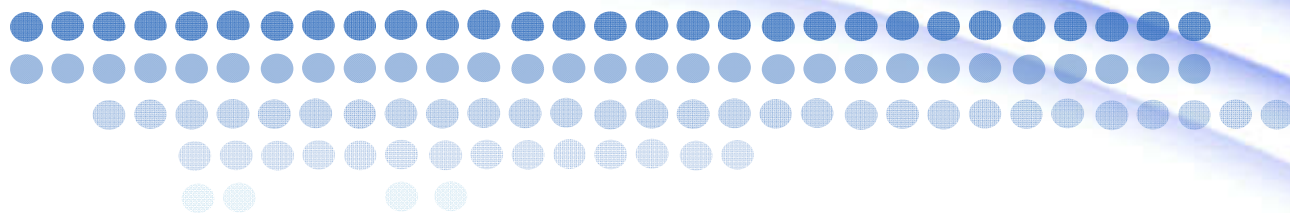


# 2021级期末考试复习版本



《操作系统原理》

## 第2章 操作系统依赖的基础硬件

教师：邹德清，李珍，苏曙光

华中科技大学网安学院

2023年10月-2024年01月

## 第2章 操作系统依赖的基础硬件

- 主要内容

- 1.CPU的态
- 2.中断机制
- 3.基本输入输出系统/BIOS
- 4.操作系统启动过程
- 5.操作系统的生成

- 重点

- CPU的态
- 中断响应的过程
- 操作系统初始引导过程





## 2. 1 CPU与CPU的态

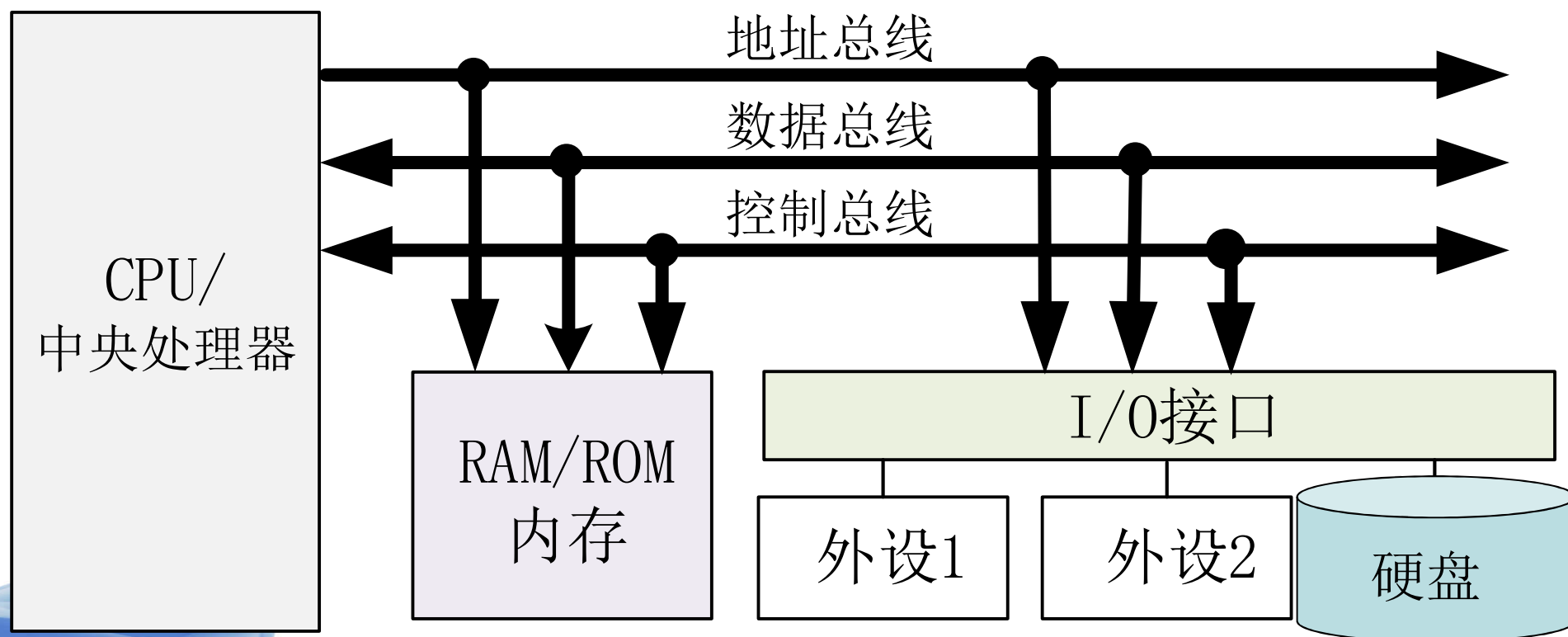
# 计算机三总线结构

- 计算机主要部件

- CPU、内存和外设

- 三总线

- 地址总线 | 数据总线 | 控制总线



# 操作系统依赖的最基本硬件

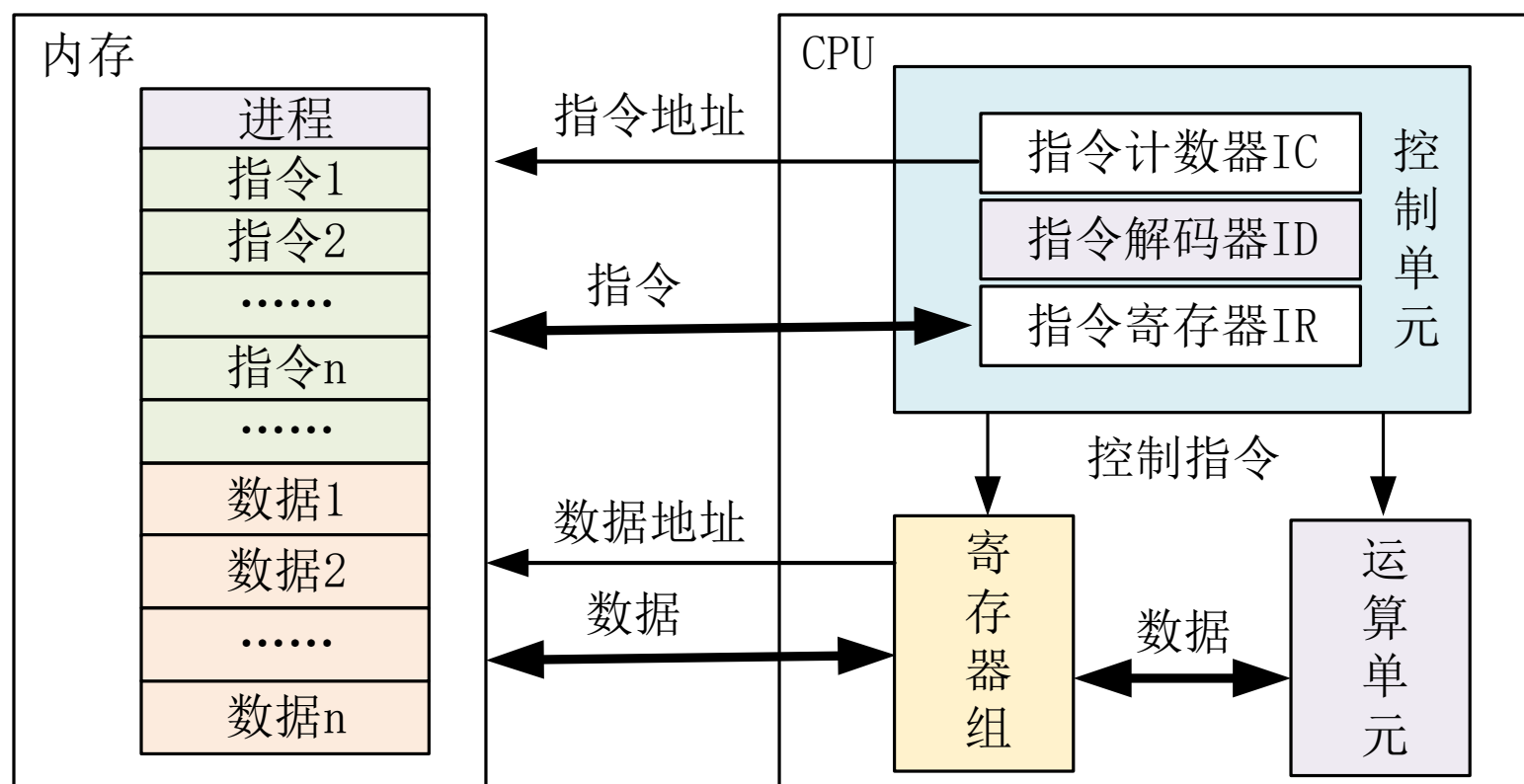
## ● 操作系统依赖的最基本硬件

■ CPU

■ 内存

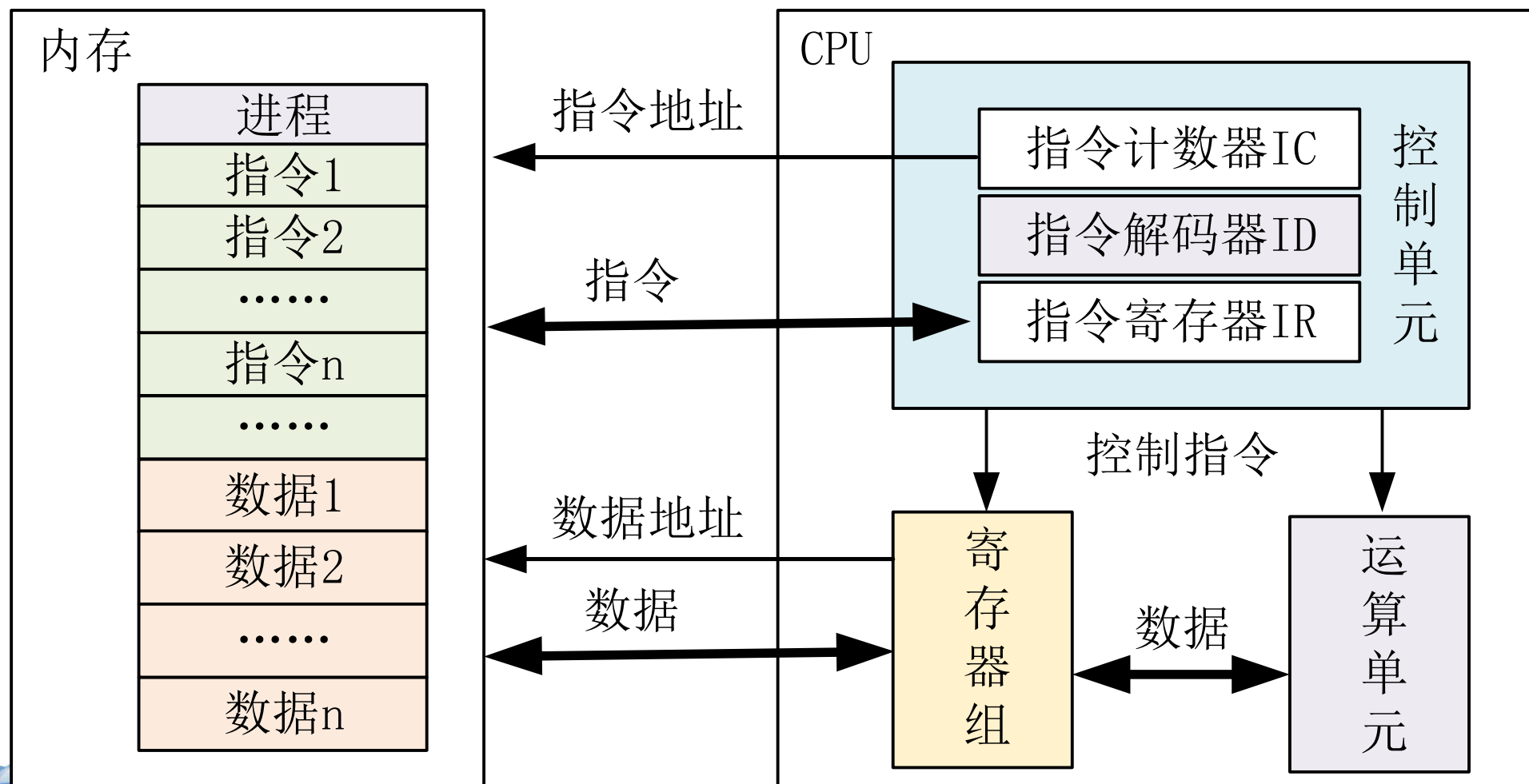
■ 中断

■ 时钟



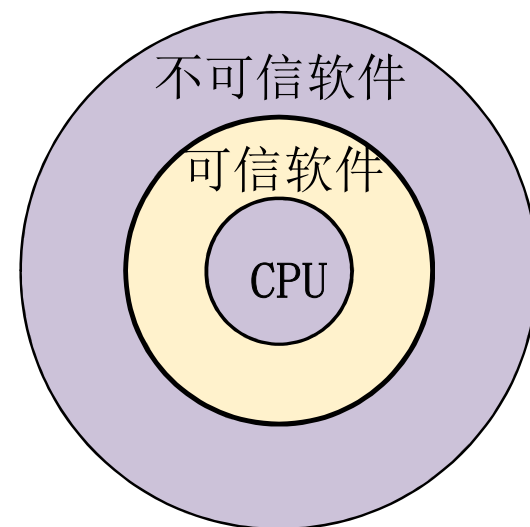
# CPU与内存的交互

- CPU结构：控制单元 | 运算单元 | 寄存器组
- CPU功能：按一定逻辑流程执行指令流：CPU ↔ 内存



# CPU的态（mode）

- CPU态（Mode）
  - CPU的工作模式
  - 对资源和指令使用权限的描述





# 特权指令

```
417  SHL EAX, 4
418  ADD EAX, LABEL_GDT ; EAX <- gdt 基地址
419  MOV DWORD [GdtPtr + 2], EAX ; [GdtPtr + 2] <- gdt 基地址
420
421  ; 为加载 IDTR 作准备
422  XOR EAX, EAX
423  MOV AX, DS
424  SHL EAX, 4
425  ADD EAX, LABEL_IDT ; EAX <- idt 基地址
426  MOV DWORD [IdtPtr + 2], EAX ; [IdtPtr + 2] <- idt 基地址
427  ; 保存 IDTR
428  SIDT [_SavedIDTR]
429  ; 保存中断屏蔽寄存器 (IMREG) 值
430  IN AL, 21h
431  MOV [_SavedIMREG], AL
432
433  ; 加载 GDTR
434  LGDT [GdtPtr]
435  ; 关中断
436  CLI
437  ; 加载 IDTR
438  LIDT [IdtPtr]
```



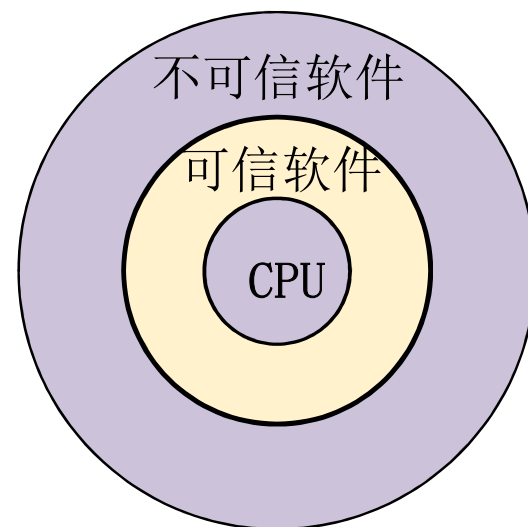
# 特权指令

## ● 类别

- ① 涉及外部设备的输入/输出指令
- ② 修改特殊寄存器的指令
- ③ 改变机器状态的指令

## ● 例子

- LGDT/LIDT/CLTS: 装载特殊寄存器
- STI/CTI: 允许和禁止中断
- HALT: 停止CPU的工作
- IN/OUT: 执行I/O操作



# CPU的态（mode）

- 态的分类

- **核态** (Kernel mode)

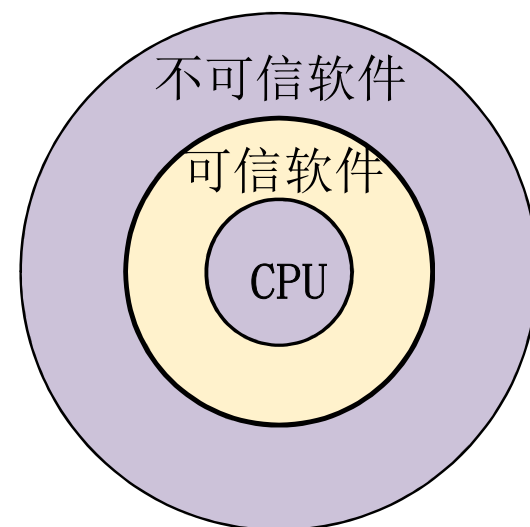
- ◆能够访问所有资源和执行所有指令
    - ◆管理程序/OS内核

- **用户态** (User mode, **目态**)

- ◆仅能访问部分资源，其它资源受限。
    - ◆用户程序

- **管态** (Supervisor mode)

- ◆介于**核态**和**用户态**之间



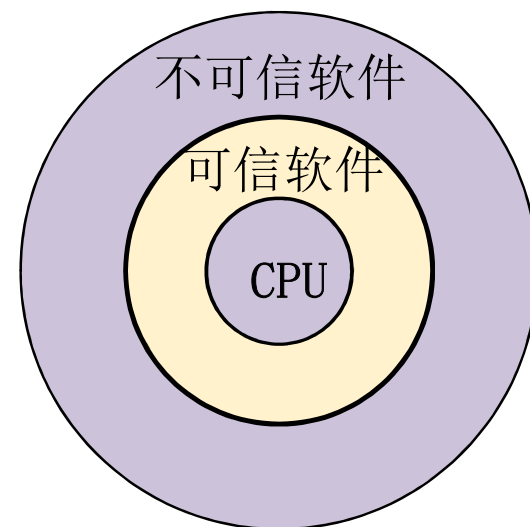
# CPU的态（mode）

## ● 硬件和OS对CPU的观察

- 硬件按“**态**”来区分CPU的状态
- OS按“**进程**”来区分CPU的状态

◆ A,B,C,D: 四个进程

◆ K/U: **K**ernel mode/ **U**ser mode



	进程	OS ↓ A	OS ↓ B	OS ↓ C	OS ↓ D
硬件→	核心态	K			K
硬件→	用户态		U	U	

# Intel CPU的态

- Ring 0 ~ Ring 3 （Ring 0 最核心， Ring 3最外层）

- 段（**Segment**）

- 一段连续的内存

- 段描述符（**Descriptor**）

- 描述段的属性， 8字节

- ◆ 段基址

- ◆ 段界限

- ◆ 段属性

- ☐ 段类型

- ☐ 该段的**特权级（D.P.L）**

- ☐ 是否在内存

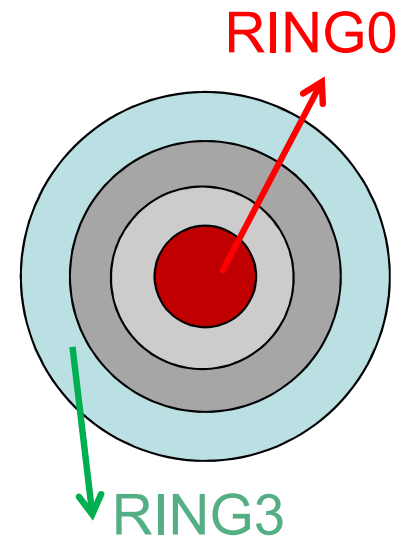
- ☐ ...

数据段

操作数

代码段

指令码



# Intel CPU的态

- 【案例】程序段A访问程序段B时的权限检查（态）

- JMP B 或 CALL B

- 程序A的特权级

- ◆ CPL = 0..3（当前特权级）

- ◆ RPL = 0..3（请求特权级）

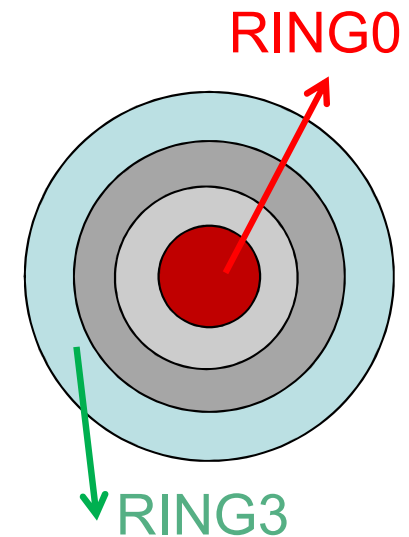
- 程序B的特权级

- ◆ DPL=0..3（描述符特权级）

- 合法的访问

- ◆ 基本原则：同级可访问

```
; Code_A  
MOV  AX, BX ;  
JMP  Code_B ;  
INC  CX ;
```



	特权级（低→高）	特权级（高→低）	相同特权级
一致代码段	Yes	No	Yes
非一致代码段	No	No	Yes
数据段	No	Yes	Yes

# 用户态和核态之间的转换

- 用户态向核态转换

- 用户请求OS提供服务
- 用户进程产生错误（内部中断）
- 用户态企图执行特权指令
- 发生中断

- 核态向用户态转换的情形

- 一般是中断返回：IRET

# 操作系统依赖的最基本硬件

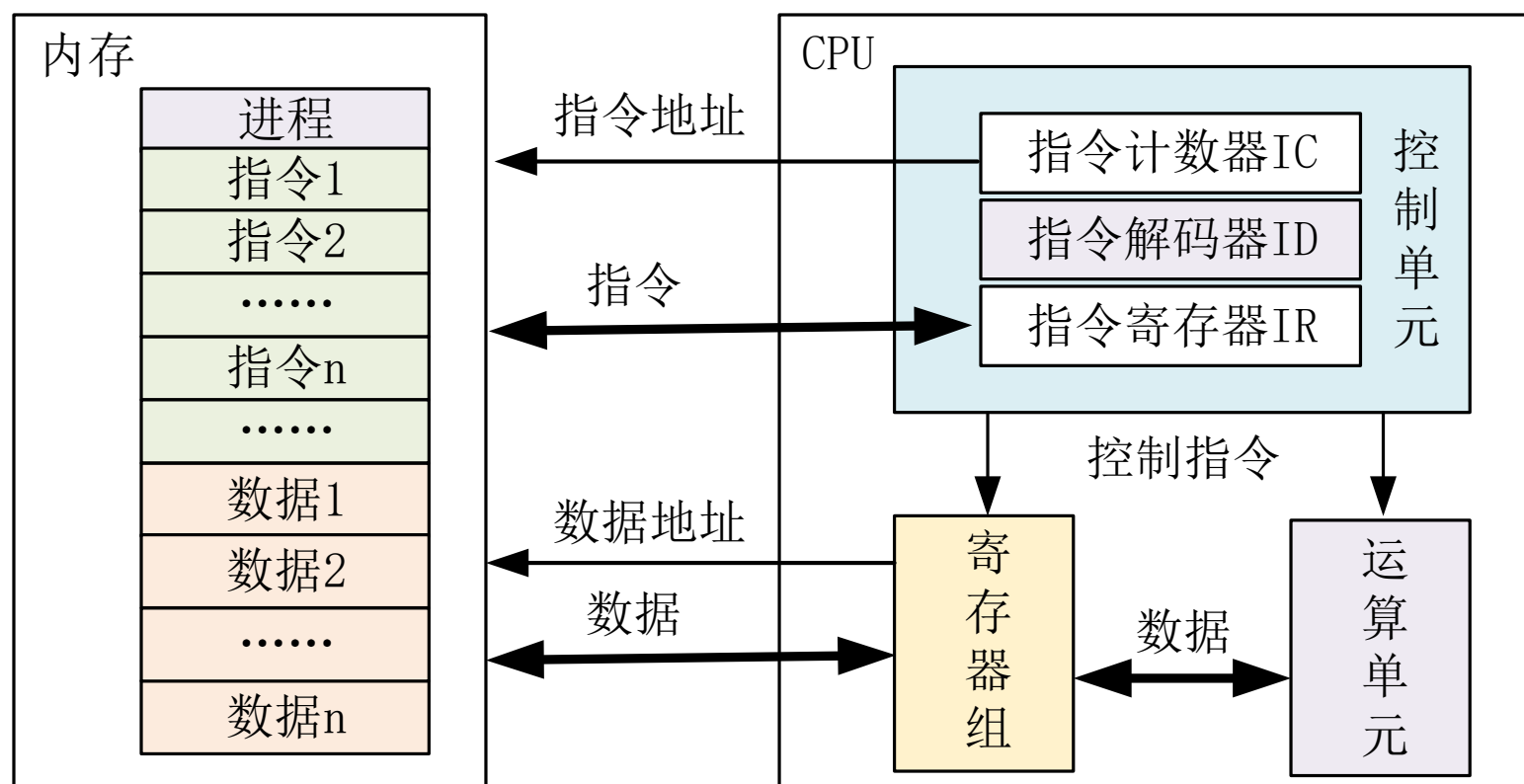
## ● 操作系统依赖的最基本硬件

■ CPU

■ 内存

■ 中断

■ 时钟





# 存储体系

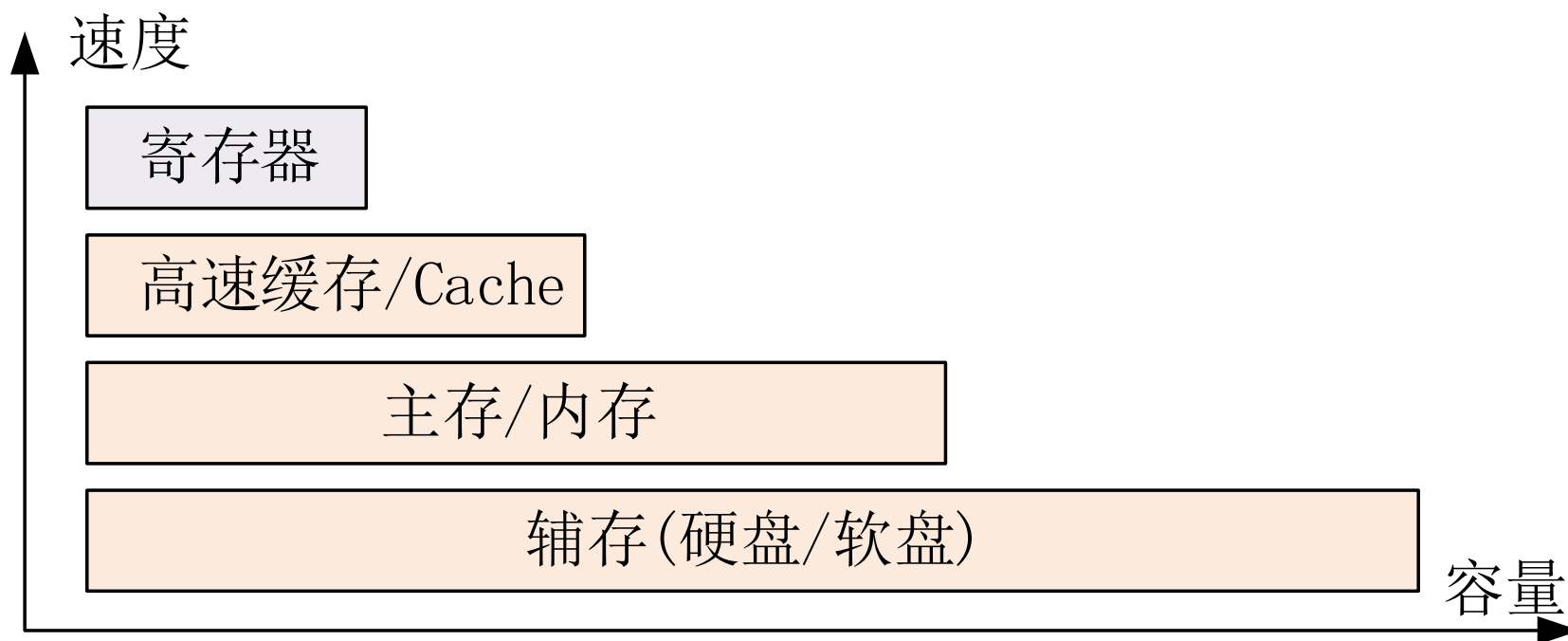
- 理想存储体系：速度快，容量大，成本低
- 实际存储体系

- 寄存器

- 高速缓存（**CACHE**）

- 主存

- 磁盘



# 操作系统依赖的最基本硬件

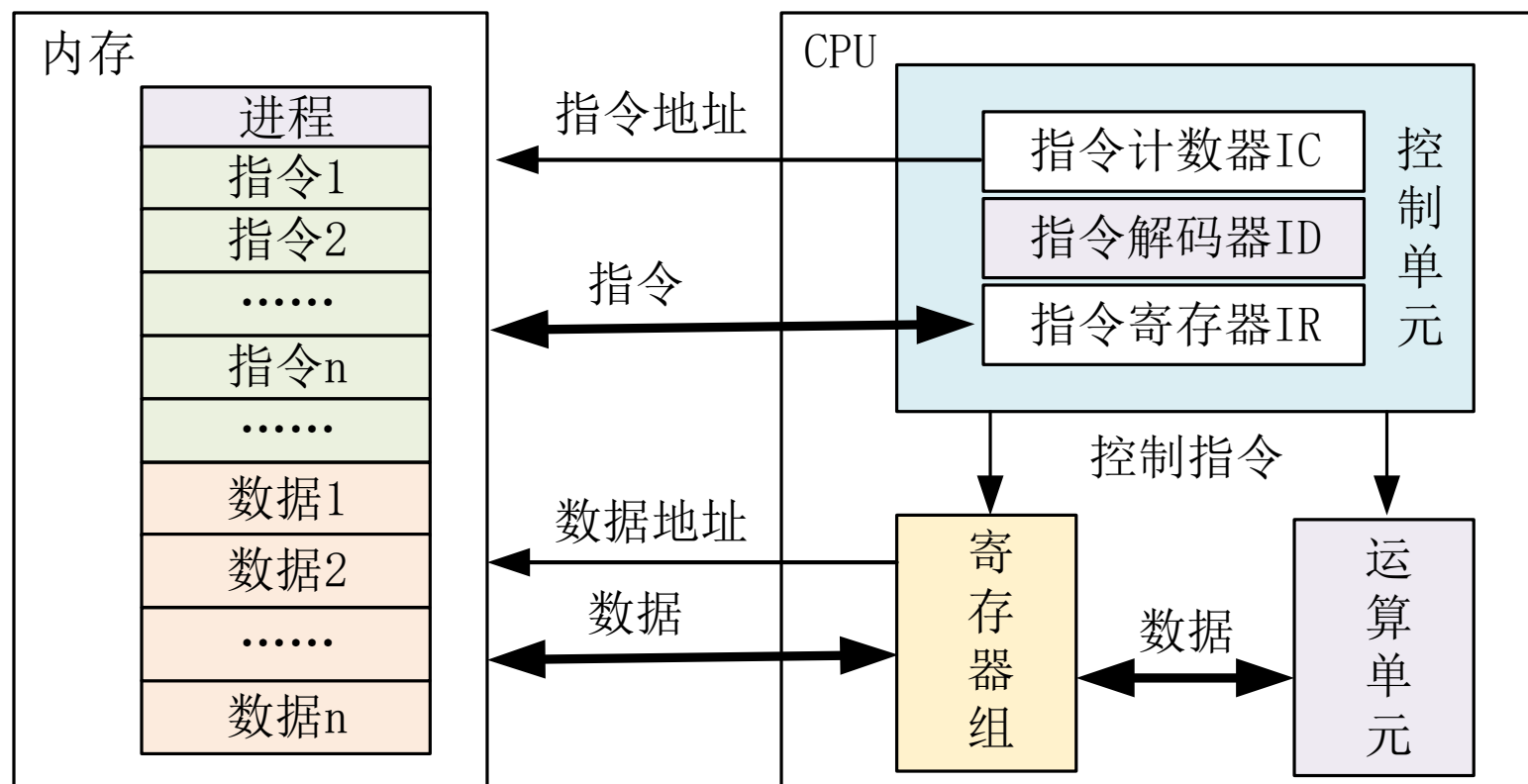
## ● 操作系统依赖的最基本硬件

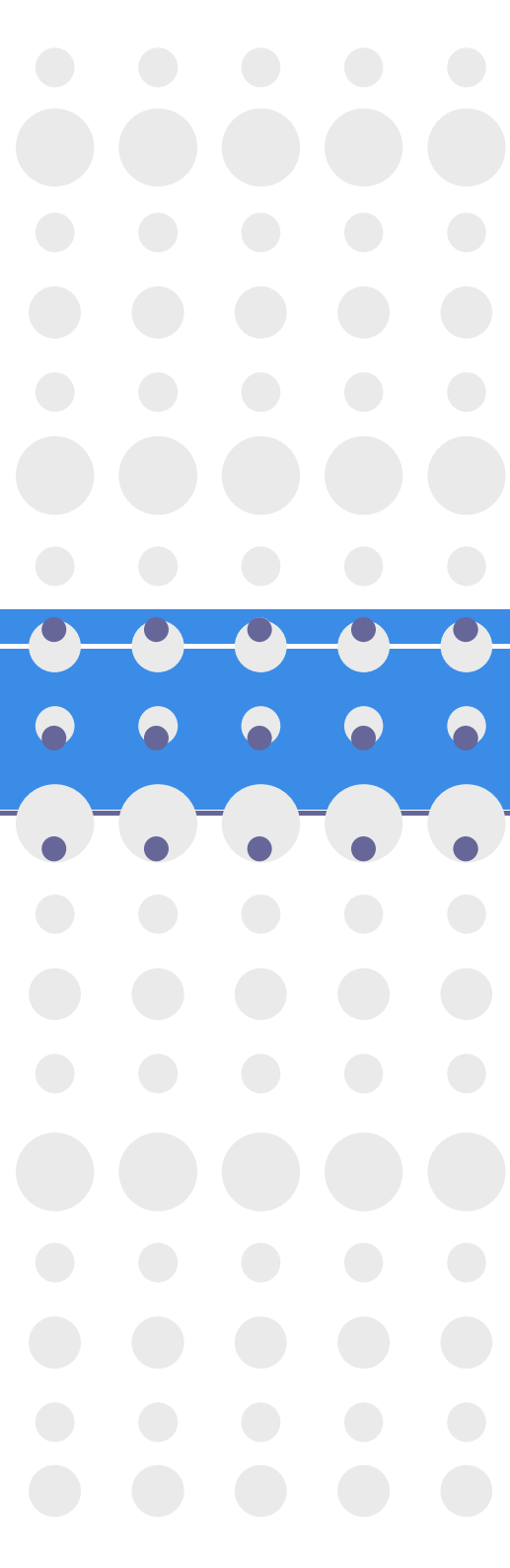
■ CPU

■ 内存

■ 中断

■ 时钟



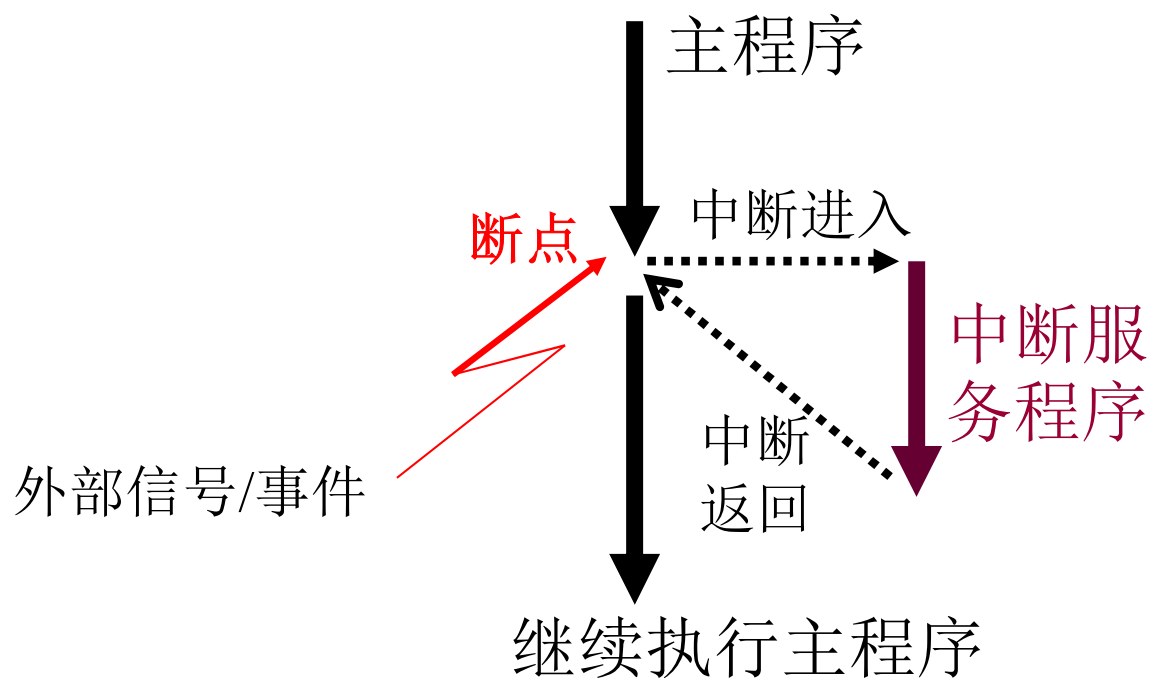


## 2.4 中断机制

# 中断机制

## ● 中断定义

- 指CPU对突发的外部事件的反应过程或机制。
- CPU收到**外部信号**（中断信号）后，停止当前工作，转去处理该**外部事件**，处理完毕后回到原来工作的**中断处**（断点）继续原来的工作。



# 中断的一些概念

- 中断源

- 引起系统中中断的事件称为**中断源**

- 中断类型

- **强迫中断**和**自愿中断**

- ◆ 强迫中断：程序没有预期：例：外部中断

- ◆ 自愿中断：程序有预期。例：访管指令/INT 21h

- **外中断**（中断）和**内中断**（俘获）

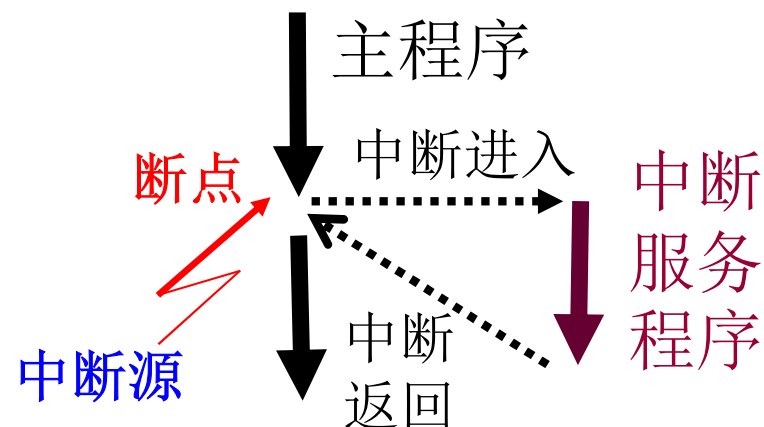
- ◆ 外中断：由**CPU**外部事件引起。例：外部中断(I/O)

- ◆ 内中断：由**CPU**内部事件引起。例：访管指令、程序中断

- 外中断：**不可屏蔽中断**和**可屏蔽中断**

- ◆ 不可屏蔽中断：中断原因很紧要，**CPU**必须响应

- ◆ 可屏蔽中断：中断原因不很紧要，**CPU**可以不响应



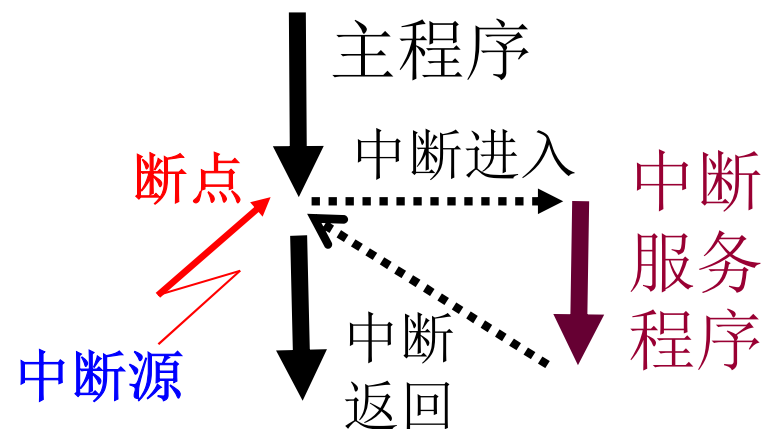
# 中断的一些概念

## ● 断点

### ■ 程序中中断的地方

◆ 将要执行的下一条指令的地址

■ CS:IP (FLAGS、SS、SP)



## ● 现场

■ 程序正确运行所依赖的信息集合。

◆ PSW (程序状态字)、相关寄存器、断点

## ● 现场的处理

■ 现场保护：进入中断服务程序之前：CPU→栈

■ 现场恢复：退出中断服务程序之后：栈→CPU

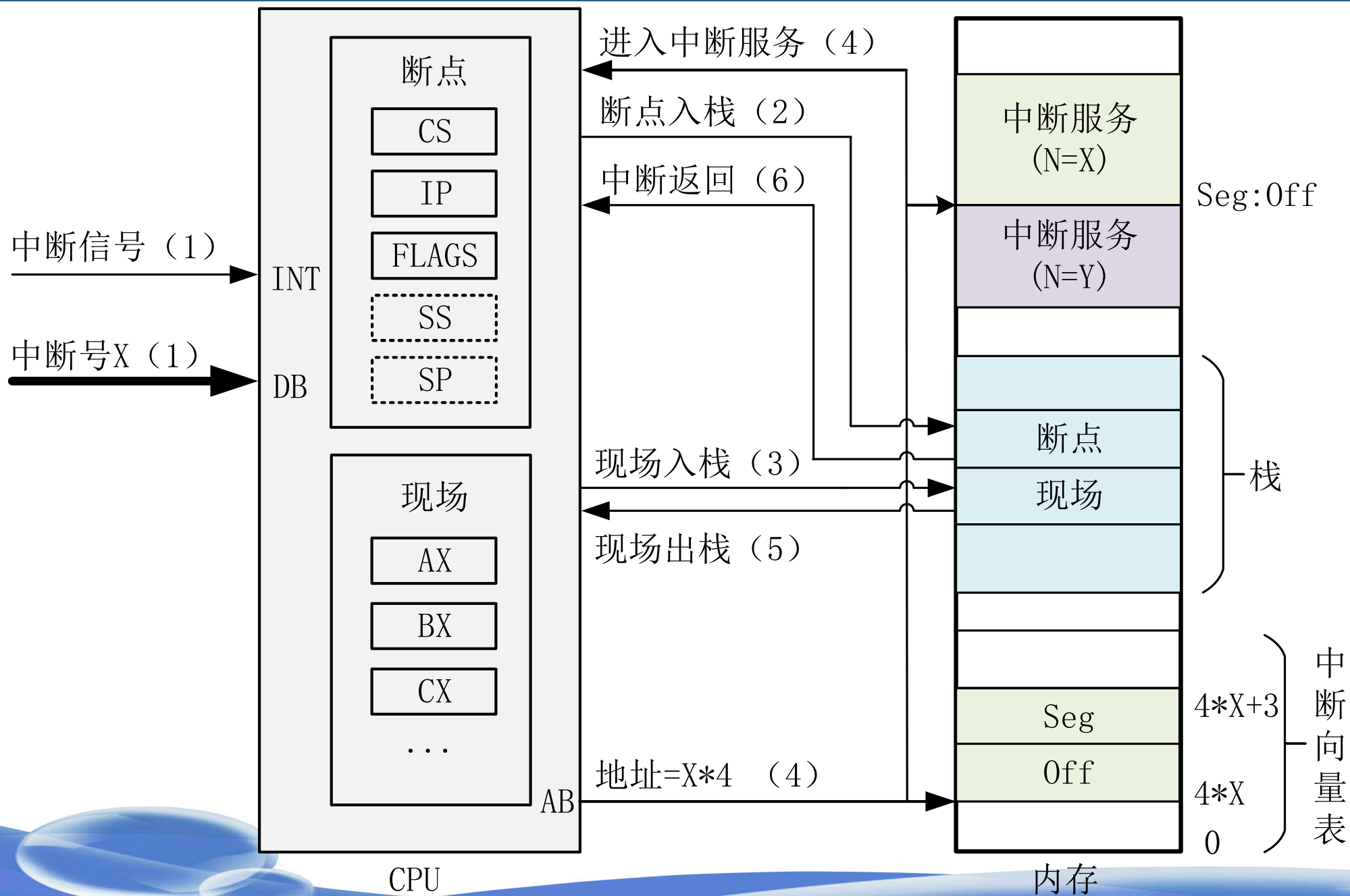
# 中断响应过程

## ● 中断响应过程

- (1) 识别中断源
- (2) 保护断点
- (3) 保护现场
- (4) 进入中断服务程序
- (5) 恢复现场
- (6) 中断返回



# 中断响应过程



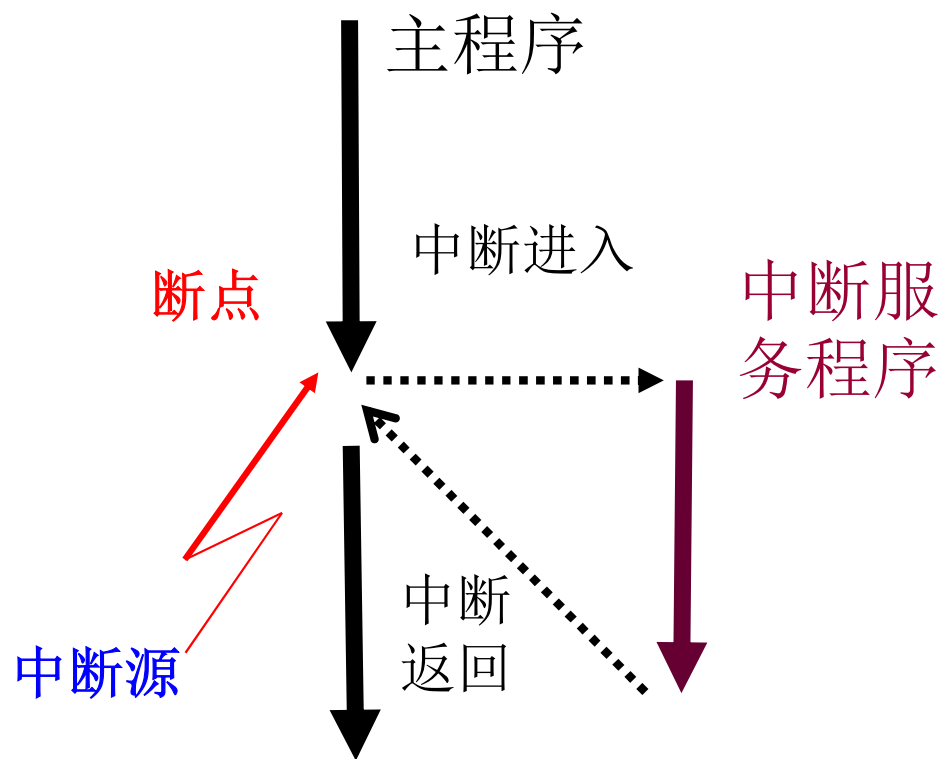
# 中断机制

- 中断响应的实质

- 交换指令执行地址
- 交换CPU的态
- 工作
  - 现场保护和恢复
  - 参数传递（通信）

- 引入中断的目的

- 实现并发活动
- 实现实时处理
- 故障自动处理





## 2.5 基本输入输出系统/BIOS

# 系统BIOS的作用

- 系统BIOS

- 固件 (firmware)

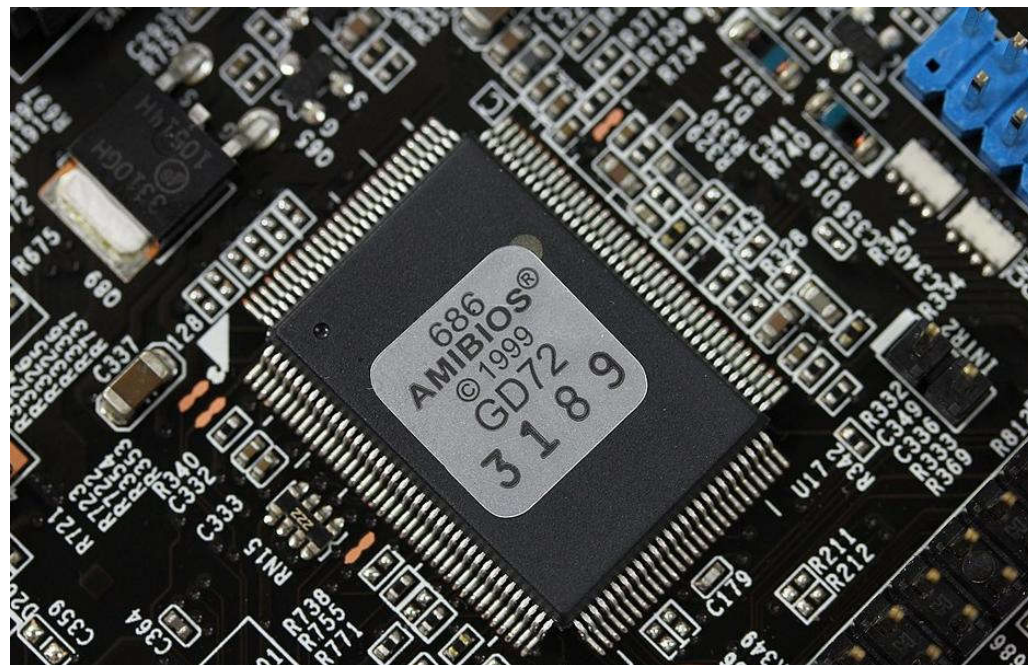
- 地址范围: F0000- FFFFF

- 系统BIOS作用

- (1) 加电自检(POST)

- ◆ 加电或复位开始执行

- .....



# 加电自检POST

## ● 加电自检(Power On Self-Test)

### ■ 初始化基本硬件

◆ CPU

◆ 内存

◆ 显卡

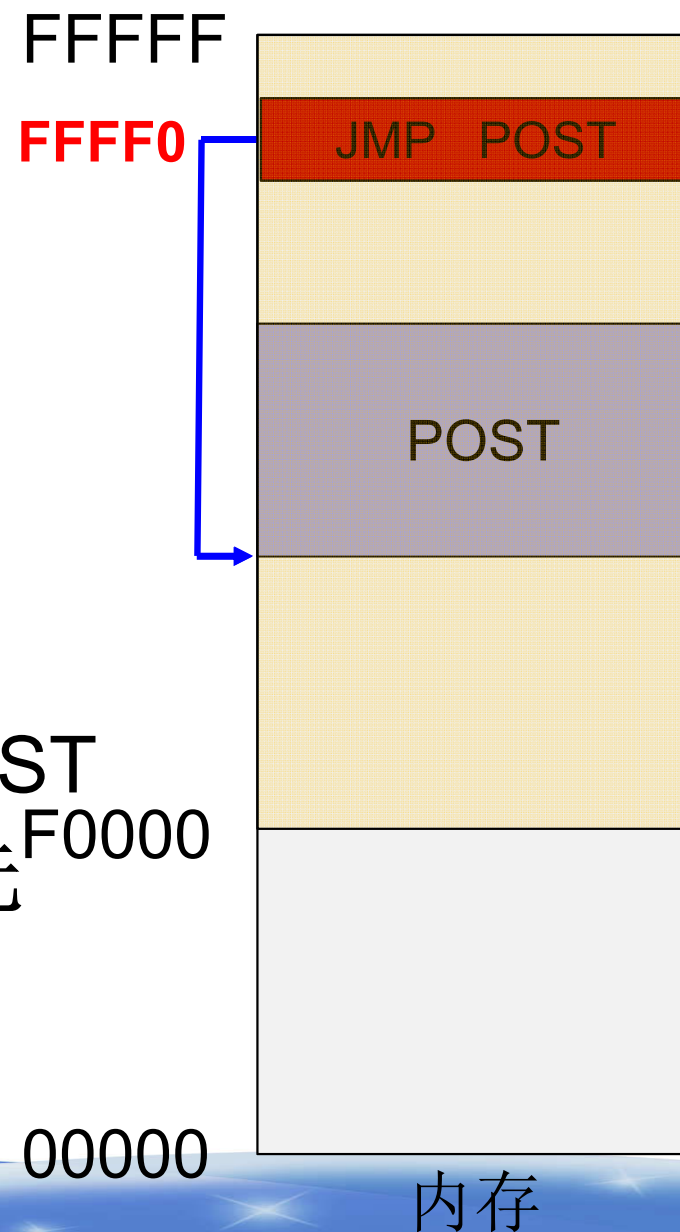
◆ ...

### ■ 自检错误通过喇叭鸣叫提示

### ■ 按下PowerOn或Reset开始执行POST

### ■ 计算机首条指令在 ( FFFF0 ) 单元

**JMP POST ; 加电自检**





# 系统BIOS的作用

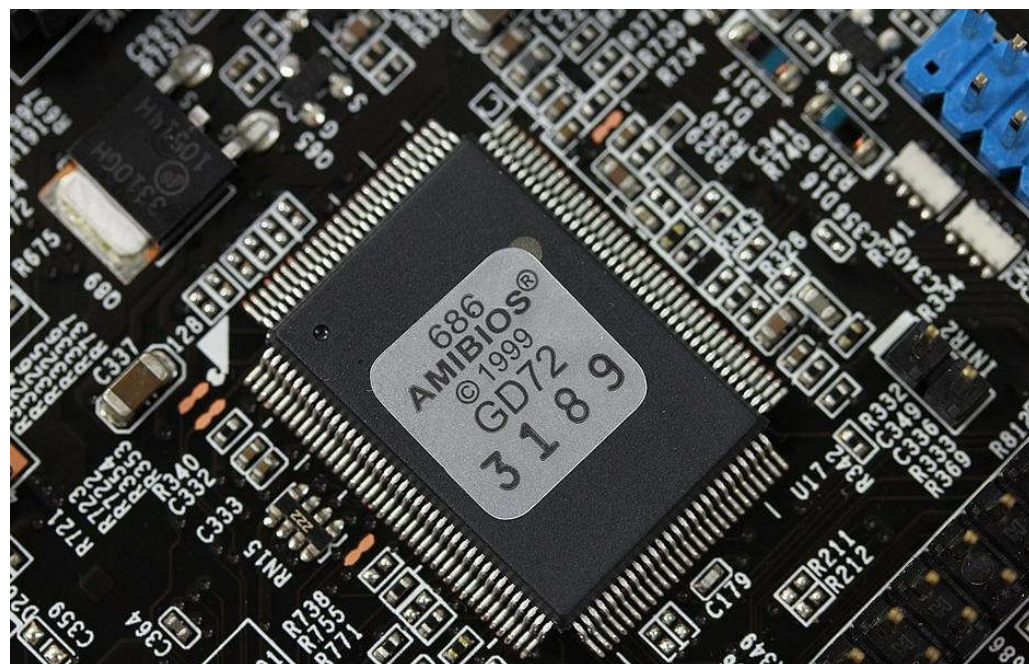
- 系统BIOS

- 固件（firmware）
- 地址：F0000- FFFFF

- 系统BIOS作用

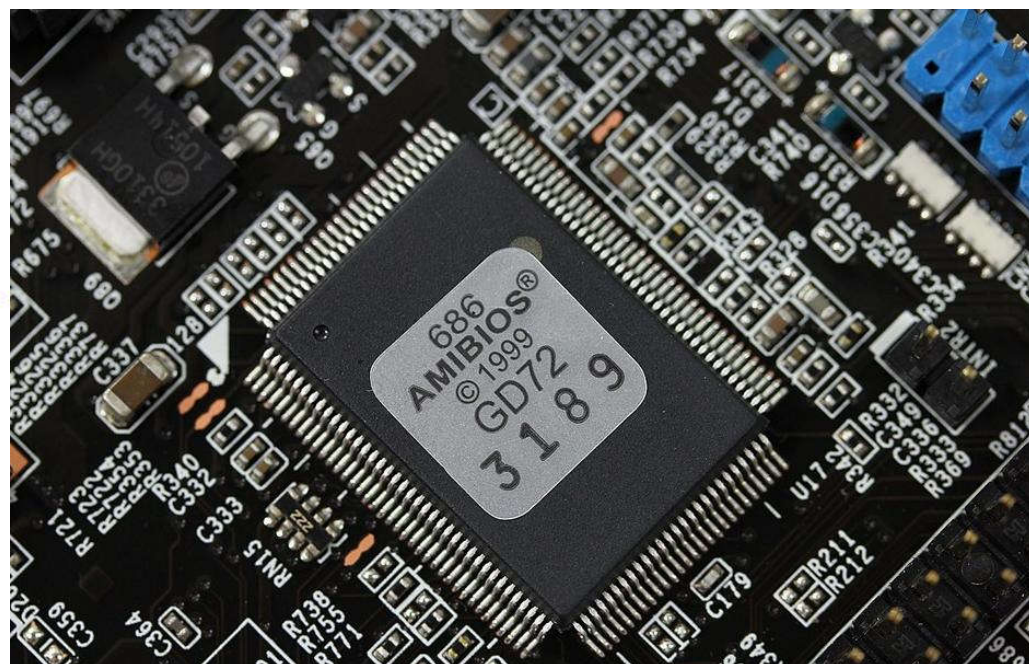
- （1）加电自检(POST)
  - ◆ 加电或复位开始执行
- （2）CMOS设置
  - ◆ 交互式设置系统参数

- .....



# 系统BIOS的作用

```
1  org 0x7c00
2  ; 清屏
3  mov ax, 0600h
4  mov bx, 0700h
5  mov cx, 0
6  mov dx, 184fh
7  int 10h
8  ; 打印字符串: Start Boot ...
9  mov ax, 1301h
10 mov bx, 000fh
11 mov dx, 0000h
12 mov cx, 10
13 push ax
14 mov ax, ds
15 mov es, ax
16 pop ax
17 mov bp, BootMessage
18 int 10h
19 ; 原地停留
20 jmp $
21 BootMessage: db "Start Boot ..."
```





# 系统BIOS的作用

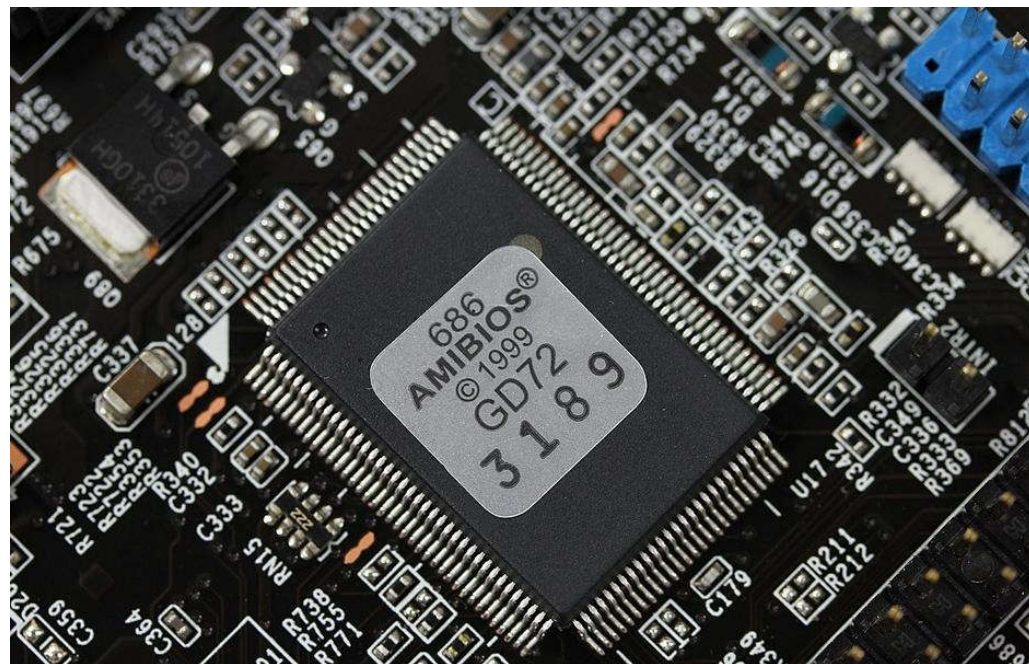
- 系统BIOS

- 固件 (firmware)
- 地址: F0000- FFFFF

- 系统BIOS作用

- (1) 加电自检(POST)
  - ◆ 加电或复位开始执行
- (2) CMOS设置
  - ◆ 交互式设置系统参数
- (3) 基本I/O服务
  - ◆ BIOS中断

- .....



```
3      mov ah, 0
4      int 16h
5      cmp al, 1ch ;al:ASCII码
```

# 基本I/O服务（BIOS中断）

## ● BIOS使用的中断类型号为10H ~ 1FH 例：INT 13H

中断号	功能	中断号	功能
10H	显示器 I/O 调用	18H	ROM BASIC 入口
11H	获取设备配置调用	19H	自举程序入口
12H	获取存储器大小调用	1AH	时间日期调用
13H	软盘 I/O 调用	1BH	Ctrl-Break 控制
14H	异步通信口调用	1CH	定时处理
15H	磁带 I/O 调用	1DH	显示器参数表
16H	键盘 I/O 调用	1EH	软盘参数表
17H	打印机 I/O 调用	1FH	字符点阵结构参数表

## 例：INT 13H 软盘I/O调用的子功能（例：02H读扇区）

子功能号	子功能	子功能号	子功能
00H	磁盘系统复位	0AH	读长扇区
01H	读取磁盘系统状态	0BH	写长扇区
02H	读扇区	0CH	查寻
03H	写扇区	0DH	硬盘系统复位
04H	检验扇区	0EH	读扇区缓冲区
05H	格式化磁道	0FH	写扇区缓冲区
06H	格式化坏磁道	10H	读取驱动器状态
07H	格式化驱动器	11H	校准驱动器
08H	读取驱动器参数	12H	控制器 RAM 诊断
09H	初始化硬盘参数	13H	控制器驱动诊断



## 例：INT 13H 软盘I/O调用的子功能（例：02H读扇区）

### ● 读第21号扇区到内存10000h处

■ 注：假定每个磁道有18个扇区

1	MOV	AX,	1000H	
2	MOV	ES,	AX	
3	MOV	BX,	0	; 内存目标地址
4	MOV	AL,	1	; 扇区数
5	MOV	CH,	0	; 柱面号 (C)
6	MOV	DH,	1	; 磁头号 (H)
7	MOV	CL,	3	; 扇区号 (S)
8	MOV	DL,	0	; 驱动器号
9	MOV	AH,	02H	; 子功能号
10	INT	13H		; BIOS中断号

# POST之后.....

- 依次查找其它设备并初始化

- 查找显卡，执行显卡BIOS(C0000-C7FFF)
- 查找IDE控制器，执行IDE BIOS (C8000-CBFFF)

- 显示启动信息

- ◆ 各组件版权信息

- 查找/设置其他外设

- ◆ 配置寄存器

- ◆ 启动设备BIOS



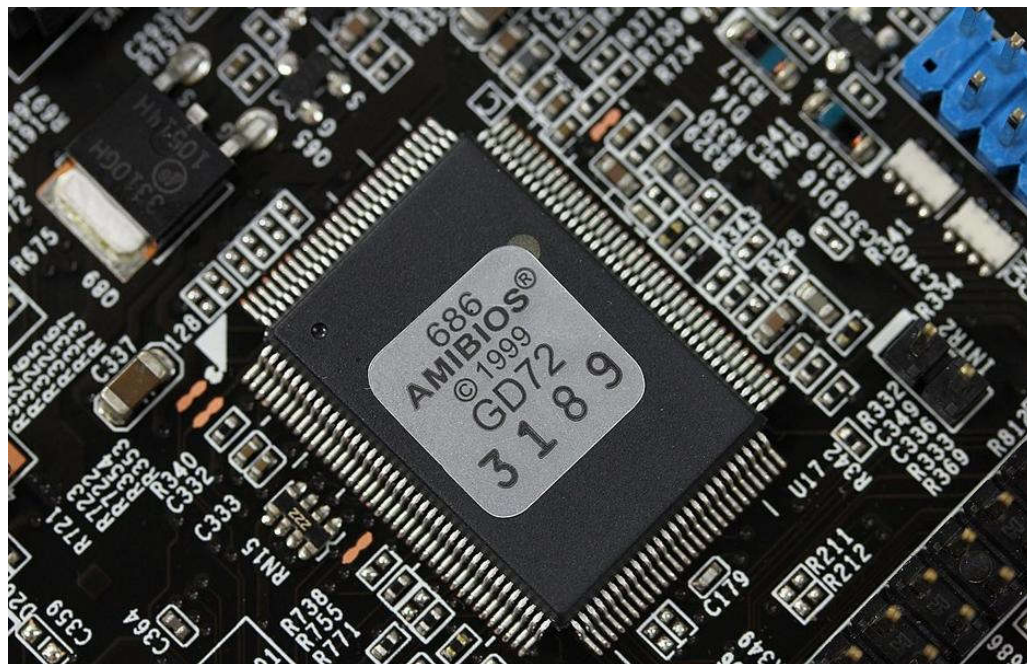
# 系统BIOS的作用

- 系统BIOS

- 固件（firmware）
- 地址：F0000- FFFFF

- 系统BIOS作用

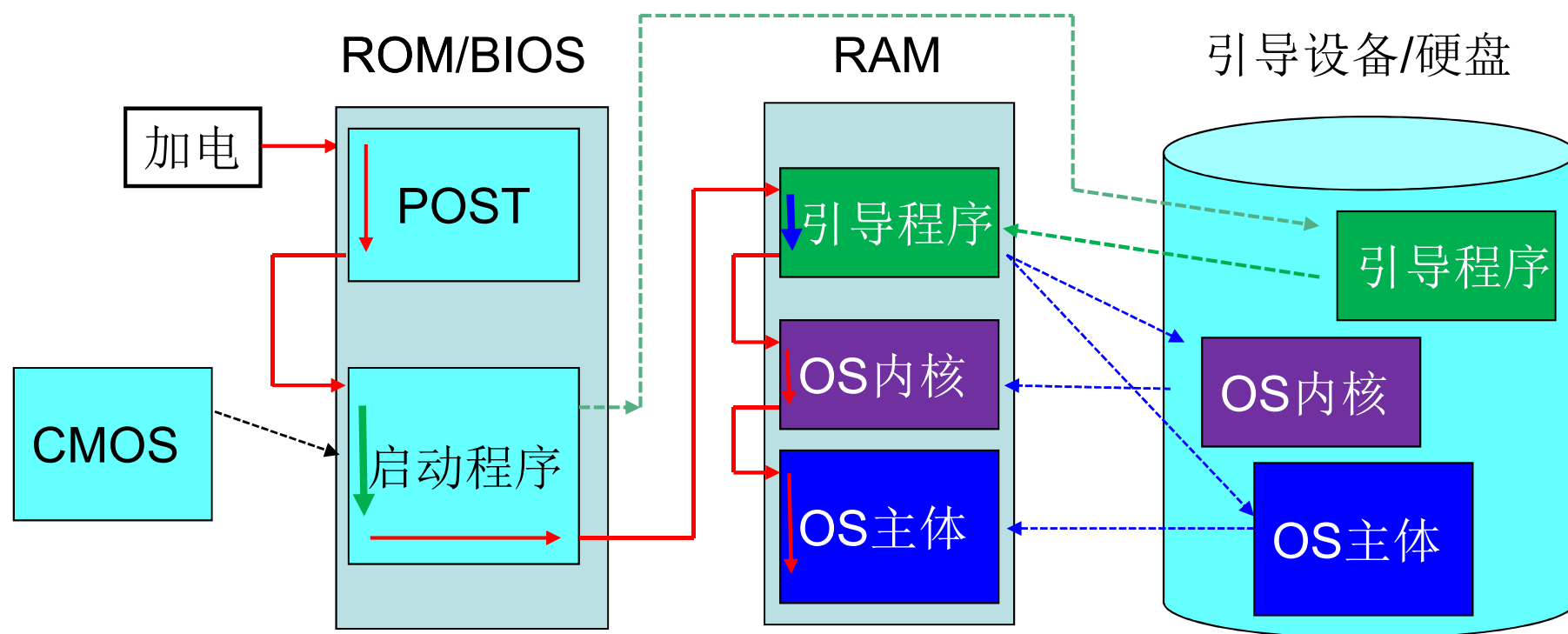
- （1）加电自检(POST)
  - ◆ 加电或复位开始执行
- （2）CMOS设置
  - ◆ 交互式设置系统参数
- （3）基本I/O服务
  - ◆ BIOS中断
- （4）系统自举/加载OS



# 系统自举/加载OS的两种方式

- 现场引导方式

- OS文件存储在本地存储设备



- 下载引导方式

- OS文件存储在网络存储设备中





## 2.8 操作系统的启动

# 实模式和保护模式

- 实模式（实地址模式，**REAL MODE**）
  - 按照8086方法访问0~FFFFFFh 空间
  - 寻址方式：物理地址(20位)=段地址:偏移地址。
  - CPU单任务运行
- 实模式的1M空间
  - 前面640K 【00000 -- 9FFFF】：基本内存
  - 中间128K 【A0000 -- BFFFF】：显卡显存
  - 末尾256K 【C0000 -- FFFFF】：**BIOS**
- 保护模式（内存保护模式，**Protect Mode**）
  - 寻址方式：段(16位)和偏移量(32位)，寻址4GB
  - CPU支持多任务

# 操作系统的启动

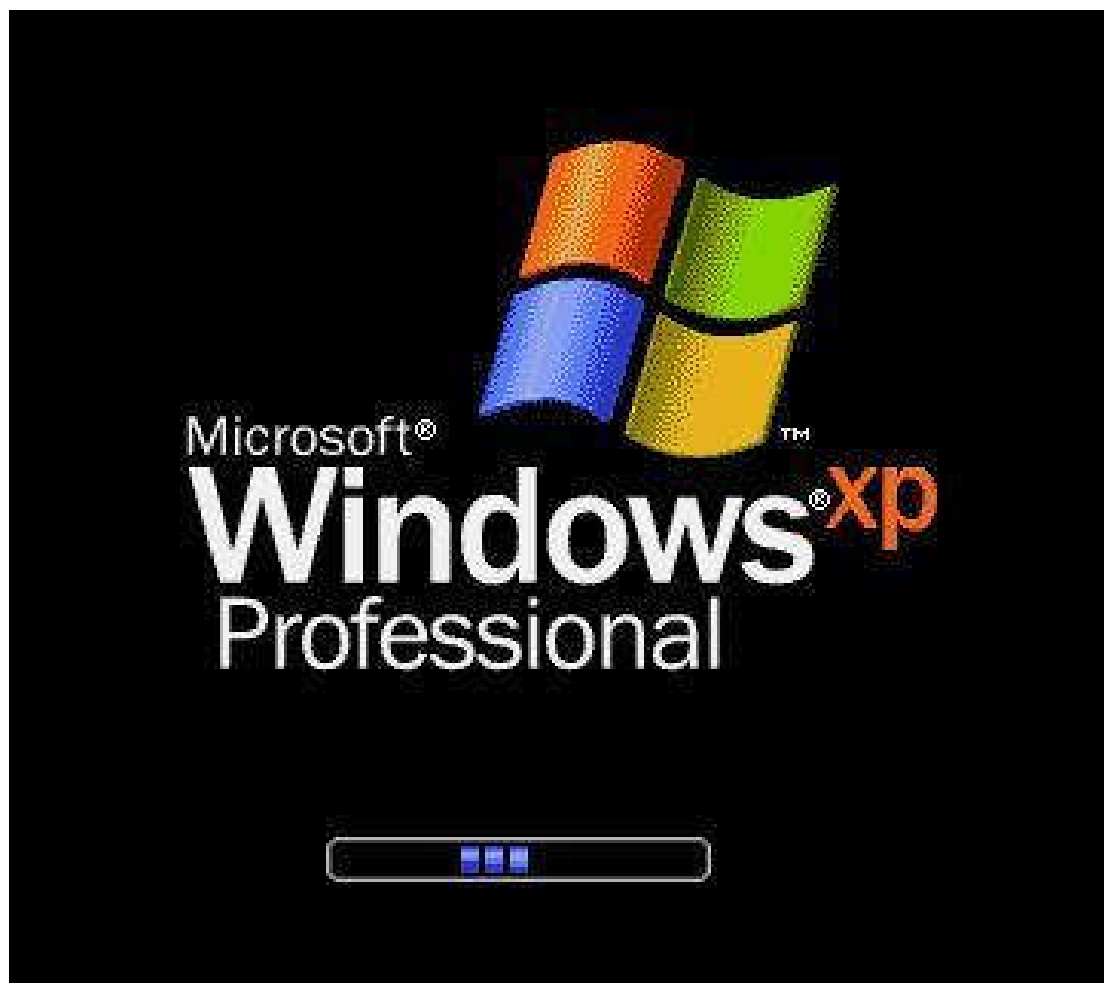
- 启动过程

- 从加电到用户工作环境准备好的过程

- ◆ (1) 初始引导

- ◆ (2) 核心初始化

- ◆ (3) 系统初始化



# 1)初始引导

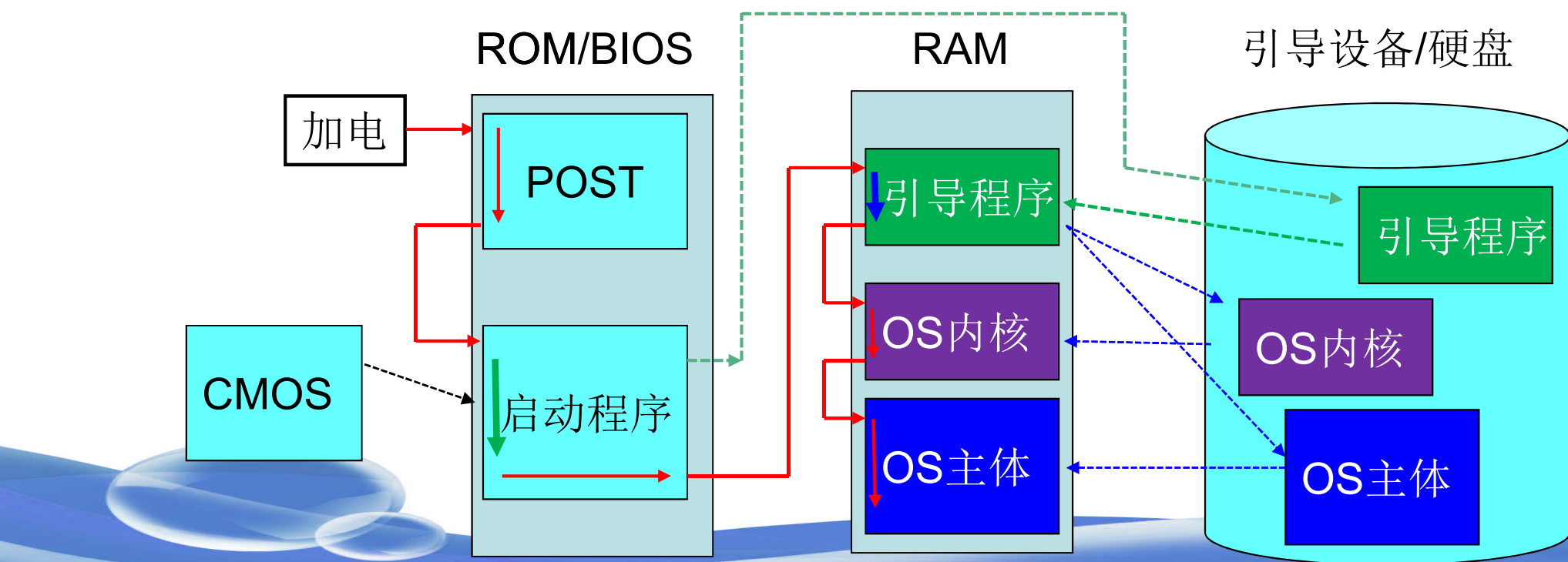
## ● 目的

■ 把OS内核装入内存并使之开始工作接管计算机系统

■ 引导程序：MBR（Master BOOT Record）

◆ GRUB

◆ bootmgr | NTLDR | LILO



# 1)初始引导

- 引导程序缺失



## 2)核心初始化

- 核心初始化

- 目的：OS内核初始化系统的核心数据

- 典型工作

- ◆ 各种寄存器的初始化

- ◆ 存储系统和页表初始化

- ◆ 核心进程构建

- ◆ .....



### 3)系统初始化

- 系统初始化

- 为用户使用系统作准备，使系统处于待命状态。

- 主要工作

- ◆ 初始化文件系统

- ◆ 初始化网络系统

- ◆ 初始化控制台

- ◆ 初始化图形界面

- ◆ .....

# DOS的启动过程

## ● DOS的启动过程

### ■ POST

◆ 加电后BIOS启动主机自检程序

### ■ 初始引导

◆ BIOS从MBR读入**引导程序**，装入内存的特定位置

◆ **引导程序**运行将io.sys及msdos.sys读入内存

◆ DOS运行起来取代BIOS接管整个系统。

### ■ 核心初始化

◆ 操作系统读入config.sys配置系统核心

### ■ 系统初始化

◆ 读入Command.com，执行autoexec.bat，系统待命

# Windows的启动过程

## ● Windows的启动过程

### ■ POST

- ◆ 加电后BIOS启动主机自检程序

### ■ 初始引导

- ◆ BIOS从MBR读入**引导程序**，装入内存的特定位置

- ◆ **引导程序**启动DOS7.0，调入操作系统核心

- ◆ WINDOWS开始接管系统

### ■ 核心初始化

- ◆ 资源状态、核心数据等初始化；

### ■ 系统初始化

- ◆ GUI界面生成，系统处于待命/消息接受状态

# Linux的启动过程

## ● LINUX的启动过程

■ POST →

■ MBR →

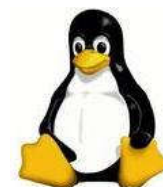
■ KERNEL映像 →

■ KERNEL映像边自解压并边执行→

■ 内核初始化→

■ 内核启动 →

■ init进程 →



## 案例3：利用Grub手工启动OS（课后要实践）

- 硬盘分区情况

- hda1: Windows7(hd0,0)

- hda2: Federa 23 Linux(hd0,1)

- hdb1: CentOS Linux(hd1,0)

- 用**Grub**命令启动**Federa**

- grub > root (hd0,1)

- grub > kernel /boot/vmlinuz root=/dev/hda2 ro

- grub > boot



## 案例3：利用Grub手工启动OS（课后要实践）

- 用**Grub**命令启动**Federa**

- grub > root (hd0,1)

- grub > kernel /boot/vmlinuz root=/dev/hda2 ro

- grub > boot

- **root** 命令等同**mount**，挂接指定分区

- **kernel**命令加载指定的**Linux**核心文件

- root参数指明根目录位置

- ro参数：只读

- **boot**命令：启动





## 案例3：利用Grub手工启动OS（课后要实践）

- 用**Grub**命令启动**CentOS**

- 硬盘分区情况

- hda1: Windows7(hd0,0)

- hda2: Federa 23 Linux(hd0,1)

- hdb1: CentOS Linux(hd1,0)

- 用**Grub**命令启动**Fedara**

- grub> root (hd0,1)

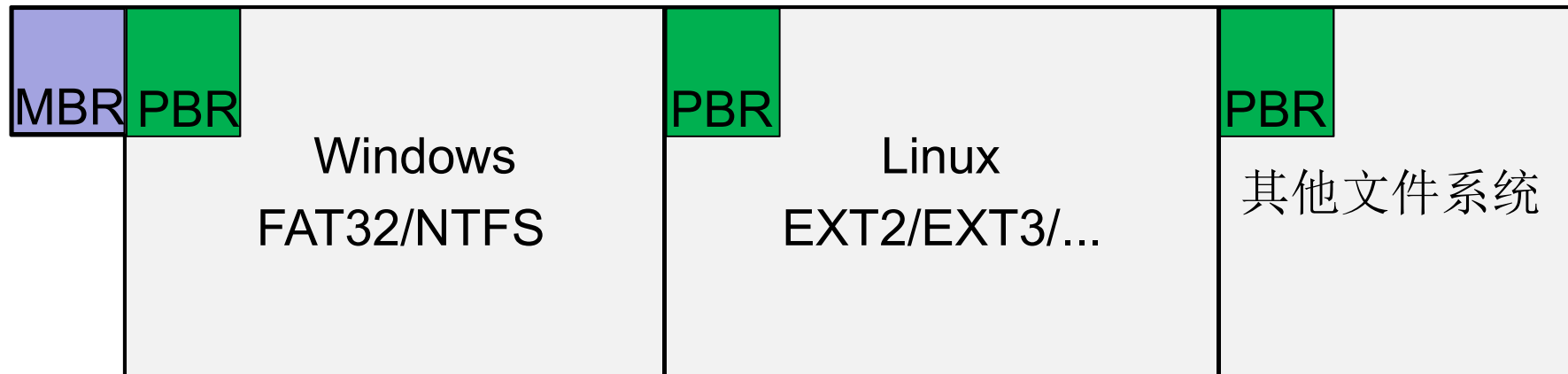
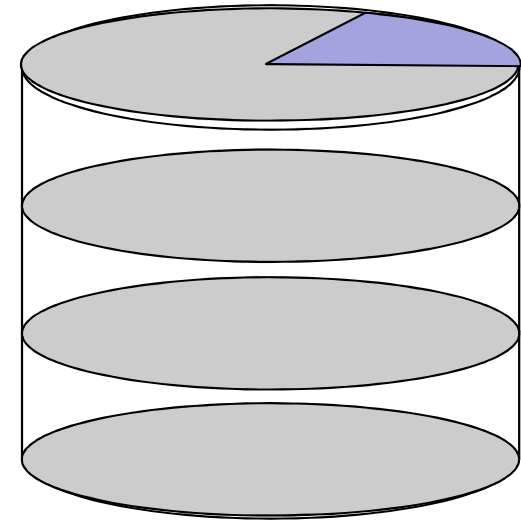
- grub > kernel /boot/vmlinuz root=/dev/hda2 ro

- grub > boot

# MBR: Main Boot Record

## ● 特点

- 主启动扇区（Main Boot Sector）
- 存放引导代码（ BootLoader ）
  - ◆ 启动相关的数据和代码
- 510字节 + AA55h

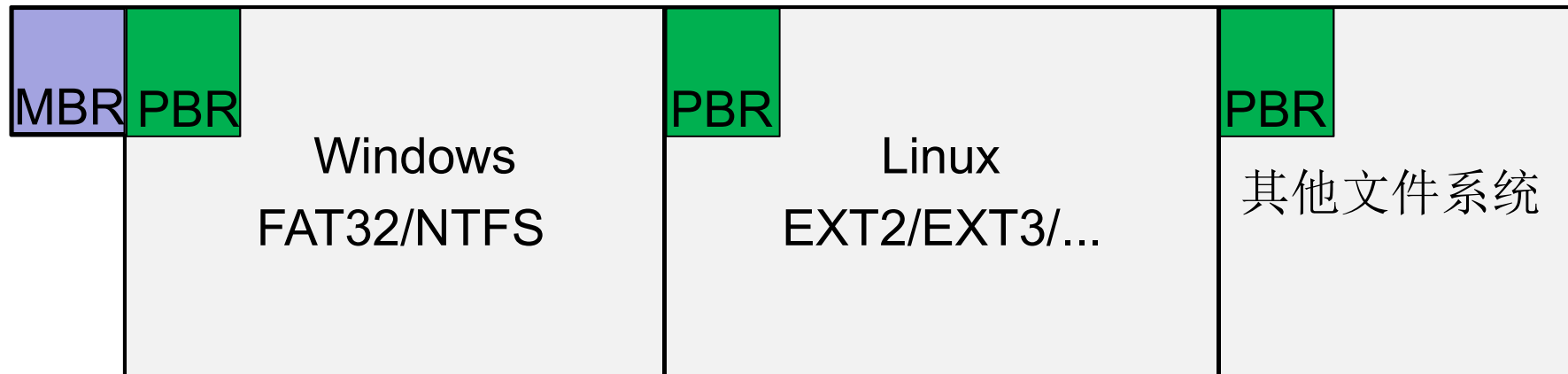


## ● PBR: 分区/次引导记录

- Partition Boot Record

# MBR: Main Boot Record

- 提供**BootLoader**或启动管理
  - 直接指向引导代码加载OS
  - 提供启动选项菜单
  - 跳转到**PBR**以加载特定OS BootLoader



# 操作系统的安装过程

## ● 安装过程

- 把OS映像拷贝到存储空间；

  - ◆ 拷贝/安装位置：硬盘

- 写启动相关代码和数据

  - ◆ MBR扇区

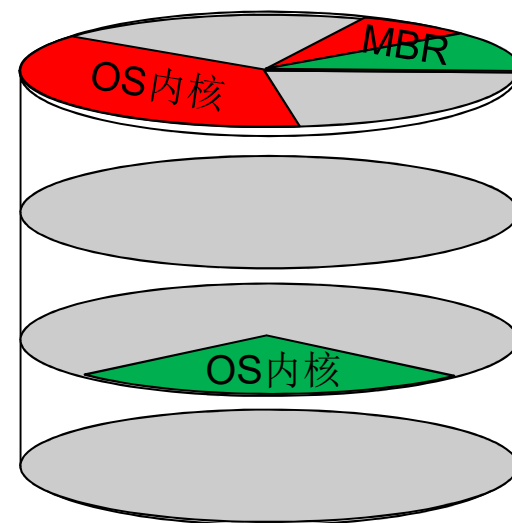
  - ◆ PBR扇区

  - ◆ 512字节

## ● 多操作系统安装

- MBR重写 vs. MBR追加

- 安装顺序



## 案例4: MBR程序 (最简单的例子)

### ● 开机后在屏幕上显示"Hello MBR!"字符串

```
1      ORG 07C00h                ;程序加载到 07c00h处
2      MOV AX,CS
3      MOV DS,AX
4      MOV ES,AX
5      CALL DispString           ;调用显示字符串的例程
6      JMP $                     ;停在此处
7  DispString:
8      MOV AX,MessageMBR
9      MOV BP,AX                 ;ES:BP:字符串的地址
10     MOV CX,10                 ;CX:字符串的长度
11     MOV AX,01301h
12     MOV BX,000Ch
13     MOV DL,0
14     INT 10h                   ;10h号中断显示字符串
15     RET
16     MessageMBR DB "Hello MBR!"
17     TIMES 510-($-$$) DB 0     ;填充若干0, 凑够510字节
18     DW 0AA55h                 ;结束标志:55AA
```



## 2. 10 操作系统的生成

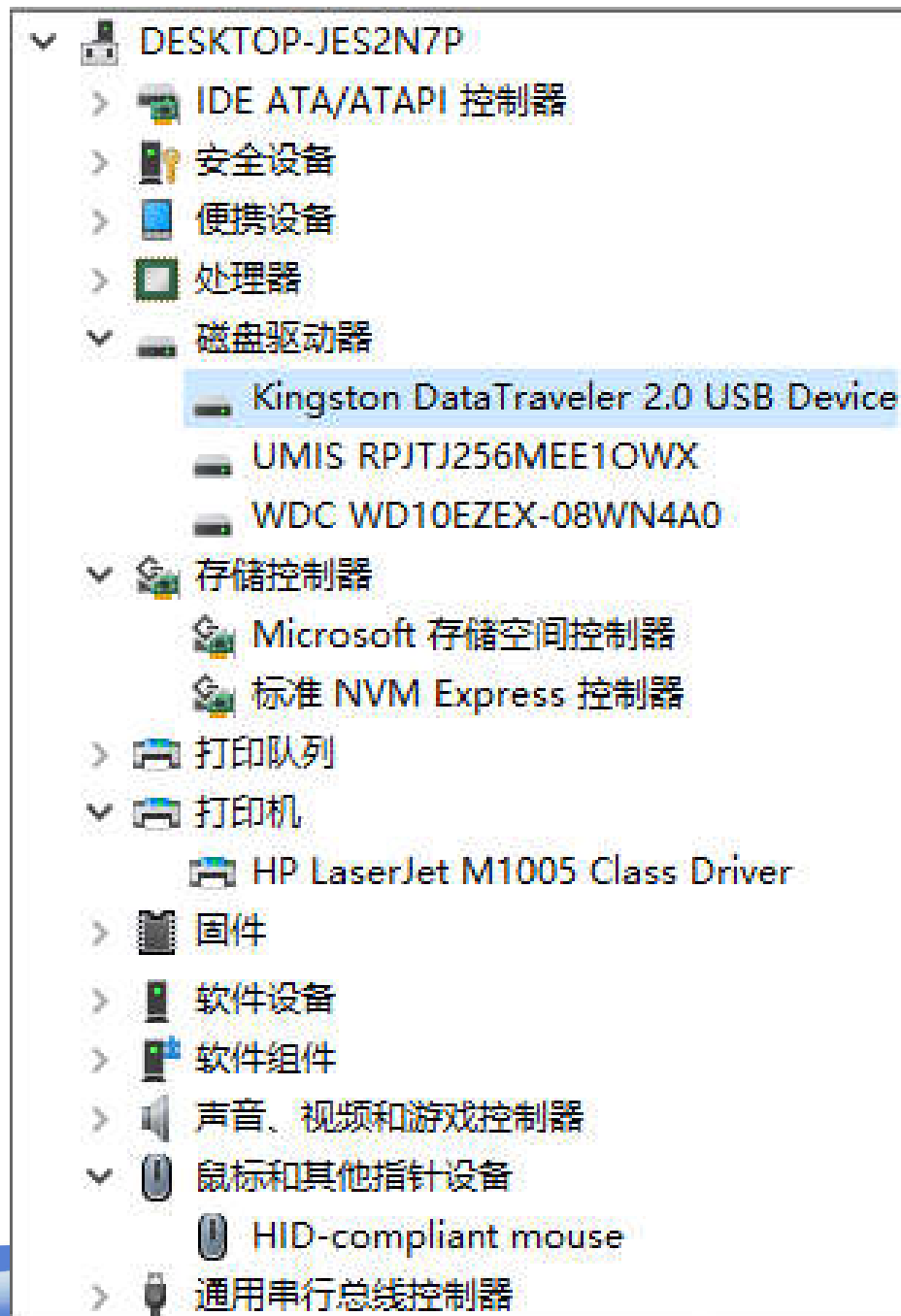


## ● 操作系统生成

- 根据硬件环境和用户需求，  
**配置**和**构建**操作系统。

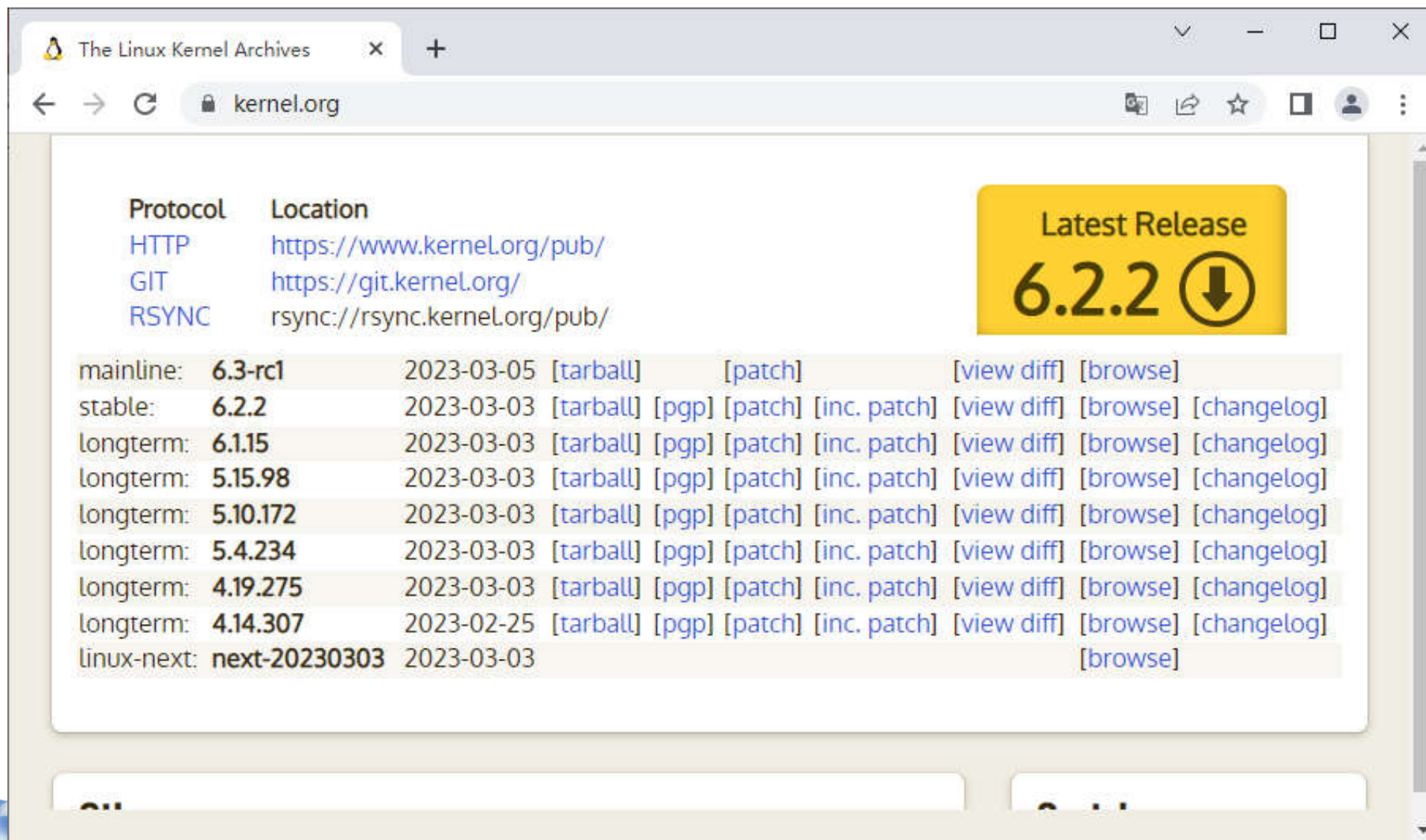
## ● 操作系统生成的前提

- 操作系统由模块构成
- 有交互式配置工具
- 有映像构建（**build**）工具
- **内核开源**



# 最新版本Linux内核 6.2.2

## ● <https://www.kernel.org/>



The screenshot shows the Linux Kernel Archives website. The browser tab is titled "The Linux Kernel Archives" and the address bar shows "kernel.org". The page features a yellow box on the right side that says "Latest Release 6.2.2" with a download icon. Below this, there is a table listing various kernel releases.

	Protocol	Location
	HTTP	<a href="https://www.kernel.org/pub/">https://www.kernel.org/pub/</a>
	GIT	<a href="https://git.kernel.org/">https://git.kernel.org/</a>
	RSYNC	<a href="rsync://rsync.kernel.org/pub/">rsync://rsync.kernel.org/pub/</a>

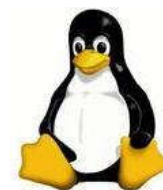
  

	Version	Date	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
mainline:	6.3-rc1	2023-03-05	[tarball]		[patch]		[view diff]	[browse]	
stable:	6.2.2	2023-03-03	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	6.1.15	2023-03-03	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	5.15.98	2023-03-03	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	5.10.172	2023-03-03	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	5.4.234	2023-03-03	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.19.275	2023-03-03	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.14.307	2023-02-25	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
linux-next:	next-20230303	2023-03-03						[browse]	

# Linux操作系统的生成

## ● 步骤

- 1、获取内核源码
- 2、配置内核
- 3、重新编译新内核
- 4、编译和安装模块
- 5、配置启动选项



# Linux操作系统的生成

- 1、获取内核源码（差距不太远的版本）

- <http://www.kernel.org/>

- # cd /usr/src**

- # tar zxvf linux-2.6.38-12.tar.gz**

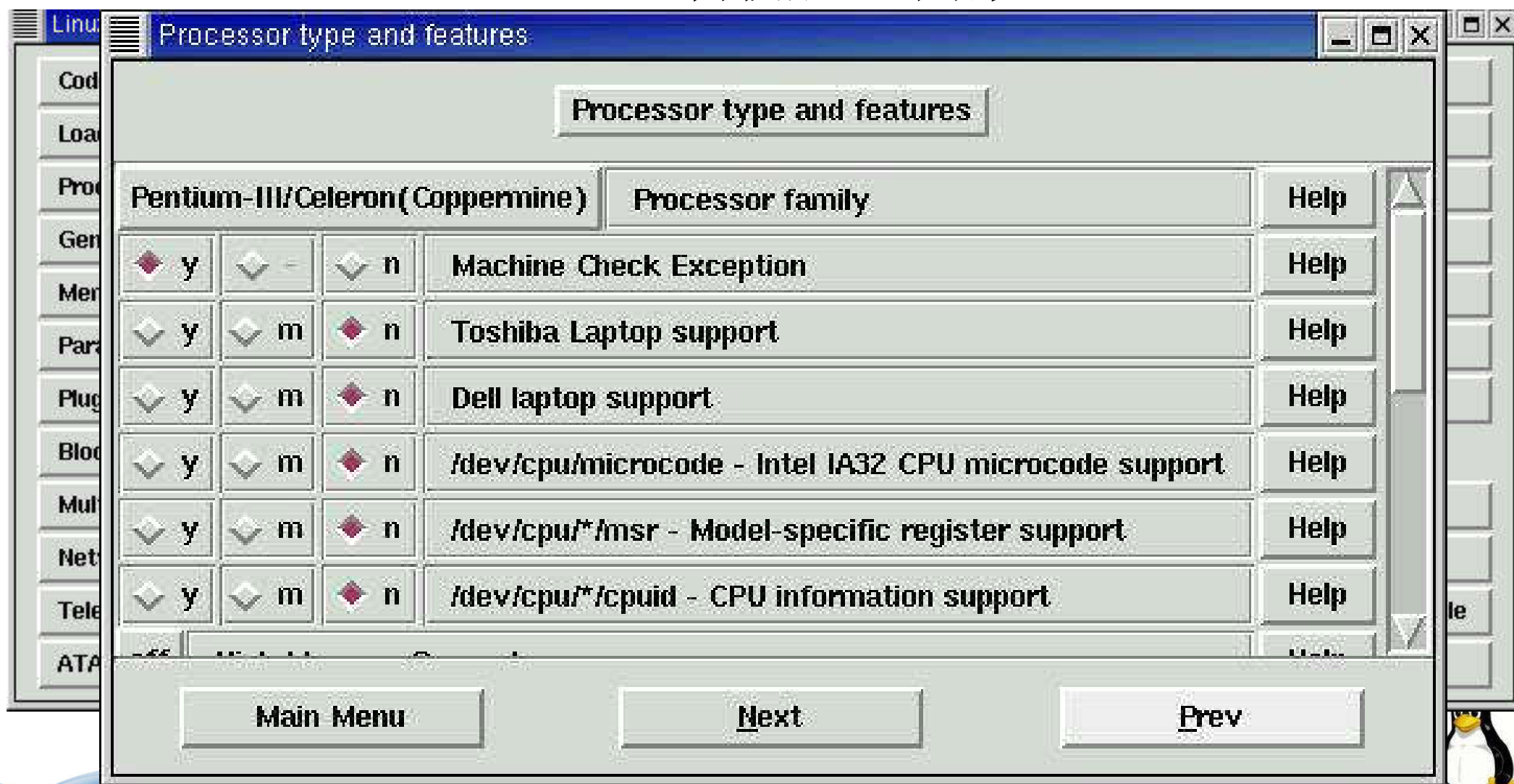
- 2、配置内核（模块和参数）

- #cd /usr/src/linux-2.6**

- #make xconfig**（xWindow图形窗口模式）

# # make xconfig

## Linux内核配置对话框



# Linux操作系统的生成

- 1、获取内核源码（差距不太远的版本）

- <http://www.kernel.org/>

- # cd /usr/src**

- # tar zxvf linux-2.6.38-12.tar.gz**

- 2、配置内核（模块和参数）

- #cd /usr/src/linux-2.6**

- #make xconfig**（xWindow图形窗口模式）

或：

- #make menuconfig**（文本选择界面，字符终端）



# # make menuconfig

```
root@susg-asus: /home/susg/LinuxKernel/linux-2.6.38.2
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
.config - Linux/i386 2.6.38.2 Kernel Configuration

                                USB support
Arrow keys navigate the menu.  <Enter> selects submenus --->.
Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
<M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
for Search.  Legend: [*] built-in [ ] excluded <M> module < >

^(-)
< >  USB Attached SCSI (NEW)
[ ]   The shared table of common (or usual) storage devices
*** USB Imaging devices ***
<M>   USB Mustek MDC800 Digital Camera support
<M>   Microtek X6USB scanner support
*** USB port drivers ***
<M>   USS720 parport driver
<M>   USB Serial Converter support --->
*** USB Miscellaneous drivers ***
<M>   EMI 6|2m USB Audio interface support
v(+)

<Select>  < Exit >  < Help >
```



# Linux操作系统的生成

## ● 3、重新编译新的内核

■ 内核配置的结果: **makefile**

# make dep      生成依赖dependency信息

# make clean    清除旧的编译结果

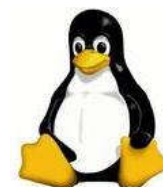
# make bzImage   **./arch/i386/boot/bzImage**

## ● 4、编译和安装模块

# make modules

# make modules\_install

模块被编译且安装到 /usr/lib/<内核版本号> 目录下。



# Linux操作系统的生成

## ● 5、配置启动选项（并启动新内核）

- cp bzImage /boot/bzImage

- GRUB (与发行版本有关)

  - ◆ 配置/boot/grub/grub.conf

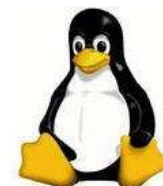
```
title newLinux build by Zhang San Feb.28, 2019  
root (hd0,1)  
kernel /boot/bzImage ro root=/dev/hda2
```

- LILO(早期版本)

  - ◆ 配置/etc/lilo.conf

```
image=/boot/bzImage  
label=newLinux build by Zhang San Feb.28, 2012
```

  - ◆ 命令 #lilo 使配置生效



# Linux操作系统的生成

- 重启计算机
- 重启新内核
  - 选择 **ubuntu**高级选项

