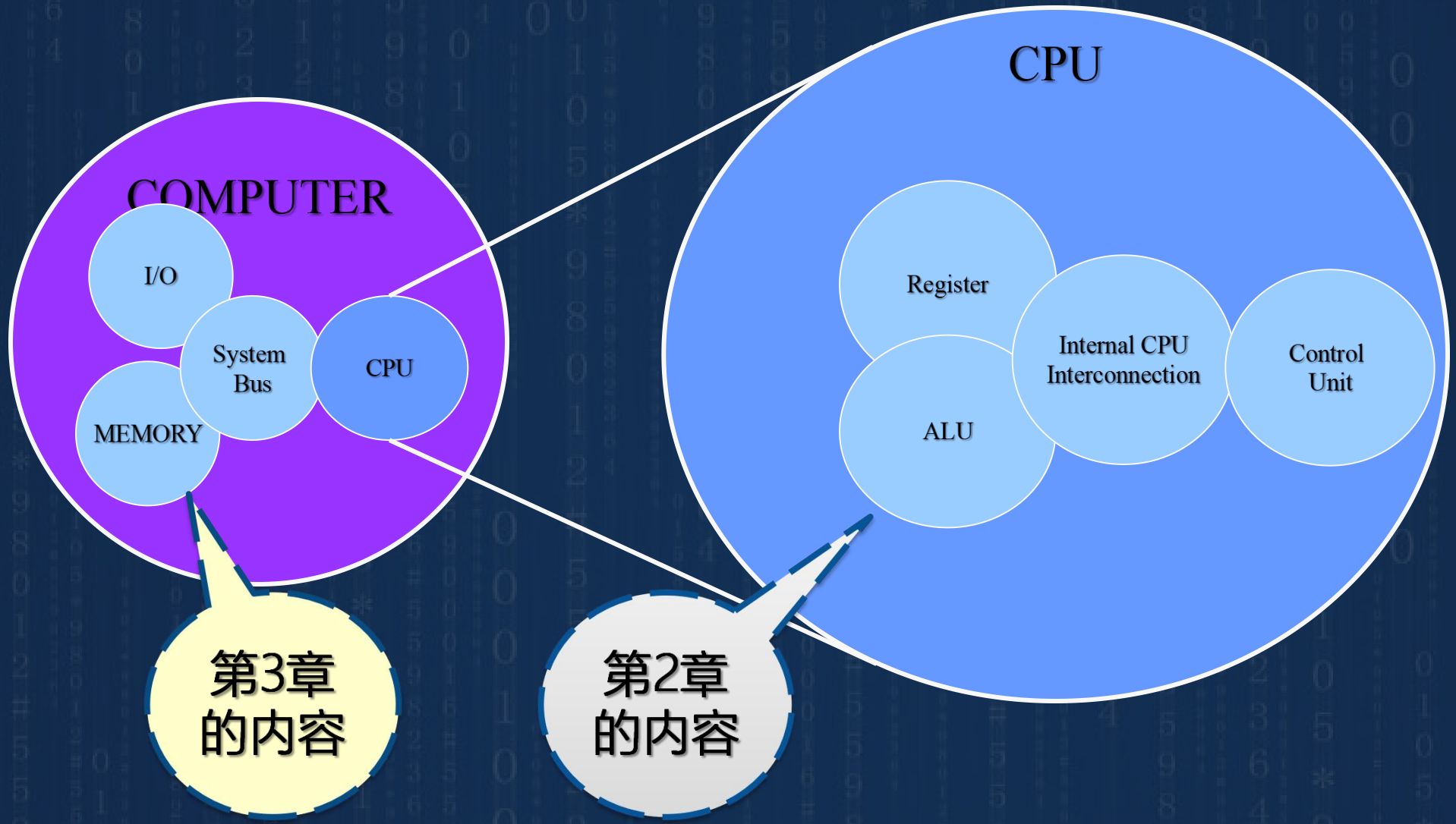
南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

第 4 章 指令系统与控制单元

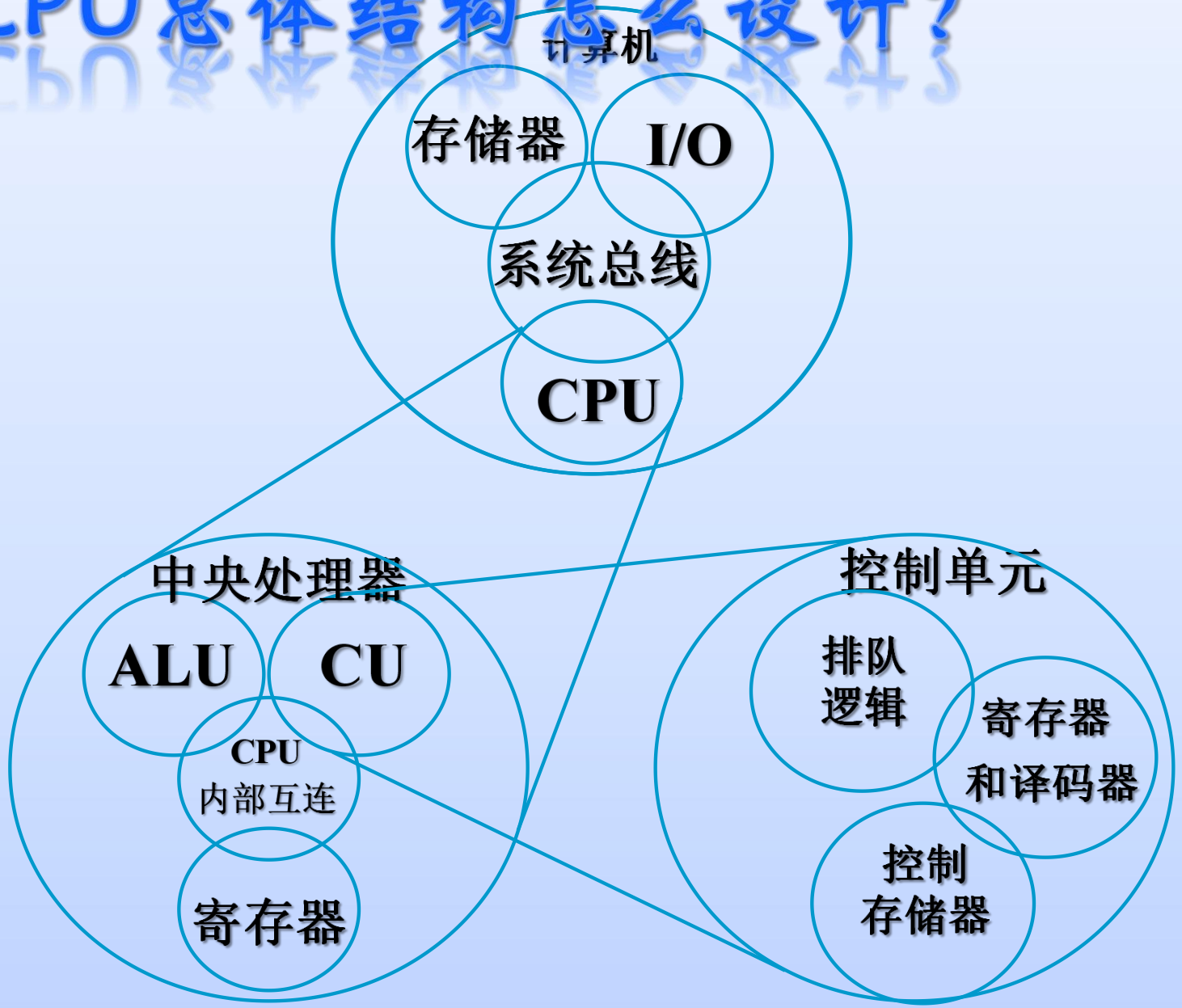
主讲：张功萱

课堂思考：

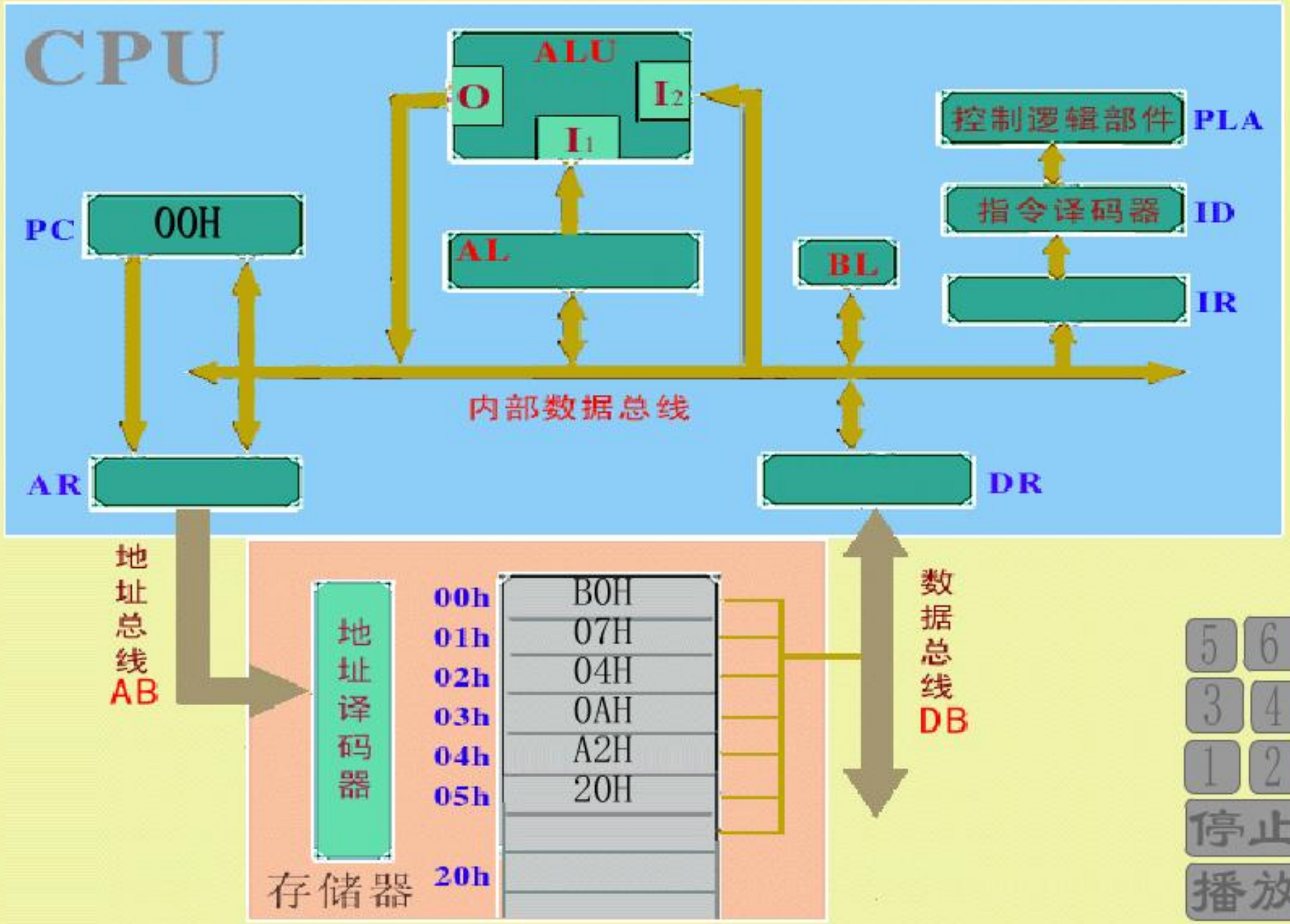
- 设计一个32位的“假想机”，给定部件如下：
- 具有ADD功能的32位ALU；
- 按字节编址的RAM（与机器的MAR, MDR连接）；
- 32位PC，32位IR;
- 32个32位的寄存器组（R0,R1,...,R31）；
- 运行指令：lw R2, 8(R1);
- ADD R2, R2, R3


COMPUTER
I/O
System Bus
CPU
MEMORY
CPU
Register
Internal CPU Interconnection
Control Unit
ALU
第3章的内容
第2章的内容

CPU总体结构怎么设计？



指令的执行演示与剖析


CPU
ALU
O
I2
I1
AL
BL
PC
00H
控制逻辑部件
PLA
指令译码器
ID
IR
内部数据总线
AR
DR
地址总线AB
数据总线DB
地址译码器
00h
01h
02h
03h
04h
05h
20h
B0H
07H
04H
0AH
A2H
20H
存储器
5
6
3
4
1
2
停止
播放

引发的思考

- 1. 什么时候表明是“指令”，什么时候又是“数据”？
- 2. 指令的功能怎样实现？
- （如ALU的加、减控制等等）
- 3. 指令和数据的流向怎么控制？
-
-

控制单元初阶-控制器设计

第4.3节

1. 指令执行的步骤有哪些?
2. 控制器的功能包括哪些?
3. 控制器有哪些部件组成?
4. 什么是控制器的组织方式?

- 控制器和运算器一起组成中央处理器，即CPU（Central Process Unit）。
- 控制器的功能：
- 根据事先编好并存放在存储器中的解题程序（机器指令），控制各部件有条不紊地、自动协调地进行工作。
- 控制器是计算机的指挥和控制中心，由它把计算机的运算器、存储器、I/O设备等联系成一个有机的系统，并根据各部件具体要求，适时地发出各种控制命令，控制计算机各部件自动、协调地进行工作。

4.3.1 指令执行的基本步骤

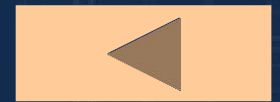
- 按照现代Von.Neumann计算机结构，大多数程序（机器指令有序集）是放在主存储器里的，因此，必须去访问存储器，读取指令。
- 这样，计算机运行程序（执行指令有序集）的基本过程是：
- 1. 取指令
- 根据指令地址（由PC提供），从存储器中取出所要执行的指令。（放到什么地方？）

2. 分析指令

- ① 对取出的指令进行译码分析。确定指令应完成的操作，产生相应操作的控制电位，参与形成该指令功能所需要的全部控制命令（微操作控制信号）。（接通指令所需要的控制信号）
- ② 根据寻址方式的分析和指令功能要求，形成操作数的有效地址，并按此地址取出操作数据（运算型指令）或形成转移地址(转移类指令)，以实现程序转移。

3. 执行指令

- 根据指令分析所产生的操作控制信号和形成的有效地址，按一定的算法形成指令操作控制序列，控制有关部件完成指令规定的功能。
- 一条指令执行结束，若没有异常情况和特殊请求，则按程序顺序，再去取出并执行下一条指令。
- 如果出现“异常情况”，则处理异常。（第6章的内容）



4.3.2 控制器的基本功能

- 1．控制指令的正确执行
- 包括指令流出的控制、分析指令和执行指令的控制、指令流向的控制。
- (1) 指令流出控制（对取指令的控制）
- 取指令时需进行的操作
- (PC)→MAR，Read　　；给出指令地址，并向MEM发出读命令
- (MDR)→IR　　；读出的指令经MDR存放到指令寄存器IR中

  (PC)增量→PC　　；为取下一条指令作准备

- (2) 分析指令和执行指令的控制
- IR中的指令经指令译码器(ID)译码分析，确定操作性质，判明寻址方式并形成操作数的有效地址。
- 控制器根据分析的结果和形成的有效地址产生相应的操作控制信号序列，控制有关的部件完成指令所规定的操作功能。
- 例：设某指令的IR_{15}～IR_{12}＝0000时为MOV指令，则 MOV的控制信号为：

$$MOV=\overline{IR}_{15}\cdot\overline{IR}_{14}\cdot\overline{IR}_{13}\cdot\overline{IR}_{12}$$

- (3) 指令流向的控制
- 指令流向控制即下条指令地址的形成控制。
- ① 按指令序列顺序执行时，通过PC自动增量形成下条指令的地址。
- ② 当需要改变指令流向时，需改变程序计数器PC中的内容。
- 转移指令的执行：把形成的转向地址送入PC；
- 转子指令的执行：把子程序入口地址送入PC；
- 中断处理：将中断服务程序入口地址送入PC。
- 为了正确返回，转子和中断还需保留PC被改变之前的内容(即返回地址)。

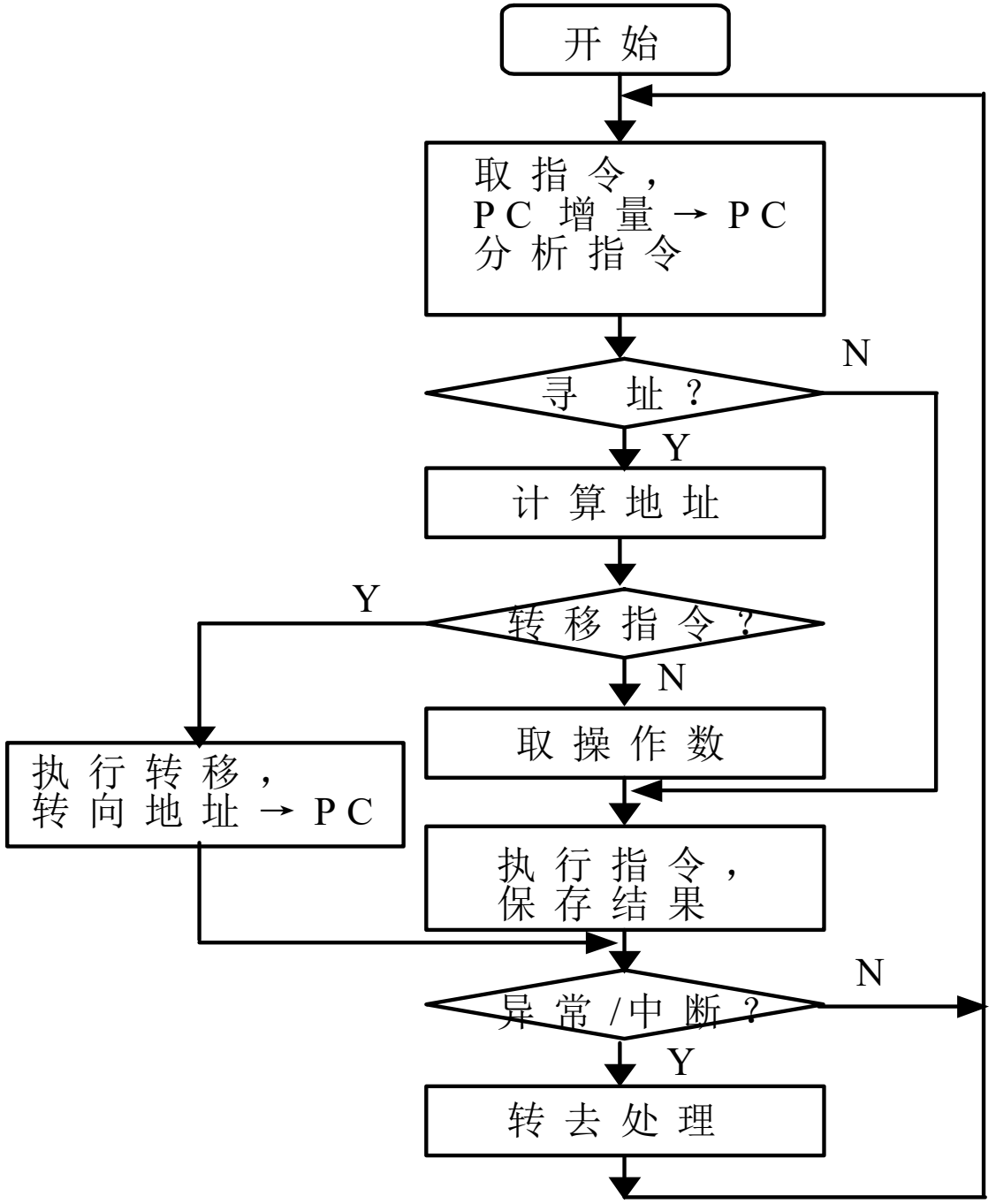
- **2. 控制程序和数据的输入及结果的输出**
- 为完成某项任务而编制的程序及所需数据，必须通过某些输入设备预先存放在存储器中，运算结果要用输出设备输出。所以必须由控制器统一指挥，完成程序和数据的输入及结果的输出。

- **3． 异常情况和特殊请求的处理**
- 机器在运行程序过程中，往往可能会遇到一些异常情况（如电源掉电、运算溢出等）或某些特殊请求（如打印机请求传送打印字符等）。这些异常和请求往往是事先无法预测的，控制器必须具有检测和处理这些异常情况和特殊请求的功能。

- 从宏观上看，每条指令的执行过程均是取指、译码、执行，但每条指令有不同的操作序列，需要在不同时间产生不同的控制序列，并有严格的时序要求。控制器必须根据不同指令产生不同的控制序列。

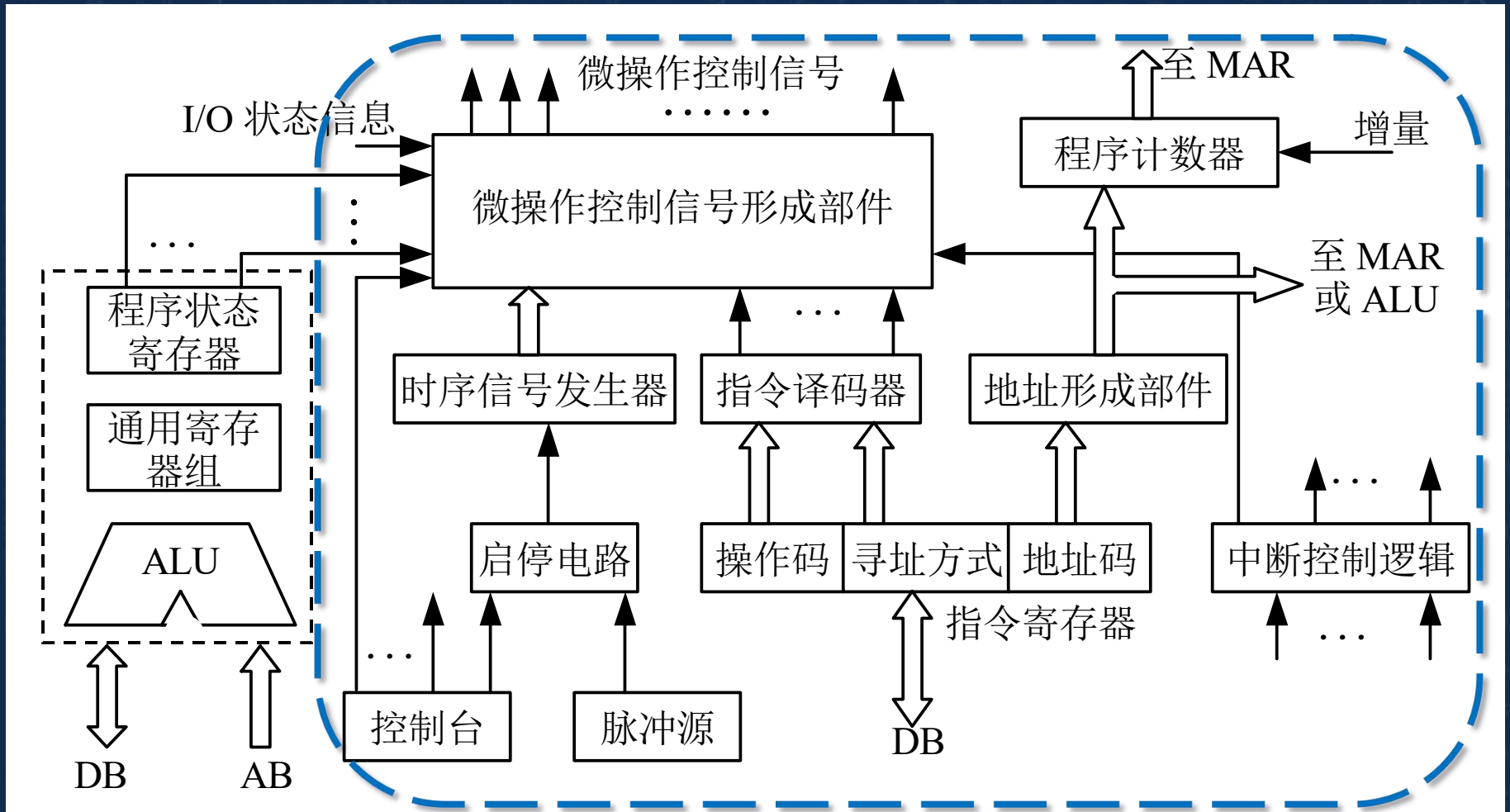
- 控制器在工作时，需要根据不同的指令、不同的状态条件，在不同的时间，产生不同的控制信号，控制计算机的各部件（ALU、RAM、I/O等）自动、协调地进行工作。

指令执行的一般流程


开 始
取 指 令，
PC 增 量 → PC
分 析 指 令
寻 址 ？
N
Y
计 算 地 址
转 移 指 令 ？
Y
N
取 操 作 数
执 行 转 移，
转 向 地 址 → PC
执 行 指 令，
保 存 结 果
异 常 /中 断 ？
N
Y
转 去 处 理

4.3.3 控制器的组成方式

- 1. 控制器的组成结构
- 下图给出CU的主要构成逻辑图



1）指令部件

- 指令部件的主要功能是完成取指令和分析指令。
- (1) 程序计数器 PC （指令计数器、指令地址寄存器）
- PC用于保证程序按规定的序列正确运行，并提供将要执行指令的指令地址。
- 由于PC可以指向主存中任一单元的地址，因此它的位数应能表示主存的最大容量并与主存地址寄存器MAR的位数相同。

- 在CPU中可以单独设置程序计数器，也可以指定通用寄存器中的某一个作为PC使用。
- 程序顺序执行时的PC增量可以通过PC本身的计数逻辑实现，也可以由运算器的ALU实现。不同机器，实现方法可有所不同。

- (2) 指令寄存器 IR
- 指令寄存器用于存放当前正在执行的指令。
- 当指令从主存取出后，经MDR传送到指令寄存器中，以便实现对一条指令执行的全部过程的控制。

- (3) 指令译码器 ID
- 指令译码器是指令分析部件，对指令寄存器中的指令操作码进行译码分析，产生相应操作的控制电位，提供给微操作控制信号形成部件。对寻址方式字段进行译码分析，以控制操作数有效地址的形成。

- (4) 地址形成部件
- 根据机器所规定的各种寻址方式，形成操作数有效地址。
- 在一些微、小型机中，为简化硬件逻辑，通常不设置专门的地址形成部件，而是借用运算器实现有效地址的计算。

2）时序控制部件

- 时序控制部件：用于产生一系列时序信号，为各个微操作定时，以保证各个微操作的执行顺序。
- 从宏观(即程序控制)上看，计算机的解题过程实质上是指令序列即一条条指令的执行过程。
- 从微观(即指令控制)上看，计算机的解题过程是微操作序列即一个个(或一组组)微操作的执行过程。
- 微操作：机器最简单的基本操作
- 一条指令的执行过程可以分解为若干微操作。这些微操作有着严格的时间顺序要求，不可随意颠倒。

- (1)脉冲源
- 脉冲源用于产生一定频率的主时钟脉冲。一般采用石英晶体振荡器作为脉冲源。计算机电源一接通，脉冲源立即按规定频率给出时钟脉冲。
- (2)启停电路
- 启停电路用于控制整个机器工作的启动与停止。实际上是保证可靠地送出或封锁主时钟脉冲，控制时序信号的发生与停止。
- (3)时序信号发生器
- 时序信号发生器用于产生机器所需的各种时序信号，以便控制有关部件在不同的时间完成不同的微操作。
- 不同的机器，有着不同的时序信号。在同步控制的机器中，一般包括周期、节拍、脉冲等三级时序信号。

3）微操作控制信号形成部件

- 微操作控制信号形成部件：根据指令部件提供的操作控制电位、时序部件所提供的各种时序信号以及有关的状态条件，产生机器所需要的各种微操作控制信号。
- 不同的指令，完成不同的功能，需要不同的微操作控制信号序列。每条指令都有自己对应的微操作序列。控制器必须根据不同的指令，在不同的时间，产生并发出不同的微操作控制信号，控制有关部件协调工作，完成指令所规定的任务。

4）中断控制逻辑（中断机构）

- 用于实现异常情况和特殊请求的处理。
- 中断机制是计算机非常重要的机制，中断机构的出现或安排使得计算机能够用于有关的“实时控制”（或“人工干预”），也是推动计算机应用的重要因素。

5）程序状态寄存器 PSR

- 程序状态寄存器：用于存放程序的工作状态（如管态、目态等）和指令执行的结果特征(如结果为零、结果溢出等)，把它所存放的内容称为程序状态字（PSW）。
- PSW表明了系统的基本状态，是控制程序执行的重要依据。不同的机器，PSW的格式及内容不完全相同。
- 为什么PC会被掌控？
- （与PSR的内容有关系）

- 例：8086 CPU中的PSW的格式
- （Flag----标志寄存器）

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

- CF：进位　　PF：奇偶　　AF：半进位
- ZF：结果为0　SF：符号　　TF：单步(陷阱)
- IF：中断允许　DF：地址增/减量　OF：溢出

6）控制台

- 控制台用于实现人与机器之间的通信联系，如启动或停止机器的运行、监视程序运行过程、对程序进行必要的修改或干预等。
- 早期有硬件控制台，用于设置地址和指令。现在，在大型机中有软件控制台。通过控制台命令，控制机器的启停，干预机器的工作。

2. 控制器的组成类型

- 控制器的组成方式主要是指微操作控制信号形成部件采用何种组成方式产生微操作控制信号。

- 根据产生微操作控制信号的方式不同，控制器可分为组合逻辑型、存储逻辑型、组合逻辑与存储逻辑结合型三种，它们的根本区别在于微操作信号发生器的实现方法不同，而控制器中的其他部分基本上是大同小异的。

1）硬联逻辑型

- 又称组合逻辑控制器，采用硬联逻辑技术来实现。其微操作信号发生器是由门电路组成的复杂树形网络构成的。
- 设计目标：使用最少器件数和取得最高操作速度。（计算机逻辑课程的核心内容）
- 优点：速度快。巨型机和RISC机为了追求高速度采用组合逻辑控制器。
- 缺点：微操作信号发生器结构不规整，设计、调试、维修较困难，难以实现设计自动化。一旦控制部件构成之后，要想增加新的控制功能是不可能的。

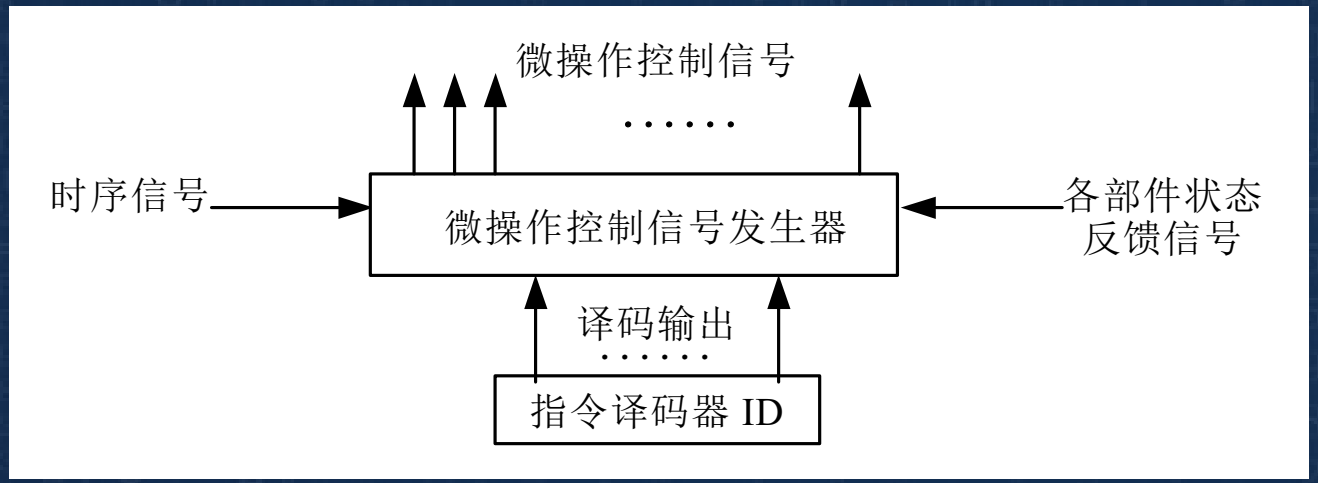
2）存储逻辑型

- 存储逻辑型控制器称为微程序控制器。它是采用存储逻辑来实现的。
- 存储逻辑型控制器的实现方法：
- 把微操作信号代码化，使每条机器指令转化成为一段微程序存入控制存储器中。执行指令时，读出控存中的微指令，由微指令产生微操作控制信号。
- 优点：设计规整，调试、维修便利，更改、扩充指令方便，易于实现自动化设计。
- 缺点：由于增加了一级控制存储器，所以指令的执行速度比组合逻辑控制器慢。

3）硬联逻辑和存储逻辑结合型

- 硬联逻辑和存储逻辑结合型控制器称为PLA控制器。
- PLA控制器是吸收前两种的设计思想来实现的。
- PLA控制器实际上也是一种组合逻辑控制器，但它的输出程序是可编程的，某一微操作控制信号由PLA的某一输出函数产生。
- PLA控制器是组合逻辑技术和存储逻辑技术结合的产物，它克服了两者的缺点，是一种较有前途的方法。

- 以上几种控制器的设计方法是不同的，但产生的微操作命令的功能是相同的，并且各个控制条件基本上也是一致的，都是由时序电路、操作码译码信号，以及被控部件的反馈信息有机配合而成的。
- 从功能上看，这几种控制器只是微操作信号发生器的结构和原理不同，而外部的输入条件和输出结果几乎完全相同。



微操作信号发生器

4.3.4 控制器的控制方式

- 计算机执行指令的过程实际上是执行一系列的微操作的过程。每一条指令都对应着一个微操作序列，这些微操作中有些可以同时执行，有些则必须按严格的时间关系执行。

- 控制器的控制方式需解决的问题：
- 如何在时间上对各种微操作信号加以控制。

- 常用的控制方式有同步控制、异步控制和联合控制。

1．同步控制方式

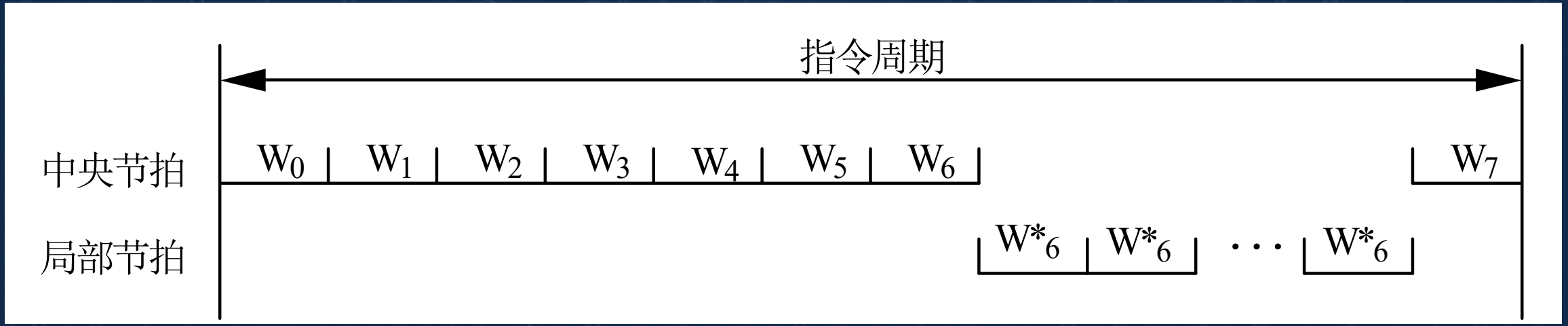
- 任何指令的运行或指令中各个微操作的执行，均由确定的具有统一基准时标的时序信号所控制。
- 即所有的操作均由统一的时钟控制，在标准的时间内完成。
- 在同步控制方式下，每个时序信号的结束就意味着安排完成的工作已经完成，随即开始执行后续的微操作或自动转向下条指令的运行。

- 典型的同步控制方式：
- 以微操作序列最长的指令和执行时间最长的微操作为标准，把一条指令执行过程划分为若干个相对独立的阶段（称为周期）或若干个时间区间（称为节拍），采用完全统一的周期（或节拍）控制各条指令的执行。
- 优点：时序关系简单，控制方便
- 缺点：浪费时间。
- 因为对比较简单的指令，将有很多节拍是不用的，处于等待。所以，在实际应用中都不采用这种典型的同步控制方式，而是采用某些折衷的方案。

(1) 采用中央控制与局部控制相结合的方法

- 中央控制：统一节拍的控制
- 根据大多数指令的微操作序列的情况，设置一个统一的节拍数，使之大多数指令均能在统一的节拍内完成。
- 局部控制：在延长节拍内的控制
- 对于少数在统一节拍内不能完成的指令，采用延长节拍或增加节拍数，使之在延长节拍内完成，执行完毕再返回中央控制。

- 例：设某计算机的指令通常用8个节拍完成，即有8个中央节拍 W_7～W_0，当某指令在8个节拍中不能完成时，就插入若干局部节拍 W_6^*，经过若干局部节拍 W_6^*后，再返回中央节拍 W_7。



(2) 采用不同的机器周期和延长节拍的方法

- 把一条指令执行过程划分为若干机器周期，如取指、取数、执行等周期。根据所执行指令的不同需要，选取不同的机器周期数。在节拍安排上，每个周期划分为固定的节拍，每个节拍都可根据需要延长一个节拍。
- 这种方法可以解决执行不同的指令所需时间不统一问题。
- 在Intel 8088 的指令执行过程中有读写周期、内部周期等，其中读写周期为4个节拍，但可以延长若干个节拍。

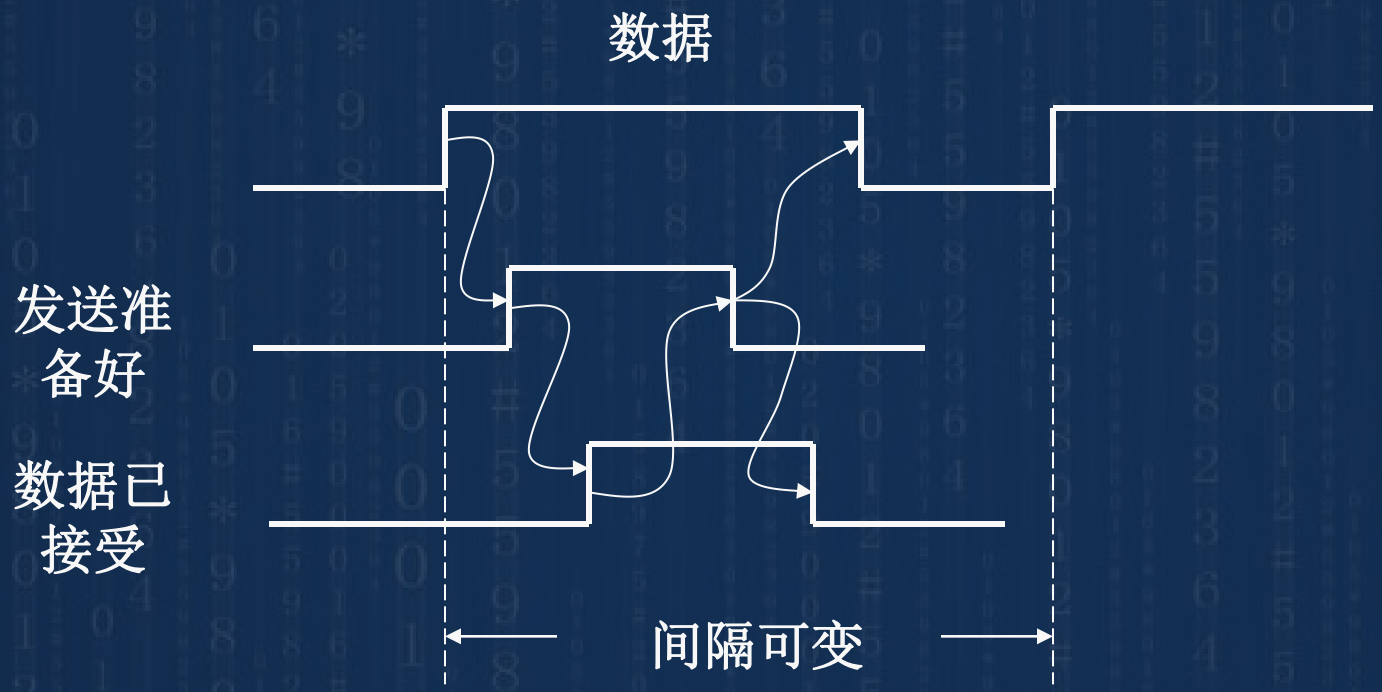
(3) 采用分散节拍的方法

- 分散节拍：运行不同指令时，需要多少节拍，时序部件就发生多少节拍。
- 这种方法可完全避免节拍轮空，是提高指令运行速度的有效方法，但这种方法使时序部件复杂化。同时还不能解决节拍内那些简单的微操作因等待所浪费的时间。

2．异步控制方式

- 没有统一的同步信号，采用问答方式进行时序协调，将前一操作的回答信号作为下一操作的启动信号。
- 异步控制方式不仅要区分不同指令对应的微操作序列的长短，而且要区分其中每个微操作的繁简，每条指令、每个微操作需要多少时间就占用多少时间。
- 这种方式不再有统一的周期、节拍，各个操作之间采用应答方式衔接，前一操作完成后给出回答信号，启动下一个操作。

- 这种方式可根据每条指令的操作的实际需要而分配时间，所以没有时间上的浪费，效率高。但设计复杂且费设备。



3．联合控制方式

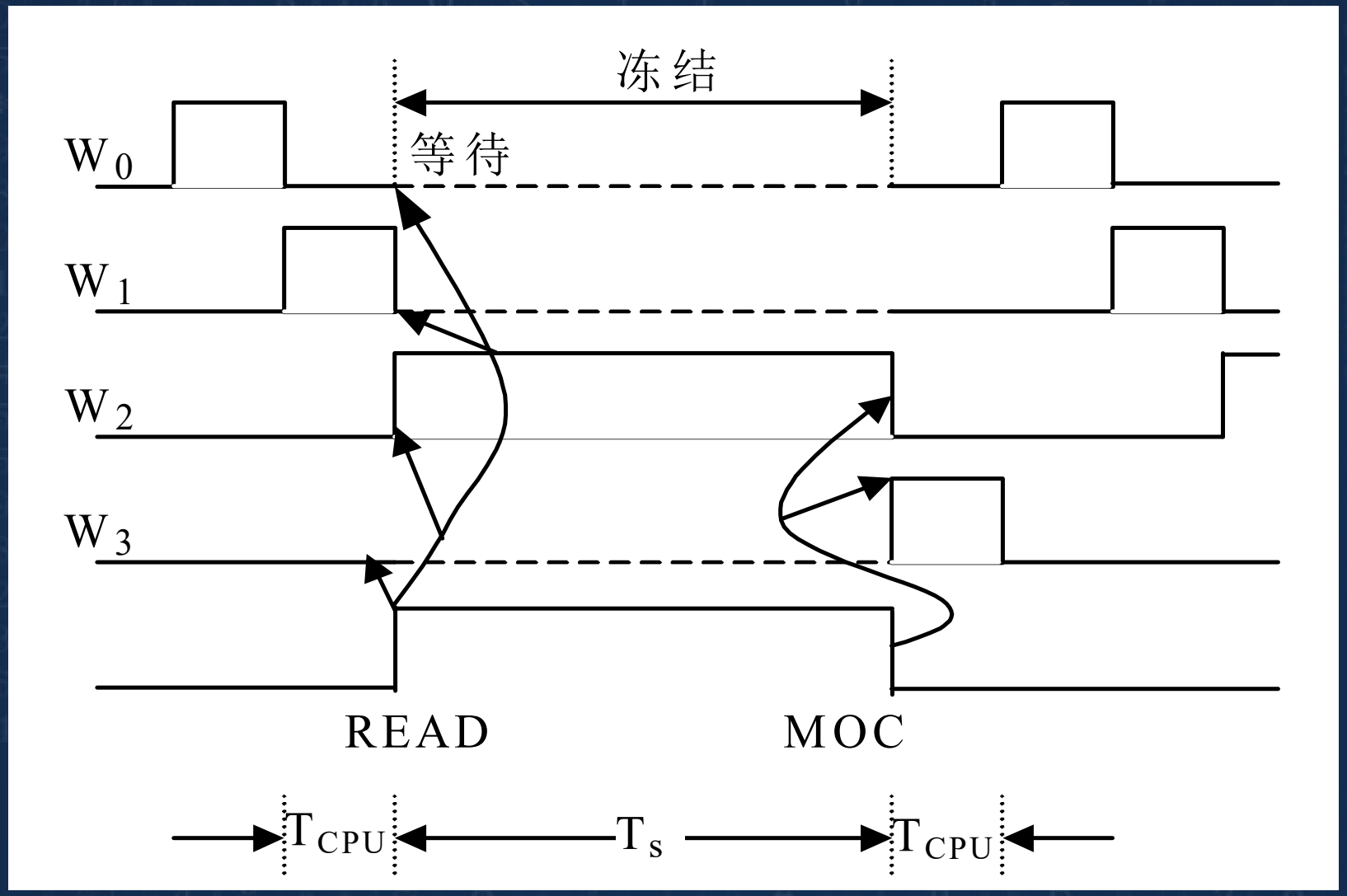
- 联合控制是将同步控制和异步控制相结合。

- 联合控制方式的设计思想：
- 在功能部件内部采用同步方式或以同步方式为主的控制方式；在功能部件之间采用异步方式。
- 通常对可以统一的微操作采用同步控制，对难以统一的微操作采用异步控制。

- 例如，在微、小型机中，CPU内部基本时序采用同步控制方式，当CPU通过总线与主存或其它外设交换数据时，转入异步控制。
- 当CPU访问外设时，只需给出起始信号，主存或外部设备即按自己的时序信号去安排操作，一旦操作结束，则向CPU发结束信号，以便CPU再安排它的后继工作。

同步与异步时序的衔接关系

- 当CPU要访主存时，在发读信号READ同时发“等待”信号，等待信号使时序由同步转入异步操作并冻结同步时序，使节拍间的相位关系不再发生变化，直到存储器按自己速度操作结束，并向CPU发回答信号MOC才解除对同步时序的冻结，机器回到同步时序按原时序关系继续运行。



- 实际上现代计算机中几乎没有完全采用同步或完全采用异步的控制方式，大多数都采用联合控制方式。

4.3.5 控制器的时序系统

- 时序系统是控制器的心脏，由它为指令的执行提供各种定时信号。通常，设计时序系统主要是针对同步控制方式的。

- 1．指令周期与机器周期
- 指令周期：从取指令、分析指令到执行完一条指令所需的全部时间。
- 由于各种指令的操作功能不同，繁简程度不同，因此各种指令的指令周期也不尽相同。

- 机器周期（CPU周期）：指令周期中的某一工作阶段所需的时间。在指令执行过程中，各机器周期相对独立。
- 一条指令的指令周期由若干个机器周期所组成，每个机器周期完成一个基本操作。所以机器周期也称为基本周期。
- 一般机器的CPU周期有取指周期、取数周期、执行周期，中断周期等。

- 每个机器周期设置一个周期状态触发器与之对应，机器运行于哪个周期，与其对应的周期状态触发器被置为“1”。显然，机器运行的任何时刻都只能建立一个周期状态，因此同一时刻只能有一个周期状态触发器被置为“1”。
- 不同工作周期所占的时间可以不等。由于CPU内部操作速度快，而CPU访存所花时间较长，所以许多计算机系统往往以主存周期为基础来规定CPU周期，以便二者工作协调配合。

2. 节拍

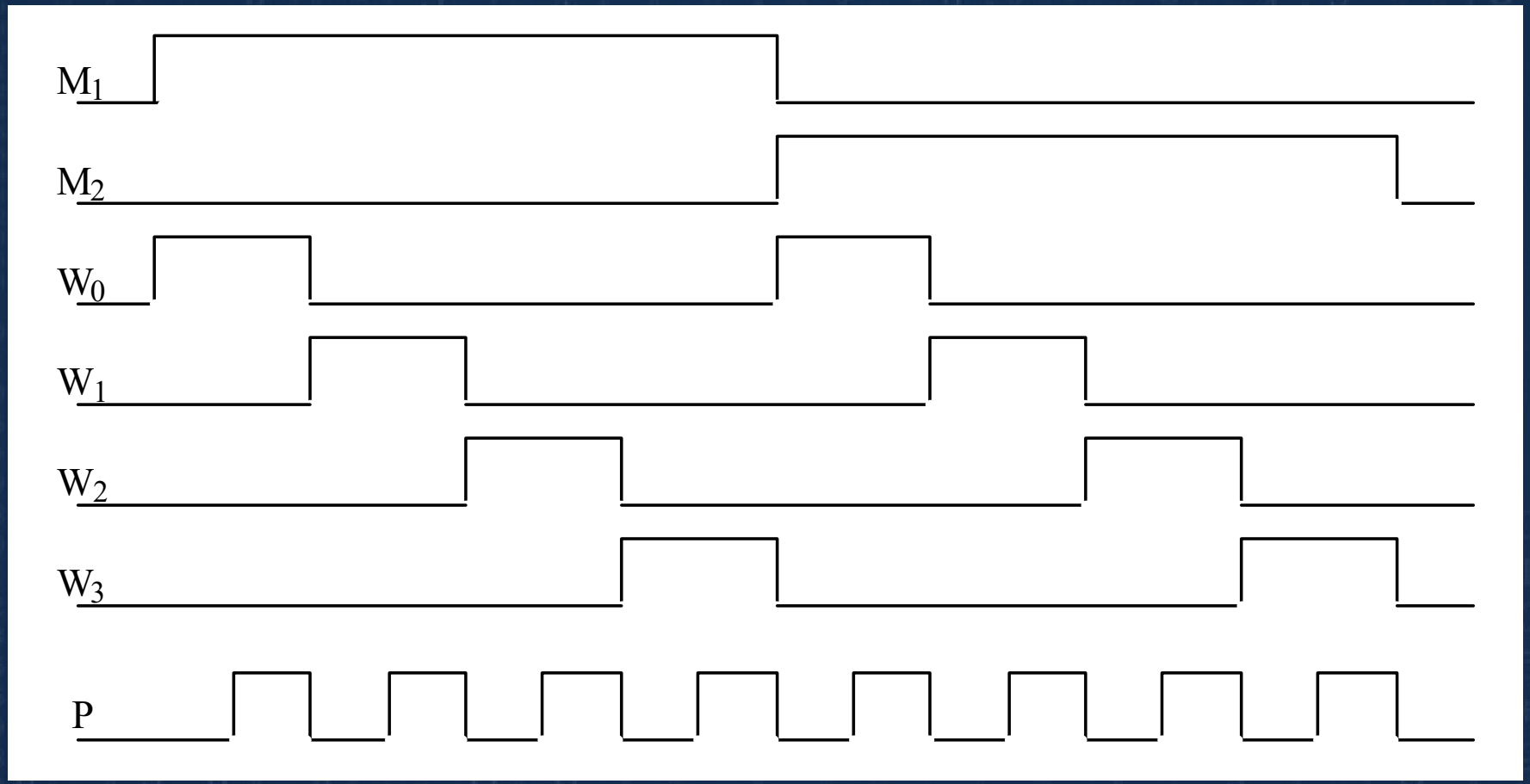
- 把一个机器周期等分成若干个时间区间，每一时间区间称为一个节拍。
- 一个节拍对应一个电位信号，控制一个或几个微操作的执行。
- 在一个机器周期内，要完成若干个微操作，这些微操作不但需要占用一定的时间，而且有一定的先后次序。因此，在同步控制方式中，基本的控制方法就是把一个机器周期等分成若干个节拍，每一个节拍完成一步基本操作，如一次传送、一次加减运算等。
- 一个节拍电位信号的宽度取决于CPU完成一个基本操作的时间。

3. 脉冲（定时脉冲）

- 节拍提供了一项基本操作所需的时间分段，但有的操作如打入寄存器，还需严格的定时脉冲，以确定在哪一时刻打入。节拍的切换，也需要严格的同步定时。所以在一个节拍内，有时还需要设置一个或几个工作脉冲，用于寄存器的复位和接收数据等。
- 脉冲：一个节拍内设置的一个或几个工作脉冲。

- 常见的设计是在每个节拍的末尾发一次工作脉冲，脉冲前沿可用来打入运算结果（或传送），脉冲后沿则实现周期的切换。
- 也有的计算机，在一个节拍中先后发出几个工作脉冲，有的脉冲位于节拍前端，可用作清除脉冲；有的脉冲位于中部，用作控制外围设备的输入/输出脉冲；有的脉冲位于尾部，前沿用作CPU内部的打入，后沿实现周期切换。

- 周期、节拍、脉冲构成了三级时序系统，它们之间关系如下图所示。图中包括两个机器周期M_1、M_2，每个周期包含四个节拍W_0～W_3，每个节拍内有一个脉冲P。



问题的回应

- **1.** 指令和数据在**CPU**中的区分主要靠“时序的阶段”（周期、节拍等）进行“界定”（或“区分”）；
- **2.** 指令和数据的“目的地”是由某电位（接收脉冲）启动某部件（寄存器、锁存器等）接收；
- **3.** 指令的功能也由微操作控制信号在某阶段启动有关功能部件完成。