嵌入式系统概论

黄色标识为 ** 考点
绿色标识为 * 考点

考试形式：闭卷、笔试，基本概念、基本原理、设计应用技术
范围：以课件涵盖内容为主
题型：简答题、问答题、设计题

知识点P?嵌入式系统的定义，特点，分类，典型应用II?

【定义】“嵌入式计算机系统”的简称

1.IEEE的定义：用于控制、监视或辅助操作机器和设备的装置（软件和硬件的综合体，可以涵盖机电等附属装置）

2.国内的定义：以应用为中心，以计算机技术为基础，软硬件可裁减，适用于应用系统对功能、可靠性、成本、体积、功耗有严格要求的专用计算机系统

3.嵌入式系统三要素：嵌入、专用、计算机

- 嵌入性：嵌入到对象体系中，有对象环境要求
- 专用性：软、硬件按对象要求设计、裁减
- 计算机：实现对象的智能化功能

【特点】

1. 形式多样，面向特定应用
2. 得到多种类型处理器和处理器体系结构的支持
3. 极其关注成本
4. 有实时性和可靠性要求
5. 使用的操作系统适应多种处理器、可剪裁、轻量型、实时可靠、可固化的嵌入式操作系统
6. 需要专门工具和特殊方法

【通用计算机与嵌入式系统对比】

特征	通用计算机	嵌入式系统
形式与类型	看得见的计算机。 按其体系结构、运算速度和结构规模等因素分为大、中、小型机和微机。	看不见的计算机。 形式多样，应用领域广泛，按应用来分。
组成	通用处理器、标准总线和外设。 软件和硬件相对独立。	面向应用的嵌入式微处理器，总线和外部接口多集成在处理器内部。 软件与硬件是紧密集成在一起的。
开发方式	开发平台和运行平台都是通用计算机	采用交叉开发方式，开发平台一般是通用计算机，运行平台是嵌入式系统。
二次开发性	应用程序可重新编制	一般不能再编程

【分类】

1. 按处理器位数：4位、8位、16位、32位、64位
2. 按应用：信息家电、汽车电子、通信、移动终端、工业控制
3. 按速度：强实时系统、一般实时系统、弱实时系统
4. 按确定性：硬实时系统、软实时系统
5. 按复杂程度：循环轮询系统、有限状态机系统、前后台系统、单处理器多任务系统、多处理器多任务系统

【典型应用】
工控设备、军用电子设备、航天航空、汽车电子、信息家电、通信、智能玩具、可穿戴设备

【IoT、SoC】

知识点2 嵌入式硬件系统基础

对性能有影响的硬件?
【冯诺伊曼结构与哈佛结构】
- 冯诺伊曼结构：数据和程序放在同一个存储单元，统一编址，指令和数据通过同一个总线访问
- 哈佛结构：程序和数据存储在不同的存储空间中，即程序存储器和数据存储器是两个相互独立的存储器，每个存储器独立编制、独立访问。与之相对应的是系统中设置的两条总线（程序总线和数据总线），从而使数据的吞吐率提高了一倍（更大存储带宽和更大可预测带宽）
- 哈佛结构不能使用自修改代码

【CISC vs RISC】
- CISC：复杂指令集（许多地址格式，许多操作）
- RISC：精简指令集（流水型指令）
- RISC机器用来减少指令周期的一种技术，可提高处理器和总线的使用率

	CISC	RISC
价格	由硬件完成部分软件功能，硬件复杂性增加，芯片成本高	有软件完成部分硬件功能，软件复杂性增加，芯片成本低
性能	减少代码尺寸，增加指令的执行周期数	使用流水线降低指令的执行周期数，增加代码尺寸
指令集	大量的混杂指令集，有简单快速的指令，也有复杂多周期指令，符合HLL	简单的但周期指令，在汇编指令方面有相应的CISC微代码指令
高级语言支持	硬件完成	软件完成
寻址模式	复杂的寻址模式，支持内存到内存寻址	简单的寻址模式，仅允许LOAD和STORE指令存取内存，其它所有的操作都基于寄存器到寄存器
寄存器数目	寄存器较少	寄存器较多

【流水线技术】**
RISC机器用来减少指令周期的一种技术，提高处理器和总线的使用率
在CPU中由若干个个不同功能的电路单元组成一条指令处理流水线，然后将一条指令分成若干步后再由这些电路单元分别执行，这样就能实现在一个CPU时钟周期完成一条指令，因此提高CPU的运算速度。
优点：提高CPU利用率、并行计算

①3级流水线(ARM7TDMI)
取指——译码——执行
②5级流水线(ARM9TDMI)
取指——译码——执行——数据缓冲——写回
Reduces work per cycle => allows higher clock frequency
Separates data and instruction memory =>reduction of CPI
③6级流水线(ARM10TDMI)
取指——发射——译码——执行——存储器——写回
Pipeline flushed and refilled on branch, causing execution to slow down
Special features in instruction set eliminate(消除) small jumps in code to obtain the best flow through pipeline

【信息存储的字节顺序】
不管是大端法还是小端法存储，计算机在内存中存放数据的顺序都是从低地址到高地址
大端法是首先取高字节的数据放在低地址；小端法是首先取低字节的数据放在低地址

知识点3 ARM**

【工作状态】：ARM微处理器的两种工作状态，用来选择哪个指令集被执行
1.ARM状态：处理器执行32位的ARM指令；ARM指令要求字对齐
2.Thumb状态：处理器执行16位的Thumb指令；Thumb指令要求半字对齐

- 在一种工作状态下可以通过转移指令切换到另一种工作状态
- ARM和Thumb之间的状态切换不影响处理器工作模式和寄存器中的内容。
- ARM微处理器在开始执行代码时，应该处于ARM状态。
- 所有异常自动进入 ARM 状态。
- ARM微处理器在开始挨靠代码时总处理ARM状态，也就是复位后进入ARM状态

【ARM微处理器两种工作状态的切换方法】
1.进入Thumb状态：
- 执行BX指令（带状态切换的跳转指令）
- 当操作数寄存器的最低位[0]为1时，可以使微处理器从ARM状态切换到Thumb状态
- BX R0 R0的最低位[0]为1
- 处理器工作在Thumb状态，如果发生异常并进入异常处理子程序，则异常完毕返回时，自动从ARM状态切换到Thumb状态

2.进入ARM状态：
- 执行BX指令
- 当操作数寄存器的最低位[0]为0时，可以使微处理器从Thumb状态切换到ARM状态
- BX R0 R0的最低位[0]为0
- 处理器工作在Thumb状态，如果发生异常并进入异常处理子程序，则进入时，自动从Thumb状态切换到ARM状态

【运行模式】
1. 用户模式(usr)： ARM处理器正常的程序执行状态
2. 系统模式(sys)： 运行具有特权的操作系统任务
3. 快中断模式(fiq)： 支持高速数据传输或通道处理
4. 管理模式(svc)： 操作系统保护模式
5. 数据访问终止模式(abt)：用于虚拟存储器及存储器保护
6. 中断模式(irq)： 用于通用的中断处理
7. 未定义指令终止模式(und)： 支持硬件协处理器的软件仿真

除用户模式外，其余6种模式称为非用户模式或特权模式；用户模式和系统模式之外的5种模式称为异常模式。ARM处理器的运行模式可以通过软件改变，也可以通过外部中断或异常处理改变。

【寄存器结构】
一共有37个寄存器，30个32位通用寄存器，16个可见R0–R15，其他的加速异常处理系统
1.31个通用寄存器，包括程序计数器（PC指针）。
• R0–R15可见，其他用于加快异常处理过程；
• R13：Stack Pointer（SP）栈寄存器（软件）
• R14：Link Register（LR） 选择性的为分支指令保存返回地址（硬件）
• R15：Program Counter（PC） 程序计数器（硬件）
2.6个状态寄存器，用以标识CPU的工作状态及程序的运行状态，只使用了其中的一部分。
• 1个当前状态寄存器，5个保存程序状态的寄存器
• 程序状态寄存器（CPSR）current program status register
• 程序状态保护寄存器（SPSR）saved program status register

【重难点】
1.每种异常模式都有一个SPSR。异常出现时，SPSR用于保留CPSP的状态。格式相同。

2.条件码标志位 / 状态标志位：N、Z、C、V（保存ALU中的当前操作信息）
• N：正负号/大小 标志位，0表示：正数/大于；1表示：负数/小于
• Z：零标志位，0表示：结果不为零；1表示：结果为零
• C：进位/借位/移出位，0表示：未进位/借位/移出0；1表示：进位/未借位/移出1
• V：溢出标志位，0表示：结果未溢出；1表示：结果溢出
• Q：DSP指令溢出标志位（用于v5以上E系列），0表示：结果未溢出；1表示：结果溢出

（1）选择“S”后缀问题：指令中可以选择s后缀，以影响状态标志。但是比较指令（cmp、cmn、tst和teq）不需要后缀S，它们总会直接影响CPSR中的状态标志。
（2）对CPSR中标志位的影响：
• N 标志位：如果结果为负，则 N 标志位置 1 ； 否则清 0 。
• Z 标志位：如果结果为 0 ，则 Z 标志位置 1 ； 否则清 0 。
• C 标志位：如果是加、减运算指令或比较指令时， C 标志位设置为 A L U 的进位输出；否则设置为移位器的移位输出。如果不需要移位，则 C 保持不变。
• V 标志位：在非加减操作中， V 标志位保持原值。在加减操作中，如果有溢出，则置 1 ； 不发生溢出，则清 0 。
（3）关于恢复CPSR原值问题：

如果指令带有S后缀（除了比较指令以外），同时又以PC为目标寄存器进行操作，
在异常模式下：则操作的同时从SPSR恢复CPSR。比如：

```
movs pc, #0xff          /* cpsr = spsr; pc = 0xff */
adds pc, r1, #0xffffff00   /* cpsr = spsr; pc = r1 + 0xffffff00 */
ands pc, r1, r2    /* cpsr = spsr; pc = r1 & r2; */
```

在user或者system模式：会产生不可预料的结果，因为在这两种模式下没有SPSR。

【中断和异常】**
1.中断的进入

- 保存CPSR位到SPSR
- 设置CPSR对应位（切换到AMR模式、切换到异常模式、关中断）
- 保存PC到LR中
- PC根据向量表设置对应的地址

2.中断的退出

- 从SPSR中恢复CPSR
- 从LR中恢复PC

3.ARM异常与中断不做严格意义上的区别

- ARM的中断向量表内存放的是响应异常和中断的转移指令而不是中断向量地址。
- 在ARM处理器中，当异常发生时，完成当前指令后跳转到相应的异常中断处理程序入口执行异常中断处理。异常处理完毕后返回原来的程序断点继续执行原来的程序。

4.ARM异常按照起因的不同分为3类：

- 指令执行引起的直接异常：软件中断、未定义指令和预取指令中止属于这一类。
- 指令执行引起的间接异常：数据中止（在读取和存储数据时的存储器故障）属于这一类。
- 外部产生的与指令流无关的异常：复位、IRQ和FIQ属于这一类。

5.中断向量表

- 中断向量表中存放了各个异常中断以及处理程序的对应关系。
- 在ARM体系结构中，异常中断向量表的大小只有32个字节。其中，每个异常中断向量占4个字节。系统初始化时，中断向量表从0号存储单元开始存放。

中断向量地址	异常中断类型	异常中断模式	优先级（6级最低）
0x0	复位	管理模式，SVC	1
0x4	未定义指令	未定义指令中止，UND	6
0x8	软件中断	管理模式，SVC	6
0xC	指令预取中止	中止模式，ABT	5
0x10	数据访问中止	中止模式，ABT	2
0x14	保留	未使用	未使用
0x18	外部中断请求，IRQ	外部中断模式，IRQ	4
0x1C	快速中断请求，FIQ	快速中断模式，FIQ	3

【WatchDog】*

1.功能：在对系统稳定性要求较高的场合，防止嵌入式控制系统运行时受到外部干扰或者系统错误，程序出现“跑飞”，导致整个系统瘫痪，加入看门狗电路。当系统“跑飞”而进入死循环时，恢复系统的运行

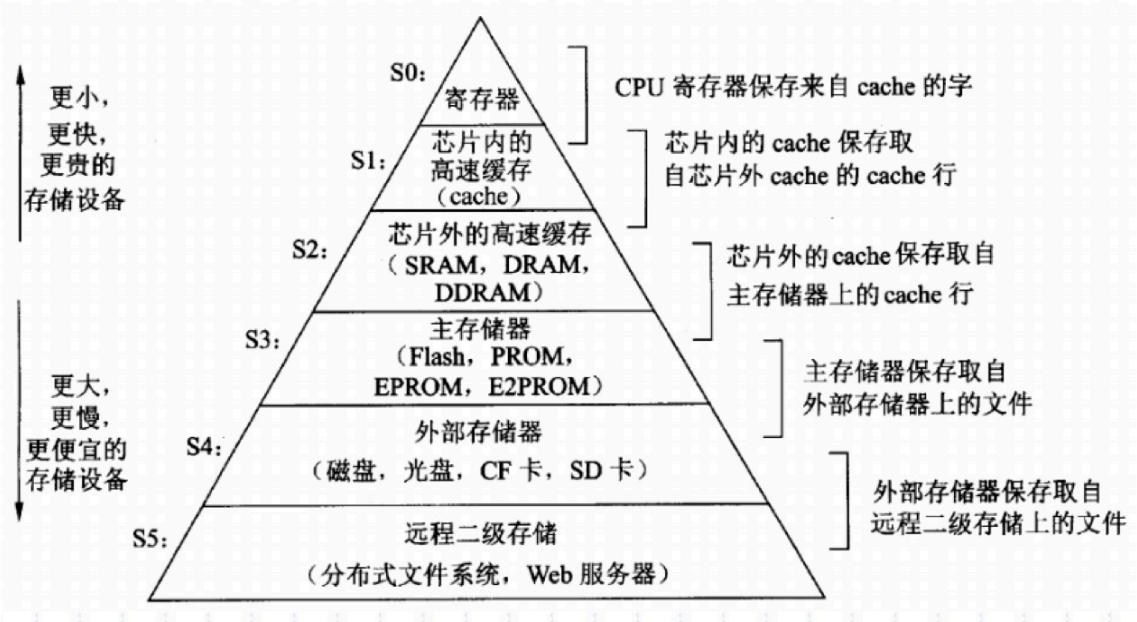
2.基本原理：设本系统程序完整运行一周期的时间是tp，看门狗的定时周期为ti，且ti>tp，在程序运行一周期后就修改（再重新设定看门狗的定时周期）定时器的计数值（俗称“喂狗”），只要程序正常运行，定时器就不会溢出。若由于干扰等原因使系统不能在tp时刻修改定时器的计数值，定时器将在ti时刻溢出，引发系统复位，使系统得以重新运行，从而起到监控作用。

3.定时器寄存器

- 看门狗定时器控制寄存器WTCON：通过该寄存器，可以使能/禁止看门狗、选择输入时钟源、使能/关闭中断、使能/关闭输出
- 看门狗定时器数据寄存器WTDAT：该数据寄存器用于设置看门狗定时器的初值。在初始的操作中，该值不会自动加载到定时器中，首次定时器初始值是0x8000，以后该寄存器的值会被自动加载到WTCNT寄存器中。一般是该定时器工作在通用定时器模式下时使用
- 看门狗定时器/计数器寄存器WTCNT：该寄存器为看门狗定时器的计数器，它的值表示该定时器的当前计数值，即到下一次溢出还需要经历的时钟数。当该定时器工作在看门狗模式时使用该寄存器，每次溢出前需要重新设置其值，以防止发生复位

知识点4 嵌入式系统的存储体系

【存储器系统：存储器系统的层次结构】



（CPU寄存器）寄存器组

（芯片内高速缓存）片内cache、写缓存、TCM、片内SRAM

（芯片外高速缓存）板卡级SRAM、DRAM和SRAM

（主存储器）FLASH PROM EPROM
（外部存储器）硬盘驱动器和光盘驱动器
嵌入式系统的存储器包括主存和外存
大多数嵌入式系统的代码和数据都存储在处理器可直接访问的存储空间即主存中。
系统上电后在主存中的代码直接运行。主存储器的特点是速度快，一般采用ROM、EPROM、Nor Flash、SRAM、DRAM等存储器件。
目前有些嵌入式系统除了主存外，还有外存。外存是处理器不能直接访问的存储器，用来存放各种信息，相对主存而言具有价格低、容量大的特点。
在嵌入式系统中一般不采用硬盘而采用电子盘做外存，电子盘的主要种类有NandFlash、SD（Secure Digital）卡、CompactFlash、SmartMedia、Memory Stick、MultiMediaCard、DOC（Disk On Chip）等

【ROM的种类与选型】

【Flash的种类与选型】

	NOR	NAND
写入/擦除一个块的操作时间	1~5s	2~4ms
读性能	1200~1500KB	600~800KB
写性能	<80KB	200~400KB
接口/总线	SRAM接口/独立的地址数据总线	8位地址/数据/控制总线，I/O接口复杂
读取模式	随机读取	串行存取数据
成本	较高	较低，单元尺寸约为NOR的一半，生产过程简单，同样大小的芯片可以做更大的容量
容量及应用场合	1~64MB，主要用于存储代码	8MB~4GB，主要用于存储数据
擦写次数（耐用性）	约10万次	约100万次
位交换（bit位翻转）	少	较多，关键性数据需要错误探测/错误更正（EDC/ECC）算法
坏块处理	无，因为坏块故障率少	随机分布，无法修正

【RAM的种类与选型】
~~【Cache】~~

知识点5 嵌入式系统总线

【总线结构：单、双、多】
1.单总线结构：使用一条单一的系统总线来链接CPU、主存和I/O设备。总线只能分时工作，使信息传送的吞吐量受到限制。

2.双总线结构：在CPU和主存之间专门设置了一组高速的存储总线，使CPU可通过专用总线与存储器交换信息，并减轻了系统总线的负担。主存仍可通过系统总线与外设之间实现DMA操作，而不必经过CPU。

3.多总线结构：在双总线系统的基础上增加I/O总线，其中系统总线是CPU、主存和通道（IOP）之间进行数据传送的公共通路，而I/O是多个外部设备与通道之间进行数据传送的公共通路。通道实际上是一台具有特殊功能的处理器，它分担了一部分CPU的功能，以实现对外设的统一管理及外设与主存之间的数据传送。

【输入输出编程：忙等IO和中断IO】**
1.忙等IO：用指令来检查设备是否就绪
• 最简单的设备驱动方式
• CPU不能在检查设备的过程中执行其他指令
• 很难处理同时发生的IO操作

2.中断IO：基于子程序调用，使下一条指令为一个子程序调用的预定位置，返回位置被保存确保执行前台程序
• CPU和设备之间通过总线连接
• CPU和设备之间进行握手
• 设备发出中断请求
• 当CPU能处理中断时发出中断确认请求

3.两种机制确保中断更加有效：优先级决定哪个中断先获得CPU；中断向量决定每个中断对应的执行代码
忙等效率比较低，CPU在等待设备的时候不能做其他工作，无法同步进行输入输出
中断IO允许设备更改CPU控制流

4.中断执行序列

知识点6 嵌入式系统软件基础

【嵌入式软件的分类】
系统软件、支撑软件、应用软件

【嵌入式软件体系结构】**
考题：常见的体系结构，什么情况下采用哪种，各自的优势和缺陷

1. 轮转结构：主循环依次检查每个I/O设备，并为需要提供服务的设备提供服务。
2. 状态机
3. 带中断的轮转：中断程序处理硬件特别紧急的需求，然后设置标记；主循环轮询这些标记，然后根据需求进行后续的处理
4. 中断
5. 函数队列

【驱动层】
- 板级初始化程序
- 与系统软件相关的驱动
- 与应用软件相关的驱动
- 与应用软件相关的驱动不一定需要与操作系统连接，这些驱动的设计和开发由应用决定。

【操作系统层】
- 操作系统层包括嵌入式内核、嵌入式TCP/IP网络系统、嵌入式文件系统、嵌入式GUI系统和电源管理等部分。
- 其中嵌入式内核是基础和必备的部分，其他部分要根据嵌入式系统的需要来确定。

【中间件层】
- 目前在一些复杂的嵌入式系统中也开始采用中间件技术，主要包括嵌入式CORBA、嵌入式Java、嵌入式DCOM和面向应用领域的中间件软件。
- 如基于嵌入式CORBA的应用于软件无线电台的应用中间件SCA（Software Core Architecture）等

【应用层】
- 应用层软件主要由多个相对独立的应用任务组成
- 每个应用任务完成特定的工作，如I/O任务、计算的任务、通信任务等，由操作系统调度各个任务的运行。

知识点7 嵌入式操作系统基础

【RTOS概念、特点、选型原则】*
实时操作系统(Real-Time Operating System, RTOS)
1.RTOS：对外来事件能在限定的响应时间内做出预定质量处理的计算机系统
2.特点：可移植性、强调实时性能、内核精简、抢占式内核、使用可重入函数、可配置、可裁剪、高可靠性
PPT：可靠性、可预测性、确定性、高性能、紧凑、可扩展性
3.商业化RTOS举例：μC/OS-II，ThreadX，VxWorks，Linux/RT，FreeRTOS
4.选型原则：综合权衡：成本，可靠性，实时性，工具链，模块丰富，RTOS 内核 RAM、ROM 占用量，支持

【实时系统及任务调度（RMS、EDF算法）】**
1.RMS：单一速率调度
静态调度策略，分配给每个进程的优先级是固定不变的。基于单一速率分析。
周期最短的进程被指定为最高优先级。每个周期开始，P进去，根据优先级调度执行，执行结束就从就绪状态退出直到他下一个周期开始。看书P202
2.EDF：最早截止时限优先调度
动态的优先级方案，在进程执行时根据进程的启动时间改变进程的优先级。
根据截止时限顺序指定优先级。最高优先级的进程是当前截至时限最临近的进程。
做法：看书p205 先写出all时间片内的截至时限，然后根据截至时限进行选择调度

【嵌入式软件开发环境——交叉开发（宿主机、目标机）】*
1.嵌入式软件的交叉开发采取宿主机和目标机的模式，软件开发在宿主机（通常是PC）上，目标机用于运行软件

2.在采用宿主机/目标机模式开发嵌入式应用软件时：
• 首先利用宿主机上丰富的资源和良好的开发环境开发和仿真调试目标机上的软件
• 然后通过串口或者以太网络连接将交叉编译生成的目标代码传输并装载到目标机上
• 并在监控程序或者操作系统的支持下利用交叉调试器进行分析和调试
• 最后目标机在特定环境下脱离宿主机单独运行

知识点8 ucOS-II

【任务调度】**
1.进程、线程、任务的概念
2.思想：近似地每时每刻总是让优先级最高的就绪任务处于运行状态
为了保证这一点，它在系统或用户任务调用系统函数及执行中断服务程序结束时总是调用调度器，来确定应该运行的任务并运行它
3.依据：任务就绪表
根据就绪表获得待运行任务的任务控制块指针

4.根据就绪表获得待运行任务的任务控制块指针：处理器的SP=任务块中保存的SP
恢复待运行任务的运行环境：处理器的PC=任务堆栈中的断点地址
其实调度器在进行调度时，在这个位置还要进行一下判断：究竟是待运行任务是否为当前任务，如果是，则不切换；如果不是才切换，而且还要保存被中止任务的运行环境
5.可抢占调度、不可抢占调度、先来先服务、时间片轮转算法、优先级算法
6.任务的实现（任务的层次结构、任务控制块、任务的状态及状态转换、任务队列）

【中断与时钟】
1.中断：由于某种事件的发生而导致程序流程的改变。产生中断的事件称为中断源。
2.CPU响应中断的条件：
• 至少有一个中断源向CPU发出中断信号；
• 系统允许中断，且对此中断信号未予屏蔽。

3.时钟节拍是一种特殊的中断；
• μC/OS需要用户提供周期性信号源，用于实现时间延时和确认超时。节拍率应在10到100Hz之间，时钟节拍率越高，系统的额外负荷就越重；
• 时钟节拍的实际频率取决于用户应用程序的精度。时钟节拍源可以是专门的硬件定时器，或是来自50/60Hz交流电源的信号。
4.响应中断的过程：系统接收到中断请求后，这时如果CPU处于中断允许状态（即中断是开放的），系统就会中止正在运行的当前任务，而按照中断向量的指向转而去运行中断服务子程序；当中断服务子程序的运行结束后，系统将会根据情况返回到被中止的任务继续运行或者转向运行另一个具有更高优先级别的就绪任务

【同步与通信】
任务间通信的管理核心：事件控制块ECB
1.任务间通信：使用邮箱、消息队列实现
2.同步与互斥：同步可以通过开关中断实现，互斥使用信号量。

【存储管理】（静态、动态）（结合ucOSII）
都是从右向左排的。记住优先级标号从0开始。
ucos-II不划分内核和用户空间，系统只有一个地址空间。
管理：多个内存分区，每个分区的块大小相同，不同分区的块大小不同，为了解决碎片问题。

实时操作系统UCOS-II如何从64个任务拓展到支持256任务?
在OS_CFG.H中 #define OS_LOWEST_PRIO 63改成#define OS_LOWEST_PRIO 254
由于新版的ucos已经支持自动根据OS_LOWEST_PRIO判断TBL的大小和算法了，所以不用修改
对于旧版本：需要在ucos_ii.h里修改TBL的大小（8改成16），然后在os_core.c里修改算法

将16位分为低八位和高八位，如果低八位为0，则在高八位，反之，在低八位。找到高低之后，再用之前的方式寻找。
总之，由8组*一组8个变成16组*一组16个.低八位照常进行，如果在高八位，则提取出高八位按照低八位运运算模式利用map，并在最后计算优先级的时候+8再算

知识点9 bsp

【嵌入式软件运行流程】
1.上电复位、板级初始化
- 嵌入式系统上电复位后完成板级初始化工作。
- 板级初始化程序具有完全的硬件特性，一般采用汇编语言实现。不同的嵌入式系统，板级初始化时要完成的工作具有一定的特殊性，但以下工作一般是必须完成的：
- CPU中堆栈指针寄存器的初始化。
- BSS段（Block Storage Space表示未被初始化的数据）的初始化。
- CPU芯片级的初始化：中断控制器、内存等的初始化。

2.系统引导/升级
- 根据需要分别进入系统软件引导阶段或系统升级阶段。
- 软件可通过测试通信端口数据或判断特定开关的方式分别进入不同阶段。

系统引导：
- 将系统软件从NOR Flash中读取出来加载到RAM中运行：这种方式可以解决成本及Flash速度比RAM慢的问题。软件可压缩存储在Flash中。
- 不需将软件引导到RAM中而是让其直接在NorFlash上运行，进入系统初始化阶段。
- 将软件从外存（如NandFlash、CF卡、MMC等）中读取出来加载到RAM中运行：这种方式的成本更低。

系统升级：
- 进入系统升级阶段后系统可通过网络进行远程升级或通过串口进行本地升级。
- 远程升级一般支持TFTP、FTP、HTTP等方式。
- 本地升级可通过Console口使用超级终端或特定的升级软件进行。

3.系统初始化
- 在该阶段进行操作系统等系统软件各功能部分必需的初始化工作，如根据系统配置初始化数据空间、初始化系统所需的接口和外设等。
- 系统初始化阶段需要按特定顺序进行，如首先完成内核的初始化，然后完成网络、文件系统等的初始化，最后完成中间件等的初始化工作。

4.应用初始化：在该阶段进行应用任务的创建，信号量、消息队列的创建和与应用相关的其它初始化工作。

5.多任务应用运行：各种初始化工作完成后，系统进入多任务状态，操作系统按照已确定的算法进行任务的调度，各应用任务分别完成特定的功能

【bsp】
1.概念：全称“板级支持包”(Board Support Packages)，一段启动代码

2.特点：
- 硬件相关性：因为嵌入式实时系统的硬件环境具有相关性，所以，作为高层软件与硬件之间的接口，BSP必须为操作系统提供操作和控制具体硬件的方法。
- 操作系统相关性：不同的操作系统具有各自的软件层次结构，因此，不同的操作系统具有特定的硬件接口形式

3.功能：
- 单板硬件初始化，主要是CPU的初始化，为整个软件系统提供底层硬件支持
- 为操作系统提供设备驱动程序和系统中断服务程序
- 定制操作系统的功能，为软件系统提供一个实时多任务的运行环境
- 初始化操作系统，为操作系统的正常运行做好准备

4.与bios、EFI区别：
- BIOS：BIOS主要是负责在电脑开启时检测、初始化系统设备（设置栈指针，中断分配，内存初始化）、装入操作系统并调度操作系统向硬件发出的指令。
- BSP是和操作系统绑在一起运行在主板上的，尽管BSP的开始部分和BIOS所做的工作类似，可是大部分和BIOS不同，作用也完全不同。程序员还可以编程修改BSP，在BSP中任意添加一些和系统无关的驱动或程序，甚至可以把上层开发的统统放到BSP中。 而BIOS程序是用户不能更改，编译编程的，只能对参数进行修改设置。更不会包含一些基本的硬件驱动。
- EFI：由于EFI框架比BIOS要大得多，其启动过程也比BIOS要复杂。于BIOS最大的区别就是EFI首先需要EBC虚拟机，然后再启动设备驱动和EFI应用程序，最后通过EFI boot manager加载操作系统引导程序。

知识点10 bootloader

【引导模式】*

1.引导：将操作系统装入内存并开始执行的过程。

2.按时间效率和空间效率不同的要求，分为两种模式：

- 需要BootLoader的引导模式：节省空间，牺牲时间，适用于硬件成本低，运行速度快，但启动速度相对慢
- 不需要BootLoader的引导模式：时间效率高，系统快速启动，直接在NOR flash或ROM系列非易失性存储介质中运行，但不满足运行速度的要求。

【bootloader及其启动过程】*

1.bootloader：嵌入式系统中的OS启动加载程序

2.作用：将操作系统内核从外部存储设备拷贝到内存中，并跳转到内核的首条指令

3.启动过程：

- 初始化硬件，如设置UART（至少设置一个），检测存储器等
- 设置启动参数，告诉内核硬件的信息，如用哪个启动界面，波特率
- 跳转到操作系统的首地址
- 消亡

知识点11 建模**

【有限状态机及其应用】

【有限状态机的实现】

（例题：三次作业）

知识点12 嵌入式系统、CPS的基本组成**

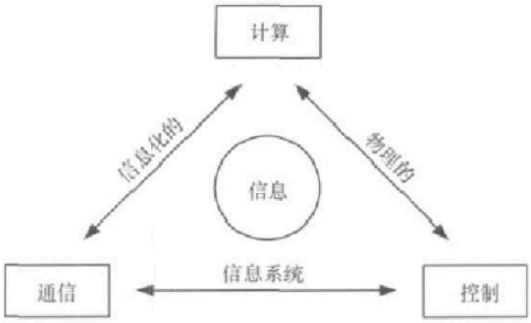
【嵌入式系统的基本组成】

1. 组成：嵌入式硬件和软件
2. 硬件：微处理器为核心集成存储器和系统专用的输入输出设备
3. 软件：初始化代码及驱动、嵌入式操作系统和应用程序有机结合，形成系统特定的一体化软件

【CPS的基本组成】

1.CPS：信息物理系统（Cyber-Physical Systems）
作为计算进程和物理进程的统一体，是集成计算、通信与控制于一体的下一代智能系统。

2.基本组成：传感器、执行器、决策控制单元
嵌入式系统+网络+控制功能



知识点13 嵌入式系统设计**

【嵌入式系统面临挑战】
1. 需要多少硬件
2. 如何满足时限要求
3. 如何处理多项功能在时间上的协调性
4. 如何降低功耗
5. 如何保证系统可靠的工作
6. 如何保证系统可升级

【传统开发过程】
1. 特征：一开始就被划分为软硬件两大部分，软硬件独立开发设计，硬件优先
2. 问题：软硬件交互收受到很大限制，凭经验划分软硬件，软硬件之间的相互性能影响很难评估；系统集成相对滞后，NRE较大
3. 后果：设计质量差，设计修改难，研制周期不能有效保障
4. 基本流程：需求分析，规格说明，体系结构，构件设计，系统集成

【软硬件划分】
1. 嵌入式系统的设计涉及硬件与软件部件，设计中必须决定什么功能由硬件实现，什么功能由软件实现：
- 硬件和软件具有双重性
- 软硬件变动对系统的决策造成影响
- 划分和选择需要考虑多种因素
- 硬件和软件的双重性是划分决策的前提

2. 决定因素：速度、灵活性、开销
- 软件实现的部分：操作系统功能（任务调度、资源管理、设备驱动），协议栈（TCP/IP），应用软件框架
- 硬件实现的部分：基本系统，物理接口，基本逻辑电路
- 双重性部分：算法（加密 / 解密，编码 / 解码，压缩 / 解压），数学运算（浮点运算， FFT）

程序设计部分有一道题，汇编、编译、解释系统的基础知识和基本工作原理

2015年真题A卷

简答题

1、什么是CPS？请画出典型结构，试述典型应用

2、什么是传感器？请写出常用的传感器（4种）

答：传感器是通过物理接口将光、声音、温度等变化转化为具体数值的装置。常用的传感器有声音传感器（声控灯），温度传感器、光传感器、压力传感器

3、嵌入式系统设计中，影响软硬件折衷方案的重要因素有哪些？

答：软硬件的双重性，速度，灵活性和开销，其中软硬件的双重性是考虑的前提

4、为什么嵌入式软件要交叉开发？比较宿主机和目标机的差异

5、如何做到从NAND Flash启动？请描述不同的解决方法

6、选择嵌入式微处理器需要考虑的因素有哪些?
答：体积、重量、可靠性、功耗、成本以及工作温度、抗电磁干扰等方面

7、 什么是看门狗？看门狗有什么作用？工作原理是什么?

8、 嵌入式微处理器有几种类型?

9、 RTOS的引导模式有哪些？试说明各自的特点

10、叙述RTOS的主要特点

问答题
1、给出嵌入式软件轮询模式的伪码描述？描述有何变种或者改进？使用时需要注意的事项有什么？具体应用举例说明。

2、是描述RTOS ucOSII采取哪些策略与机制来确保任务的实时性

3、单CPU，对于下面的进程：

进程	执行时间	最后期限
P1	1	3
P2	1	4
P3	1	12

a.用RMS策略调度这些进程 b.用EDF策略调度这些进程
在每种情况下，计算时间间隔等于进程周期的最小公倍数的调度，时间从t=0开始

设计题
请基于FSM（有限状态机）模型，设计投币公用电话单元。具体需求如下：
1.用户拿起话筒的时候，对呼叫过程进行初始化操作
2.用户拿起话筒之后，还需要投入1元，才能执行通话操作
3.如果电话线路繁忙，那么用户挂断电话之后，硬币会退回给用户
4.如果电话线路畅通，那么允许用户使用的谈话时间为60s，在45s时给出提示音，提醒用户在投入1元才能继续通话
5.如果用户没有再投入1元，那么60s时间用完后，通话终止
6.用户挂上电话之后，系统回到原先的就绪状态，准备接受下一次通话请求
7.如果电话线路出现故障，那么系统处于故障状态
要求：1.分别给出基于FSM和层次型有限状态机的设计
2.上述需求中没有周全的考虑某些细节，请补充并给出修改后的设计
3.实现有几种方式，选择哪种实现？说明原因

2015年真题B卷

简答题

1、试叙述冯诺伊曼体系结构和哈佛体系结构的区别

2、ARM7指令是几级流水线，各阶级执行是什么操作

3、什么是上下文切换？上下文切换都包括哪些步骤?

4、给出有限状态机的定义，有哪两类状态机？两者有何区别？给出至少三种示例应用5、什么是优先级反转？如何解决?

5、比较三类嵌入式处理器，即嵌入式微处理器EMPU、微控制器、MCU、DSP的区别

问答题

1、为什么需要仿真开发？仿真开发具有哪些特点?

2、嵌入式软件系统主要采用的软件体系结构及各自的优缺点（至少5种）

3、假设你现在某从事嵌入式产品开发的公司工作，被要求开发一通用的数字电视接收控制器，即通常所说的“机顶盒”，对于机顶盒的开发，请你结合嵌入式系统的设计流程叙述一下将采取的设计开发流程。（整个流程需包括从产品定义到产品测试和发布）

更早的真题

简答题

1、 什么是嵌入式系统？请说明嵌入式系统各种不同的应用（至少五种）（7分）

2、 请描述嵌入式系统的特点（5分）

3、 嵌入式系统设计中软硬件划分需要考虑哪些因素？举例说明哪些功能通常由软件实现，哪些通常用硬件实现（8分）

4、 嵌入式微处理器与通用处理器之间的区别（5分）

5、 分支指令对流水线性能有何影响？如何解决？（4分）

6、 ARM处理器有哪七种运行模式？（7分）

7、 请比较忙等I/O与中断驱动I/O（4分）

8、 简述Nor Flash和NAND Flash之间的区别（至少五种）（5分）

9、 实时操作系统ucOS-II如何从64个任务扩展到支持256个任务（5分）

10、叙述RTOS的主要特点（5分）

11、画出嵌入式系统存储结构的金字塔图[2014新增]

12、画出嵌入式系统双总线结构[2014新增]

问答题

1、详细描述嵌入式软件的运行过程（10分）

2、比较常用的嵌入式软件体系结构（10分）

3、对于下面的进程，RMS与EDF调度方法，在每种情况下，计算时间间隔等于进程周期的最小公倍数的调度，时间从t=0开始（10分）

进程	执行时间	最后期限
P1	1	3
P2	1	4
P3	2	6

4、说明UCOSII为了实现实时性所做的工作[2014新增]

预测题

1、嵌入式微处理器与通用微处理器之间的区别?

- 体积小、重量轻、可靠性高
- 功耗低
- 成本低：片上存储、引脚与封装、代码密度
- 工作温度、抗电磁干扰、可靠性等方面增强

2、分支指令对流水线性能有何影响？如何解决?

- 分支指令将控制阻滞延时引入到流水线中，也就是我们通常所说的分支损失。是否执行条件分支BNE要等到指令的第三个时钟周期才能确定，因为第三个时钟周期才能计算出分支指令的目标地址。因此就有两个周期浪费在执行路径不明确的指令上。
- 解决办法：引入延时分支，在这种形式的分支指令中，往往有一些直接跟在分支指令后面的指令被执行，无论分支指令是否执行它们都会被执行，这样在分支指令执行期间CPU能够让流水线保持满负载运行。

3、分层状态机的特点?

- 明确的起始状态、提供更多增加状态的方式、减少了所需状态数目

4、FSM实现方式

- switch case
- 状态表：维护一个二维状态表，横坐标表示当前状态，纵坐标表示输入，表中一个元素存储下一个状态和对应的操作。这一招易于维护，但是运行时间和存储空间的代价较大。
- 使用State Pattern

一、考试形式：闭卷
基本概念、基本原理、基本技术

二、题型：
1. 选择题
2. 简答题
3. 程序理解与填空（汇编）
4. 问答题

三、复习
1. 嵌入式系统的定义

“嵌入式系统”实际上是“嵌入式计算机系统”的简称。

IEEE（国际电气和电子工程师协会）的定义：

嵌入式系统是“用于控制、监视或者辅助操作机器和设备的装置”（Devices used to control, monitor, or assist the operation of equipment, machinery or plants）。

可以看出此定义是从应用上考虑的，嵌入式系统是软件和硬件的综合体，还可以涵盖机电等附属装置。

国内普遍被认同的定义：

嵌入式系统是“以应用为中心，以计算机技术为基础，软硬件可裁减，适用于应用系统对功能、可靠性、成本、体积、功耗有严格要求的专用计算机系统”。

嵌入式系统就是一个具有特定功能或用途的隐藏在某种设备中的计算机软硬件集合体，没有固定的特征形状。

特点(自行缩减)

- 嵌入式系统通常是形式多样、面向特定应用的
  - 一般用于特定的任务，其硬件和软件都必须高效率地设计，量体裁衣、去除冗余，而通用计算机则是一个通用的计算平台。
  - 它通常都具有低功耗、体积小、集成度高等特点，能够把通用微处理器中许多由板卡完成的任务集成在芯片内部。
  - 嵌入式软件是应用程序和操作系统两种软件的一体化程序。

- 嵌入式系统得到多种类型的处理器和处理器体系结构的支持
  - 通用计算机采用少数的处理器类型和体系结构，而且主要掌握在少数大公司手里。
  - 嵌入式系统可采用多种类型的处理器和处理器体系结构。
  - 在嵌入式微处理器产业链上，IP 设计、面向应用的特定嵌入式微处理器的设计、芯片的制造已相成巨大的产业。大家分工协作，形成多赢模式。
  - 有上千种的嵌入式微处理器和几十种嵌入式微处理器体系结构可以选择。

- 嵌入式系统通常极其关注成本
  - 嵌入式系统通常需要注意的成本是系统成本，特别是量大的消费类数字化产品，其成本是产品竞争的关键因素之一。
  - 嵌入式的系统成本包括:
    - 一次性的开发成本 NRE(Non-Recurring Engineering)成本
    - 产品成本:硬件 BOM、外壳包装和软件版税等

        - 批量产品的总体成本=NRE 成本+每个产品成本*产品总量
        - 每个产品的最后成本=总体成本/产品总量=NRE 成本/产品总量+每个产品成本

- 嵌入式系统有实时性和可靠性的要求
    - 一方面大多数实时系统都是嵌入式系统
    - 另一方面嵌入式系统多数有实时性的要求，软件一般是固化运行或直接加载到内存中运行，具有快速启动的功能。并对实时的强度要求各不一样，可分为硬实时和软实时。
    - 嵌入式系统一般要求具有出错处理和自动复位功能，特别是对于一些在极端环境下运行的嵌入式系统而言，其可靠性设计尤其重要。
    - 在大多数嵌入式系统的软件中一般都包括一些机制，比如硬件的看门狗定时器，软件的内存保护和重启动机制。

- 嵌入式系统使用的操作系统一般是适应多种处理器、可剪裁、轻量型、实时可靠、可固化的嵌入式操作系统
    - 由于嵌入式系统应用的特点，像嵌入式微处理器一样，嵌入式操作系统也是多姿多彩的。
    - 大多数商业嵌入式操作系统可同时支持不同种类的嵌入式微处理器。可根据应用的情况进行剪裁、配置。
    - 嵌入式操作系统规模小，所需的资源有限如内核规模在几十 KB，能与应用软件一样固化运行。
    - 一般包括一个实时内核，其调度算法一般采用基于优先级的可抢占的调度算法。
    - 高可靠嵌入式操作系统：时、空、数据隔离

- 嵌入式系统开发需要专门工具和特殊方法
    - 多数嵌入式系统开发意味着软件与硬件的并行设计和开发，其开发过程一般分为几个阶段：
        - 产品定义
        - 软件与硬件设计与实现
        - 软件与硬件集成
        - 产品测试与发布
        - 维护与升级
    - 由于嵌入式系统资源有限，一般不具备自主开发能力，产品发布后用户通常也不能对其中的软件进行修改，必须有一套专门的开发环境。
    - 该开发环境包括专门的开发工具（包括设计、编译、调试、测试等工具），采用交叉开发的方式进行，交叉开发环境如图所示。

嵌入式系统发展的 4 个阶段。？？

不知道为什么是 4 各阶段 PPT 上只有三个

如果是 4 个的阶段的就是话嵌入式操作系统

- *无操作系统阶段*
- *简单操作系统阶段*
- *实时操作系统阶段*
- *面向 Internet 的阶段*

- 嵌入式系统的出现和兴起 1960-1970

  出现：20 世纪 60 年代以晶体管、磁芯存储为基础的计算机开始用于航空等军用领域。

  兴起：在 1965～1970 年，当时计算机已开始采用集成电路，即第三代计算机。在军事、航空航天领域、工业控制的需求推动下。

- 嵌入式系统开始走向繁荣，软件和硬件日臻完善 1971-1989

  嵌入式系统大发展是在微处理器问世之后

  单片机、DSP 出现

  软件技术的进步使嵌入式系统日臻完善

- 嵌入式系统应用走向纵深 1990 年-现在

  进入 20 世纪 90 年代, 在分布控制、柔性制造、数字化通信和数字化家电等巨大需求的牵引下，嵌入式系统的硬件、软件技术进一步加速发展、应用领域进一步扩大。

  嵌入式系统的硬件

  嵌入式系统的软件

第一阶段：以四位到八位单片机为核心的可编程控制器系统，同时具有检测伺服指示设备相配合的功能。
第二阶段：以八位和十六位嵌入式中央处理器为基础，以简单操作系统为核心的嵌入式系统。
第三阶段：以三十二位 RISC 嵌入式中央处理器加嵌入式操作系统为标志的嵌入式系统。
第四阶段：以基于 INTERNET 接入为表示的嵌入式系统。

分类。

- 按嵌入式处理器的位数来分类
  - 4 位嵌入式系统
  - 8 位嵌入式系统
  - 16 位嵌入式系统
  - 32 位嵌入式系统
  - 64 位嵌入式系统
- 按应用来分类
  - 信息家电
  - 移动终端
  - 工业控制
  - 汽车电子
  - 通信类
- 按速度分类
  - 强实时系统, 其系统响应时间在毫秒或微秒级。

- 一般实时系统，其系统响应时间在几秒的数量级上,其实时性的要求比强实时系统要差一些。
- 弱实时系统，其系统响应时间约为数十秒或更长。这种系统的响应时间可能随系统负载的轻重而变化。

- 按确定性来分类

  根据确定性的强弱，可将嵌入式系统分为硬实时、软实时系统：
  - 硬实时：系统对系统响应时间有严格的要求，如果系统响应时间不能满足，就要引起系统崩溃或致命的错误。
  - 软实时：系统对系统响应时间有要求，但是如果系统响应时间不能满足，不会导致系统出现致命的错误或崩溃。
- 按嵌入式系统软件复杂程度来分类
  - 循环轮询系统
  - 有限状态机系统
  - 前后台系统
  - 单处理器多任务系统
  - 多处理器多任务系统

2. 嵌入式系统的基本结构。

嵌入式系统一般由嵌入式硬件和软件组成

硬件以微处理器为核心集成存储器和系统专用的输入/输出设备

软件包括：初始化代码及驱动、嵌入式操作系统和应用程序等，这些软件有机地结合在一起，形成系统特定的一体化软件。

3. 嵌入式系统的设计方法。

系统定义（需求分析）

软硬件划分

结构规划 – 处理器类型，软硬件之间的接口类型，等.

划分目的 – 满足系统速度,延迟，体积,成本等方面的要求.

划分策略 - high level partitioning by hand, automated partitioning using various techniques, etc.

调度

Operation scheduling in hardware

Instruction scheduling in compilers

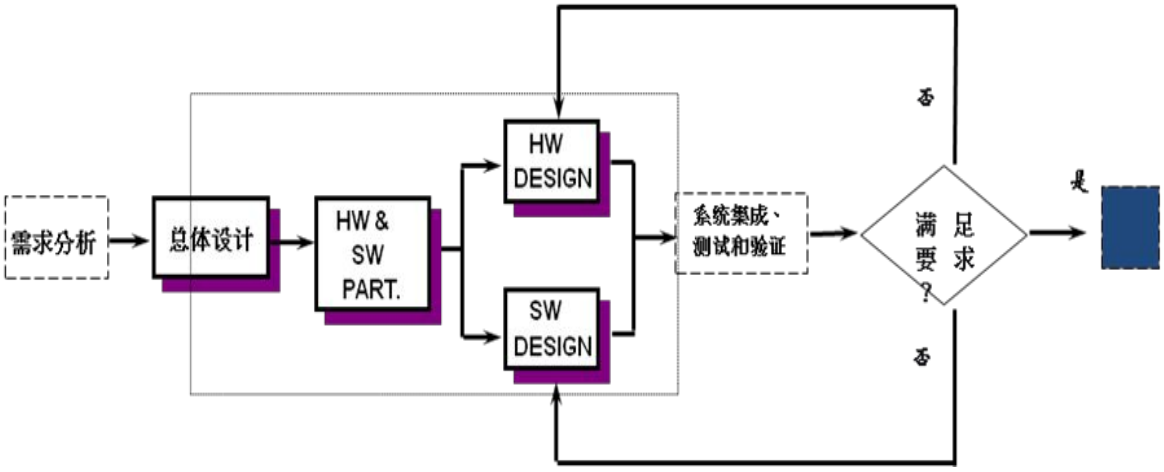
Process scheduling in operating systems

软硬件设计过程中的建模

软硬件协同的设计方式

- 传统的嵌入式系统设计方法
- 软硬件协同设计

- 传统的嵌入式系统设计方法



- 软硬件协同设计

H 软硬件协同设计定义

- The meeting of system-level objectives by exploiting the trade-offs between hardware and software in a system through their concurrent design

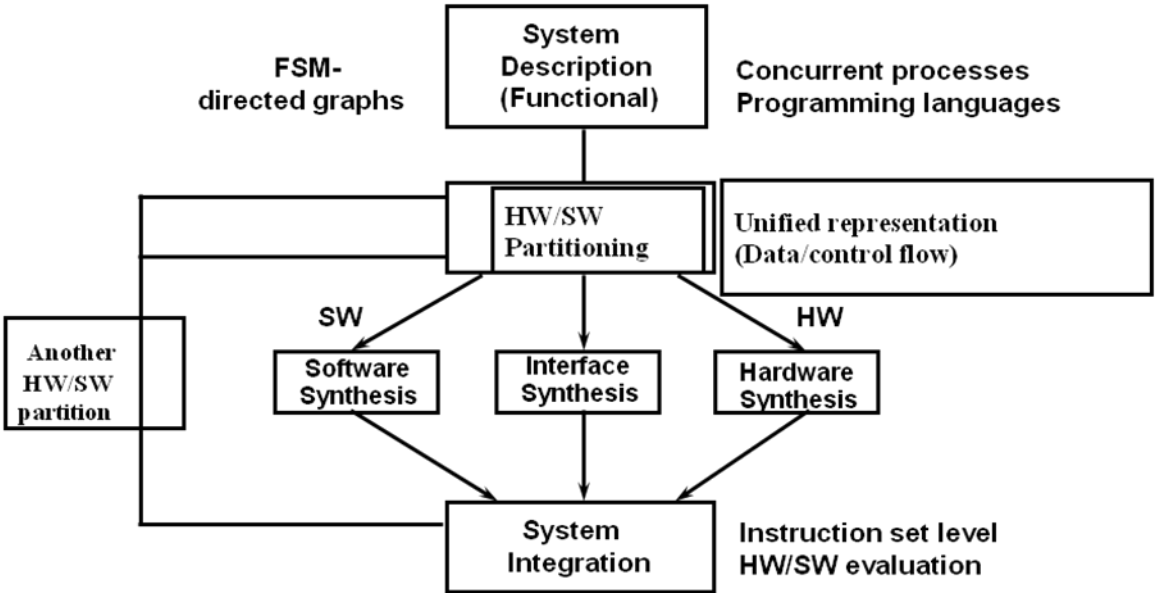
H 主要概念

- Concurrent（并发）： hardware and software developed at the same time on parallel paths
- Integrated（一体化）: interaction between hardware and software developments to produce designs that meet performance criteria and functional specifications

软硬件协同设计的基本需求

- 统一的软硬件描述方式
  - 软硬件支持统一的设计和分析工具（技术）
  - 允许在一个集成环境中仿真（评估）系统软硬件设计
  - 支持系统任务在软件和硬件设计之间的相互移植
- 交互式软硬件划分技术
  - 允许多个不同的软硬件划分设计进行仿真和比较
  - 辅助最优系统实现方式决策
  - 将软硬件划分应用到模块设计，以便最佳地实现系统的设计指标。
    Partitioning applied to modules to best meet design criteria (功能和性能目标)
- 完整的软硬件模型基础
  - 支持在设计过程中的几个阶段的综合评价
  - 支持软硬件逐步的开发和集成
- 正确的验证方法
  - 确保系统设计达到的目标要求

典型的软硬件协同设计过程



4. 嵌入式硬件系统基础。

嵌入式系统的硬件是以嵌入式微处理器为核心，主要由嵌入式微处理器、总线、存储器、输入/输出接口和设备组成。

- 嵌入式微处理器
- 总线
- 存储器
- 输入/输出接口和设备

嵌入式微处理器体系结构：

诺伊曼体系结构和哈佛体系结构。

嵌入式微处理器体系结构可采用冯•诺依曼（Von Neumann）结构或哈佛（Harvard）结构

- 传统的微处理器采用的冯·诺依曼结构将指令和数据存放在同一存储空间中，统一编址，指令和数据通过同一总线访问。
- 哈佛结构则是不同于冯·诺依曼结构的一种并行体系结构，其主要特点是程序和数据存储在不同的存储空间中，即程序存储器和数据存储器是两个相互独立的存储器，每个存储器独立编制、独立访问。与之相对应的是系统中设置的两条总线（程序总线和数据总线），从而使数据的吞吐率提高了一倍。

RISC 和 CISC。基本架构。

嵌入式微处理器的指令系统可采用精简指令集系统 RISC（Reduced Instruction Set Computer）或复杂指令集系统 CISC（Complex Instruction Set Computer）

	CISC	RISC
价格	由硬件完成部分软件功能，硬件复杂性增加，芯片成本高	有软件完成部分硬件功能，软件复杂性增加，芯片成本低
性能	减少代码尺寸，增加指令的执行周期数	使用流水线降低指令的执行周期数，增加代码尺寸
指令集	大量的混杂指令集，有简单快速的指令，也有复杂多周期指令，符合 HLL	简单的但周期指令，在汇编指令方面有相应的 CISC 微代码指令
高级语言支持	硬件完成	软件完成
寻址模式	复杂的寻址模式，支持内存到内存寻址	简单的寻址模式，仅允许 LOAD 和 STORE 指令存取内存，其它所有的操作都基于寄存器到寄存器
控制单元	微码	直接执行
寄存器数目	寄存器较少	寄存器较多

嵌入式系统中的总线、

H 嵌入式系统的总线一般集成在嵌入式微处理器中。

H 从微处理器的角度来看，总线可分为片外总线(如：PCI、ISA 等)和片内总线（如：AMBA、 AVALON、OCP、WISHBONE 等）。

H 选择总线和选择嵌入式微处理器密切相关，总线的种类随不同的微处理器的结构而不同。

存储设备（种类，优缺点）。

H 嵌入式系统的存储器包括主存和外存。

H 大多数嵌入式系统的代码和数据都存储在处理器可直接访问的存储空间即主存中。

H 系统上电后在主存中的代码直接运行。主存储器的特点是速度快，一般采用 ROM、EPROM、Nor Flash、SRAM、DRAM 等存储器件。

H 目前有些嵌入式系统除了主存外，还有外存。外存是处理器不能直接访问的存储器，用来存放各种信息，相对主存而言具有价格低、容量大的特点。

H 在嵌入式系统中一般不采用硬盘而采用电子盘做外存，电子盘的主要种类有

NandFlash、 SD （Secure Digital）卡、CompactFlash、SmartMedia、Memory Stick、MultiMediaCard、、 DOC（Disk On Chip）等。

5. ARM 体系结构；

看书 P81

ARM 汇编语言。

重点、难点：

CPSR、（当前程序状态寄存器）

31	30	29	28	27	26…8	7	6	5	4	3	2	1	0
N	Z	C	V	Q	保留	I	F	T	M4	M3	M2	M1	M0

每种异常模式都有一个 SPSR。异常出现时，SPSR 用于保留 CPSP 的状态。格式相同。

1、条件码标志位\状态标志位：N、Z、C、V（保存 ALU 中的当前操作信息）

N：正负号/大小 标志位

- 0 表示：正数/大于；1 表示：负数/小于

Z：零标志位

- 0 表示：结果不为零；1 表示：结果为零

C：进位/借位/移出位

- 0 表示：未进位/借位/移出 0；1 表示：进位/未借位/移出 1

V：溢出标志位

- 0 表示：结果未溢出；1 表示：结果溢出

Q：DSP 指令溢出标志位（用于 v5 以上 E 系列）

0 表示：结果未溢出；1 表示：结果溢出

（1）选择“S”后缀问题：

指令中可以选择 s 后缀，以影响状态标志。但是比较指令（cmp、cmn、tst 和 teq）不需要后缀 S，它们总会直接影响 CPSR 中的状态标志。

（2）对 CPSR 中标志位的影响：

- N 标志位：如果结果为负，则 N 标志位置 1 ； 否则清 0 。
- Z 标志位：如果结果为 0 ，则 Z 标志位置 1 ； 否则清 0 。
- C 标志位：如果是加、减运算指令或比较指令时， C 标志位设置为 A L U 的进位输出；否则设置为移位器的移位输出。如果不需要移位，则 C 保持不变。
- V 标志位：在非加减操作中， V 标志位保持原值。在加减操作中，如果有溢出，则置 1 ； 不发生溢出，则清 0 。

（3）关于恢复 CPSR 原值问题：

- 如果指令带有 S 后缀（除了比较指令以外），同时又以 PC 为目标寄存器进行操作，
- 在异常模式下：则操作的同时从 SPSR 恢复 CPSR。比如：
- movs pc, #0xff /* cpsr = spsr; pc = 0xff */
- adds pc, r1, #0xffffff00
- /* cpsr = spsr; pc = r1 + 0xffffff00 */
- ands pc, r1, r2 /* cpsr = spsr; pc = r1 & r2; */
- 在 user 或者 system 模式：会产生不可预料的结果，因为在这两种模式下

没有 SPSR。

-

寻址模式

有 7 种寻址方式：

- 立即寻址：操作数本身在指令中给出
- 寄存器寻址：操作数=寄存器中的数值
- 寄存器间接寻址：寄存器为操作数的地址指针
- 基址寻址：操作数有效地址=基址寄存器的值+指令中给出的位移量
- 堆栈寻址 、
- 块拷贝寻址、
- 相对寻址：操作数有效地址=基址寄存器的值（当前程序计数器）+指令中的地址字码段

处理器模式（7 种）

模式	模式描述
用户 User	ARM 微处理器正常的程序执行状态
快速中断 FIO	用于高速数据传输或通道处理
外部中断 IRQ	用于通用的中断处理
管理 Super Visor	操作系统使用的保护模式
数据访问中止 Abort	当数据或指令预取终止时进入该模式，可用于虚拟存储及存储保护
系统 System	运行特权操作系统任务
未定义 Undifined	支持硬件协处理器的软件仿真

处理器状态（2 种）

ARM 状态：32 位，执行字对准的 ARM 指令。

Thumb 状态：16 位，执行半字对准的 Thumb 指令；

ARM 微处理器在开始执行代码时，应该处于 ARM 状态。

进入 Thumb 状态：当操作数寄存器的状态位（位 0）为 1 时，可以采用执行 BX 指令的方法，使微处理器从 ARM 状态切换到 Thumb 状态。此外，当处理器处于 Thumb 状态时发生异常（如 IRQ、FIQ、Undef、Abort、SWI 等），则异常处理返回时，自动切换到 Thumb 状态。

进入 ARM 状态：当操作数寄存器的状态位为 0 时，执行 BX 指令时可以使微处理器从 Thumb 状态切换到 ARM 状态。此外，在处理器进行异常处理时，把 PC 指针放入异常模式链接寄存器中，并从异常向量地址开始执行程序，也可以使处理器切换到 ARM 状态。

一些基本的汇编语句，汇编语言程序中一些常用的代码段。

6. 嵌入式软件系统体系结构。

驱动层

- 板级初始化程序
- 与系统软件相关的驱动

- 与应用软件相关的驱动
- 与应用软件相关的驱动不一定需要与操作系统连接，这些驱动的设计和开发由应用决定。

操作系统层

- 操作系统层包括嵌入式内核、嵌入式 TCP/IP 网络系统、嵌入式文件系统、嵌入式 GUI 系统和电源管理等部分。
- 其中嵌入式内核是基础和必备的部分，其他部分要根据嵌入式系统的需要来确定。

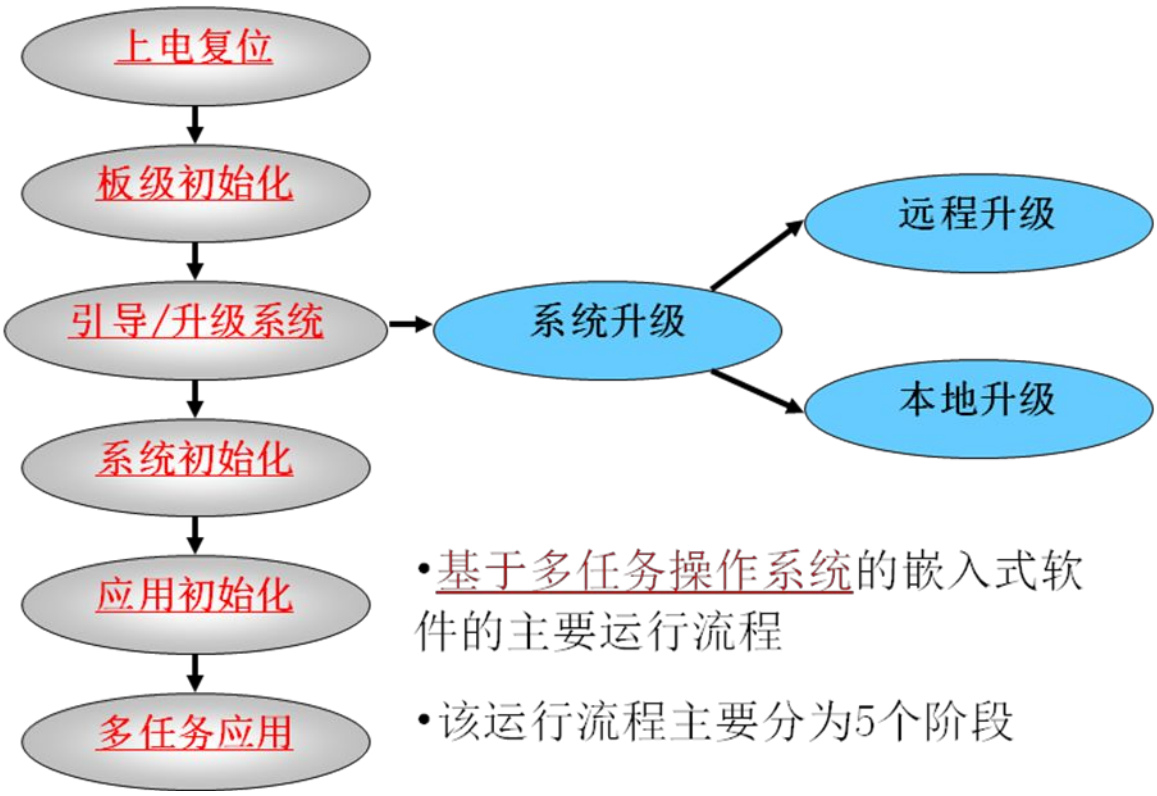
中间件层

- 目前在一些复杂的嵌入式系统中也开始采用中间件技术，主要包括嵌入式 CORBA、嵌入式 Java、嵌入式 DCOM 和面向应用领域的中间件软件。
- 如基于嵌入式 CORBA 的应用于软件无线电台的应用中间件 SCA（Software Core Architecture）等。

应用层

- 应用层软件主要由多个相对独立的应用任务组成
- 每个应用任务完成特定的工作，如 I/O 任务、计算的任务、通信任务等，由操作系统调度各个任务的运行。

嵌入式软件运行流程。



◘ 上电复位、板级初始化阶段

- 嵌入式系统上电复位后完成板级初始化工作。
- 板级初始化程序具有完全的硬件特性，一般采用汇编语言实现。不同的嵌入式系统，板级初始化时要完成的工作具有一定的特殊性，但以下工作一般是

必须完成的：

- CPU 中堆栈指针寄存器的初始化。
- BSS 段（Block Storage Space 表示未被初始化的数据）的初始化。
- CPU 芯片级的初始化：中断控制器、内存等的初始化。

◘ 系统引导/升级阶段

- 根据需要分别进入系统软件引导阶段或系统升级阶段。
- 软件可通过测试通信端口数据或判断特定开关的方式分别进入不同阶段。

◘ 系统引导阶段

- 系统引导有几种情况：
- 将系统软件从 NOR Flash 中读取出来加载到 RAM 中运行：这种方式可以解决成本及 Flash 速度比 RAM 慢的问题。软件可压缩存储在 Flash 中。
- 不需将软件引导到 RAM 中而是让其直接在 NorFlash 上运行，进入系统初始化阶段。
- 将软件从外存（如 NandFlash、CF 卡、MMC 等）中读取出来加载到 RAM 中运行：这种方式的成本更低。

◘ 系统升级阶段

- 进入系统升级阶段后系统可通过网络进行远程升级或通过串口进行本地升级。
- 远程升级一般支持 TFTP、FTP、HTTP 等方式。
- 本地升级可通过 Console 口使用超级终端或特定的升级软件进行。

◘ 系统初始化阶段

- 在该阶段进行操作系统等系统软件各功能部分必需的初始化工作，如根据系统配置初始化数据空间、初始化系统所需的接口和外设等。
- 系统初始化阶段需要按特定顺序进行，如首先完成内核的初始化，然后完成网络、文件系统等的初始化，最后完成中间件等的初始化工作。

◘ 应用初始化阶段

在该阶段进行应用任务的创建，信号量、消息队列的创建和与应用相关的其它初始化工作。

◘ 多任务应用运行阶段

各种初始化工作完成后，系统进入多任务状态，操作系统按照已确定的算法进行任务的调度，各应用任务分别完成特定的功能

7. 实时系统的定义。

◘ 实时系统定义：

IEEE 对实时系统的定义是“哪些正确性不仅取决于计算的逻辑结果，也取决于产生结果所花费的时间的系统”。

◘ POSIX 1003.b 作了这样的定义：指系统能够在限定的响应时间内提供所需水平的服务

8. 嵌入式操作系统的体系结构

嵌入式内核的功能。

◘ 任务管理

- 内核的核心部分，具有任务调度、创建任务、删除任务、挂起任务、解挂任务、设置任务优先级等功能。
- 通用计算机的操作系统追求的是最大的吞吐率，为了达到最佳整体性能，其调度原则是公平，采用 Round-Robin 或可变优先级调度算法，调度时机主要以时间片为主驱动。
- 而嵌入式操作系统多采用基于静态优先级的可抢占的调度，任务优先级是在运行前通过某种策略静态分配好的，一旦有优先级更高的任务就绪就马上进行调度。

◘ 内存管理
- 嵌入式操作系统的内存管理比较简单。
- 通常不采用虚拟存储管理，而采用静态内存分配和动态内存分配（固定大小内存分配和可变大小内存分配）相结合的管理方式。
- 有些内核利用 MMU 机制提供内存保护功能。
- 通用操作系统广泛使用了虚拟内存的技术，为用户提供一个功能强大的虚存管理机制。

◘ 通信、同步和互斥机制
- 这些机制提供任务间、任务与中断处理程序间的通信、同步和互斥功能。
- 一般包括信号量、消息、事件、管道、异步信号和共享内存等功能。
- 与通用操作系统不同的是，嵌入式操作系统需要解决在这些机制的使用中出现的优先级反转问题。

◘ 中断管理，一般具有以下功能：
- 安装中断服务程序
- 中断发生时，对中断现场进行保存，并且转到相应的服务程序上执行
- 中断退出前，对中断现场进行恢复
- 中断栈切换
- 中断退出时的任务调度

◘ 时间管理
- 提供高精度、应用可设置的系统时钟，该时钟是嵌入式系统的时基，可设置为十毫秒以下。
- 提供日历时间，负责与时间相关的任务管理工作如任务对资源有限等待的计时、时间片轮转调度等，提供软定时器的管理功能等。
- 通用操作系统的系统时钟的精度由操作系统确定，应用不可调，且一般是几十个毫秒。

◘ 任务扩展功能
- 任务扩展功能就是在内核中设置一些 Hook 的调用点，在这些调用点上内核调用应用设置的、应用自己编写的扩展处理程序，以扩展内核的有关功能。
- Hook 调用点有任务创建、任务切换、任务删除、出错处理等。

常用的嵌入式操作系统。

μC/OS-II

嵌入式μCLinux

RTLinux

Windows CE 操作系统

VxWorks
Nucleus
Symbian
Palm OS

9. 板级支持包 BSP 与 bootloader。

BSP 的概念、

BSP 全称“板级支持包”（Board Support Packages），说的简单一点，就是一段启动代码，和计算机主板的 BIOS 差不多，但提供的功能区别就相差很大。PPT

BSP 是嵌入式系统的基础部分，也是实现系统论可移植的关键。它负责上电时的硬件初始化、启动 RTOS 或应用程序模块、提供底层硬件驱动，为上层软件提供访问底层硬件的手段。BSP 针对具体的目标板设计，其结构和功能虽目标版的不同呈现较大的差异。

BSP 特点与功能。

H 硬件相关性

因为嵌入式实时系统的硬件环境具有应用相关性，所以，作为高层软件与硬件之间的接口，BSP 必须为操作系统提供操作和控制具体硬件的方法。

H 操作系统相关性

不同的操作系统具有各自的软件层次结构，因此，不同的操作系统具有特定的硬件接口形式。

bootloader 的概念，

bootloader 是在操作系统内核运行之前执行的一段小程序，它将操作系统内核从外部存储设备拷贝到内存中，并跳转到内核的首条指令

嵌入式系统引导加载过程。

操作系统引导概念：将操作系统装入内存并开始执行的过程。

按时间效率和空间效率不同的要求，分为两种模式：

- 需要 BootLoader 的引导模式：节省空间，牺牲时间，适用于硬件成本低，运行速度快，但启动速度相对慢
  不需要 BootLoader 的引导模式：时间效率高，系统快速启动，直接在 NOR flash 或 ROM 系列非易失性存储介质中运行，但不满足运行速度的要求

10. 嵌入式开发。

嵌入式系统的调试技术。

H 交叉调试方式

- Crash and Burn
- Rom Monitor
- Rom Emulator
- In Circuit Emulator
- On Chip Debugging

H Simulator 方式（非交叉）

交叉调度	非交叉调度
调试器和被调试程序运行在不同的计算	调试器和被调试程序运行在同一台计算机

机上	上
可独立运行，无需操作系统支持	需要操作系统的支持
被调试程序的装载有调试器完成	被调试程序的装载有专门的 Loader 程序完成
需要通过外部通信的方式来控制被调试程序	不需要通过外部通信的方式来控制被调试程序
可以直接调试不同指令集	只能直接调试相同指令集的程序

理解实验流程。

理解交叉编译；

把在宿主机上编写的高级语言程序编译成可以运行在目标机上的代码，即在宿主机上能够编译生成另一种 CPU（嵌入式微处理器）上的二进制程序。

理解下载并运行程序的不同方法。

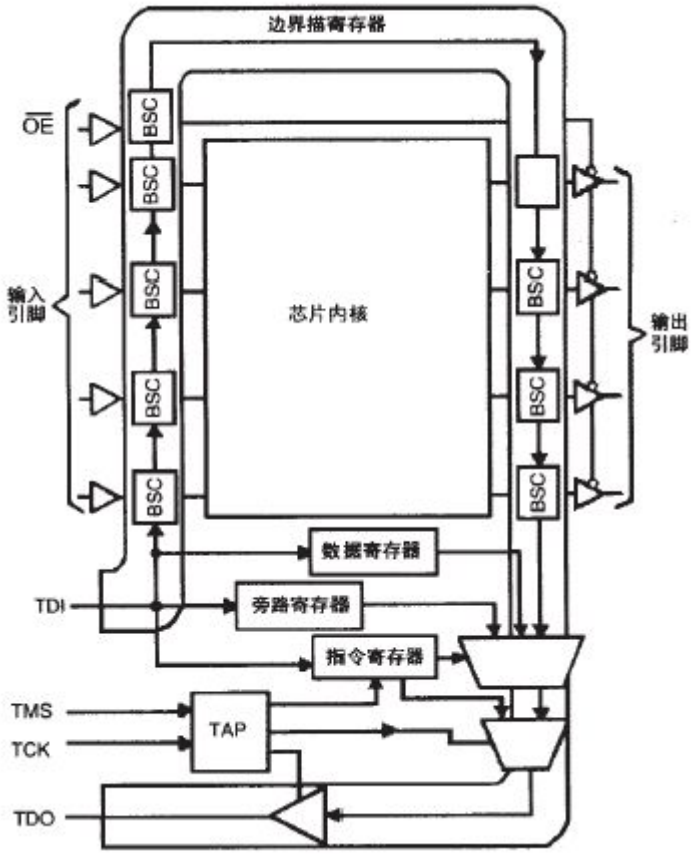
JTAG 接口。

- JTAG（Joint Test Access Group）（主流方式）

JTAG——标准测试访问接口与边界扫描结构（Standard Test Access Port and Boundary Scan Architecture），已被 IEEE1149.1 标准所采纳，是面向用户的测试接口。

该接口一般由 4 个引脚组成：

- 测试数据输入（TDI）
- 测试数据输出（TDO）
- 测试时钟（TCK）
- 测试模式选择引脚（TMS）
- 异步测试复位引脚（TRST，可选）



11. ucOS 原理：

任务调度

多任务操作系统的核心工作就是任务调度。

所谓调度，就是通过一个算法在多个任务中确定该运行的任务，做这项工作的函数就叫做调度器。

μC/OS_II 进行任务调度的思想是“近似地每时每刻总是让优先级最高的就绪任务处于运行状态”。为了保证这一点，它在系统或用户任务调用系统函数及执行中断服务程序结束时总是调用调度器，来确定应该运行的任务并运行它 。

μC/OS_II 进行任务调度的依据就是任务就绪表

根据就绪表获得待运行任务的任务控制块指针

处理器的 SP=任务块中保存的 SP

恢复待运行任务的运行环境

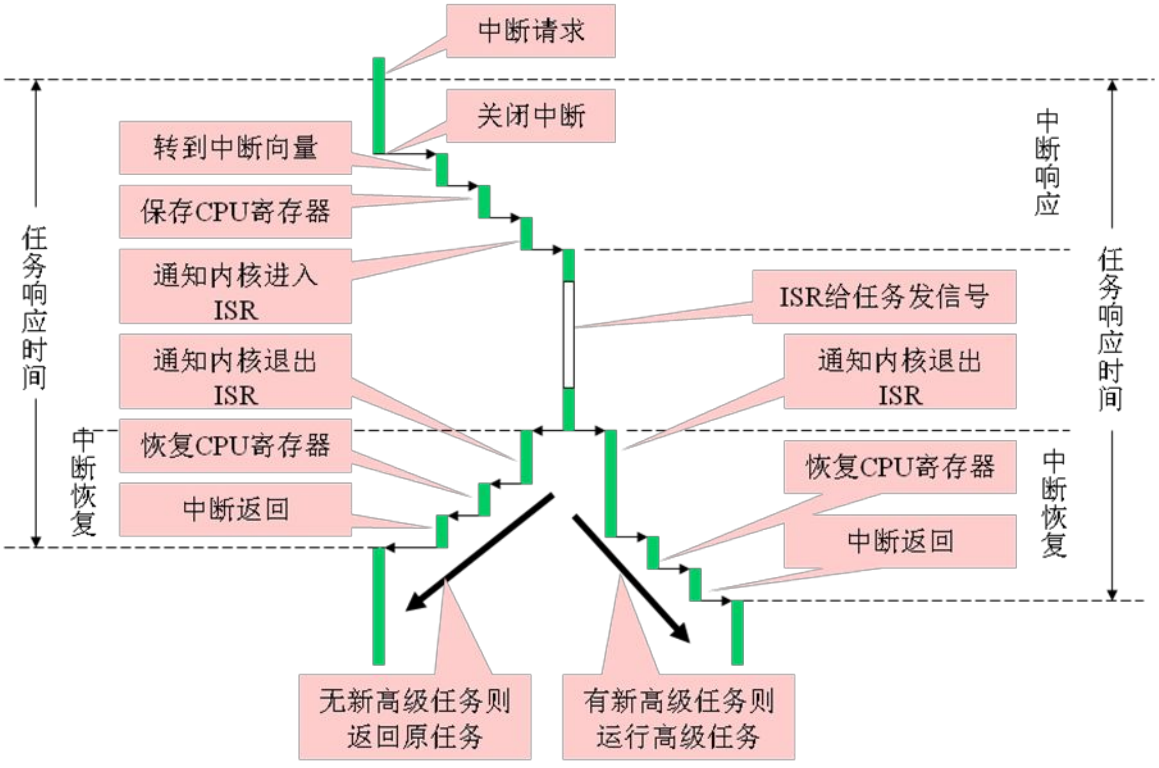
处理器的 PC=任务堆栈中的断点地址

其实，调度器在进行调度时，在这个位置还要进行一下判断：究竟是待运行任务是否为当前任务，如果是，则不切换；如果不是才切换，而且还要保存被中止任务的运行环境

中断与时钟

μC/OS-II 系统响应中断的过程为：

系统接收到中断请求后，这时如果 CPU 处于中断允许状态（即中断是开放的），系统就会中止正在运行的当前任务，而按照中断向量的指向转而去运行中断服务子程序；当中断服务子程序的运行结束后，系统将会根据情况返回到被中止的任务继续运行或者转向运行另一个具有更高优先级别的就绪任务。



中断的响应过程

同步与通信

系统中的多个任务在运行时，经常需要互相无冲突地访问同一个共享资源，或者需要互相支持和依赖，甚至有时还要互相加以必要的限制和制约，才保证任务的顺利运行。因此，操作系统必须具有对任务的运行进行协调的能力，从而使任务之间可以无冲突、流畅地同步运行，而不致导致灾难性的后果。

与人们依靠通信来互相沟通，从而使人际关系和谐、工作顺利的做法一样，计算机系统是依靠任务之间的良好通信来保证任务与任务的同步的。

存储管理