# 课时5 中央处理器



| 考点 | 重要程度 | 占分 | 题型 |
| --- | --- | --- | --- |
| 1.CPU的功能和基本结构 | ★★★ | 0～3 | 选择、填空 |
| 2.指令执行过程 | 必考 | 6～10 | 大题 |
| 3.数据通路的基本结构 | 必考 | 6～10 | 大题 |
| 4.指令流水线 | ★★ | 3～5 | 选择、填空 |

# 5.1 CPU的功能和基本结构

## 1.CPU的功能

中央处理器（CPU）由运算器和控制器组成。

其中，控制器的功能是负责协调并控制计算机各部件执行程序的指令序列，包括取指令、分析指令和执行指令；运算器的功能是对数据进行加工。

# 1.CPU的功能

## CPU的具体功能包括：

1）指令控制. 完成取指令、分析指令和执行指令的操作，即程序的顺序控制。

2）操作控制. 一条指令的功能往往由若干操作信号的组合来实现。CPU管理并产生由内存取出的每条指令的操作信号，把各种操作信号送往相应的部件，从而控制这些部件按指令的要求进行动作。

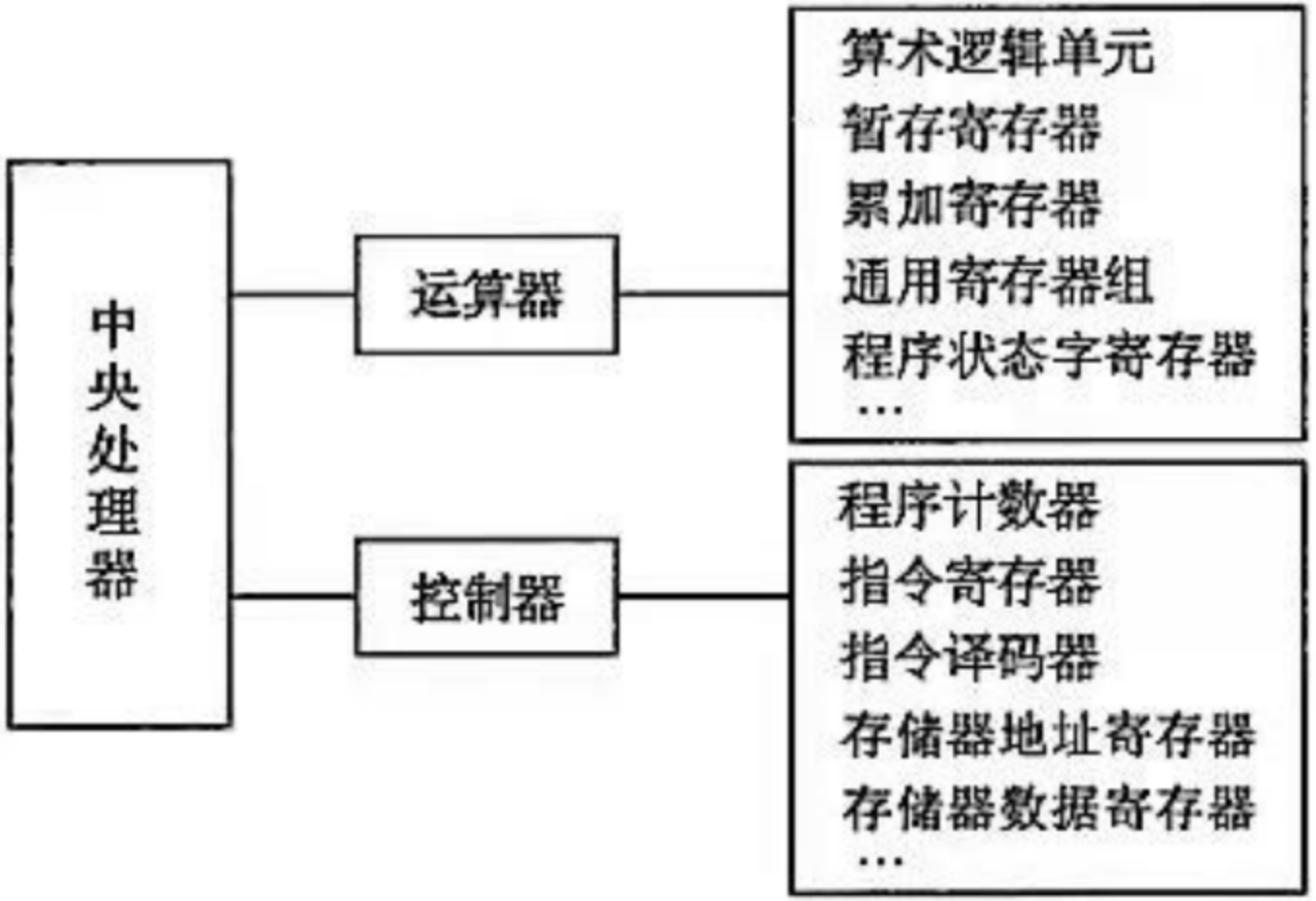
3）时间控制. 对各种操作加以时间上的控制。时间控制要为每条指令按时间顺序提供应有的控制信号。

4）数据加工. 对数据进行算术和逻辑运算。

5）中断处理. 对计算机运行过程中出现的异常情况和特殊请求进行处理。

## 2.CPU的基本结构

在计算机系统中，中央处理器主要由运算器和控制器两大部分组成，如图所示。




中央处理器的组成

# 2.CPU的基本结构

## 一．运算器

运算器接收从控制器送来的命令并执行相应的动作，对数据进行加工和处理。

运算器是计算机对数据进行加工处理的中心，它主要由算术逻辑单元（ALU）、暂存寄存器、累加寄存器（ACC）、通用寄存器组、程序状态字寄存器（PSW）、移位器、计数器（CT）等组成。

1）算术逻辑单元.主要功能是进行算术／逻辑运算。

2）暂存寄存器. 用于暂存从主存读来的数据，该数据不能存放在通用寄存器中，否则会破坏其原有内容。暂存寄存器对应用程序员是透明的。

3）累加寄存器. 它是一个通用寄存器，用于暂时存放ALU运算的结果信息，可以作为加法运算的一个输入端。

## 2.CPU的基本结构

4）通用寄存器组.如AX、BX、CX、DX、SP等，用于存放操作数（包括源操作数、目的操作数及中间结果）和各种地址信息等。SP是堆栈指针，用于指示栈顶的地址。

5）程序状态字寄存器.保留由算术逻辑运算指令或测试指令的结果而建立的各种状态信息，如溢出标志（OF）、符号标志（SF）、零标志（ZF）、进位标志（CF）等。PSW中的这些位参与并决定微操作的形成。

6）移位器.对操作数或运算结果进行移位运算。

7）计数器.控制乘除运算的操作步数。

# 2.CPU的基本结构

## 二．控制器

控制器是整个系统的指挥中枢，在控制器的控制下，运算器、存储器和输入／输出设备等功能部件构成一个有机的整体，根据指令的要求指挥全机协调工作。控制器的基本功能是执行指令，每条指令的执行是由控制器发出的一组微操作实现的。

控制器由程序计数器（PC）、指令寄存器（IR）、指令译码器、存储器地址寄存器（MAR）、存储器数据寄存器（MDR）、时序系统和微操作信号发生器等组成。

1）程序计数器.用于指出下一条指令在主存中的存放地址。CPU 根据PC的内容去主存中取指令。因程序中指令（通常）是顺序执行的，所以PC有自增功能。

2）指令寄存器.用于保存当前正在执行的那条指令。

3）指令译码器.仅对操作码字段进行译码，向控制器提供特定的操作信号。

## 2.CPU的基本结构

4）存储器地址寄存器.用于存放要访问的主存单元的地址。

5）存储器数据寄存器.用于存放向主存写入的信息或从主存读出的信息。

6）时序系统. 用于产生各种时序信号，它们都由统一时钟（CLOCK）分频得到。

7）微操作信号发生器.根据IR的内容（指令）、PSW的内容（状态信息）及时序信号，产生控制整个计算机系统所需的各种控制信号，其结构有组合逻辑型和存储逻辑型两种。

## 2.CPU的基本结构

控制器的工作原理是，根据指令操作码、指令的执行步骤（微命令序列）和条件信号来形成当前计算机各部件要用到的控制信号。

计算机整机各硬件系统在这些控制信号的控制下协同运行，产生预期的执行结果。

注意：CPU内部寄存器可分两类：

1、用户可见的寄存器，可对这类寄存器编程，如通用寄存器组、程序状态字寄存器；

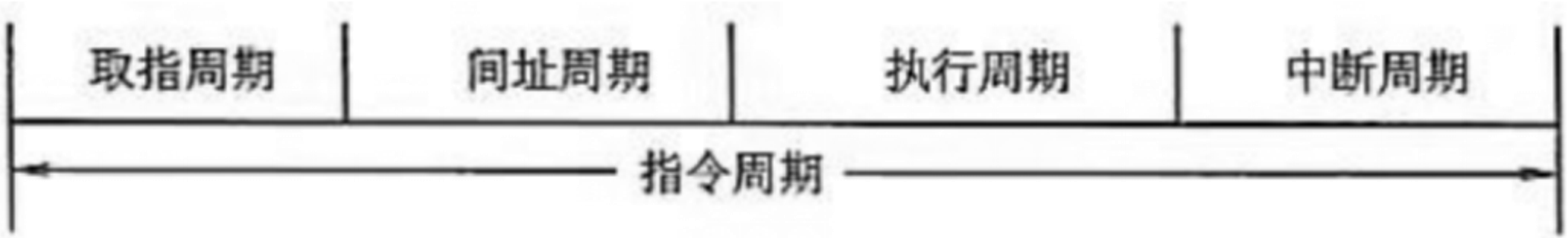
2、用户不可见的寄存器，对用户是透明的，不可对这类寄存器编程，如存储器地址寄存器、存储器数据寄存器、指令寄存器。

# 5.2 指令执行过程

## 1.指令周期

CPU从主存中取出并执行一条指令的时间称为指令周期，不同指令的指令周期可能不同。

当CPU采用中断方式实现主机和I／O设备的信息交换时，CPU在每条指令执行结束前，都要发中断查询信号，若有中断请求，则CPU进入中断响应阶段，又称中断周期.一个完整的指令周期应包括取指、间址、执行和中断4个周期，如图所示.



# 1.指令周期

4个工作周期都有CPU访存操作，只是访存的目的不同.

取指周期是为了取指令，间址周期是为了取有效地址，

执行周期是为了取操作数，中断周期是为了保程序断点.

为了区别不同的工作周期，在CPU内设置4个标志触发器FE、IND、EX和INT，

它们分别对应取指、间址、执行和中断周期，并以“1”状态表示有效，

分别由1→FE、1→IND、1→EX和1→INT这4个信号控制.

扫码听课，视频讲解更清晰

## 2.指令周期的数据流

数据流是根据指令要求依次访问的数据序列.

在指令执行的不同阶段，要求依次访问的数据序列是不同的。而且对于不同的指令，它们的数据流往往也是不同的。

### 一.取指周期

取指周期的任务是根据PC中的内容从主存中取出指令代码并存放在IR中。

取指周期的数据流如图所示。PC中存放的是指令的地址，根据此地址从内存单元中取出的是指令，并放在指令寄存器IR中，取指令的同时，PC加1。

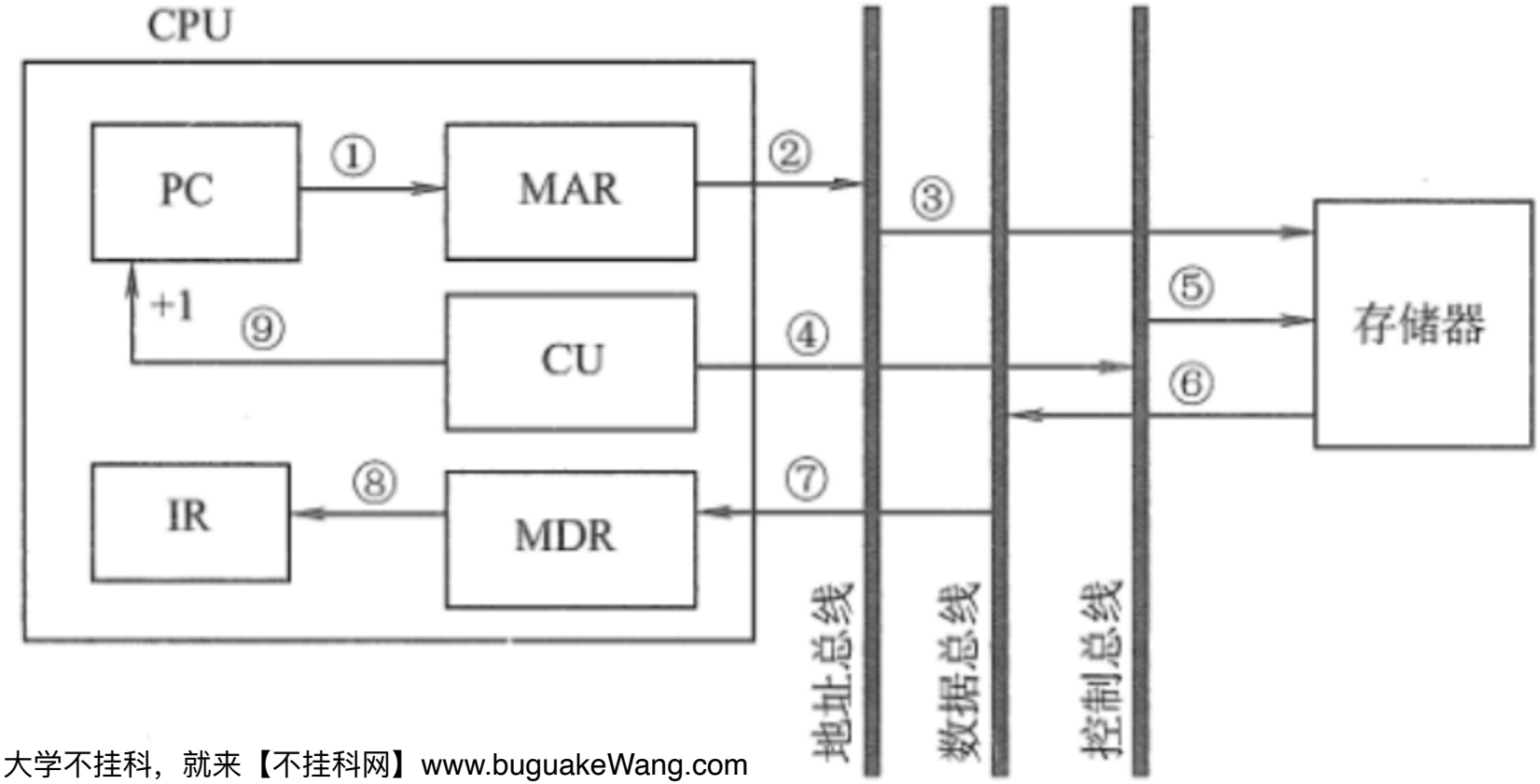
取指周期的数据流向如下：

1）PC①MAR②地址总线③主存.

2）CU发出控制信号④控制总线⑤主存.

3）主存⑥数据总线⑦MDR⑧IR（存放指令）.

4）CU发出读指令⑨PC内容加1.

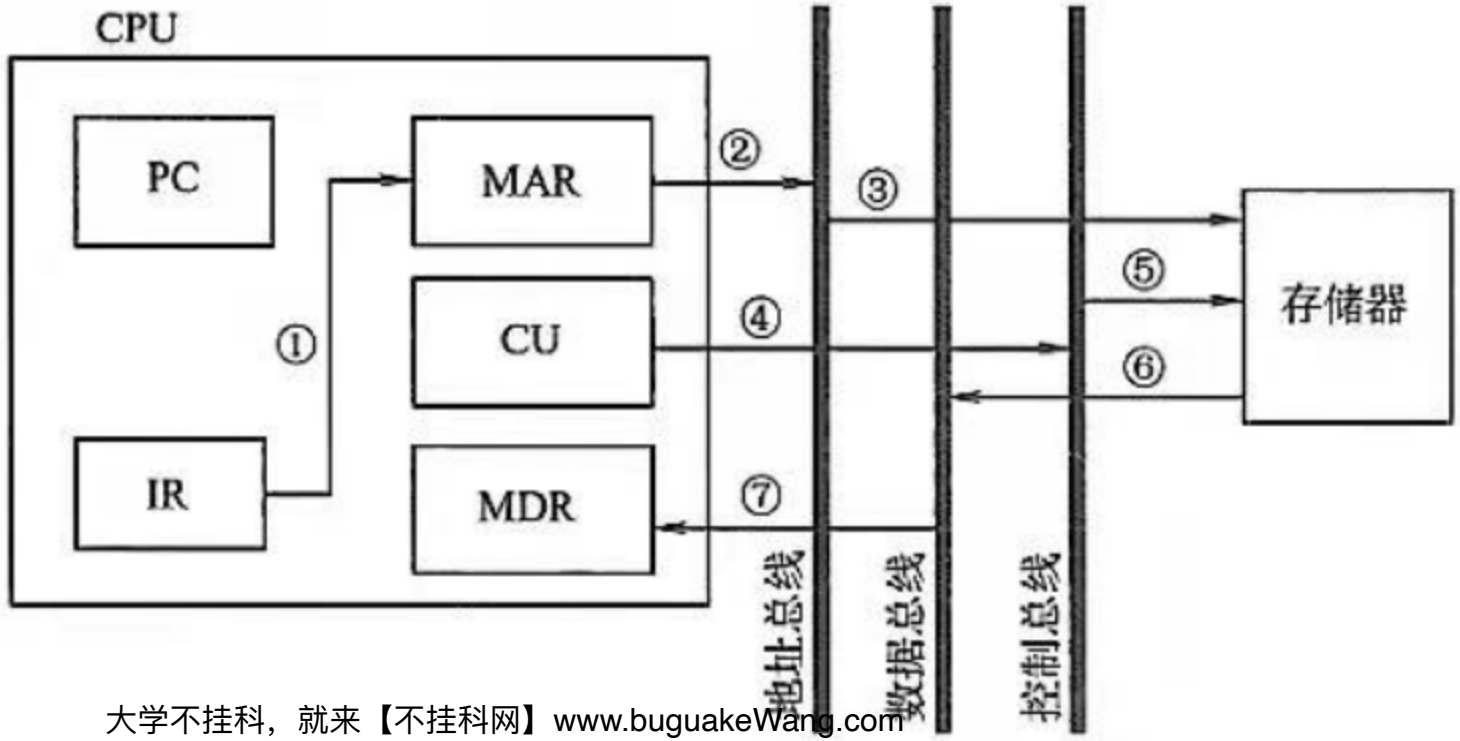


# 2.指令周期的数据流

## 二.间址周期

间址周期的任务是取操作数有效地址.

以一次间址为例（见图），将指令中的地址码送到MAR并送至地址总线，此后CU向存储器发读命令，以获取有效地址并存至MDR。



## 二.间址周期

间址周期的数据流向如下：

1）Ad(IR)(或MDR) ①→ MAR ②→ 地址总线 ③→ 主存.

2）CU发出读指令 ④→ 控制总线 ⑤→ 主存.

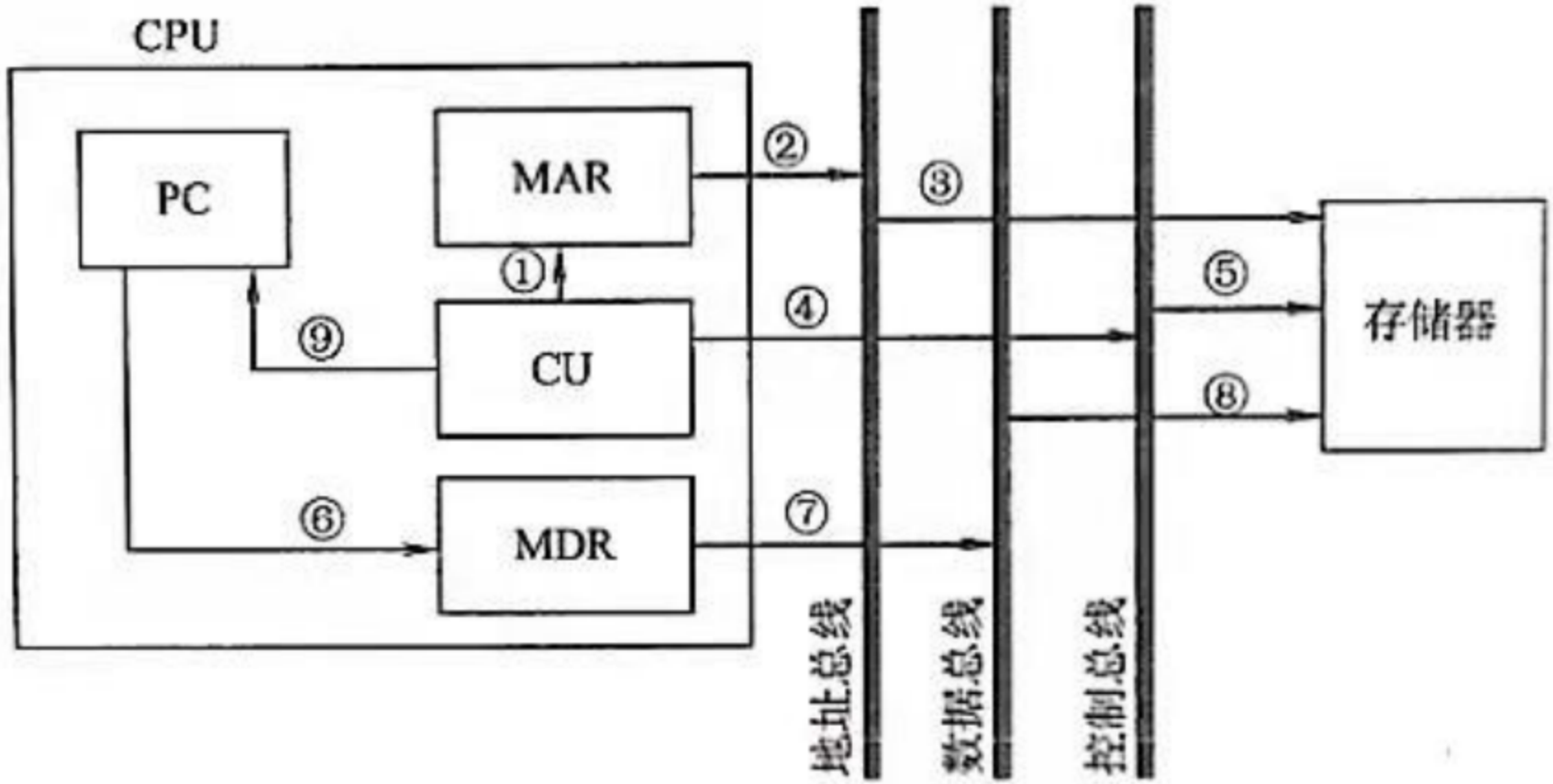
3）主存 ⑥→ 数据总线 ⑦→ MDR（存放有效地址）.

## 三.执行周期

执行周期的任务是根据IR中的指令字的操作码和操作数通过ALU操作产生执行结果.不同指令的执行周期操作不同，因此没有统一的数据流向。

# 四.中断周期

中断周期的任务是处理中断请求.假设程序断点存入堆栈中，并用SP指示栈顶地址，而且进栈操作是先修改栈顶指针，后存入数据，数据流如图所示。



# 四.中断周期

中断周期的数据流向如下：

1）CU控制酱SP减1，SP ①MAR② 地址总线③ 主存.

2）CU发出写命令④控制总线⑤主存.

3）PC⑥MDR⑦数据总线⑧主存（程序断点存入主存）.

4）CU（中断服务程序的入口地址)⑨PC.

## 3.指令执行方案

一个指令周期通常要包括几个时间段（执行步骤），每个步骤完成指令的一部分功能，几个依次执行的步骤完成这条指令的全部功能。

出于性能和硬件成本等考虑，可以选用3种不同的方案来安排指令的执行步骤。

### 一.单指令周期

对所有指令都选用相同的执行时间来完成，称为单指令周期方案。

此时每条指令 都在固定的时钟周期内完成，指令之间串行执行，

即下一条指令只能在前一条指令执行结束后才能启动.

因此，指令周期取决于执行时间最长的指令的执行时间。对于那些本来可以在更短时间内完成的指令，要使用这个较长的周期来完成，会降低整个系统的运行速度。

# 3.指令执行方案

## 二 . 多指令周期

对不同类型的指令选用不同的执行步骤来完成，称为多指令周期方案.

指令之间串行执行，即下一条指令只能在前一条指令执行结束后才能启动。但可选用不同个数的时钟周期来完成不同指令的执行过程，指令需要几个周期就为其分配几个周期，而不再要求所有指令占用相同的执行时间。

## 三 . 流水线方案

指令之间可以并行执行的方案，称为流水线方案.

其追求的目标是力争在每个时钟脉冲周期完成一条指令的执行过程（只在理想情况下才能达到该效果）。这种方案通过在每个时钟周期启动一条指令，尽量让多条指令同时运行，但各自处在不同的执行步骤中。

# 5.3 数据通路的基本结构

## 1. 数据通路的基本结构

数据通路的基本结构主要有以下几种：

1）CPU内部单总线方式。

将所有寄存器的输入端和输出端都连接到一条公共通路上，

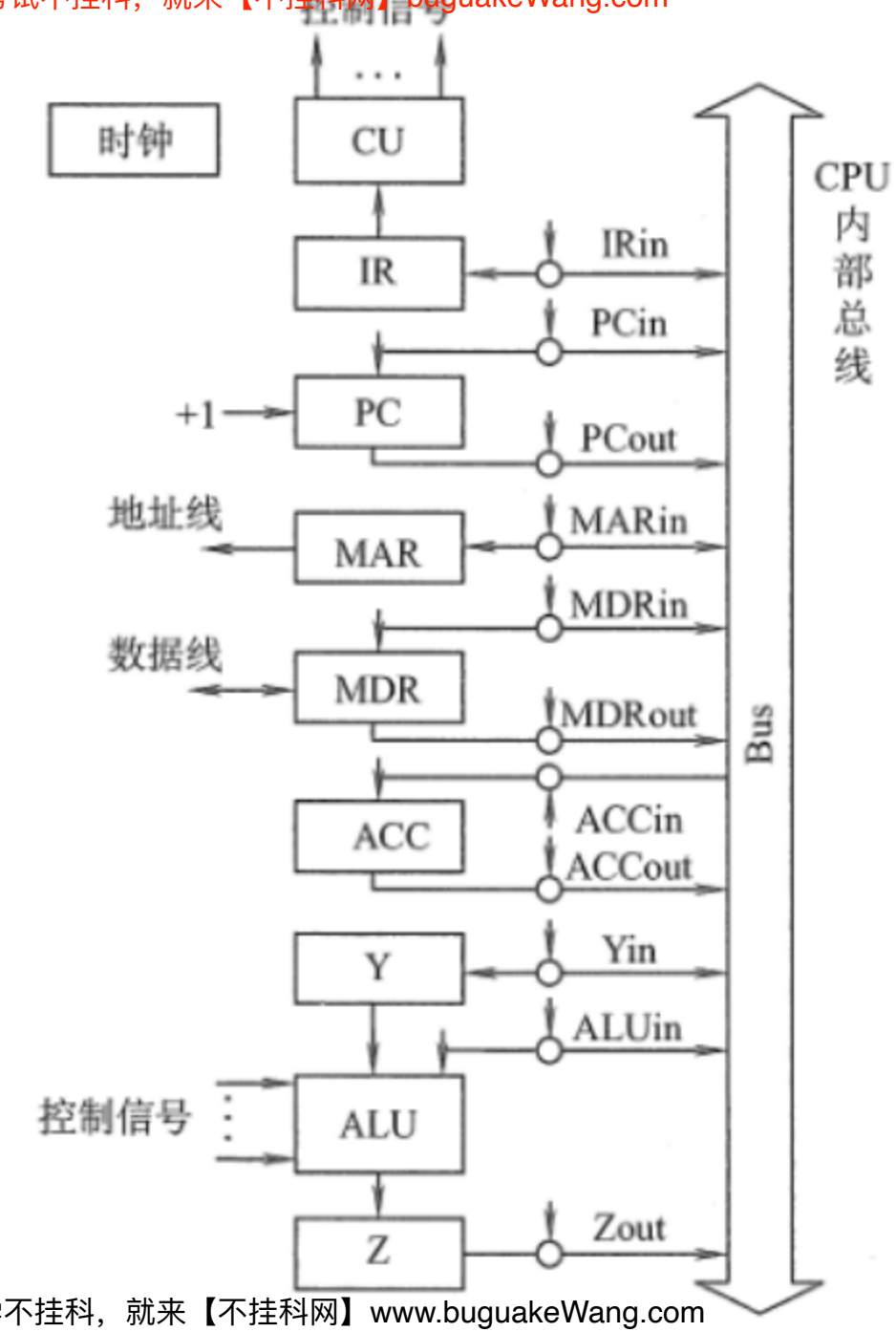
这种结构简单，但数据传输存在较多的冲突现象，性能较低。

连接各部件的总线只有一条时，称为单总线结构；

CPU中有两条或更多的总线时，构成双/多总线结构.

图示为CPU内部总线的数据通路和控制信号。

# 1. 数据通路的基本结构



# 1. 数据通路的基本结构

2）CPU内部三总线方式将所有寄存器的输入端和输出端都连接到多条公共通路上，

相比之下单总线中一个时钟内只允许传一个数据，

因而指令执行效率很低，

因此采用多总线方式，同时在多个总线上传送不同的数据，提高效率。

3）专用数据通路方式。

根据指令执行过程中的数据和地址的流动方向安排连接线路，

避免使用共享的总线，性能较高，但硬件量大。

# 1. 数据通路的基本结构

图中，规定各部件用大写字母表示，

字母加“in”表示该部件的允许输入控制信号；

字母加“out”表示该部件的允许输出控制信号。

注意：内部总线是指同一部件，

如CPU内部连接各寄存器及运算部件之间的总线；

系统总线是指同一台计算机系统的各部件，

如CPU、内存、通道和各类I／O接口间互相连接的总线.

# 1. 数据通路的基本结构

## 一. 寄存器之间的数据传送

寄存器之间的数据传送可通过CPU内部总线完成。

在上图中，某寄存器AX的输出和输入分别由AXout和AXin控制。

以PC寄存器为例，把PC内容送至MAR，实现传送操作的流程及控制信号为

PC→Bus　　PCout有效，PC内容送总线

Bus→MAR　　MARin有效，总线内容送MAR

# 1. 数据通路的基本结构

## 二.主存与CPU之间的数据传送

主存与CPU之间的数据传送也要借助CPU内部总线完成。

现以CPU从主存读取指令为例说明数据在数据通路中的传送过程。

实现传送操作的流程及控制信号为

| | |
|---|---|
| PC→Bus→MAR | PCout和MARin有效，现行指令 |
| 1→R | CU发读命令 |
| MEM(MAR)→MDR | MDRin 有效 |
| MDR→Bus→IR | MDRout和IRin有效，现行指令→IR |

# 三.执行算术或逻辑运算

执行算术或逻辑操作时，由于ALU本身是没有内部存储功能的组合电路，

因此如要执行加法运算，相加的两个数必须在ALU的两个输入端同时有效。

图中的暂存器Y即用于该目的。

先将一个操作数经CPU内部总线送入暂存器Y保存，

Y的内容在ALU的左输入端始终有效，

再将另一个操作数经总线直接送到ALU的右输入端。

这样两个操作数都送入了ALU，运算结果暂存在暂存器Z中。

# 1. 数据通路的基本结构

| | |
|---|---|
| Ad(IR)→Bus→MAR | MDRout和MARin有效 |
| 1→R | CU发读命令 |
| MEM→数据线→MDR | 操作数从存储器→数据线→MDR |
| MDR→Bus→Y | MDRout和Yin有效，操作数→Y |
| (ACC)+(Y)→Z | ACCout和ALUin有效，CU向ALU发加命令，结果→Z |
| Z→ACC | Zout和ACCin有效，结果→ACC |

数据通路结构直接影响 CPU 内各种信息的传送路径，数据通路不同，

指令执行过程的微操作序列的安排也不同，它关系着微操作信号形成部件的设计。

# 5.4 指令流水线

## 1. 指令流水线的基本概念

### 计算机的流水线

把一个重复的过程分解为若干子过程，每个子过程与其他子过程并行执行。

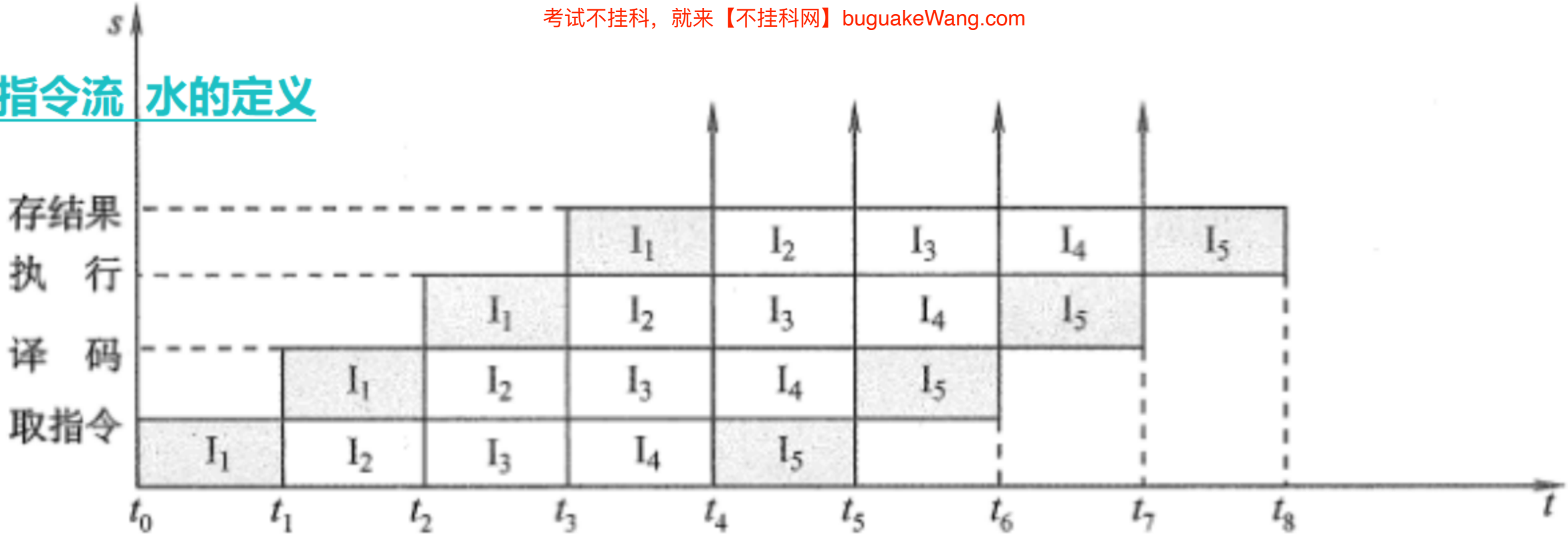
只需增加少量硬件就能把计算机的运算速度提高几倍，成为计算机中普遍使用的技术。

### 一. 指令流水的定义

通常用时空图来直观地描述流水线的工作过程，如图所示。

在时空图中，横坐标表示时间，即输入流水线中的各个任务在流水线中所经过的时间。流水线中各个流水段的执行时间都相等时，横坐标就被分割成相等长度的时间段。纵坐标表示空间，即流水线的每个流水段（对应各执行部件）。

# 一.指令流水的定义




图中，第一条指令I1在时刻to进入流水线，在时刻t4流出流水线。

第二条指令I2在时刻t1进入流水线，在时刻ts流出流水线。

以此类推，每经过一个Δt时间，便有一条指令进入流水线，

从时刻t4开始有一条指令流出流水线。

## 一.指令流水的定义

图中可看出，当t8 = 8Δt时，流水线上便有5条指令流出。

若采用串行方式执行指令，当t8 = 8Δt时，只能执行2条指令，

可见使用流水线方式成倍地提高了计算机的速度。

## 二.流水线方式的特点

与传统的串行执行方式相比，采用流水线方式具有如下特点：

1）把一个任务（一条指令或一个操作）分解为几个有联系的子任务，每个子任务由一个专门的功能部件来执行，并依靠多个功能部件并行工作来缩短程序的执行时间。

## 1. 指令流水线的基本概念

2）流水线每个功能段部件后面都要有一个缓冲寄存器，或称锁存器，其作用是保存本流水段的执行结果，供给下一流水段使用。

3）流水线中各功能段的时间应尽量相等，否则将引起堵塞、断流。

4）只有连续不断地提供同一种任务时才能发挥流水线的效率，所以在流水线中处理的必须是连续任务。在采用流水线方式工作的处理机中，要在软件和硬件设计等多方面尽量为流水线提供连续的任务。

5）流水线需要有装入时间和排空时间。装入时间是指第一个任务进入流水线到输出流水线的时间。排空时间是指最后一个任务进入流水线到输出流水线的时间。

## 2. 影响流水线的因素

影响流水线性能的因素主要有两方面：资源冲突和相关问题。

由于多条指令在同一时刻争用同一资源而形成的冲突称为资源冲突，有两种解决办法：

1）前一指令访存时，使后一条相关指令（以及其后续指令）暂停一个时钟周期。

2）单独设置数据存储器和指令存储器，使两项操作各自在不同的存储器中进行。

扫码听课，视频讲解更清晰

## 2. 影响流水线的因素

相关问题：流水线中的相关问题是指相邻指令间存在某种关联，使指令流水线出现停顿，降低了流水线的效率。主要有数据相关和控制相关两类。

### 一.数据冲突（数据冒险）

在一个程序中，下一条指令会用到这一条指令计算出的结果，此时这两条指令即为数据冒险。当多条指令重叠处理时就会发生冲突，解决的办法有以下几种：

1）把遇到数据相关的指令及其后续指令都暂停一至几个时钟周期，直到数据相关问题消失后再继续执行，可分为硬件阻塞（stall）和软件插入“NOP”指令两种方法。

## 一.数据冲突（数据冒险）

2）设置相关专用通路，即不等前一条指令把计算结果写回寄存器组，下一条指令也不再读寄存器组，而直接把前一条指令的ALU的计算结果作为自己的输入数据开始计算过程，使本来需要暂停的操作变得可以继续执行，这称为数据旁路技术。

3）通过编译器对数据相关的指令编译优化的方法，调整指令顺序来解决数据相关。

## 二.控制冲突（控制冒险）

一条指令要确定下一条指令的位置，例如在执行转移、调用或返回等指令时会改变PC值，而造成断流，会引起控制冒险。解决的办法有以下几种：

1）对转移指令进行分支预测，尽早生成转移目标地址。分支预测分为简单（静态）预测和动态预测。静态预测总是预测条件不满足，即继续执行分支指令的后续指令。动态预测根据程序执行的历史情况，进行动态预测调整，有较高的预测准确率。

2）预取转移成功和不成功两个控制流方向上的目标指令。

3）加快和提前形成条件码。

4）提高转移方向的猜准率。

## 2. 影响流水线的因素

注意：Cache缺失的处理过程也会引起流水线阻塞。

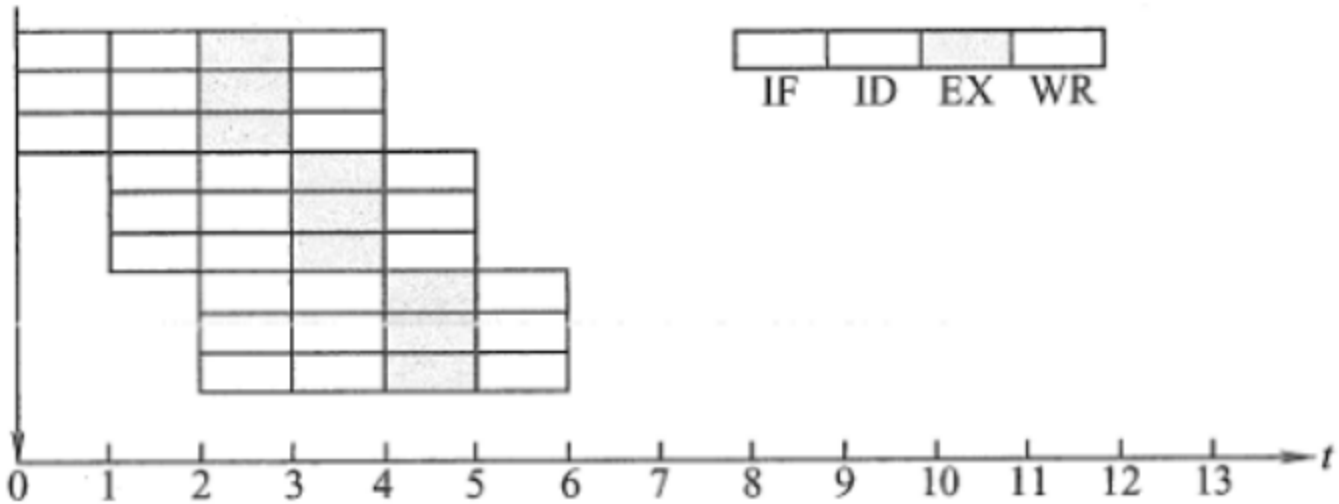
在不过多增加硬件成本的情况下，如何尽可能地提高指令流水线的运行效率是选用指令流水线技术必须解决的关键问题。

# 3. 超标量流水线的基本概念

## 一.超标量流水线技术

每个时钟周期内可并发多条独立指令，即以并行操作方式将两条或多条指令编译并执行，为此需配置多个功能部件。

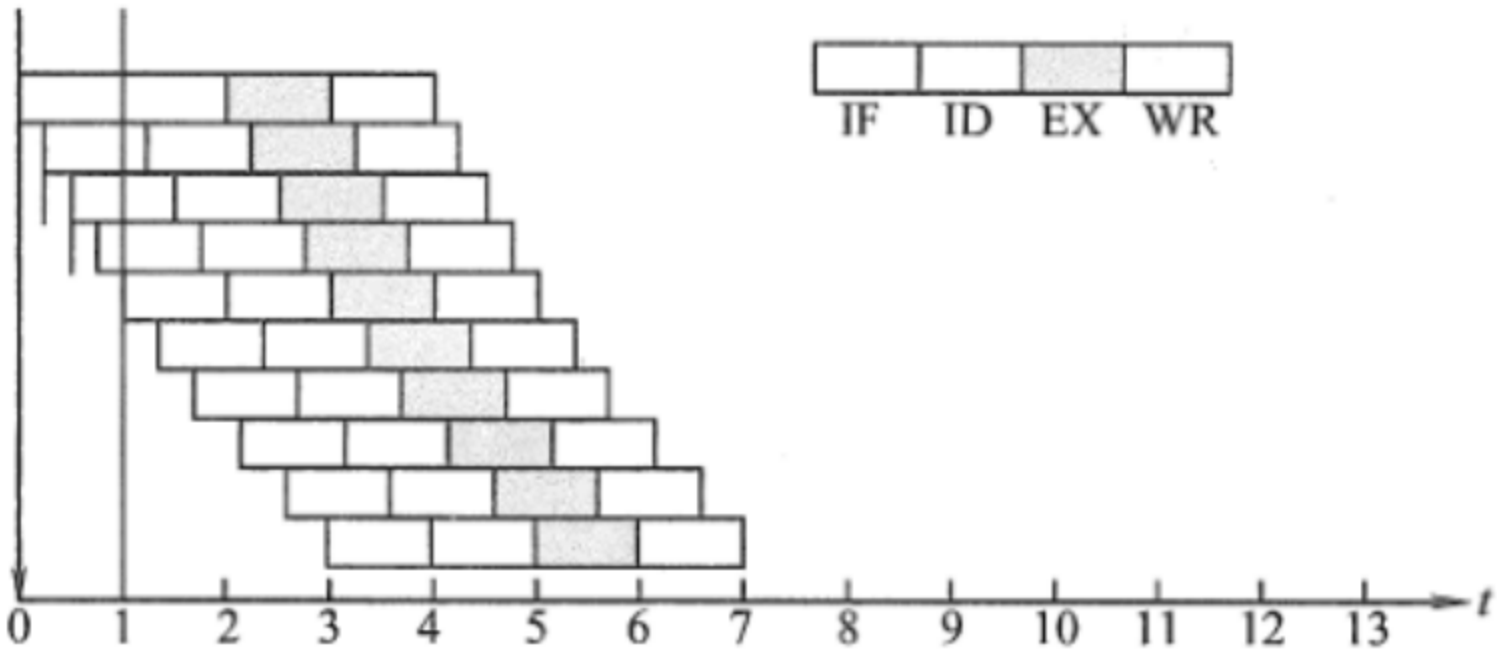
超标量计算机不能调整指令的执行顺序，因此通过编译优化技术，把可并行执行的指令搭配起来，挖掘更多的指令并行性，如图所示。



# 3. 超标量流水线的基本概念

## 二.超流水线技术

在一个时钟周期内再分段，在一个时钟周期内一个功能部件使用多次。不能调整指令的执行顺序，靠编译程序解决优化问题，如图所示。



# 3. 超标量流水线的基本概念

## 三.超长指令字

由编译程序挖掘出指令间潜在的并行性，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），为此需要采用多个处理部件。

【题1】下列寄存器中，汇编语言程序员可见的是（ B ）。

A. 存储器地址寄存器（MAR）

B.程序计数器（PC）

C.存储器数据寄存器（MDR）

D. 指令寄存器（IR）

解析：汇编语言程序员可见的是程序计数器（PC），即汇编语言程序员通过汇编程序可以对某个寄存器进行访问。汇编程序员可以通过指定待执行指令的地址来设置PC的值，如转移指令、子程序调用指令等。而IR、MAR、MDR是CPU的内部工作寄存器，对程序员不可见。

注意：CPU内部寄存器大致可分为两类：一类是用户可见的寄存器，可对这类寄存器编程，如通用寄存器组、程序状态字寄存器；另一类是用户不可见的寄存器，对用户是透明的，不可对这类寄存器编程，如存储器地址寄存器、存储器数据寄存器、指令寄存器。

【题2】下间址周期结束时，CPU内寄存器MDR中的内容为（ B ）。

A.指令　　B. 操作数地址　　C.操作数　　D. 无法确定

---

解析：间址周期的作用是取操作数的有效地址，因此间址周期结束后，MDR中的内容为操作数地址。

扫码听课，视频讲解更清晰