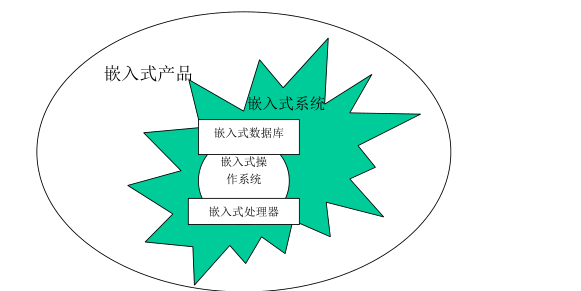
**1． 园环套图概念关系：嵌入式处理器/嵌入式操作系统/嵌入式数据库/嵌入式系**

**统/嵌入式产品相互关系**

嵌入式微处理器：指具备强的中断、I/O、内存和能耗管理能力、具有定制多类体系构架的特征，适当计算处理性能的微处理器



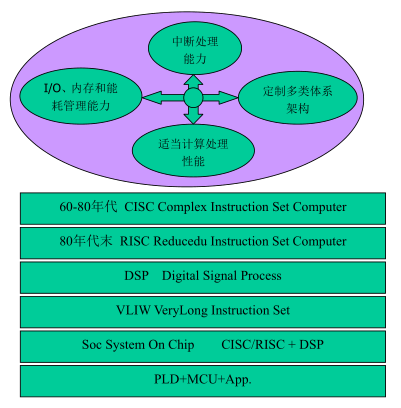
关系：嵌入式产品包含了嵌入式系统，但并不是嵌入式系统。嵌入式系统包含嵌入式数据库、嵌入式操作系统、嵌入式处理器，其中嵌入式操作系统是嵌入式系统的核心，嵌入式处理器是硬件基础，嵌入式数据库是嵌入式系统的上层应用。

1. **依据下图对比讨论嵌入式系统五层结构与可裁剪性的关系？**

五层结构可裁剪，用以对嵌入式系统进行抽象、概括和描述，方便对系统进行实验、分析与预测，并且作为一个框架，用来协调各层标准的制定。嵌入式系统的五层结构分别是应用层、UI层、内核层、引导驱动层和硬件层。

最底层的硬件层包括供电、时钟、处理器、总线，还有内存、键盘、显示、接口部分。引导驱动层下面是引导及固件，上面则由驱动作为内核的接口。硬件和引导驱动层共同构成了最小系统。再上一层是内核层，包括相对底层的OS内核部分，它连接着内核与驱动。也包括文件、网络、任务等。UI层为使用者界面，包括SUI、文件、网络、任务。而应用层则是与消费者最接近的一层，是由各种应用程序需组成。

1. **依据下图如何理解嵌入式处理器内涵，及其指令系统分类？**



上图从应用角度给出嵌入式微处理器的定义：指具备强中断、I/O、内存和能耗管理能力，具有定制多类体系架构特征，适当计算处理性能的微处理器。而这些功能、特性由中间的指令系统支持，微处理器与指令系统密不可分。

指令系统从产生到现在的形态主要有五种风格；复杂指令集（CISC），精简指令集（RISC），超长指令字（VLIW），数据信号处理指令集（DSP），专用指令集（ASIP）。

CISC是一种为了便于编程和提高记忆体访问效率的晶片设计体系，常用指令只占20%，效率不高。

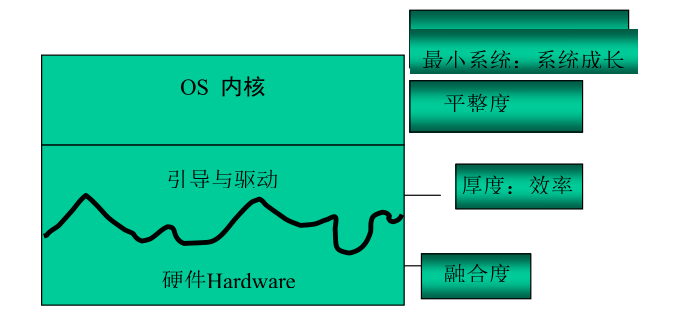
RISC是为了提高处理器运行速度而设计的晶片设计体系。关键技术在于流水线（pipelining）操作：在一个时钟周期里完成多条指令。提高了执行速度和可靠性，成本较低，但对编译器要求较高。

VLIW由编译器将多条可同时发送的指令并排在一条超长指令字中，并行执行多个操作，扩展了指令并行度。

DSP面向特殊应用，具有独特的体系结构，其芯片采用改进的哈佛结构，指令系统为流水线操作，数字信号处理可通过软件修改处理参数，灵活性强。具有良好的多机并行运行特性，采用专用硬件乘法器，电压较低。

ASIP的核心思想是，针对特定应用设计专用的指令集，再根据设计好的指令集优化处理器的体系机构，使优化后的体系结构更适合执行专用指令，进一步提高指令执行效率。

**4． 锯齿融合图：引导与驱动的缓冲与分割原则解释**



1.启动前操作系统仅仅是存储在不挥发介质中的代码，从启动的一刻开始，操作系统将被装载到内存中，成为完整功能的运行系统。这个装载的过程就是引导。引导是系统启动和运转的第一步，是系统从硬件走向系统软件的开始，从这一刻开始，系统将从固化的代码和静态的电路转化为动态运行的程序和具有功能的系统。

2.为了保证系统功能可以正常调用，所做的初始化准备就是驱动。

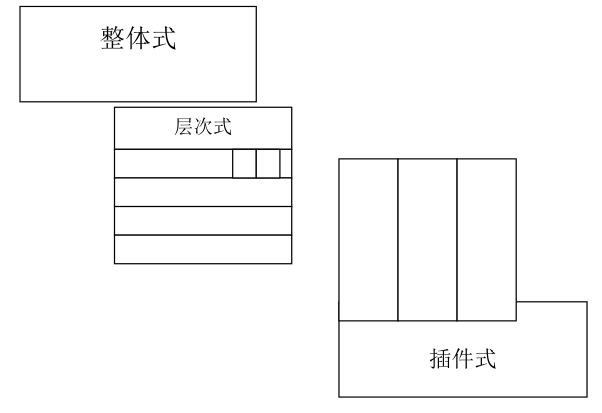
3.顺利进行引导与驱动两部分工作就是引导层要做的工作。

融合度是指引导层对于下级的硬件层控制和兼容能力。由于硬件的参差不齐，所以融合度不同，才有了引导与驱动这一层，厚度是指引导层的规模。一般来说，引导层的不稳定性与引导层的规模成正比。平整度是指引导层对上级的中间件层所提供的操作接口的统一性和友好性。

引导层的作用之一就是屏蔽掉底层硬件的差异性，为上层的操作系统和各种程序调用提供统一的接口。如果引导层任务太多，虽然会向上提供一个平整的接口，但效率会明显下降。如果引导层太薄，效率会提高，但设备无关性会降低，应结合具体应用分析。引导层的平整度、厚度和融合度是衡量引导层的重要指标。

最小系统分两种，一是OS内核，引导和驱动，而是引导和驱动。它是一个含有微处理器的系统。具有制造功能和用户功能所需的最少硬件及软件环境，是嵌入式系统能工作的最低要求，不具有智能性，是系统成长的基础。

**5． 嵌入式操作系统典型结构构造图：整体、层次、插件描述层次与微内核OS的特征？**

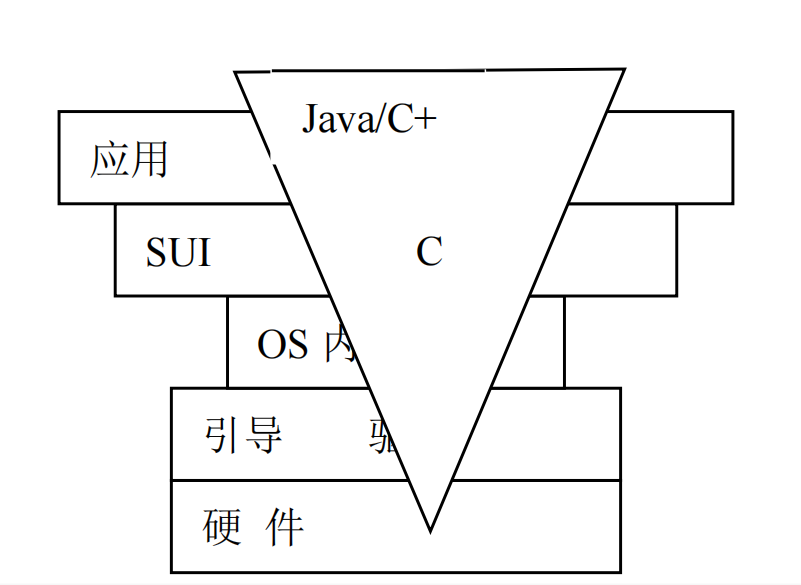


整体内核结构（单内核，大内核），是一种几乎涵盖了所有操作系统功能模块（如进程调度、进程间通信、内存管理、设备管理、文件系统、网络系统）的内核结构。大部分模块遵守特定的接口规范，相互协调。所有模块在编译时链接在一起，形成一个可执行文件。运行时所有操作系统功能模块处于内核态，而其他用户程序和系统程序处于用户态，通过系统调用可以切换到内核态。但调试困难，难于裁剪，稳定性相对较差。如WinCE就是采用此结构。

插件式内核结构（微内核结构），是一种类似于C/S模式的内核结构。微内核运行在核心态，提供所有操作系统的基本操作。结构规范，易于裁剪和编程，且调试简单。扩展性、移植性、鲁棒性好，但整体效率低，资源管理复杂。如QNX就是采用此结构。

层次式内核结构是一种基于分层思想的内核结构。它把操作系统的所有功能划分为若干模块，按功能流程的调用次序排列成若干层。（下层模块封装内部细节并向上提供统一的抽象和接口，上层模块调用下层模块提供的接口，各个层级间的实现比较独立。各个模块间的组织结构和依赖关系清晰明确，适合进行系统功能的划分）分层思想有利于组织操作系统的开发，易于调试和诊断，系统的修改和扩展相对容易。但是对系统功能进行定制和裁剪相对困难，且它是一个理论化的结构，实际中是没有完全符合层次的操作系统。

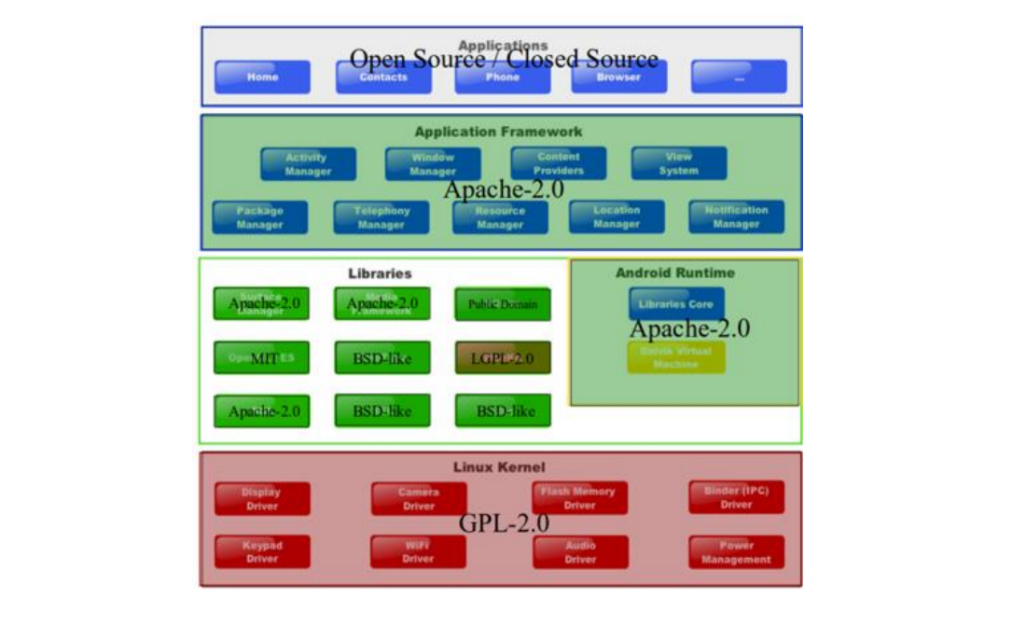
1. **依据此图描述系统五层结构与编程语言关系**



嵌入式系统的五层结构中与用户最近的一层就是第五层应用层。而应用层则是由各种程序组成的，这些程序又出自于各种不同的程序设计语言，比如Java、C+、C、汇编语言等。

该图把程序及语言放在五层结构的上面，意思是对于普通的用户和消费者而言，这些才是看得到用得到的部分，它们遮盖住了嵌入式系统的复杂结构，是嵌入式系统与用户的统一接口。任何系统的根本目的都是为了对数据进行处理，而这种处理正是通过程序这种特殊的数据来实现的。如何保证应用程序及数据的开放和兼容性，是行业面临的基础问题。因此最理想的情况下是软件及程序能够具有非常好的兼容和移植性，能够使用户看不出复杂的底层结构上的差别。

7.**以Android系统中开源软件使用为例，说明其中使用的几种软件许可证？进而说明软件许可证交叉使用的行业价值**

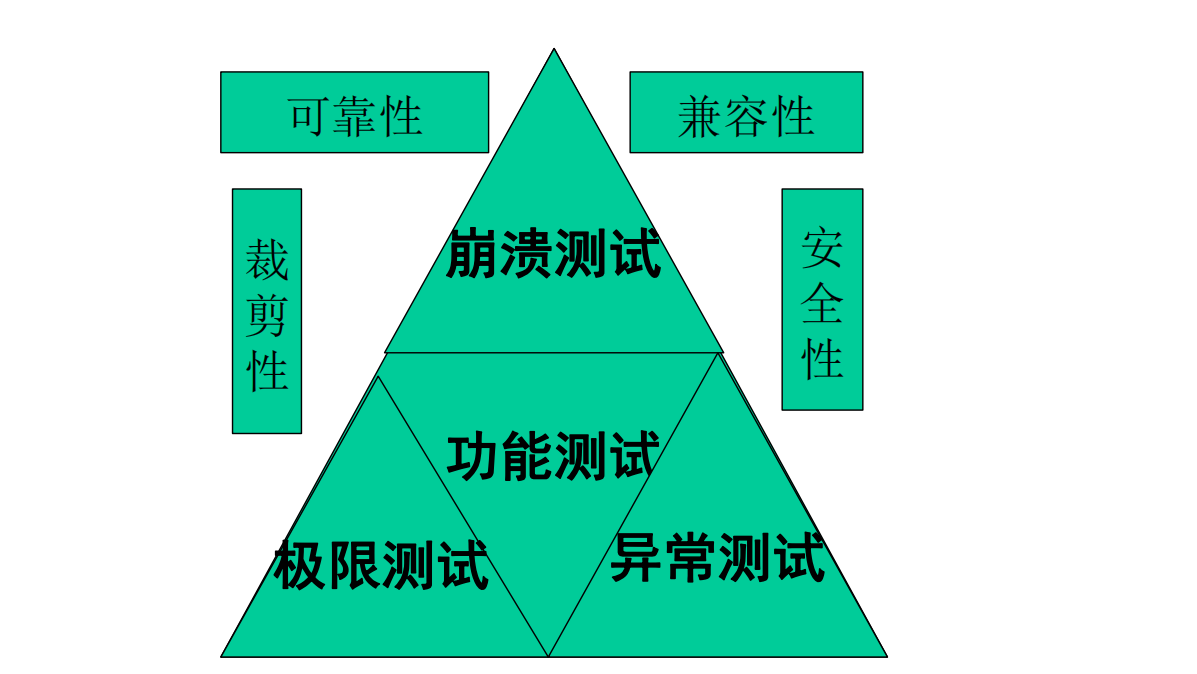


**Apache许可证:** 即Apache License，是一个由Apache软件基金会(ASF)发布的自由软件许可证。Apache许可证最初为Apache Web服务器而撰写,Apache许可证在Apache社区内外被广泛使用；Apache基金会下属所有项目都使用Apache许可证；许多非Apache基金会项目也使用了Apache许可证。

**GPL许可证：**GNU通用公共许可协议，是被广泛使用的自由软件许可证，给予了终端用户运行、学习、共享和修改软件的自由。GPL是一个Copyleft许可证，这意味着只要项目的某个部分（如动态链接库）以GPL发布，则整个项目以及派生作品只能以相同的许可条款分发，这与宽松自由软件许可证有所区别。GPL是第一个普遍使用的Copyleft许可证。根据GPL许可的优异自由软件程序的例子有Linux内核和GNU编译器集合（GCC）。

**软件许可证的最大行业价值是让自由开源软件既能独立发展，又能与私有、商业软件有效对接，共同发展，解决了发展的排他问题。**

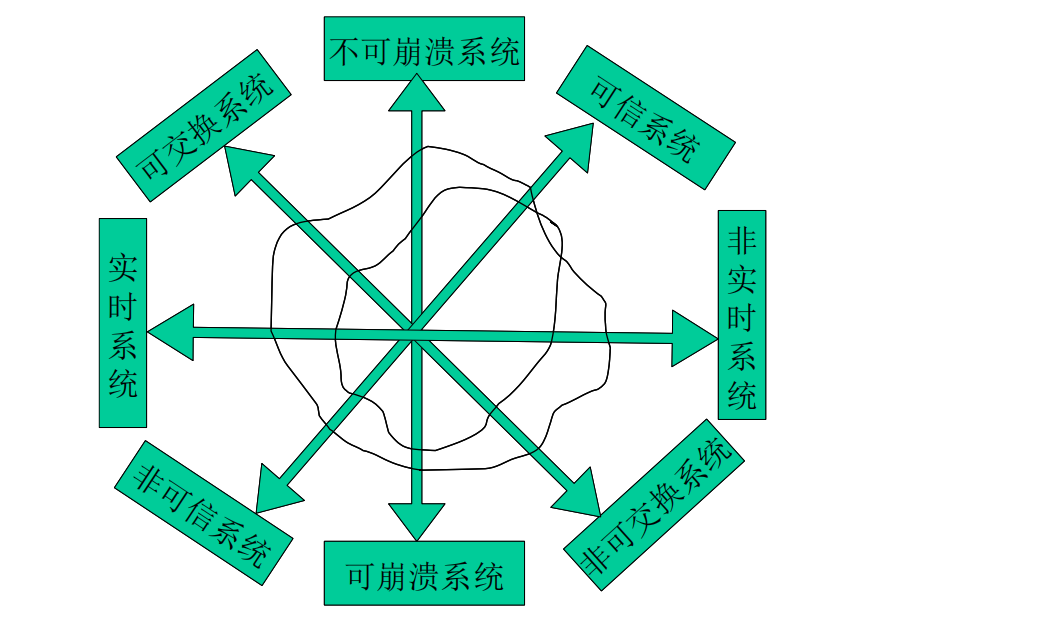
8.**依据此图说明嵌入式系统测试种类及相关内涵？**



该图中嵌入式系统的测试的种类有：功能测试、极限测试、崩溃测试、异常测试，以及可靠性、裁剪性、兼容性、安全性测试。

相关内涵为：功能测试是嵌入式系统测试的最基本部分，同时崩溃测试、极限测试、异常测试也是非常重要的测试环节。而且可靠性测试可以测试系统的可靠程度，裁剪性测试则测试出了系统的裁剪性，兼容性测试可看出系统的兼容程度，而异常测试则是测试系统是否会发生异常。只有这样才能尽量使系统具有最好的稳定性。

1. **从系统辩证的角度，分析嵌入式系统分类？**

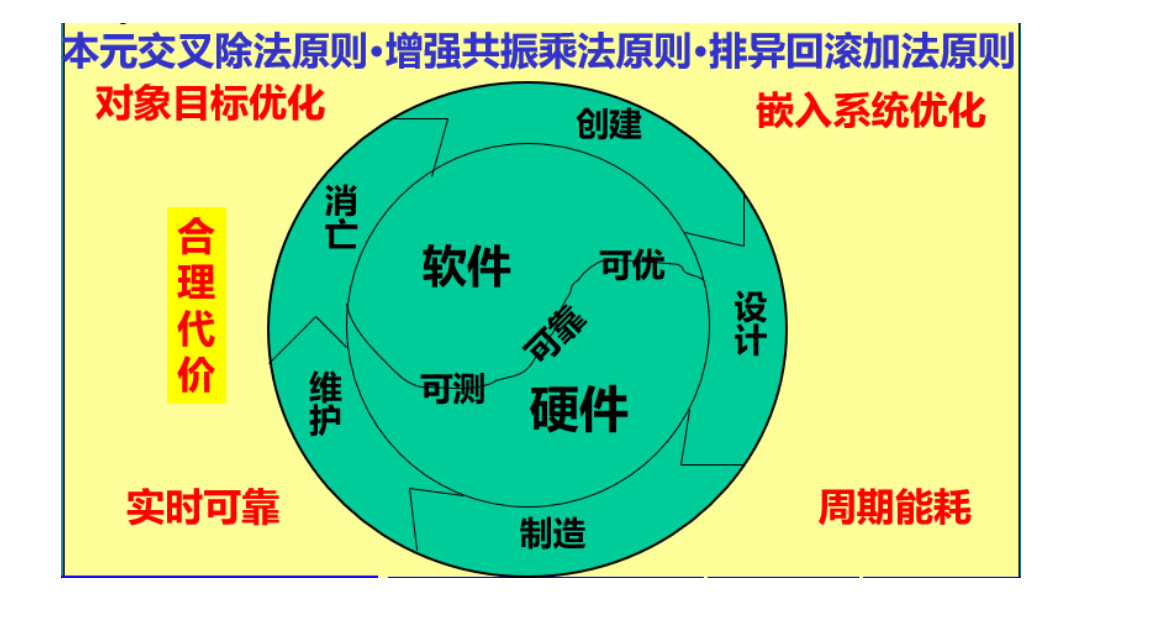


此图为设计结构的八角图，也称为蜘蛛图。

根据嵌入式系统的特性可以分类为：可信系统与非可信系统，可交换系统与非可交换系统，实时系统与非实时系统，可崩溃系统与不可崩溃系统四种。任何一个操作系统都不可能同时满足任何对角的系统特性，这就必须要根据实际的需求来选择各个属性的程度，来组建一个最符合用户需求的嵌入式系统。在实际的工作中，往往也需要靠经验和对规律的总结来设定到底采用哪些系统方式。

每条对角线线的端点是分类中对立特性的两种系统。对一个嵌入式操作系统，在资源相同的条件下，只能在对立特性的要求上权衡，对于不同的嵌入式系统，就产生了不同的形状，如图中的两个环，代表两个不同的系统，每一种系统在对立的两个性能指标间所占的比例不同，体现了两种系统的差异。

1. **依据下图总结本课程的主题内容关联关系？由此解释嵌入式系统的生命周期？**



该图类似于太极阴阳图，是对于整个嵌入式系统的高度概括和总结。

中间的硬件和软件是两个不同的部分，但又通过某些方式交织在一起，互相对立，又互相包含。外层则是围绕着整个嵌入式系统的软硬件的一种动态的流程与方法，从创建到设计，再由设计到制造，由制造到维护，由维护又回到创建。如此循环往复，周而复始，使得整个系统动静结合，得以不断发展完善，以此得到了嵌入式系统的生命周期。