第四章 嵌入式系统软件设计

2.嵌入式软件的种类和每一类的特点是什么?

3.嵌入式软件的体系结构包括哪些部分？每部分的作用是什么?

4.嵌入式软件的运行流程一般分为几个阶段？每个阶段完成的主要工作是什么?

6.详细说明嵌入式操作系统的特点有哪些?

8.论述OpenHarmony操作系统的技术特点。

10.OpenHarmony内核的特点是什么?

11.简述LiteOS-m和LiteOS-a系统架构的组成，它们之间的区别是什么?

第五章 任务管理与调度

1.什么是任务？任务有哪些主要特性，主要包含哪些内容?

4.什么是抢占式调度和非抢占式调度?

5.任务主要包含哪些状态,就OpenHarmony LiteOS m任务状态之间的变迁情况进行描述。

6.什么是任务切换?任务切换通常在什么时候进行,说明任务切换的主要过程。

8.优先级位图算法的工作原理是什么?为什么可以提高实时内核的确定性?

9.什么是优先级反转?解决优先级反转有哪些主要方法?分别就这些方法进行描述

第六章 嵌入式操作系统

1.什么是中断？中断与自陷、异常之间有哪些联系与区别?

3.简述中断处理的基本过程，以及编写中断服务程序时需要注意的事项。

4.论述嵌入式实时系统时钟设备的组成和内核提供的时间管理功能。

7.嵌入式系统是如何实现对I/O系统管理的?

第七章 Boot loader

1.什么是Boot Loader？它的主要功能是什么?

2.Boot Loader有几种操作模式？它们分别以何种方式工作?

5.Boot Loader的工作过程一般来说分为哪两个阶段？每个阶段的核心工作是什么?

9.Hi3861的Boot分为几部分？每部分实现的功能是什么?

第八章 嵌入式系统设计

1.嵌入式系统的设计过程包括哪几个阶段？每一个阶段的主要工作有哪些

3.什么是软硬件协同设计方法?

5.嵌入式软件与一般软件的主要区别是什么?

9.论述嵌入式系统低功耗设计的方法

本文档为笔者对老师留的课后习题整理的答案
建议的使用方式：对着每个问题，在PPT/书中找到对应部分做参考。本文档仅仅是将重点提取了出来，PPT/书中介绍的更全面
本文档不保证100%正确、不保证100%覆盖所有知识点，**仅供参考**
请勿售卖该免费资料，谢谢。
--@jielahou

第一章 嵌入式系统概述

1.什么是嵌入式系统？嵌入式系统的特点是什么？

什么是嵌入式系统?

- 操作系统和功能软件集成在计算机硬件当中
- 带有**微处理器**的**专用**软硬件系统
- 使用**嵌入式处理器**构成系统，具有**自己的操作系统**、具有**某些特定功能**的系统
- 以应用为中心、以计算机技术为基础、软件硬件可裁剪

特点：嵌入性 专用性 综合性强 设计高效（程序小） 程序固化（程序需固化）独立开发系统（程序需要专门软件开发） 生命周期长　成本低、功耗低、可靠性高...

2.根据你的理解说明什么是嵌入式智能系统?

通过软硬件结合方式，对传统设备进行改造，让其拥有智能化功能

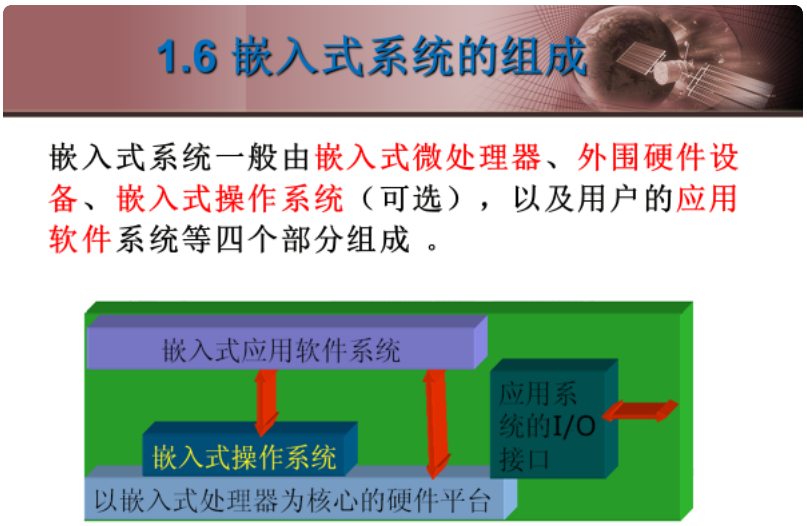
3.就你身边常使用的设备说明什么是智能硬件?

通过软硬件结合方式，对传统设备进行改造，让其拥有智能化功能

6.简述嵌入式系统的组成结构。

嵌入式系统由嵌入式计算机系统和执行机构组成

嵌入式计算机系统由嵌入式微处理器、外围硬件设备、嵌入式操作系统、用户的应用软件组成



8.嵌入式系统和通用计算机系统有何异同?

（嵌入式微处理器）实时性好、功耗低

（嵌入式操作系统）可靠性、可移植

需要专门的开发工具

9.列举几个生活中应用到嵌入式系统的例子，说明他们的组成和功能

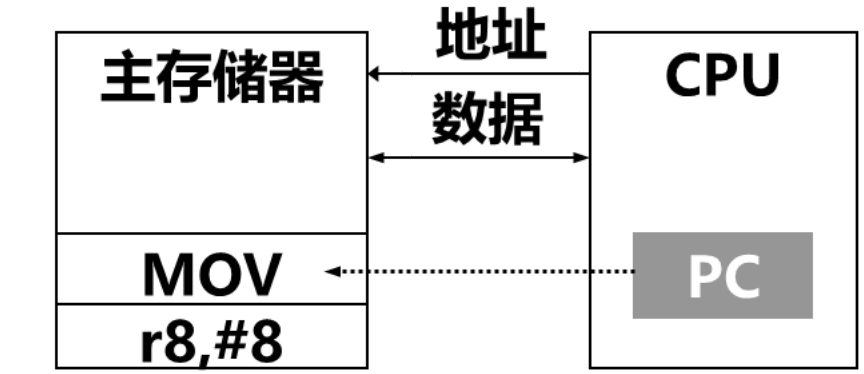
略

第二章 嵌入式微处理器

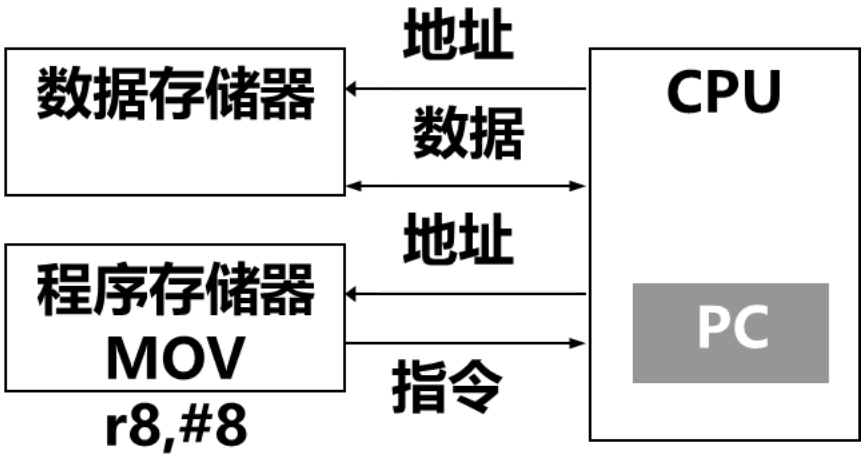
1.嵌入式微处理器常用的体系结构有几种，主要特点是什么?

- **冯·诺伊曼结构**
  - 代码和数据存储在同一存储器中
  - 代码和数据地址统一编址

    - 单一的地址和数据总线
- **哈佛结构**
    - 代码和数据存储在两个存储器中
    - 代码和数据分别编址
    - 两个地址和数据总线



冯·诺依曼结构



哈佛结构

3.与CISC对比，RISC的主要特点是什么?

- **指令长度**：RISC采用固定长度的指令格式
- **指令种类**：RISC指令种类更少
- **指令执行时间**：RISC单条指令执行时间更短
- **访存模式**：RISC访问存储器需使用Load/Store等专门指令

- **寻址方式**：RISC寻址方式更少

4.与PC领域通用微处理器对比，嵌入式微处理器的特点是什么?

- 功耗低
- 实时性强
- 外围IO丰富

5.简述嵌入式微处理器的分类及各自的主要特点。

- 嵌入式微处理器
  - 由通用CPU发展而来
  - 在功耗、可靠性等方面做了增强
- 嵌入式微控制器
  - 即单片机
  - 将整个计算机系统集成到一块芯片中
- 数字信号处理器(DSP)
  - 高效乘累加计算
  - 高效数据存取
- 片上系统
  - 将很多功能模块集成到单个芯片上

7.简述ARM系列嵌入式微处理器的特点。

- 支持双指令集
  - ARM指令集
  - Thumb指令集
- RISC的优点(T3)
- 嵌入式处理器的优点(T4)

8.ARM处理器的工作模式有哪些？什么是特权模式和异常模式?

工作模式有：

1. 用户模式
2. 系统模式
3. 管理模式
4. 快速中断模式
5. 外部中断模式
6. 终止模式
7. 未定义指令模式

特权模式：除了用户模式的6种模式

异常模式：除了用户模式和系统模式的5种模式

9.ARM处理器有哪两种运行状态？如何在这两种状态之间直接进行切换

ARM运行状态和Thumb运行状态

使用BX和BLX这两条带状态切换的跳转指令（看操作数最低为是0是1，是0，从Thumb跳到ARM；是1，从ARM跳到Thumb）

12.开发人员在选用ARM处理器时有哪些基本原则?

1. ARM微处理器内核的选择
2. 系统的工作频率
3. 片内存储器的容量
4. 片内外围电路的选择

13.为什么Hi3861芯片适应于智能家电等物联网智能终端领域?

- 接口丰富
- 性能优秀
- 低功耗
- 支持WIFI、蓝牙等

附：ARM的寄存器组织

ARM状态下：

- R0~R7：通用寄存器，只有1组物理寄存器
- R8~R12：有2组物理寄存器 FIQ模式单独一组
- R13：堆栈寄存器，有6组物理寄存器
- R14：返回地址寄存器，有6组物理寄存器
- R15用作PC
- CPSR(当前程序状态寄存器)
- SPSR（5组，只有异常模式那5个有）异常发生时，SPSR用于保存CPSR的值。弄完后再恢复。

Thumb状态下：没有R8~R12

附：ARM的异常处理流程

进入异常处理（CPU自动完成）

1. 将CPSR记录到SPSR中
2. 进入ARM模式
3. 关FIQ、IRQ中断
4. 保存返回地址到LR
5. PC跳转到异常向量处（ARM的异常向量表存的是跳转指令）

退出异常处理（手工写程序）

1. 恢复SPSR
2. 将LR中的值调整后送回PC

第三章 嵌入式系统硬件平台设计

1.嵌入式微处理器如何选择?

- 处理器I/O接口选择

    - GPIO的需求
    - 是否有基本通信接口UART 、I2C、SPI
    - 是否需要USB总线
- 处理器存储系统选择
    - 寻址空间多大
    - 是否有MMU
    - 是否支持SDRAM
- 市场因素的影响
    - 价格
    - 是否容易购买
    - 技术支持与售后服务

2.接口和总线有什么区别?

接口是指**两个**系统或部件之间的**交接**部分

- 硬件接口：两个硬件之间的电子线路
- 软件接口：软件之间为交换信息设置的逻辑边界

总线是计算机系统中**各个**部件（各个系统之间）传递信息的一组**共享的信息通道**，一般由传输线、接口和总线控制器组成

3.嵌入式系统常用的存储器有哪些？各自的特点是什么?

ROM：不易失

SRAM：速度最快，易失

DRAM：速度快，易失

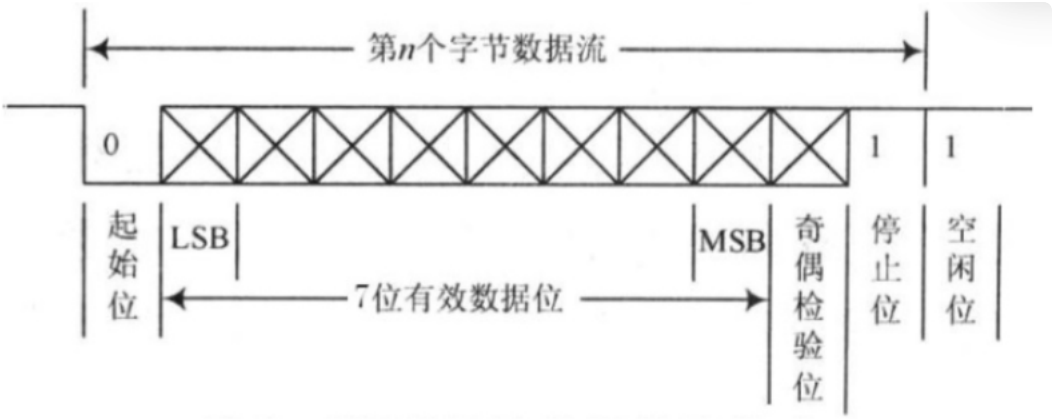
Flash：不易失

EEPROM：不易失，电可擦除

4.简述UART串行通信的基本原理与数据帧格式。

UART是**异步串行**的通信接口，串行通信的基本原理是发送设备和接收设备都使用**相同波特率**来进行通信。

数据帧格式：起始位、数据位、奇偶检验位和停止位。其中数据位和停止位的长度、奇偶校验可配置。



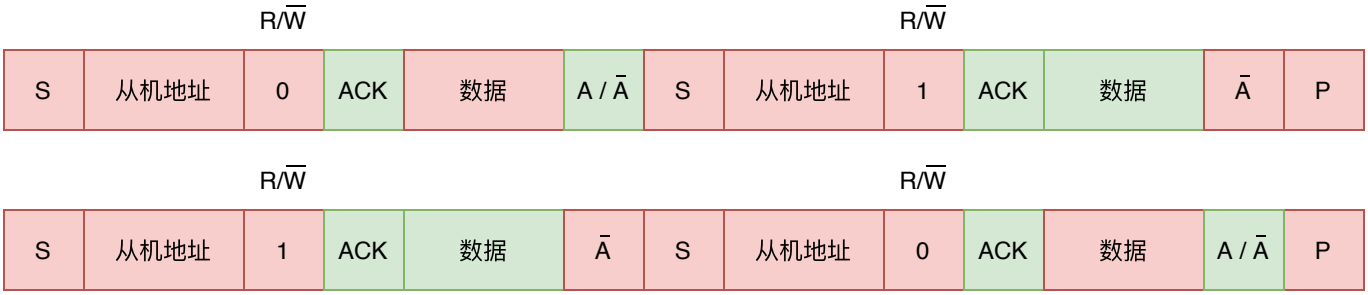
5.I2C总线上数据传送的顺序是什么?

主机发，从机收

起始位、地址、（读写位）、ACK、数据、ACK、数据、ACK.....数据、ACK/非ACK、停止位

主机收，从机发

起始位、地址、（读写位）、ACK、数据、ACK、数据、ACK.....数据、非ACK、停止位



注：①红色表示数据由主机向从机发送；绿色表示数据由从机向主机发送
②ACK表示应答；ACK表示非应答
③S是起始信号，P是终止信号

6.嵌入式系统的外围器件如何选择?

（上课没讲，PPT里没有，这能考？）

一个嵌入式系统除了嵌入式微处理器外，外围核心元器件指扩展外围电路的主要元器件。为减少开发过程中的困难，降低开发成本和风险，需要精心选择主要元器件。

一般来说，外围核心元器件的选型应该遵守以下原则：

1.普遍性原则

外围的元器件尽可能采用被广泛使用、验证过的成熟芯片，少使用冷门、偏门的芯片。这样可以减少由于使用冷偏门芯片而导致的风险，这种风险既存在于研发工程中，也存在于后续批量生产期间，或导致研发延期，或生产供货的滞后。

2.高性价比原则

所谓高性价比原则指在功能、性能、使用率都相近的情况下，尽量选择价格比较低的元器件，降低成本。但必须提醒的是，高性价比并不意味着选用最便宜的元器件，而是在能够满足系统对性能、可靠性等需求的基础上，寻找**价廉质优**的元器件。

3.采购方便原则

不同的元器件有不同的生产厂商和供货商家，因此造成同类型器件的供货周期、服务、备货都不尽相同。为保证研发、生产进度，在选择元器件时，应尽量选择容易买到、供货周期短的元器件。

4.持续发展原则

持续发展原则指尽量选择在可预见的时间周期内，不会停产的元器件；或即使因不可预见原因，该元器件停止供货，也可找到相应的引脚完全兼容的替代元器件。

5.向上兼容原则

向上兼容原则指**尽量选择以前老产品用过的元器件**，原因在于：一则熟悉此元器件的特性，可以避免由于采用不熟悉的元器件导致的设计缺陷或不可预料问题；二则可以消化库存，降低由库存、积压导致的成本提升。

6.资源节约原则

资源节约原则指的是：

（1）对于某些集成了多个相同部件的元器件，例如：双运放/四运放芯片，在一个芯片上集成了两个/四个完全相同的运算放大器，这些放大器共用一组电源。这样，如果电路中需要两个运算放大器，则应优先选用双运放芯片。因为四运放芯片固然也可以实现电路功能，但不可避免地会增加电路板体积、成本与功耗，而且还存在引入额外干扰的风险。

（2）尽可能使用集成芯片上的所有引脚，当然必须根据系统设计需求来确定，不能盲目追求资源的利用率。

9.请归纳Hi3861端口的概况，如何使用GPIO端口控制继电器的开和断?

Hi3861有GPIO、UART、PWM、I2C接口等

对于控制继电器的开和断，可以采用以下步骤：

1. 将GPIO端口设置为GPIO模式
2. 将GPIO端口设置为GPIO模式中的输出模式。
3. 通过写入数据（如高电平或低电平）来控制继电器的开关状态。

结合实验，GPIO、UART、PWM的使用

TODO

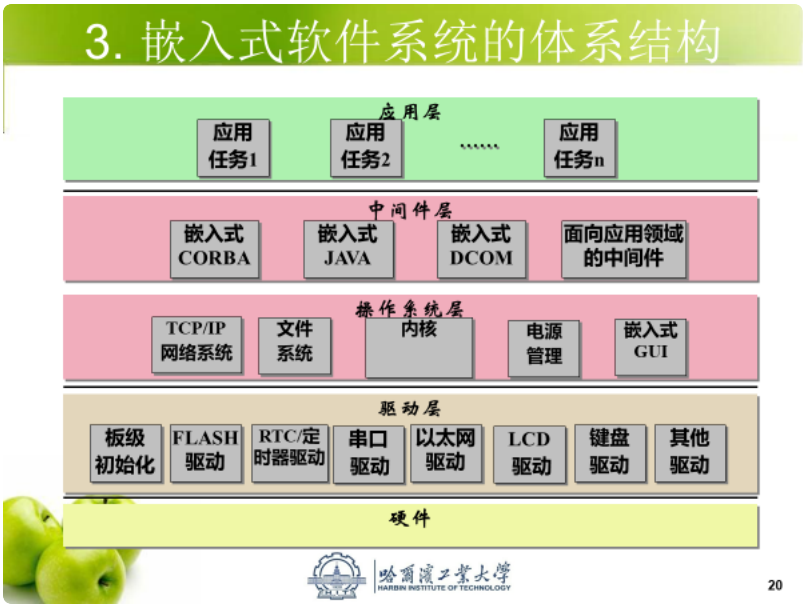
第四章 嵌入式系统软件设计

2.嵌入式软件的种类和每一类的特点是什么?

- **系统软件**：**控制、管理计算机资源的软件**
- **支撑软件**：**辅助**嵌入式软件**开发**的**工具软件**
- **应用软件**：面向特定应用领域的软件

3.嵌入式软件的体系结构包括哪些部分？每部分的作用是什么?

- **应用层**：每个应用完成特定的任务
- **中间件层**：连接应用层和操作系统层
- **操作系统层**：提供嵌入式内核、嵌入式TCP/IP、嵌入式文件系统等部分
- **硬件驱动层**：对操作系统和应用提供驱动支持



4.嵌入式软件的运行流程一般分为几个阶段？每个阶段完成的主要工作是什么？

1. 上电复位/板级初始化
    a. 初始化相关硬件
    b. 初始化堆栈指针
2. 系统引导/升级
    a. 系统引导：将OS搬运到RAM中来
    b. 系统升级：通过网络/串口拿到包进行升级
3. 系统初始化
    a. 初始化内核
    b. 初始化文件系统
    c. 初始化中间件
4. 应用初始化
    a. 进行应用任务的创建、信号量、消息队列等的创建
5. 多任务应用阶段
    a. 根据已确定的算法对任务进行调度

6.详细说明嵌入式操作系统的特点有哪些？

- **可移植性**：硬件相关部分、硬件无关部分

- **可重入函数**：一份函数代码被多个任务并发调用，且不产生数据干扰错误
- **可配置**：可调节的参数多
- **可裁剪**：可以将不必要的配件删除
- **可靠性高、稳定性好**：稳定、安全
- **实时性**
- **内核小**：适应嵌入式计算机存储空间小的限制
- **抢占式内核**：为保证高优先级任务的执行，采用抢占式内核

8.论述OpenHarmony操作系统的技术特点。

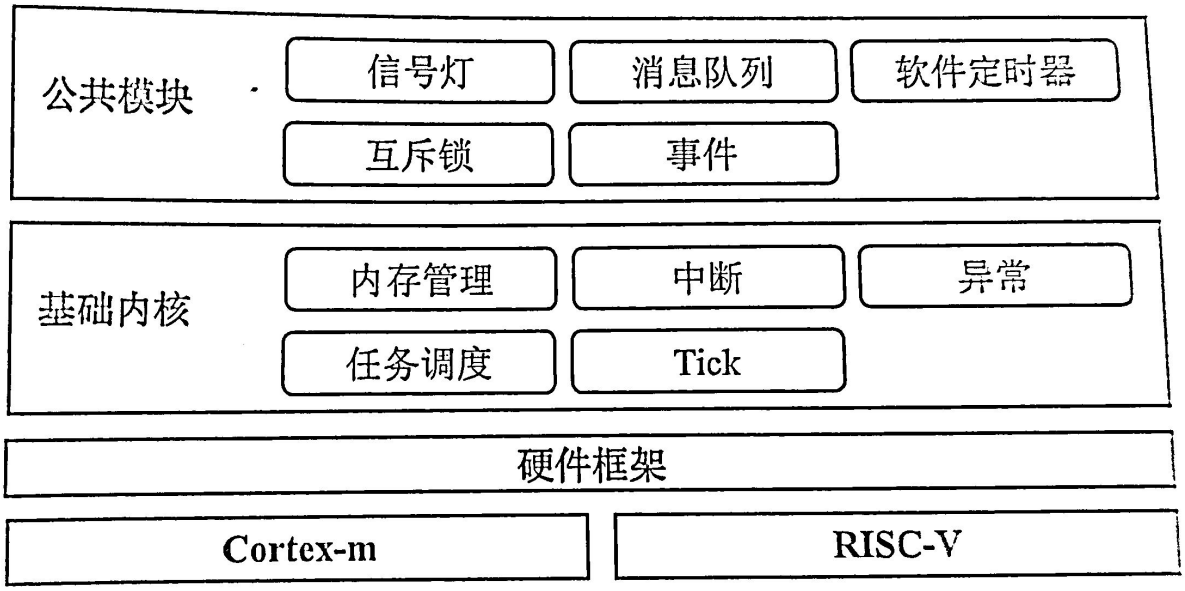
- **可裁剪**：内核/驱动/系统服务/框架都可裁
- **虚拟超级终端**：通过分布式软总线，用软件和通信把不同的硬件连接在一起
- **易开发**：一次开发，多段部署；易用的IDE

10.OpenHarmony内核的特点是什么?

- **匹配不同硬件**：针对L0-L5不同的设备匹配不同的内核
- **支持分布式调度**：分布式相关特性需要内核层面的支持
- **提升处理效率**：部分场景需提供超越安卓和IOS的性能

11.简述LiteOS-m和LiteOS-a系统架构的组成，它们之间的区别是什么?

LiteOS-m：硬件框架/基础内核/公共模块



LiteOS-a：内核空间/Syscall/MUSL/Shell

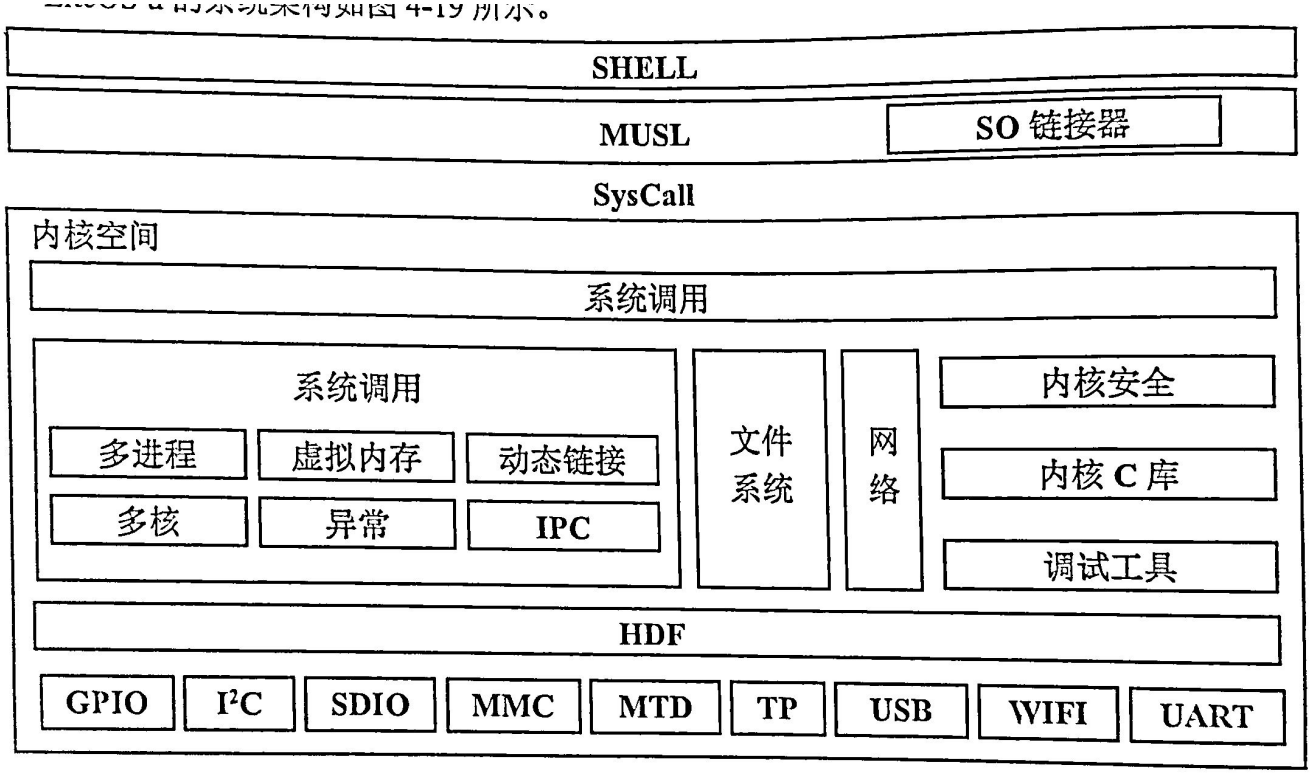


图 4-19 LiteOS-a 系统架构

区别：

1. 适配硬件不同：LiteOS-m对应RISC-V和ARM架构的Cortex-m核心，

LiteOS-a对应ARM架构的Cortex-a核心

2. 尺寸不同：LiteOS-m ≈128K，LiteOS-a ≈128M
3. 对虚拟内存的支持：LiteOS-m不支持，LiteOS-a支持
4. 对多核的支持：LiteOS-m不支持，LiteOS-a支持

第五章 任务管理与调度

1.什么是任务？任务有哪些主要特性，主要包含哪些内容?

任务是一个具有独立功能的、无限循环的程序段的一次运行活动，是实时内核调度的单位，具有动态性、并行性和异步独立性等特性。

①动态性：任务状态是不断变化的。任务状态一般分为就绪态、执行态和等待态。在多任务系统中，任务的状态将随着系统的需要不断进行变化。

②并行性：系统中同时存在多个任务，这些任务在宏观上是同时运行的。

③异步独立性：每个任务按其相互独立的、不可预知的速度运行，走走停停。

任务主要包含以下内容：①代码，一段可执行的程序；②数据，程序所需要的相关数据（变量、工作空间、缓冲区等）；③堆栈；④程序执行的上下文环境。

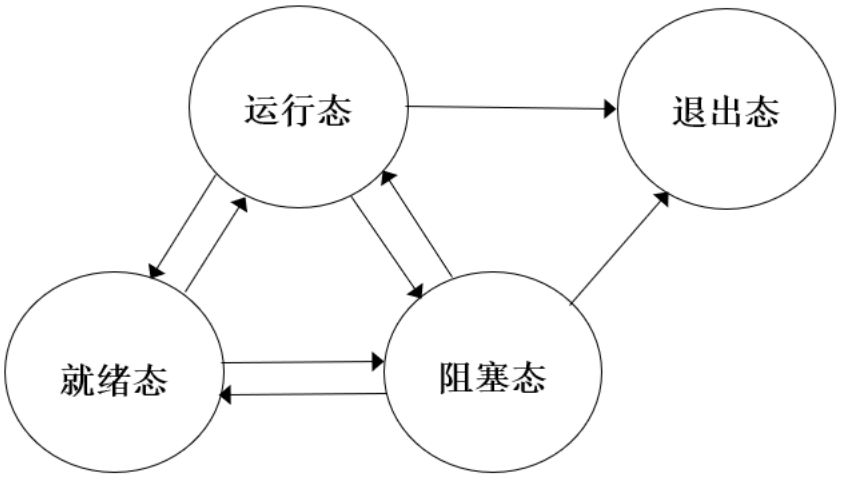
4.什么是抢占式调度和非抢占式调度?

抢占式调度：正在运行的任务可能被其他任务所打断。

非抢占式调度：一旦任务开始运行，该任务只有在运行完成而主动放弃CPU资源，或是因为等待其他资源被阻塞的情况下才会停止运行。

实时内核**大都采用了抢占式调度算法**，使关键任务能够打断非关键任务的执行，**确保关键任务的截止时间能够得到满足**。非抢占式调度算法用于那些任务需要按照预先确定的顺序进行执行的情况。

5.任务主要包含哪些状态,就OpenHarmony LiteOS m任务状态之间的变迁情况进行描述。



6.什么是任务切换?任务切换通常在什么时候进行,说明任务切换的主要过程。

任务切换是指保存当前任务的上下文，并恢复需要执行任务的上下文的过程。

通常在下述过程进行任务切换：

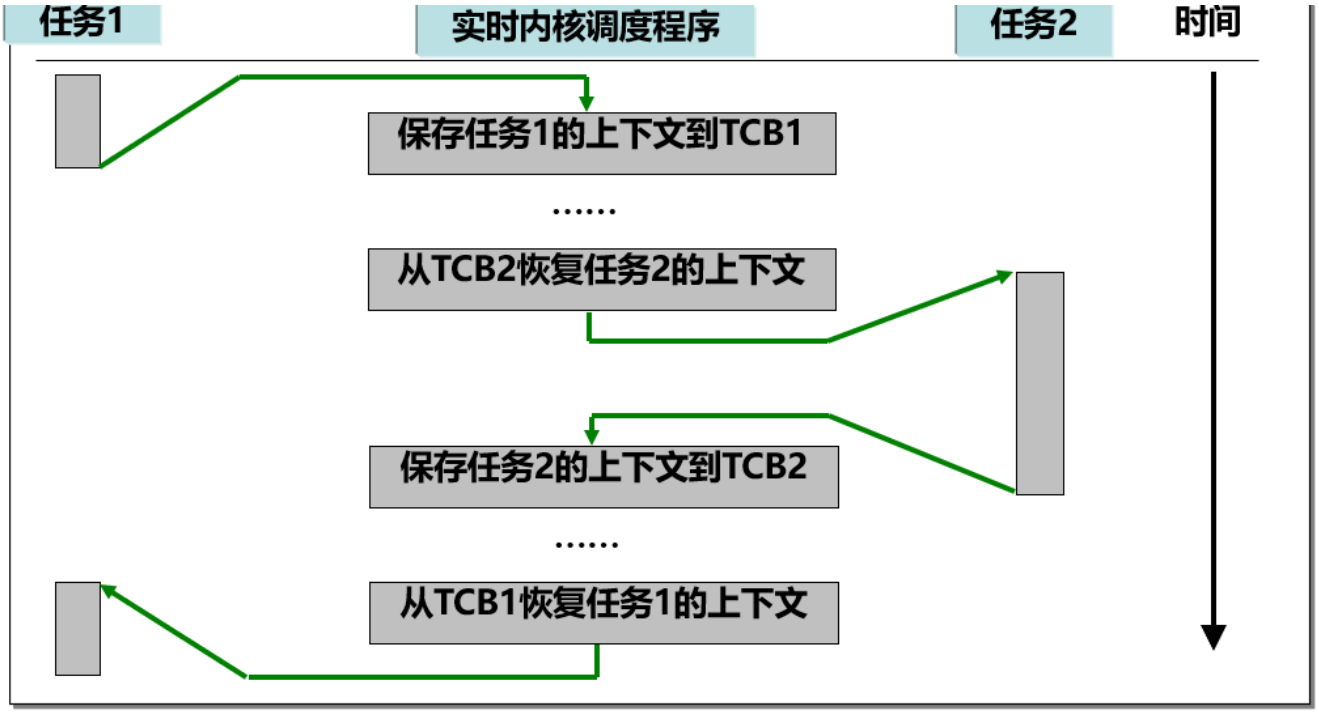
①中断、自陷。当I/O中断发生的时候，如果I/O活动是一个或多个任务正在等待的事件，则实时内核将把相应的处于等待态的任务转换为就绪态，同时，实时内核还将确定是否继续执行当前处于运行状态的任务，或用高优先级的就绪任务抢占该任务。自陷是由于执行任务中当前指令所引起的，将导致实时内核处理相应的错误或异常事件，并根据事件类型，确定是否进行任务的切换。

②运行任务因缺乏资源而被阻塞。例如，任务执行过程中进行I/O操作时（如打开文件），如果此前该文件已被其他任务打开，则将导致当前任务处于等待态，而不能继续执行。

③采用时间片轮转调度时，实时内核将在时钟中断处理程序中确定当前正在运行**任务的执行时间是否已经超过了设定的时间片**，如果超过了时间片，则实时内核将停止当前任务的运行，把当前任务的状态变为就绪态，并把另一个任务投入运行。

④采用基于优先级的抢占式调度算法时，一个高优先级任务处于就绪时，则将导致当前任务停止运行，使更高优先级的任务处于运行状态。

任务切换的基本步骤：



1. 保存任务上下文环境（到TCB中）
2. 更新当前运行任务的控制块内容，将其状态改为就绪或等待状态
3. 将任务控制块移到相应队列（就绪队列或等待队列）
4. 选择另一个任务执行（调度）
5. 改变需投入运行任务的控制块内容，将其状态变为运行状态
6. （根据TCB）恢复需投入运行任务的上下文环境

8.优先级位图算法的工作原理是什么?为什么可以提高实时内核的确定性?

1. 相关变量

假设OS最多支持64个优先级不同的任务；每个任务的优先级都不同（不存在两个拥有相同优先级的任务）

`char priorityReadyTable[8]`：每1位代表1个优先级的任务是否就绪。数组中的每1项是1个字节，代表8个任务。数组总共8项，代表64个任务。

`char priorityReadyGroup`：8位，每1位代表`priorityReadyTable`中的1行是否有就绪任务。

`char priorityMapTable[8]`：可以理解为译码器，将0~7数字转换为第几位为1、其余位为0的8位二进制数。

`priorityMapTable[0]==8'b0000_0001`、`priorityMapTable[1]==8'b0000_0010`、`priorityMapTable[2]==8'b0000_0100`...etc

`char priorityDecisionTable[2^8=256]`：给定一个8位二进制数，查表后得该二进制数 为1的 最低位是哪一位。

e.g. `priorityDecisionTable[8'b0010_0100]=2`

2. 相关操作

每个任务都有一个8位的优先级`priority`。高3位代表在第几行、低3位代表在第几列。

- 任务进入就绪态时
  - 将相关任务的位设置为1
    - `priorityReadyGroup |= priorityMapTable[priority>>3]`
    - `priorityReadyTable[priority>>3] |= priorityMapTable[priority&0x7]`
- 任务退出就绪态
  - 将相关任务的位设置为0

```cpp
//将priorityTable中对应行干成0，如果置完后Group一整行都为0，将priorityReadyGroup对应位置0
if( (priorityReadyTable[priority>>3] &= ~priorityMapTable[priority&0x7]) == 0)
    priorityReadyGroup &= ~priorityMapTable[priority>>3];
```

- 获取当前所有就绪任务中，优先级最高的任务
  - `high3bit = priorityDecisionTable[priorityReadyGroup]`
  - `low3bit = priorityDecisionTable[priorityReadyTable[high3bit]]`
  - `p = high3bit << 3 + low3bit`

为什么可以提高确定性？因为相关操作的时间和任务的数量没有关系。

9.什么是优先级反转?解决优先级反转有哪些主要方法?分别就这些方法进行描述

书P137

优先级反转（priority inversion）：**高优先级任务需要等待低优先级任务释放资源，而低优先级任务又正在等待中等优先级任务的现象。**

解决方案：

①优先级继承协议：当一个任务**阻塞了**一个或多个高优先级任务时，该任务将不使用其原来的优先级，而使用**被该任务所阻塞的所有任务的最高优先级作为其执行临界区的优先级**。当该任务**退出临界区时，又恢复到其最初的优先级**。

②优先级天花板协议：对于控制临界区的信号量，设置**信号量的优先级天花板**为可能申请该信号量的所有任务中**具有最高优先级任务的优先级**；**如果任务成功获得信号量，任务的优先级将立即被抬升**为信号量的优先级天花板；任务执行完临

界区，释放信号量后，其优先级恢复到其最初的优先级；**如果任务不能获得所申请的信号量，任务将被阻塞。**

第六章 嵌入式操作系统

1.什么是中断？中断与自陷、异常之间有哪些联系与区别？

广义中断：导致程序**正常执行流程发生改变**的事件

狭义中断：中断是由于**CPU外部的原因**而改变程序执行流程的过程

自陷：自陷表示**通过处理器所拥有的软件指令**，可预期地使程序的执行流程发生变化，以执行特定的程序。

异常：CPU自动产生的自陷，以处理异常事件，如被0整除等。

联系：狭义中断、自陷、异常均属于广义中断，**都能够使程序正常执行流程发生改变。**

区别：狭义中断属于**异步**事件，自陷、异常属于**同步**事件

3.简述中断处理的基本过程，以及编写中断服务程序时需要注意的事项。

- 中断检测
  - 在每条指令结束时进行中断检测，检测是否有中断请求或是否满足异常条件
- 中断响应
  - 中断响应是由处理器内部硬件完成的中断序列，而不是由程序执行的。
  - 获取中断向量、查中断向量表、跳转到中断处理程序
- 中断处理
  - 即执行中断服务程序
    - **保存上下文**：保存中断服务程序将要使用的所有寄存器的内容，在退出中断服务程序之前要进行恢复；

- **确定产生中断的设备**：如果中断向量被多个设备所共享，为了确定产生该中断信号的设备，需要轮询这些设备的中断状态寄存器；
- **获取中断相关的其他信息**（如果有）；
- **对中断进行具体的处理**；
- **恢复保存的上下文**；
- **执行中断返回指令**，使CPU的控制返回到被中断的程序继续执行。

4.论述嵌入式实时系统时钟设备的组成和内核提供的时间管理功能。

系统时钟设备的组成：

- 实时时钟（real time clock，RTC）
  - 一般靠电池供电，即使**系统断电，也可以维持日期和时间**。
  - 实时时钟独立于操作系统，所以也被称为**硬件时钟**，为整个系统**提供一个计时标准**。
- 定时器/计数器（timer/counter）
  - 实时内核需要一个定时器作为系统时钟（或称OS时钟），并由实时内核控制系统时钟工作。
  - 一般情况下，系统时钟的最小粒度是由应用和操作系统的特点决定的。

内核提供的时间管理功能：

- **维护时间**：维护相对时间（单位tick）和日历时间
  - 设置系统时间
  - 获得系统时间
  - 维护系统时基、处理定时事件
- **计时功能**
  - 对任务进行有限等待的计时
  - 时间片轮转调度的计时

- **定时功能**

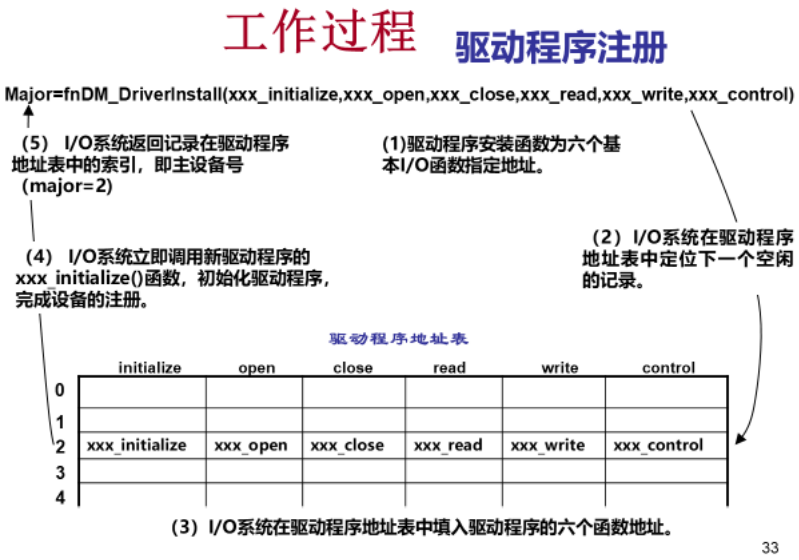
7.嵌入式系统是如何实现对I/O系统管理的?

- 设备管理
  - 驱动程序地址表：各个驱动的功能函数地址
  - 设备名表：(设备名，主设备号，次设备号)
  - 文件描述符表：记录当前打开的文件与设备
- 驱动逻辑
  - 实现了所有的类别的驱动
  - 每一类驱动维护一个设备信息表，记录每个设备的次设备号、设备数据与设备接口函数表等
- 硬件抽象
  - 可变的部分：初始化
  - 不可变的部分：对硬件的抽象
- 具体实现
  - 驱动程序注册
    - 准备好相关函数实现（`initialize`、`open`、`close`等）
    - 在驱动程序地址表中找到一个空项
    - 将相关信息填写到驱动程序地址表中
    - 调用`initialize`函数
    - 返回驱动程序地址表中的索引（主设备号）
  - 注册设备
    - 准备好设备名、主设备号、次设备号
    - 将信息填入设备名表中
  - 打开设备
    - 先到 设备名表中 找到相关设备
    - 在文件描述符表中找到一项 写入相关信息
    - 将设备名表对应项的地址 写入 文件描述符表
    - 根据设备名表中的主设备号 去驱动程序地址表中 找到`open`函数

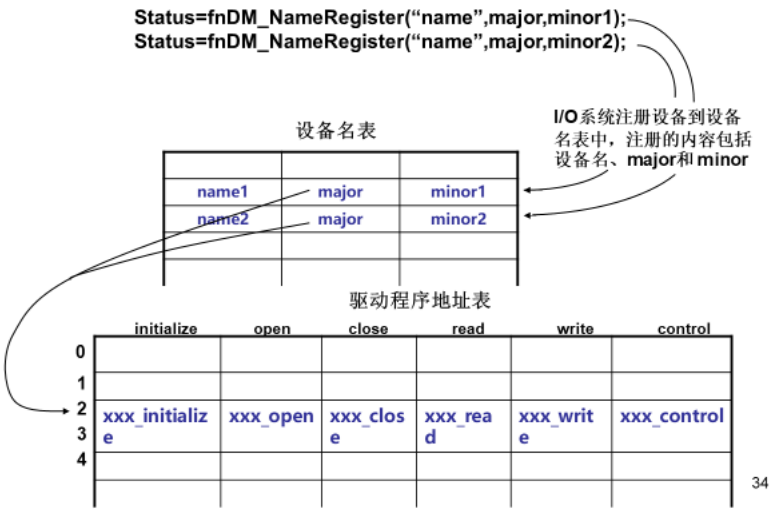
- 将相关信息（次设备号等）传给 `open` 函数
- 将 `open` 函数返回值写入文件描述符表
- 返回文件描述符（文件描述符表的索引）

○ 写设备

- 根据文件描述符 去 文件描述符表中找对应项
- 根据文件描述符表中对应项的驱动信息，去找 设备名表的 对应项
- 根据设备名表的对应项 中的主设备号，去找 设备驱动程序地址表中的 `write` 函数
- 相关参数（次设备等）传给 `write` 并调用
- 将 `write` 的返回值（操作的字节数）输出

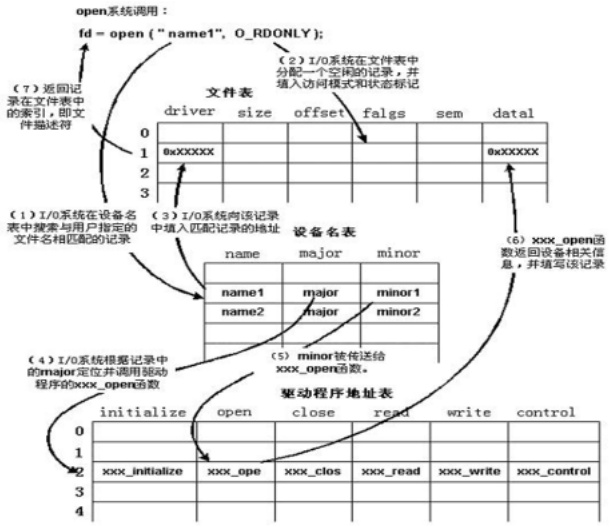


工作过程 设备注册

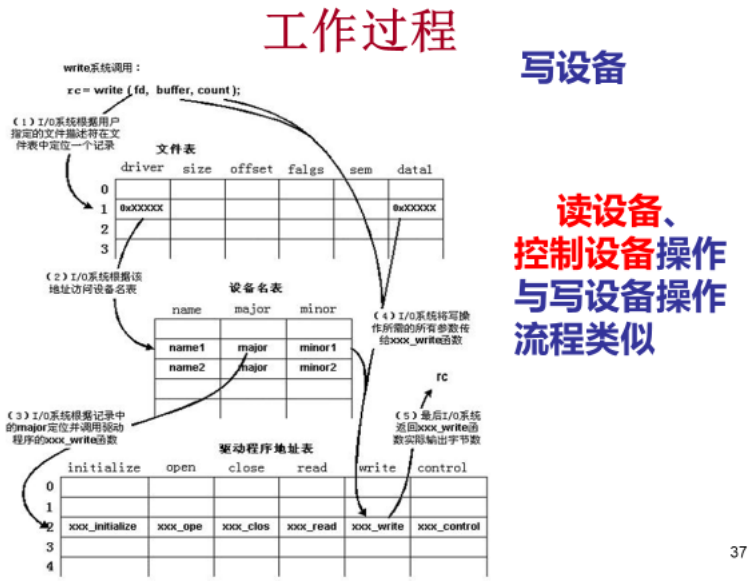


34

工作过程 打开设备



35



第七章 Boot loader

1.什么是Boot Loader？它的主要功能是什么？

BootLoader是在操作系统内核运行之前运行的一段小程序

主要功能

- 初始化硬件设备
- 建立内存空间的映射图
- 调整系统的软硬件环境，以便操作系统内核启动

2.Boot Loader有几种操作模式？它们分别以何种方式工作？

有两种操作模式：启动加载模式和下载模式

启动加载模式：以自主（Autonomous）模式，无需用户介入，直接将OS加载到RAM中运行；

下载模式：需用户介入进入下载模式。在控制台打印提示信息，等待用户输入相关信息，从主机下载所需的映像（先保存在RAM，再保存在Flash等介质中）

5.Boot Loader的工作过程一般来说分为哪两个阶段？每个阶段的核心工作是什么？

大多数 Boot Loader 都分为 stage1 和 stage2 两大部分。

stage1 通常包括以下步骤

1. 硬件设备初始化
2. 为加载 Boot Loader 的 stage2 准备 RAM 空间
3. 拷贝 Boot Loader 的 stage2 到 RAM 空间中
4. 设置好堆栈
5. 跳转到 stage2 的 C 入口点

stage2 通常包括以下步骤

1. 初始化本阶段要使用到的硬件设备
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 加载到 RAM 中
4. 为内核设置启动参数
5. 调用内核

9.Hi3861的Boot分为几部分？每部分实现的功能是什么？

- RomBoot
  - 加载LoaderBoot到RAM
  - 或加载FlashBoot到RAM
- LoaderBoot
  - 下载镜像到Flash
  - 烧写EFUSE
- FlashBoot
  - 升级固件

    - 检验并引导固件 注：分AB面
- CommonBoot
    - LoaderBoot和FlashBoot共同的部分

第八章 嵌入式系统设计

1.嵌入式系统的设计过程包括哪几个阶段？每一个阶段的主要工作有哪些

1. **需求**：收集系统的非形式描述，提炼出规格说明
2. **规格说明**：作为客户和设计者之间的协议，说明了系统的体系结构
3. **体系结构**：描述如何实现（功能和非功能）要求、规划整体结构、划分软硬件实现
4. **组件**：软硬件详细设计
5. **系统集成**：将软硬件集成在一起，进行测试，改正错误
6. **系统测试**：测试是否满足规格说明的要求

3.什么是软硬件协同设计方法?

软硬件协同设计方法**使用统一的表述形式描述软硬件**、根据描述和软硬件**划分结果，决定系统中软硬件、接口的实现方法，并将其集成**。

5.嵌入式软件与一般软件的主要区别是什么?

略

9.论述嵌入式系统低功耗设计的方法

- 动态电源管理
- 动态电压缩放
- 低功耗硬件系统
    - 选低功耗处理器
- 低功耗软件系统

- 低功耗编译
- 优化算法