操作系统简介

类别	典型用途	需求与指标	典型案例
桌面操作系统	个人计算机	交互性好、运行流畅、生态完善 适合娱乐和轻度生产力	Windows MacOS
云操作系统	服务器集群	资源利用效率高、信息安全性好	Linux
移动操作系统	手机、平板	低功耗、低资源占用 前后台综合优化	Android iOS
物联网操作系统	小家电等	极低资源占用 良好的网络连接能力 一定程度的信息安全性	Zephyr RT-Thread
实时操作系统	轨道交通等	极高的实时性 极高的功能安全性 极高的容错性	FreeRTOS μC/OS ThreadX
安全操作系统	政府国防等	极高的信息安全性 自主可控 可审计性	seL4 统信uOS

操作系统历史

1. 手工操作阶段

人机速度矛盾

2. 批处理阶段

按照一定顺畅，逐一执行事先编组好的成批计算任务

GM—NAA I/O GM NAA

单道批处理：优点：缓解上述矛盾，缺点：资源利用率低

多道批处理：优点：并发执行，资源利用率高，缺点：不提供人机交互能力

3. 分时操作系统

同时装入多个任务，每个任务分配一定时间和空间运行

CTSS MIT

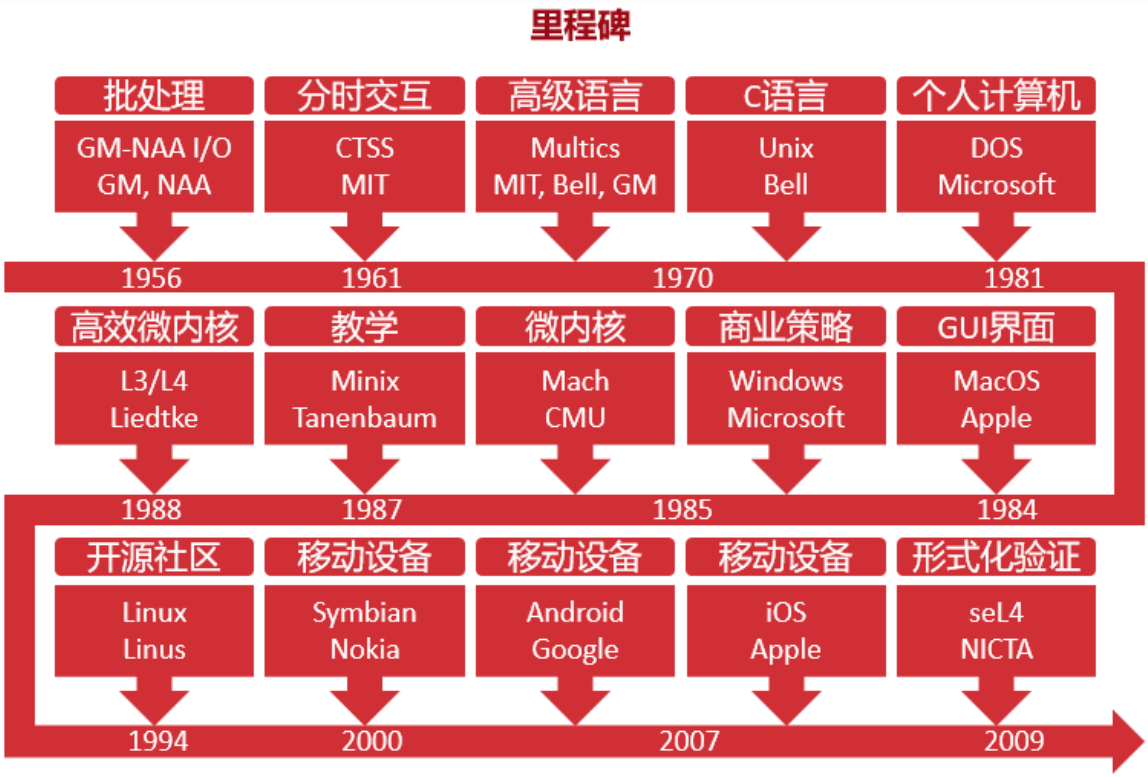
优点：提供人机交互

缺点：不能有限处理紧急任务

4.实时操作系统

5. 网络操作系统核分布式计算机系统

6. 个人计算机操作系统



操作系统中的抽象概念

原则 操作系统的最高设计目标，通常由操作系统所在的领域的需求决定。

设计 操作系统的高层次抽象描述，描述各个组件的抽象功能。

实现 操作系统的低层次具体描述，描述各个组件的具体实施。

策略 操作系统中对资源进行管理的方法和政策。

机制 操作系统为实现策略使用的工具和手段

图灵机模型

题目：1进制加法

通用图灵机

特点：一台可以将任意图灵机及其输入作为输入的图灵机，他可以模拟任何图灵机的运行

重要性：

存储程序架构（冯诺依曼模型）思想的原点

存储的程序即为其他图灵机的状态描述和转换规则

存储的数据即为其他图灵机所使用的数据

输入：纸带上的状态描述和转换规则以及所使用的数据

输出：纸带的内容

计算机系统

计算机系统=软件+硬件

软件——运行，组织管理维护机电设备和物理机制所使用的程序。

包括**系统软件**和**用户软件**

硬件——机电设备物理机制，

主要包括运算器，控制器，存储器（前面三个算微处理器），输入设备和输出设备

处理器性能

晶体管数（越多越好）

制程（线路宽越小越好）

主频（越高越好，越高越热）

IPC（每时钟周期执行的指令数）

MIPS（每秒执行百万指令数）

基准测试分数

字长

寻址空间

价格

处理器和总线

数据总线宽度

外部数据总线宽度

地址总线宽度

外部地址总线宽度

控制总线

CPU流水

取指——>译码——>执行——>回写

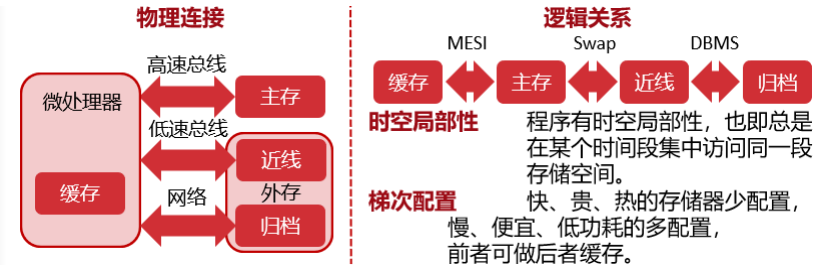
存储器

频率，延迟，容量，易失性，

用途：存放程序，数据

存储器	原理	速度	功耗	易失性	价格	用途
SRAM	触发器	最快	极高	易失	极贵	缓存，便笺
DRAM	电容	快	高	易失	贵	主存
Flash	电荷注入	较快	低	非易失	较贵	外存，高速

存储器	原理	速度	功耗	易失性	价格	用途
磁盘	磁化	慢	低	非易失	便宜	外存，近线
磁带	磁化	极慢	极低	非易失	极便宜	外存，归档



数据在存储器中的表示：一般而言，一个存储单元就是一个字节

大段字节序，小段字节序

指令集架构

指令——用编码表示CPU的一种操作

指令系统——全部指令集

CISC复杂指令集

RISC精简指令集

掌握指令的具体功能包括几个方面：

- 助记符
- 操作数
- 操作
- 性能

汇编语言有助记符和操作数作为组成汇编码的基本描述

指令的六大分类

分类	英文	指令数	代表指令	用途
数据传送	Data Transfer	25	MOV	改变数据的存储位置
算术运算	Arithmetic	21	ADD	进行加减乘除等运算
位操作	Bit Manipulation	21	AND	进行按位运算与位移
串操作	String Manipulation	7	MOVS	处理字符串
控制转移	Control Transfer	47	JMP	转移程序控制流
处理机控制	Processor Control	15	CLI	设置处理器状态

AX累加器（Accumulator）

BX基地址寄存器（Base）

CX计数寄存器（Counter）

DX数据暂存器（Data）

SI源变址寄存器（Source Index）

DI目标变址寄存器（Destination Index）

BP目的基址指示器（Base Pointer）

SP堆栈指针（Strack Pointer）

IP用于指令地址记录，自增1

FLAGS程序状态指示（PSW）

段式内存管理

分段

逻辑地址（偏移量或有效地址）

线性地址（物理地址）

CS代码段寄存器

DS数据段寄存器

SS堆栈寄存器

ES附加段寄存器

地址生成算法：线性地址=（段寄存器值<<4）+逻辑地址

线性地址=段起始地址+逻辑地址

寻址方式

数据传送寻址

- 存储器寻址
- 直接寻址
- 间接寻址
- 相对寻址
- 基址变址寻址
- 相对基址变址寻址

控制转移寻址

- 段内直接寻址
- 段内间接寻址
- 段间直接寻址
- 段间间接寻址

端口寻址

- 直接寻址

- 间接寻址

寻址方式分类：寄存器存志，存储器寻址，立即数寻址

设备和中断

输入输出设备：

外部设备：人机交互，机间通讯

外部结构：包括三个寄存器：数据寄存器，状态寄存器，命令寄存器

信息交换：四种方式：直接交换，查询（轮询），中断，成组传送，（中断相对而言是主流方式）

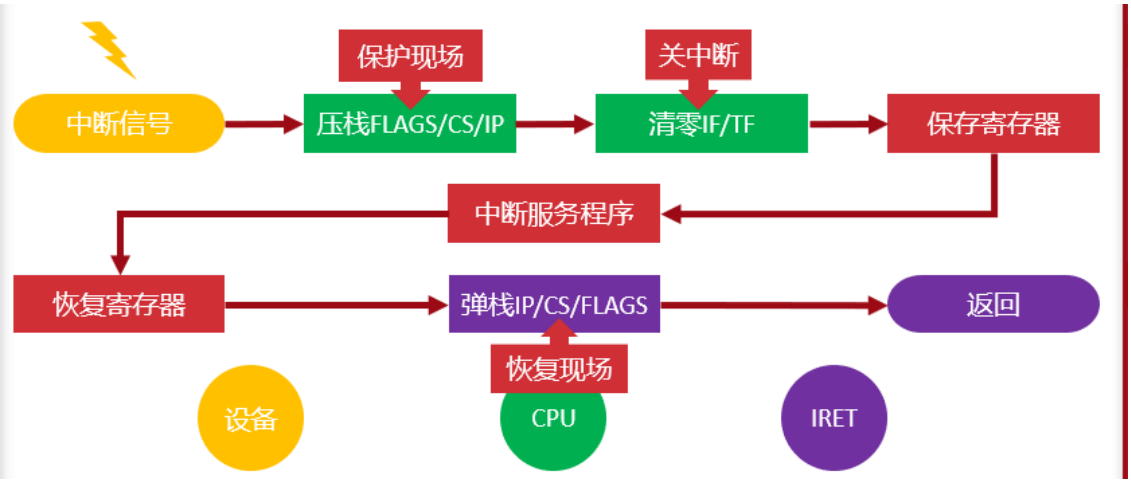
编制方式：独立编址，统一编址

中断输入输出

特点：设备准备好接收操作时用中断通知CPU，CPU调用事先准备好的一段程序来处理输入输出

中断服务程序（ISR）（中断例行程序，中断向量）

8086中断响应流程



中断向量表：

CPU查询中断向量的入口时使用的表格，

两种方式访问：

1. 由用户程序直接读写对应的中断向量
2. 使用DOS系统调用

中断嵌套：

软件中断：

软中断指令：当几乎所有处理器都提供软件触发中断指令

异常捕获：当程序中的指令执行发生错误，就会抛出异常，异常会被同步捕获，引起异常的指令并不会被执行，压栈的IP等于该异常指令的地址，并且CPU窑先跳转到异常处理程序处执行网，再回来视图重新执行该指令。

陷阱设置：系统调用时陷阱的一种特殊情况

特权级与系统调用

实现隔离机制：

软件实现和硬件实现的优缺点：

软件：

需要额外的代码检查并限制程序行为，会拖累程序的执行效率，增加程序的资源开销。

硬件：

需要额外的硬件电路检查并限制程序行为，会增加系统的固定造价。

处理器特权级

硬件特权级：CPU硬件实现，分为用户模式（用户态）和内核模式（内核态）

用户模式：执行应用程序的模式，不允许直接访问系统的敏感资源

内核模式：执行操作系统的模式，可直接执行敏感指令和访问敏感资源

特权指令：能对系统中敏感资源进行操作的指令，仅能再内核模式下执行

处于内核态时，说明正在运行的是内核程序，即可执行特权指令，又可执行非特权指令。
处于用户态时，说明正在运行的是应用程序，只能执行非特权指令

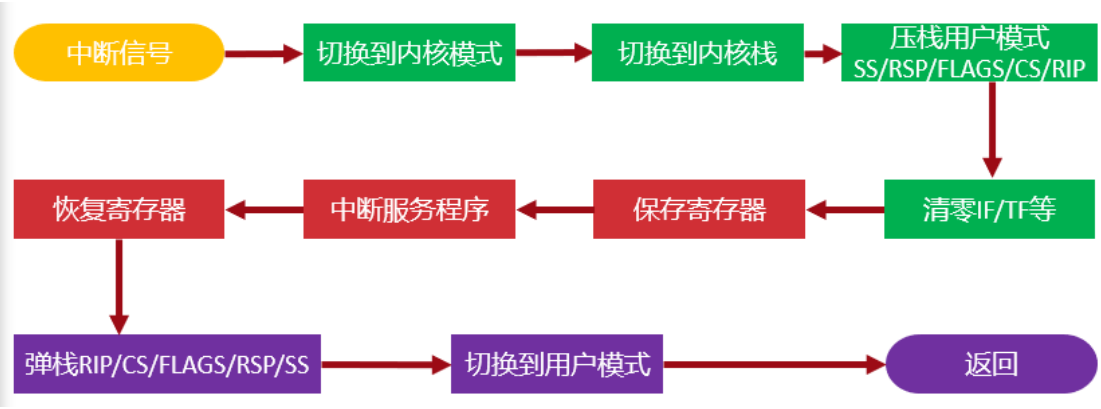
问：特权级可以通过什么机制互相切换?
方法一；中断机制

方法二：专用指令机制

问：用户程序使用什么方法调用操作系统

使用上述两种机制之一，之后陷入操作系统内核，执行被操作系统接管

中断机制引起的AMD64特权级切换



问：相比8086，为何AMD64要切换到内核栈且多压栈了SS和RSP?

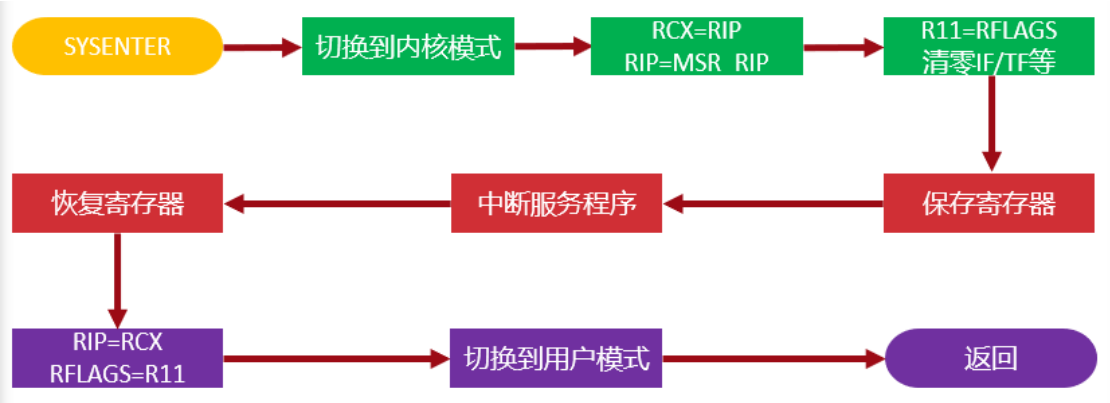
答：进入内核模式时，必须修改FLAGS/CS/RIP以关中断而且跳转到中断服务程序。我们可以把它们的原值保存在栈上。但此时的 SS:RSP是指向用户空间的不可信的值，不能在内核模式对它进行操作，否则一旦应用程序有恶意就导致内核出错。

这就导致进入中断时，CPU必须切换到内核的SS:RSP，才能压栈 FLAGS/CS/RIP。但这会使原来的用户模式SS:RSP丢失。为了避免 产生此问题，CPU会将用户态的SS:RSP压在内核栈上。

问：其他解决方案?

答：提供专用的寄存器来存放FLAGS/CS/RIP。内核进入中断时，不操 作栈，而是将原用户模式下的FLAGS/CS/RIP放入专用寄存器。少了一步访问内存的栈操作，进入中断服务速度快。

特殊指令（SYSCALL/SYSRET）机制引起的AMD64特权级切换



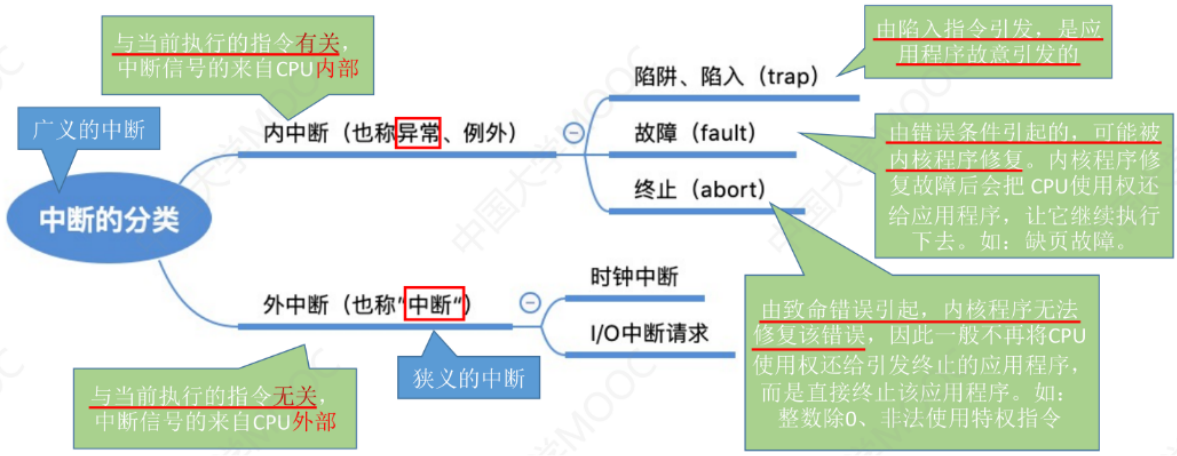
这种操作省去栈操作，

系统调用

系统调用是操作系统给 **编程人员/应用程序** 使用的接口，可以理解为一种可供应用程序调用的特殊函数，应用程序可以通过系统调用来请求获得操作系统内核的服务。

中断

通常具有不可预测性。系统调用若使用外部中断来做具体实现， 为了和设备的外部中断处理共享一套代码，往往需要在中断服 务程序（内核模式）中保存和恢复全部的上下文。



陷阱

具备绝对的可预测性。用户软件（用户模式）可以承担一部分上下文保存和恢复的责任

SYSCALL SYSRET

用户程序只要确保SYSCALL之前RCX和R11寄存器没有使用就好。如果有使用到，那么可以先将它们压栈，再在SYSRET返回用户态 后，从栈里面把它们弹出来。

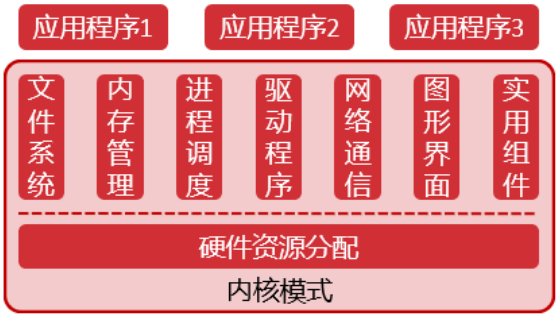
操作系统结构

1.库结构

特点：**没有内核模式和用户模式的区别**，所有应用程序和内核都在同一个保护域中，应用程序可以随时对任何资源做任何操作，应用程序间为合作干系，操作系统的角色片中协调而非管理

适用场景：设备功能简单，预算低的场合，eg运动手环，无人机

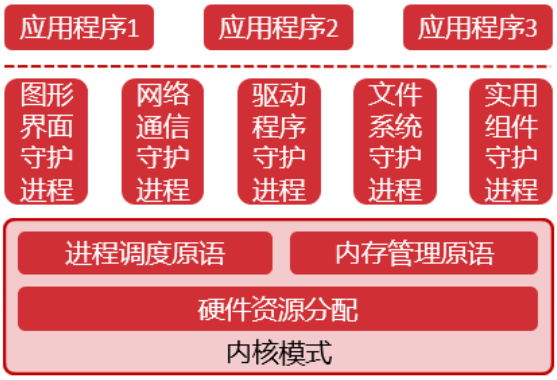
2.宏内核结构



特点：**有内核模式和用户名是的区分**，每个应用程序在不同保护域，内核所有功能位于同一个保护域，应用程序必须请求内核完成敏感资源操作，应用程序间为合作或竞争挂安心，操作系统的协调和管理并重。

适用场合：桌面计算等复杂度和性能要求适中的常规应用场合最广泛适用的内核结构之一。eg笔记本

3. 微内核结构



特点：**有内核模式和用户模式的区分**，每个应用程序在不同的保护域，内核除基本功能外，其他功能分别位于不同的而用户模式进程中，应用程序必须**请求守护进程中的策略**分配敏感资源，守护进程则转而适用**内核提供的机制**完成这些分配操作。

强调机制和策略的高度分离

适应场合：高性能，高可靠性或高灵活性计算等应用场合。嵌入式和非标领域最广泛使用的内核结构之一。抗软件故障和攻击，操作系统的一部分损坏不会影响其他部分。

4. 外核结构

特点：**有内核模式和用户模式的区分**，每个应用程序在不同保护域，内核仅负责硬件资源的安全分配与管理功能。应用程序必须自行和被分配的硬件资源打交道完成功能。

去抽象化：将硬件资源直接暴漏给应用程序以获得最大性能增益。

适用场合：高性能或该灵活性计算等应用场合，现在被微内核挤压，因为微内核也是越做越小，和外核已经非常相似。与外核相似，现代微内核分内存分配和进程调度也可以以极高的效率放到用户态甚至应用程序中去。

5. 其他结构

虚拟机结构：

用一份物理硬件模拟出积分虚拟硬件，在每个虚拟硬件上可以运行一个单独的操作系统实列。不同的操作需要实列间在不同的保护域之中，相互隔离。

准虚拟化：模拟的硬件和真实硬件有区别。需要修改客户机操作系统的底层

全虚拟化：模拟的硬件和真实硬件无差别。最常用。

多内核结构

多核机器堪称互相通信的单核机器。每个CPU核上运行一个操作系统内核，内核间通信完成整个功能，