第8章 CPU 的结构和功能

> 课件来源：唐朔飞《计算机组成原理》配套课件 cod 版 Slide 413–465（唯一来源，无其他版本混用）。本章是全书"CPU 篇"的总纲，把运算器、控制器、寄存器、指令周期、流水线、中断系统串成一条线，是期末**概念辨析 + 流程/数据流默写 + 流水线计算 + 中断屏蔽字设计**四类题的高发区，地位★★★。

0. 本章考点清单

考点	常见题型	重要度
流水线性能三指标（吞吐率 Tp、加速比 Sp、效率 E）及计算	计算	★★★
中断屏蔽字、响应优先级 vs 处理优先级、多重中断	综合设计/简答	★★★
指令周期四种工作周期（取指/间址/执行/中断）及流程	填空/简答/流程	★★★
取指/间址/中断周期数据流（微操作序列）	简答/默写	★★★
流水线三类相关（结构/数据/控制）及解决办法	选择/简答	★★★
CPU 组成（运算器/控制器/寄存器/中断系统）与功能	选择/填空	★★
寄存器分类（用户可见 vs 控制状态；MAR/MDR/IR 不可见、PC 可见）	选择/填空	★★
中断隐指令三件事、中断响应条件与时间	简答	★★
中断服务程序入口寻找（硬件向量法/软件查询法）	简答	★★
中断判优（硬件排队器/软件查询）、中断请求标记 INTR	简答	★★
保护现场/恢复现场、断点保护（进栈 vs 0 地址）	简答	★★
流水线多发技术（超标量/超流水线/超长指令字）对比	选择/简答	★★
指令流水原理（串行→二级→六级）、指令预取	简答	★★
提高机器速度途径、并行/并发、并行性等级	选择/填空	★
流水线结构（指令流水 7 段、运算流水 3 段）	填空	★
引起中断的因素、中断系统需解决的 7 个问题	简答	★

1. CPU 的功能与结构框图（8.1）

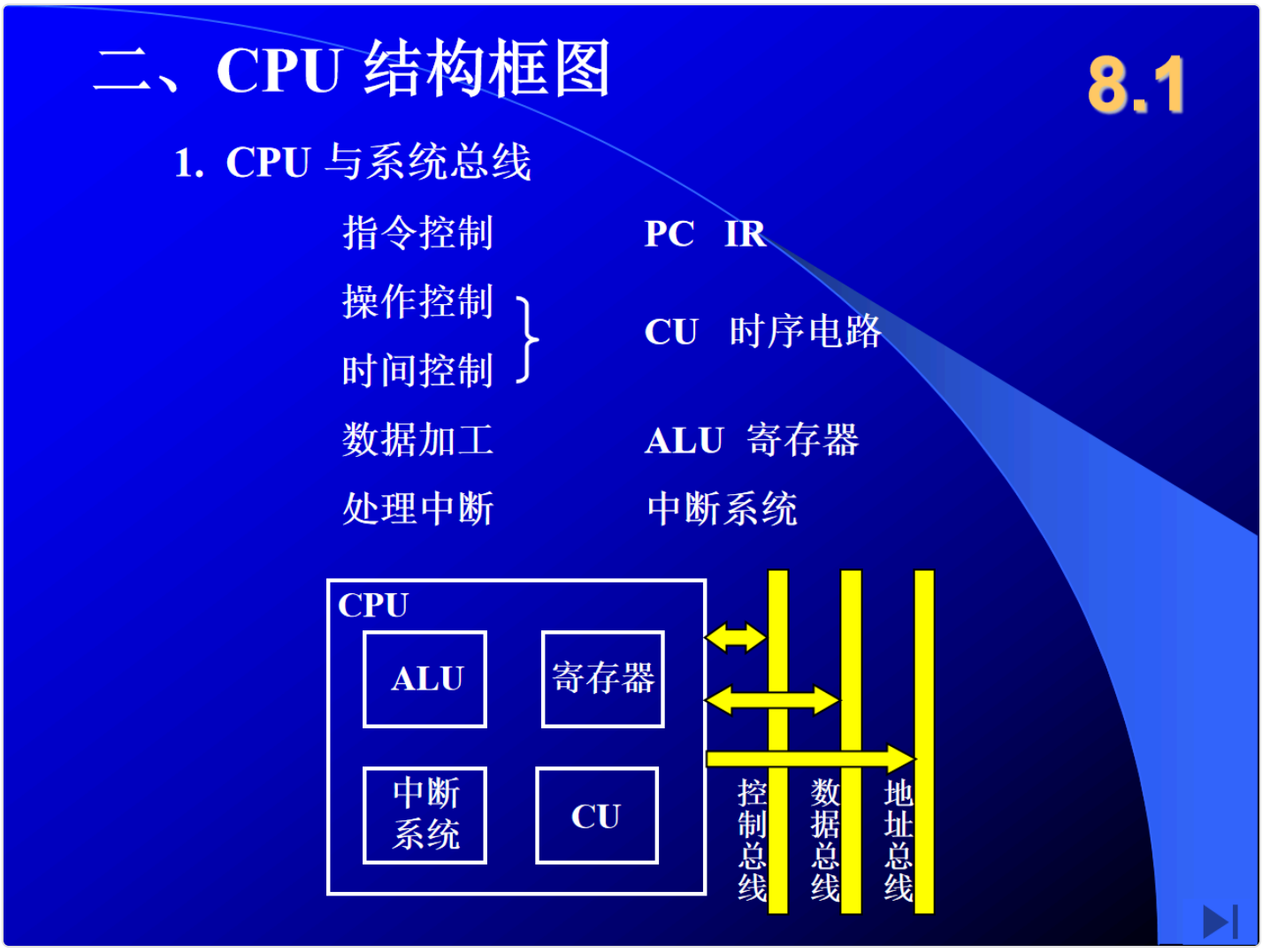
概念精讲：CPU = 运算器 + 控制器（+ 寄存器 + 中断系统）。控制器负责"指挥"，运算器负责"加工"。

控制器的功能（课件 p.414）：取指令、分析指令、执行指令并发出各种操作命令、控制程序输入及结果的输出、总线管理、处理异常情况和特殊请求。归纳为五大功能：**指令控制、操作控制、时间控制、数据加工、处理中断**（其中"数据加工"实际由运算器完成，控制器协调）。 **运算器的功能**（课件 p.414）：实现算术运算和逻辑运算。

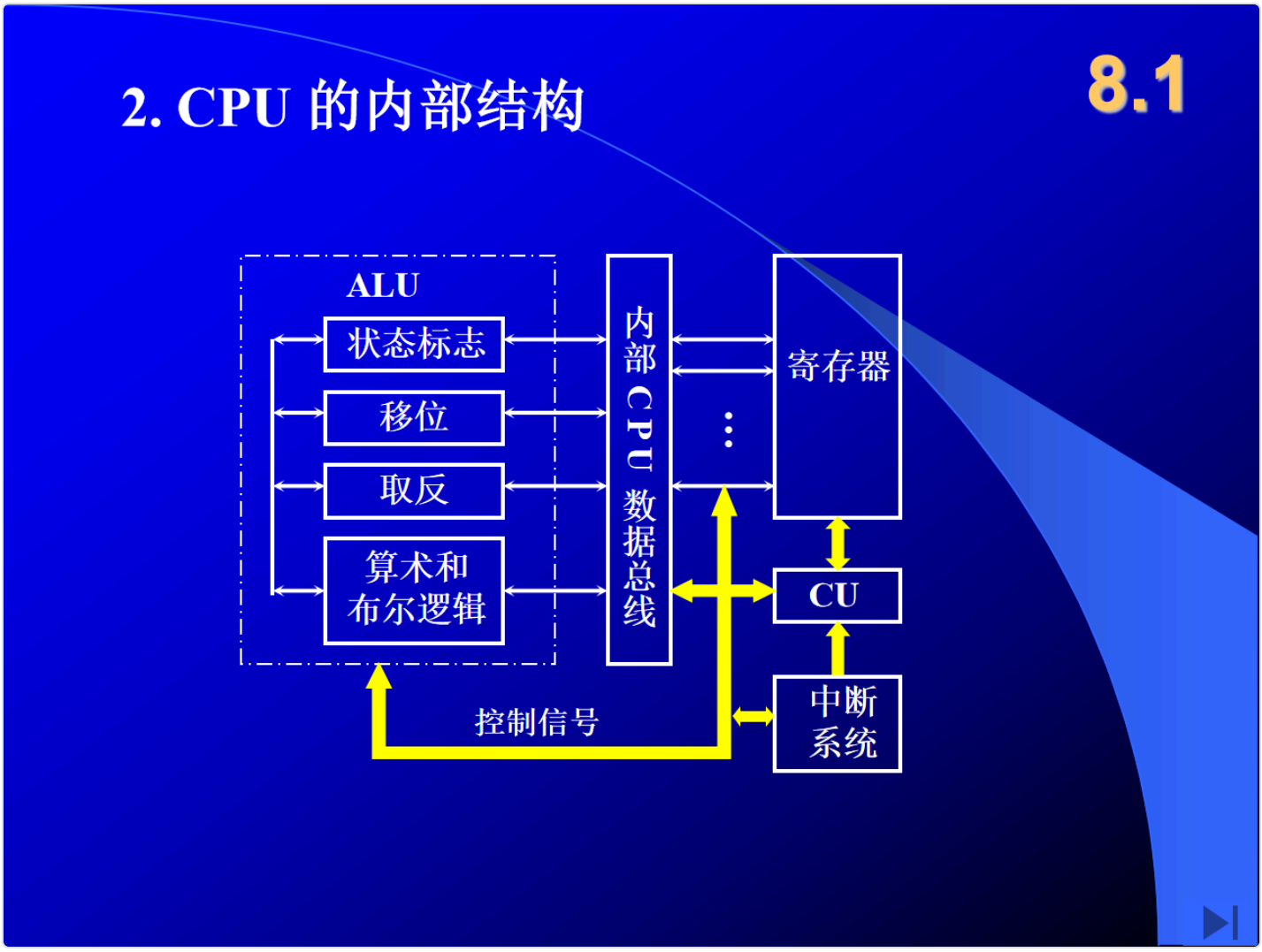
功能→部件对应（课件 p.415）：

功能	对应部件
指令控制	PC、IR
操作控制 + 时间控制	CU、时序电路
数据加工	ALU、寄存器
处理中断	中断系统

CPU 与系统总线（课件 p.415）：CPU 内含 ALU、寄存器、中断系统、CU 四大块；对外接**控制总线（双向）、数据总线（双向）、地址总线（单向，CPU→外）**。



CPU 内部结构（课件 p.416）：ALU 内部含**算术和布尔逻辑、取反、移位、状态标志**四个功能块；各部件通过**内部 CPU 数据总线**互联；CU 发出**控制信号**控制 ALU 与总线；中断系统与 CU、寄存器相连。



要点：地址总线单向、数据/控制总线双向是常考填空；ALU 的"状态标志"块是条件码的来源。 **易错警示**：不要把"数据加工"归给控制器单独完成——它是运算器功能、控制器协调；CU 与时序电路合起来才实现操作控制+时间控制。

2. CPU 的寄存器（8.1）

概念精讲：寄存器按"程序员能否直接访问"分两大类。

1）用户可见寄存器（课件 p.417）——汇编/机器语言可直接引用：

类型	作用
通用寄存器	存放操作数；可作某种寻址方式所需的专用寄存器
数据寄存器	存放操作数（满足各种数据类型）；两个寄存器拼接存放双倍字长数据
地址寄存器	存放地址，位数应满足最大地址范围；用于特殊寻址方式（段基值、栈指针）
条件码寄存器	存放条件码，作程序分支依据（正、负、零、溢出、进位等）

2）控制和状态寄存器（课件 p.418）： - 控制寄存器：控制 CPU 操作，如 **PC**。 - 状态寄存器：存放条件码；**PSW 寄存器**存放程序状态字。 - **可见性结论（必背）**：**MAR、MDR、IR 用户不可见；PC 用户可见**（课件 p.418）。

3）举例（课件 p.418）：Z8000、8086、MC68000 三种机器的寄存器组织（课件仅列机型，具体寄存器组见各自指令系统章）。

要点：MAR（存储器地址寄存器）、MDR（存储器数据寄存器）是访存接口，IR（指令寄存器）存当前指令，三者对程序员透明；PC 虽属控制寄存器但用户可见（可被转移/调用指令修改、可读）。 **易错警示**：① "PC 是控制寄存器所以不可见"——错，PC 用户可见；② 条件码寄存器既出现在"用户可见"（作分支依据）又与状态寄存器/PSW 相关，答题时按课件两处表述分别对应，不要混为一谈。

3. 控制单元 CU、中断系统与 ALU 的定位（8.1）

CU（课件 p.419）：产生全部指令的**微操作命令序列**。两种设计路线：

设计方法	逻辑类型
组合逻辑设计	硬连线逻辑
微程序设计	存储逻辑

（CU 详细设计参见课件第 4 篇；中断系统参见本章 8.4 节；ALU 参见第 6 章。课件 p.419）

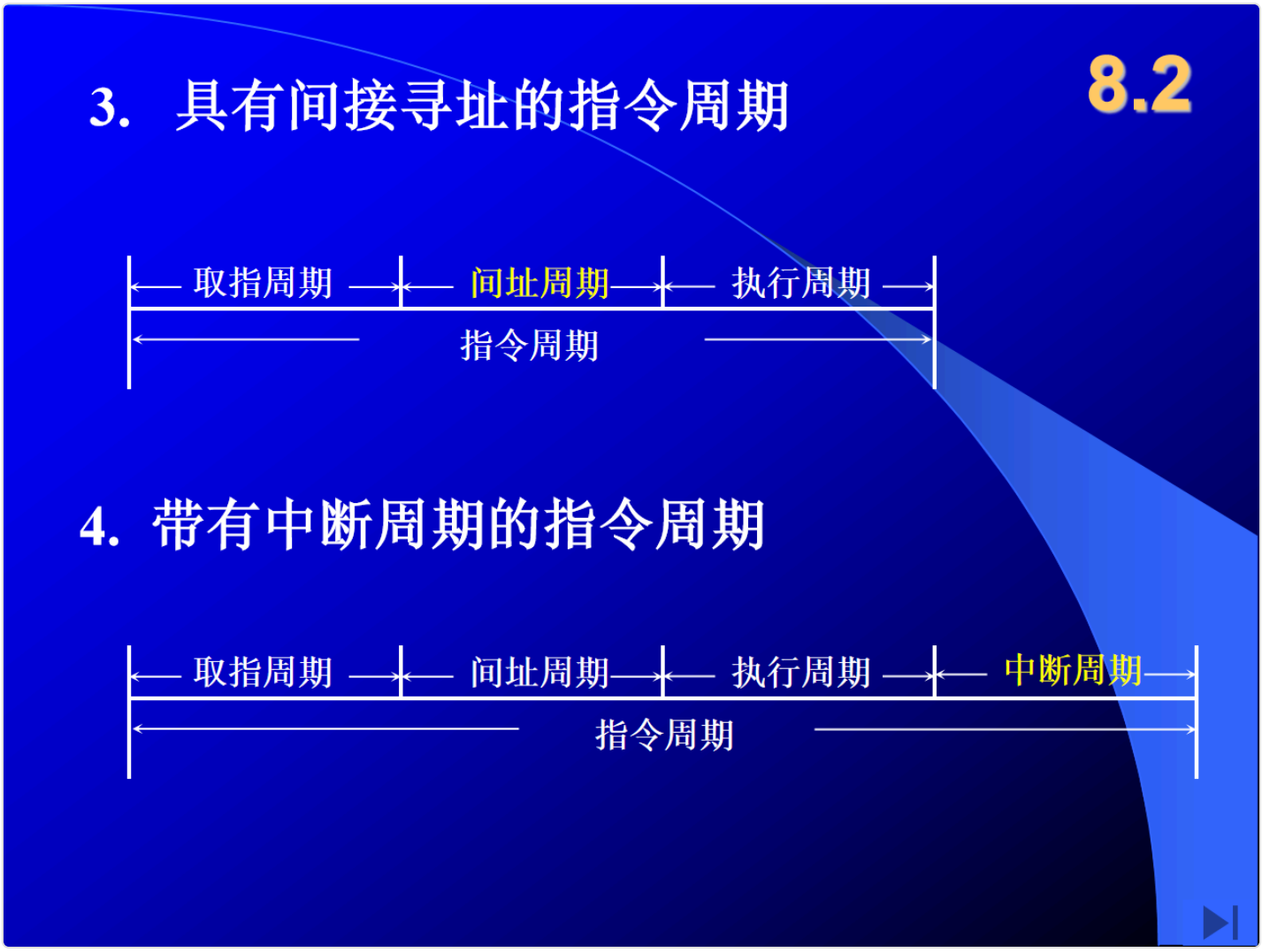
要点：硬连线=组合逻辑=速度快、难修改；微程序=存储逻辑=规整、易扩展。此对应关系是选择题高频点。 **易错警示**：课件 p.419 提取文本中"参见 8.4 节/第 6 章"顺序错乱，以整页图为准：**CU 设计→第 4 篇；中断系统→8.4 节；ALU→第 6 章**。

4. 指令周期的基本概念（8.2）

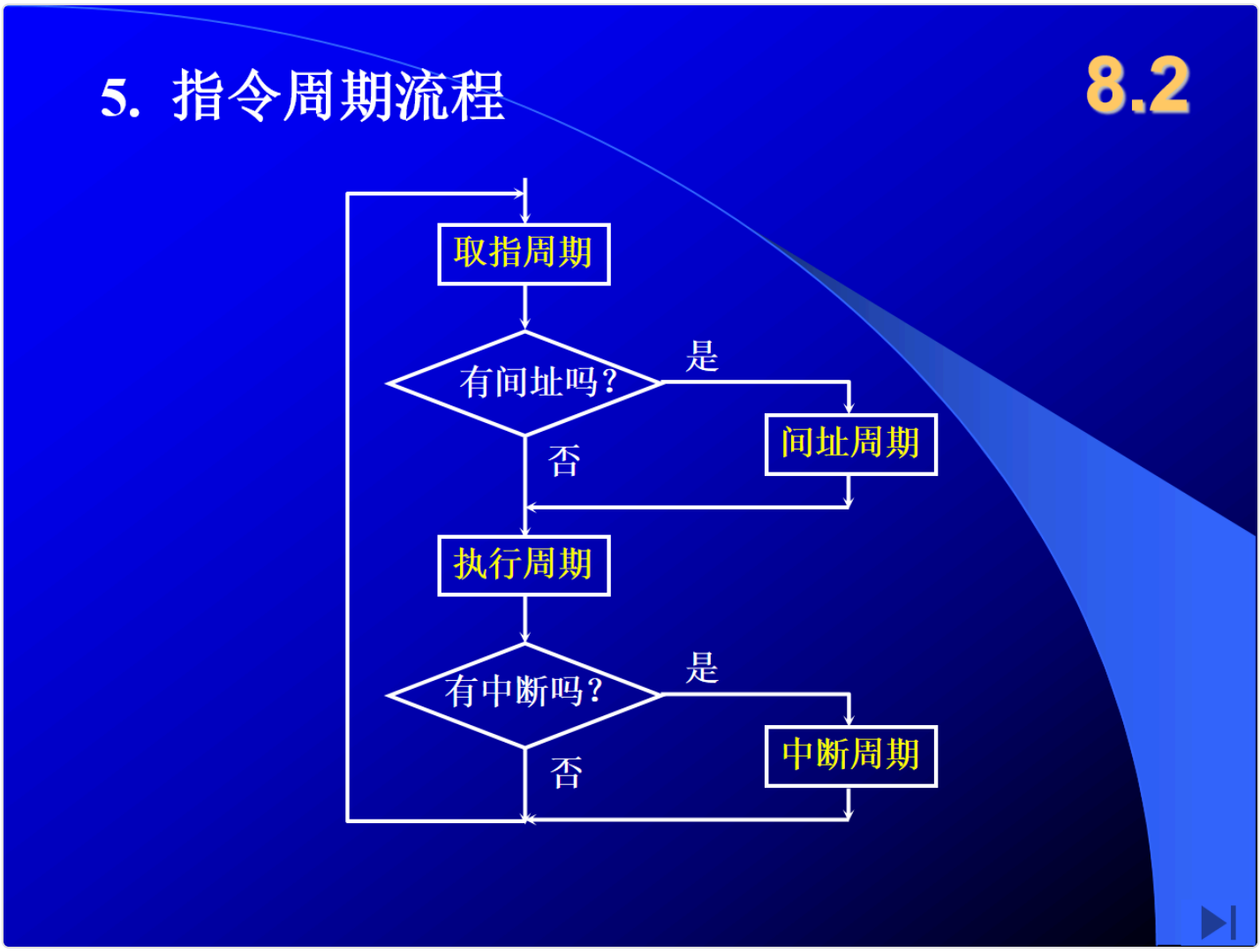
概念精讲：**指令周期 = 取出并执行一条指令所需的全部时间**（课件 p.420）。一条指令 = 取指、分析（取指阶段/取指周期）+ 执行（执行阶段/执行周期）。

每条指令的指令周期不同（课件 p.421）： - **NOP**：指令周期 = 取指周期（无执行周期）。 - **ADD mem**：取指周期 + 执行周期。 - **MUL mem**：取指周期 + 更长的执行周期。

四种工作周期组合（课件 p.422）： - 基本：取指 + 执行。 - 含间接寻址：取指 + **间址** + 执行。 - 含中断：取指 + 间址 + 执行 + **中断**。



指令周期流程（课件 p.423）：取指周期 → 判断"有间址吗？"是→间址周期→执行周期，否→直接执行周期 → 执行周期后判断"有中断吗？"是→中断周期→回取指，否→直接回取指。



CPU 工作周期的标志（课件 p.424）：CPU 访存有四种性质，对应四个工作周期，用四个触发器标志：

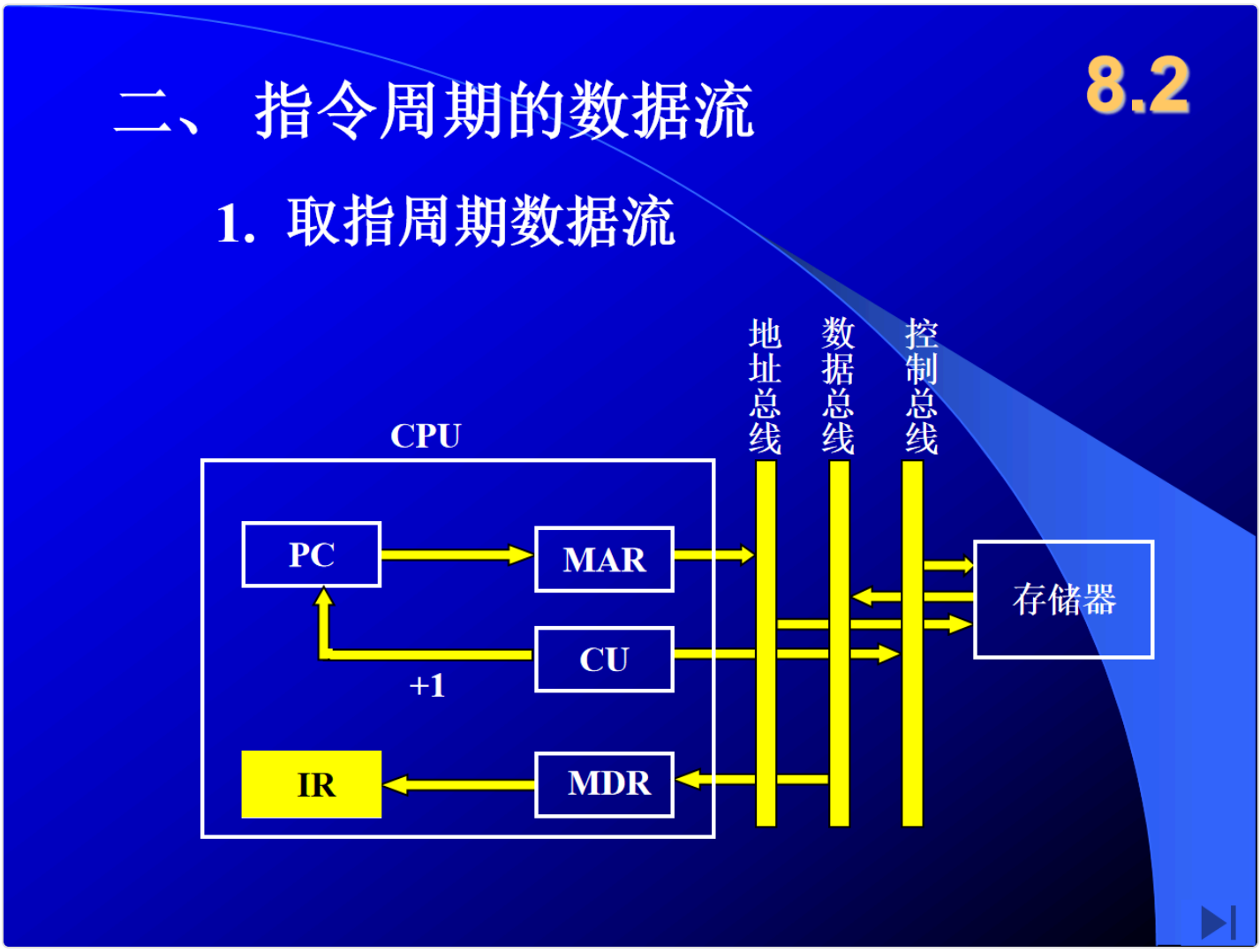
访存性质	工作周期	标志触发器
取指令	取指周期	FE
取地址	间址周期	IND
取操作数	执行周期	EX
存程序断点	中断周期	INT

每个触发器在对应周期被置 1（1→FE、1→IND、1→EX、1→INT），由 CLK 同步。

要点：取指周期和執行周期是任何指令都有的（NOP 除外执行）；间址周期仅间接寻址指令有；中断周期仅响应中断时插入。**指令周期 ⊃ 机器周期（工作周期）⊃ 时钟周期（节拍）**（补充常识：课件用"工作周期"即机器周期，一个工作周期含若干时钟周期）。**易错警示**：① 中断周期不是每条指令都有，只在执行周期结束且允许中断、有请求时插入；② 间址周期取的是"操作数的地址"（有效地址），不是操作数本身；③ NOP 无执行周期（课件 p.421 图明确）。

5. 指令周期的数据流（8.2）

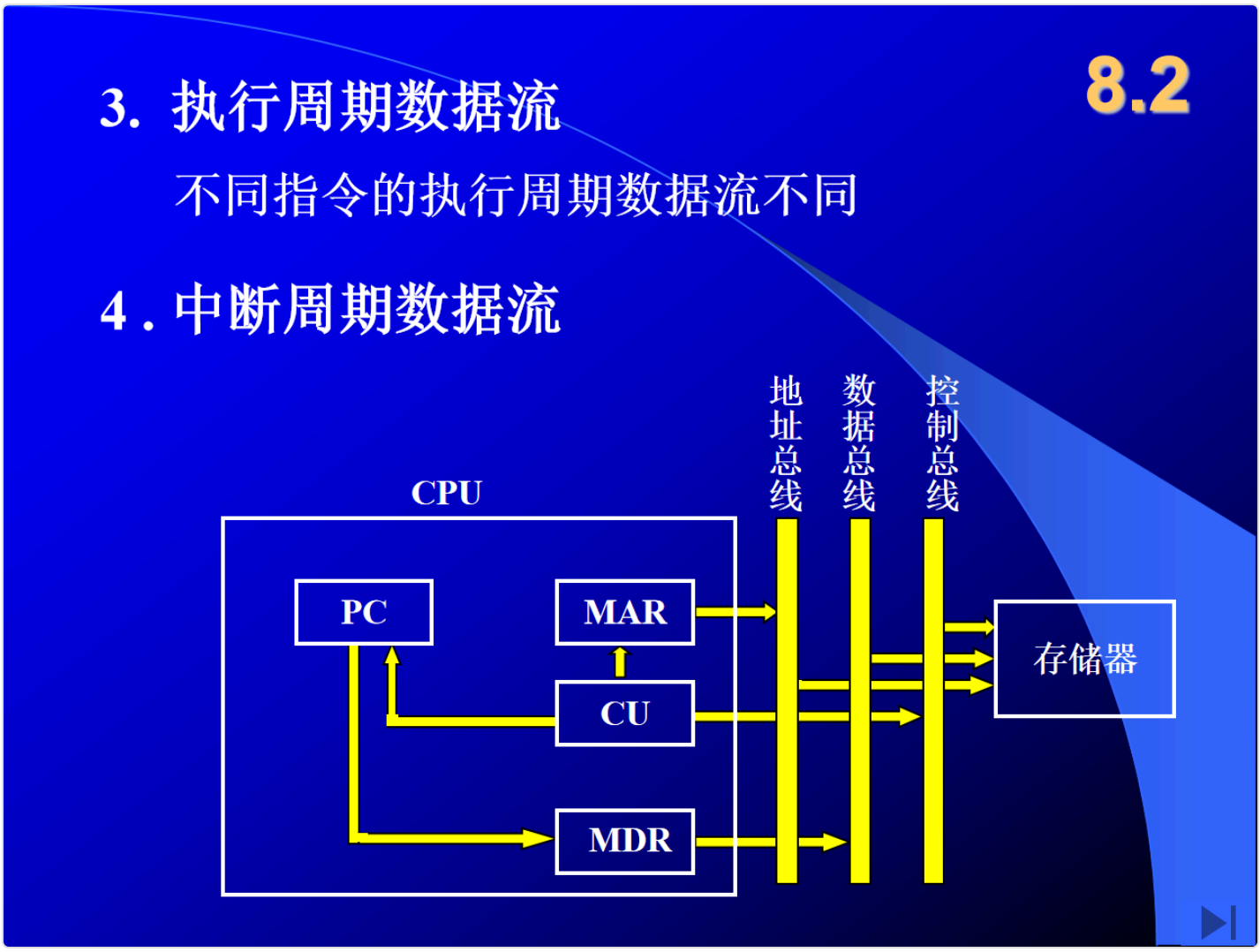
1）取指周期数据流（课件 p.425）： 1. (PC)→MAR，送地址总线； 2. CU 经控制总线发**读**命令； 3. 存储器将该单元内容经数据总线→MDR； 4. (MDR)→IR； 5. CU 使 (PC)+1→PC（指向下一条指令）。



2）间址周期数据流（课件 p.426）：把 IR 中的形式地址（地址码）送 MAR（Ad(IR)→MAR），CU 发读命令，存储器内容经数据总线→MDR，再 (MDR)→Ad(IR)（把有效地址写回 IR 地址字段）。本质是"取地址"。

3）执行周期数据流（课件 p.427）：**不同指令的执行周期数据流不同**（课件只给结论，不统一画）。

4）中断周期数据流（课件 p.427）： 1. CU 形成断点存放地址→MAR（0 号地址或栈指针）； 2. (PC)→MDR（断点送 MDR）； 3. CU 发**写**命令，(MDR)→存储器（保存断点）； 4. 向量地址（或识别程序入口 M）→PC（转向服务程序）。



要点：取指=读指令、间址=读地址、执行=读/写操作数、中断=写断点；四个周期对存储器的读/写性质是填空高频。 **易错警示**：中断周期是**写**存储器（存断点），而取指/间址是**读**；不要记反。取指周期 PC+1 发生在取指阶段而非执行阶段。

6. 提高机器速度与系统的并行性（8.3）

提高机器速度三条途径 + 整机（课件 p.428）：

方向	手段
提高访存速度	高速芯片、Cache、多体并行
提高 I/O 与主机传送速度	中断、DMA、通道、I/O 处理机、多总线
提高运算器速度	高速芯片、改进算法、快速进位链
提高整机处理能力	高速器件、改进系统结构开发系统的并行性

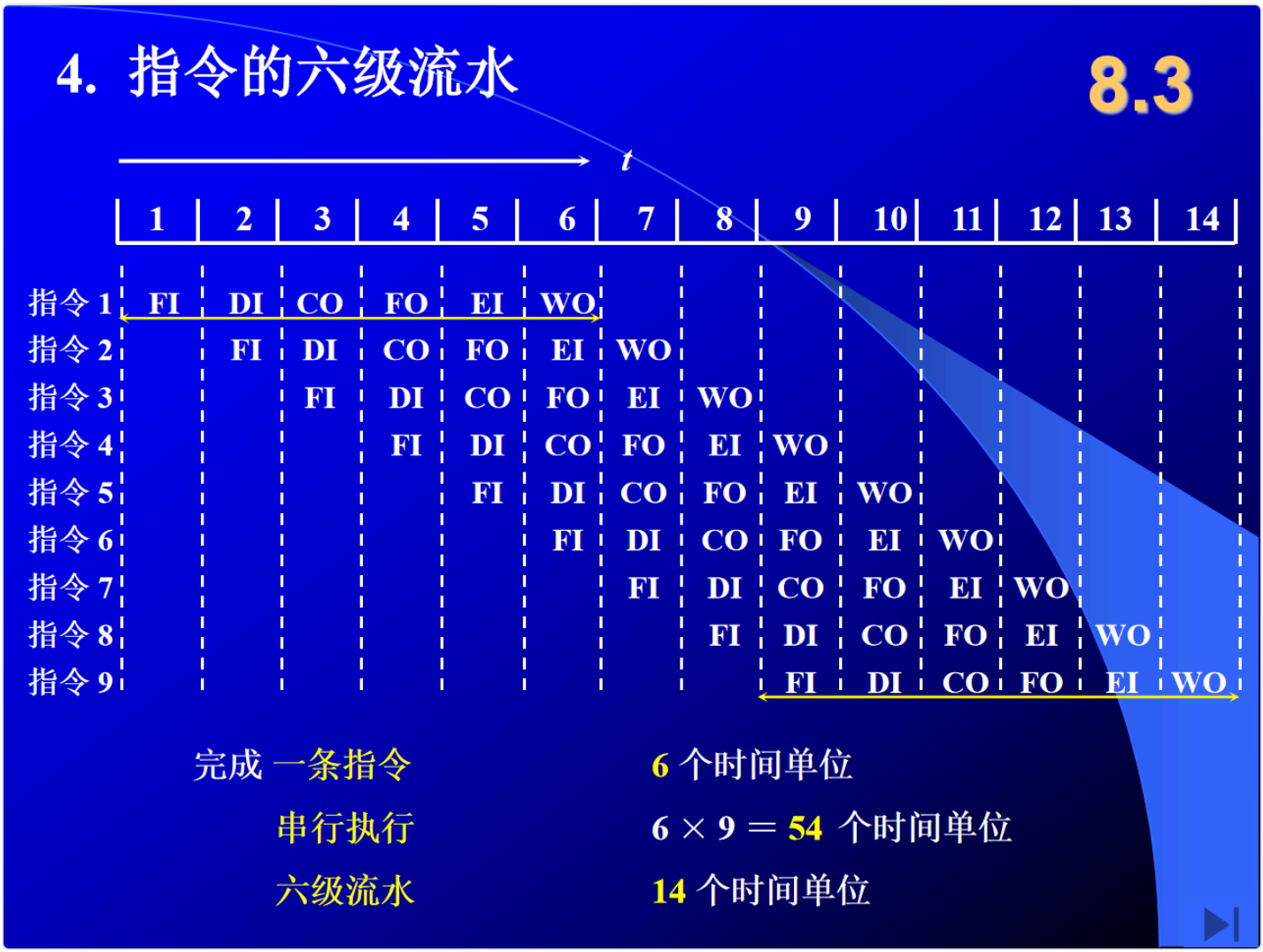
并行 vs 并发（课件 p.429）： - **并行**：两个或两个以上事件在**同一时刻**发生。 - **并发**：两个或两个以上事件在**同一时间段**发生（时间上互相重叠）。

并行性等级（课件 p.429）： - **指令级**（指令之间、指令内部）：**细粒度**，**硬件**实现。 - **过程级**（程序、进程）：**粗粒度**，**软件**实现。

要点：流水线属于指令级并行（细粒度、硬件）。 **易错警示**：并行=同一时刻、并发=同一时间段，二者不可互换；指令级=硬件/细粒度，过程级=软件/粗粒度，对应不要反。

7. 指令流水原理（8.3）

1）串行执行（课件 p.430）：取指令部件与执行指令部件交替工作，**总有一个部件空闲**。 **2）二级流水**（课件 p.430）：引入**指令预取**，取指与执行在时间上重叠。若取指和执行阶段时间上**完全重叠**，则**指令周期减半、速度提高 1 倍**。 **3）影响流水效率加倍的因素**（课件 p.431）： - (1) **执行时间 > 取指时间**（两段不等长，无法完全重叠）； - (2) **条件转移指令**：必须等上条指令执行结束才能确定下条指令地址，造成时间损失。解决办法：**指令部件缓冲区 + 猜测法**。 **4）六级流水**（课件 p.432）：六段缩写 **FI、DI、CO、FO、EI、WO**（课件仅给缩写；按教材惯例依次为取指、译码、计算操作数地址、取操作数、执行、写回，属补充常识）。9 条指令： - 完成一条指令 = **6** 个时间单位； - 串行执行 = 6×9 = **54** 个时间单位； - 六级流水 = **14** 个时间单位（= m + n − 1 = 6 + 9 − 1）。



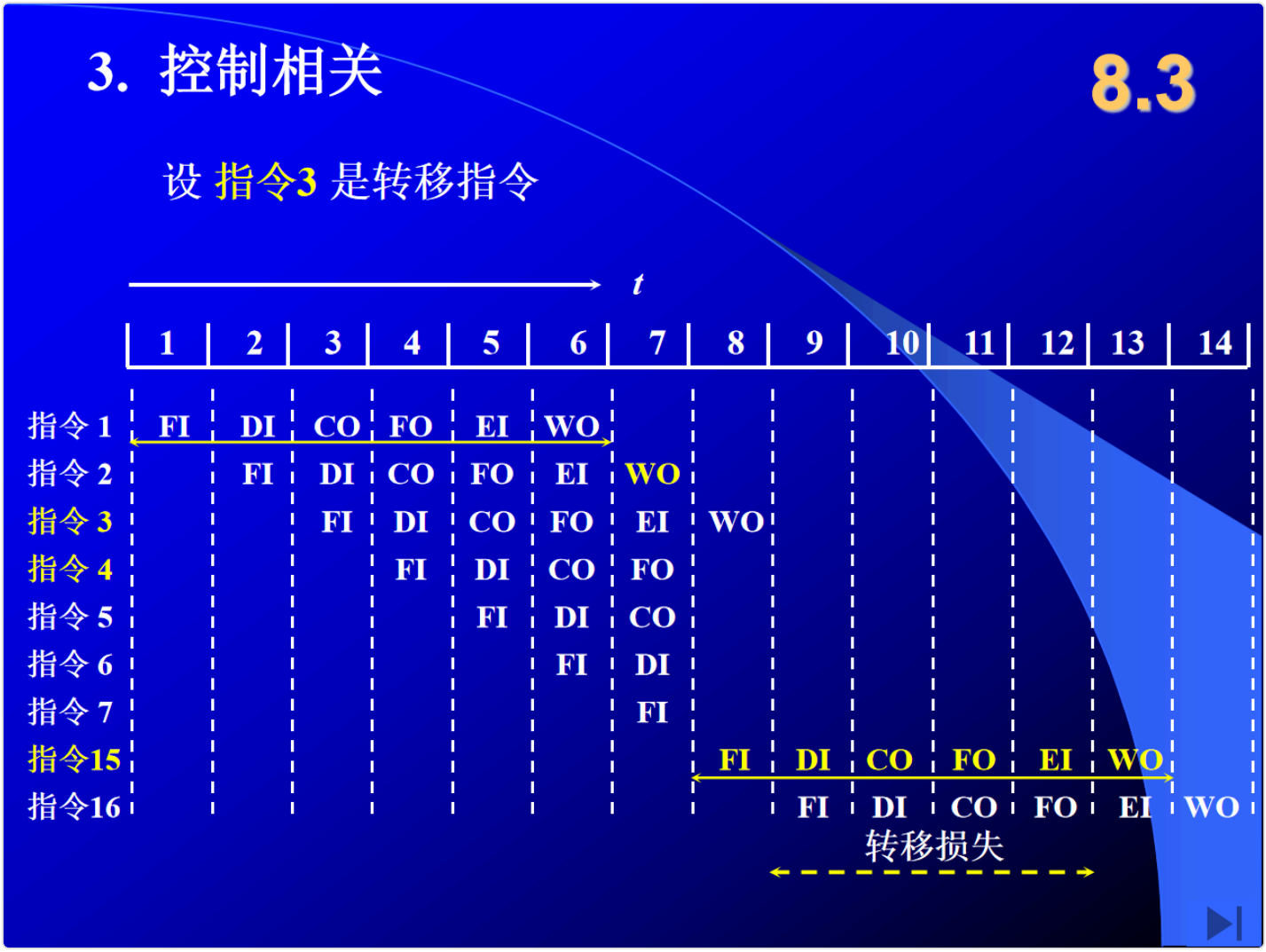
要点：m 段流水完成 n 条指令的总时间 = (m + n − 1)·Δt（各段等长 Δt 时），这是所有流水线计算的根公式。 **易错警示**：二级流水"速度提高 1 倍"= 变为 2 倍，不是提高 2 倍；六级流水 14 而非 6×9=54，也非 9×?；首条指令仍需 6 段（建立时间），不能省。

8. 影响流水线性能的因素（三类相关）（8.3）

1）结构相关（课件 p.433）：不同指令**争用同一功能部件**产生资源冲突（如指令 1 与指令 4、指令 2 与指令 5 争用存储器）。解决办法：**停顿**、**指令存储器和数据存储器分开**、**指令预取技术**（适用于访存周期短的情况）。 **2）数据相关**（课件 p.434）：不同指令因重叠操作，可能改变操作数的**读/写**访问顺序。三种：

相关	含义	课件例	语义
RAW 写后读	后指令读前指令要写的寄存器	SUB R1,R2,R3；ADD R4,R5,R1	(R2)−(R3)→R1；(R5)+(R1)→R4
WAR 读后写	后指令写前指令要读的寄存器	STA M,R2；ADD R2,R4,R5	(R2)→M；(R4)+(R5)→R2
WAW 写后写	两指令写同一寄存器	MUL R3,R2,R1；SUB R3,R4,R5	(R2)×(R1)→R3；(R4)−(R5)→R3

解决办法：**后推法**（推迟相关指令）、**采用旁路技术**（forwarding，把结果直接旁路给需要的部件）。 **3）控制相关**（课件 p.435、436）：由**转移指令**引起。例：BNE 必须等 CPX 的结果才能判断转移还是顺序。设指令 3 是转移指令，则其后已预取的指令 4~7 作废，转移目标（指令 15）重新取指，产生**转移损失**。



要点：结构相关=资源冲突；数据相关=操作数读写顺序；控制相关=转移。三者名称与成因一一对应是选择题核心。 **易错警示**：① RAW 是"写后读"（先写后读同一寄存器），不要按字面"读后写"记反；② 旁路技术解决数据相关，不能解决控制相关；③ 结构相关靠"分开指令/数据存储器"或预取，不是靠旁路。

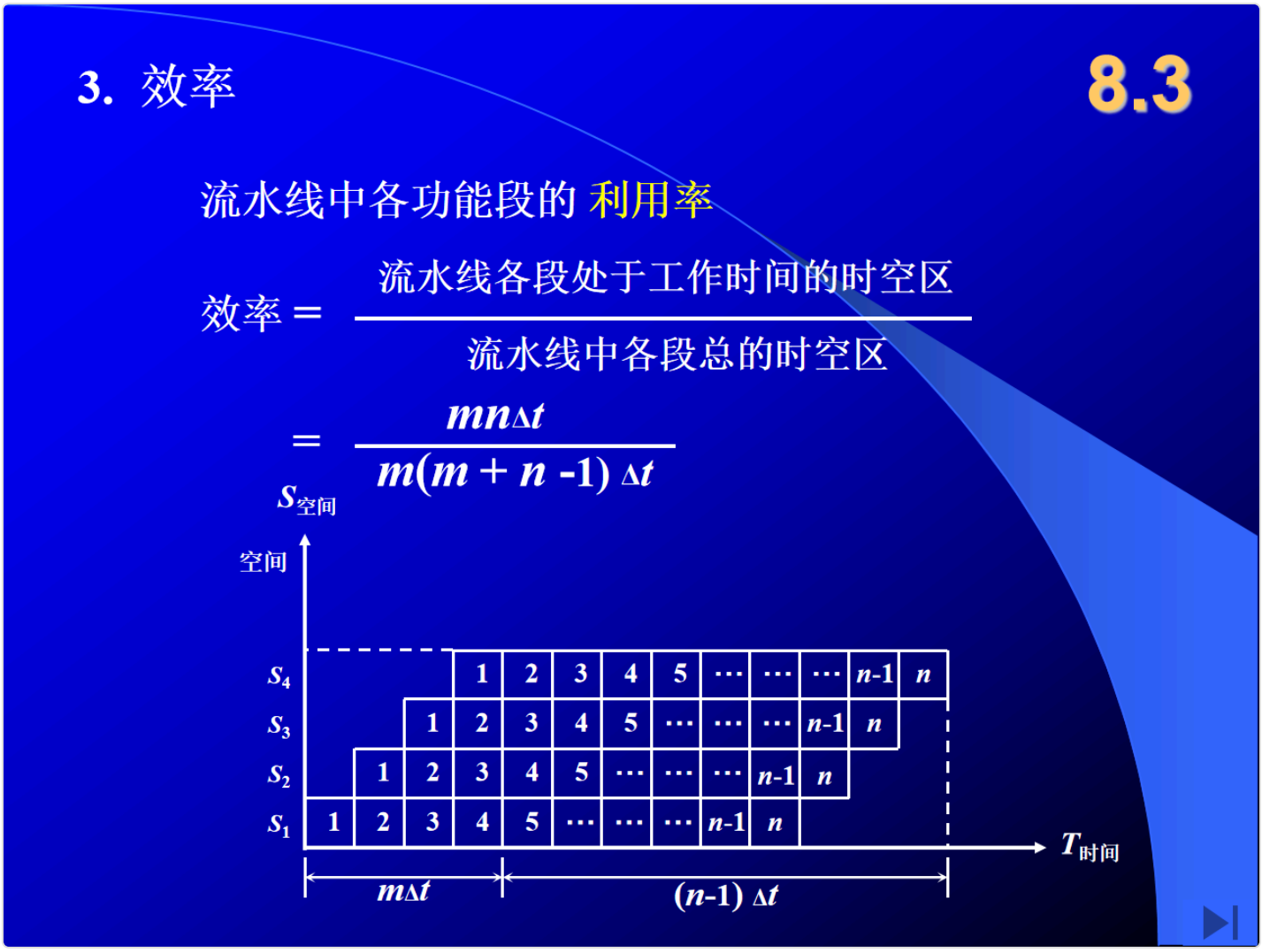
9. 流水线性能指标（8.3）

设 m 段流水线、各段时间均为 Δt、连续处理 n 条指令。

1）吞吐率 Tp（课件 p.437）：单位时间内流水线完成的指令（或输出结果）数量。 - 最大吞吐率：**Tpmax = 1/Δt** - 实际吞吐率：**Tp = n / [m•Δt + (n−1)•Δt] = n / [(m+n−1)•Δt]**

2）加速比 Sp（课件 p.438）：m 段流水线速度与等功能非流水线速度之比。 - 流水总时间 T = m·Δt + (n−1)·Δt = (m+n−1)·Δt - 非流水总时间 T′ = n·m·Δt - **Sp = T′/T = nm•Δt / [(m+n−1)•Δt] = nm / (m+n−1)**

3）效率 E（课件 p.439、440）：流水线中各功能段的利用率（设备不可能一直工作，因有**建立时间**和**排空时间**）。 - **E = 各段处于工作时间的时空区 / 各段总的时空区 = mn•Δt / [m(m+n−1)•Δt] = n / (m+n−1)**



三者关系（补充推导）：E = Sp/m；Tp = E/Δt；当 n→∞ 时 Sp→m、E→1、Tp→Tpmax。 **要点**：分母统一为 (m+n−1)；Sp 分子 nm、E 分子 n、Tp 分子 n（再除 Δt）。 **易错警示**：① 效率分母是 m(m+n−1)Δt（总时空区=段数×总时间），不是 (m+n−1)Δt；② 加速比上限是 m 不是 n；③ 实际吞吐率永远小于最大吞吐率（因建立/排空）。

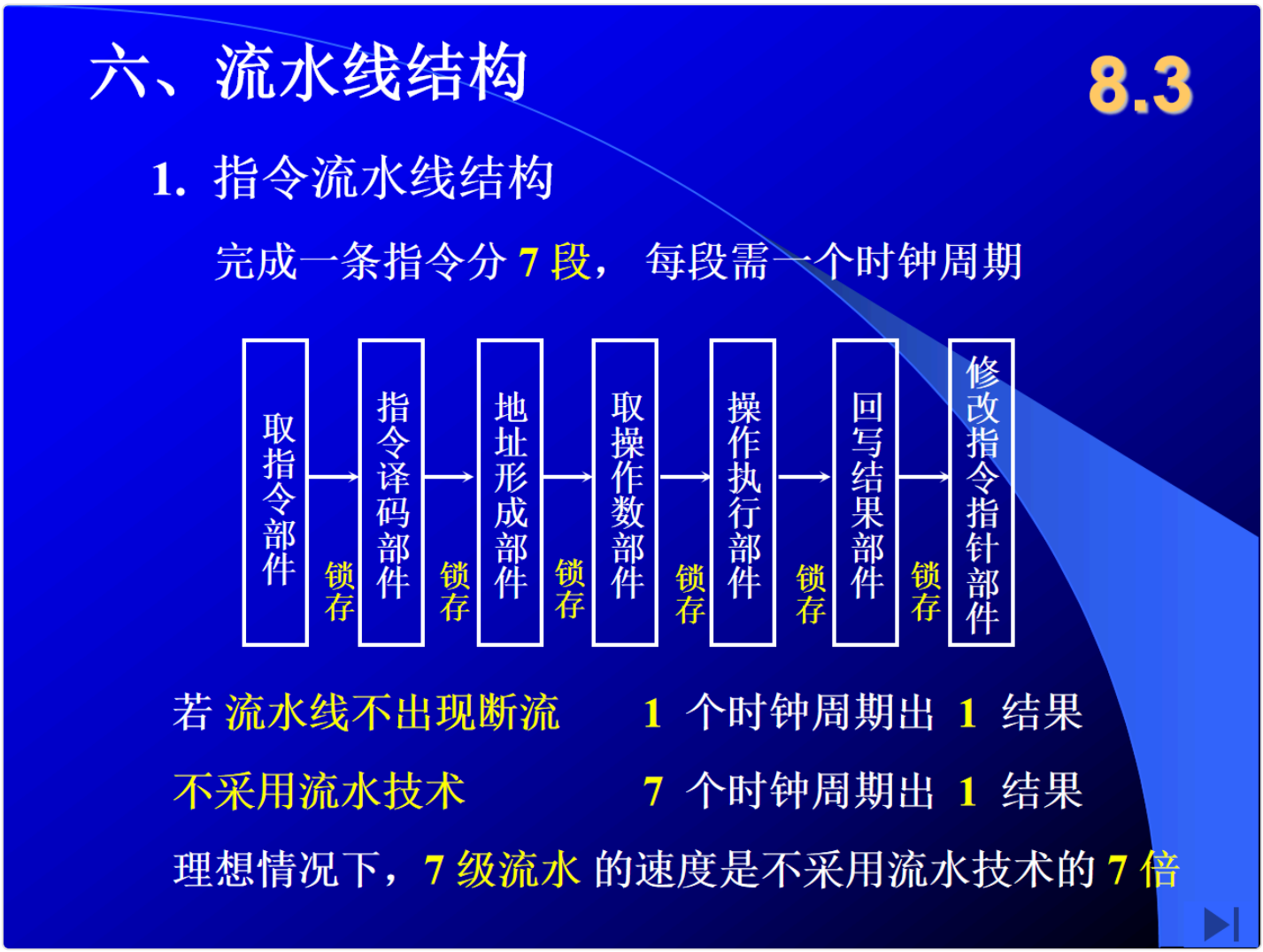
10. 流水线的多发技术（8.3）

技术	核心思想	能否调整指令执行顺序	依赖	课件页
超标量	每个时钟周期内**并发多条独立指令**；配置**多个功能部件**	不能	编译优化把可并行指令搭配	p.441
超流水线	一个时钟周期内**再分段（3 段）**；一个功能部件**使用多次（3 次）**；速度为原来 **3 倍**	不能	编译程序优化	p.442
超长指令字 VLIW	编译程序**挖掘指令间潜在并行性**，将多条能并行操作的指令**组合成一条**；具有**多个操作码字段**的超长指令字（可达几百位）；采用**多个处理部件**	（由编译组合）	编译程序	p.443

要点：超标量=空间并行（多部件同时发多条）；超流水线=时间细分（一段拆多段、部件复用）；VLIW=编译把多条打包成一条长指令。三者都靠编译优化，超标量/超流水线**不能调整指令执行顺序。** **易错警示**：超流水线"3 倍"来自一个时钟周期内再分 3 段、部件用 3 次，不是配 3 套部件（那是超标量思路）；VLIW 的并行性由**编译**静态挖掘，不是硬件动态调度。

11. 流水线结构（8.3）

1）指令流水线结构（课件 p.444）：完成一条指令分 **7 段**，每段一个时钟周期，段间用**锁存器**隔离： 取指令部件 → 指令译码部件 → 地址形成部件 → 取操作数部件 → 操作执行部件 → 回写结果部件 → 修改指令指针部件。 若流水线不出现断流：**1 个时钟周期出 1 结果**；不采用流水：**7 个时钟周期出 1 结果**；理想情况下 7 级流水速度是不采用流水的 **7 倍**。



2）运算流水线（课件 p.445）：浮点加减运算可分**对阶、尾数求和、规格化**三段，段间锁存器。**分段原则：每段操作时间尽量一致。** **要点**：段间锁存器是流水线能重叠的物理基础；分段等长才能最大化效率。 **易错警示**：运算流水的分段原则是"时间尽量一致"，不是"功能尽量多"；段数越多理想加速比越大，但锁存开销与相关冲突也越多（补充常识）。

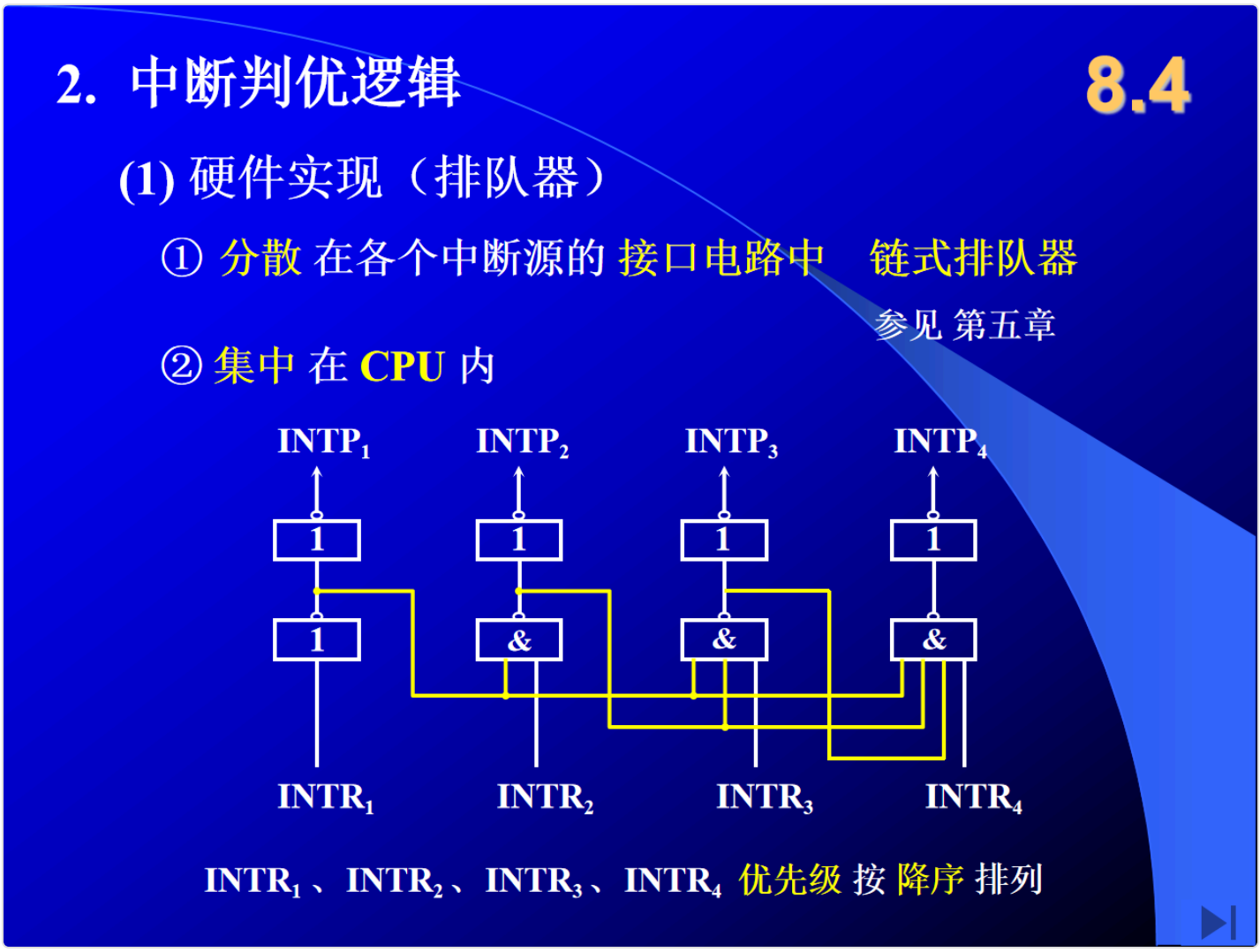
12. 中断系统概述、请求标记与判优（8.4）

引起中断的因素（课件 p.446）：(1) 人为设置的中断（如转管指令）；(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（如键盘中断现行程序）。

中断系统需解决的 7 个问题（课件 p.447）：① 各中断源如何向 CPU 提出请求；② 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 处理中断过程中又出现新中断怎么办。靠**硬件 + 软件**共同解决。

中断请求标记 INTR（课件 p.448）：**一个请求源对应一个 INTR 触发器**；多个 INTR 组成**中断请求标记寄存器**；INTR 可**分散**在各中断源接口电路中，或**集中**在 CPU 中断系统内。例：掉电、过热、阶上溢、主存读写校验错、非法除法、键盘输入、打印机输出。

中断判优逻辑（课件 p.449、450）： - **硬件实现（排队器）**：① 分散在各中断源接口电路中→**链式排队器**（参见第五章）；② 集中在 CPU 内→排队器电路。INTR1~INTR4 优先级**降序**，输出 INTP1~INTP4（高优先级屏蔽低优先级）。 - **软件实现（程序查询）**：按优先级降序依次查询"是否 A 请求？是否 B 请求？是否 C 请求？"，命中即转对应服务程序入口。



要点：判优=决定"同时请求时先响应谁"；硬件排队器快、软件查询慢但灵活。 **易错警示**：INTR 是"请求标记"（每源一个触发器），INTP 是"排队器输出"（判优后的选中信号），二者不要混；链式排队器属于"分散"式而非集中式。

13. 中断服务程序入口地址的寻找（8.4）

1）硬件向量法（课件 p.451）：排队器输出→**向量地址形成部件**→产生**向量地址**（如 12H、13H、14H）→该地址单元内容即服务程序**入口地址**（200、300、400）。两种存放形式：向量地址单元放 **JMP 入口** 指令，或直接放**入口地址**。

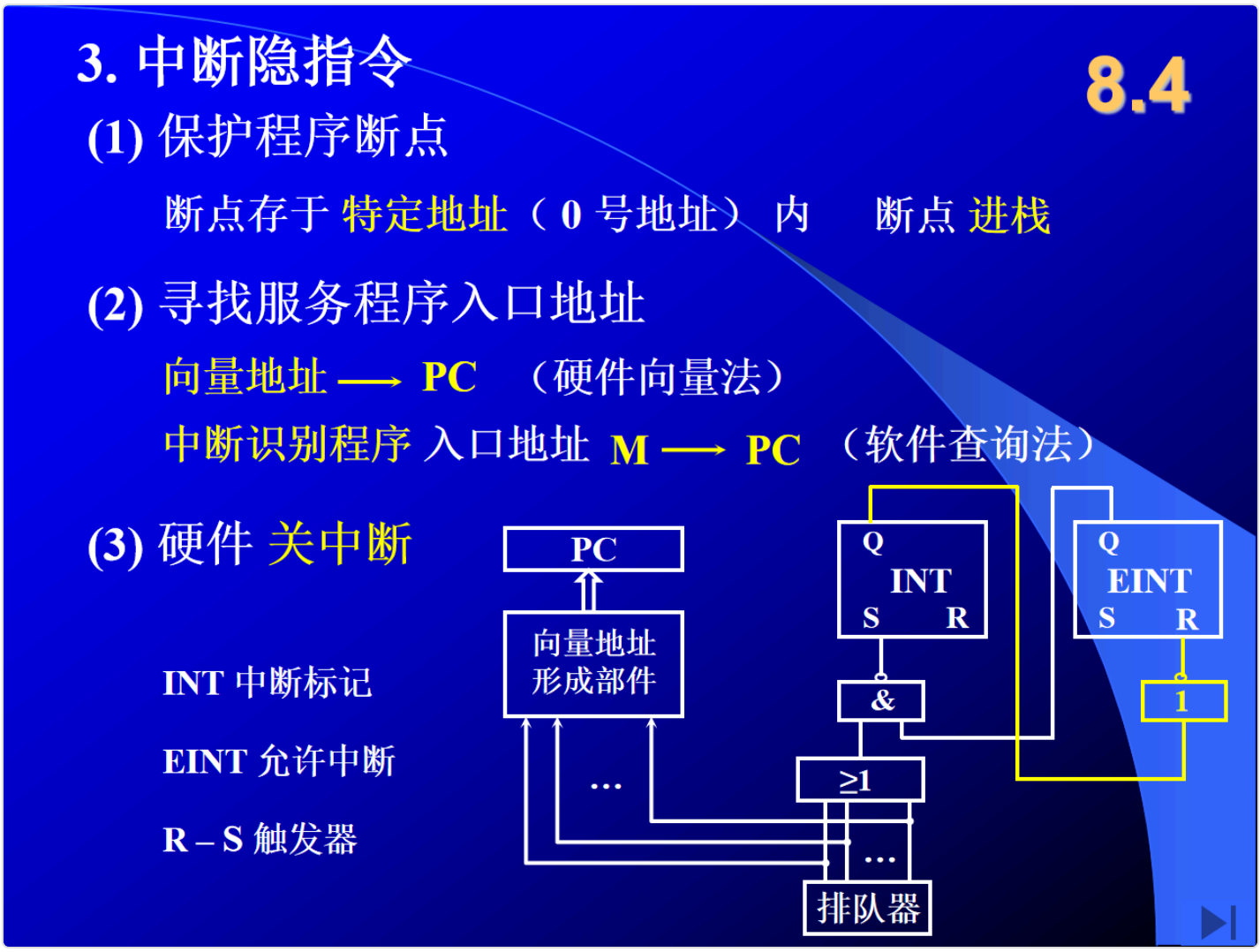


2）软件查询法（课件 p.452）：中断识别程序入口地址 **M**；按降序依次用 **SKP DZ i#**（D 为完成触发器，D=0 跳）+ **JMP i# SR** 查询并转服务程序；八个中断源 1~8 降序。 **要点**：硬件向量法靠硬件直接给出入口（快）；软件查询法靠一段识别程序逐个查（慢、省硬件）。向量地址≠入口地址：向量地址是"存放入口地址（或 JMP）的单元地址"。 **易错警示**：12H/13H/14H 是**向量地址**，200/300/400 才是**入口地址**，填空常把两者对调设 trap。

14. 中断响应与中断隐指令（8.4）

响应条件（课件 p.453）：**允许中断触发器 EINT = 1**。 **响应时间**（课件 p.453）：**指令执行周期结束时刻**由 CPU 发查询信号（不是任意时刻）。

中断隐指令（课件 p.454）——硬件自动完成、无对应机器指令的三件事： 1. **保护程序断点**：断点存于特定地址（0 号地址）内，或断点**进栈**； 2. **寻找服务程序入口地址**：硬件向量法→**向量地址→PC**；软件查询法→**中断识别程序入口地址 M→PC**； 3. **硬件关中断**：置 EINT=0（用 INT 中断标记、EINT 允许中断两个 R-S 触发器配合）。



要点：隐指令=硬件在中断周期自动做的三步；"关中断"是硬件做的，保证保护断点/找入口过程不被打断。 **易错警示**：① 中断响应在**执行周期结束**而非取指结束；② 保护**断点**由隐指令（硬件）完成，保护**现场**（寄存器）由服务程序（软件 PUSH）完成，二者主体不同；③ 隐指令不是指令系统里的一条指令，程序员看不到。

15. 保护现场、恢复现场与多重中断概念（8.4）

保护现场（课件 p.455）：寄存器内容用 **PUSH** 保存（视不同请求源而定）；断点已由中断隐指令保存。 **恢复现场**（课件 p.455）：用 **POP** 恢复寄存器；**中断返回 IRET**。 顺序：中断隐指令完成（断点）→ 服务程序保护现场 → 中断服务 → 恢复现场 → 中断返回。

多重中断概念（课件 p.456）：在执行某中断服务程序时又响应更高优先级中断，形成嵌套。主程序断点 k+1、第一次服务断点 l+1、第二次服务断点 m+1，逐层返回。 **实现多重中断的条件**（课件 p.457）：(1) **提前设置开中断指令**（在服务程序中尽早开中断）；(2) **优先级别高的中断源有权中断优先级别低的中断源**（A、B、C、D 降序）。 **要点**：单重中断=服务期间关中断不再响应；多重中断=服务期间开中断允许更高优先级嵌套。 **易错警示**：多重中断必须"提前开中断"，若全程关中断则退化为单重；嵌套只能由高优先级打断低优先级，反向不行。

16. 中断屏蔽技术（8.4）

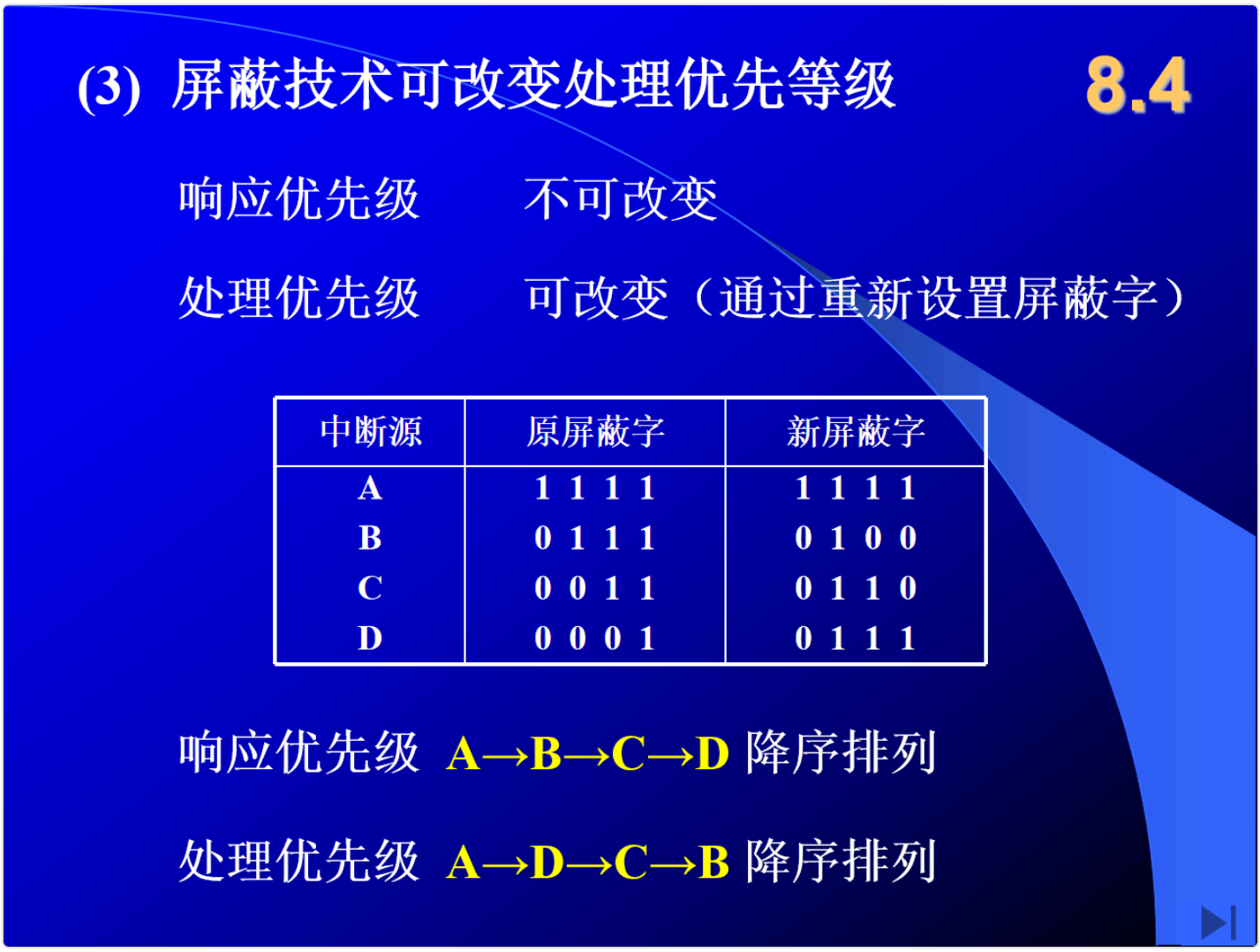
屏蔽触发器的作用（课件 p.458）：**MASK=0（未屏蔽）**时 INTR 能被置 1；**MASKi=1（屏蔽）**时 **INTPi=0**（不能被排队选中）。即屏蔽字通过封锁排队器输入改变"谁可以被响应/处理"。

屏蔽字（课件 p.459）：16 个中断源 1~16 降序。优先级 k 的屏蔽字：第 1~k−1 位为 0（允许比它高的），第 k~16 位为 1（屏蔽自身及比它低的）。如优先级 1 全 1；优先级 2 = 0 后全 1；…优先级 16 = 全 0 末位 1。

屏蔽技术可改变处理优先等级（课件 p.460）： - **响应优先级**：由硬件排队器决定，**不可改变**（A→B→C→D 降序）。 - **处理优先级**：**可改变**（通过重新设置屏蔽字）。

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

原屏蔽字→处理优先级 A→B→C→D；新屏蔽字→处理优先级 **A→D→C→B**。（屏蔽字位序 A B C D；1=屏蔽该源。）



CPU 执行程序轨迹：原屏蔽字下 A、B、C、D 同时请求，处理完顺序 B→C→D→A（课件 p.461）；新屏蔽字下处理完顺序 D→C→B→A（课件 p.462）。 **屏蔽技术的其他作用**（课件 p.462）：便于程序控制；可以人为地屏蔽某个中断源的请求。 **新屏蔽字的设置（服务程序内顺序）**（课件 p.463）：保护现场 → **置屏蔽字** → **开中断** → 中断服务 → **关中断** → 恢复现场 → **恢复屏蔽字** → **开中断** → 中断返回。

要点：响应优先级硬件定死、处理优先级屏蔽字可调——这是本章最核心的辨析；屏蔽字"1=屏蔽、0=开放"，某源运行时屏蔽自身+处理优先级低于它的源。 **易错警示**：① 改屏蔽字改的是**处理**优先级，**响应**优先级不变；② 屏蔽字中"1"是屏蔽不是开放，写反全错；③ 置屏蔽字要在开中断**之前**、恢复屏蔽字要在恢复现场之后/开中断之前，顺序错会丢中断或错嵌套。

17. 多重中断的断点保护（8.4）

两种方式（课件 p.464）： - (1) **断点进栈**：天然支持多重中断（每层断点压栈，互不覆盖）。 - (2) **断点存入 0 地址**：中断周期 0→MAR、(PC)→MDR、(MDR)→存储器。**三次中断三个断点都存入 0 地址→后一次覆盖前一次→断点丢失**。问题：**如何保证断点不丢失?**

0 地址方式的断点保护程序（课件 p.465）：

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（存放 ACC 内容）
	LDA 0 / STA RETURN	0 地址内容转存（把断点转存到 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

思路：服务程序一开始就把 0 地址里的断点**转存**到 RETURN 单元，腾出 0 地址供下一次中断使用；返回时用 **JMP @ RETURN** 间址跳回断点。配合"置屏蔽字"控制嵌套。



要点：进栈法无需额外转存；0 地址法必须"先转存断点再开中断"，否则多重中断覆盖断点。 **易错警示**：0 地址法若不在开中断前转存断点，第二层中断会把第一层断点覆盖→返回错误；JMP @ RETURN 是**间址**返回（RETURN 里存的是断点地址），不是直接跳 RETURN。

18. 易错点与高频陷阱汇总

1. PC 属控制寄存器但**用户可见**；MAR/MDR/IR 用户不可见（p.418）。
2. 地址总线**单向**、数据/控制总线**双向**（p.415）。
3. NOP 指令周期**只有取指周期**（p.421）。
4. 间址周期取的是**有效地址**（取地址），执行周期才取操作数（p.424）。
5. 中断周期对存储器是**写**（存断点），取指/间址是**读**（p.425–427）。
6. 取指周期内 (PC)+1，不在执行周期（p.425）。
7. 中断响应在**执行周期结束**时刻，条件 EINT=1（p.453）。
8. 保护**断点**=硬件隐指令；保护**现场**=软件 PUSH（p.454、455）。
9. 向量地址（12H…）≠入口地址（200…）（p.451）。
10. 并行=同一时刻、并发=同一时间段；指令级=硬件细粒度、过程级=软件粗粒度（p.429）。

11. 二级流水完全重叠→周期减半、速度提高**1 倍**（=2 倍）（p.430）。
12. m 段 n 条总时间=(m+n−1)Δt；六级 9 条=14 不是 54（p.432）。
13. RAW=写后读、WAR=读后写、WAW=写后写，按"先操作后操作"命名（p.434）。
14. 旁路/后推解决**数据**相关；分开指令/数据存储器+预取解决**结构**相关；转移引起**控制**相关（p.433–436）。
15. Sp=nm/(m+n−1)、E=n/(m+n−1)、Tp=n/[(m+n−1)Δt]；E=Sp/m（p.437–440）。
16. 效率分母=m(m+n−1)Δt（总时空区），漏乘 m 必错（p.440）。
17. 超标量/超流水线**不能调整指令执行顺序**；超流水线 3 倍靠周期内再分 3 段（p.441–442）。
18. 响应优先级**不可改**、处理优先级**可改**（屏蔽字）（p.460）。
19. 屏蔽字 1=屏蔽、0=开放；某源屏蔽自身+处理优先级更低者（p.459、460）。
20. 0 地址断点法必须**先转存断点再开中断**，否则多重中断覆盖断点（p.464、465）。
21. 多重中断两条件：提前开中断 + 高优先级可打断低优先级（p.457）。
22. 硬连线=组合逻辑、微程序=存储逻辑（p.419）。

19. 典型例题精讲

例1（流水线三指标，课件 p.432/437–440 数据）：六级流水（m=6）各段 Δt，连续执行 9 条指令（n=9）。求总时间、Tp、Sp、E。 解：总时间 T=(m+n−1)Δt=(6+9−1)Δt=14Δt（与课件 p.432 的 14 一致）。 非流水 T′=nmΔt=54Δt（与课件 54 一致）。 Sp=T′/T=54/14=27/7≈3.86。 E=n/(m+n−1)=9/14≈0.643。 Tp=n/T=9/(14Δt)；Tpmax=1/Δt；验证 E=Sp/m=(27/7)/6=9/14 ✓。 教学点：先写根公式 (m+n−1)Δt，再套三指标，最后用 E=Sp/m 自检。

例2（屏蔽字→处理优先级，课件 p.460 变式）：四源 A、B、C、D 响应优先级降序。某源运行时屏蔽字为：A=1111、B=0100、C=0110、D=0111（位序 ABCD，1=屏蔽）。求处理优先级。 解：看每个源运行时"允许谁打断"（屏蔽字中 0 的位，除自身外）： A=1111 允许无人→A 最高； D=0111 允许 A→D 次之； C=0110 允许 A、D→C 再次； B=0100 允许 A、C、D→B 最低。 故处理优先级 A→D→C→B（与课件 p.460 结论一致）。 教学点：处理优先级=看"谁运行时不被别人打断"，屏蔽字 0 位=可打断者；逐源列出允许集合再排序。

例3（取指+中断数据流默写，课件 p.425/427）：写出取指周期与中断周期的微操作序列。 解：取指：(PC)→MAR；CU 发读；M(MAR)→MDR（经数据总线）；(MDR)→IR；(PC)+1→PC。 中断：CU 形成断点地址→MAR；(PC)→MDR；CU 发写；(MDR)→M(MAR)；向量地址→PC。 教学点：抓住"读/写"性质与 PC 的去向（取指+1、中断被替换），即可完整默写。

附：本章速查表

周期与标志：指令周期=取指+（间址）+执行+（中断）；标志触发器 FE/IND/EX/INT；访存性质 取指令/取地址/取操作数/存断点。 **数据流口诀**：取指读指令 PC+1；间址读地址回 IR；执行读写操作数；中断写断点换 PC。 **流水线公式**（m 段、等长 Δt、n 条）： - T = (m+n−1)·Δt - Tpmax = 1/Δt；Tp = n/[(m+n−1)·Δt] - Sp = nm/(m+n−1) - E = n/(m+n−1) = mn·Δt/[m(m+n−1)·Δt] - 关系：E = Sp/m；Tp = E/Δt；n→∞ 时 Sp→m、E→1 **六级流水例**：m=6、n=9 → T=14Δt、串行 54Δt。 **三相关**：结构（资源冲突：停顿/分开存储器/预取）；数据（RAW/WAR/WAW：后推/旁路）；控制（转移：转移损失）。 **多发技术**：超标量（多部件并发多条、不可调

序）；超流水线（周期内再分 3 段、部件用 3 次、3 倍、不可调序）；VLIW（编译打包多条成一条、多操作码字段几百位、多处理部件）。 **指令流水 7 段**：取指→译码→地址形成→取操作数→操作执行→回写结果→修改指令指针；理想 7 倍。 **运算流水 3 段**：对阶→尾数求和→规格化；分段原则各段时间尽量一致。 **中断**：响应条件 EINT=1；响应时间执行周期结束；隐指令三件事=保护断点+找入口+硬件关中断；保护现场 PUSH/恢复 POP/返回 IRET。 **入口寻找**：硬件向量法（向量地址 12H/13H/14H→入口 200/300/400）；软件查询法（入口 M，SKP DZ + JMP）。 **判优**：硬件排队器（分散链式/集中 CPU 内）；软件查询（降序逐个问）。 **屏蔽**：MASK=1 屏蔽→INTP=0；响应优先级不可改、处理优先级可改；屏蔽字 1=屏蔽 0=开放；新屏蔽字设置顺序=保护现场→置屏蔽字→开中断→服务→关中断→恢复现场→恢复屏蔽字→开中断→返回。 **断点保护**：进栈（天然多重）；0 地址（需先 LDA 0/STA RETURN 转存再 ENI，返回 JMP @ RETURN）。