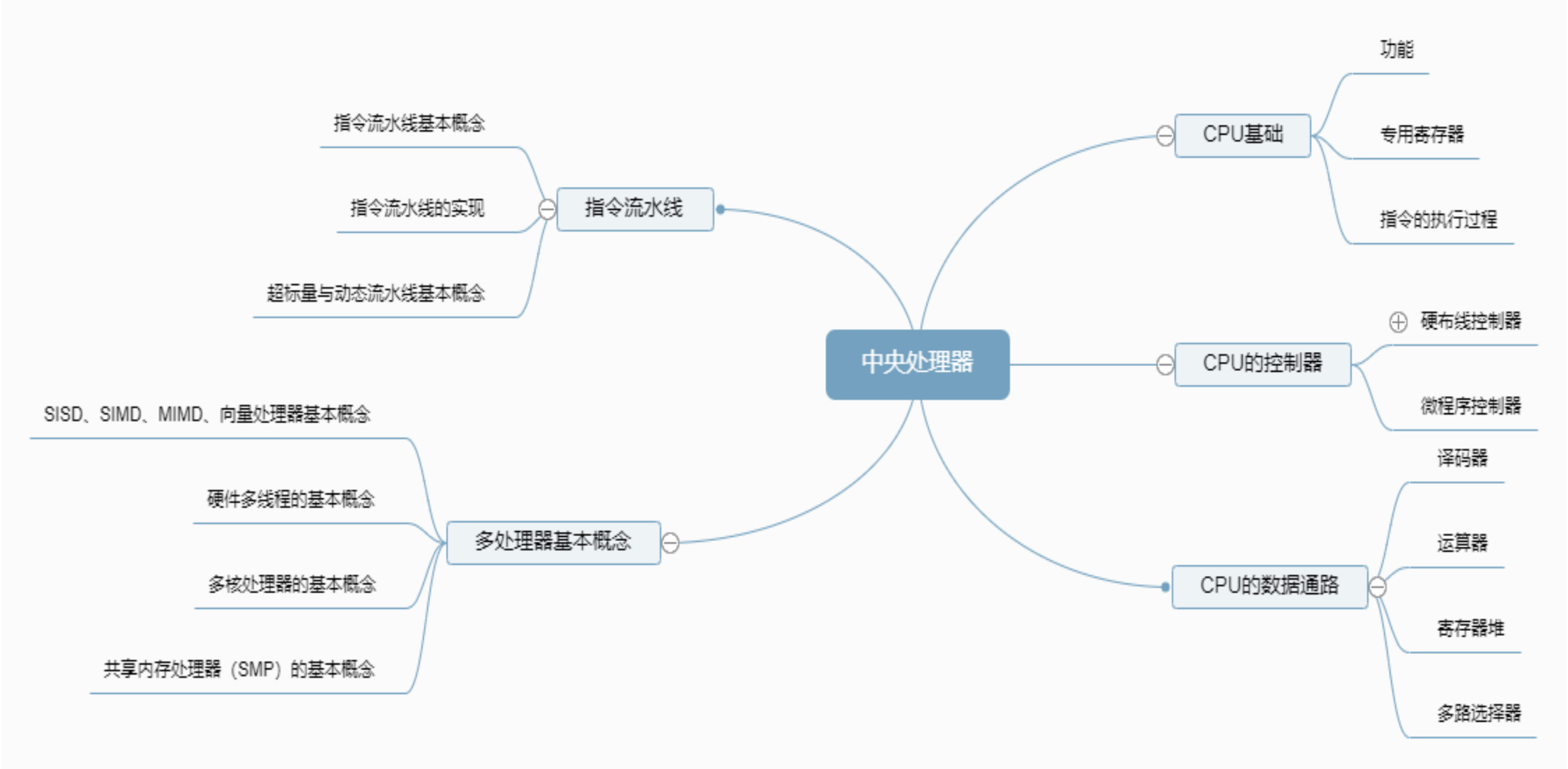
计算机组成与结构-中央处理器

人工智能专业

主讲教师：王 娟

中央处理器思维导图


中央处理器
CPU基础
功能
专用寄存器
指令的执行过程
CPU的控制器
硬布线控制器
微程序控制器
CPU的数据通路
译码器
运算器
寄存器堆
多路选择器
指令流水线
指令流水线基本概念
指令流水线的实现
超标量与动态流水线基本概念
多处理器基本概念
SISD、SIMD、MIMD、向量处理器基本概念
硬件多线程的基本概念
多核处理器的基本概念
共享内存处理器（SMP）的基本概念

中央处理器（CPU）是整个计算机的核心，它包括运算器和控制器。本章着重讨论CPU的功能和组成，控制器的工作原理和实现方法，微程序控制原理，基本控制单元的设计以及先进的CPU系统设计技术。

一．中央处理器的功能和组成

二．控制器的组成和实现方法

三．时序系统与控制方式

四．微程序控制原理

五．流水线基本原理

中央处理器的功能和组成

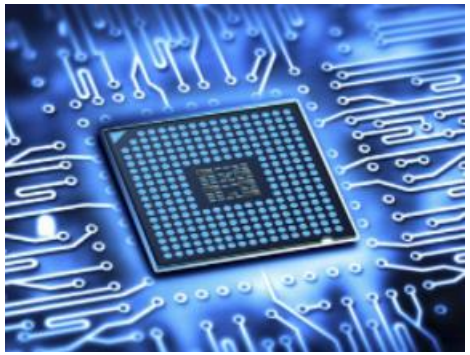
CPU的主要功能：

控制指令的执行顺序；

产生控制信号控制部件工作；

控制各步操作的时序；

数据处理：算术和逻辑运算；



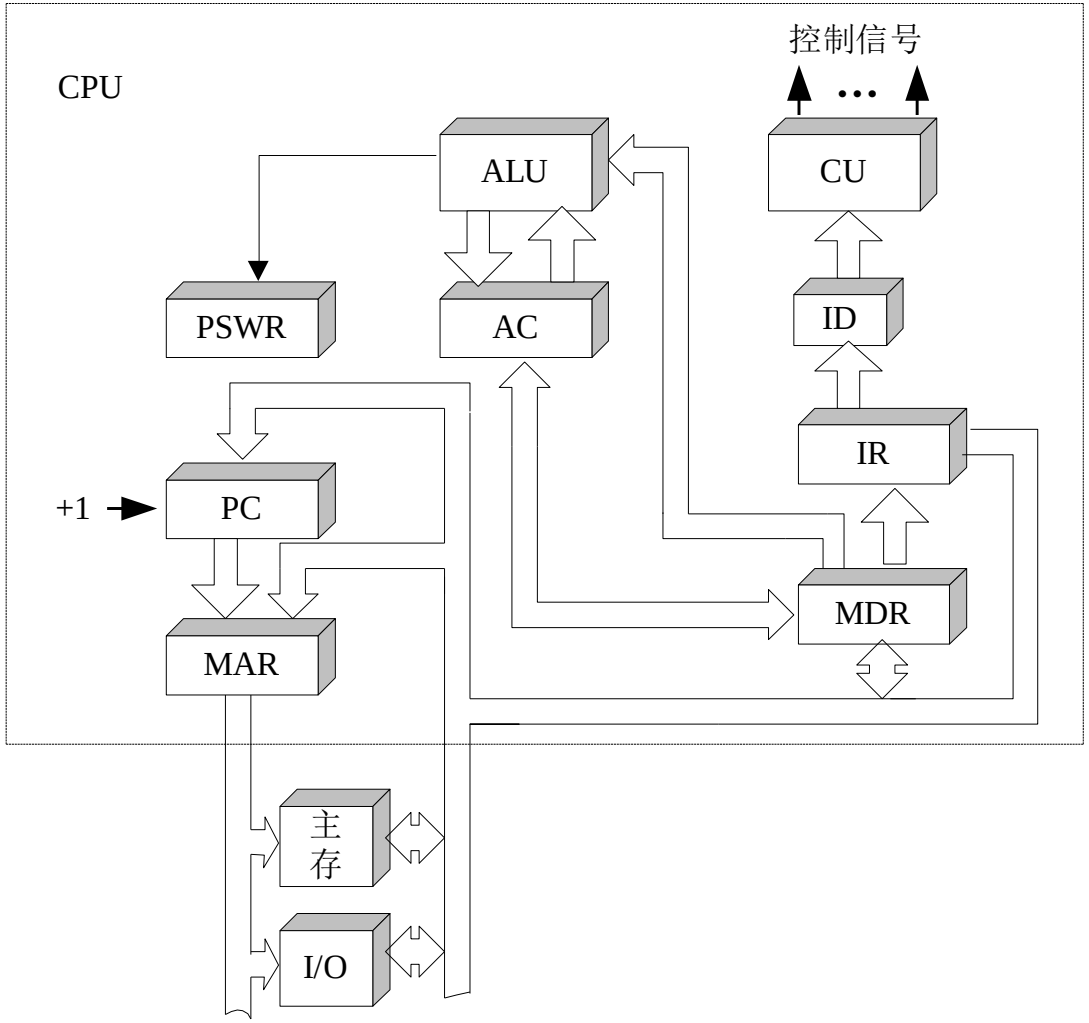
CPU的组成从硬件设计的角度：

CPU 包含 数据通路(执行部件) 和 控制器（控制部件）两部分。

数据通路指：指令执行过程中，数据所经过的路径，包括路径中的部件。它是指令的**执行部件**，包含寄存器、运算部件等。

控制器：对指令进行译码，生成指令对应的控制信号，控制数据通路的动作。它能对执行部件发出控制信号，因此是**指令的控制部件**。如译码器、时序控制等。

CPU模型



CPU中的主要寄存器是用来暂时保存在运算和控制过程中的中间结果、最终结果以及控制、状态信息的，它又可分为通用寄存器和专用寄存器两种。

1.通用寄存器

通用寄存器可用来存放原始数据和运算结果，有的还可以作为变址寄存器、计数器、地址指针等。通用寄存器一般可以由CPU直接访问。

2.专用寄存器

专用寄存器专门用来完成某一种特殊功能的寄存器。CPU中至少要有五个专用的寄存器。它们是：程序计数器（PC）、指令寄存器（IR）、存储器地址寄存器（MAR）、存储器数据寄存器（MDR）、状态标志寄存器（PSWR）

状态标志寄存器（PSWR）

包含机器状态标志位或者控制信息位两部分。**标志位**，反映CPU的当前状态。一般指令执行时，根据情况自动设置这些特征位，作为后续操作的判断依据，通常有5类:

进位 CF	溢出 OF	零值 ZF	符号 SF	奇偶 PF	…

自动设置(具备该特征，就设置该标志位=1)

控制位：程序设计者可指定，以决定程序的调试、对中断的响应、程序的运行模式等。

跟踪位 TF	允许中断 IF	I/O特权 P	…

一．中央处理器的功能和组成

二．控制器的组成和实现方法

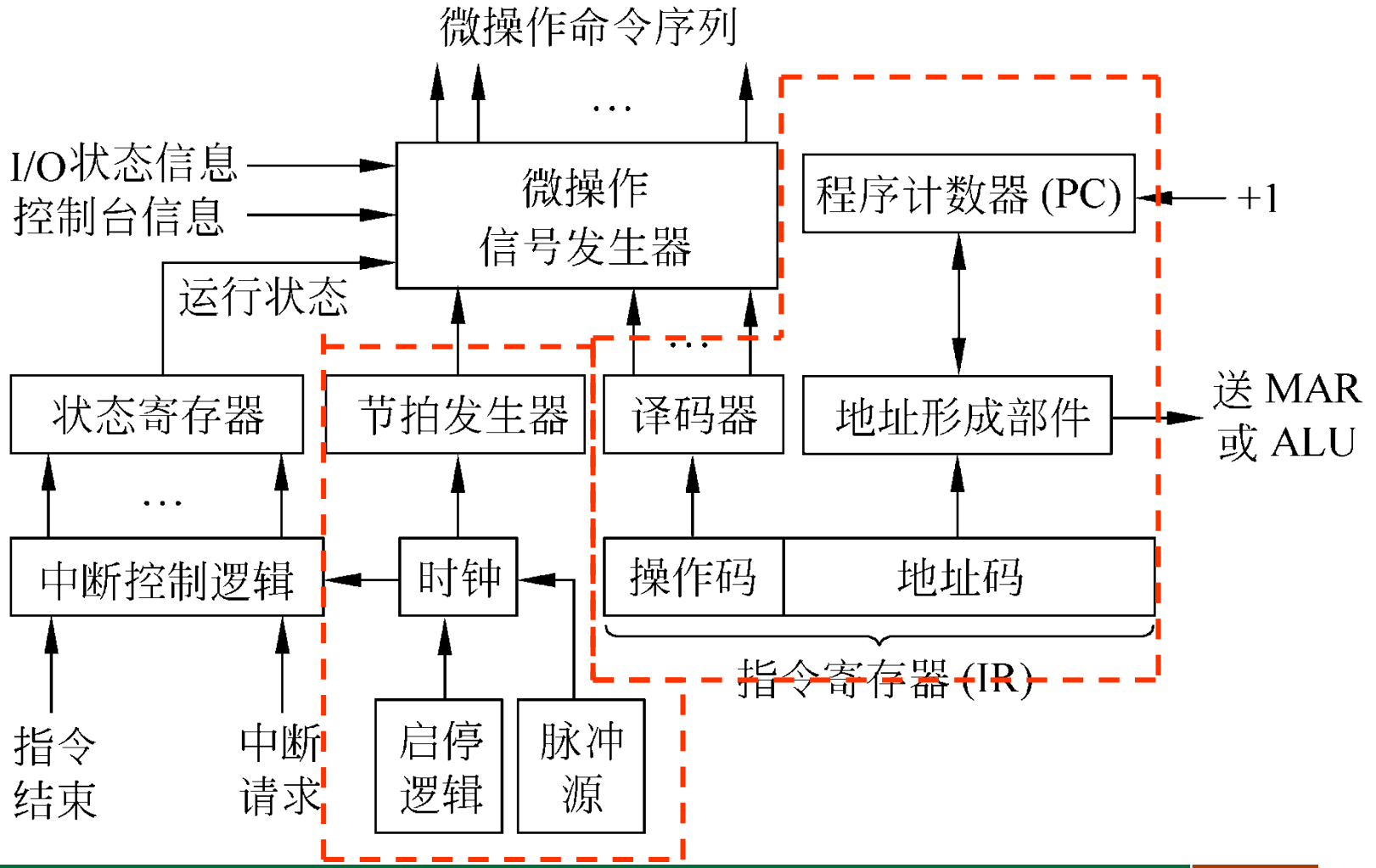
三．时序系统与控制方式

四．微程序控制原理

五．流水线基本原理

控制器的组成和实现方法

控制器是计算机系统的指挥中心，它把运算器、存储器、输入/输出设备等部件组成一个有机的整体，然后根据指令的要求指挥全机的工作。



控制器的组成和实现方法

1.指令部件

指令部件的主要任务是完成取指令并分析指令。指令部件包括：

(1)程序计数器（PC）

(2)指令寄存器（IR）

(3)指令译码器（ID）：指令译码器又称操作码译码器或指令功能分析解释器。暂存在指令寄存器中的指令只有在其操作码部分经过译码之后才能识别出这是一条什么样的指令，并产生相应的控制信号提供给微操作信号发生器。

(4)地址形成部件

根据指令的不同寻址方式，用来形成操作数的有效地址，在微、小型机中，一般不设专门的地址形成部件，而是利用运算器来进行有效地址的计算。

控制器的组成和实现方法

2.时序部件

时序部件能产生一定的时序信号，以保证机器的各功能部件有节奏地进行信息传送、加工及信息存储。时序部件包括：

(1)脉冲源

用来产生具有一定频率和宽度的时钟脉冲信号，为整个机器提供基准信号。

(2)启停控制逻辑

启停控制逻辑的作用是根据计算机的需要，可靠地开放或封锁脉冲，控制时序信号的发生或停止，实现对整个机器的正确启动或停止。启停控制逻辑保证启动时输出的第一个脉冲和停止时输出的最后一个脉冲都是完整的脉冲。

(3)节拍信号发生器

节拍信号发生器又称脉冲分配器。脉冲源产生的脉冲信号，经过节拍信号发生器后产生出各个机器周期中的节拍信号，用以控制计算机完成每一步微操作。

控制器的组成和实现方法

3.微操作信号发生器

一条指令的取出和执行可以分解成很多最基本的操作，这种最基本的不可再分割的操作称为微操作。微操作信号发生器也称为控制单元（CU）。不同的机器指令具有不同的微操作序列。

4.中断控制逻辑

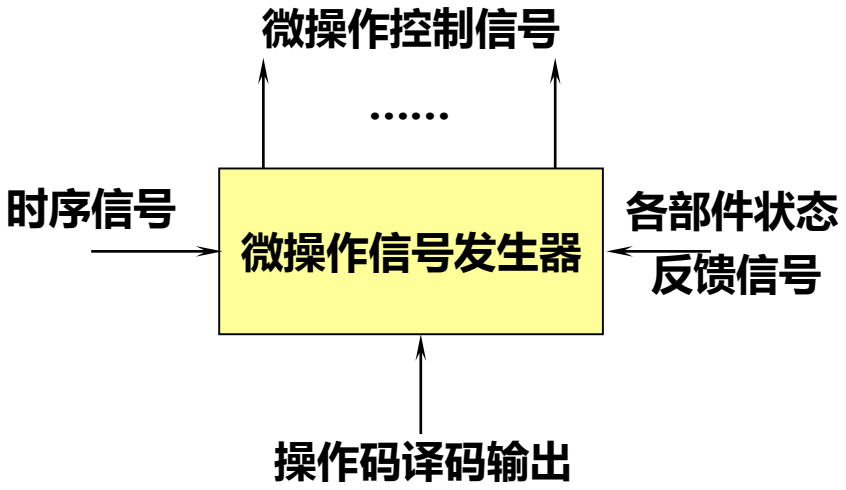
中断控制逻辑是用来控制中断处理的硬件逻辑。

控制器的硬件实现方法

控制器的输入是机器指令代码，输出是微操作控制信号，因此微操作信号发生器是控制器的核心。根据产生微操作控制信号的方式不同，控制器可分为3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，而控制器中的其它部分基本上是大同小异的。

常见的微操作信号发生器实现方式：

- 组合逻辑型
- 存储逻辑型（微程序方式）
- 组合逻辑与存储逻辑结合型



控制器的硬件实现方法

1.组合逻辑型

这种控制器称为常规控制器或硬布线控制器，它是采用组合逻辑技术来实现的，其微操作序列形成部件是由门电路组成的复杂树形网络。

组合逻辑控制器的最大优点是速度快，但是微操作信号发生器的结构不规整，使得设计、调试、维修较困难，难以实现设计自动化。一旦微操作信号发生器构成之后，要想增加新的控制功能是不可能的。

控制器的硬件实现方法

2.存储逻辑型

这种控制器称为微程序控制器，它是采用存储逻辑来实现的，也就是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并存入一个专门的存储器（控制存储器）中，微操作控制信号由微指令产生。

微程序控制器的设计思想和组合逻辑设计思想截然不同。它具有设计规整、调试、维修以及更改、扩充指令方便的优点，易于实现自动化设计，已成为当前控制器的主流。但是，由于它增加了一级控制存储器，所以指令执行速度比组合逻辑控制器慢。

3.组合逻辑和存储逻辑结合型

这种控制器称为PLA控制器，它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。

一．中央处理器的功能和组成

二．控制器的组成和实现方法

三．时序系统与控制方式

四．微程序控制原理

五．流水线基本原理

时序系统概念

时序系统是控制器的心脏，其功能是为指令的执行提供各种定时信号。

1.指令周期和机器周期

指令周期是指取指令、分析指令到执行完该指令所需的全部时间。由于各种指令的操作功能不同，有的简单，有的复杂，因此各种指令的指令周期不尽相同。

机器周期通常又称CPU周期，通常把一条指令划分为若干个机器周　　指令周期＝i×机器周期

通常，每个机器周期都有一个与之对应的周期状态触发器。机器运行在不同的机器周期，其对应的周期状态触发器被置“1”，显然，在机器运行的任何时刻只能建立一个周期状态，因此，有一个且仅有一个触发器被置“1”。每个机器周期完成一个基本操作。

时序系统概念

2.节拍

在一个机器周期内，要完成若干个微操作。这些微操作有的可以同时执行，有的需要按先后次序串行执行。因而需要把一个机器周期分为若干个相等的时间段，每一个时间段对应一个电位信号，称为节拍电位信号。节拍的宽度取决于CPU完成一次基本操作的时间。

由于不同的机器周期内需要完成的微操作内容和个数是不同的，因此，不同机器周期内所需要的节拍数也不相同。节拍的选取一般有以下几种方法：

(1)统一节拍法：以最复杂的机器周期为准定出节拍数，每一节拍时间的长短也以最繁的微操作作为标准。这种方法采用统一的、具有相等时间间隔和相同数目的节拍，使得所有的机器周期长度都是相等的，因此称为**定长CPU周期**。

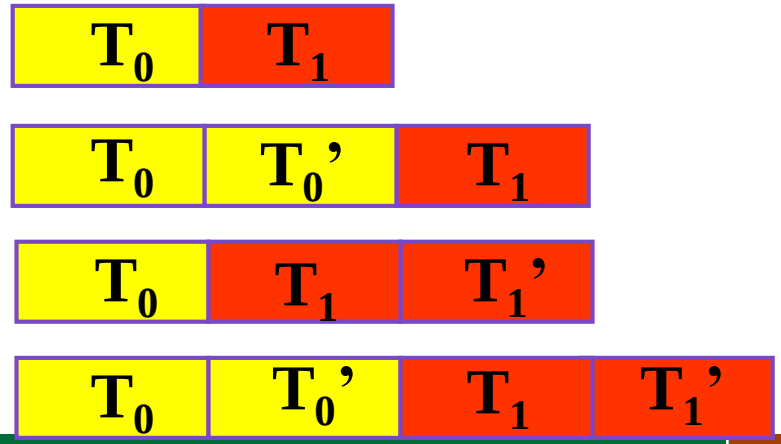
时序系统概念

(2)分散节拍法

按照机器周期的实际需要安排节拍数，需要多少节拍，就发出多少节拍，这样可以避免浪费，提高时间利用率。由于各机器周期长度不同，又称为**不定长CPU周期**。

(3)延长节拍法

在照顾多数机器周期要求的情况下，选取适当的节拍数，作为基本节拍，如果在某个机器周期内统一的节拍数无法完成该周期的全部微操作，则可以延长节拍。

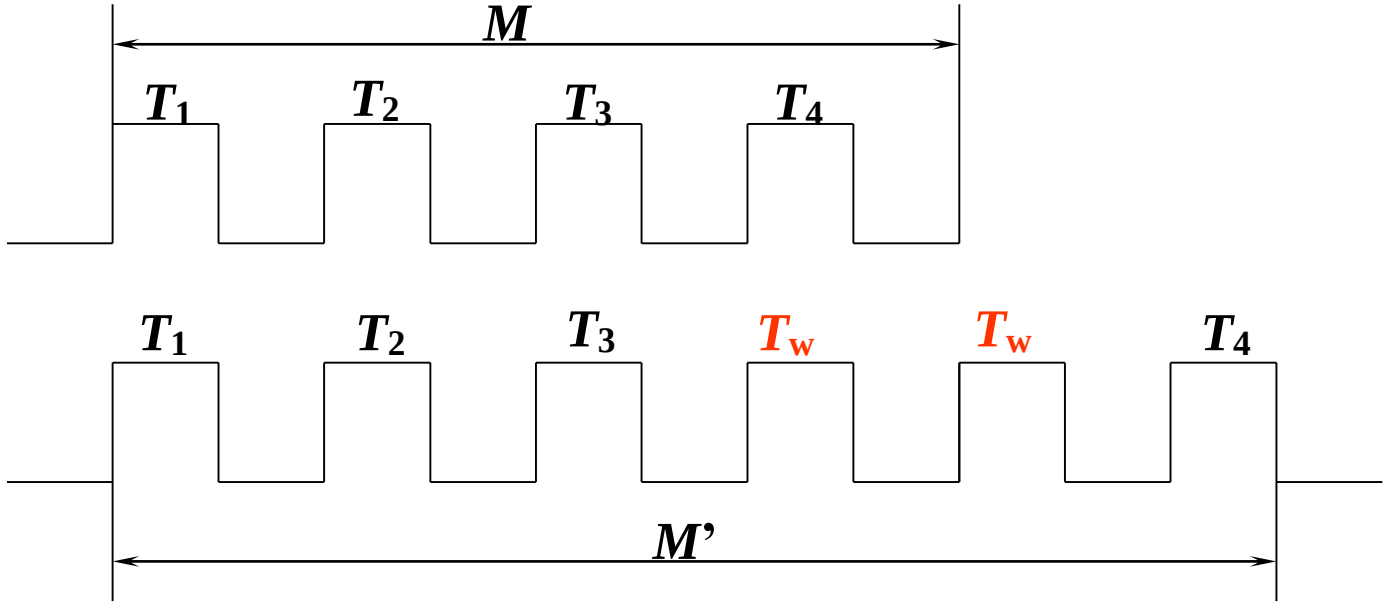


时序系统概念

(4)时钟周期插入

在一些微型机中，时序信号中不设置节拍，而直接使用时钟周期信号。一个机器周期中含有若干个时钟周期，时钟周期的数目取决于机器周期内完成微操作的多少及相应功能部件的速度。一个机器周期的基本时钟周期数确定之后，还可以不断插入等待时钟周期。

8086总线周期



时序系统概念

3.工作脉冲

在节拍中执行的有些操作需要同步定时脉冲，为此，在一个节拍内常常设置一个或几个工作脉冲，作为各种同步脉冲的来源。工作脉冲的宽度只占节拍电位宽度的1/n，并处于节拍的末尾，只要能保证所有触发器都可靠、稳定地翻转就可以了。

在只设置机器周期和时钟周期的微型机中，一般不再设置工作脉冲，因为时钟周期既可以作为电位信号，其前后沿又可以作为脉冲触发信号。

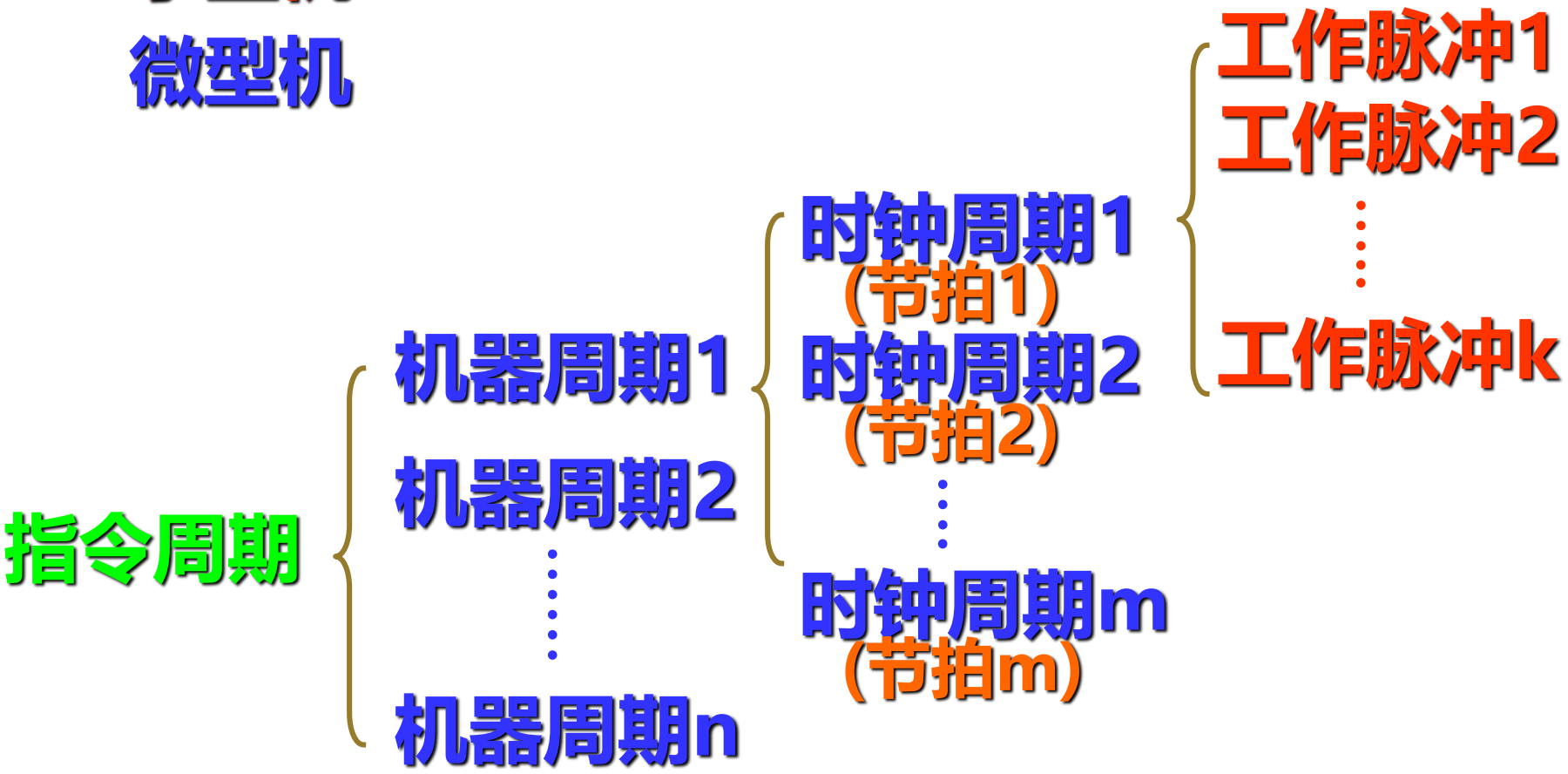
4.多级时序系统

小型机中常采用机器周期、节拍、工作脉冲三级时序系统。每个机器周期M中包括若干节拍，每个节拍内有一个脉冲。在机器周期间、节拍电位间、工作脉冲间既不允许有重叠交叉，也不允许有空隙，应该是一个接一个的准确连接。

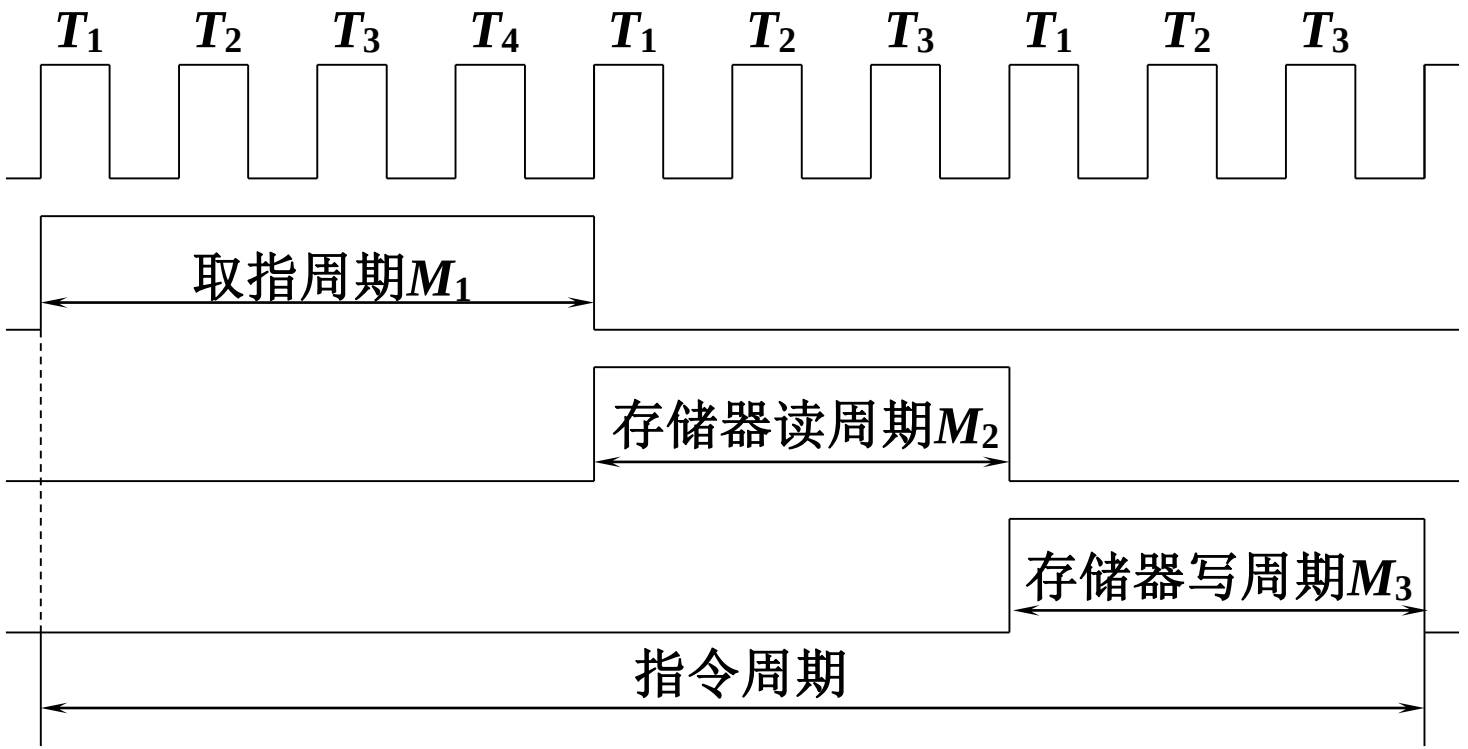
时序系统概念

小型机

微型机



时钟周期时序系统



CPU各部件之间的控制方式

1.同步控制方式

同步控制方式即固定时序控制方式，各项操作都由统一的时序信号控制，在每个机器周期中产生统一数目的节拍电位和工作脉冲。由于不同的指令，操作时间长短不一致，同步控制方式应以最复杂指令的操作时间作为统一的时间间隔标准。

这种控制方式设计简单，容易实现，但是对于许多简单指令来说会有较多的空闲时间，造成较大数量的时间浪费，从而影响了指令的执行速度。

在同步控制方式中，各指令所需的时序由控制器统一发出，所有微操作都与时钟同步，所以又称为**集中控制方式或中央控制方式**。

CPU各部件之间的控制方式

2.异步控制方式

异步控制方式即可变时序控制方式。各项操作不采用统一的时序信号控制，而根据指令或部件的具体情况决定，需要多少时间，就占用多少时间。异步控制采用不同时序，没有时间上的浪费，因而提高了机器的效率，但是控制比较复杂。

由于这种控制方式没有统一的时钟，而是由各功能部件本身产生各自的时序信号自我控制，故又称为**分散控制方式或局部控制方式**。

3.联合控制方式

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，大多数是采用联合控制方式。通常的设计思想是：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。

指令执行的基本过程

一条指令执行过程可以分为三个阶段：**取指令阶段、分析取数阶段和执行阶段**。

1.取指令阶段

取指令阶段完成的任务是将现行指令从主存中取出来并送至指令寄存器中去，所有指令均有，称为**公共操作**。具体的操作为：

①将程序计数器（PC）中的内容送至存储器地址寄存器（MAR），并送地址总线（AB）。(PC)→MAR

②由控制单元（CU）经控制总线（CB）向主存发读命令。 Read

③从主存中取出的指令通过数据总线（DB）送到存储器数据寄存器(MDR)。 M(MAR) →MDR

④将MDR的内容送至指令寄存器（IR）中。 (MDR)→IR

⑤将PC的内容递增，为取下一条指令做好准备。 (PC) + 1→PC

指令的执行

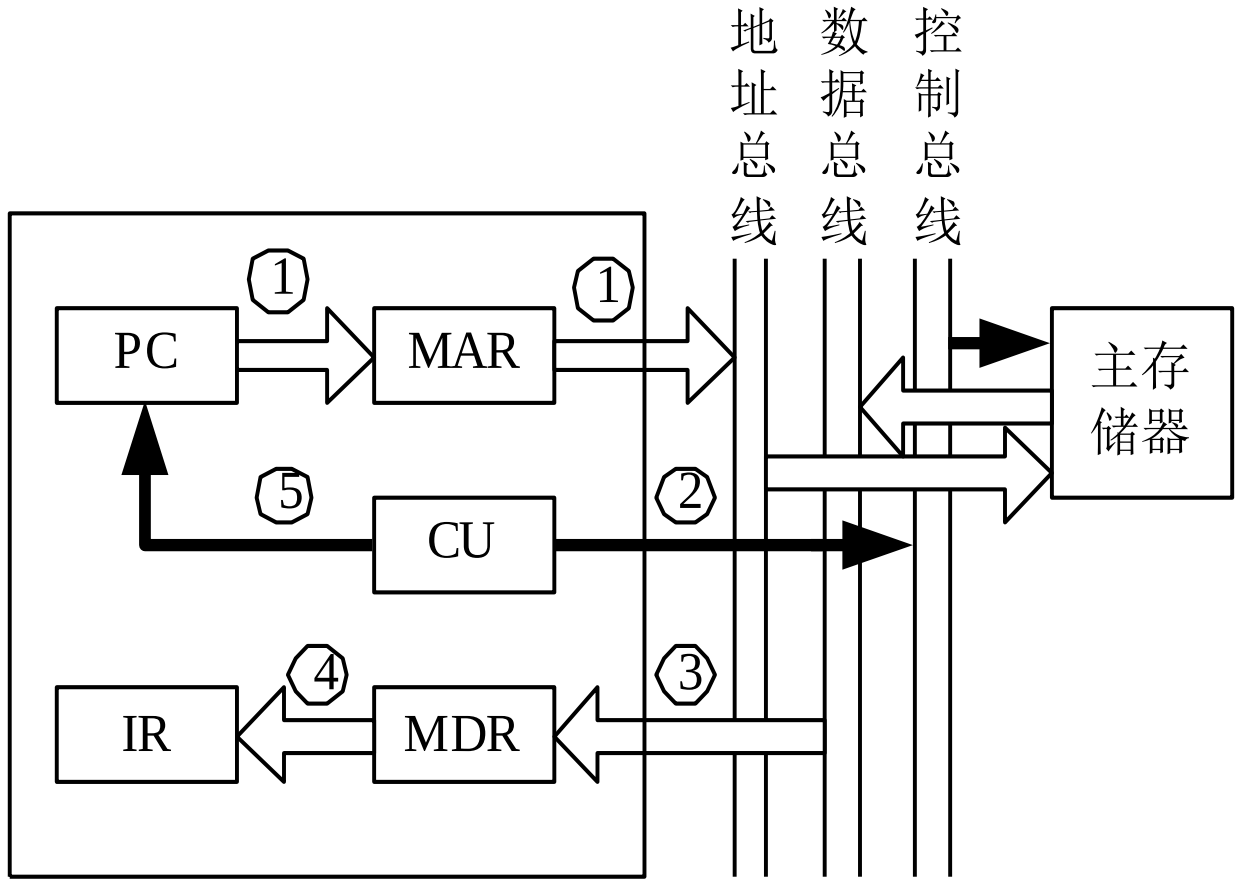
2.分析取数阶段

取出指令后，机器立即进入分析指令阶段，指令译码器ID可识别和区分不同的指令类型及各种获取操作数的方法。由于各条指令功能不同，寻址方式也不同，所以分析取数阶段的操作是各不相同的。

3.执行阶段

执行阶段完成指令规定的各种操作，形成稳定的运算结果，并将其存储起来。

计算机的基本工作过程可以概括成为取指令、分析取数、执行指令，然后再取下一条指令，……。如此周而复始，直至遇到停机指令或外来的干预为止。



指令的微操作序列

控制器在实现一条指令的功能时，总要把每条指令分解成为一系列时间上先后有序的最基本、最简单的微操作，即微操作序列。微操作序列是与CPU的内部数据通路密切相关的，**不同的数据通路就有不同的微操作序列**。

1.加法指令ADD @R_0 ,R_1

这条指令完成的功能是把R_0的内容作为地址送到主存以取得第一操作数，再与R_1的内容相加，最后将结果送回主存中。即实现：

$((R_0)) + (R_1) \rightarrow (R_0)$

指令的微操作序列

字母加下标in表示该部件的接收控制信号，实际上就是该部件的输入开门信号；字母加下标out表示该部件的发送控制信号，实际上就是该部件的输出开门信号。

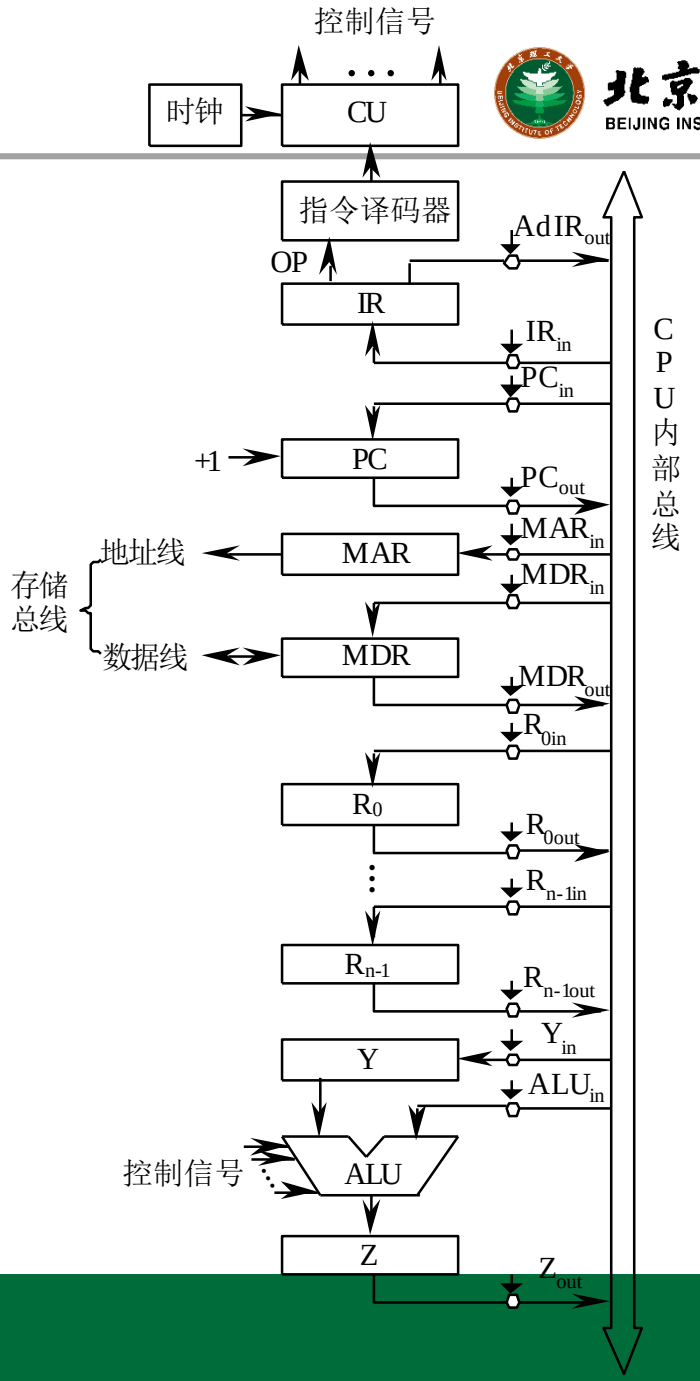
取指令阶段的公共操作，它完成的任务为：

① PC_{out}和MAR_{in}有效，(PC)→MAR

②通过控制总线发Read命令。

③ MDR_{out}和Ir_{in}有效M(MAR)→MDR→IR

④(PC) + 1→PC



指令的微操作序列

(2)取数周期

取数周期要完成取操作数的任务，被加数在主存中，加数已放在通用寄存器R_1中。

① R_{0out}和MAR_{in}有效，完成将被加数地址送至MAR的操作，记作$(R_0)\rightarrow MAR$；

② 向主存发读命令，记作Read；

③ 存储器通过数据总线将MAR所指单元的内容（数据）送至MDR，同时MDR_{out}和Y_{in}有效，记作$M(MAR)\rightarrow MDR\rightarrow Y$；

指令的微操作序列

(3)执行周期

执行周期完成加法运算的任务，并将结果写回主存。

① R_{1out}和ALU_{in}有效，同时CU向ALU发“ADD”控制信号，使R_1的内容和Y的内容相加，结果送寄存器Z中，记作$(R_1)+Y\rightarrow Z$；

② Z_{out}和MDR_{in}有效，将运算结果送MDR，记作$(Z)\rightarrow MDR$。

③ 向主存发写命令，记作Write。

2转移指令JC A

这是一条条件转移指令，若上次运算结果有进位（C=1），就转移；若上次运算结果无进位（C=0），就顺序执行下一条指令。设A为位移量，转移地址等于PC的内容加位移量。尝试自行完成相应的微操作序列。

一．中央处理器的功能和组成
二．控制器的组成和实现方法
三．时序系统与控制方式
四．微程序控制原理
五．流水线基本原理

微程序控制原理

微程序设计技术的实质是将程序设计技术和存储技术相结合，即用程序设计的思想方法来组织操作控制逻辑，将微操作控制信号按一定规则进行信息编码（代码化），形成控制字（微指令），再把这些微指令按时间先后排列起来，存放在一个只读存储器中。

每一条机器指令对应一段“程序”，该“程序”被存放在一个只读的控制存储器中，因为每段“程序”的执行结果是实现了一条机器指令的功能，所以我们将这些“程序”称为指令的微程序。

基本术语

(1)微命令和微操作

一条机器指令可以分解成一个微操作序列，这些微操作是计算机中最基本的、不可再分解的操作。微命令是控制计算机各部件完成某个基本微操作的命令。

微命令和微操作是一一对应的。**微命令是微操作的控制信号，微操作是微命令的操作过程。**

微命令有兼容性和互斥性之分，**兼容性微命令是指那些可以同时产生，共同完成某一些微操作的微命令；而互斥性微命令是指在机器中不允许同时出现的微命令。**兼容和互斥都是相对的，一个微命令可以和一些微命令兼容，和另一些微命令互斥。对于单独一个微命令，谈论其兼容和互斥都是没有意义的。

(2)微指令、微地址

微指令是指控制存储器中的一个单元的内容，即控制字，它是若干个微命令的集合。存放控制字的控制存储器的单元地址就称为微地址。

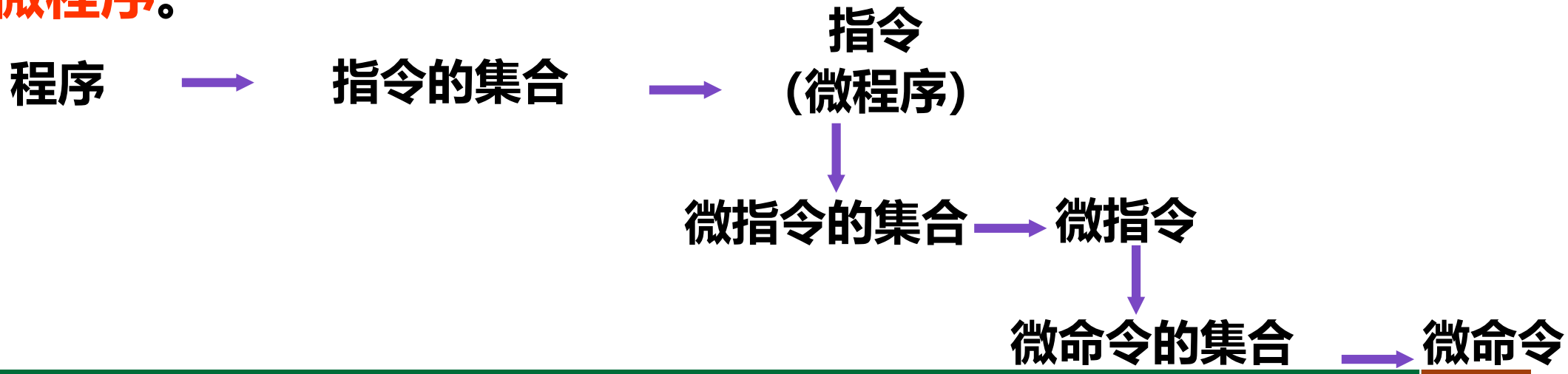
微指令有垂直型和水平型之分，垂直型微指令接近于机器指令的格式，每条微指令只能完成一个基本操作。水平型微指令则具有良好的并行性，每条微指令可以完成较多的基本操作。

(3)微周期

从控制存储器中读取一条微指令并执行相应的微命令所需的全部时间称为微周期。

(4)微程序

一系列微指令的有序集合就是微程序。一条机器指令对应于一段微程序。



微程序控制原理

微程序和程序是两个不同的概念。微程序是由微指令组成的，它用于描述机器指令，实际上是机器指令的实时解释器，它是由计算机的设计者事先编制好并存放在控制存储器中的。对于程序员来说，计算机系统中微程序一级的结构和功能是透明的。而程序则最终由机器指令组成，它是由软件设计人员事先编制好并存放在主存或辅存中的。

微程序控制的计算机涉及到两个层次：一个是机器语言或汇编语言程序员所看到的传统机器层，包括：**机器指令、工作程序、主存储器**；另一个是机器设计者看到的微程序层，包括：**微指令、微程序和控制存储器**。

一条微指令通常至少包含两大部分信息：

① 操作控制字段，又称微操作码字段，用以产生某一步操作所需的各微操作控制信号。

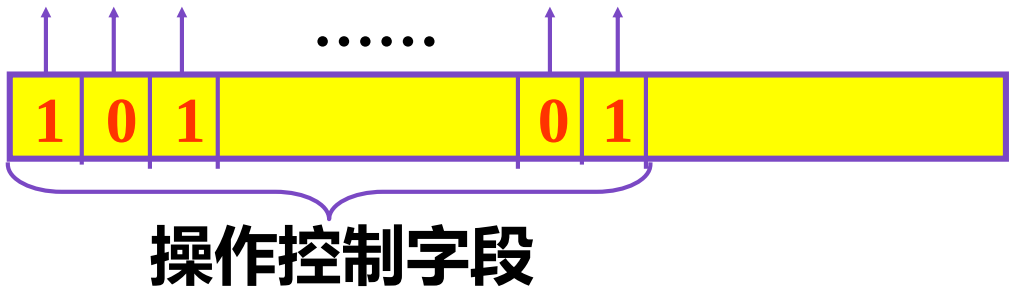
② 顺序控制字段，又称微地址码字段，用以控制产生下一条要执行的微指令地址。

微操作码	微地址码

微指令编码法

微指令编码法指的就是操作控制字段的编码方法。各类计算机的微指令编码法不同。

1.直接控制法（不译码法）



操作控制字段中的各位分别可以直接控制计算机，不需要进行译码。操作控制字段的每一个独立的二进制位代表一个微命令，该位为“1”表示这个微命令有效，为“0”表示这个微命令无效。每个微命令对应并控制数据通路中的一个微操作。

这种方法结构简单，并行性强，操作速度快，但是微指令字太长，若微命令的总数为N个，则微指令字的操作控制字段就要有N位。另外，在N个微命令中，有许多是互斥的，不允许并行操作，将它们安排在一条微指令中是毫无意义的，只会使信息的利用率下降。

2.最短编码法

最短编码使得微指令字最短。这种方法将所有的微命令统一编码，每条微指令只定义一个微命令。若微命令的总数为N，操作控制字段的长度为L，则：

$$L \geq \log_2 N$$

最短编码法的微指令字长最短，但要通过一个微命令译码器译码以后才能得到需要的微命令。微命令数目越多，译码器就越复杂。这种方法同一时刻只能产生一个微命令，不能充分利用机器硬件所具有的并行性，使得机器指令对应的微程序变得很长，而且对于某些要求在同一时刻同时动作的组合性微操作将无法实现。

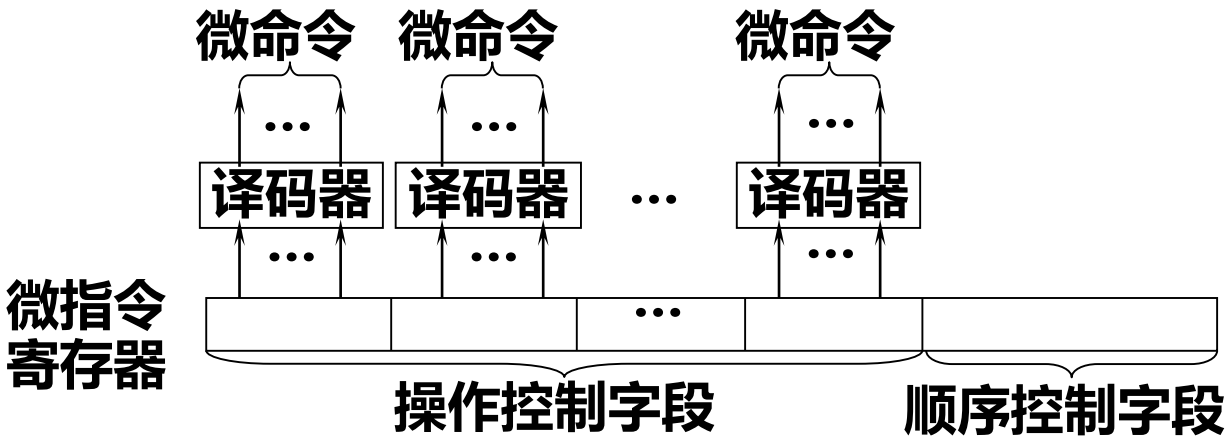
3.字段编码法

前两种编码法的一个折衷的方法，既具有两者的优点，又克服了它们的缺点。这种方法将操作控制字段分为若干个小段，每段内采用最短编码法，段与段之间采用直接控制法。

- 字段直接编码法
- 字段间接编码法

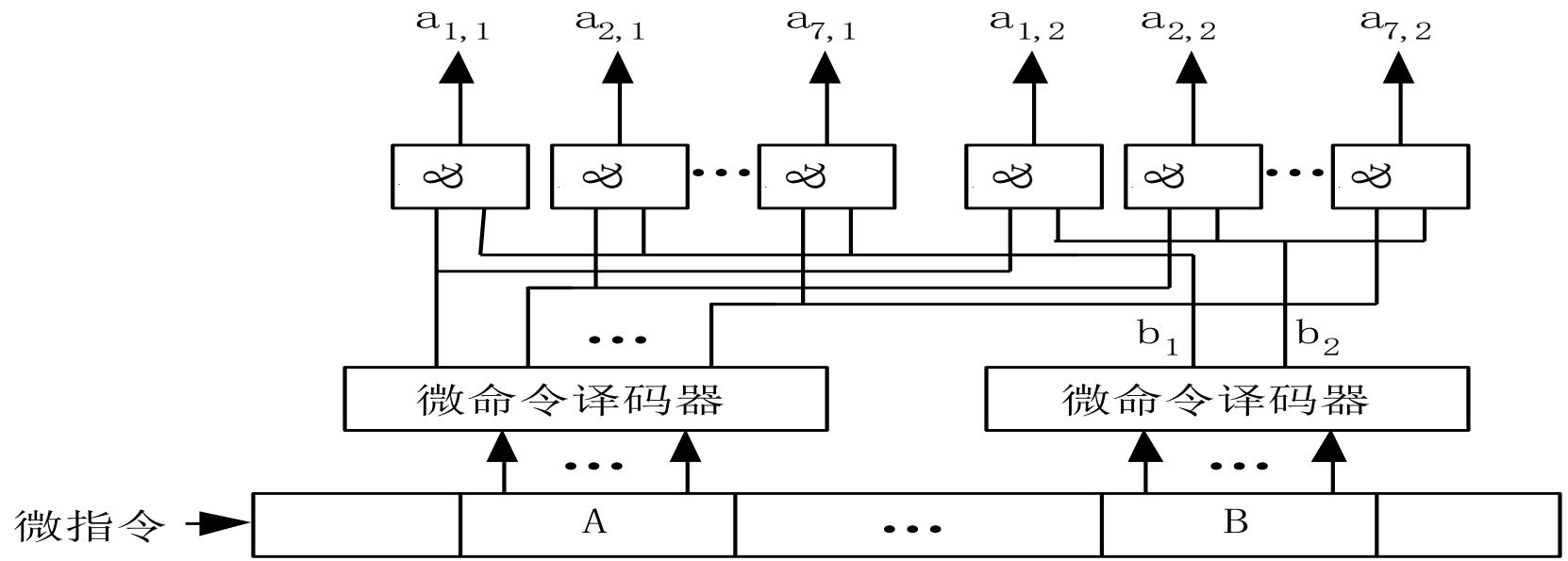
(1)字段直接编码法

各字段都可以独立地定义本字段的微命令，而和其他字段无关，因此又称为显式编码或单重定义编码方法。这种方法缩短了微指令字，因此得到了广泛的应用。



(2)字段间接编码法

字段间接编码法是在字段直接编码法的基础上，用来进一步缩短微指令字长的方法。间接编码的含义是，一个字段的某些编码不能独立地定义某些微命令，而需要与其他字段的编码来联合定义，因此又称为隐式编码或多重定义编码方法。



字段编码法中操作控制字段的分段原则：

① 把互斥性的微命令分在同一段内，兼容性的微命令分在不同段内。这样不仅有助于提高信息的利用率，缩短微指令字长，而且有助于充分利用硬件所具有的并行性，加快执行的速度。

② 应与数据通路结构相适应。

③ 每个小段中包含的信息位不能太多，否则将增加译码线路的复杂性和译码时间。

④ 一般每个小段还要留出一个状态，表示本字段不发出任何微命令。因此当某字段的长度为三位时，最多只能表示七个互斥的微命令，通常用000表示不操作。

例如，运算器的输出控制信号有直传、左移、右移、半字交换等四个，这四个微命令是互斥的，它们可以安排在同一字段编码。同样，存储器的读和写命令也是一对互斥的微命令。还有象A→C、B→C（A、B、C都是寄存器）这样的一类微命令也是互斥的微命令，它们不允许在同一时刻出现。

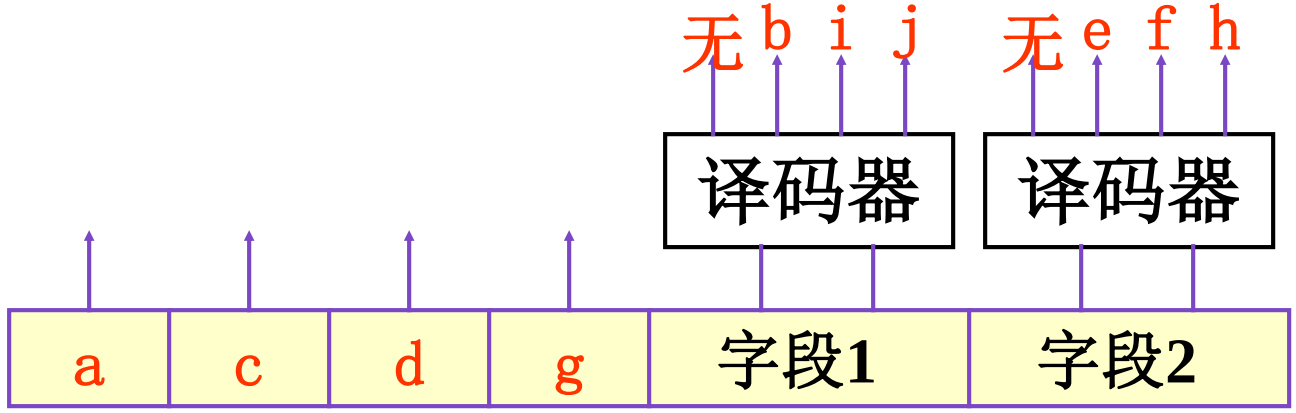
假设某计算机共有256个微命令，如果采用直接控制法，微指令的操作控制字段就要有256位；而如果采用最短编码法，操作控制字段只需要8位就可以了；如果采用字段直接编码法，若4位为一个段，共需18段，操作控制字段只需72位，而且在同一时刻可以并行发出18个不同的微命令。

6-20 某机有8条微指令$I_1 \sim I_8$，每条微指令所含的微命令控制信号如下表所示：

微指令	微命令信号									
	a	b	c	d	e	f	g	h	i	j
I_1	√	√	√	√	√					
I_2	√			√		√	√			
I_3		√						√		
I_4			√							
I_5			√		√		√		√	
I_6	√							√		√
I_7			√	√				√		
I_8	√	√						√		

a～j分别代表10种不同性质的微命令信号。

假定采用直接控制法，则需要10位。例题中微指令中有多个微命令是兼容的微命令，必须同时出现，如微指令I1中的a～e，故也不可以采用最短编码法。



字段1
- 00 无
- 01 b
- 10 i
- 11 j

字段2
- 00 无
- 01 e
- 10 f
- 11 h

I1：11100101
I2：10110010
I3：00000111
I4：01000000
I5：01011001
I6：10001111
I7：01100011
I8：10000111

微程序控制器的组成和工作过程

1.微程序控制器的基本组成

微程序控制器比组合逻辑控制器多出以下几个部件：

(1)控制存储器（CM）

这是微程序控制器的核心部件，用来存放微程序。

(2)微指令寄存器（μIR）

用来存放从CM取出的正在执行的微指令。

(3)微地址形成部件

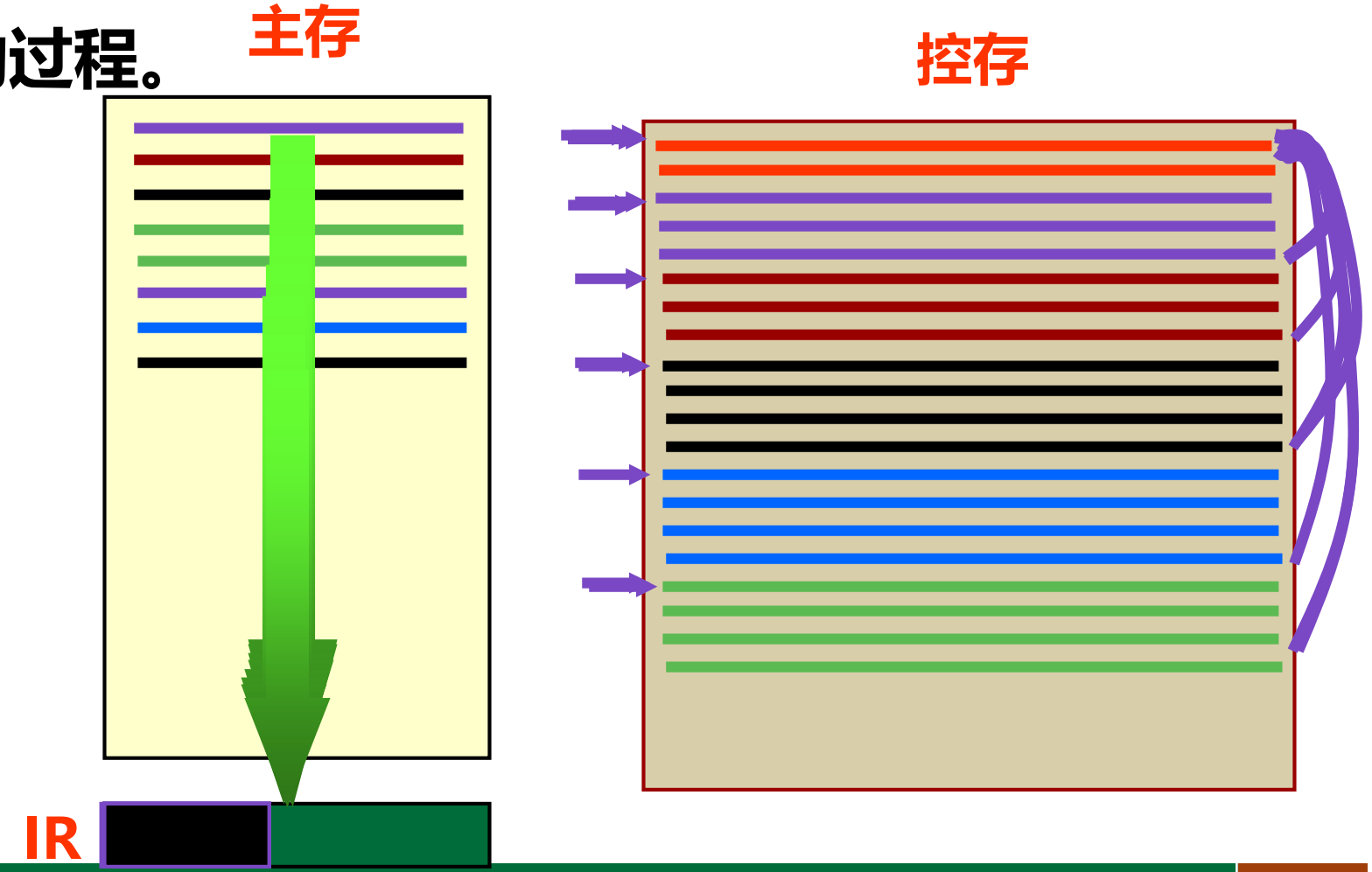
用来产生初始微地址和后继微地址。

(4)微地址寄存器（μMAR）

它接受微地址形成部件送来的微地址，为在CM中读取微指令作准备。

微程序控制器的工作过程

微程序控制器的工作过程实际上就是在微程序控制器的控制下，计算机执行机器指令的过程。



微程序控制器的工作过程

(1)执行取指令公操作。在机器开始运行时，自动将取指微程序的入口微地址送μMAR，并从CM中读出相应的微指令送入μIR。微指令的操作控制字段产生有关的微命令，用来控制实现取机器指令的公共操作。取指微程序的入口地址一般为CM的0号单元，当取指微程序执行完后，从主存中取出的机器指令就已存入指令寄存器IR中了。

(2)由机器指令的操作码字段通过微地址形成部件产生出该机器指令所对应的微程序的入口地址，并送入μMAR。

(3)从CM中逐条取出对应的微指令并执行之。

(4)执行完对应于一条机器指令的一段微程序后又回到取指微程序的入口地址，继续第(1)步，以完成取下条机器指令的公共操作。

以上是一条机器指令的执行过程，如此周而复始，直到整个程序执行完毕为止。

微程序入口地址的形成

每条机器指令对应一段微程序，当公用的取指微程序从主存中取出机器指令之后，**由机器指令的操作码字段指出各段微程序的入口地址（初始微地址）**。这是一种多分支（或多路转移）的情况，由机器指令的操作码转换成初始微地址的方式主要有三种。

1.一级功能转换

如果机器指令操作码字段的位数和位置固定，可以直接使操作码与入口地址码的部分位相对应。

一级功能转换

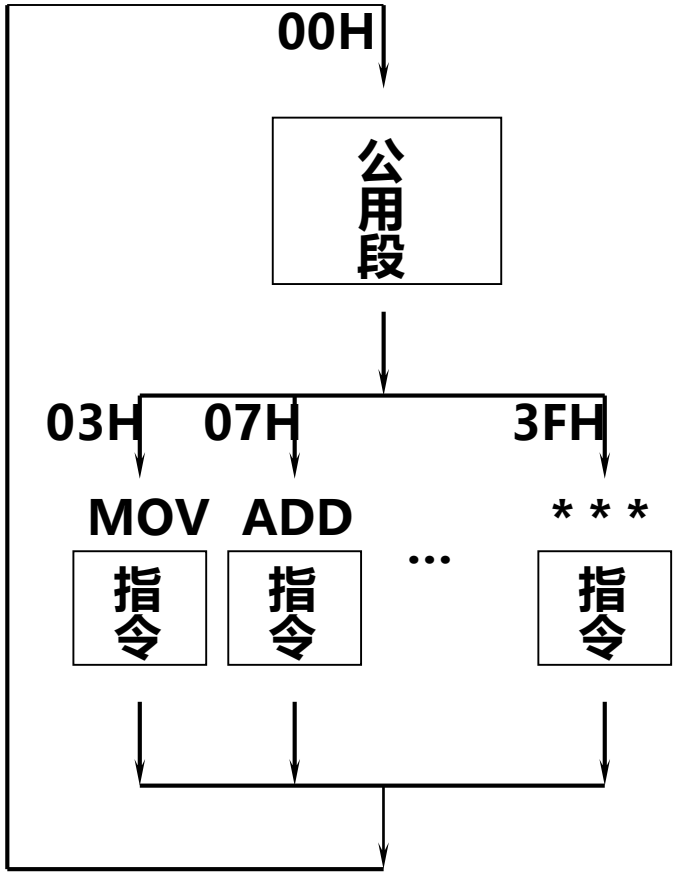
例如，某机有16条机器指令，指令操作码由4位二进制数表示，现以字母θ表示操作码，令微程序的入口地址为：

θ 11B

0 0 1 0 1 1

0 B H

各微程序的入口地址相差4个单元。



2.二级功能转换

当同类机器指令的操作码字段的位数和位置固定，而不同类机器指令的操作码的位数和位置不固定时，就不能再采用一级功能转换的方法。所谓二级功能转换是指第一次先按指令类型标志转移，以区分出指令属于哪一类，如：是单操作数指令，还是双操作数指令等。因为每一类机器指令中操作码字段的位数和位置是固定的，所以第二次即可按操作码区分出具体是哪条指令，以便找出相应微程序的入口微地址。

3.通过PLA电路实现功能转换

当机器指令的操作码位数和位置都不固定时，可以采用PLA电路将每条机器指令的操作码翻译成对应的微程序入口地址。这种方法对于变长度、变位置的操作码显得更有效，而且转换速度较快。

后继微地址的形成

找到初始微地址之后，可以开始执行微程序，每条微指令执行完毕都要根据要求形成后继微地址。后继微地址的形成方法对微程序编制的灵活性影响很大，它主要有两大基本类型：**增量方式和断定方式**。

1.增量方式（顺序 - 转移型微地址）

这种方式和机器指令的控制方式很类似，它也有顺序执行、转移和转子之分。顺序执行时后继微地址就是现行微地址加上一个增量（通常为1）；转移或转子时，由微指令的顺序控制字段产生转移微地址。因此，在微程序控制器中应当有一个微程序计数器（μPC），为了降低成本，一般情况下都是将微地址寄存器μMAR改为具有计数功能的寄存器，以代替μPC。

后继微地址的形成

2.断定方式

断定方式的后继微地址可由微程序设计者指定，或者根据微指令所规定的测试结果直接决定后继微地址的全部或部分值。

这是一种直接给定与测试断定相结合的方式，其顺序控制字段一般由两部分组成：非测试段和测试段。

(1)非测试段，可由设计者指定，一般是微地址的高位部分，用来指定后继微地址在CM中的某个区域内。

(2)测试段，根据有关状态的测试结果确定其地址值，一般对应微地址的低位部分。这相当于在指定区域内断定具体的分支。所依据的测试状态可能是指定的开关状态、指令操作码、状态字等。

微程序设计方法

(1)水平型微指令及水平型微程序设计

水平型微指令是指一次能定义并能并行执行多个微命令的微指令。它的并行操作能力强，效率高，灵活性强，执行一条机器指令所需微指令的数目少，执行时间短；但微指令字较长，增加了控存的横向容量，同时微指令和机器指令的差别很大，设计者只有熟悉了数据通路，才有可能编制出理想的微程序，一般用户不易掌握。

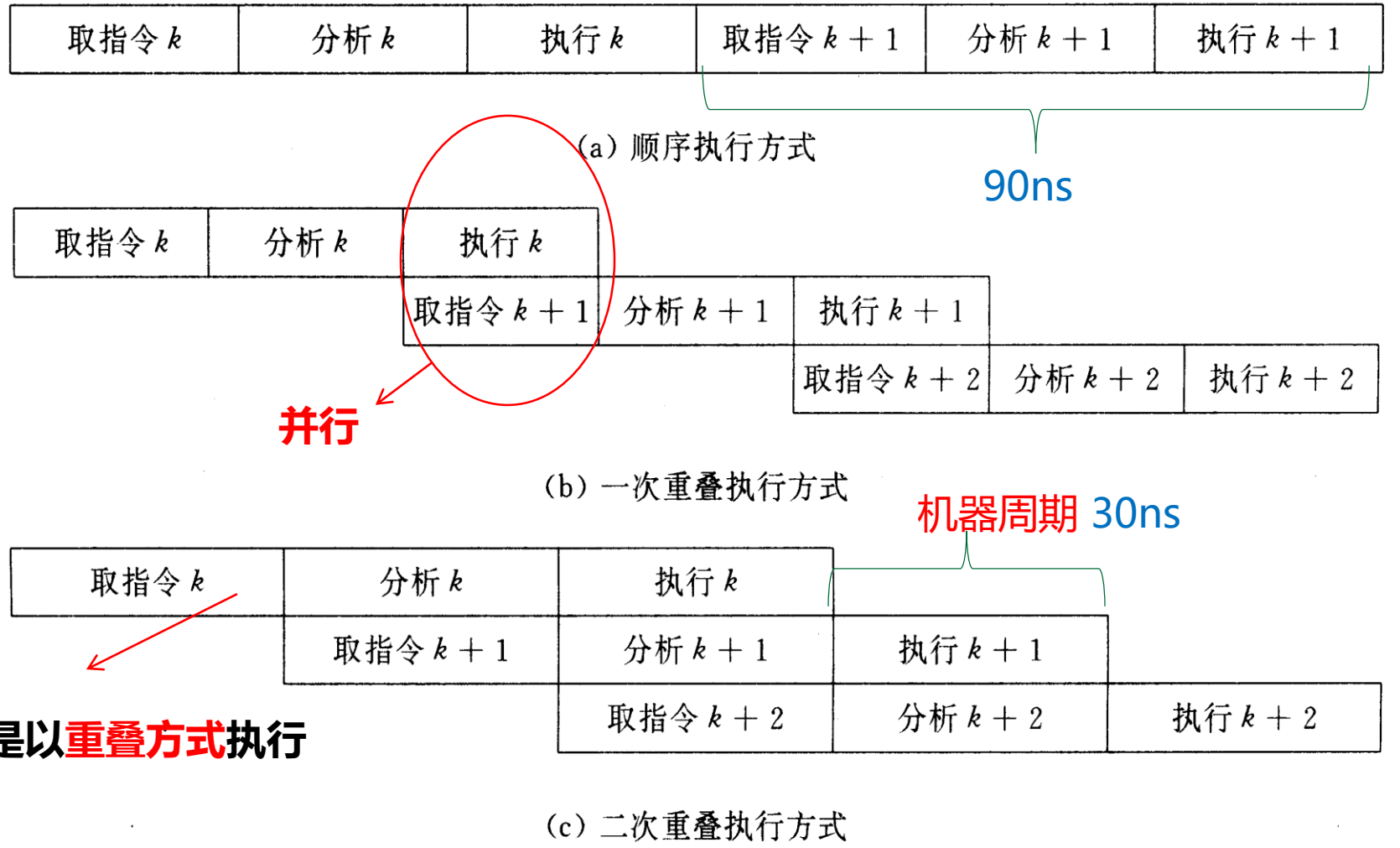
(2)垂直型微指令和垂直型微程序设计

垂直型微指令是指一次只能执行一个微命令的微指令。它的并行操作能力差，一般只能实现一个微操作，控制一、二个信息传送通路，效率低，执行一条机器指令所需的微指令数目多，执行时间长；但是微指令与机器指令很相似，所以容易掌握和利用，编程比较简单，不必过多地了解数据通路的细节，且微指令字较短。

一．中央处理器的功能和组成
二．控制器的组成和实现方法
三．时序系统与控制方式
四．微程序控制原理
五．流水线基本原理

例：假定一条指令的执行分为三个阶段：取指令、分析、执行

下图表示了三种执行方式相邻指令间的时序关系。

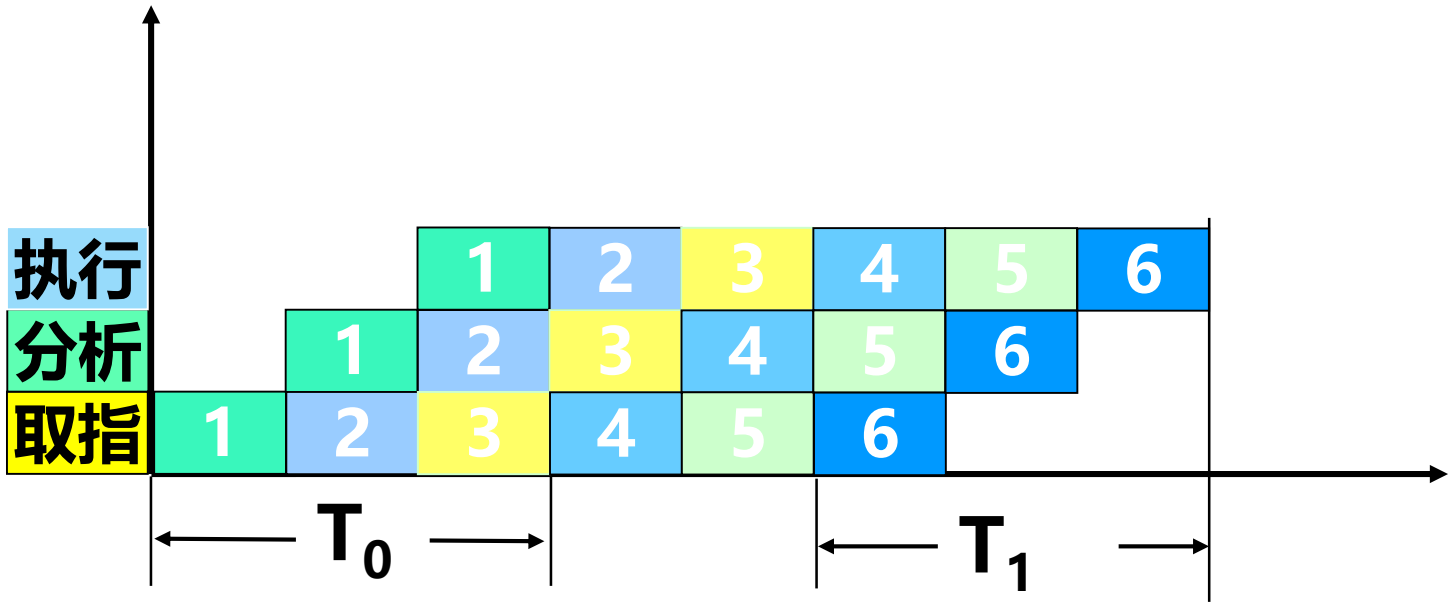


流水线的本质就是以重叠方式执行指令。

流水线常用时空图描述

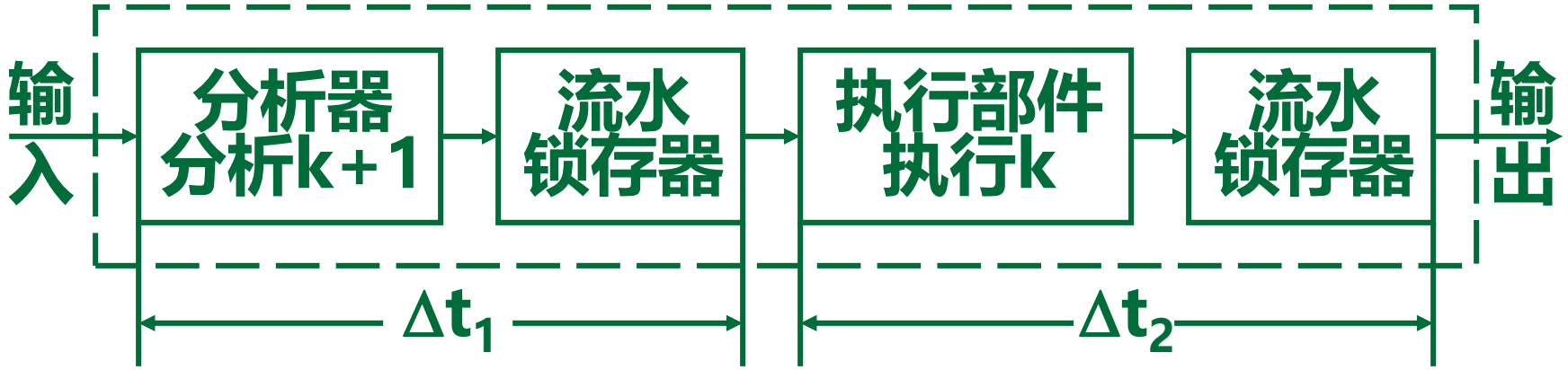
流水线从开始启动工作到流出第一个结果，需要经过一段流水线的**建立时间T_0**，在这段时间里，流水线没有流出任何结果。

排空时间：最后一个任务进入到结果流出所需要的时间



T_0为建立时间，T_1为排空时间

简单流水线



流水线的每一个阶段称为流水步、流水步骤、流水段、流水线阶段、流水功能段、功能段、流水级、流水节拍等。

流水线段间采用同步时钟控制，**流水线周期**等于时间最长的流水段的时间长度，每个流水线周期输出一个任务结果。**各流水段的时间应尽量相等**。

关于流水线的一些概念和结论

吞吐量：单位时间从流水线流出的指令数。流水线是开发串行指令流中并行性的一种实现技术，并不加快每一条指令的执行时间，而是通过增大吞吐量，来使程序运行更快，执行时间变短。

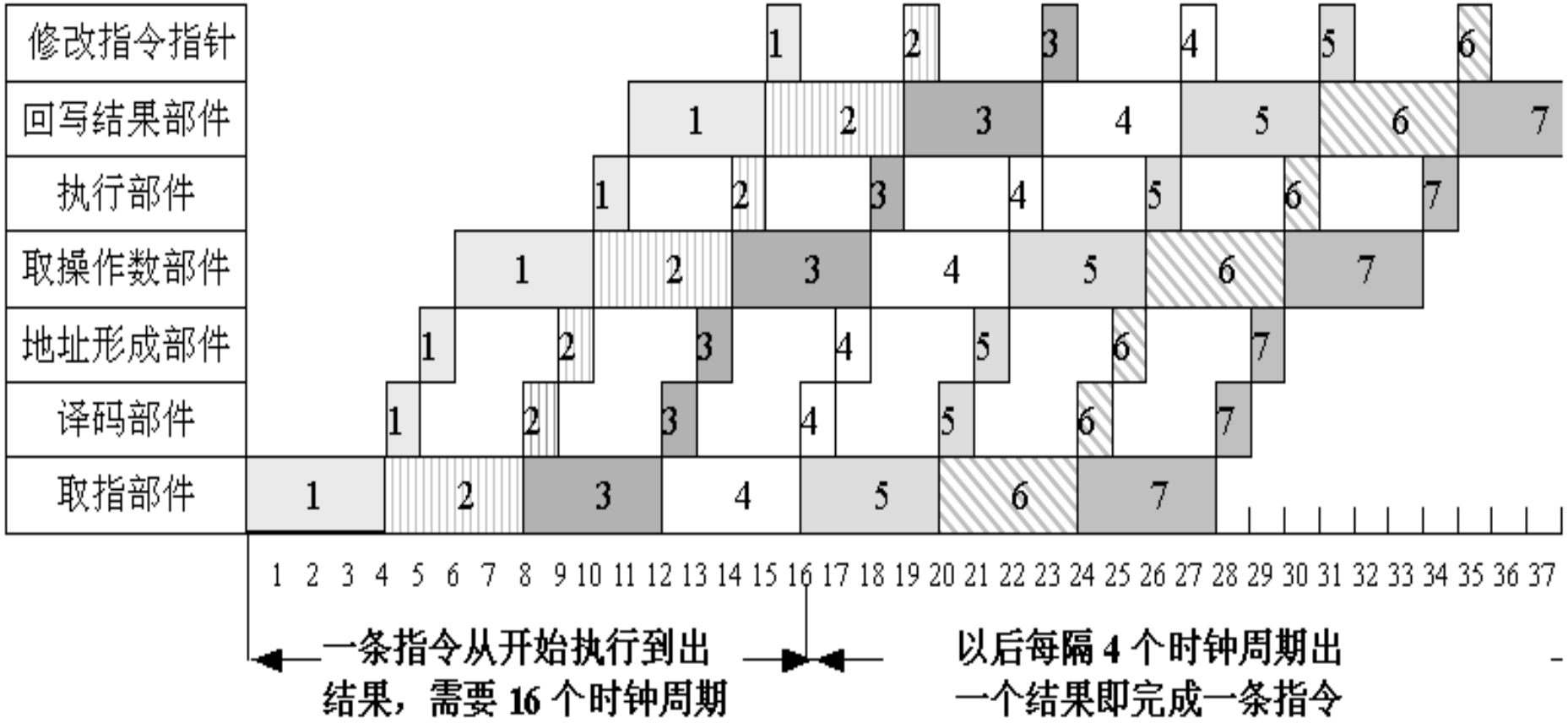
流水线段不是越多越好，附加的一些控制和流水线寄存器延迟和时钟偏移等限制了时钟周期频率，有可能比非流水线实现增加开销。

不平衡： 各段时间以最慢的为基准，会降低性能(如图)

理想情况：流水线的加速比等于流水线机器的段数。

流水线技术（硬件实现）对编程者透明。

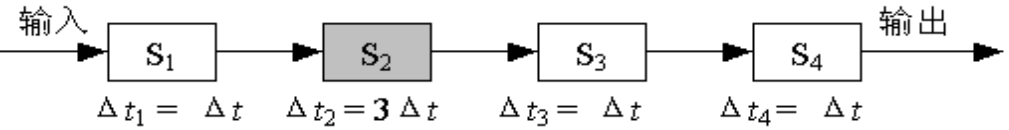
不平衡流水段示意



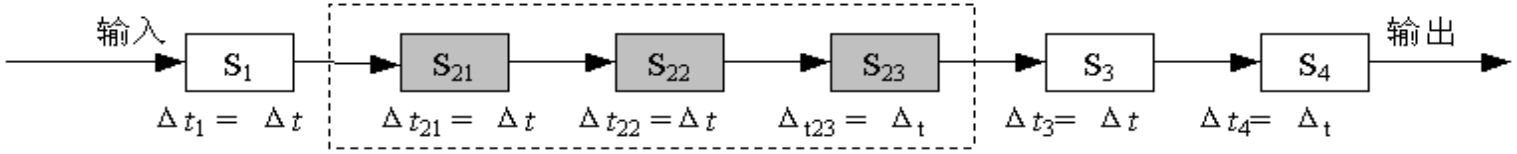
不均衡流水线的工作过程

不平衡流水段是瓶颈，解决方法

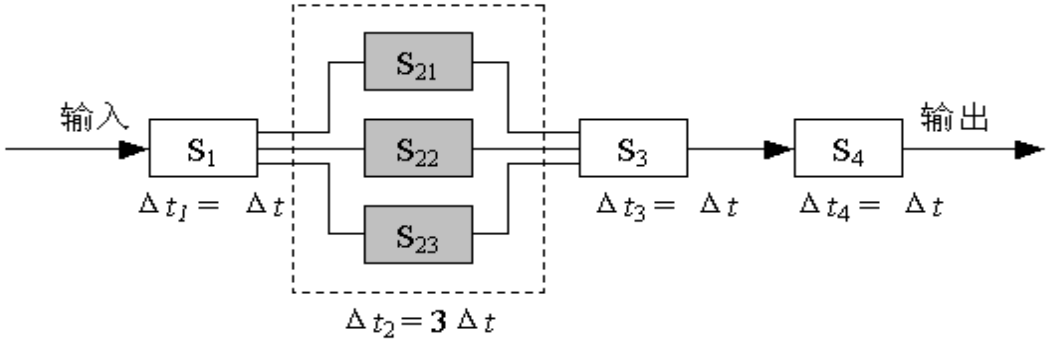
- 一是将“瓶颈”部分再细分，如图 (b)所示，当分成与其他时间步(设为1个时钟周期)几乎相等的功能段时，就会每一个时间步(1个时钟周期)出一条指令；
- 另一种是采用如图©所示的“瓶颈”段复制的方法，用数据分配器，将多条指令的“瓶颈”段(访存)并行地执行,加快执行过程。当然，后者的复杂度要高



（a） 不均衡流水线中的“瓶颈”



（b）“瓶颈”段的再细分



（c）“瓶颈”段的重复设置

不均衡流水线及其“瓶颈”的消除

影响流水线性能的主要障碍

结构冒险

发生硬件资源冲突，常见引起结构冒险的情况：多重访问寄存器堆,多重访问存储器

没有或没有充分流水功能部件等

解决：可以增加硬件资源；或者停顿流水线

数据冒险

几条指令重叠执行时，一条指令依赖前面指令的结果却没有准备好（还没有计算或存储）

解决：寄存器堆在一个周期内时序控制先写后读，Forwarding 通路，软件调度或者停顿流水线

控制冒险：（流水线执行转移指令时）

进入下一个时钟周期时，转移条件和转移目标PC不能按时提供给IF段取指令。

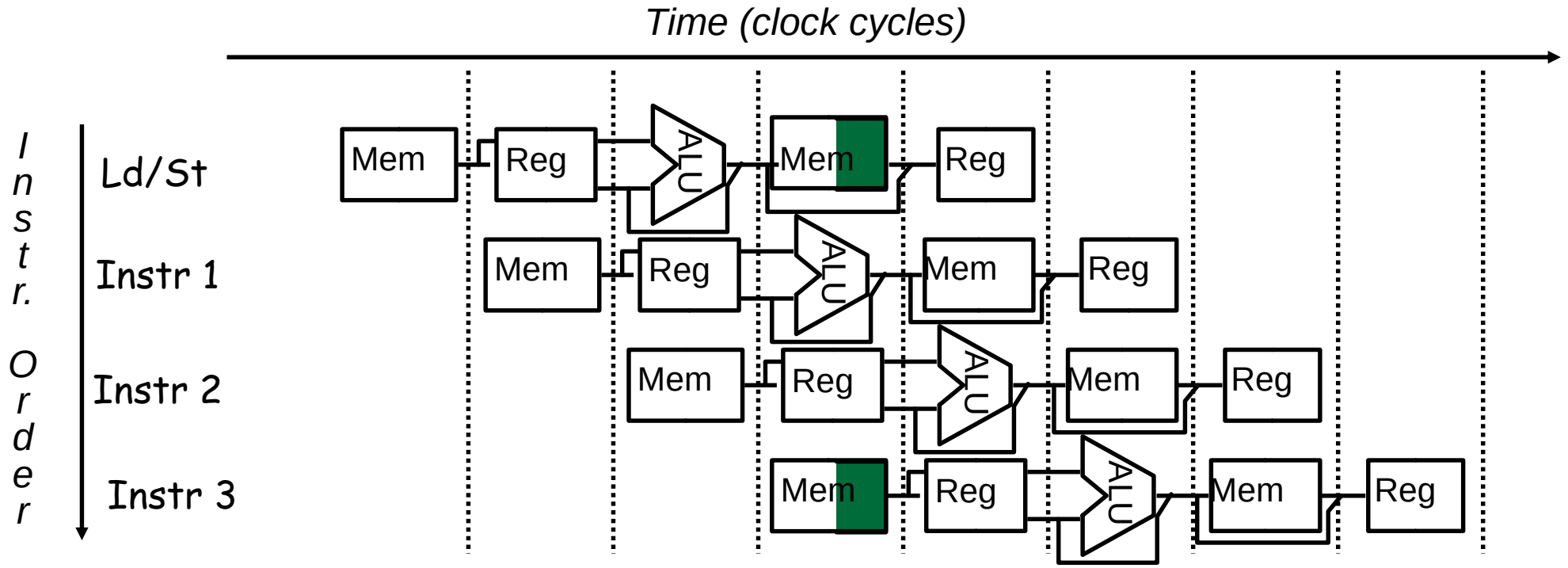
解决：编译器调度指令顺序，分支预测或则停顿流水线。

结构冒险示例

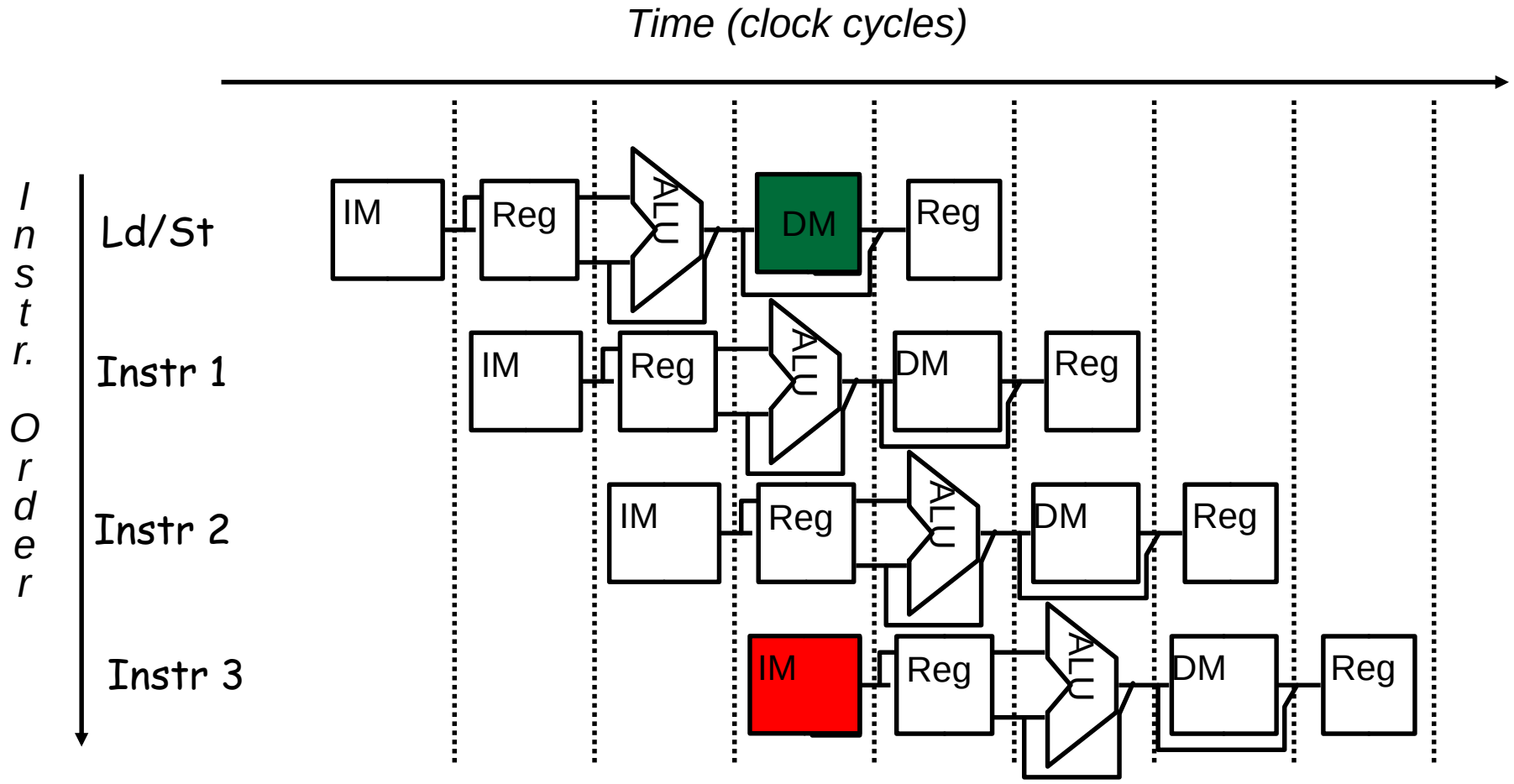
北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

注意：在同一时钟周期不同操作不能使用同一数据通路资源。

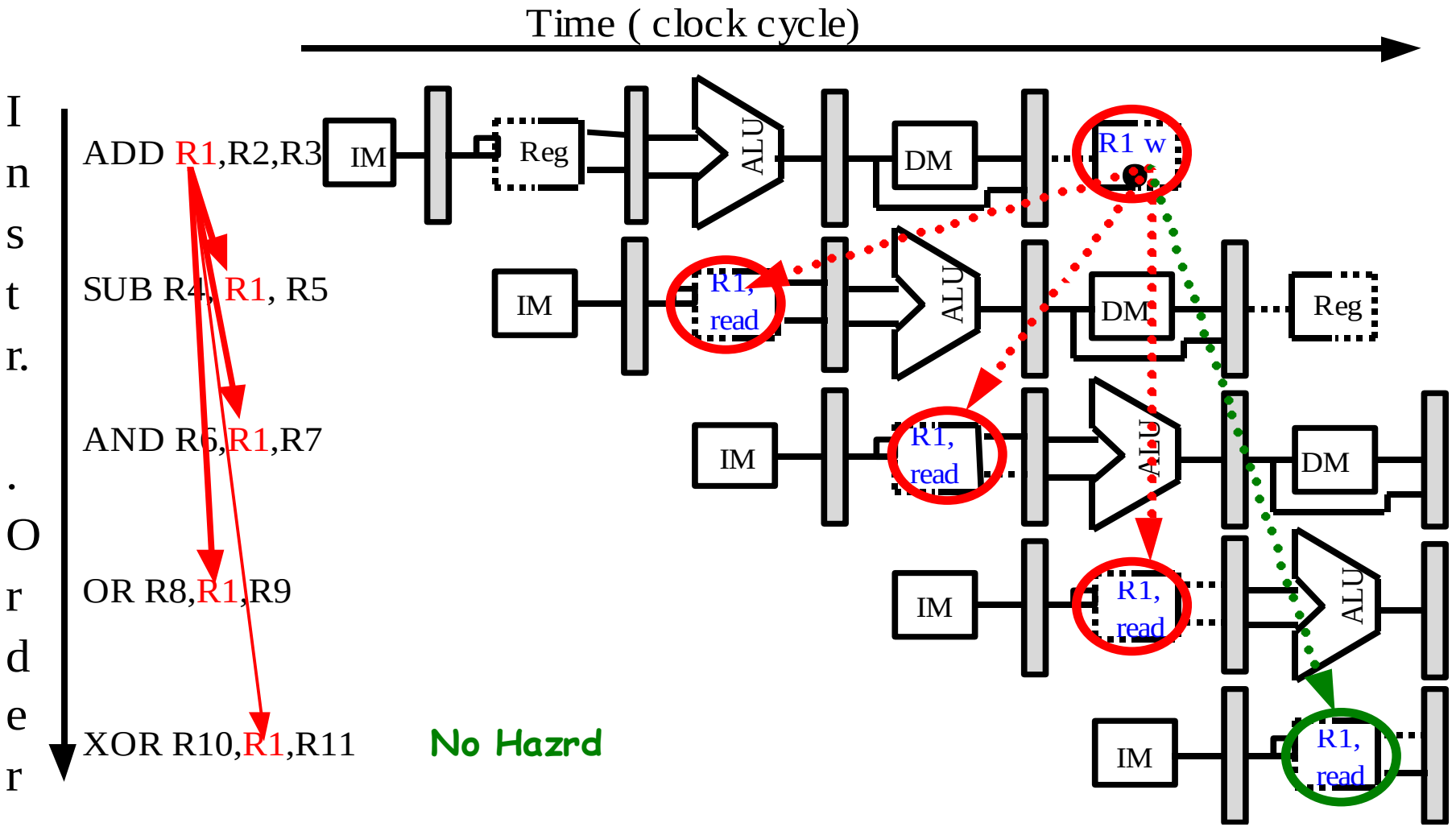
- **有访问存储器冲突！**



解决方案：使用分开的指令cache和数据cache

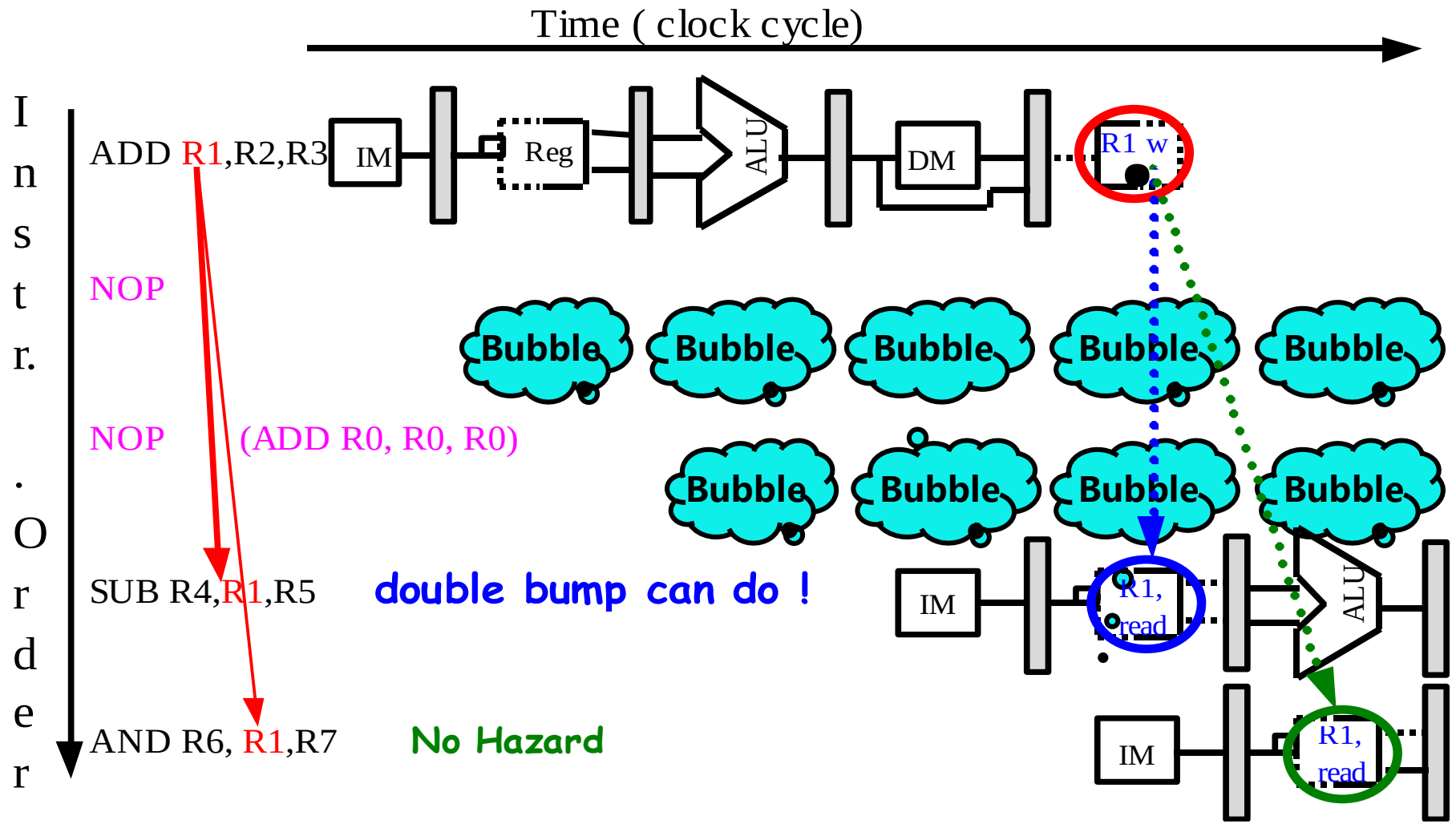


数据冒险示例


Time (clock cycle)
Instr. Order
ADD R1,R2,R3
SUB R4, R1, R5
AND R6,R1,R7
OR R8,R1,R9
XOR R10,R1,R11
No Hazrd
IM
Reg
ALU
DM
R1 w
R1, read

冒险总是可以用停顿解决

- 解决冒险最简单的方式就是停顿流水线，即暂停流水线一个或多个时钟周期。
- 一条指令被停顿后，其后的所有指令被停顿；该指令之前的指令必须继续执行。
- 一个流水线停顿也称为流水线气泡或气泡。


Time (clock cycle)
Instr. Order
ADD R1,R2,R3
NOP
NOP
SUB R4,R1,R5
AND R6, R1,R7
(ADD R0, R0, R0)
double bump can do !
No Hazard
IM
Reg
ALU
DM
R1 w
Bubble
R1, read

流水线的主要性能指标

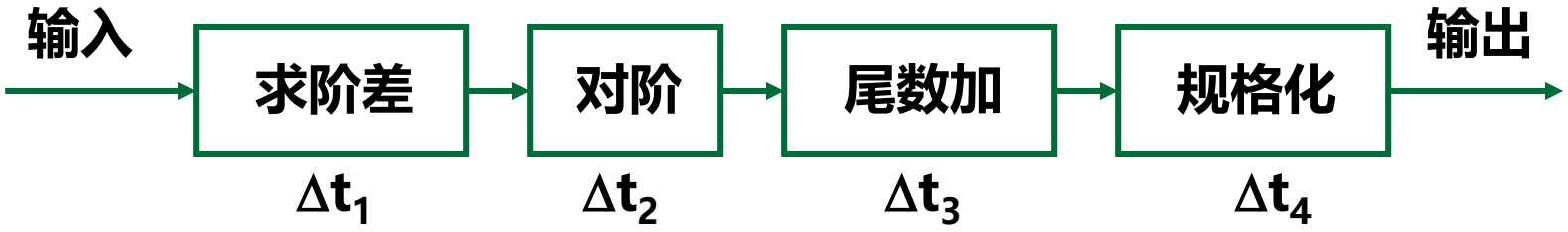
- 衡量流水线处理机的主要性能指标：**吞吐率（TP)和效率（η）。**
- 不论何种流水线，都可以通过时—空图求出吞吐率和效率

$$TP = \frac{\text{任务数}}{\text{从开始流入到n个任务全部流出的时间}}$$

$$\eta = \frac{\text{n个任务的时空区面积}}{\text{m个段的总的时空区面积}}$$

流水线分类

- 从流水处理的级别上分类:
  - 部件级（又称为运算操作流水线 ）
    - 指构成部件内的各子部件之间的流水
    - 例如，运算器内部浮点加法流水线



部件级流水线（操作流水线）——浮点加法器流水线

流水线分类

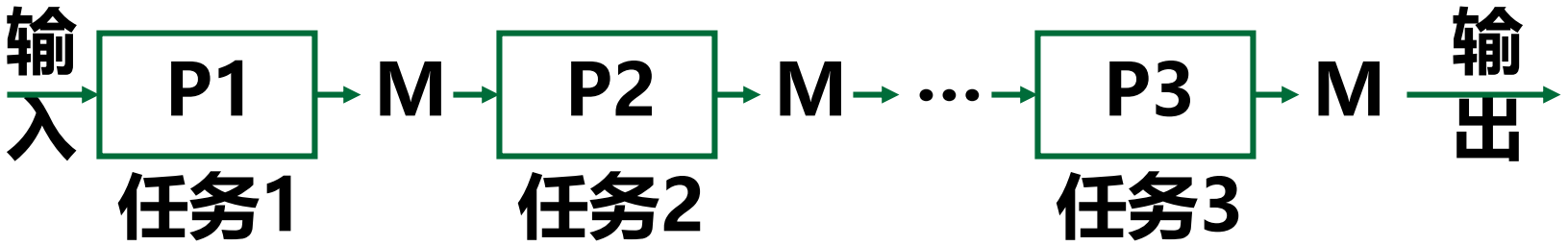
处理机级（又称为指令流水线）

指构成处理机的各个部件之间的流水

例如，“取指”、“分析”、“执行”之间的流水

系统级（又称为宏流水）

指构成计算机系统的多个处理机之间的流水

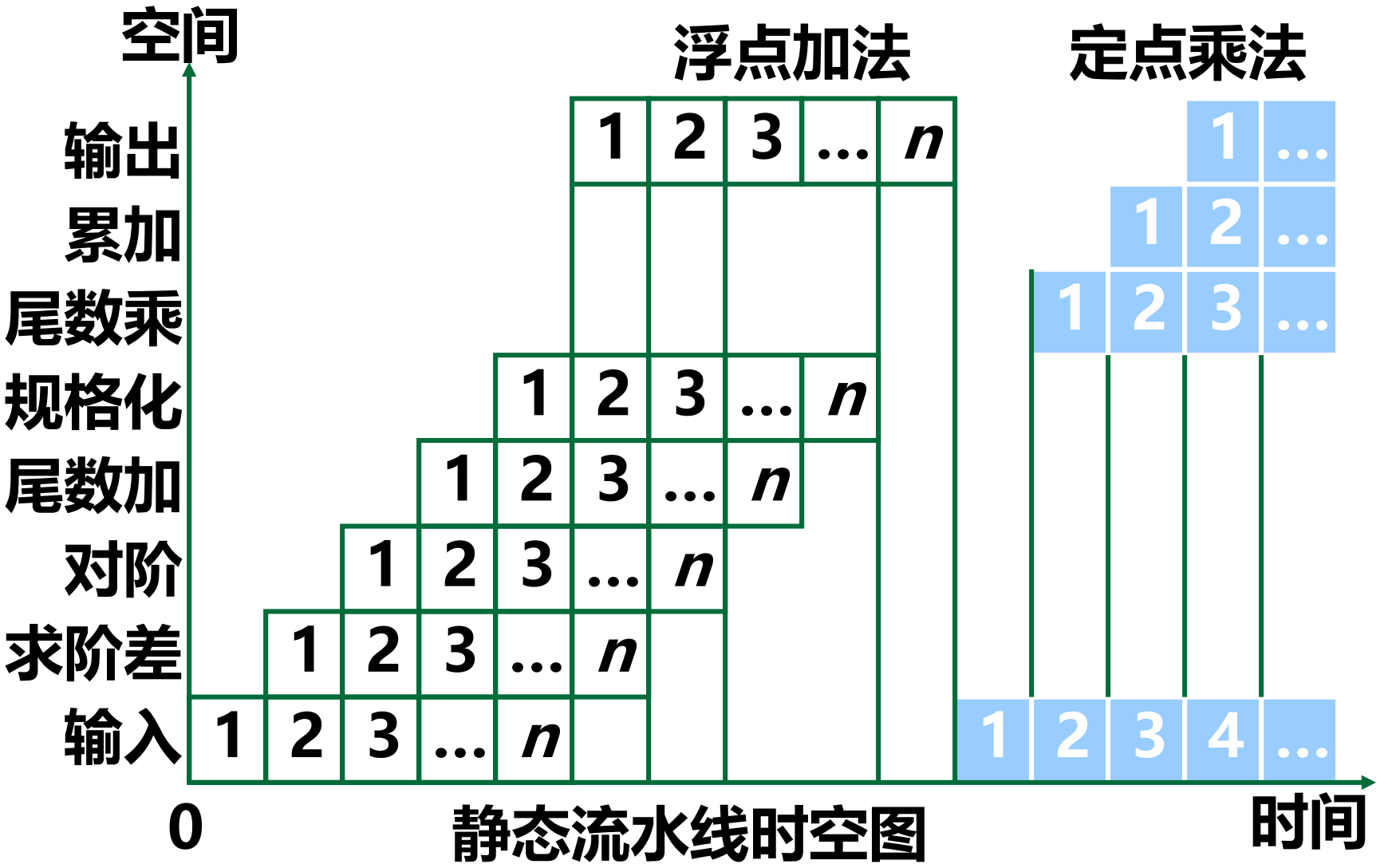


处理机之间的流水线——宏流水线

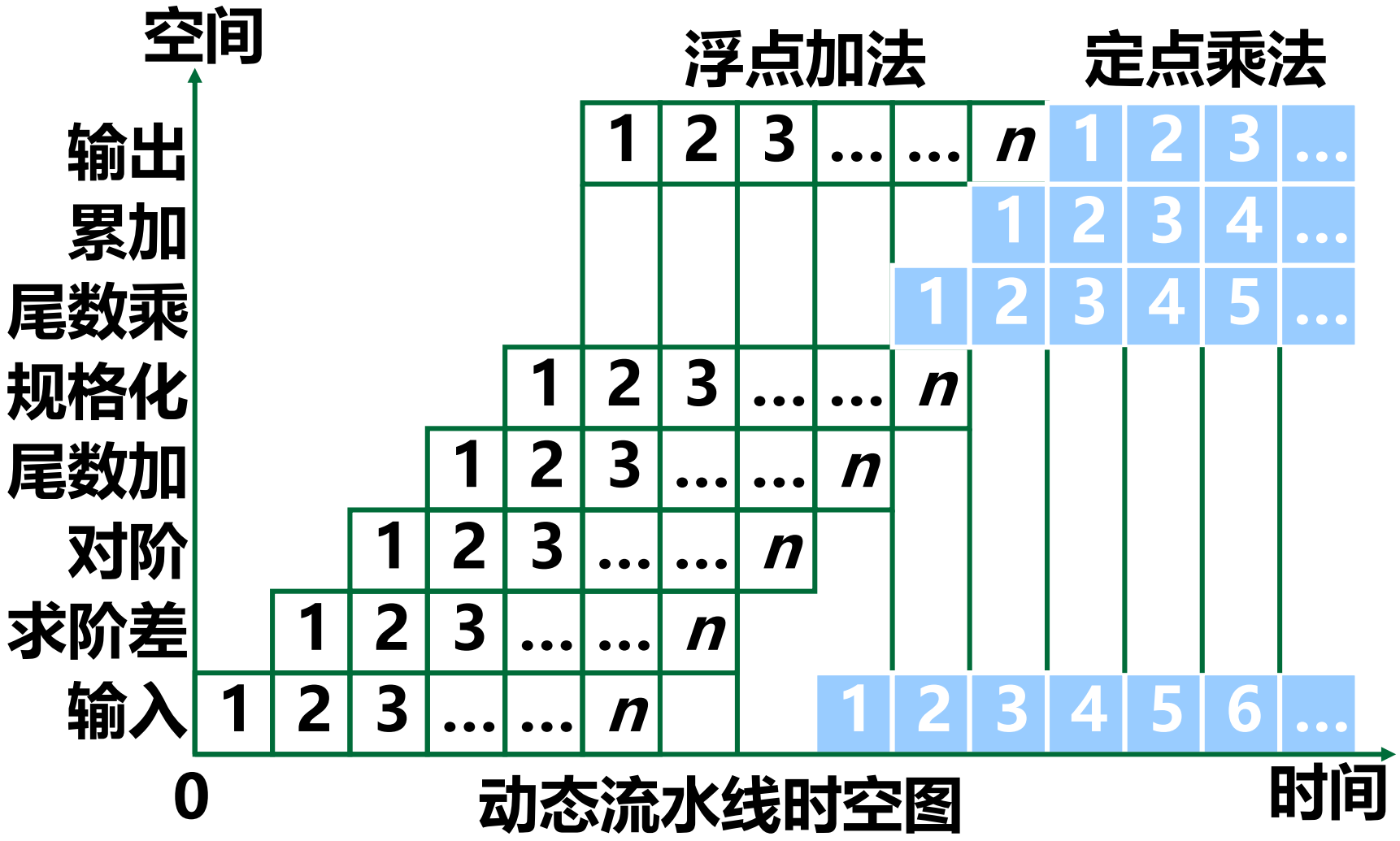
流水线分类

- 从流水线具有的功能分类
  - 单功能流水线
    - 指只能实现一种功能的流水线
    - 例如，只实现浮点加减的流水线
    - 可以将多条单功能流水线组合起来，完成多功能的流水
  - 多功能流水线
    - 指同一流水线的各个段之间可以有多种不同的联接方式，以实现多种不同的运算或功能
    - 静态流水线
    - 动态流水线

静态流水线示例



动态流水线示例



动态流水线时空图

感谢聆听