2019 ACM - IEEE CS Eckert-Mauchly Award

June 5, 2019

Dr. Mark D. Hill

Professor, University of Wisconsin-Madison

CITATION: For contributions to the design and evaluation of memory systems and parallel computers.

- In the 1980s Hill developed the **"3C" model of cache misses**.
  - The model was influential, as it led to important innovations such as victim caches and stream buffers, and is now a standard concept in computer architecture textbooks.
- Memory consistency models
- Transactional memory
- Evaluation of parallel computers

AWARD WINNER

Mark Hill

ACM-IEEE CS Eckert-Mauchly Award (2019)

ACM Fellows (2004)

IEEE Fellows (2000)

Eckert-Mauchly Award - computer architecture community's most prestigious award, named for John Presper Eckert and John William Mauchly, who collaborated on the design and construction of the Electronic Numerical Integrator and Computer (ENIAC)

https://www.acm.org/media-center/2019/june/eckert-mauchly-award-2019
https://awards.acm.org/award-winners/HILL_2155109

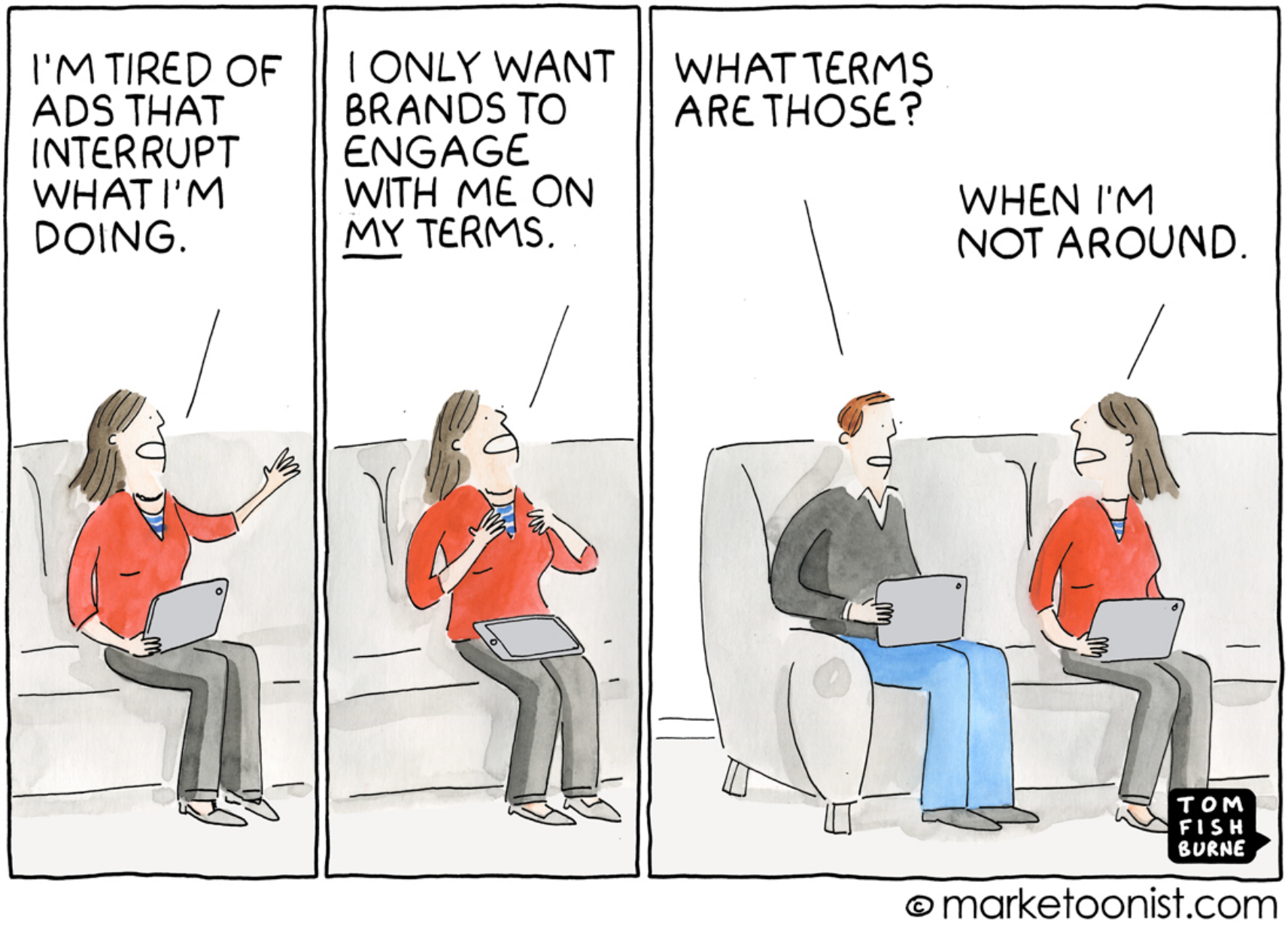
Types of Cache Misses: The 3C model

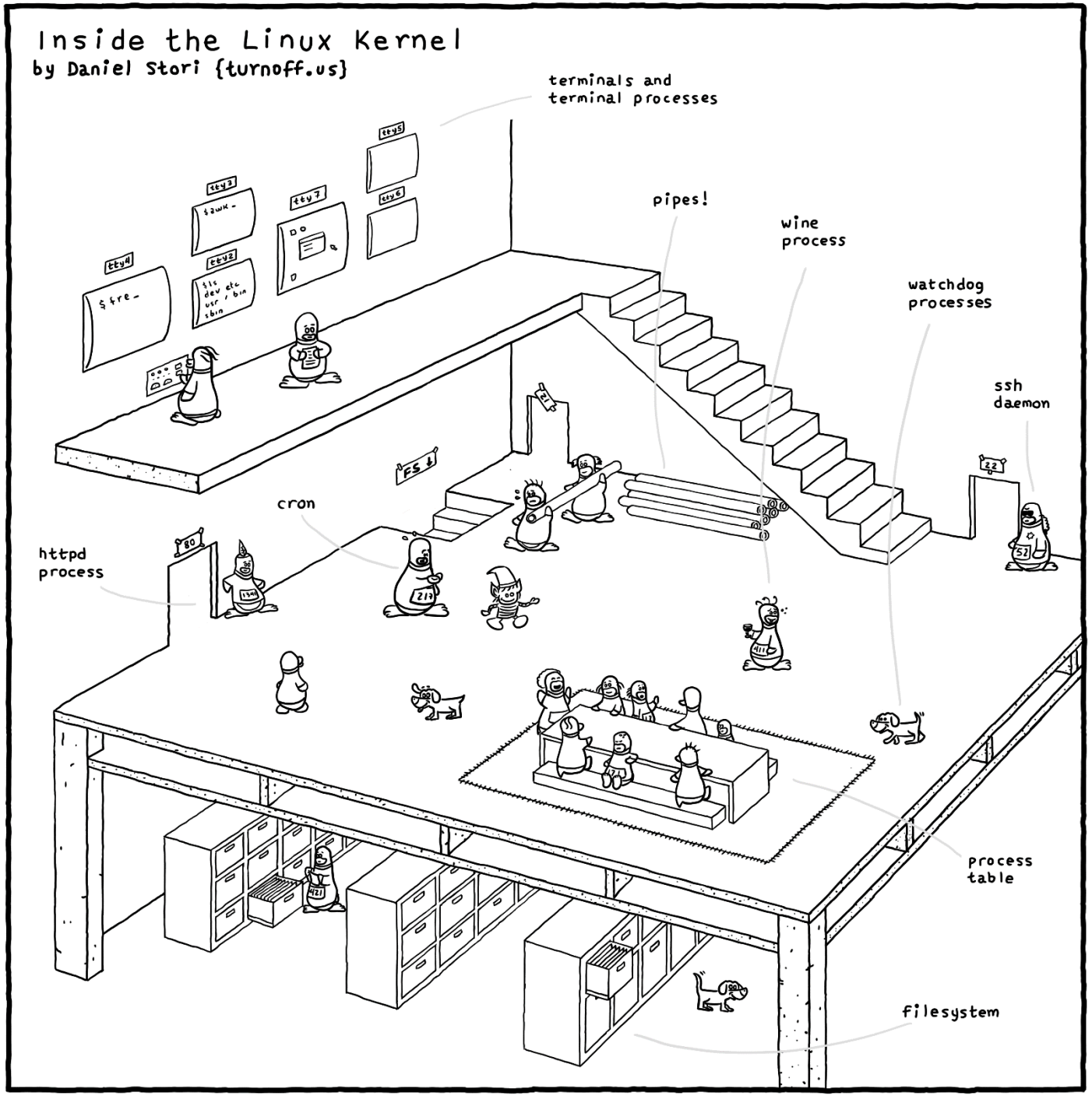
- **Compulsory:**
  - **Never accessed before; On the first access to a block; the block must be brought into the cache; also called cold start misses, or first reference misses.**
- **Capacity:**
  - **Occur because blocks are being discarded from cache because cache cannot contain all blocks needed for program execution (program working set is much larger than cache capacity); Accessed long ago and already replaced**
- **Conflict:**
  - **In the case of set associative or direct mapped block placement strategies, conflict misses occur when several blocks are mapped to the same set or block frame; also called collision misses or interference misses.**
- *Coherence (4C): explained more in future advanced course*
  - *Occur in multi-core/multi-processor, or uniprocessor w/ IO writes*
  - *Multiple physical copies of one logical location*
  - *Copies become inconsistent when writes happen*

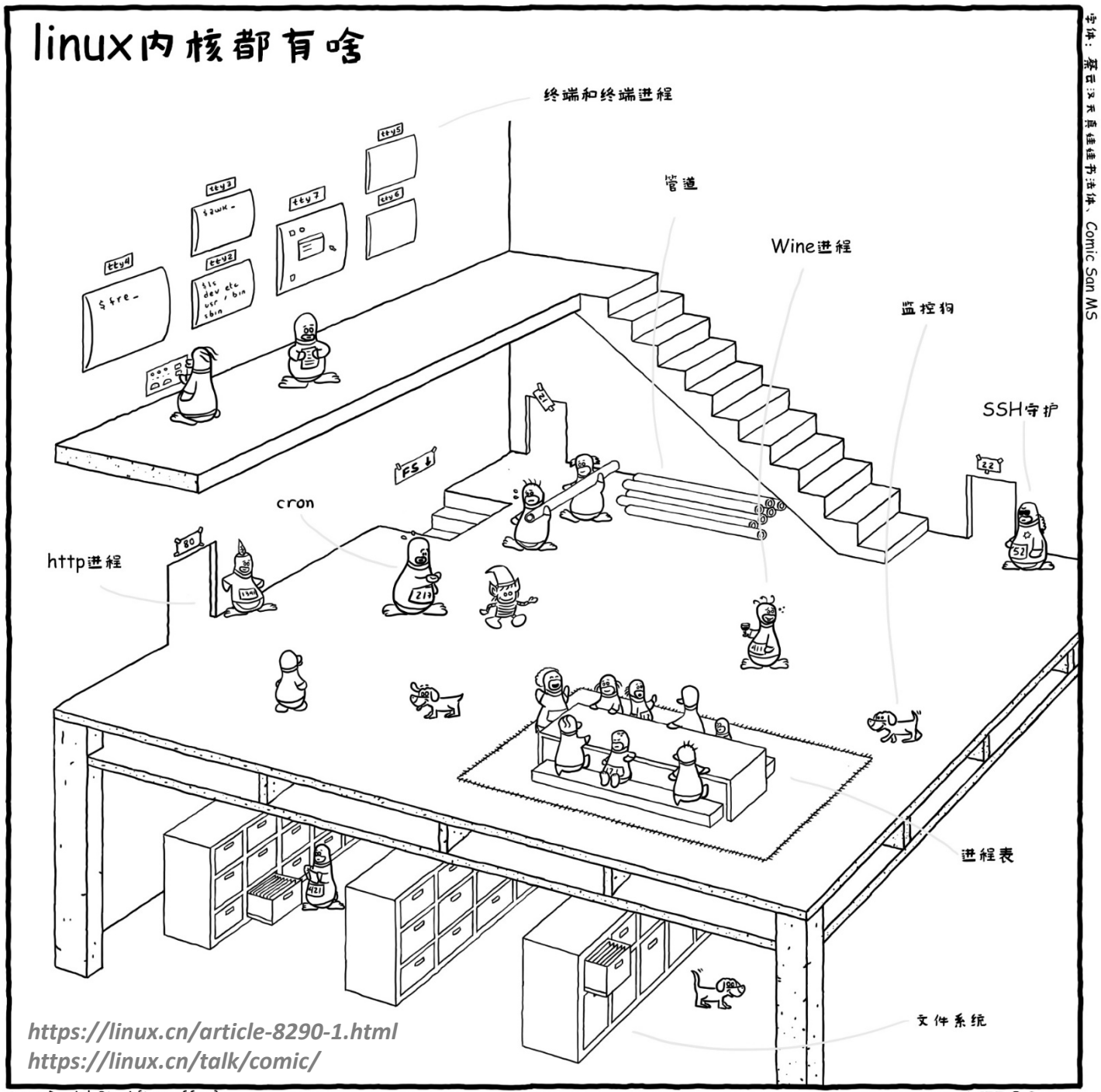
复习：

1. 磁盘的技术指标：道密度、位密度
2. 平均寻址时间 = 寻道时间 + 等待时间
3. 平均存取时间 = 平均寻道时间 + 平均旋转等待时间 + 数据传输时间
4. 希捷16TB容量的Exos X企业级硬盘
   - ① 3.5英寸磁记录产品，九碟装，充填氦气，2019年6月4日上市
   - ② 转速7200RPM，缓存256MB
   - ③ 接口可选SATA 6 Gbps或SAS 12 Gbps，内部持续传输率261 MB/s，4K随机读写速度170/440 IOPS
   - ④ 访问平均延迟：4.16ms
   - ⑤ 功耗：待机5.0W，最大10.0W，随机读写6.3W
5. 补充知识：SSD、SCM、NVMe、NVDIMM

Specifications	SATA 6Gb/s	12Gb/s SAS
Capacity	16TB	16TB
Standard Model FastFormat™ (512e/4Kn)[1]	ST16000NM001G	ST16000NM002G
SED Model FastFormat (512e/4Kn)[1,2]	ST16000NM003G	ST16000NM004G
SED-FIPS FastFormat (512e/4Kn)[1,2]	—	ST16000NM009G
Features		
Helium Sealed-Drive Design	Yes	Yes
Protection Information (T10 DIF)	—	Yes
SuperParity	Yes	Yes
Low Halogen	Yes	Yes
PowerChoice™ Idle Power Technology	Yes	Yes
PowerBalance™ Power/Performance Technology	Yes	Yes
Hot-Plug Support[3]	Yes	Yes
Cache, Multisegmented (MB)	256	256
Organic Solderability Preservative	Yes	Yes
RSA 2048 Firmware Verification (SD&D)	Yes	Yes
Reliability/Data Integrity		
Mean Time Between Failures (MTBF, hours)	2,500,000	2,500,000
Reliability Rating @ Full 24×7 Operation (AFR)	0.35%	0.35%
Nonrecoverable Read Errors per Bits Read	1 sector per 10E15	1 sector per 10E15
Power-On Hours per Year (24×7)	8760	8760
512e Sector Size (Bytes per Sector)	512	512, 520, 528
4Kn Sector Size (Bytes per Sector)	4096	4096, 4160, 4224
Limited Warranty (years)	5	5
Performance		
Spindle Speed (RPM)	7200RPM	7200RPM
Interface Access Speed (Gb/s)	6.0, 3.0	12.0, 6.0, 3.0
Max. Sustained Transfer Rate OD (MB/s,MiB/s)	261, 249	261, 249
Random Read/Write 4K QD16 WCD (IOPS)	170/440	170/440
Average Latency (ms)	4.16	4.16
Interface Ports	Single	Dual
Rotation Vibration @ 20-1500 Hz (rad/sec²)	12.5	12.5
Power Consumption		
Idle A (W) Average	5.00W	5.00W
Max Operating, Random Read/Write 4K/16Q (W)	10.0, 6.3	10.2, 6.2
Power Supply Requirements	+12 V and +5 V	+12 V and +5 V
Environmental		
Temperature, Operating (°C)	5°C – 60°C	5°C – 60°C
Vibration, Nonoperating: 2 to 500Hz (Grms)	2.27	2.27
Shock, Operating 2ms (Read/Write) (Gs)	50	50
Shock, Nonoperating 2ms (Gs)	200	200
Physical		
Height (mm/in, max)[4]	26.11mm/1.028in	26.11mm/1.028in
Width (mm/in, max)[4]	101.85mm/4.010in	101.85mm/4.010in
Depth (mm/in, max)[4]	147.00mm/5.787in	147.00mm/5.787in
Weight (g/lb)	670g/1.477lb	670g/1.477lb


I'M TIRED OF ADS THAT INTERRUPT WHAT I'M DOING.
I ONLY WANT BRANDS TO ENGAGE WITH ME ON MY TERMS.
WHAT TERMS ARE THOSE?
WHEN I'M NOT AROUND.
TOM FISH BURNE
© marketoonist.com


Inside the Linux Kernel
by Daniel Stori {turnoff.us}
terminals and terminal processes
pipes!
wine process
watchdog processes
ssh daemon
cron
httpd process
process table
filesystem


linux内核都有啥
终端和终端进程
管道
Wine进程
监控狗
SSH守护
cron
http进程
进程表
文件系统
https://linux.cn/article-8290-1.html
https://linux.cn/talk/comic/
字体：基于汉真广角娃娃书法体、Comic San MS
原创：Daniel Stori {turnoff.us}
汉化：LCTT @ Bestony

B0911006Y-01　2018-2019学年春季学期

计算机组成原理

第26讲　中断

中断请求如何产生？如何被CPU处理？如何设置中断的屏蔽字？I/O中断的流程

主讲教师：　张　科

2019年6月10日

中国科学院大学 University of Chinese Academy of Sciences
中科院计算所 INSTITUTE OF COMPUTING TECHNOLOGY,CAS

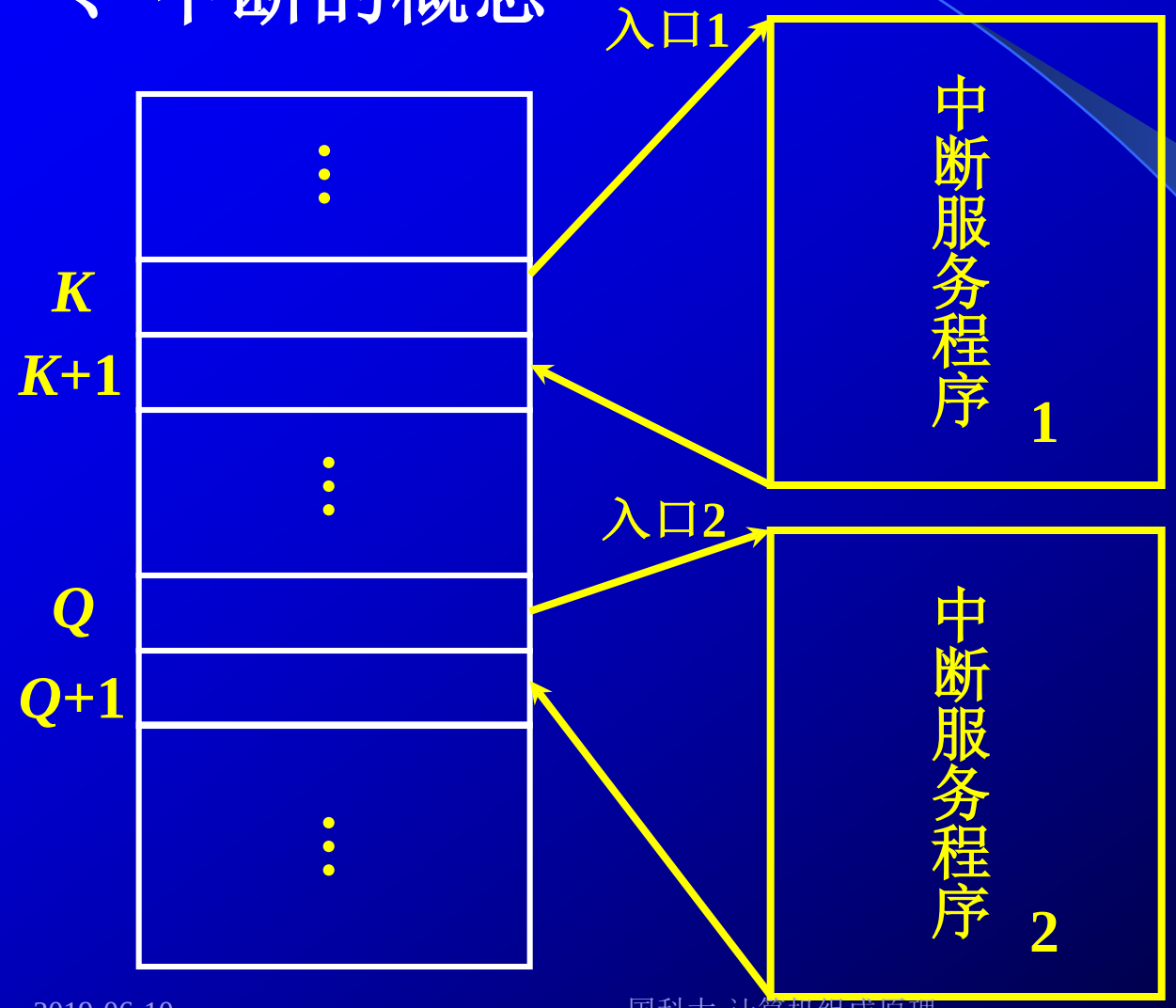
第11章　中 断

11.1 中断系统 （教材P358-第8.4节）

11.2 I/O中断 （教材P194-第5.5节）

11.1 中断系统

一、中断的概念



异常情况或特殊请求时，停止现行程序的运行，转向对这些异常情况或特殊请求的处理，处理结束后再返回到现行程序的间断处

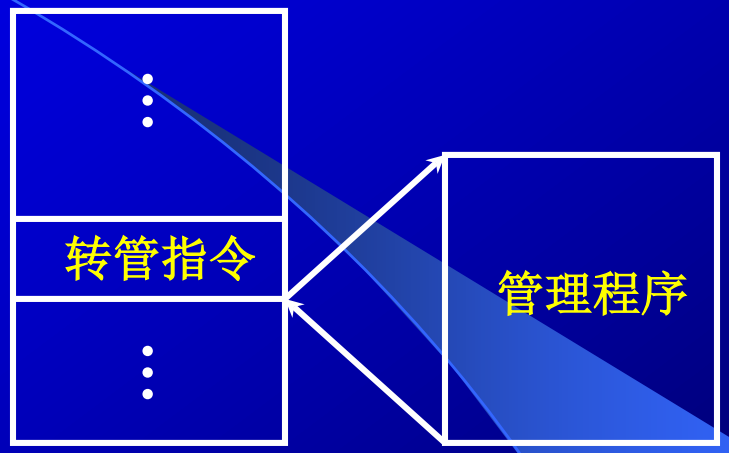
二、概述

1. 引起中断的各种因素

(1) 人为设置的中断

如　转管指令

完成系统调用的
Trap陷阱指令



(2) 程序性事故　溢出、操作码不能识别、除法非法

(3) 硬件故障

(4) I/O 设备

(5) 外部事件　　用 键盘中断 现行程序

2. 中断系统需解决的问题

11.1

(1) 各中断源 如何 向 CPU 提出请求？

(2) 各中断源 同时 提出 请求 怎么办？

(3) CPU 什么 条件、什么 时间、以什么 方式 响应中断？

(4) 如何 保护现场？

(5) 如何 寻找入口地址？

(6) 如何 恢复现场，如何 返回？

(7) 处理中断的过程中又 出现新的中断 怎么办？

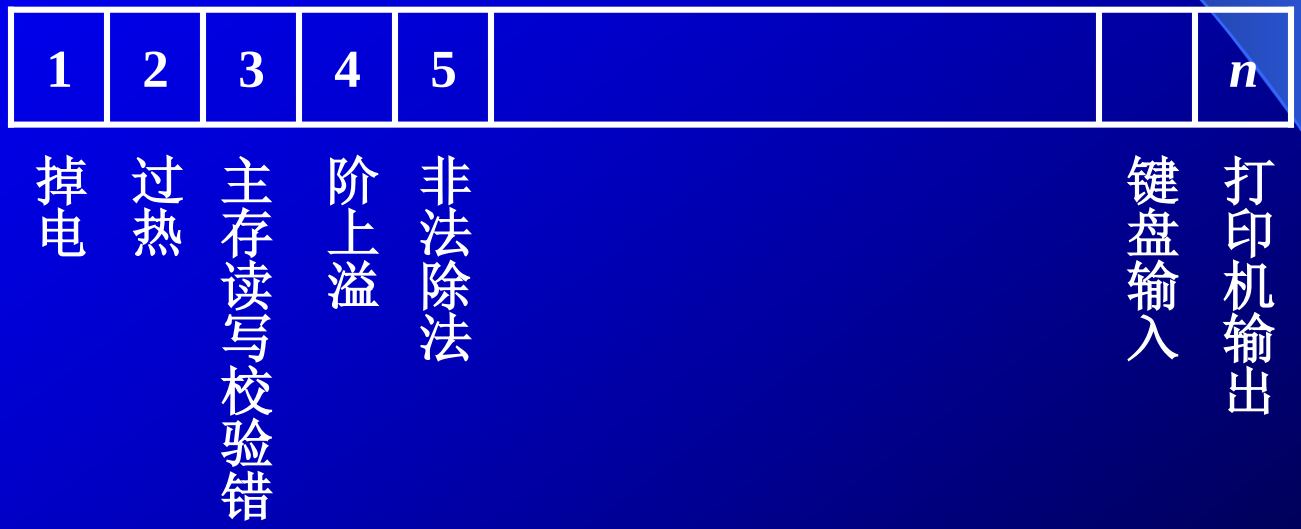
硬件 + 软件

二、中断请求标记和中断判优逻辑 11.1

1. 中断请求标记 INTR（中断请求标记触发器）

一个请求源 一个 INTR

多个INTR 组成 中断请求标记寄存器



INTR 可以 分散 在各个中断源的 接口电路中

INTR 亦可 集中 在 CPU 的中断系统 内

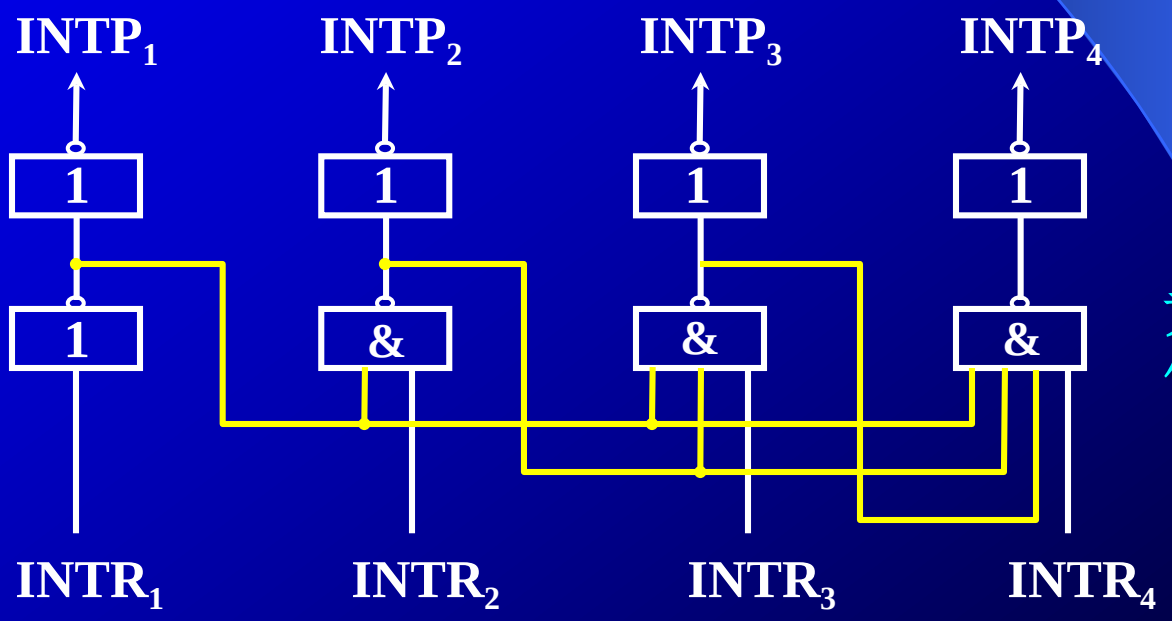
2. 中断判优逻辑（中断源优先级） 11.1

(1) 硬件实现（排队器）

① 分散 在各个中断源的 接口电路中　链式排队器

参见 11.2 I/O中断

② 集中 在 CPU 内

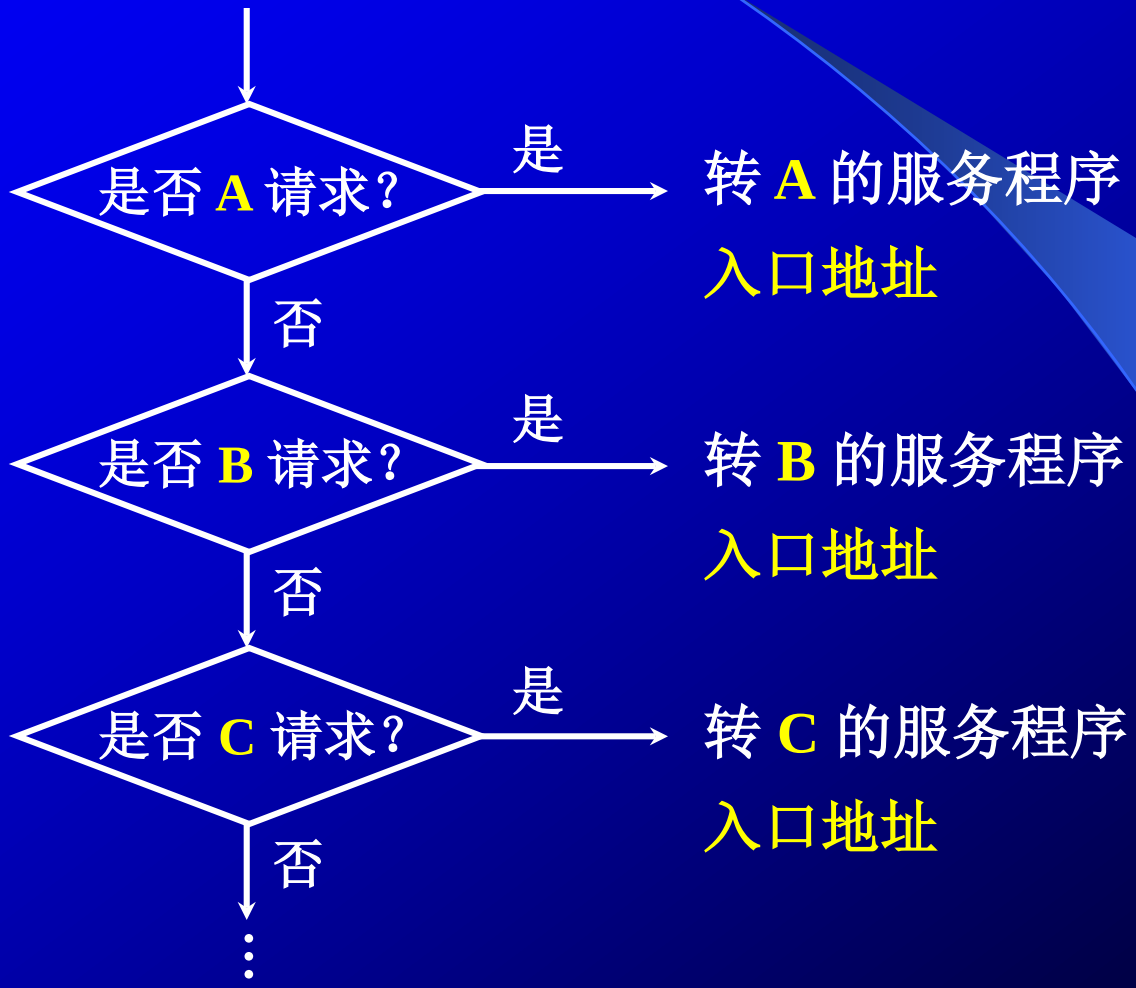


黄线“封住”低优先级中断源请求

$INTR_1$、$INTR_2$、$INTR_3$、$INTR_4$ 优先级 按 降序 排列

(2) 软件实现（程序查询）

A、B、C 优先级按 降序 排列



三、中断服务程序入口地址的寻找

11.1

1. 硬件向量法（硬件→向量地址→入口地址）

向量地址　12H、13H、14H

入口地址　200、 300、 400

2. 软件查询法（与软件排队配合）

11.1

八个中断源 1，2，… 8 按 降序 排列

中断识别程序（入口地址 M）

地 址	指 令	说 明
M	SKP DZ $1^{\#}$	$1^{\#}$ D = 0 跳 (D为完成触发器)
	JMP $1^{\#}$ SR	$1^{\#}$ D = 1 转$1^{\#}$ 服务程序
	SKP DZ $2^{\#}$	$2^{\#}$ D = 0 跳下下条指令
	JMP $2^{\#}$ SR	$2^{\#}$ D = 1 转$2^{\#}$ 服务程序
	⋮	
	SKP DZ $8^{\#}$	$8^{\#}$ D = 0 跳
	JMP $8^{\#}$ SR	$8^{\#}$ D = 1 转$8^{\#}$ 服务程序

P307

State Register

四、中断响应

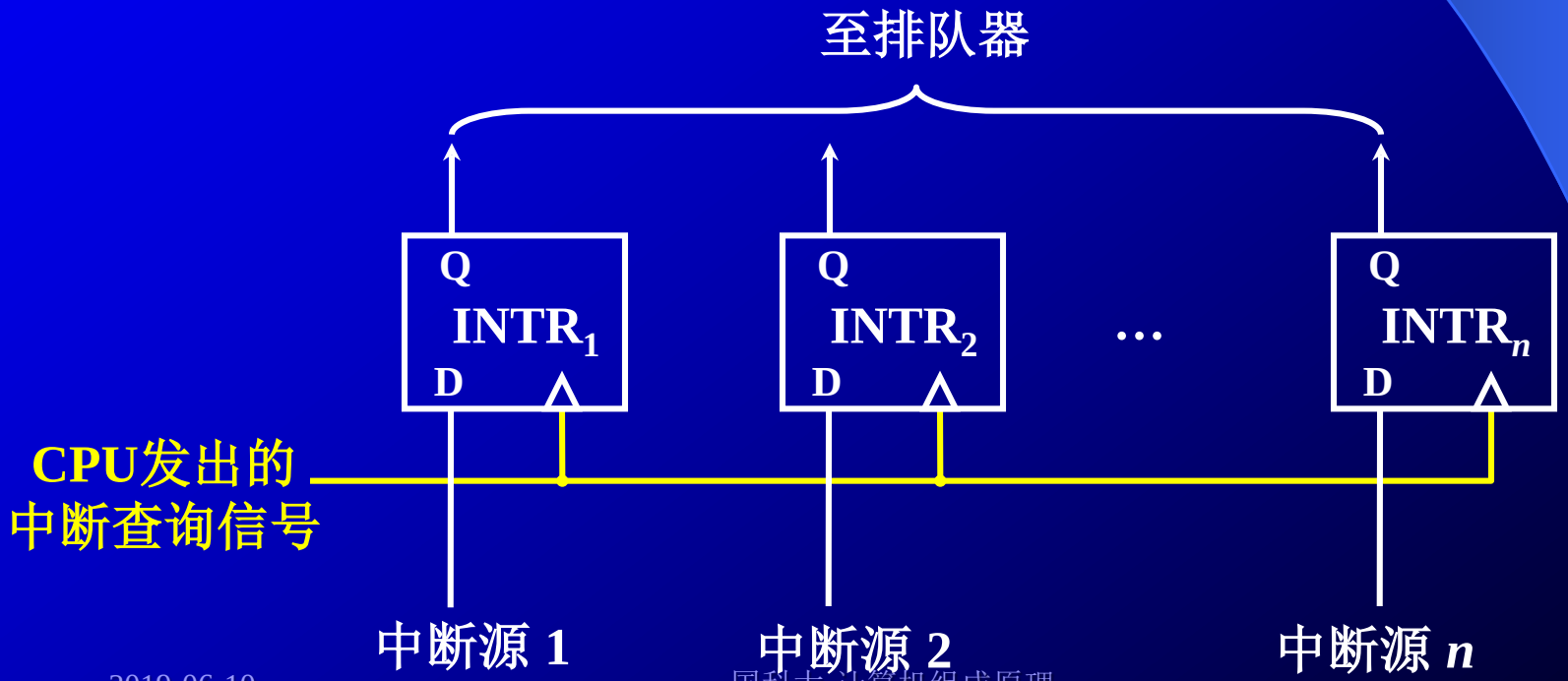
11.1

1. 响应中断的 条件

允许中断触发器　EINT = 1 (开中断置1，关中断置0)

2. 响应中断的 时间

指令执行周期结束时刻由CPU 发查询信号



3. 中断隐指令

CPU在中断周期内由硬件自动完成的一条在机器指令系统中没有的指令，以执行一系列操作

11.1

(1) 保护程序断点

断点(PC当前内容)存于 特定地址内（ 0 号地址）或 断点 进栈

(2) 寻找服务程序入口地址

向量地址 ⟶ PC （硬件向量法）

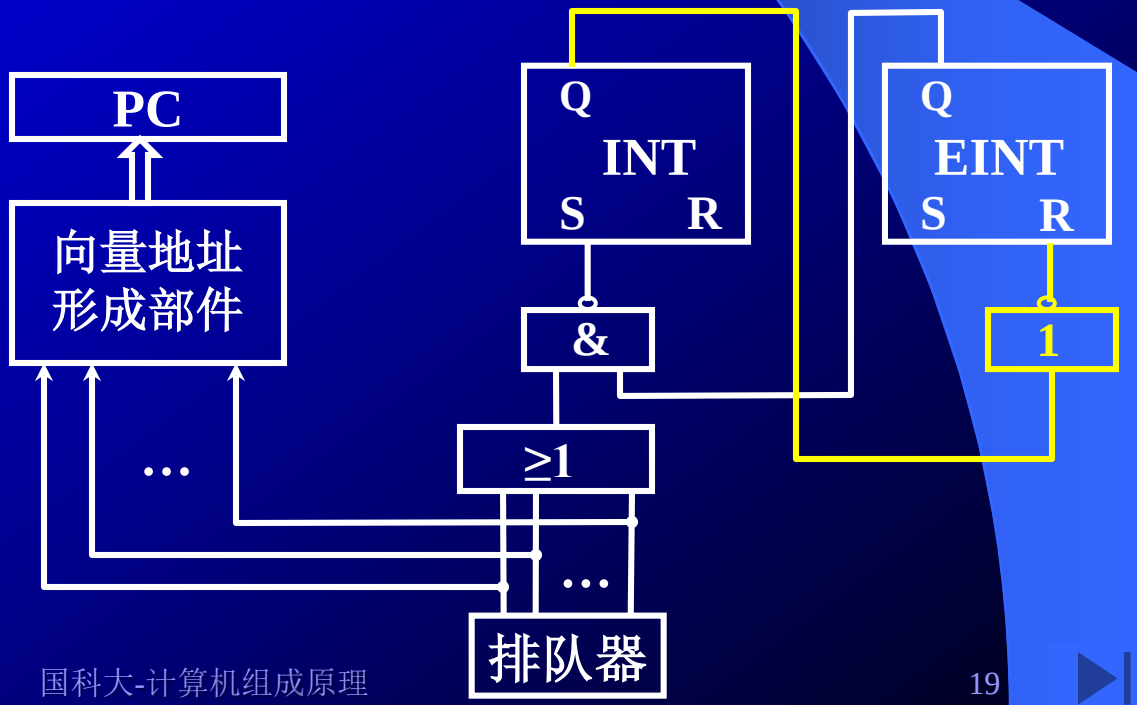
中断识别程序 入口地址 M ⟶ PC （软件查询法）

(3) 硬件 关中断

INT 中断标记

EINT 允许中断

R – S 触发器



五、保护现场和恢复现场

11.1

1. 保护现场
   - 断点 保护　　中断隐指令 完成
   - 寄存器 内容保护　中断服务程序 完成

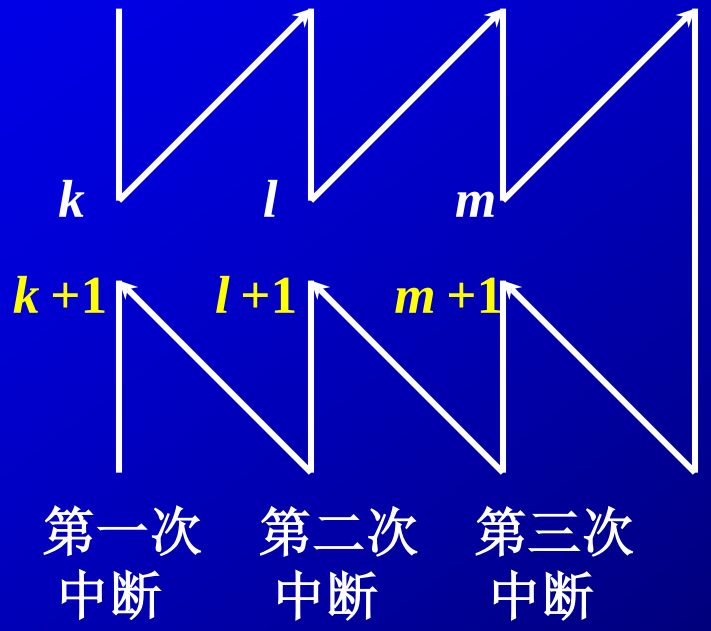
2. 恢复现场　中断服务程序 完成

中断服务程序

六、中断屏蔽技术

11.1

1. 多重中断的概念

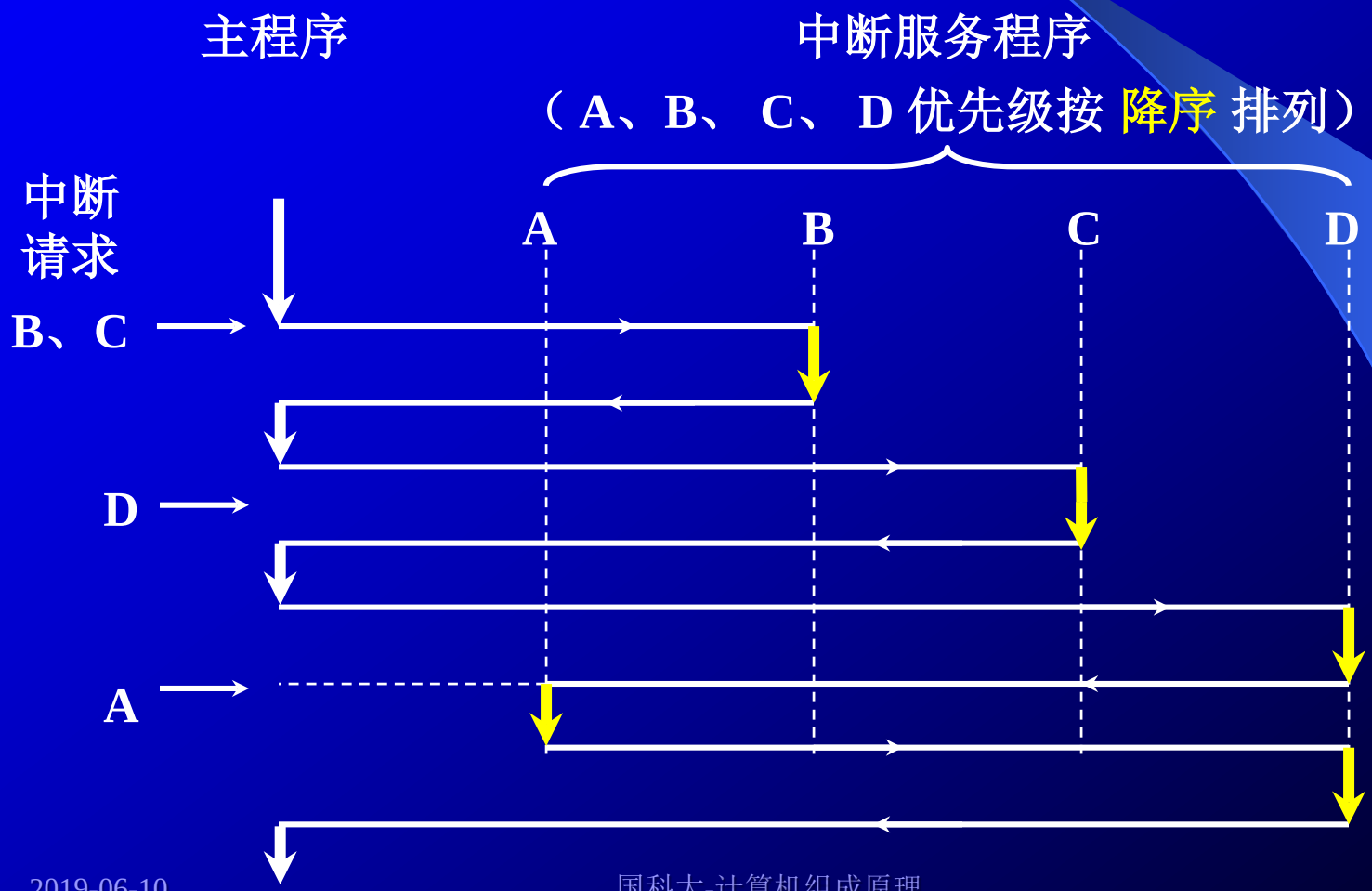


程序断点 $k+1$ ， $l+1$ ， $m+1$

2. 实现多重中断的条件

(1) 提前 设置 开中断 指令

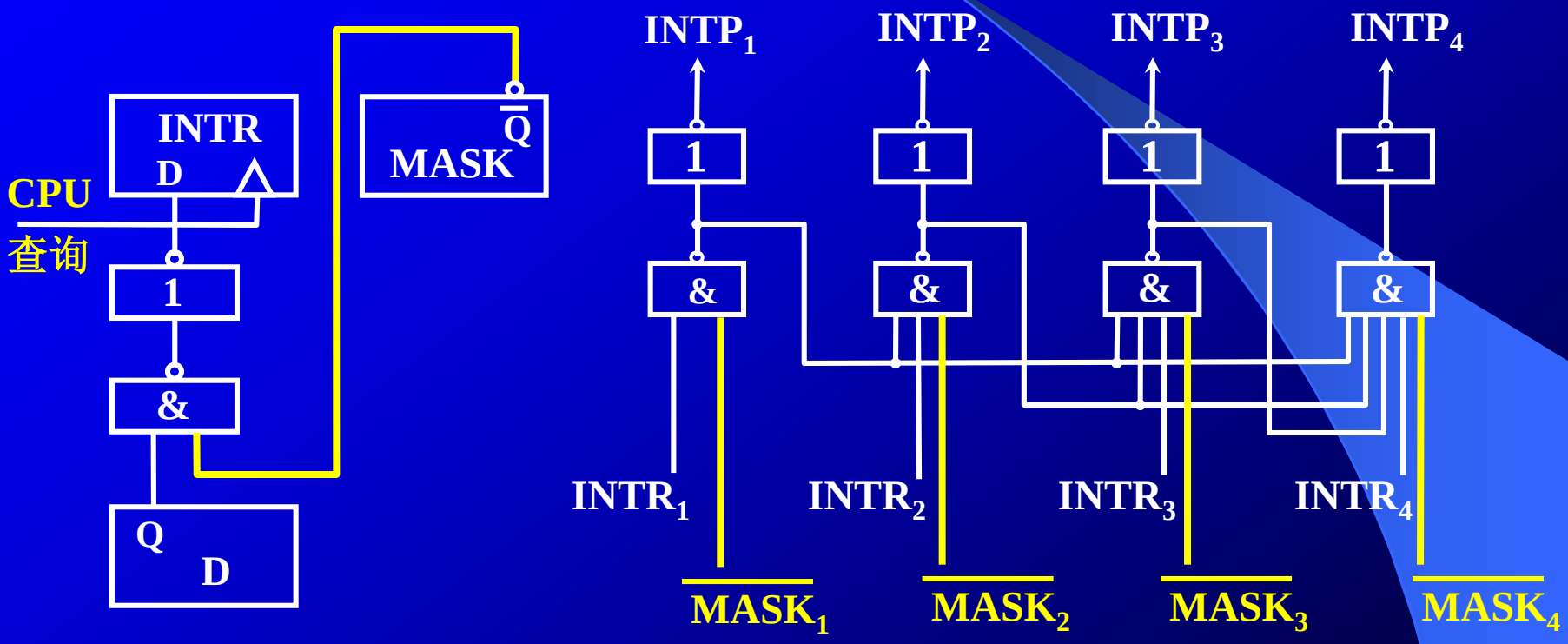
(2) 优先级别高 的中断源 有权中断优先级别低 的中断源



3. 屏蔽技术 (高优先级屏蔽低优先级)

11.1

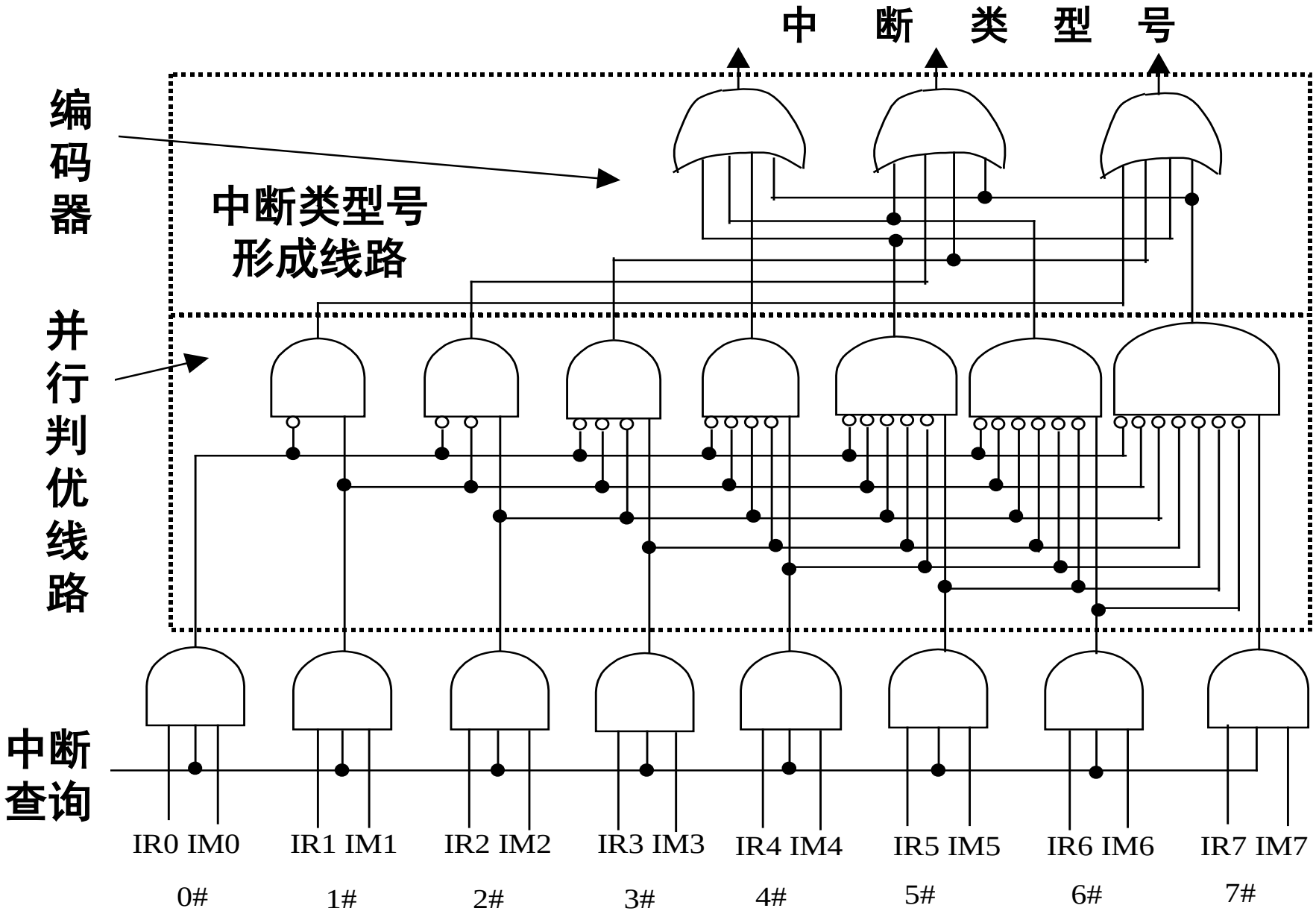
(1) 屏蔽触发器的作用



MASK = 0（未屏蔽）　　$MASK_i$ = 1 （屏蔽）

INTR　能被置 “1”　　$INTP_i$ = 0 （不能被排队选中）

中断优先权编码器


中 断 类 型 号
编码器
中断类型号
形成线路
并行判优线路
中断查询
IR0 IM0
IR1 IM1
IR2 IM2
IR3 IM3
IR4 IM4
IR5 IM5
IR6 IM6
IR7 IM7
0#
1#
2#
3#
4#
5#
6#
7#

(2) 屏蔽字

在中断服务程序中设置屏蔽字：
屏蔽字是位于硬件中断控制器里面的寄存器，
可通过OS内核态指令进行设置和修改

16个中断源 1，2，3，… 16 按 降序 排列

优先级	屏 蔽 字
1(最高)	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3	0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
4	0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
5	0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
6	0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
⋮	⋮
15	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
16(最低)	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

(3) 屏蔽技术可改变处理优先等级

11.1

优先等级

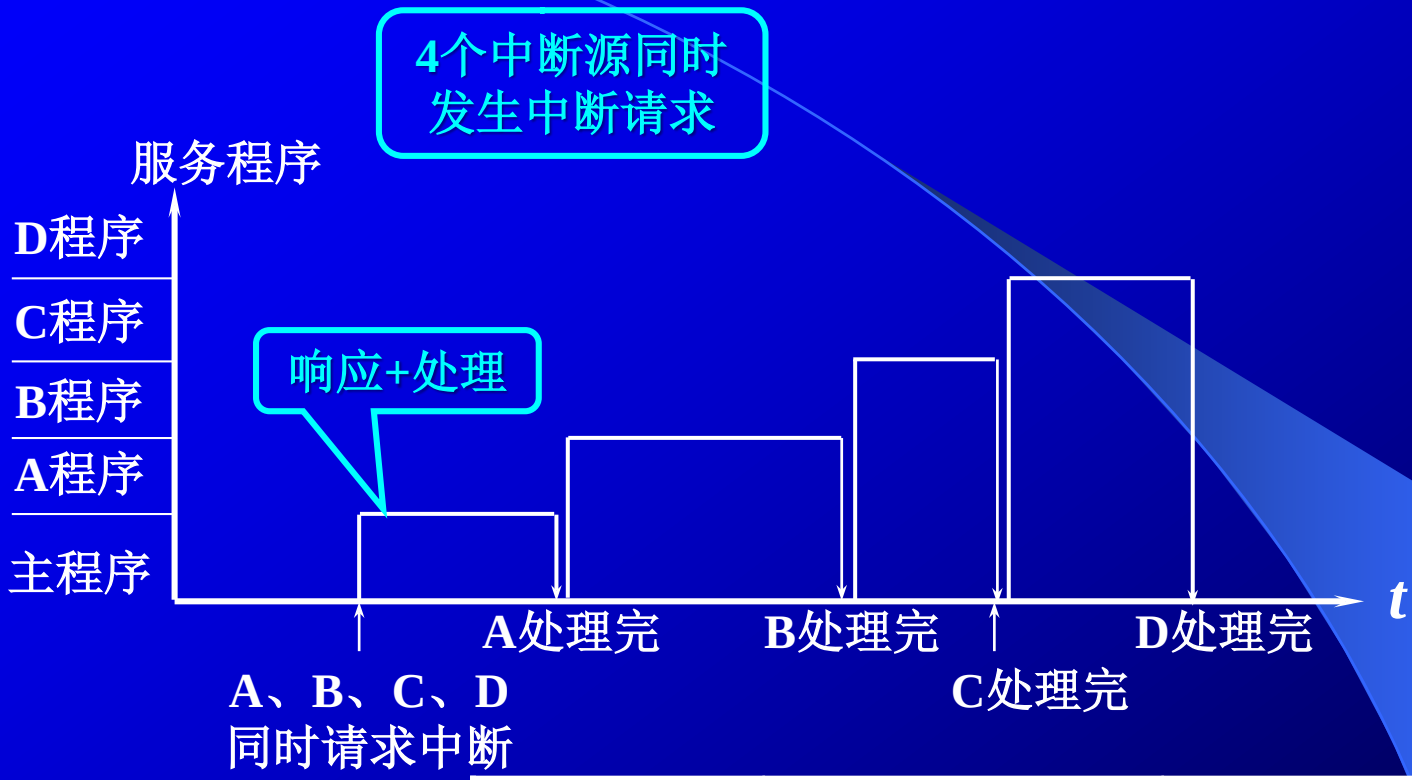
- 响应优先级　　不可改变 (硬件设计)
- 处理优先级　　可改变（通过设置新的屏蔽字）

中断源	原屏蔽字	新屏蔽字
A	1 1 1 1	1 1 1 1
B	0 1 1 1	0 1 0 0
C	0 0 1 1	0 1 1 0
D	0 0 0 1	0 1 1 1

响应优先级 **A→B→C→D** 降序排列

处理优先级 **A→D→C→B** 降序排列

(3) 屏蔽技术可改变处理优先等级

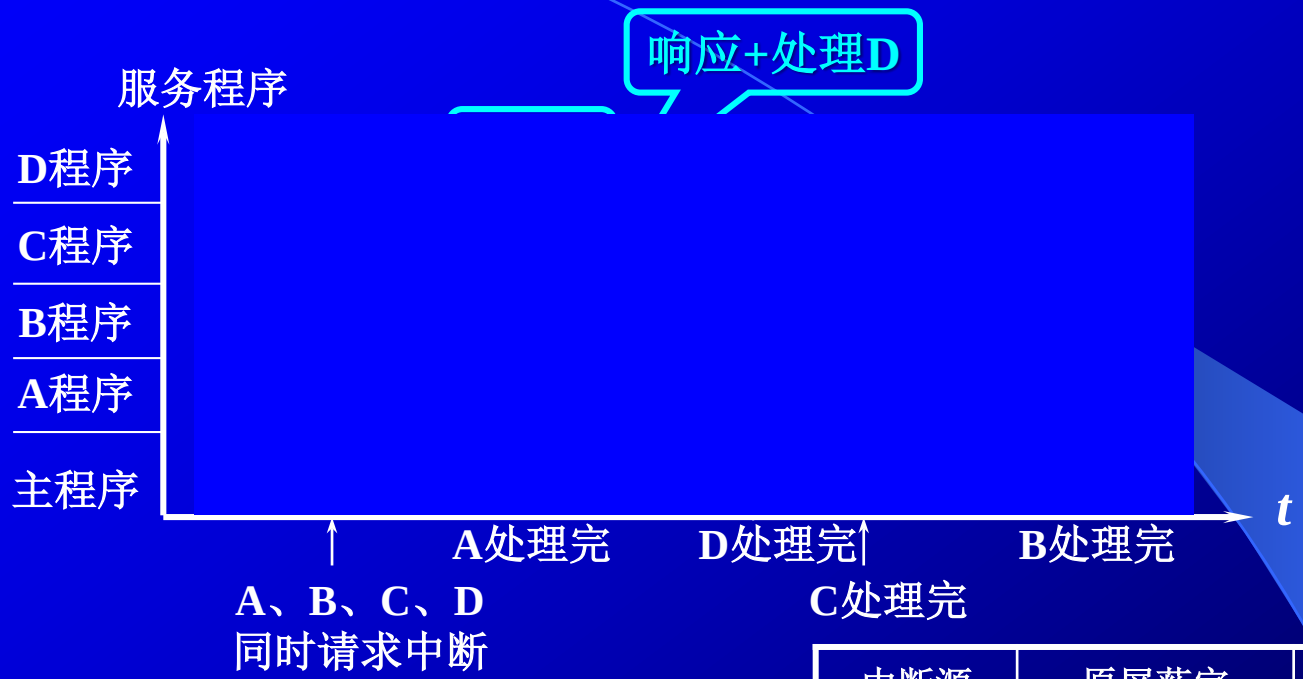


CPU 执行程序轨迹
（原屏蔽字）

中断源	原屏蔽字	新屏蔽字
A	1 1 1 1	1 1 1 1
B	0 1 1 1	0 1 0 0
C	0 0 1 1	0 1 1 0
D	0 0 0 1	0 1 1 1

(3) 屏蔽技术可改变处理优先等级

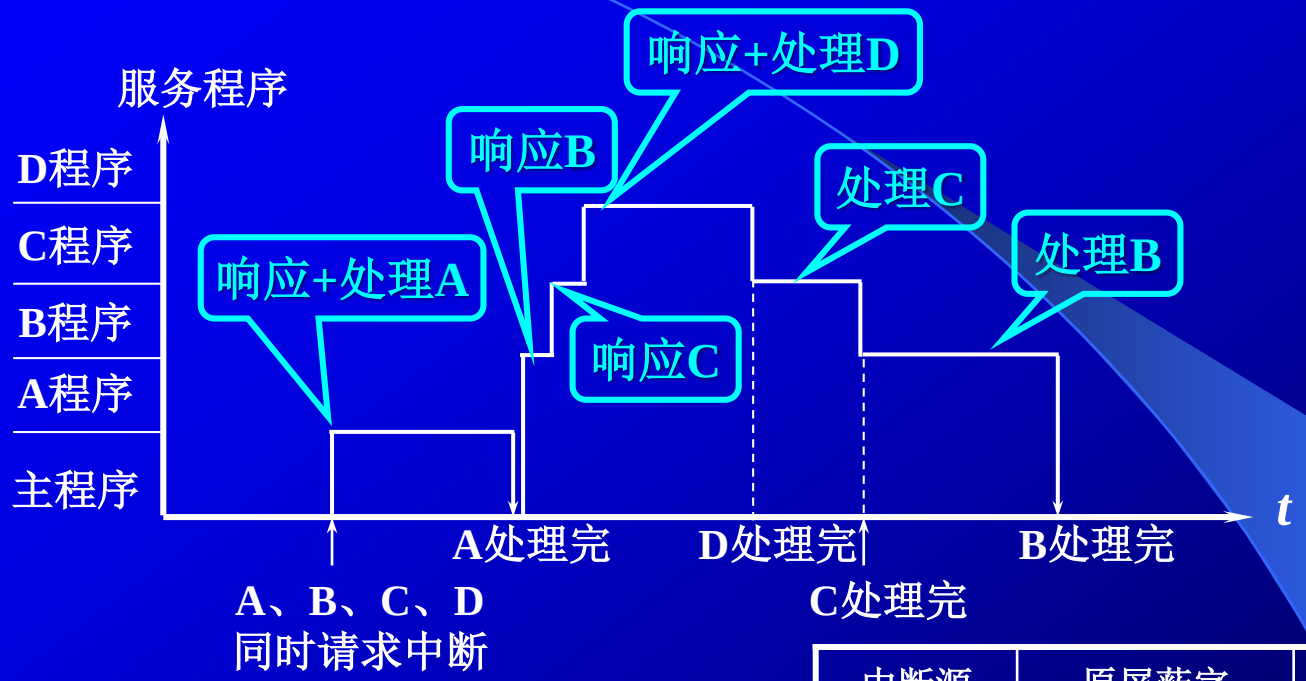
11.1



CPU 执行程序轨迹
（新屏蔽字）

中断源	原屏蔽字	新屏蔽字
A	1 1 1 1	1 1 1 1
B	0 1 1 1	0 1 0 0
C	0 0 1 1	0 1 1 0
D	0 0 0 1	0 1 1 1

(3) 屏蔽技术可改变处理优先等级



CPU 执行程序轨迹
（新屏蔽字）

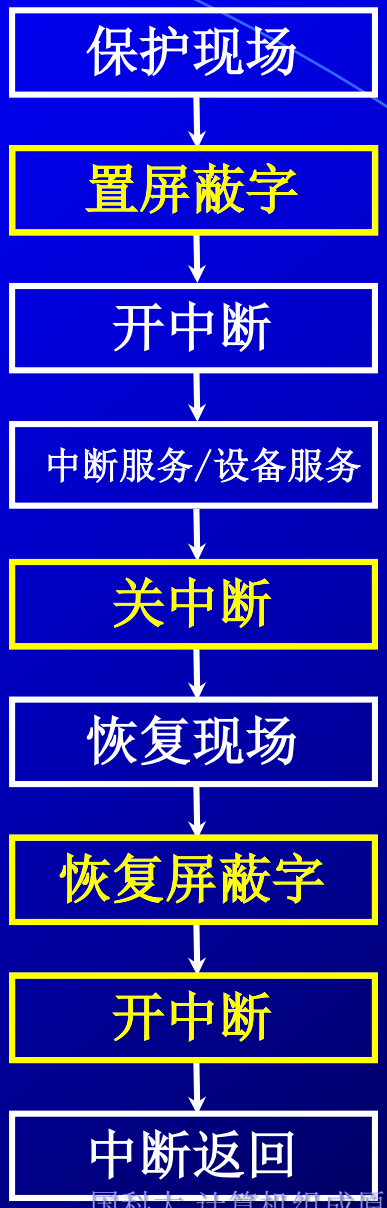
中断源	原屏蔽字	新屏蔽字
A	1 1 1 1	1 1 1 1
B	0 1 1 1	0 1 0 0
C	0 0 1 1	0 1 1 0
D	0 0 0 1	0 1 1 1

(4) 屏蔽技术的其他作用

可以 人为地屏蔽 某个中断源的请求

便于程序控制

(5) 新屏蔽字的设置



4. 多重中断的断点保护

(1) 断点进栈　　中断隐指令 完成

(2) 断点存入“ 0 ”地址　　中断隐指令 完成

中断周期　　**0 ⟶ MAR**

命令存储器写

PC ⟶ MDR　　断点 ⟶ **MDR**

(MDR) ⟶ 存入存储器

假设有三次中断，三个断点都存入 “ 0 ” 地址

？ 如何保证断点不丢失？

(3) 程序断点存入“0”地址的断点保护11.1

地 址	内 容	说 明
0	××××	存程序断点
5	JMP SERVE	5 为向量地址
SERVE	STA SAVE	保护现场
	⋮	
	LDA 0	} 0 地址内容转存
置屏蔽字 →	STA RETURN	
	ENI	开中断
	⋮	} 其他服务内容
	LDA SAVE	恢复现场
	JMP @ RETURN	间址返回
SAVE	××××	存放 ACC 内容
RETURN	××××	转存 0 地址内容

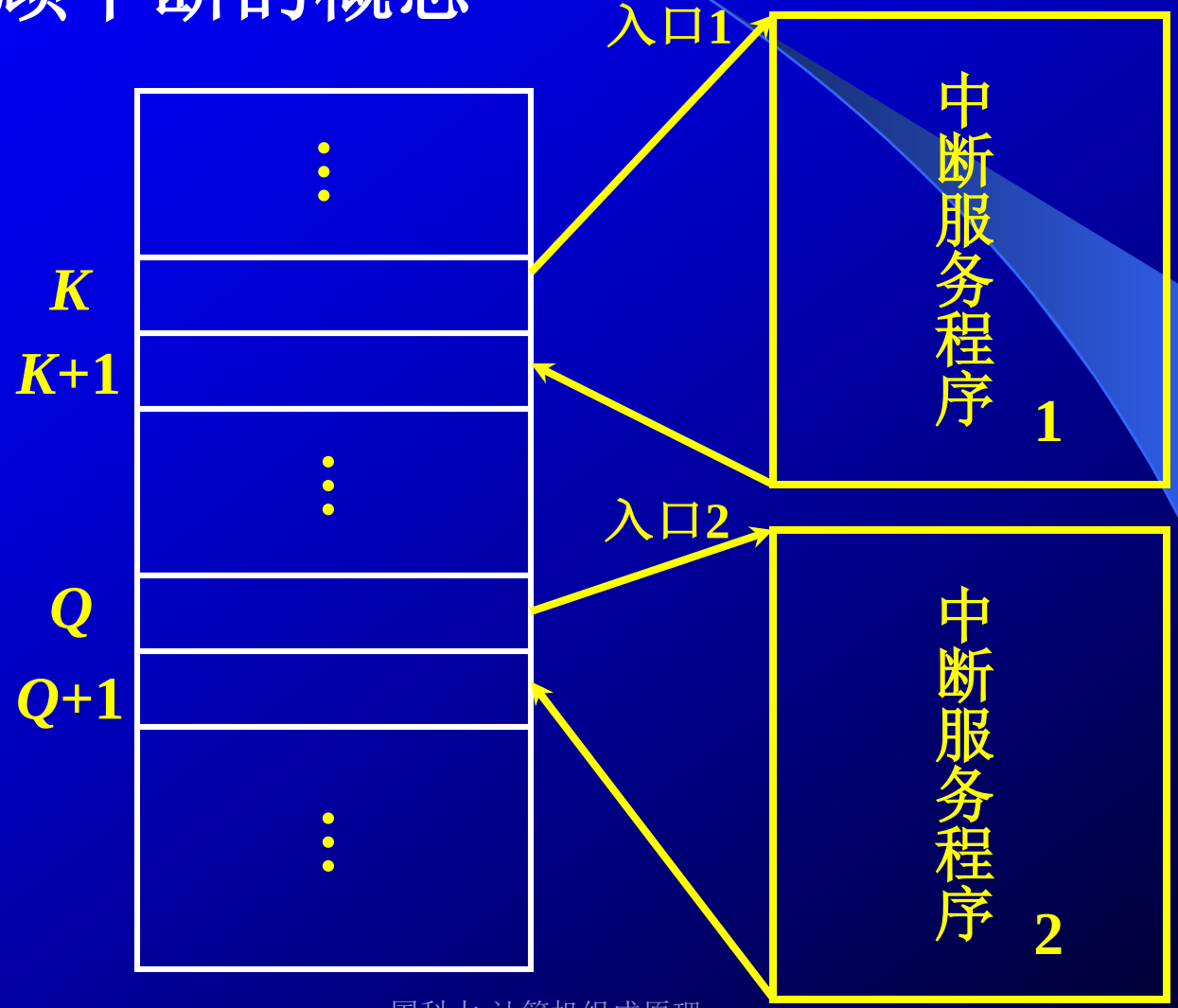
小结：

中断系统需解决的七个问题

1. 各中断源 如何 向 CPU 提出请求 ？
   - 中断请求标记寄存器 INTR
2. 各中断源 同时 提出 请求 怎么办 ？
   - 中断判优逻辑——硬件排队 vs. 程序查询
3. CPU 什么 条件、什么 时间、以什么 方式响应中断 ？
   - 允许中断触发器 EINT = 1 (开中断置1，关中断置0)
   - 指令执行周期结束时刻由CPU发查询信号
   - 通过中断隐指令依次完成：保护程序断点、寻找中断服务程序入口地址、硬件关中断 EINT=0
4. 如何 保护现场 ？
   - 中断隐指令保护程序断点（PC） + 中断服务程序保护CPU内部各寄存器内容
   - STORE vs. PUSH
5. 如何 寻找入口地址 ？
   - 硬件向量 vs. 软件查询
   - 改变PC，执行中断服务程序
6. 如何 恢复现场，如何 返回 ？
   - 恢复CPU寄存器内容 （LOAD vs. POP）
   - 中断返回指令 （恢复PC）
   - 在返回指令之前开中断
7. 处理中断的过程中又 出现新的中断 怎么办 ？——多重中断、提前开中断、中断屏蔽技术

11.2 I/O中断

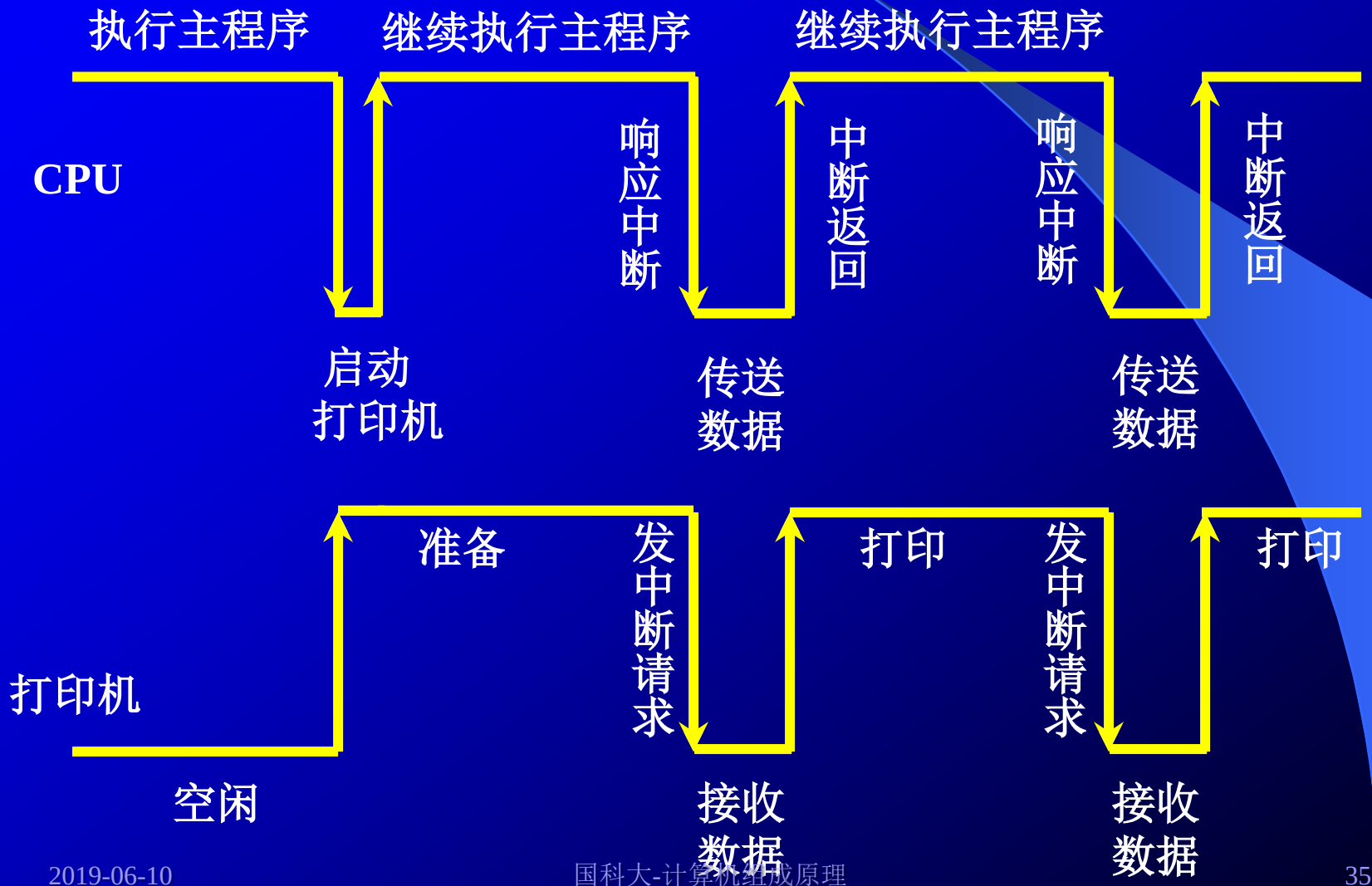
一、回顾中断的概念



二、I/O 中断的产生

11.2

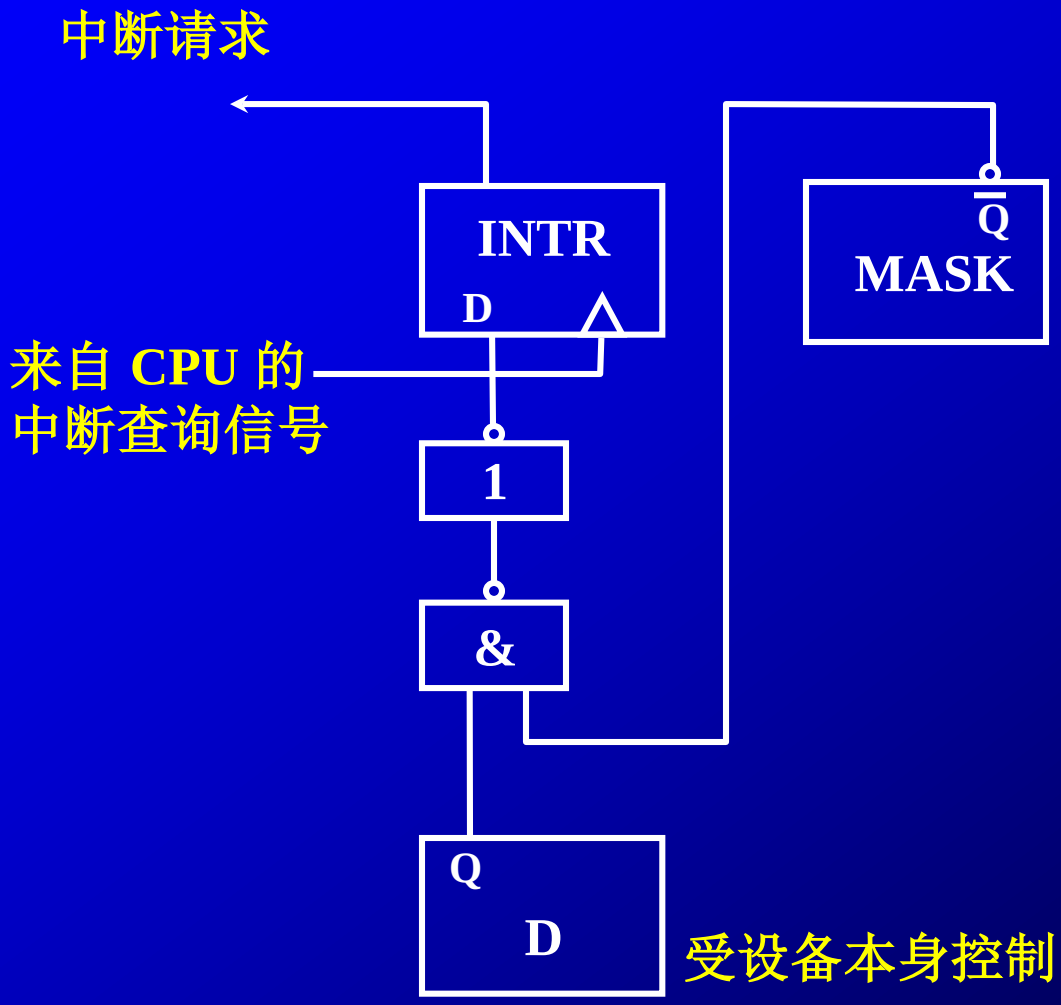
以打印机为例　CPU 与打印机并行工作



三、程序中断方式的接口电路

11.2

1. 配置中断请求触发器和中断屏蔽触发器



INTR
中断请求触发器

INTR = 1 有请求

MASK
中断屏蔽触发器

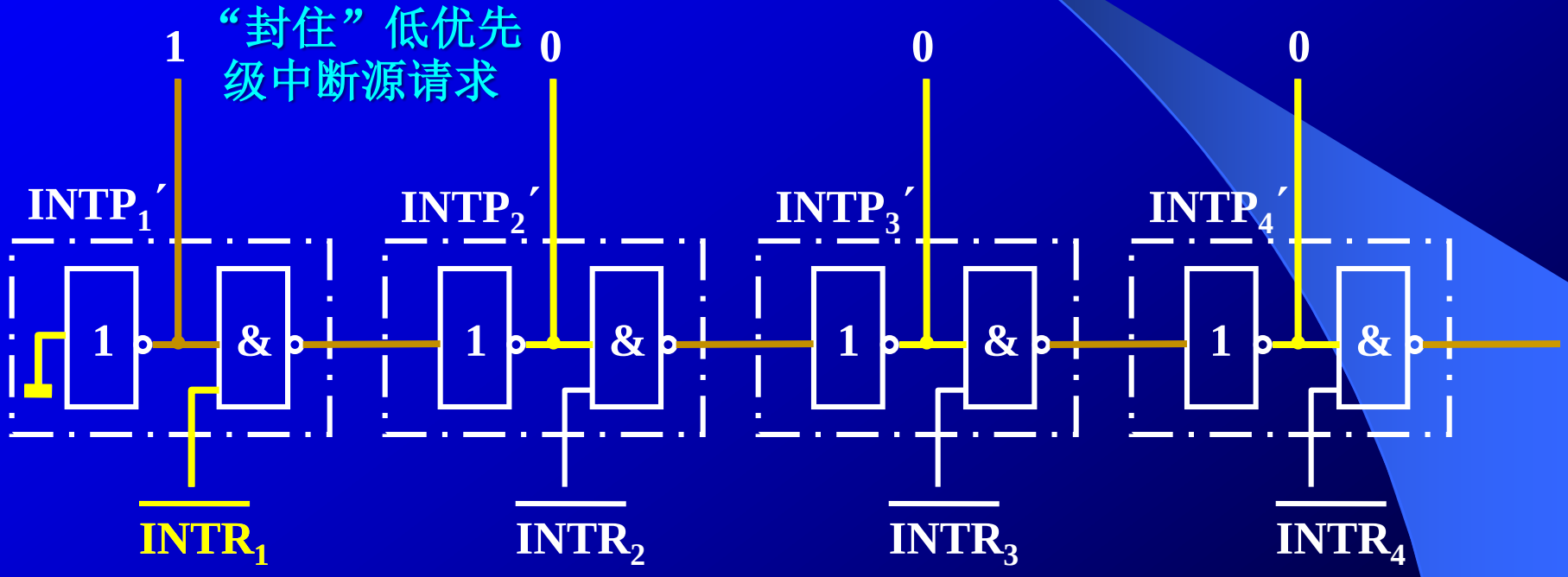
MASK = 1 被屏蔽

D 完成触发器

2. 排队器

11.2

排队 { 硬件 在 CPU 内或在接口电路中（链式排队器）
软件

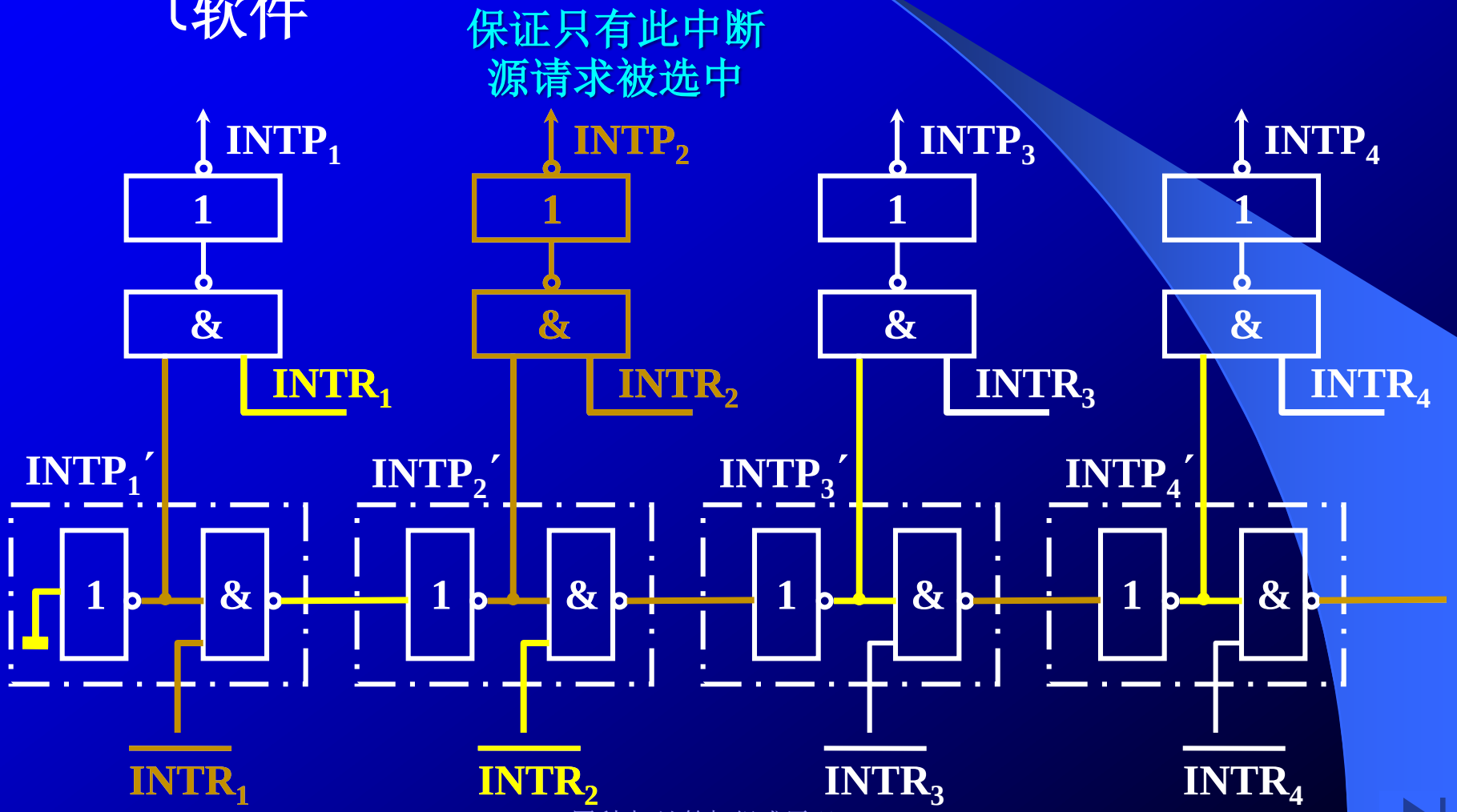


设备 1#、2#、3#、4# 优先级按 降序排列

$INTR_i = 1$ 有请求　　即 $\overline{INTR_i} = 0$

2. 排队器

排队 { 硬件 在 CPU 内或在接口电路中（链式排队器）
软件



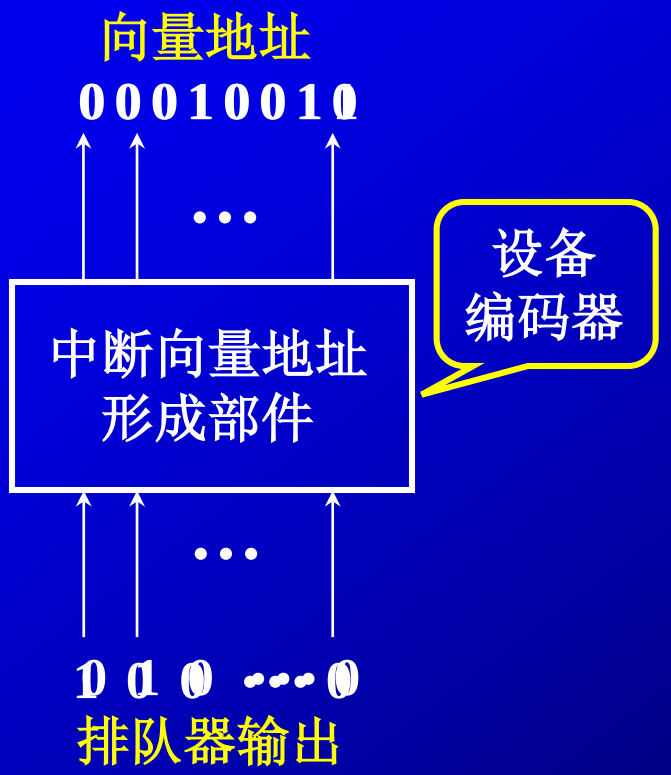
3. 中断向量地址形成部件

11.2

入口地址 { 由软件产生
硬件向量法

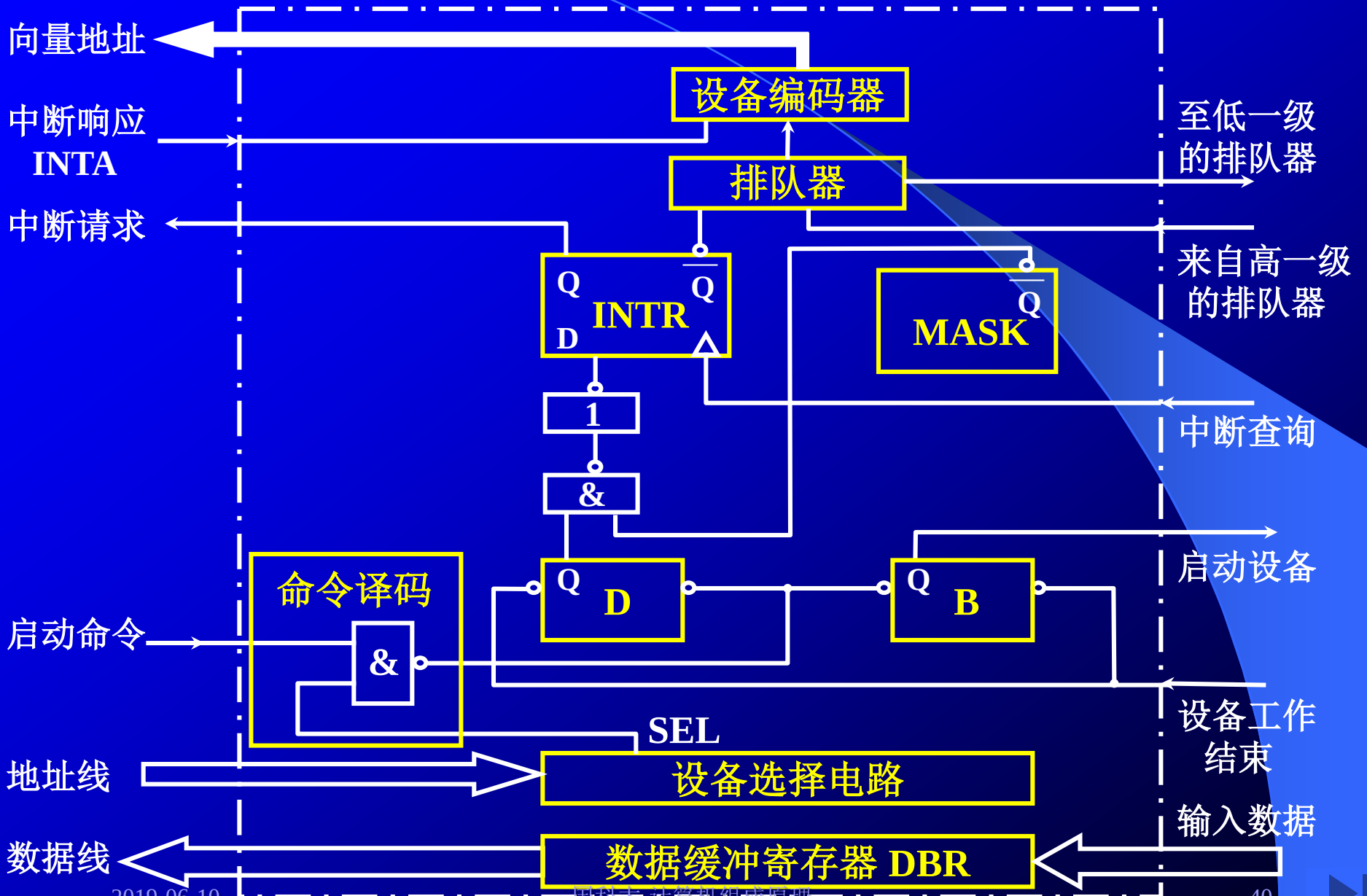
由 硬件 产生 向量地址

再由 向量地址 找到 入口地址



4. 程序中断方式接口电路的基本组成

11.2



四、I/O 中断处理过程

11.2

1. CPU 响应中断的条件和时间

(1) 条件

允许中断触发器 **EINT = 1**

用 开中断 指令将 EINT 置 “1”

用 关中断 指令将 EINT 置“0”或硬件自动复位

(2) 时间

I/O中断异步产生

当 **D = 1**（随机）且 **MASK = 0** 时

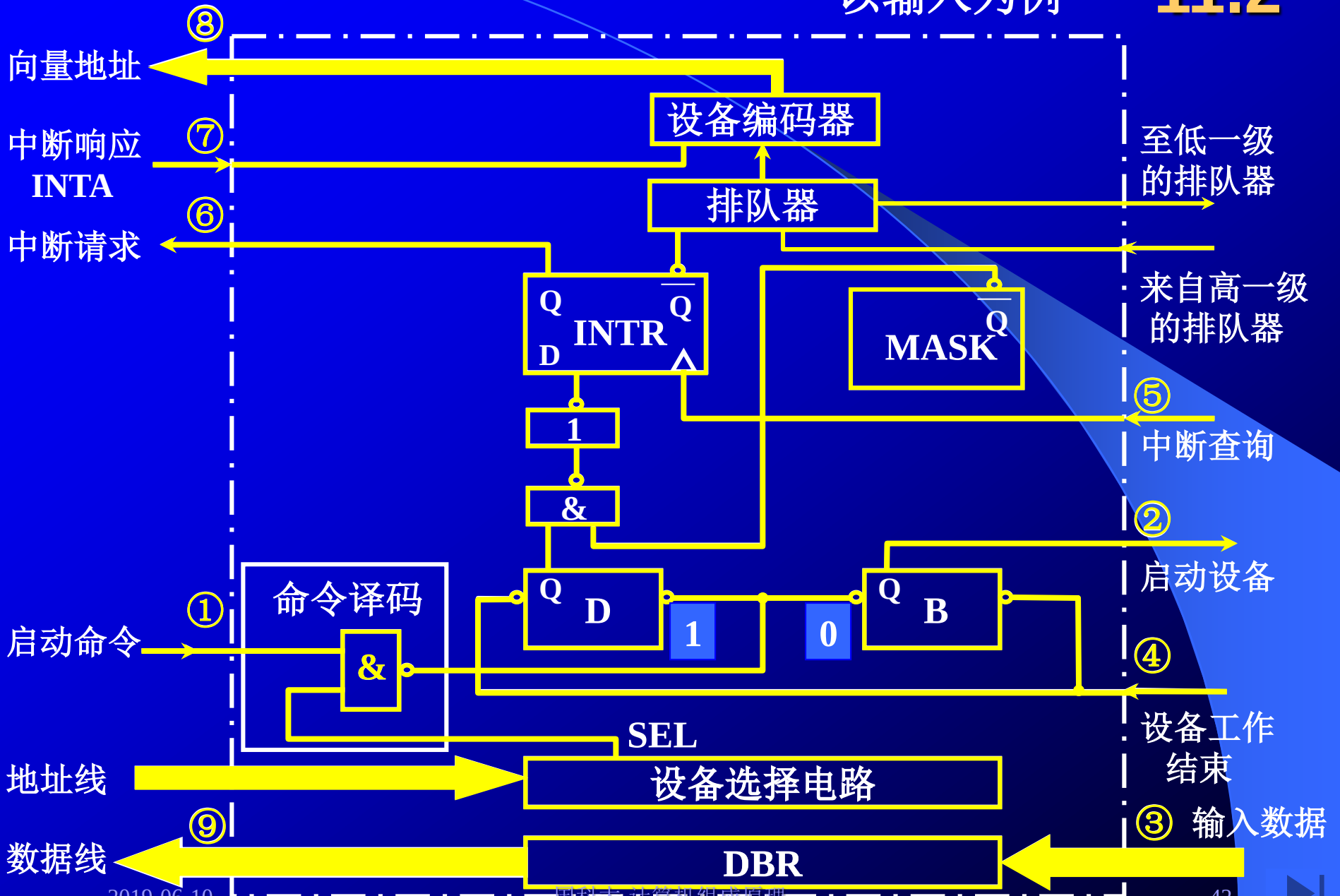
在每条指令执行阶段的结束前（统一时刻）

CPU 发 中断查询信号（控制INTR输出置“1”）

2. I/O 中断处理过程

以输入为例

11.2



五、中断服务程序流程

11.2

1. 中断服务程序的流程

(1) 保护现场

- 程序断点的保护　　中断隐指令完成
- 寄存器内容的保护　进栈指令

(2) 中断服务

对不同的 I/O 设备具有不同内容的设备服务

(3) 恢复现场　　出栈指令

(4) 中断返回　　中断返回指令

2. 单重中断和多重中断

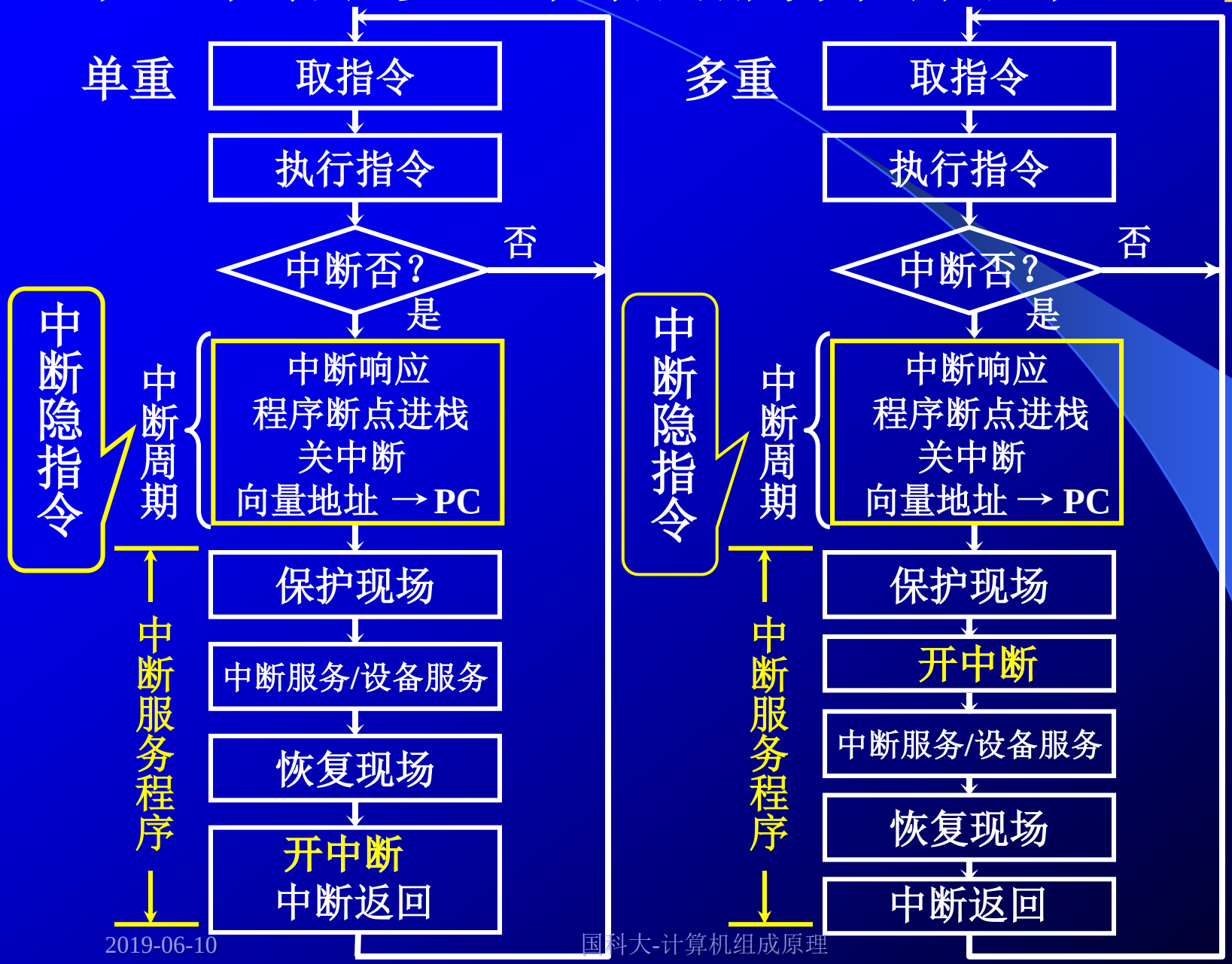
单重 中断　不允许中断 现行的 中断服务程序

多重 中断　允许级别更高 的中断源

中断 现行的 中断服务程序

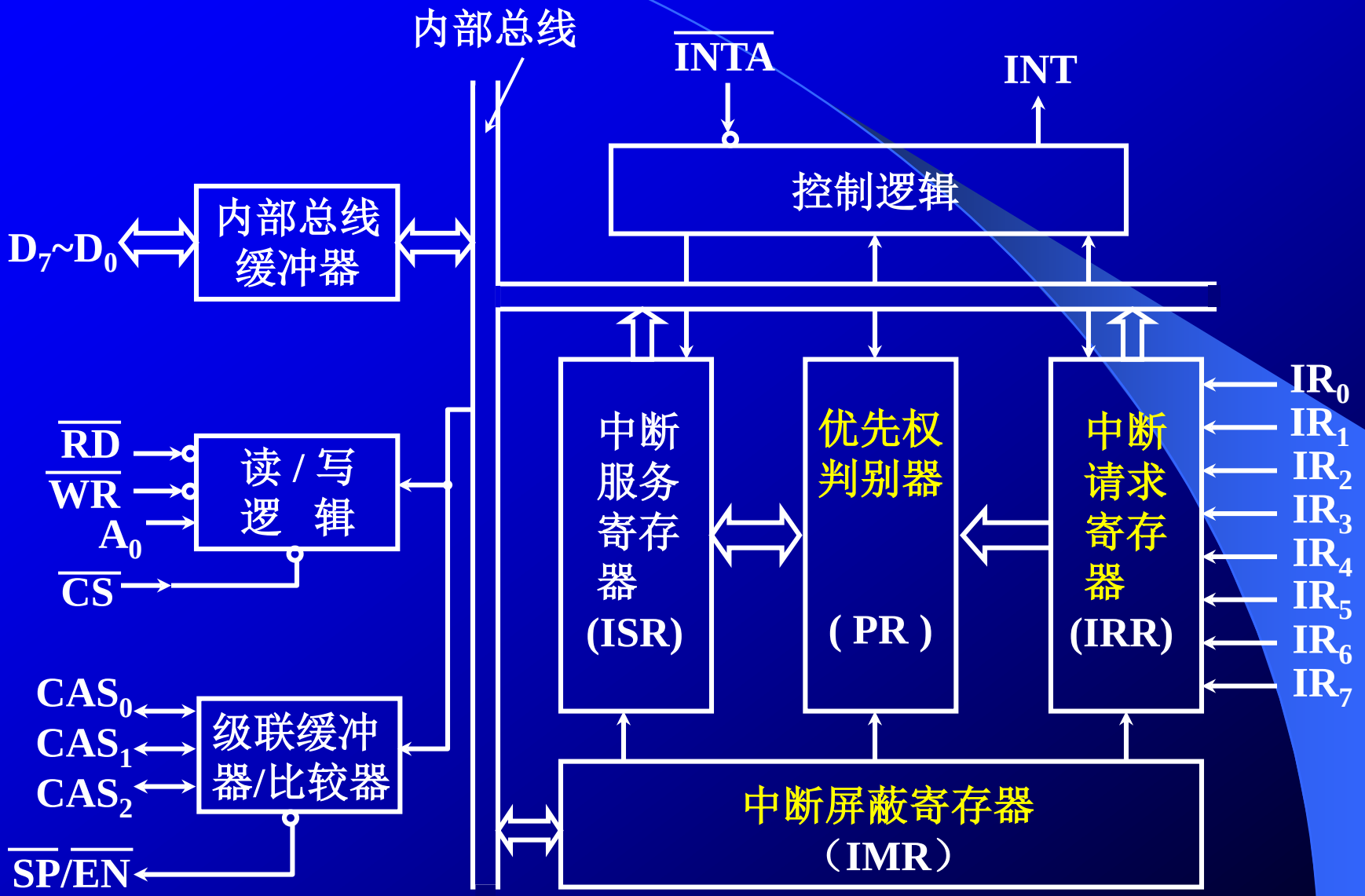
3. 单重中断和多重中断的服务程序流程

11.2

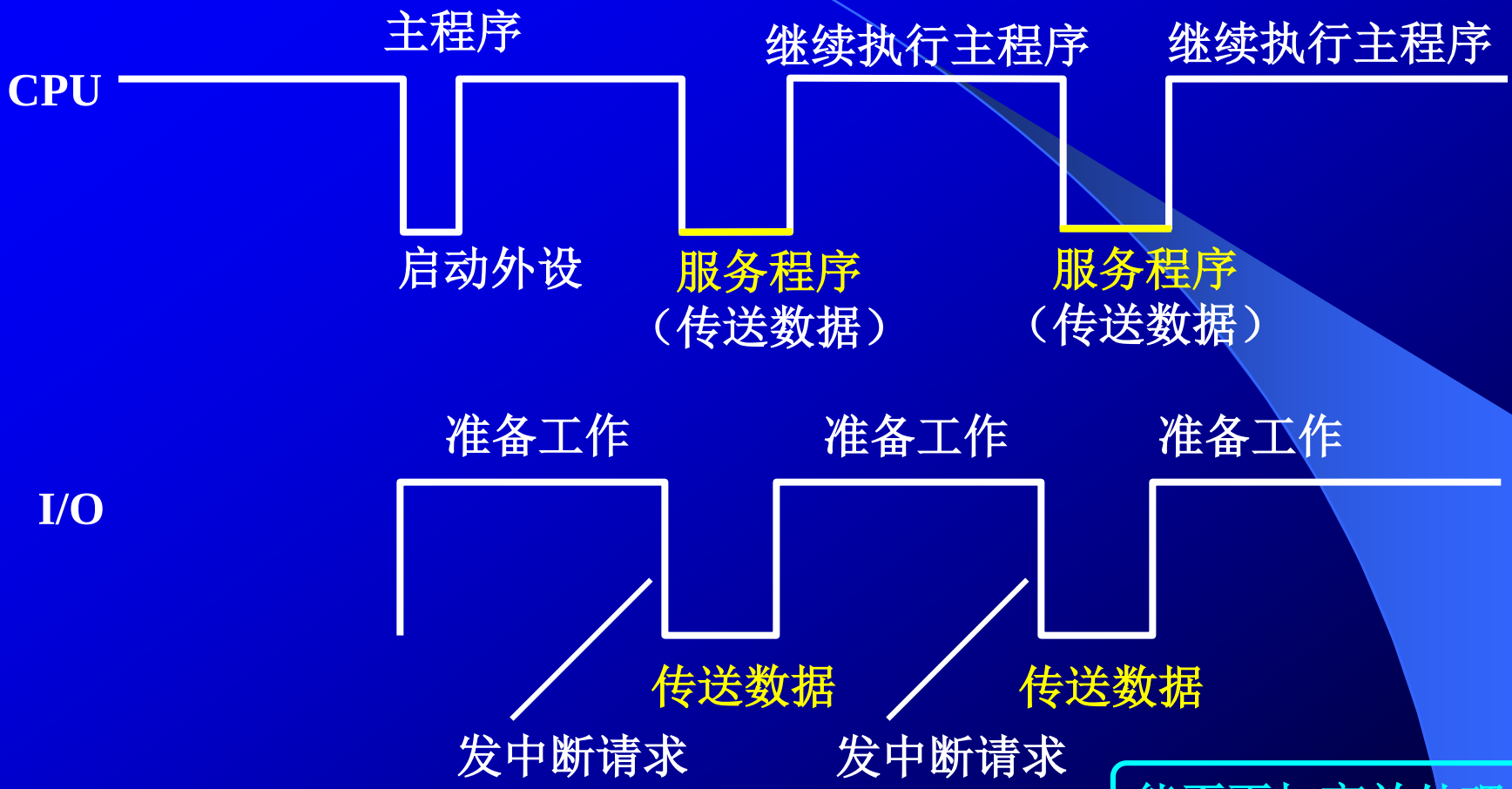


程序中断接口芯片 8259A 的内部结构

11.2



主程序和服务程序抢占 CPU 示意图 11.2



能否更加高效处理？

宏观 上 CPU 和 I/O 并行 工作

微观 上 CPU 中断现行程序 为 I/O 服务

作业

- 习题：8.24 ~ 8.28
- 本次作业提交截止时间
  - 请于6月17日上课前提交

Q & A ?


中国科学院大学
University of Chinese Academy of Sciences
ICT
中科院计算所
INSTITUTE OF COMPUTING TECHNOLOGY,CAS