

第一章 概述

基本概念： 存储程序
冯.诺依曼体制
总线及组成
接口及组成

1.1964年,DJS-104,参与

2.上世纪六十年代末，第二代晶体管，441-B，
DJD-130

3.上世纪七、八十年代，第三代高速计算机，保
密通信机930，JKX-351工控机

第二章 计算机中的信息表示

- * 重点：1.进位计数制及其相互转换（二—八—十—十六进制间的转换）
- * 2.机器数
- * 3.IEEE754标准浮点表示格式
- * 4.能直接被CPU访问的操作数存放位置
- * 5.寻址方式

续前



2.1数值型数据的表示方法

一个数值型数据的完整表示需三个方面：

续前

- * 1.进位计数制:权、基数
- * 2.符号的数字化（机器数）:真值、机器数（原码、反码、补码）
- * 3.小数点的处理：定点、浮点

续前

* 4. IEEE754标准浮点格式

- * IEEE754有3种浮点表示格式，分别称为：
- * 短浮点数(或称短实数、单精度实数)
- * 长浮点数（或称长实数、双精度实数）
- * 临时浮点数（或称临时实数、扩展精度实数）
- * 它们的具体格式如表所示。

续前

* IEEE754的3种浮点表示格式

类型	数符 (位)	阶码 (位)	尾数数 值 (位)	总位数 (位)	偏 置 值	
					十六进 制	十进制
短浮点	1	8	23	32	7FH	127
数	1	11	52	64	3FFH	1023
长浮点	1	15	64	80	3FFFH	16383
数						
临时浮 点数						

续前

- * 32位短浮点数表示：
- * 公式法： $(-1)^s \times 1.M \times 2^{E-127}$
- * 指令、指令系统、指令与硬件电路之间的关系
- * 指令格式：

操作码	地址码
-----	-----

- * 1.指令字长
- * 2.操作码结构：定长、扩展、复合型

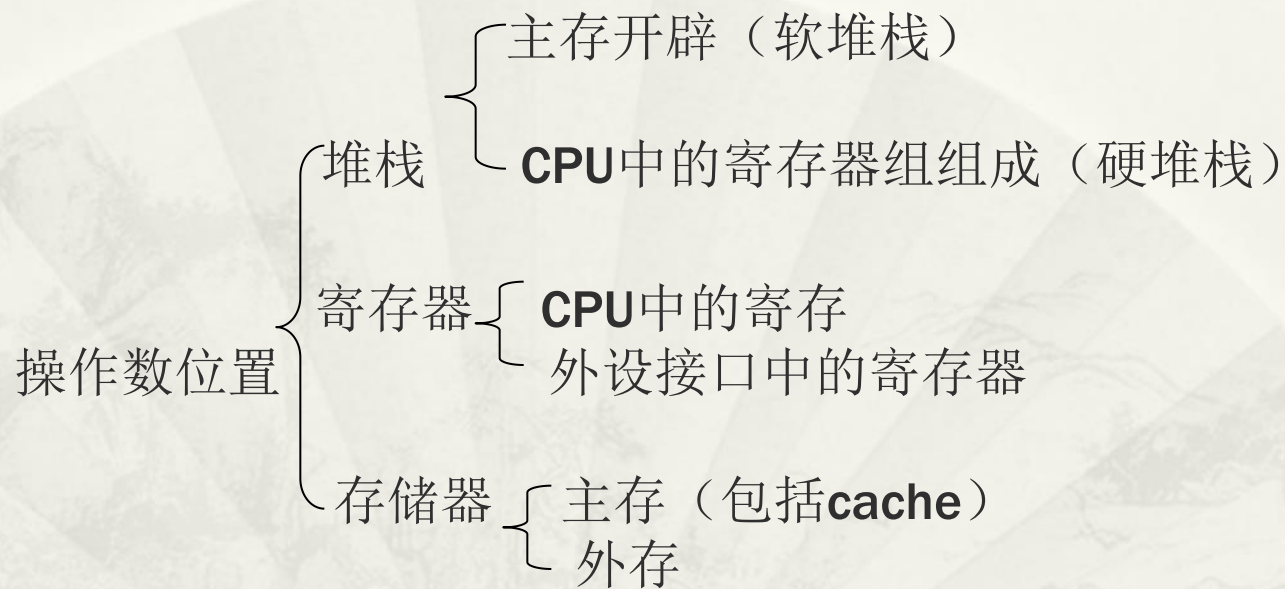
续前

3. 地址结构：显地址方式、隐地址方式

4. 寻址方式：寻找操作数地址或操作数的方式；

四类寻址：立即寻址、直接类寻址、间接类寻址、变址类寻址

指令中的地址结构



CPU能直接访问
的操作数位置

主存（包括**cache**、外设接口中的寄存器）

CPU中的寄存器

结论：①**CPU**能够直接访问的操作数只能存放在主存储器或**CPU**内的寄存器中，②由于主存储器的容量远远大于**CPU**内的寄存器的容量，因此，**CPU能够直接访问的操作数主要存放在主存储器中。**

指令给出操作数地址方式 { 显式：直接、间接、变址、基址等
隐式：隐含约定寄存器号、主存储器单元号

显地址：如果在指令中明显地给出地址,如写明主存储单元号或**CPU**的寄存器编号,则这种地址表达称为显地址。显地址又分为：三地址指令、二地址指令、一地址指令、零地址指令。

隐地址：如果在指令中的地址是以隐含的方式约定，如系统事先隐含约定操作数在**CPU**某个寄存器中,或是在堆栈中,因此在指令中并不给出地址码,这种隐含约定的地址就称为隐地址。

简化地址结构的基本途径：尽量使用隐地址。

寻址方式

寻址方式归纳为以下四大类，其它的寻址方式则是它们的变型或组合。

- ① 立即寻址。在读取指令时也就从指令之中获得了操作数，即操作数包含在指令中。（操作数在指令中，不需要访问主存，获取操作数速度最快）
- ② 直接寻址类。直接给出主存地址或寄存器编号，从**CPU**内或主存单元内读取操作数。（**R**:操作数在**CPU**的**R**中，不需要访问主存，获取操作数速度很快；**M**: 操作数在主存中，需要访问一次主存，获取操作数速度较慢）
- ③ 间接寻址类。先从某寄存器中或主存中读取地址，再按这个地址访问主存以读取操作数。（**(R)**、**(R) +**、**— (R)**:操作数在主存中，需要访问一次主存，获取操作数速度较慢；**(M)**: 操作数在主存中，需要访问两次主存，获取操作数速度很慢）
- ④ 变址类。指令给出的是形式地址（不是最终地址），经过某种变换（例如相加、相减、高低位地址拼接等），才获得有效地址，据此访问主存储器以读取操作数。（**X(R)**: 操作数在主存中，需要访问一次主存，获取操作数速度较慢； 实际操作与定义有变化）

一、直接类寻址：两种

1.寄存器寻址（寄存器直接寻址）方式：R

模型机中7个通用寄存器编号： **$R0=000$, $R1=001$, $R2=010$, $R3=011$, $SP=100$, $PSW=101$, $PC=111$**

寻址过程：寄存器号 $\xrightarrow{R_i}$ 操作数

操作数**S**与寄存器**R_i** 的关系为： **$S = (R_i)$**

2.直接寻址（主存直接寻址）方式：M

寻址过程：操作数地址 \xrightarrow{M} 操作数

操作数**S**与地址码**A**的关系为： **$S = (A)$**

主存储器直接寻址与寄存器寻址方式的比较：

a. 直接寻址是访问一次主存才能读取所需操作数；寄存器寻址是从**CPU**的寄存器中读取操作数，不需访问主存，所需时间大约是从主存中读数时间的几分之一到几十分之一，因而寄存器寻址比直接寻址快得多。

b. 由于寄存器数远少于主存储器的单元数，所以指令中存放寄存器号的字段位数也就大大少于存放主存地址码所需位数。采用寄存器寻址方式或其他以寄存器为基础的寻址方式（例如寄存器寻址、寄存器间址方式），可以大大减少指令中一个地址的位数，从而有效地缩短指令长度。这也使读取指令的时间减少，提高了工作速度。

注意：减少指令中地址数目与减少一个地址的位数是两个不同的概念。采用隐地址可以减少指令中地址的数目，而采用寄存器寻址方式、寄存器间址方式可以使指令中为给出一个地址所需的位数减少。

二、间接寻址及其变形：六种

1.间接寻址（主存间接寻址）方式：(M)

间址单元地址 \xrightarrow{M} 操作数地址 \xrightarrow{M} 操作数

操作数**S**与地址**A1**的关系为: **S= ((A1))**

2.寄存器间接寻址方式: (R)

寻址过程：寄存器号 $\xrightarrow{R_i}$ 操作数地址 \xrightarrow{M} 操作数

操作数**S**与寄存器**Ri** 的关系为: **S= ((Ri))**

3.自增型寄存器间址方式: (R)+

寻址过程：寄存器号 $\xrightarrow{R_i}$ 操作数地址 \xrightarrow{M} 操作数
└─→ **R_i** 内容加**1**

操作数**S**与寄存器**Ri** 的关系为: **S= ((Ri))**

4.自減型寄存器間址方式: $-(R)$

寻址过程：寄存器号 $\xrightarrow{R_i}$ 操作数地址 = $(R_i) - 1 \xrightarrow{M}$ 操作数

操作数**S**与寄存器**Ri** 的关系为: **S**= ((Ri - 1))

1) 压栈: 寄存器号 $\xrightarrow{\text{SP}}$ 操作数地址 = **(SP) - 1** $\xrightarrow{\text{M}}$ 操作数

2) 出栈: 寄存器号 $\xrightarrow{\text{SP}}$ 操作数地址 $\xrightarrow{\text{M}}$ 操作数

1) 寄存器多重间址:

寄存器号 $\xrightarrow{R_i}$ 一级间址 \xrightarrow{M} \xrightarrow{M} 操作数

2) 存储器多重间址:

一级间址 \xrightarrow{M} 二级间址 \xrightarrow{M} \xrightarrow{M} 操作数

1.变址寻址: $X(R)$

寻址过程可描述为:

形式地址 **D** $\xrightarrow{\hspace{2cm}}$ $\left. \begin{array}{l} \text{变址寄存器号} \xrightarrow{R_x} \text{变址量 } N \end{array} \right\} \rightarrow A = D + N \xrightarrow{M} \text{操作数}$

操作数**S**与形式地址**D**、变址寄存器**R_x**的关系为：**S**=（（**R_x**）+ **D**））

指令类型

按指令功能分类：传送指令、输入/输出（I/O）指令、算术运算指令、逻辑运算指令、程序控制类指令、处理机控制类指令等。

1. 传送类指令

1) 一般传送指令

源地址 $\xrightarrow{\text{数}}$ 目的地址

在具体设置传送指令时，一般应当对以下三个方面做出说明：

- ① 传送范围，即指令允许数据在什么范围内传送。操作数的来源与目的地主要是：CPU寄存器、主存储器。
- 2) 传送单位：即数据可以按字节、字、双字或数组为单位进行传送。
- 3) 设置寻址方式：本书模型机的0型、1型、2型、3型、4型、5型、6型等。

2. 输入/输出（I/O）指令

从广义的角度来看，I/O指令也是一种传送指令，只是传送范围的一方固定为输入/输出（I/O）设备，如键盘、鼠标、显示器、打印机等。

外围设备编址

访问外围设备，首先要知道其地址，这就涉及到对设备的编址问题。对外围设备的编址方法一般有两类三种。

- ① 对外围设备单独编址：a. 单独编址到设备级，是为每台外围设备分配一个设备码；
b. 单独编址到寄存器级：现在普遍采用的方法是：为各I/O接口中的有关寄存器分配一

种I/O端口地址，即编址到寄存器一级。

②外围设备与主存储器统一编址

统一编址到寄存器级,具体做法是将每个外围设备接口中的有关寄存器视作一个主存单元,分配一个存储单元地址(总线地址)。一个接口视其需要可占用一个或多个总线地址,根据指令给出的地址码,可以判明是访问主存还是访问外围设备,是访问哪一个接口的哪一个寄存器。

单独编址与统一编址的比较

	单独编址方式	统一编址方式
优点缺点	I/O指令和传送指令容易区分,外设地址线少,译码简单,主存空间不会闲置	可用传送指令代替专用I/O指令,通过地址总线访问外设接口中的寄存器(如同通过地址总线访问主存单元一样)
优点缺点	控制类总线中增加了I/O Read 和 I/O Write 信号线	接口中的寄存器占用主存一部分地址,减少了主存的可用空间

I/O指令的设置方法

通常有三类常见的I/O指令设置方法，一台计算机可以选取其中的一种或数种。

① 设置专用的I/O指令

如果外围设备单独编码〈设备码、或端口地址〉，指令系统中有专门的I/O指令，这是明显的，又称为显式I/O指令。

② 采用通用的数据传送指令实现I/O操作

若将外围设备接口中的寄存器与主存单元统一编址，由同样的总线地址访问。这样，外设便可看作是总线地址所覆盖的存储空间的一部分，因而主机可以通过传送指令用相同的方式访问存储器和外围设备。对存储器的各种寻址方式同样适合于对外设的寻址，使编制程序灵活方便。这种I/O指令是隐含在传送指令之中的，所以又称为隐式I/O指令。

③ 通过I/O处理器（或I/O处理机）控制I/O操作

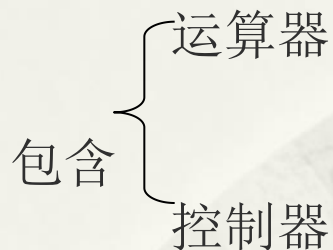
控制/状态字格式基本格式:

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

出错	故障		忙					完成	允许中断				维护	校验	启动
----	----	--	---	--	--	--	--	----	------	--	--	--	----	----	----

第三章 中央处理器

CPU是计算机系统的核心部件：



通过本章的学习，应在CPU一级上建立起整机的概念，它包含两个方面的内容：

- 1、CPU的逻辑组成；内部有哪些部件，以数据通路为核心的总体结构，与外部的连接。
2. CPU如何工作，即如何分时形成控制命令序列，以执行指令序列，分为：
 - (1) 从寄存器传送级分析指令的执行流程；
 - (2) 从微操作命令序列一级分析寄存器级传送的具体实现。

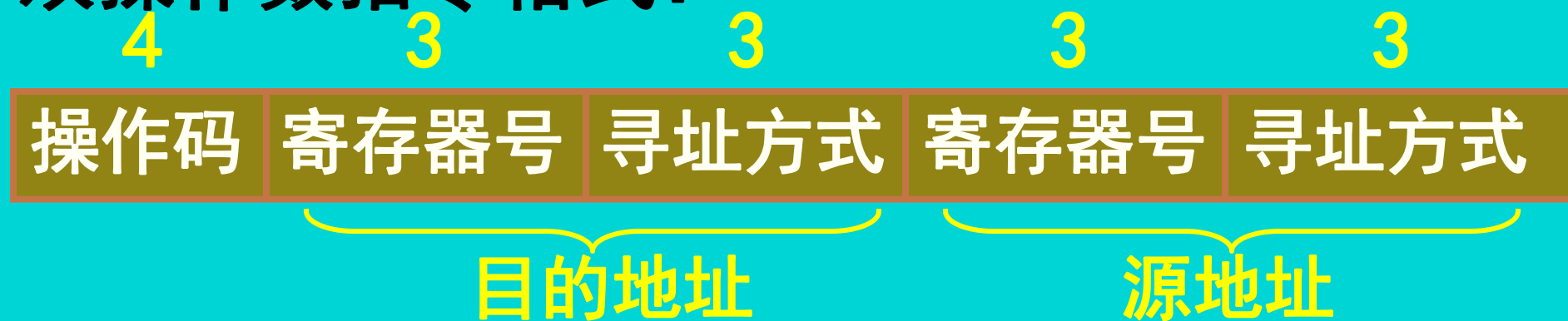
模型机的总体设计

模型机指令系统

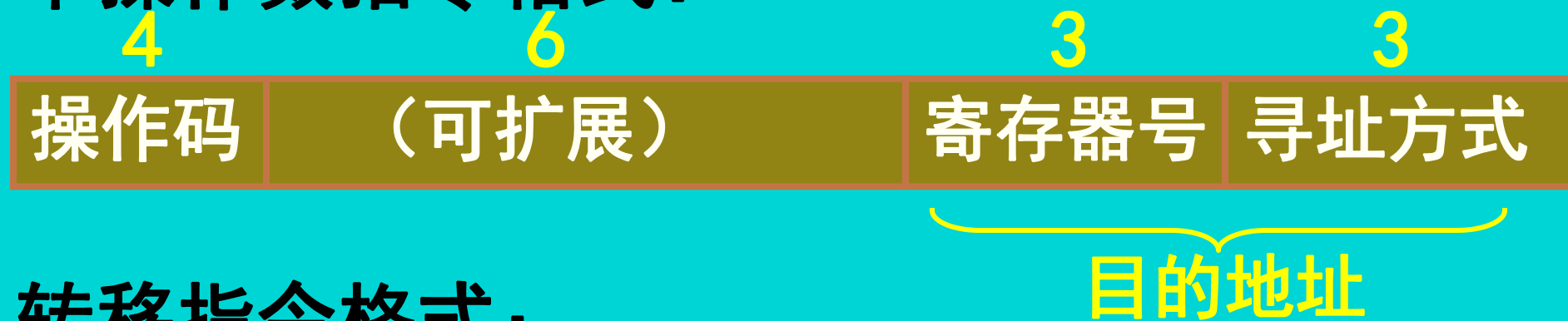
一、指令格式：指令字长16位，采用寄存器型寻址，指令中给出寄存器号。

三种类型指令格式：

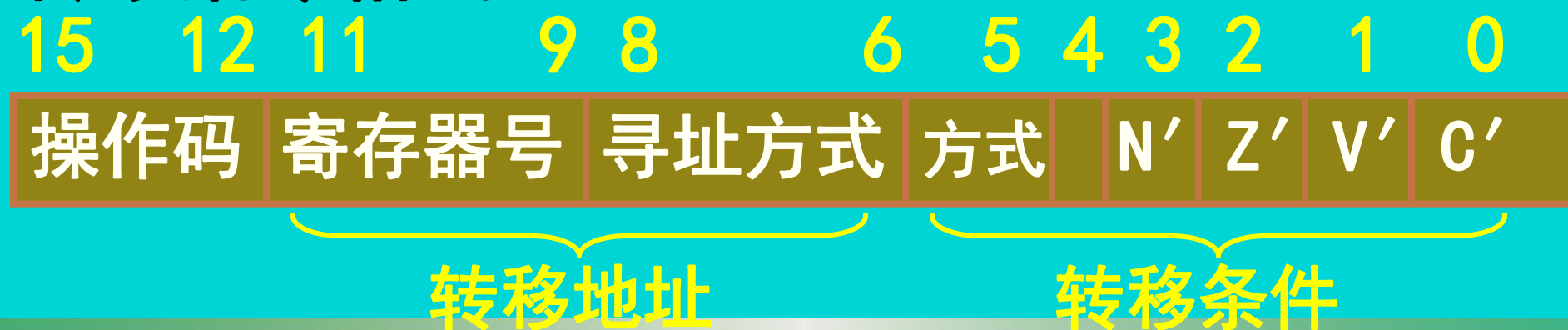
双操作数指令格式:



单操作数指令格式:



转移指令格式:



二、操作类型

操作码4位，设置15种指令

- 1、传送指令：一种
- 2、双操作数算逻指令：五种
- 3、单操作数算逻指令：六种
- 4、程序控制类指令：三种，但其中两种操作码相同

三、寻址方式

CPU可编程访问的寄存器：通用寄存器 R_0 （000）、 R_1 （001）、 R_2 （010）、 R_3 （011）指令计数器PC（111）、堆栈指针SP（100）、程序状态字PSW（101）

寻址方式

助记符

可指定寄存器

- | | | |
|-------------|---------------------|-----------------|
| 1、立即寻址: | I | 无 |
| 2、寄存器寻址: | R | R0~R3、SP、PC、PSW |
| 3、寄存器间址: | (R) | R0~R3 |
| 4、自减型寄存器间址: | - (R) | R0~R3、SP |
| 5、自增型寄存器间址: | (R) + | R0~R3、SP、PC |
| 6、自增型双间址: | @ (R) +
@ (PC) + | R0~R3、SP、PC |
| 7、变址方式: | X (R)
X (PC) | R0~R3、PC |
| 8、跳步方式: SKP | | |

操作数寻址方式

寻址方式	操作数地址1（间址地址、形式地址的地址）	操作数地址2	操作数位置	操作数的访存次数
立即寻址			指令中	0
寄存器直接寻址：R		指令给出R编号（即操作数地址）	R中	0
R寄存器间接寻址：(R)		指令给出M单元号（即操作数地址）	M中	1
自增型寄存器间接寻址： (R)+、(SP)+	指令给出R编号（间接地址）	R中给出M单元号（即操作数地址）	M中	1
自减型寄存器间接寻址： —(R)、—(SP)	指令给出R编号	R中内容减1后，为M单元号（即操作数地址）	M中	1
自增型双间址：@ (R)+	指令给出R编号（间址地址）	从M中取回的操作数地址	M中	2
变址寻址：X(R)	指令给出M单元号（形式地址的地址）	指令给出R编号（即变址地址N），及从M中取回的形式地址D；运算后的M单元值 $A=D+N$ （即操作数地址）	M中	2
跳步：SKP				

CPU的组成及CPU内部数据通路结构

CPU组成

运算器：三级, 输入选择器/锁存器—>ALU—>移位器

寄存器：三组，用于处理、控制、作用于主存接口的寄存器

总线：四组，CPU内总线、系统总线、部件间总线，外总线CPU的组成

时序系统：一个脉冲源、一组计数分频逻辑

控制器：组合逻辑控制器/微程序控制器

一、运算器

分为三级

输入选择器/锁存

ALU

移位器（通过斜位传送实现移位功能）

二、寄存器设置

1. 用于处理的寄存器：

1) 通用寄存器组：

这是一组可编程访问的R，在指令系统中为这些R分配了编号，有：R0~R3，PC，SP，PSW。

2) 暂存器：

2. 用于控制的寄存器：C、D

1) 指令寄存器IR

2) 程序计数器PC

3) 程序状态字寄存器PSW

3. 用作主存接口的寄存器：

1) 地址寄存器MAR

2) 数据缓冲寄存器MDR

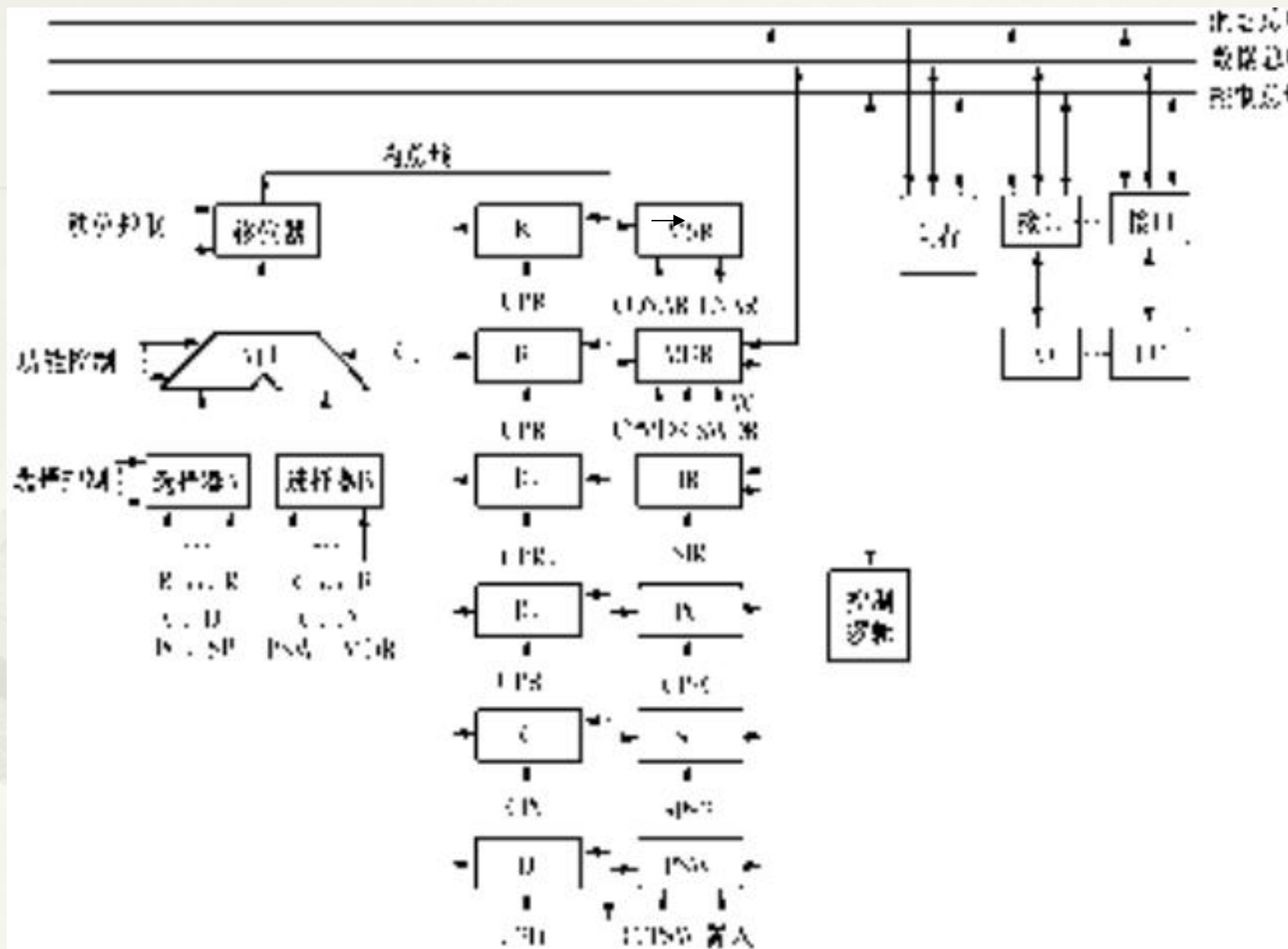
三、总线

分类 { CPU内总线
系统总线
部件间总线
系统外总线

四、时序系统

{ 振荡器
一组计数分频器

模型机数据通路框图



一、取指令地址及指令信息的传送

1、取指令地址：

$PC \rightarrow A \rightarrow ALU \rightarrow \text{移位器} \rightarrow \text{CPU内总线} \rightarrow \text{MAR(传送途径)}$, 简化后：

$PC \rightarrow \text{MAR}$

同时，后继指令地址：

$PC \rightarrow A \rightarrow ALU \text{ (加1)} \rightarrow \text{移位器} \rightarrow \text{CPU内总线} \rightarrow PC \text{ (传送途径)}$, 简化后： $PC+1 \rightarrow PC$

2、指令信息的传送：

$M \rightarrow \text{数总} \rightarrow IR$, 简化后： $M \rightarrow IR$

二、取操作数地址及操作数信息的传送(典型)

1. 寄存器→寄存器（CPU内的R）

$R_i \rightarrow A/B \rightarrow ALU \rightarrow \text{移位器} \rightarrow \text{CPU内总线} \rightarrow R_j \text{ (传送途径)}$, 简化后：

$R_i \rightarrow R_j$

2. 主存→寄存器

$M \rightarrow \text{数据总线DB} \rightarrow \text{MDR} \rightarrow B \rightarrow ALU \rightarrow \text{移位器} \rightarrow \text{CPU内总线} \rightarrow C \text{ (传送途径)}$, 简化后： $M \rightarrow \text{MDR} \rightarrow C$

微命令设置

将模型机的数据传送路径分为两大类操作,并分段设置有关的微命令:

一、数据通路操作

1.ALU输入选择:

$R_i \rightarrow A$, R_i 的取值: R_0-R_3, C,D,PC,SP ;
 $R_j \rightarrow B$, R_j 的取值: R_0-R_3,C,D,MDR,PSW 。

2.AUL功能选择:

S_0-S_3, C_0, M

3.移位器功能选择:

直传DM、左移、右移

4.分配脉冲（打入到寄存器中的脉冲）:

$CPR_0-CPR_3, CPC,CPD,CPMDR,CPMAR,CPPSW, CPPC,CPSP$

二、与系统线及主存有关的微命令

$EMAR,R,W,SIR,SMDR$

写入主存: $EMAR,W$

读入CPU:



S3 S2 S1 S0	M = 1 (逻辑运算)	M = 0 (算术运算)	
		C0=0 (无进位)	C0=1 (有进位)
0000	$F = \neg A$	A减1	A
0001	$F = \neg (A+B)$	AB减1	AB
0010	$F = \neg A + B$	A/B减1	A/B
0011	逻辑1	全1	0
0100	$F = \neg (A+B)$	A加 (A+/B)	A加 (A+/B) 加1
0101	$F = \neg B$	AB加 (A+/B)	AB加 (A+/B) 加1
0110	$F = \neg (A \oplus B)$	A加/B	A减B
0111	$F = (A+/B)$	A+/B	A+/B加1

S3 S2 S1 S0	M = 1 (逻辑运算)	M = 0 (算术运算)	
		C0=0 (无进位)	C0=1 (有进位)
1000	$F = \neg A \cdot B$	F=A加 (A+B)	F=A加 (A+B) 加1
1001	$F = A \oplus B$	F=A加B	F=A加B加1
1010	$F = B$	F=A/B加 (A+B)	F=A/B加 (A+B) 加1
1011	$F = A + B$	F=A+B	F=A+B加1
1100	逻辑0	0	1
1101	$F = A / B$	F=AB加A	F=AB加A加1
1110	$F = A \cdot B$	F=A/B加A	F=A/B加A加1
1111	$F = A$	F=A	F=A加1

常用算术逻辑运算微命令

	M	S3	S2	S1	S0	CO	
A+1	0	1	1	1	1	1	算 术 运 算
A-1	0	0	0	0	0	0	
A+B	0	1	0	0	1	0	
A-B	0	0	1	1	0	1	
传送A	1	1	1	1	1	0	逻 辑 运 算
传送B	1	1	0	1	0	0	

例如**1**：**FT**周期内的操作（既有数据通路操作，又有访存操作）

PC+1→PC：

PC→A→ALU（加1）→移位器→CPU内总线→PC (传送途径)

微命令：**PC →A A+1 DM CPPC**

M→IR：

M→数总→IR

EMAR, R, SIR

例如**2**：（既有数据通路操作，又有访存操作）

M→MDR→C：

M→数据总线DB→MDR→B→ALU→移位器→CPU内总线→C (传送途径)

微命令：**M→MDR：** **EMAR,R,SMDR(读入CPU)**

MDR→C： **MDR→B, B, DM, CPPC**

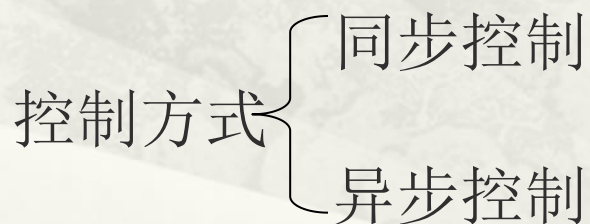
时序控制方式与时序系统

计算机的工作需要分步地执行，这就需要一种时间划分的信号标志——时序信号，以反映在什么时间段或时刻，计算机做了什么操作。

为了形成控制流，在时序方面有三个问题需要考虑：

- 1.操作与时序信号之间的关系，即时序控制方式。
- 2.指令之间的衔接方式；
- 3.如何形成所需的时序信号，即时序系统

一、时序控制方式



一、同步控制方式

- 1.定义：如果各项操作与统一的时序信号同步，称为同步控制。
- 2.时间分配（基本特征）：同步控制方式的基本特征是将操作时间划分为许多时钟周期，周期长度固定，每个时钟周期完成一步操作。
- 3.同步定时：在许多操作中需要严格地同步定时，如同步打入脉冲。
- 4.各部件间的协调：在**CPU**内，一般采用由**CPU**提供的统一时序信号来控制部件间信息的传送的。
- 5.特点：时序关系比较简单，在是时间安排利用上可能不经济的。

二、异步控制方式

- 1.定义：异步控制是指各项操作按其需要选择不同的时间，不受统一的时钟周期的约束；各操作之间的衔接与各部件之间的信息交换采取应答方式。
- 2.基本特征：没有统一的节拍划分与同步定时脉冲，但存在着申请、响应、询问、回答一类的应答关系。
- 3.主从设备的概念：申请使用总线，并获得批准后掌管总线控制权的设备，称为主设备，否则为从设备。
- 4.特点：时间紧凑，能按不同部件，设备的实际需要分配时间，实现异步应答所需的控制比较复杂。

同步控制的时序系统

一、定义及组成：

1.时序系统：产生节拍，脉冲等时序信号的部件，称为时序系统。

2.组成：
时序系统 { 一个脉冲源
 一组计数分频逻辑

二、时序划分层次：

1.指令周期：读取并执行一条指令所需的时间，称为指令周期。一般不作为时序的一级。

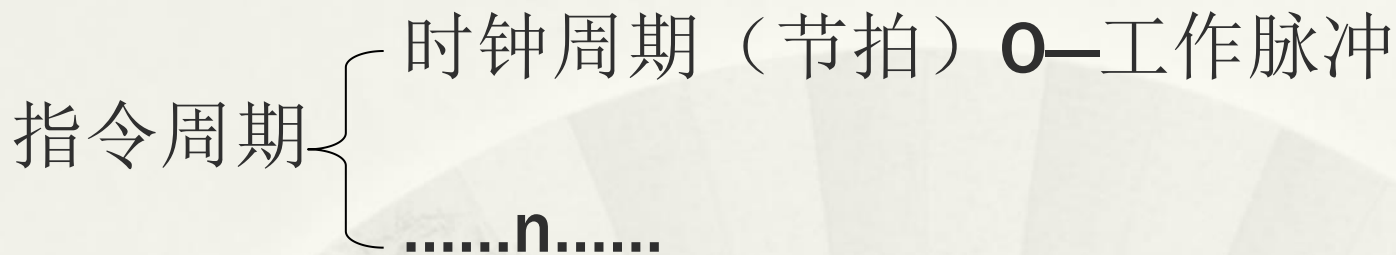
2.CPU工作周期：在指令周期中的某一工作阶段所需的时间，称为一个工作周期。一般不同。

3.时钟周期（节拍）：是时序系统中最基本的时间分段。各节拍的长度相同。

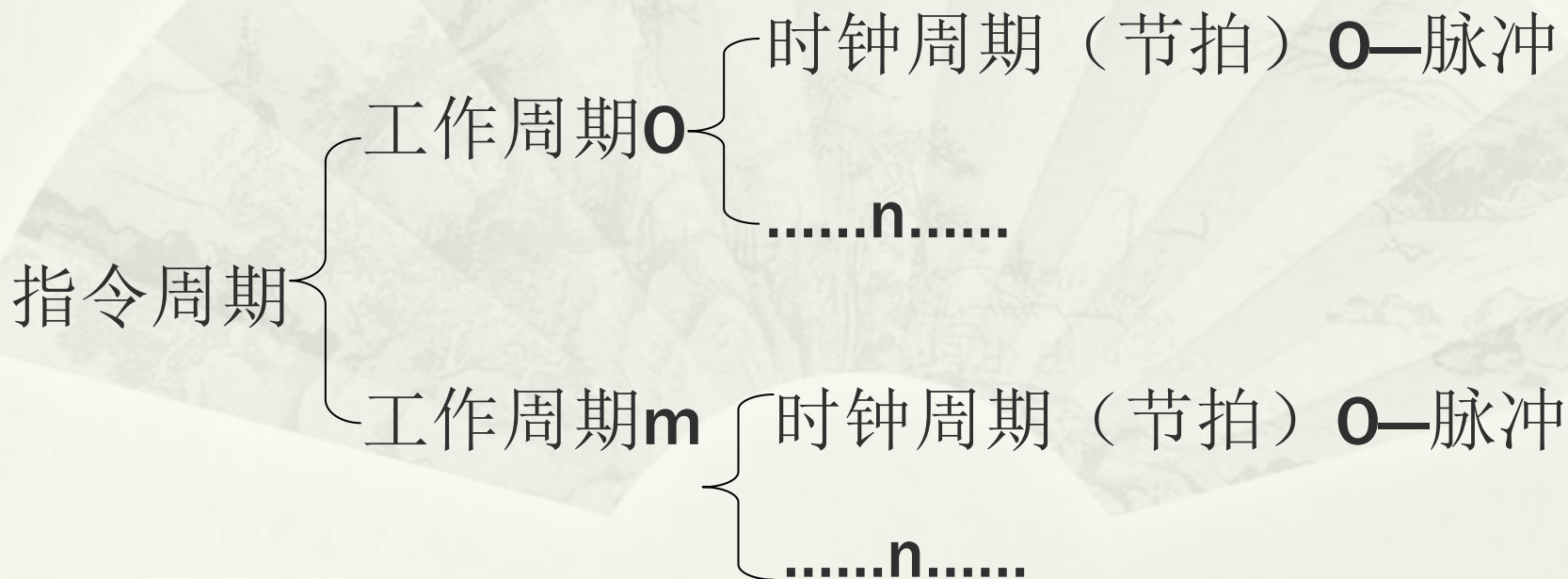
4.定时脉冲（工作脉冲）：有的操作如打入**R**，还需严格的定时脉冲，以确定在哪一刻打入。

三、多级时序的划分：

1. 二级时序（用在微程序控制器中）



2. 三级时序（用在组合逻辑控制器中）



模型机的组合逻辑控制器

指令流程与相应微命令序列的拟定主要取决于数据通路结构，所以两种控制器在这一部分的设计有不少内容是相似的，其区别主要在时序划分以及最后形成微命令的方式上

组合逻辑控制器概述

一、组合逻辑控制器定义

组合逻辑控制器的微命令是由组合逻辑电路来实现。每种微命令都需要一组逻辑电路，全机所有微命令所需的逻辑电路就构成了微命令发生器。

二、组合逻辑控制器的硬件组成

组成 { 程序计数器**PC**、指令寄存器**R**、状态寄存器**PSW**
时序系统
微命令发生器
译码器、地址形成部件等

三、组合逻辑控制器工作原理

组合逻辑控制器时序系统

组合逻辑控制器依靠不同的时间标志，使**CPU**分步工作，模型机按常规采用工作周期、时钟周期、工作脉冲三级时序。

1. 工作周期

模型机设置了六种工作周期状态，用六个周期状态触发器作为它们的标志。其中，四个工作周期（取指、源、目的、执行）用于指令的正常执行，两个工作周期（中断、**DMA**）用于**I/O**传送控制。某一时期内只有其中一个周期状态触发器为**1**，指明**CPU**现在所处的工作周期状态，为该阶段的工作提供时间标志与依据①取指周期**FT**：在**FT**中完成的操作是公共性操作，

② 源周期**ST**——如果需要从主存中读取源操作数（非寄存器寻址），则进入**ST**。

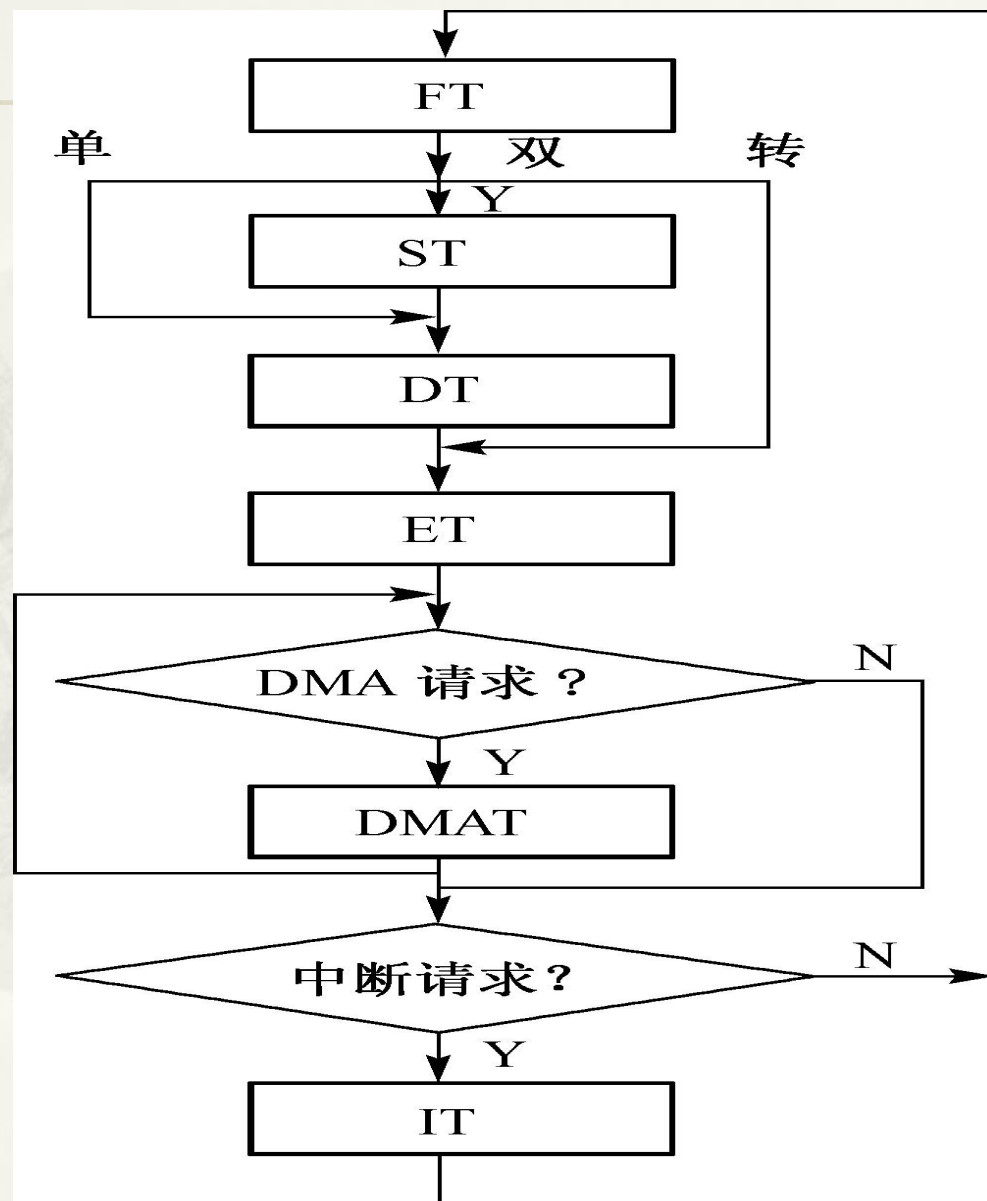
③ 目的周期**DT**——如果需要从主存中读取目的地址或目的操作数（非寄存器寻址），则进入**DT**。

④ 执行周期**ET**——取得操作数后，**CPU**进入**ET**，这也是各类指令都需经历的最后一个工作阶段。

⑤ 中断周期**IT**——除了考虑指令的正常执行，还需考虑外部请求带来的变化。在响应中断请求之后，到执行中断服务程序之前，需要一个过渡期，称为中断周期**IT**。

⑥ **DMA**周期**DMAT**——响应**DMA**请求之后，**CPU**进入**DMAT**。在**DMAT**中，**CPU**交出系统总线的控制权，即**MAR**、**MDR**与系统总线断开（呈高阻态），改由**DMA**控制器控制系统总线，实现主存与外围设备间的数据直传。

CPU控制流程各工作周期状态之间的转换示于图。



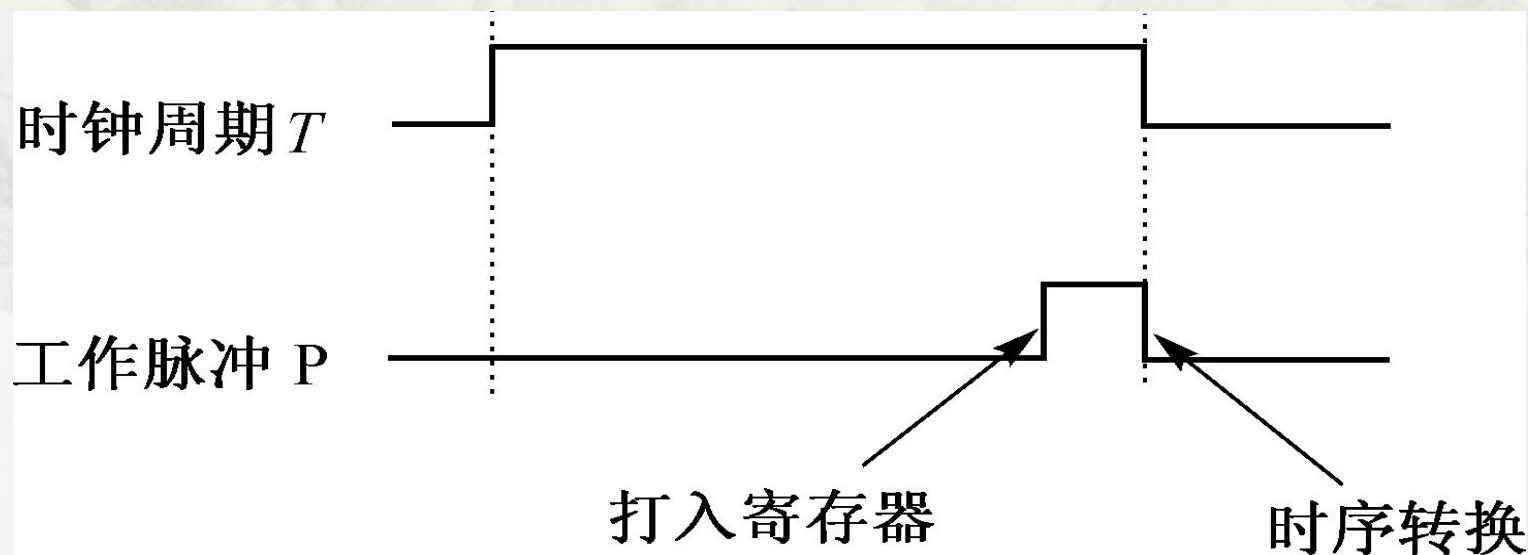
2. 时钟周期（节拍）

指令的读取与执行既有**CPU**内部数据通路操作，也包含访问主存的操作。为简化时序控制，模型机将两类操作周期统一起来，即以主存访问周期所需时间为时钟周期的宽度，这里设为**1**微秒。

3. 工作脉冲

有些操作需要同步定时脉冲进行控制，如将稳定的运算结果打入寄存器，又如进行周期状态切换。模型机在每个时钟周期的末尾发一个工作脉冲**P**，作为各种同步脉冲的来源，

工作脉冲**P**的前沿作为打入寄存器的定时，它标志着一次数据通路操作的完成。**P**的后沿作为时序转换的定时，在此刻如果本工作周期未结束，则对时钟周期计数器**T**计数，进入新的节拍；如果本工作周期结束，则将**T**清零，并清除本工作周期状态标志，设置新的工作周期状态标志。



指令流程与操作时间表

通过有关讨论应能较深入地了解**CPU**执行指令的工作机制。我们将分成两个层次进行讨论：

- ⊙ 在寄存器传送级拟定各类指令的执行流程，也就是确定指令执行的具体步骤，即各类信息如何分步地按要求流动。
- ⊙ 拟定操作时间表，即给出实现上述流程所需的微操作命令序列。其中包含维持一个时钟周期的电位型微命令，以及短暂的脉冲型微命令。操作时间表还将表明出现各种微命令的逻辑条件与时间条件。

通过这两种层次的工程化描述，我们就清晰地确定了**CPU**在什么时间、根据什么条件、发出什么命令、做什么事。

这里是以指令为线索，按指令类型分别拟定操作流程。这种方法的优点是对指令的执行过程拟出了清晰的线索，便于理解**CPU**的工作过程。

一、取指周期FT

1. 进入FT的方式和条件

- (1) 由S端异步置入：
a. 上电初始化
b. 复位初始化

(2) 运算过程中同步打入FT(由D端打入)：

- a. 程序正常执行，应进入FT；
b. DMA周期执行后，应进入FT；
c. IT周期执行后，应进入FT。

2. 取指流程

$M \rightarrow IR$, $PC+1 \rightarrow PC$

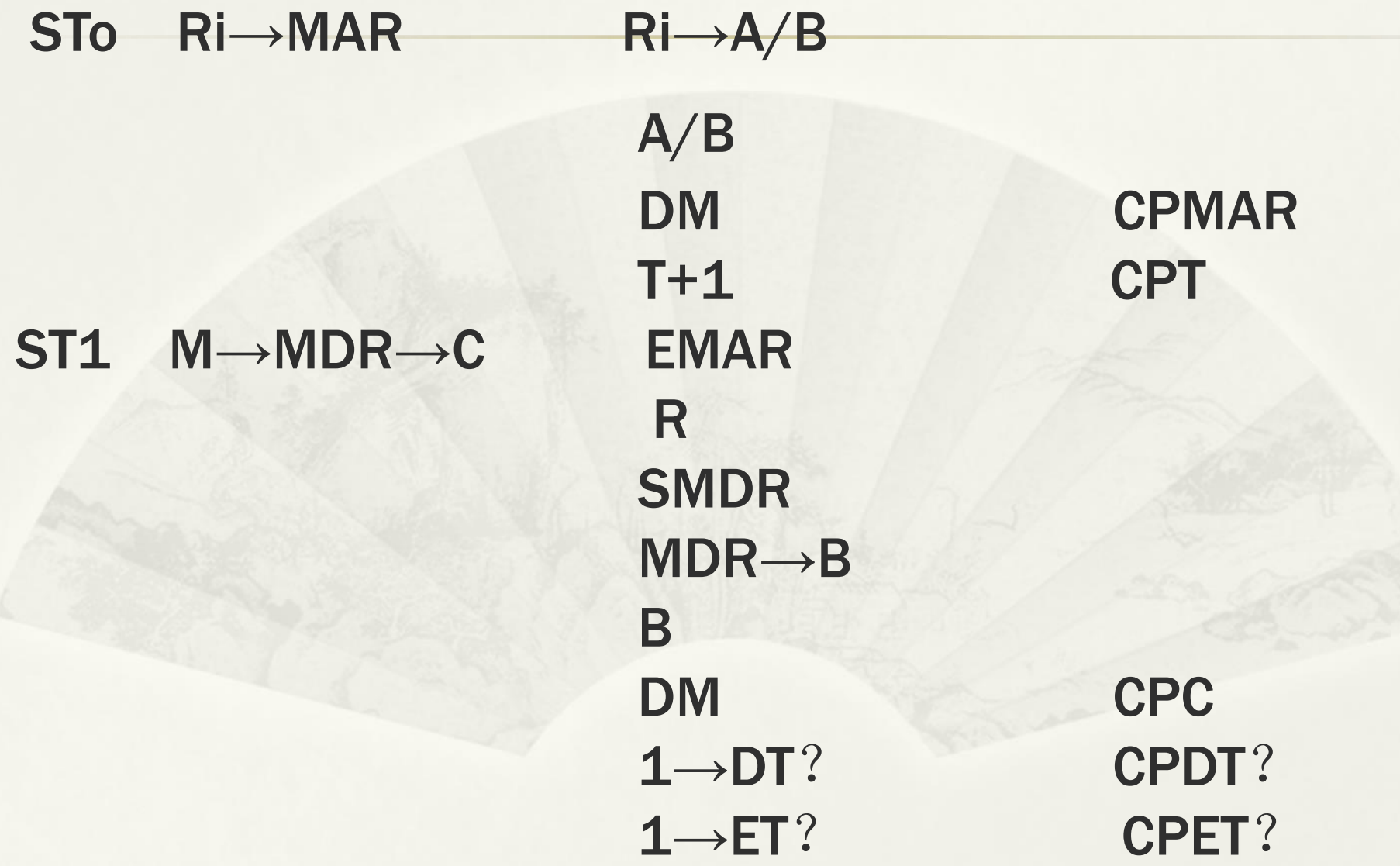
3. 操作时间表

EMAR, R , SIR ; $PC \rightarrow A$, $A+1$, DM ,CPPC ,

(1) $1 \rightarrow ST, CPST$; (2) $1 \rightarrow DT, CPDT$; (3) $1 \rightarrow ET, CPET$;

二、寻址方式

1、（R）型：



2、- (R) 型:

ST₀ Ri-1→MAR,Ri

电平

Ri→A/B

脉冲

A-1/B-1

DM

CPMAR,CPRi

T+1

CPT

ST₁ M→MDR→C

EMAR

R

SMDR

MDR→B

B

DM

CPC

1→DT?

CPDT?

1→ET?

CPET?

3、 (R) +型:

电平

脉冲

STo Ri→MAR

Ri→A/B

A/B

DM

CPMAR

T+1

CPT

ST1 M→MDR→C

EMAR

R

SMDR

MDR→B

B

DM

CPC

T+1

CPT

ST2 Ri+1→Ri

Ri→A/B

A+1

DM

CPRi

1→DT?

CPDT?

1→ET?

CPET?

4.

@ (R) +:

电平

脉冲

ST0 Ri→MAR

Ri→A/B

A/B

DM

T+1

EMAR

R

SMDR

MDR→B

B

DM

T+1

Ri→A/B

A+1

DM

T+1

CPMAR

CPT

ST1 M→MDR→C

CPC

CPT

ST2 Ri+1→Ri

CP Ri

CPT

ST3 **$C \rightarrow \text{MAR}$**

$C \rightarrow A/B$

A/B

DM

CPMAR

$T+1$

CDT

ST4 **$M \rightarrow \text{MDR} \rightarrow C$**

EMBR

R

SMDR

$\text{MDR} \rightarrow B$

B

DM

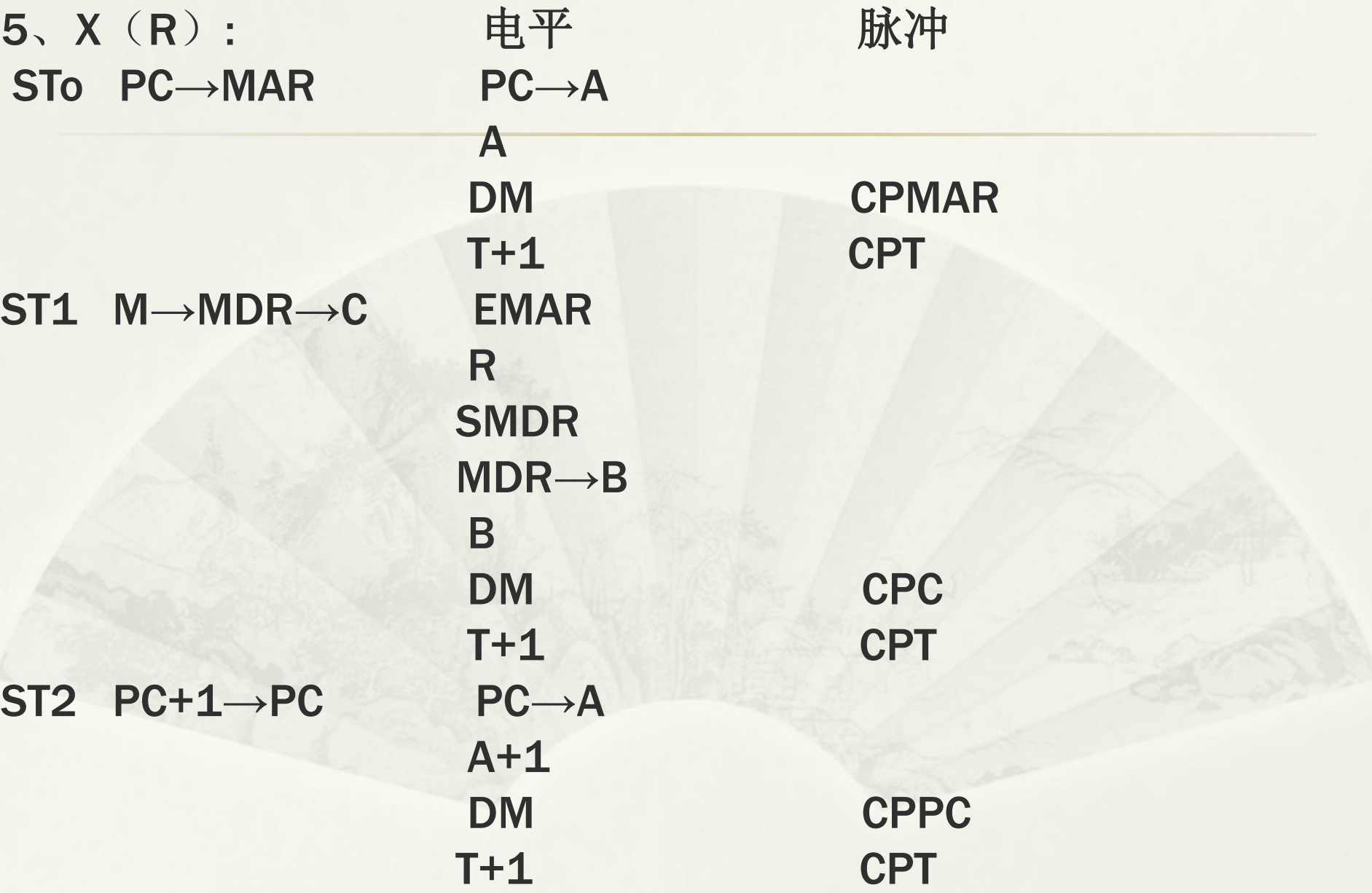
CPC

$1 \rightarrow \text{DT?}$

CPDT?

$1 \rightarrow \text{ET?}$

CPET?



ST3 $R_i + C \rightarrow \text{MAR}$

$R_i \rightarrow A/B, C \rightarrow B/A$

$A + B$

DM

$T + 1$

$EMBR$

R

$SMDR$

$MDR \rightarrow B$

B

DM

$1 \rightarrow DT?$

$1 \rightarrow ET?$

$CPMAR$

CPT

CPC

$CPDT?$

$CPET?$

ST4 $M \rightarrow MDR \rightarrow C$

ET周期中：

1.SR、DR: R(或立即寻址), 操作数不在主存中

**2./SR、 /DR: (R)、 $-(R)$ 、 $(R)+$ 、
@ $(R)+$ 、 $X(R)$, 操作数在主存中**

ET周期中: **MOV指令**

1.SR.DR

ET0: $R_i \rightarrow R_j$ $R_i \rightarrow A/B, A/B, DM, CPR_j, T+1, CPT$

ET1: $PC \rightarrow MAR$ $PC \rightarrow A, A, DM, CPMAR$

(1) $1 \rightarrow DMAT, CPDMAT;$

(2) $1 \rightarrow IT, CPIT;$

(3) $1 \rightarrow FT, CPFT$

2. SR./DR

ET0: $R_i \rightarrow MDR$ $R_i \rightarrow A/B, A/B, DM, CPMDR, T+1, CPT$

ET1: $MDR \rightarrow M$ $EMAR, /W, T+1, CPT$

ET2: $PC \rightarrow MAR$ $PC \rightarrow A, A, DM, CPMAR$

(1) $1 \rightarrow DMAT, CPDMAT;$

(2) $1 \rightarrow IT, CPIT;$

(3) $1 \rightarrow FT, CPFT$

3./SR.DR

ET0: $C \rightarrow R_j$

$C \rightarrow A/B, A/B, DM, CPR_j, T+1, CPT$

ET1: $PC \rightarrow MAR$

$PC \rightarrow A, A, DM, CPMAR$

(1) $1 \rightarrow DMAT, CPDMAT;$

(2) $1 \rightarrow IT, CPIT;$

(3) $1 \rightarrow FT, CPFT$

4. /SR./DR

ET0: $C \rightarrow MDR$

$C \rightarrow A/B, A/B, DM, CPMDR, T+1, CPT$

ET1: $MDR \rightarrow M$

$EMAR, /W, T+1, CPT$

ET2: $PC \rightarrow MAR$

$PC \rightarrow A, A, DM, CPMAR$

(1) $1 \rightarrow DMAT, CPDMAT;$

(2) $1 \rightarrow IT, CPIT;$

(3) $1 \rightarrow FT, CPFT$

ET周期中：双操作数指令

1.SR.DR

ET0: $R_i \text{ OP } R_j \rightarrow R_j$

$R_i \rightarrow A/B$ 且 $R_j \rightarrow B/A, A \text{ OP } B,$
DM, CPR_j, T+1, CPT

ET1: $PC \rightarrow MAR$

$PC \rightarrow A, A, DM, CPMAR$

(1) $1 \rightarrow DMAT, CPDMAT;$

(2) $1 \rightarrow IT, CPIT;$

(3) $1 \rightarrow FT, CPFT$

2. SR./DR

ET0: $R_i \text{ OP } D \rightarrow MDR$

$R_i \rightarrow A/B$ 且 $D \rightarrow B/A, A \text{ OP } B,$
DM, CPM_{DR}, T+1, CPT

ET1: $MDR \rightarrow M$

$EMAR, /W, T+1, CPT$

ET2: $PC \rightarrow MAR$

$PC \rightarrow A, A, DM, CPMAR$

(1) $1 \rightarrow DMAT, CPDMAT;$

(2) $1 \rightarrow IT, CPIT;$

(3) $1 \rightarrow FT, CPFT$

3./SR.DR

ET0: C OP Rj \rightarrow Rj

$C \rightarrow A/B \text{ 且 } Rj \rightarrow B/A, A \text{ OP } B,$
DM,CPRj,T+1,CPT

ET1: PC \rightarrow MAR

PC \rightarrow A, A, DM,CPMAR

(1) 1 \rightarrow DMAT,CPDMAT;

(2) 1 \rightarrow IT,CPIT;

(3) 1 \rightarrow FT,CPFT

4./ SR./DR

ET0: C OP D \rightarrow MDR

$C \rightarrow A/B \text{ 且 } D \rightarrow B/A, A \text{ OP } B,$
DM,CPMDR,T+1,CPT

ET1:MDR \rightarrow M

EMAR,/W,T+1,CPT

ET2: PC \rightarrow MAR

PC \rightarrow A, A, DM,CPMAR

(1) 1 \rightarrow DMAT,CPDMAT;

(2) 1 \rightarrow IT,CPIT;

(3) 1 \rightarrow FT,CPFT

例题：拟定指令“**ADD @（R1）十，X（R3）**”的执行流程及操作时间表的安排；**X（R3）**表示源寻址，**@（R1）十**表示目的寻址。

FT:	M→IR	EMAR, R , SIR	
	PC+1→PC	PC→A , A+1 , DM ,CPPC	
		1→ST,CPST	
ST: STo	PC→MAR	PC→A	
		A	
		DM	CPMAR
		T+1	CPT
ST1:	M→MDR→C	EMAR	
		R	
		SMDR	
		MDR→B	
		B	
		DM	CPC
		T+1	CPT

ST2 $PC+1 \rightarrow PC$

$PC \rightarrow A$

$A+1$

DM

$T+1$

CPPC

CPT

ST3 $R3+C \rightarrow MAR$

$R3 \rightarrow A/B, C \rightarrow B/A$

$A+B$

DM

$T+1$

CPMAR

CPT

ST4 $M \rightarrow MDR \rightarrow C$

EMBR

R

SMDR

$MDR \rightarrow B$

B

DM

$1 \rightarrow DT$

CPC

CPDT

DT0 **R1→MAR**

R1→A/B

A/B

DM

T+1

CPMAR

CPT

DT1 **M→MDR→D**

EMAR

R

SMDR

MDR→B

B

DM

T+1

CPD

CPT

DT2 **R1+1→R1**

R1→A/B

A+1

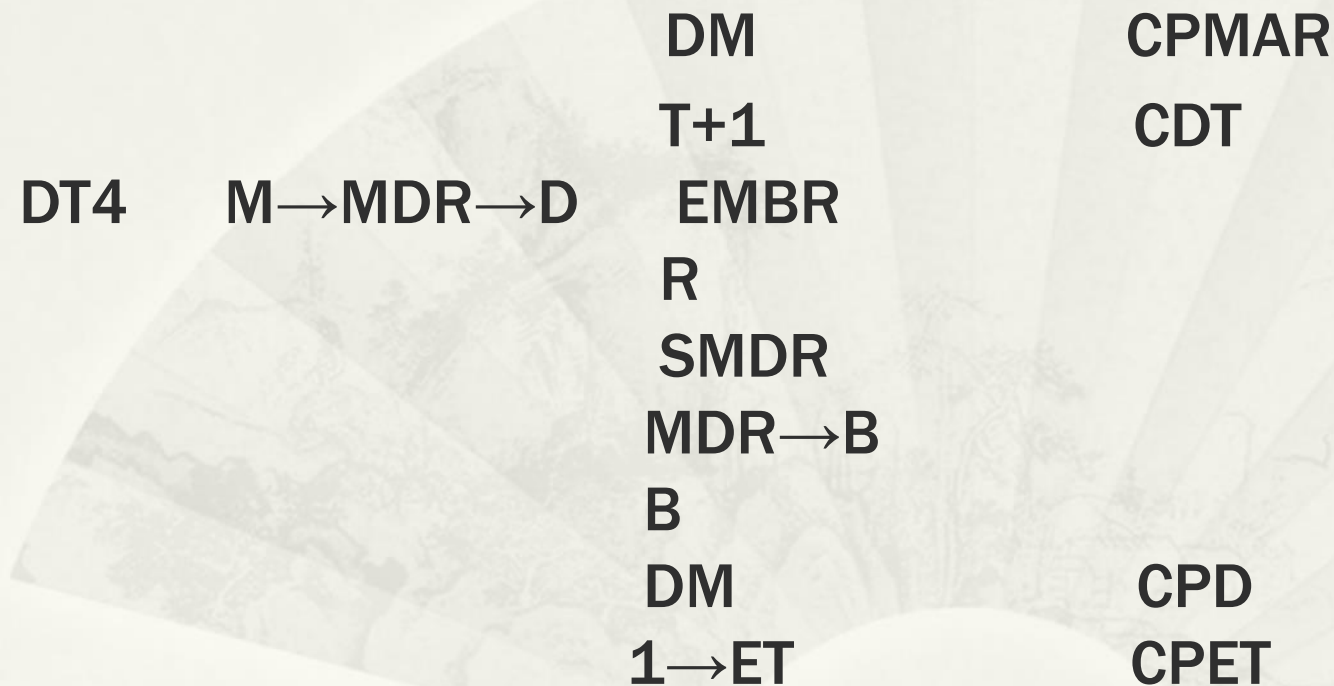
DM

T+1

CPR1

CPT

DT3 **C → MAR** **C → A/B**
 A/B



DT4

M → MDR → D

DM

T+1

EMBR

R

SMDR

MDR → B

B

DM

1 → ET

CPMAR

CDT

CPD

CPET

CPMAR
CDT

CPD
CPET

ET: ET0 $C+D \rightarrow MDR$ $C \rightarrow A/B, D \rightarrow B/A$

ET1 $MDR \rightarrow M$

ET2 $PC \rightarrow MAR$

- (1) $1 \rightarrow FT$
- (2) $1 \rightarrow DMAT$
- (3) $1 \rightarrow IT$

$A+B$

DM

$T+1$
EMAR

W

$T+1$
 $PC \rightarrow A$

A

DM

CPFT
CPDMAT
CPIT

CPMDR
CPT

CPT

CPMAR

组合逻辑控制方式的优缺点

缺点：① 设计不规整。组合逻辑控制方式是用许多门电路产生微命令的，而这些门电路所需的逻辑形态很不规整，因此组合逻辑控制器的核心部分比较繁琐、零乱，设计效率较低，检查调试也比较困难。

② 不易修改或扩展。组合逻辑控制方式的另一缺点是不易修改或扩展指令功能。这是因为设计结果用印制电路板（硬连逻辑）固定下来以后，就很难再修改与扩展。

模型机的微程序控制器

一、基本概念：

1.微命令：构成控制信号序列的最小单位，又称微信号，指那些直接作用于部件或控制门电路的命令。如：打开或关闭某传送通融的电信命令，或对触发器或R进行同步打入，置位、复位等的控制脉冲。

2.微操作：由微命令控制实现的最基本的操作称为微操作，如：开门、关门、选择。

3.微周期：从控制存储器中读取一条微指令并执行相应的一步操作所需的时间，称为一个微周期或微指令周期。通常一个时钟周期为一个微周期。

4.微指令：每个微周期的操作所需的微命令组成一条微指令。从控制存储器的组织角度讲，每个单元存放一条微指令。

5.微程序：一系列微指令的有序集合称为微程序，用来解释执行一条机器指令。

6.对应关系：一条机器指令 \longleftrightarrow 一段微程序 \leftarrow 一系列微指令
 \longleftarrow (若干) 条微指令 \longleftarrow (若干) 条微命令

二、微程序控制的概念：

1、将控制器所需的微命令，以代码（微码）形式编成微指令，存入一个**ROM**构成的控制存储器中。.....将存储逻辑引入**CPU**。

2、将各种机器指令的操作分解为若干微操作序列。.....将程序技术引入**CPU**的构成级。

上面从两个角度阐明了微程序控制的基本概念：微命令的产生方式，微程序与机器指令之间的对应关系。

与机器指令之间的对应关系。

源 目的

例如: **mov (Ro) , (R1)**

FT M→IR **EMAR(一条微命令)**

R (一条微命令)

SIR (一条微命令)

PC+1→PC **PC→A (一条微命令)**

A+1 (一条微命令)

DM(一条微命令) **CPPC (一条微命令)**

1→ST (一条微命令) **CPST (一条微命令)**

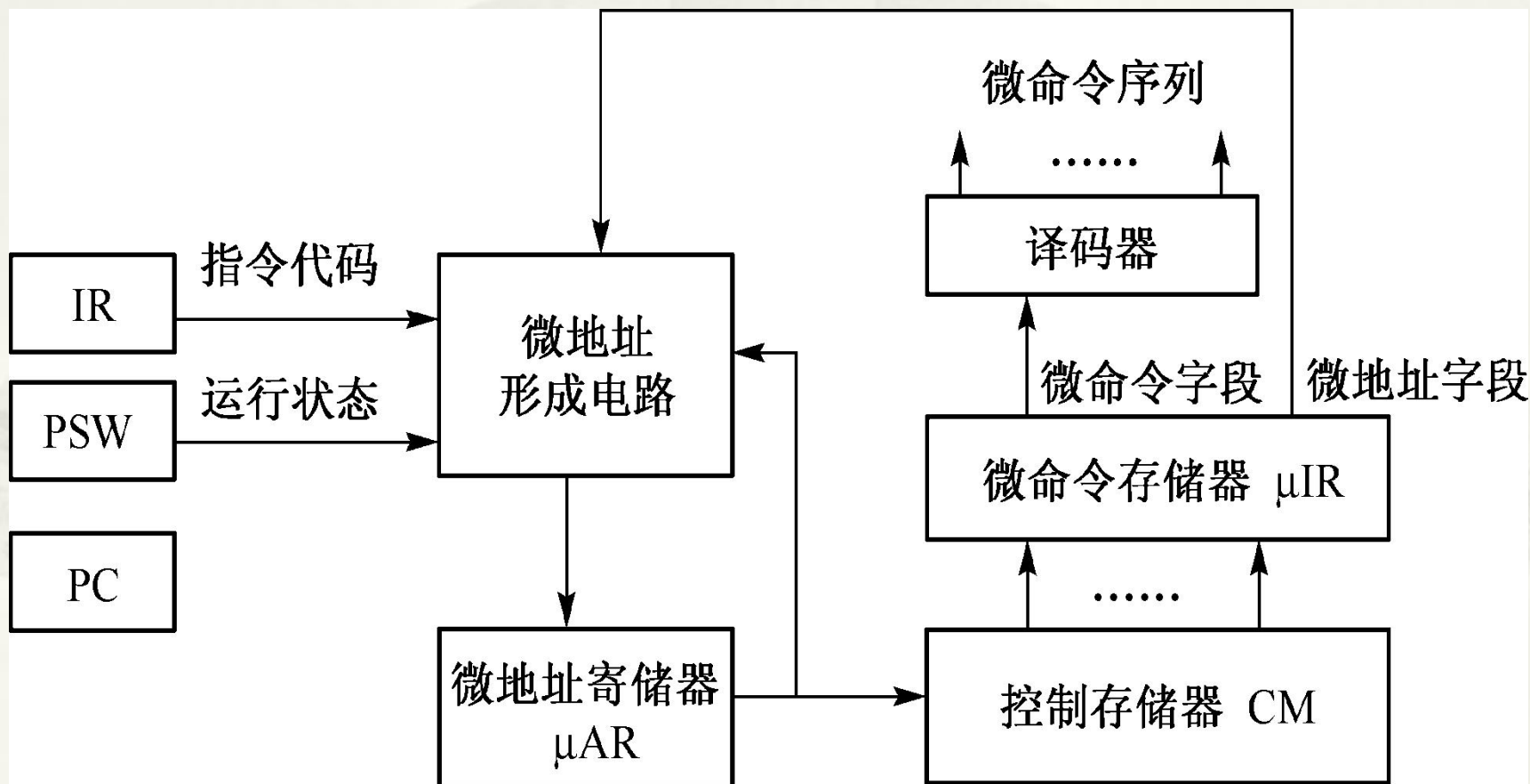
一条微指令

一段微程序



微程序控制器组成及工作原理

微程序控制器的硬件组成及各部分的作用



组成 { **IR、PSW、PC**
时序系统
控制存储器**CM**
微指令寄存器、**UIR**微地址形成电路、微地址寄存器**UAR**、
译码器

1.控制存储器CM: 用来存放微程序，它的每一单元用来存放一条微指令，一段微程序需要几十位。

2.微指令寄存器UIR: 从**CM**读取的微指令，存放于**UIR**中，**UI**分为两部分

{ 微操作控制字段——产生微命令的依据（相当于**I**中的操作码）
{ 顺序控制字段——产生后继微地址指令的依据，用以控制微程序的连续执行

3.微地址形成电路: 根据微程序执行顺序的需要，应有多种后继微指令地址的形成方式，依据以下几种信息的一部分去形成后继微地址：顺序控制字段，现行微指令地址，微程序转移时的微地址等。

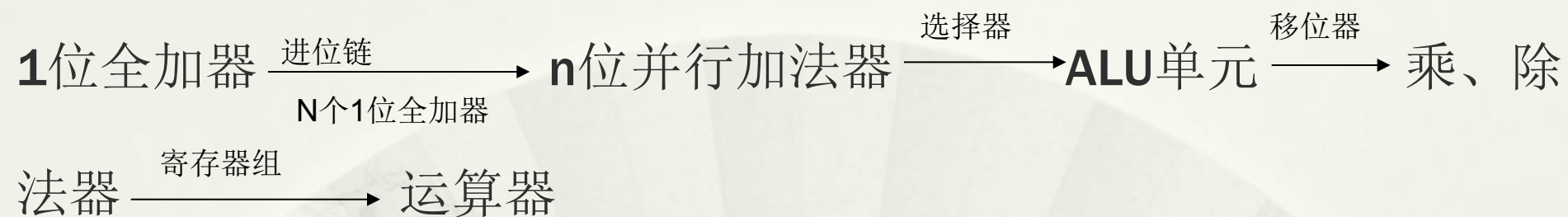
4.微地址寄存器UAR

在从**CM**中读取**UI**时，**MAR**中保存着**CM**的地址（微地址），指向**CM**单元（如同**PC**或堆栈指针）。读出微指令或完成一个微指令周期操作后，微地址形成电路将后继微地址打入**UAR**中，为读取下一条微指令做准备。

微程序控制器工作原理

执行指令时，从控制存储器中找到相应的微程序段，逐次取出微指令，送入微指令寄存器，译码后产生所需微命令，控制各步操作的完成。

算术逻辑运算部件



3.2.1 加法单元

若：+1101，+1111两数相加，且分别放入A、B两个寄存器。

$$\begin{array}{r} 001101 \\ +) 001111 \\ \hline 011100 \end{array} \quad (\text{补码、双符号位表示})$$

第3位：输入量： $A_i(1), B_i(1), C_{i-1}(1)$

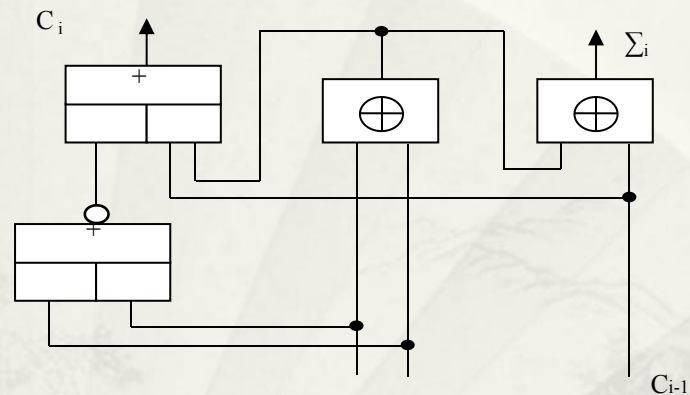
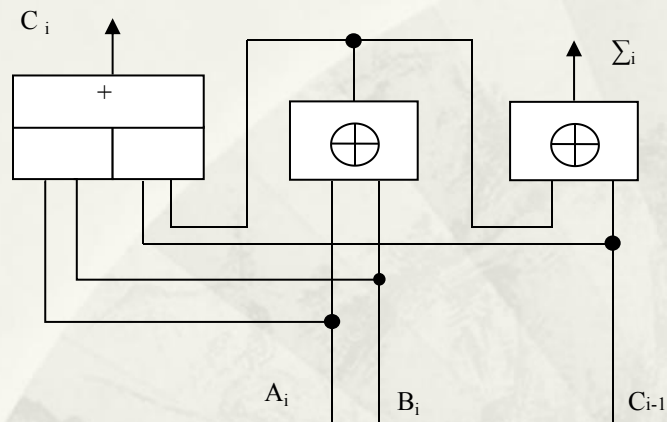
输出量： $\sum_i (1), C_i(1)$

输入、输出量之间的关系式：

$$\Sigma_i = (A_i \oplus B_i) \oplus C_{i-1} \quad (1)$$

$$C_i = A_i B_i + (A_i \oplus B_i) C_{i-1} \quad (2)$$

根据上式，得出一位全加器的逻辑电路图：



根据（1）式得：如果三个输入中**1**的个数为奇数，则本位和为**1**，否则为**0**。
根据（2）式得：当本位的两个输入 **A_i** 、 **B_i** 均为**1**时，不管低位有无进位 **C_{i-1}** 传来，都必然产生进位 **C_i** ；若 **C_{i-1}** 为**1**，只要 **A_i** 、 **B_i** 中有一个为**1**，也必然产生进位。

原码一位乘法

一、定义：取两个操作数的绝对值相乘，每步处理一位乘法，符号位单独处理。

二、运算规则：

1.寄存器分配与初始值：

A,B,C三个寄存器

A存放部分积累加和，其初始值为**0**（双符号位**00**表示）；

B存放被乘数**X**（绝对值），此时，符号位为双符号位**00**（在乘的过程中，**B**中的值一直保持不变）；

C存放**Y**（绝对值），将符号位去掉；**C**寄存器的初始值是乘数**Y**的尾数（有效位数），以后每乘一次，将已处理的低位乘数右移舍去，同时将**A**寄存器的末位移入**C**寄存器的高位。

2.符号位：

A,B均设置双符号位

3.基本操作：

在原码一位乘中，每步只处理一位乘数，即位于**C**寄存器末位的乘数，也称之为判断位**C_n**；

若 $C_n=1$ ，则部分积为 B ，执行 $A+B$ 操作，然后将累加和右移一位，用“ \rightarrow ”表示。（ C_n 位去掉）

若 $C_n=0$ ，则部分积为 0 ，执行 $A+0$ 操作，然后右移。或直接让 A 右移一位。（ C_n 位去掉）

右移时， A 的末位移入 C 的高位， A 的第二符号位移入尾数最高位，第一符号位移入第二符号位，而第一符号位本身则补 0 。

4.操作步骤:

N 此累加与 n 次移位（最后一次累加后要移位）

5.加符号位

补码一位乘法

一、定义：操作数与结果均以补码表示，连同符号位一起，按相应算法运算。

运算方法及关系式：比较法

$$[XY]_{\text{补}} = [A_n]_{\text{补}} + (Y_{n+1} - Y_n)[X]_{\text{补}}$$

二、运算规则：

1. 寄存器分配、初始值及符号位

A,B,C三个寄存器

A存放部分积累加和，其初始值为**0**（双符号位**00**表示）；

B存放被乘数 $X_{\text{补}}$ ，（双符号位**00**、或**11**表示）；

C存放乘数 $Y_{\text{补}}$ ，单符号位（符号位参与运算），**Y**的末位添**0**，称为附加位 Y_{n+1}

2.基本操作

用**C**寄存器最末两位（含增加的**C_{n+1}**）作判断位，即（**Y_n**，**Y_{n+1}**）为判断位，

若**Y_n Y_{n+1}** 为**00**，或**11**，执行**A+0**，右移，实际上可直接让**A**右移一位；

若**Y_n Y_{n+1}** 为**01**，执行**A+X_补**，右移；

若**Y_n Y_{n+1}** 为**10**，执行**A+【-X_补】**，右移。

在右移时，**A**寄存器中的第二符号位值移入尾数的最高数位（有效位的最高位），第一符号位值移入第二符号位，第一符号位本身不变，而**A**寄存器末位移入**C**寄存器。

3.操作步数：为有效位位数的**n+1**

原码不恢复余数除法

一、定义：取两个操作数的绝对值相除，符号位单独处理。

二、运算关系式：

根据余数 r_i 符号判断是否够减： r_i 为正表示够减，上商 $Q_i = 1$ ；

r_i 为负表示不够减，上商 $Q_i = 0$ ；

通式：

若第 i 步够减， $Q_i = 1$ ，则第 $i + 1$ 步应做 $2r_i - Y$ ；

若第 i 步不够减， $Q_i = 0$ ，则第 $i + 1$ 步应做 $2r_i + Y$ 。

三、运算规则

1. 寄存器分配与符号位

A, B, C三个寄存器

A初始值存放被除数（绝对值），以后存放各次余数，**A**取双符号位，从第一符号位判断是否够减，从而决定商值；

B寄存器存放除数的绝对值，取双符号位；

C存放商，取单符号位；商由末位置入，在每次置入新商时，原商同时左移一位。

2.基本操作与上商:

a.第一步操作必为 $2r_0 - Y$

b.以后各部根据如下条件进行:

ri 为正表示够减, 即 $Q_i = 1$, 则第 $i+1$ 步应为 $2r_i - Y$,

ri 为负表示不够减, 即 $Q_i = 0$, 则第 $i+1$ 步应为 $2r_i + Y$,

c.最后一步: 若第 n 步(最后一步)余数为负, 则需增加一步恢复余数, 这增加的一步不移位, 操作为 $r_n + Y$

3.操作步数: 要求得 n 位商(不含符号位), 则需做 n 步(次)“左移—加减”循环。

4.符号: 同号相除为正, 异号反之。

补码不恢复余数除法

一、定义：指被除数、除数，所求得的商，余数等都用补码表示，连同符号位一起运算。

二、运算规则：

1. 寄存器分配与符号位

A, B, C三个寄存器

A初始值存放被除数（补码表示），以后存放各次余数，A取双符号位，；

B寄存器存放除数（补码表示），取双符号位；

C存放商，取单符号位，初始值为0。

2. 假商符

在第一步操作之前，先根据 r_0 （即X），Y符号比较确定假商符（与真商符相反），即： r_0 ，Y同号为1； r_0 ，Y异号为0

3.基本操作

1) 第一步操作，假商符为**1** (r_0 , Y 同号)，做 **$2r_0 - Y_{\text{补}}$** 操作；

假商符为**0** (r_0 , Y 异号)，做 **$2r_0 + Y_{\text{补}}$** 操作；

2) 其余操作根据如下规则进行：

a. 若 $(X_i)_{\text{补}}$, $Y_{\text{补}}$ 同号: $(r_i)_{\text{补}}$ $Y_{\text{补}}$ 同号（够减），上商**1**，
下一步， **$2(r_i)_{\text{补}} - Y_{\text{补}}$**

$(r_i)_{\text{补}}$ $Y_{\text{补}}$ 异号（不够减），上商**0**，下一步， **$2(r_i)_{\text{补}} + Y_{\text{补}}$**

b. 若 $(X_i)_{\text{补}}$, $Y_{\text{补}}$ 同号: $(r_i)_{\text{补}}$ $Y_{\text{补}}$ 异号（不够减），上
商**1**，下一步， **$2(r_i)_{\text{补}} - Y_{\text{补}}$**

$(r_i)_{\text{补}}$ $Y_{\text{补}}$ 异号（够减），上商**0**，下一步， **$2(r_i)_{\text{补}} + Y_{\text{补}}$**

c. 最后一步要对假商校正。

某计算机字长**16**位，**CPU**内部包含如下部件：通用寄存器**R0**、**R1**、**R2**、**R3**，累加器**AC**，算术逻辑单元**ALU**及其数据暂存器**A**和**B**，程序计数器**PC**，指令寄存器**IR**，存储器地址寄存器**MAR**，存储器读数据缓冲器**MER**，存储器写数据缓冲器**MDR**。**ALU**支持加（**A+B**）、减（**A-B**）、与（**A∧B**）、或（**A∨B**）4种算术逻辑运算，分别由**Add**、**Sub**、**And**、**Or** 4个控制信号控制。所有寄存器、数据总线及内总线均为**16**位。下图图是该**CPU**内部数据通路图。

加法运算指令 **ADD R1, 1000H(R2)**。其中源操作数**1000H(R2)**是基址寻址，目的操作数**R1**是寄存器直接寻址，指令编码长度**32**位，指令编码格式如下：

Opcode(8)	Ms(2)	Rs(2)	Mt(2)	Rt(2)
Offset(16)				

Opcode: 操作码

Offset: 位移量

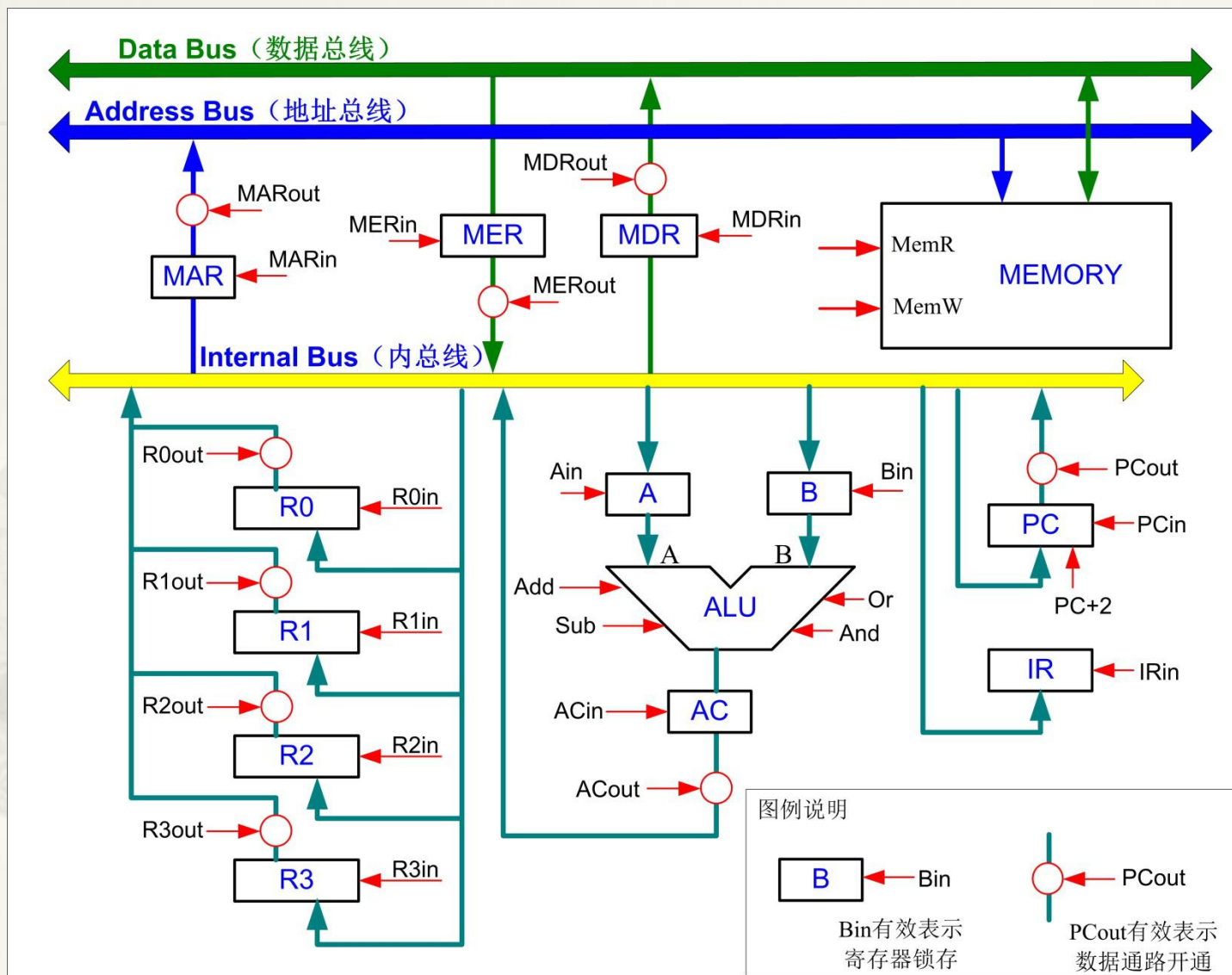
Ms: 源操作数寻址方式

Rs: 源寄存器

Mt: 目的操作数寻址方式

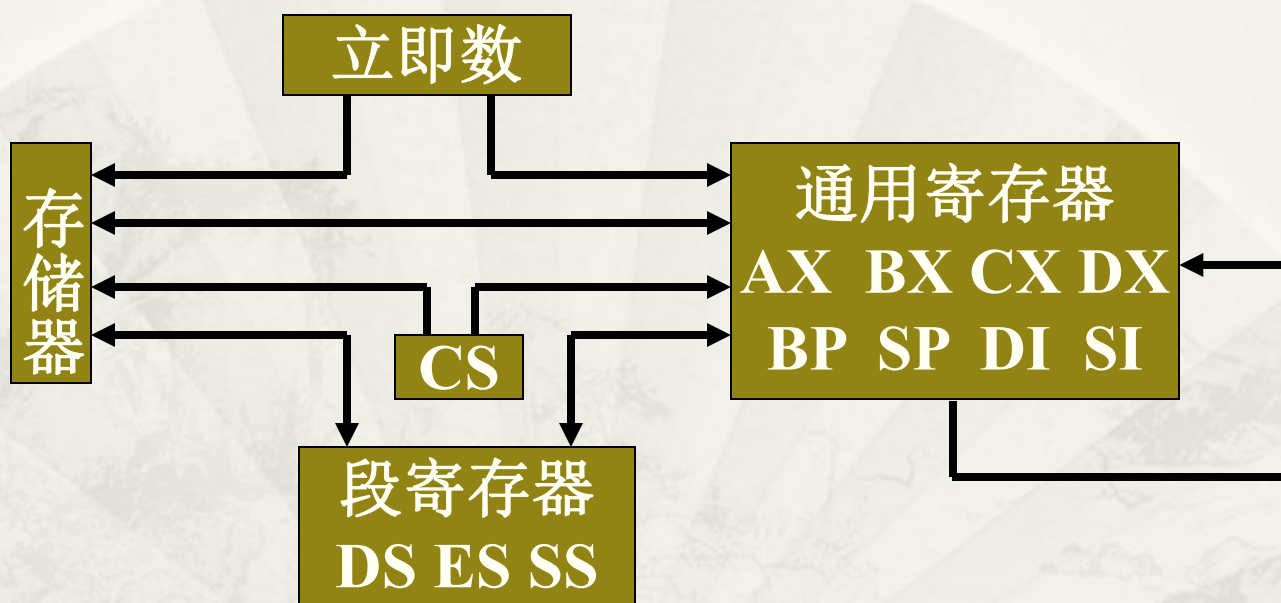
Rt: 目的寄存器

请根据数据通路分析该指令执行过程，把指令执行过程中各时钟周期的微操作及应处于有效状态的控制信号填入下表（参照表中已给出的取指令周期的表示方法）。



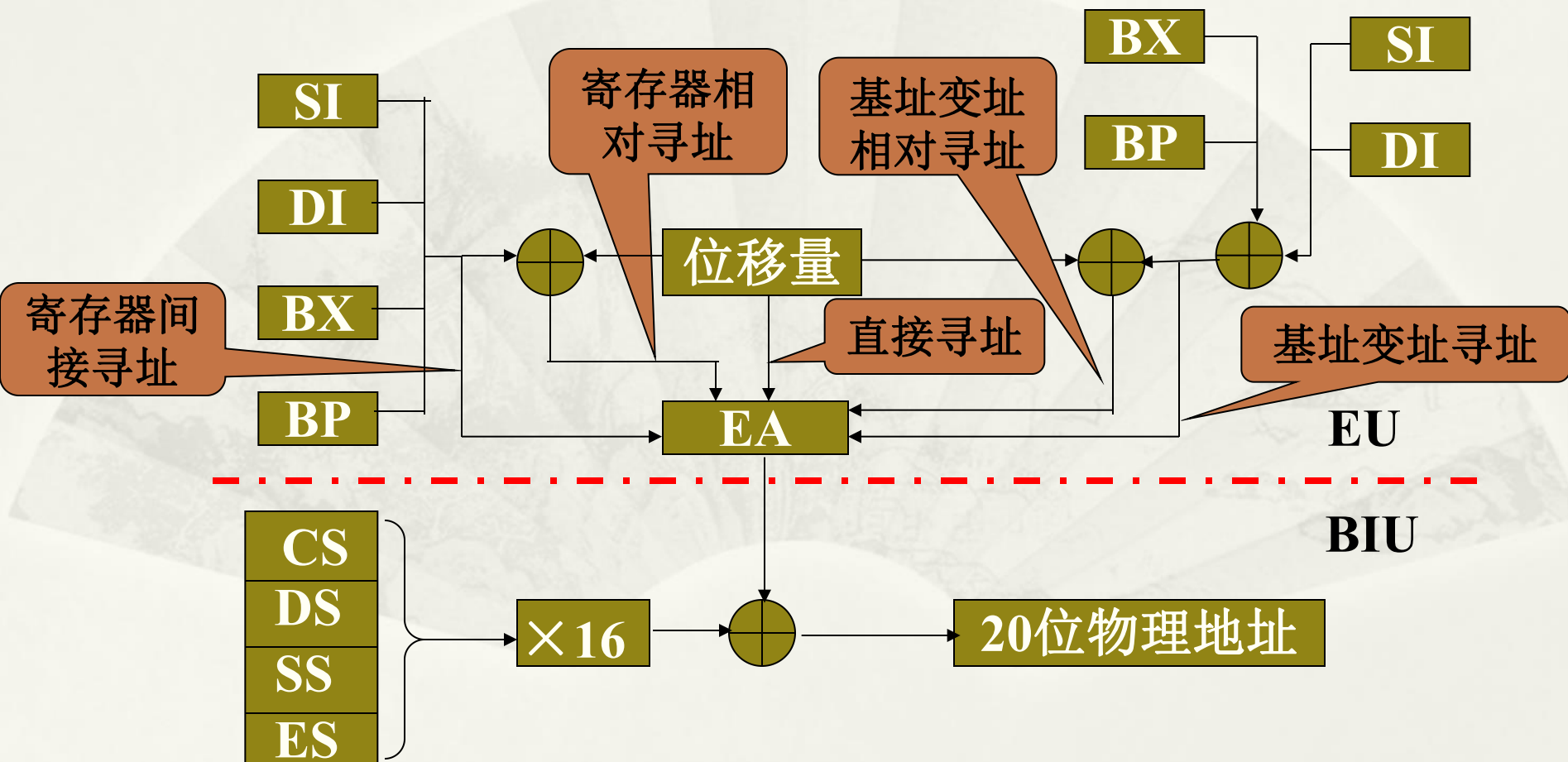
	时钟周期	微操作（功能）	控制信号
取指令	T1	指令地址送MAR, ($MAR \leftarrow PC$)	PCout, MARin
	T2	读指令送MER, ($MER \leftarrow M(MAR)$)	MARout, MemR, MERin
	T3	PC调整, 指令送IR, 译码 ($PC \leftarrow PC + 2, IR \leftarrow MER$)	PC+2, MERout, IRin
取位移量	T4	$MAR \leftarrow PC$	PCout, MARin
	T5	$MER \leftarrow M(MAR)$ $PC \leftarrow PC + 2, B \leftarrow R2$	MARout, MemR, MERin PC+2, R2out, Bin
计算有效地址	T6	$A \leftarrow MER, AC \leftarrow A+B$	MERout, Ain, Add, ACin
读取源操作数	T7	$MAR \leftarrow AC$	Acout, MARin
	T8	$MER \leftarrow M(MAR), A \leftarrow R1$	MARout, MemR, MERin R1out, Ain
执行指令	T9	$B \leftarrow MER, AC \leftarrow A+B$	MERout, Bin, Add, ACin
	T10	$R1 \leftarrow AC$	Acout, R1in

数据传送指令的传送方向示意图



存储器操作数寻址方式中地址形成小结

8086/8088



第四章 存储子系统

存储器是用来存放大量程序与数据的计算机部件。

存储器的层次结构

典型的三级存储体系结构，分为“高速缓冲存储器—主存—外存”三个层次。

1. 主存储器：主存储器是能被CPU直接编程访问的存储器，它存放当前CPU需要执行的程序与需要处理的数据。

为满足CPU编程直接访问的需要，对主存储器的基本要求有三条：

- (1) 随机访问
- (2) 工作速度快
- (3) 具有一定的存储容量

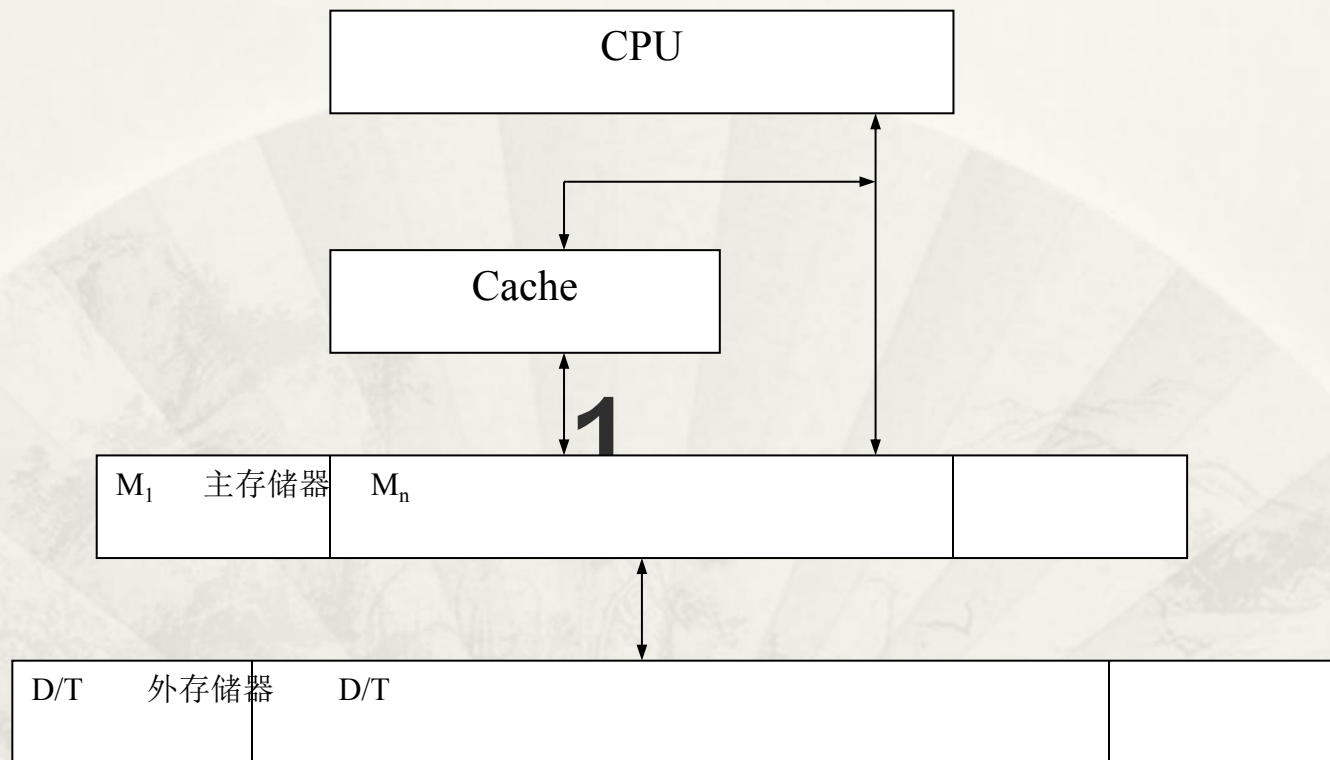
2. 外存储器

外存储器用来存放需要联机保存但暂不使用的程序与数据。

程序与数据只有进入主存才能真正运行，而外存储器是作为后援的。

3. 高速缓存

高速缓存中存放的是最近要使用的程序与数据，作为主存中当前活跃信息的副本。



分层存储体系结构示意图

静态MOS存储单元与芯片（SRAM）

一、定义：若**T1**通导而**T2**截止，存入信息为**0**。若**T1**截止而**T2**通导，存入信息为**1**。

二、NMOS六管静态存储单元电路

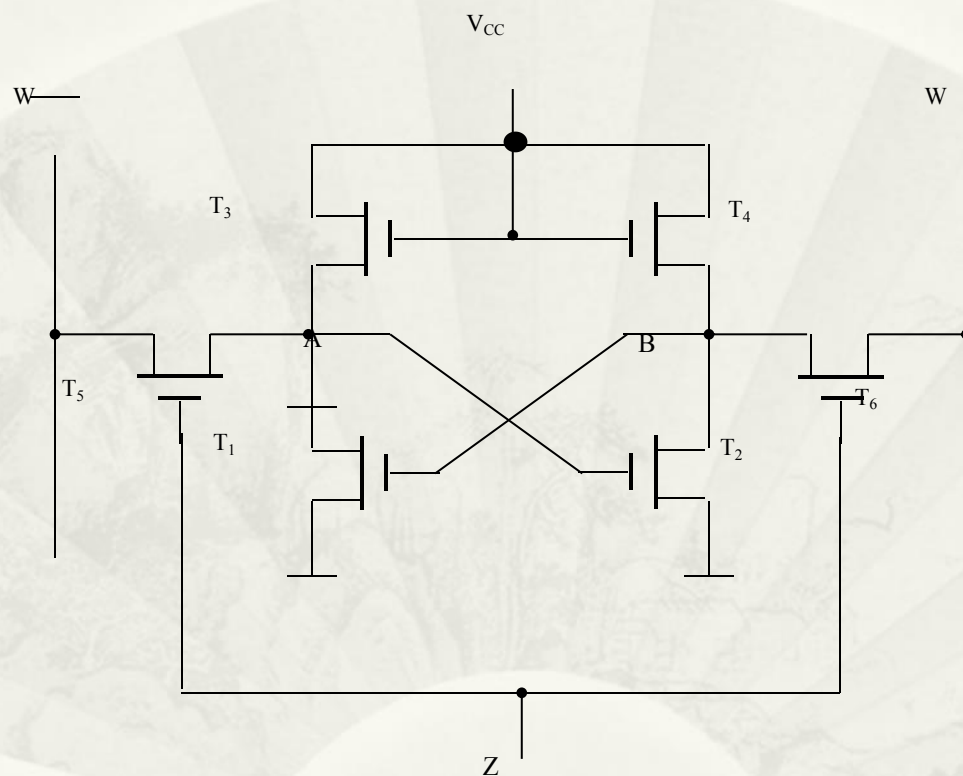


图4-7 NMOS六管静态存储单元

三、读/写、保持状态：

1. 写入：

首先字线Z加高电平时， T_5 、 T_6 导通。

(1) 写” 0”：

a. $/W$ 加低电平，若 $/W$ 电压 $>V_A$ ，则 $/W \rightarrow T_5 \rightarrow A \rightarrow T_1 \rightarrow$ 对地放电，使 A_1 变为低电平 $\rightarrow T_2$ 截止；

b. 同时， W 加高电平，通过 $W \rightarrow T_6 \rightarrow B \rightarrow B_1$ 加高电平，使 T_1 导通。

(2) 写” 1”：

a. $/W$ 加高电平，通过 $/W \rightarrow T_5 \rightarrow A \rightarrow A_1$ 加高电平，使 T_2 导通；

b. 同时， W 加低电平，若 W 电压 $>V_B$ ，则 $W \rightarrow T_6 \rightarrow B \rightarrow T_2 \rightarrow$ 对地放电，使 B_1 变为低电平 $\rightarrow T_1$ 截止；

2. 读出:

先对位线W、 \overline{W} 充电至高电平，该电平是浮动的，可随充放电而变；然后对字线Z加高电平，使 T_5 、 T_6 导通。

(1) 读0: (即 T_1 导通)

位线 \overline{W} 上的电压 $> V_A$ 的电压,
 $\overline{W} \rightarrow T_5 \rightarrow A \rightarrow T_1 \rightarrow$ 地, 形成放电回路, 即有电流经
 $\overline{W} \rightarrow T_1$, 经放大为“0”信号, 表明原存储信息为0.
此时 T_2 截止, W上无电流。

(2) 读1: (A_1 为高电平)

位线W上的电压 $> V_B$ 的电压
 $W \rightarrow T_6 \rightarrow B \rightarrow T_2 \rightarrow$ 地, 形成放电回路, 即有电流经
 $W \rightarrow T_2$, 经放大为“1”信号, 表明原存储信息为1.
此时 T_1 截止, \overline{W} 上无电流。

3. 保持:

字线Z加低电平，门管 T_5 与 T_6 断开，位线与双稳态电路隔离，双稳态电路依靠自身的交叉反馈保持原有状态不变。

总之， $/W$ 上有电流为0， W 上有电流为1。上述读出过程并不改变双稳态电路原有状态，属于非破坏性读出。

动态MOS存储单元与芯片

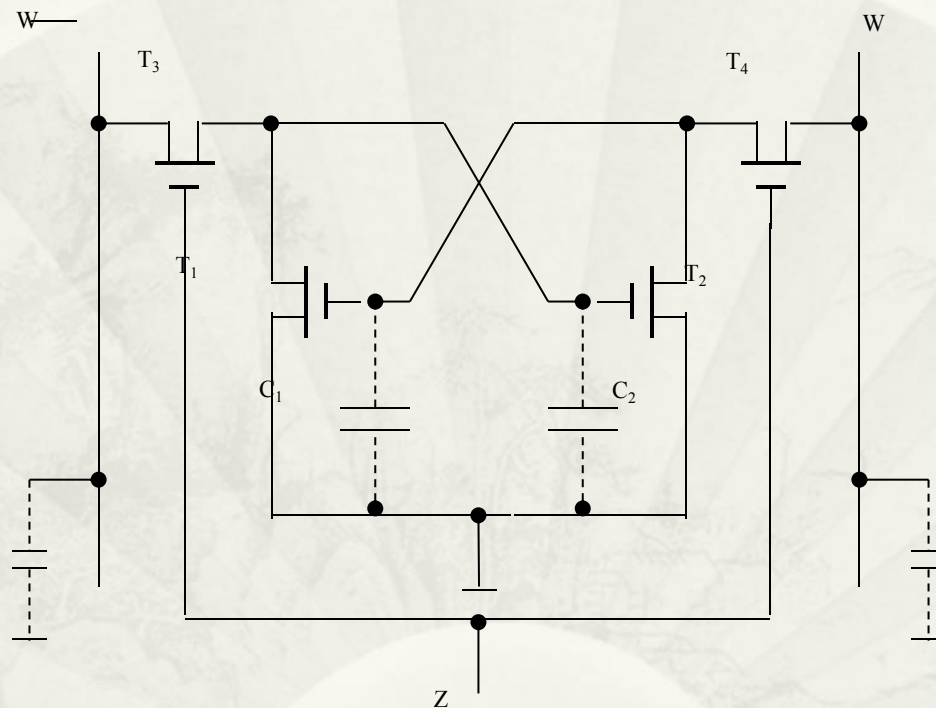
一、动态MOS存储器的基本存储原理：是将存储信息以电荷形式存于电容之中，这种电容可以是MOS管栅极电容，或者是专用的MOS电容，通常定义电容充电至高电平时为1；放电至低电平时为0。

二、概念：

1. 刷新：在MOS管断开之后，电容总存在泄漏通路，难以使泄漏电阻达到无穷大。时间过长，电容上的电荷会通过泄漏电阻放电，使所存储的信息丢失。为此，经过一定时间后就需要对存储内容重写一遍，也就是对存1的电容重新充电，称为刷新。

2. 重写(再生)：对于单管动态MOS存储单元而言，读操作后C上的电荷将发生变化，属于破坏性读出，需要读后对存1的电容补充电荷，称为重写(再生)。这一过程由芯片内的外围电路自动实现。

三、动态MOS四管存储单元电路：（非双稳态电路）



动态MOS四管存储单元

四、读/写、保持状态：

1. 写入：

首先字线Z加高电平时， T_3 、 T_4 导通。

(1) 写0：

a. \overline{W} 加低电平， $C_2 \rightarrow A_1 \rightarrow A \rightarrow T_1 \rightarrow$ 对地放电，（使变为低电平 $\rightarrow T_2$ 截止；）

$C_2 \rightarrow A_1 \rightarrow A \rightarrow T_3 \rightarrow \overline{W}$ 放电（瞬间）

b. W加高电平，通过 $W \rightarrow T_4 \rightarrow B \rightarrow B_1 \rightarrow C_1$ 充电至高电平，使 T_1 导通。

(2) 写1：

a. \overline{W} 加高电平， $\overline{W} \rightarrow T_3 \rightarrow A \rightarrow A_1 \rightarrow C_2$ 充电至高电平，使 T_2 导通

b. W加低电平， $C_1 \rightarrow B_1 \rightarrow B \rightarrow T_2 \rightarrow$ 对地放电，（使 B_1 变为低电平 $\rightarrow T_1$ 截止；）

$C_1 \rightarrow B_1 \rightarrow B \rightarrow T_4 \rightarrow$ 对W放电（瞬间）

2. 读出:

先对位线 W 、 \overline{W} 充电至高电平，该电平是浮动的；
然后对字线 Z 加高电平，使 T_3 、 T_4 导通。

(1) 读0:

C_1 上有电荷为高电平， T_1 导通， $\overline{W} \rightarrow T_3 \rightarrow A \rightarrow T_1 \rightarrow$ 对地放电，即 \overline{W} 上有电流通过，放大后作为0信号读出；

同时， $W \rightarrow T_4 \rightarrow B \rightarrow B_1 \rightarrow C_1$ 充电至高电平，补充泄漏掉的电荷 ➡ 四管单元为非破坏性读出，且读出过程为刷新过程。

(2) 读1:

C_2 上有电荷为高电平， T_2 导通， $W \rightarrow T_4 \rightarrow B \rightarrow T_2 \rightarrow$ 对地放电，即 W 上有电流通过，放大后作为1信号读出；

同时， $\overline{W} \rightarrow T_3 \rightarrow A \rightarrow A_1 \rightarrow C_2$ 充电至高电平，补充泄漏掉的电荷 ➡ 四管单元为非破坏性读出，且读出过程为刷新过程。

3. 保持:

字线Z加低电平时, T_3 、 T_4 断开, 基本上无放电回路, 仅存在泄漏电流, 信息可暂存数毫秒。

注: 读出过程就是刷新过程。

4.4 磁表面存储原理

磁表面存储器有：磁卡、磁鼓、磁带、磁盘等。

读/写原理：1.写入：电转磁

2.读出：磁转电

4.4.1 磁记录编码方式

记录方式对提高记录密度至关重要。

归零制 (RZ)

不归零制 (NRZ)

不归零-1制 (NRZ1)

调相制 (PM)

调频制 (FM)

改进型调频制 (M2F)

群码制 (GCR)

传统的磁记录方式

一、不归零-1制 (NRZ1)

1.写/读规律：

(1) 写入

写入规律可概括为：电流见1则翻。

a.写0时，写入电流维持原方向不变(-I或+I)。

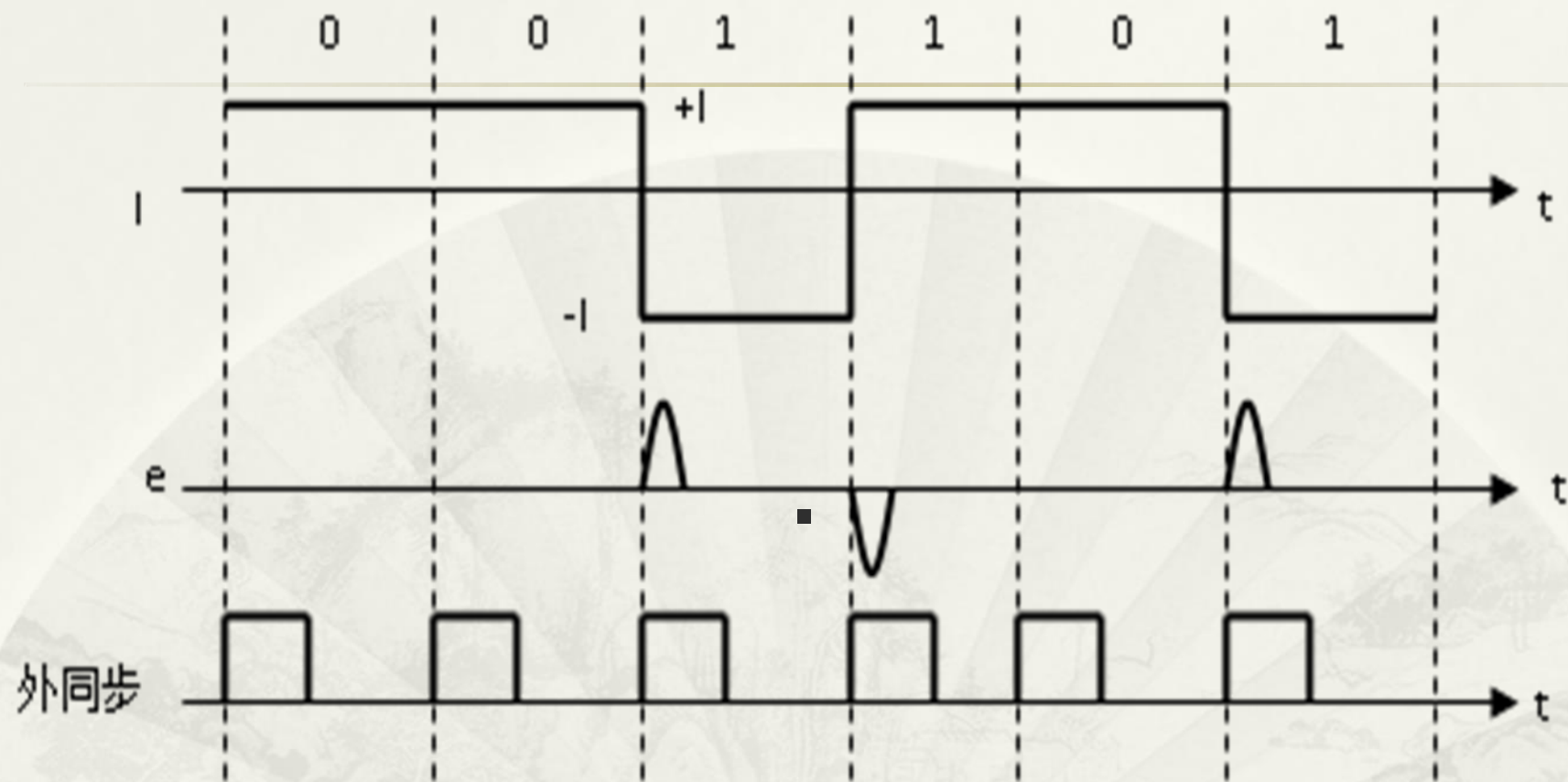
b.写1时，写入电流方向翻转(由-I→+I，或由+I→-I)。

(2)读出：

a.读0，无读出信号

b.读1，有读出信号

2.举例：001101



不归零-1制写入电流波形

3.特点：

1)NRZ1制产生的转变区数较少，可以提高记录密度，但读一连串0时，由于没有转变区存在而没有读出信号，如何识别这是几个0呢？这就需要外加同步信号来辨识各位单元，称为外同步方式。换句话说，NRZ1制没有自同步能力。

2)改进方式有两种：a.每位写入一个同步信号；b.各位采取奇校验。

适用范围：早期低俗磁带机。

二、调相制 (PM)

1. 写/读规律：

(1) 写入

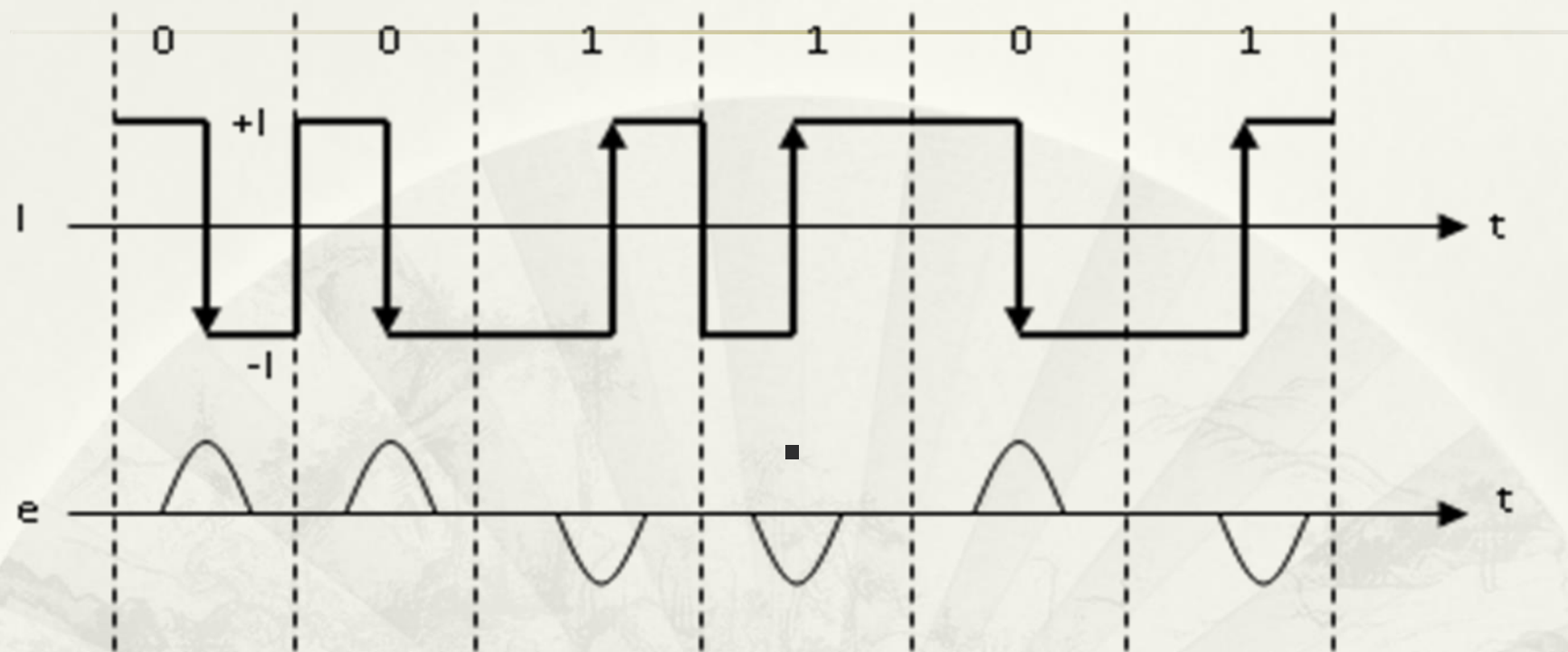
a. 写0: 在位单元中间位置让写入电流负跳变，由 $+I \rightarrow -I$ ；

b. 写1: 在位单元中间位置让写入电流正跳变，由 $-I \rightarrow +I$ 。

(2) 读出

读出时，位单元中间的转变区将产生读出信号。它既是数据信号。也是同步信号，所以调相制具有自同步能力。

2. 举例：001101



调相制波形

特点：可靠性较高，有自同步能力，密度较高；适用常规磁带机。

三、调频制 (FM)

1. 写/读规律：

(1) 写入：每个位单元起始处, 写入电流都改变一次方向, 留下一个转变区，作为本位的同步信号；

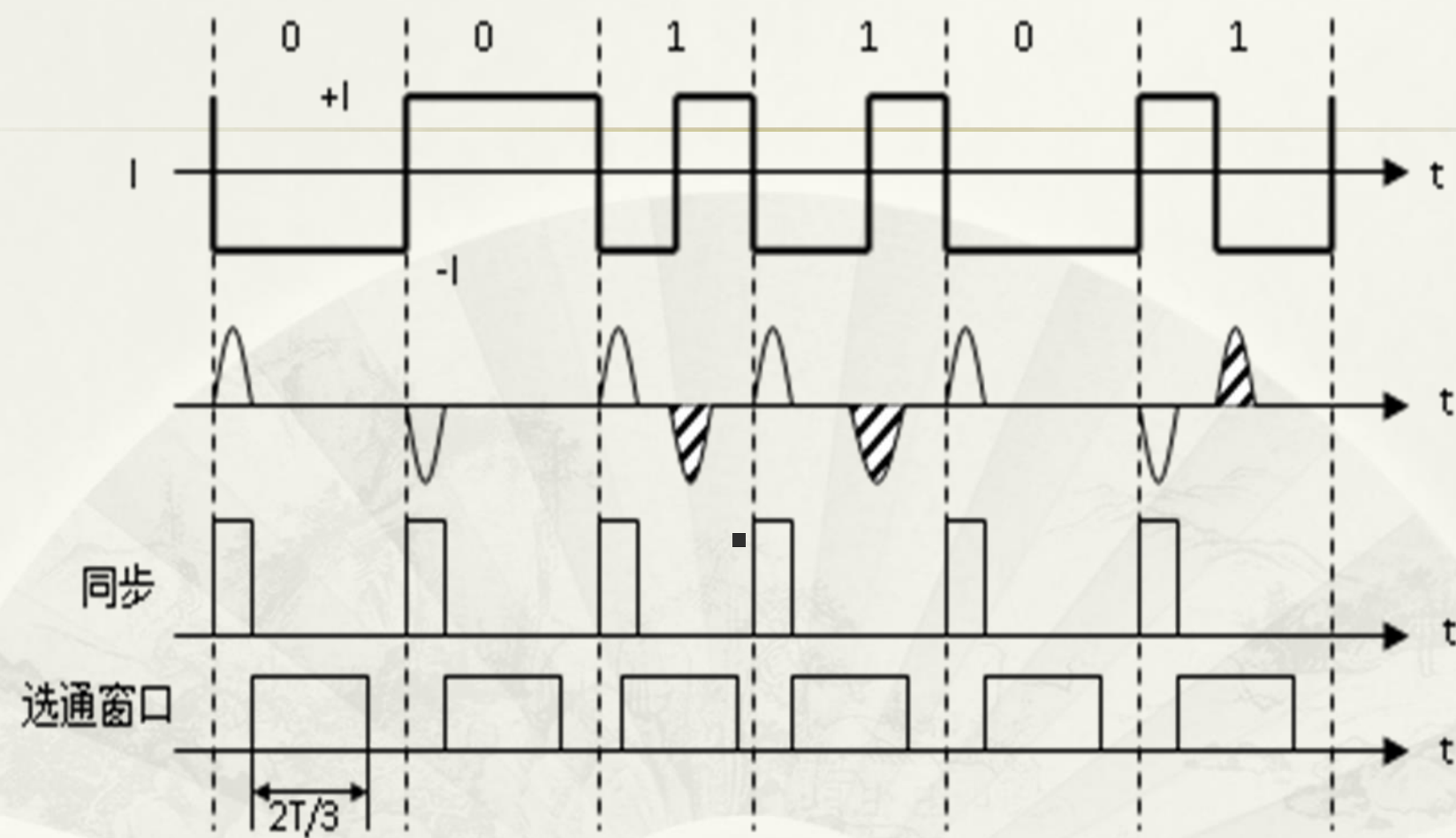
a. 写0：位单元中间不变（整个位单元只变一次）；

b. 写1：位单元中间改变一次电流方向（整个位单元变两次）。

(2) 读出：

每个转变区都将产生一个感应电势，所以读出信号序列中包含了同步信号和数据信号。

2. 举例：001101



调频制波形

3.特点：密度较高，有自同步能力，适用于早期磁盘机。

四、改进型调频制（M2F）

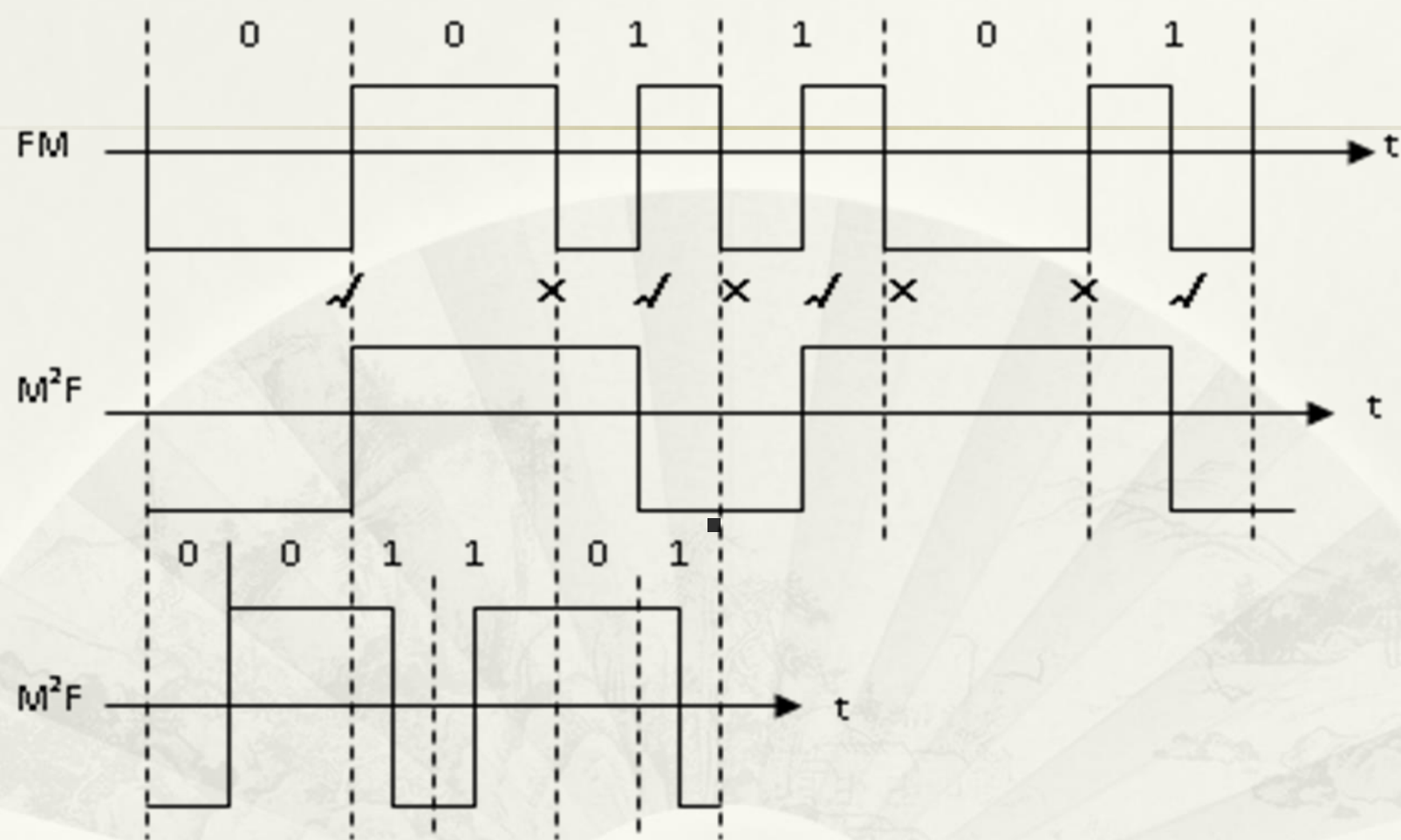
1. 写/读规律：

(1)写1时，在位单元中间改变写入电流方向；

(2) 写入两个以上0时，在它们的交界处改变写入电流方向；

(3) 在0、1或1、0交界处写入电流方向不改变。

2. 举例：001101



改进型调频制波形

3.特点：密度高，有自同步能力。

五、群码制（GCR），对NRZ1的改进

4位数据码 $\xrightarrow{\text{转换}}$ 5位记录码

4.4.3 磁表面存储器的校验方法

分为 { 海明校验（早期）
循环校验

设待编的有效信息k位，分成r组,每组增设一个校验位,共需增设r位校验位，组成一个n位的海明校验码。

$$n = k + r \leq 2^r - 1$$

若k = 4，则r ≥ 3，组成7位海明码。

循环校验(CRC)

1.校验规则是：让校验码能为某一约定代码所除尽。如果除得尽，表明代码正确；如果除不尽，余数将指明出错位所在位置。

2.运算规则： n 位校验与生成多项式按位异或。

3.生成多项式的选取：CCITT(国际电报电话咨询委员会)、IEEE(美国电气和电子工程师学会)推荐。

第五章 输入/输出系统

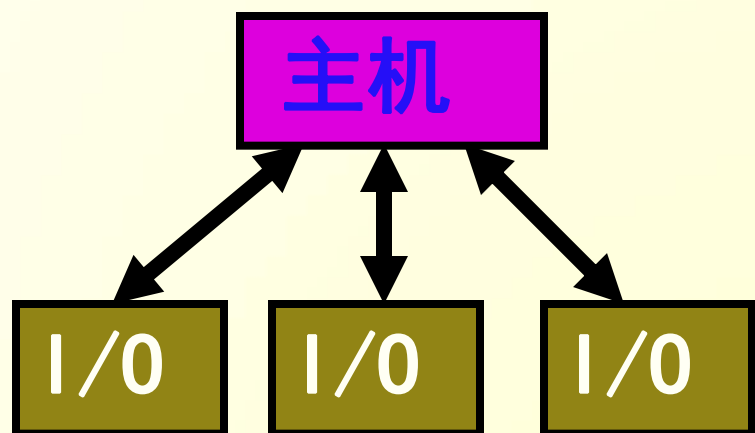
计算机互联进行信息交换的基础 { 系统总线
各种接口—中断、**DMA**接口
信息传输的控制方式
相应的程序软件

5.1 概述

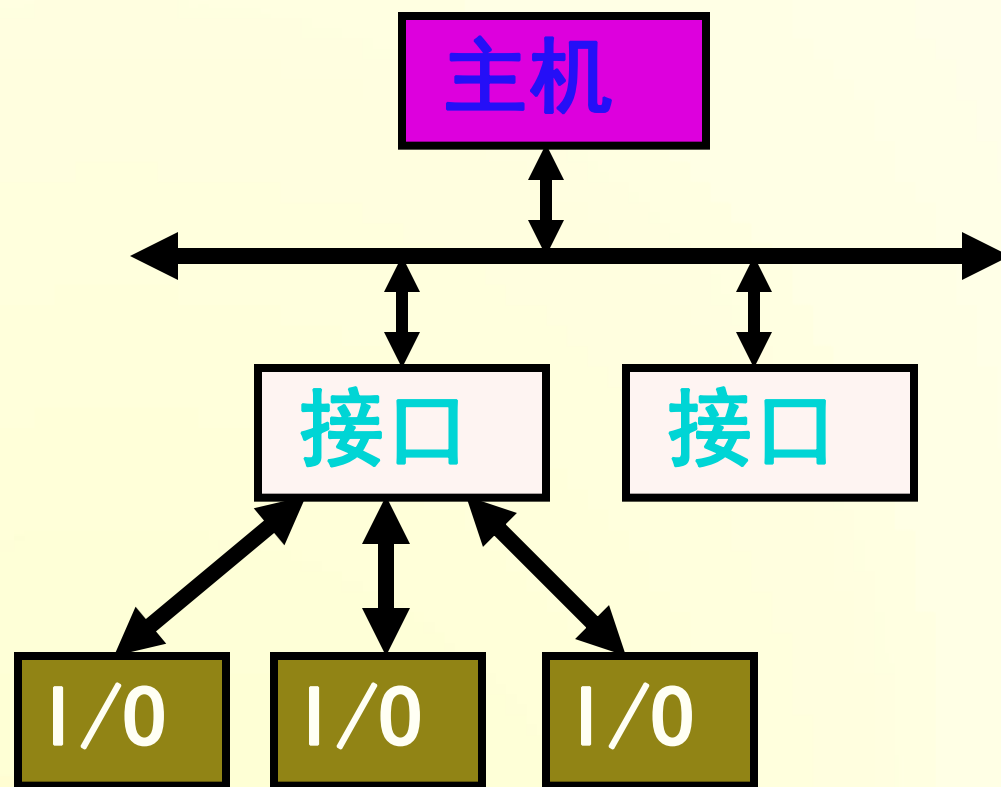
I/O系统 { 硬件 { I/O设备
接口
系统总线
软件 { 用户I/O程序:与控制方式有关: **DMA**, 中断等
设备驱动程序:**OS**中, 屏蔽外设的物理细节
设备控制程序:固化在I/O控制器中的控制程序,
控制外设的**R,W**, 以及总线上的访问控制信号

5.1.1 主机和外设的连接方式

1. 辐射式



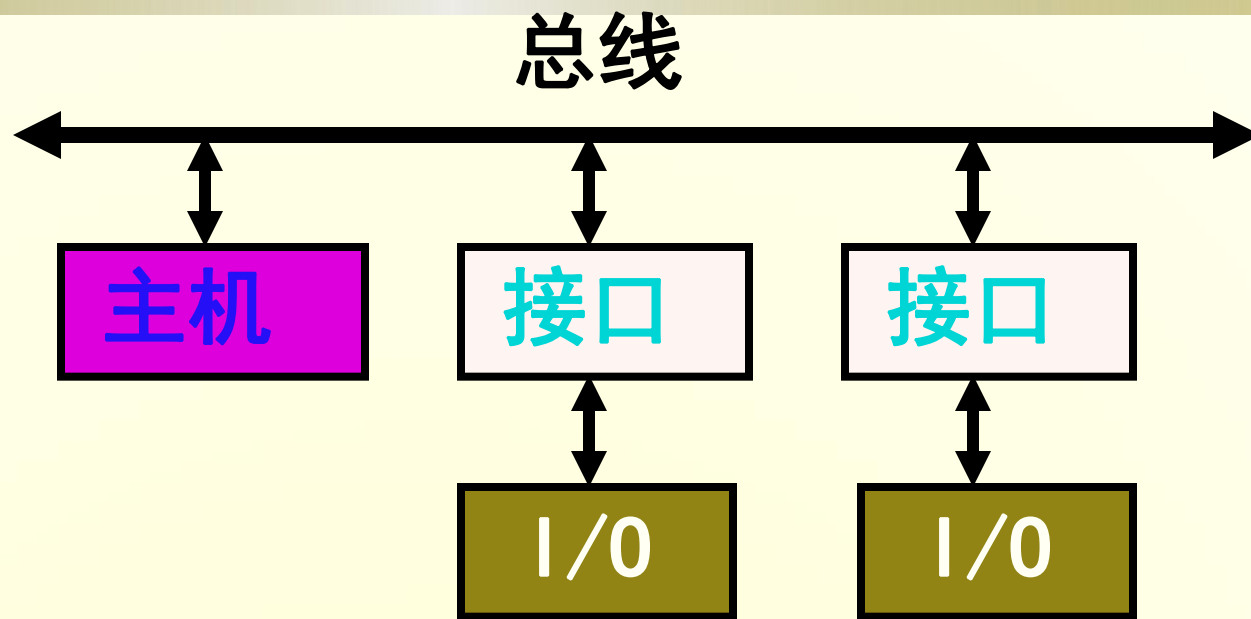
早期：不易扩展



现在：便于扩展

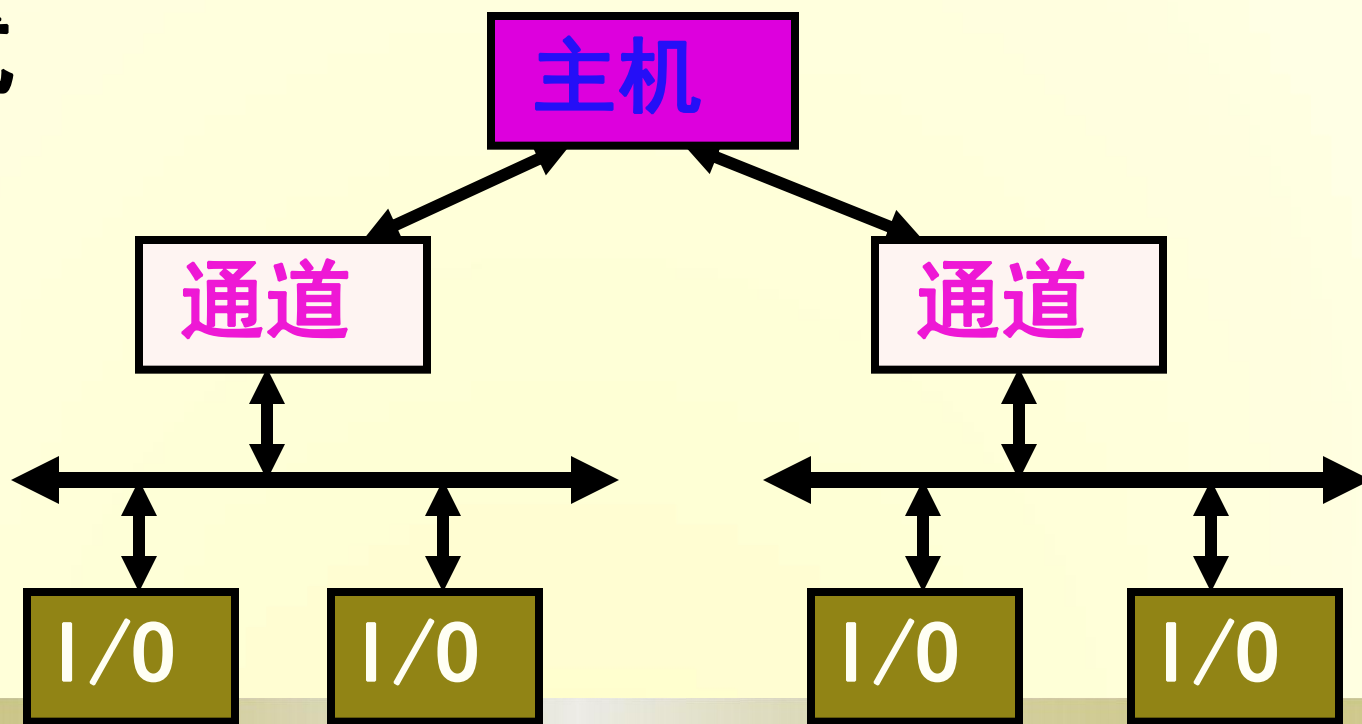
2. 总线式

便于扩展



3. 通道式

并行能力
提高



5.1.2 总线类型与总线标准

一.定义:是一组能为多个部件分时共享的公共信息传送线路, 及相应的控制逻辑。

二.总线类型

1.从系统组成角度分类

分类 { 单组总线
多组总线

2. 按数据传输格式分类

分类

并行总线:用多条数据线同时传送一个字节或一个字的所有代码位。（适用于近距离传送）

串行总线:按位串行传送数据，即按数据代码位流的顺序逐位传送。（适用于远距离传送）

3. 按时序控制方式分类

分类

同步总线:数据传送操作由统一的系统时钟同步定时，其特点是有严格的时钟周期划分。（适用于近距离传送）

异步总线:对总线操作的控制和数据传送，是以应答方式实现的，其特征是没有固定的时钟周期划分。（适用于远距离传送）

总线标准:IEEE制定

5.1.3 接口的功能与分类

一.定义:接口泛指设备部件（硬、软）之间的交接部分

接口类型 {
 硬件接口
 软件接口
 软硬接口

I/O接口:主机（系统总线）与外围设备或其他外部系统之间的接口逻辑。

二.**I/O接口**的基本功能

1.寻址:接口逻辑接收总线送来的寻址信息，经过译码，选择该接口中的某个有关的寄存器。

2.数据传输与缓冲:设置接口的基本目的是为设备之间提供数据传送通路，但各种设备的工作速度不同，特别是**CPU**、主存与外围设备之间，往往速度差异较大。

3. 数据格式变换、电平变换等预处理:接口与系统总线之间, 通常采用并行传送; 接口与外围设备之间, 有可能采取并行传送, 也有可能采取串行传送, 视具体的设备类型而定。因此, 接口就有可能需要担负数据的串并格式转换功能。

设备使用的电源与系统总线使用的电源有可能不同, 因此它们之间的信号电平有可能不同。

4. 控制逻辑: 主机通过总线向接口传送命令信息, 接口予以解释, 并产生相应的操作命令发送给设备。接口形成设备及接口本身的有关状态信息, 也通过总线回送给CPU。

分类 { 中断逻辑
DMA逻辑

三. I/O接口的分类

1. 按数据传输格式分类

分类 { 并行接口: 接口与系统总线之间、接口与外部设备之间, 都以并行方式传送数据信息。
串行接口: 接口与外部设备之间采用串行方式传送数据, 而接口与系统总线之间一般仍采用并行方式传送数据。

2. 按时序控制方式分类

分类

同步接口:与同步总线连接的接口,接口与系统总线间的信息传送由统一的时序信号控制。

串行接口:与异步总线相连的接口,接口与系统总线间的信息传送采用异步应答的控制方式。

3. 按信息传送的控制方式分类

分类

中断接口:如果主机与外围设备之间的信息传送采用程序中断方式控制,则接口需有相应的中断系统所需的逻辑。

DMA接口:如果主机与高速外围设备之间的信息传送采用DMA方式控制,则接口中需有相应的DMA逻辑。

5.2 系统总线

5.2.1 总线信号组成

模型机系统总线共78条按功能可分为四组

电源线与地线（16条）

地址线（16条）

数据线（16条）

控制信号线（30条）

一. 电源线与地线（16条）

- 1. 电源线10条: +5V 2条（主电源线），-5V 2条，+12V 2条，-12V 2条
- 2. 地线4条
- 3. 附加地2条: 将电源线与信号线分开，有利于抑制干扰

二. 地址线（16条）

寻址空间64KB，包括I/O端口地址。

三. 数据线（16条）

四. 控制信号线（30条）

复位信号线 (RESET): 1条

同步定时信号线: 6条

异步应答信号线: 3条

总线控制权信号线: 3条: BREQ, BACK, BUSY

中断请求与批准信号线: 10条, IREQ0—
IREQ7, INTA, INT

优先权判定线: 2条

数据传送控制信号: 5条, MEMR, MEMW,
IOR, IOW, BHEN

7组

5.2.2 总线操作与时序

一.同步控制方式的总线操作

主要特征:以时钟周期为划分时间段的基准。

总线周期为时钟周期的整数位。

二.异步控制方式的总线特征

主要特征:没有统一的时钟周期划分, 而采取应答方式实现总线的传送操作, 所需时间视需要而定。

根据主、从设备的请求信号、回答信号及设备自身定时的关系

异步应答分 { 不互锁
半互锁
全互锁

1. 不互锁: 请求信号引发回答信号, 两个请求信号的结束时设备自身定时决定的。
2. 半互锁: 请求信号引发回答信号, 设备1的请求信号的结束是设备2决定的。
3. 全互锁: 请求信号引发回答信号, 设备1的请求信号的结束是设备2决定的, 设备2的回答结束信号是根据设备1决定的。

5.3 直接程序传送方式及接口

一. 定义:

CPU直接利用I/O指令程序实现I/O传送, 在外设工作期间, CPU不执行与I/O无关的操作。

二. 主机状态:

CPU处于查询—等待—执行状态.

三. 外设状态:

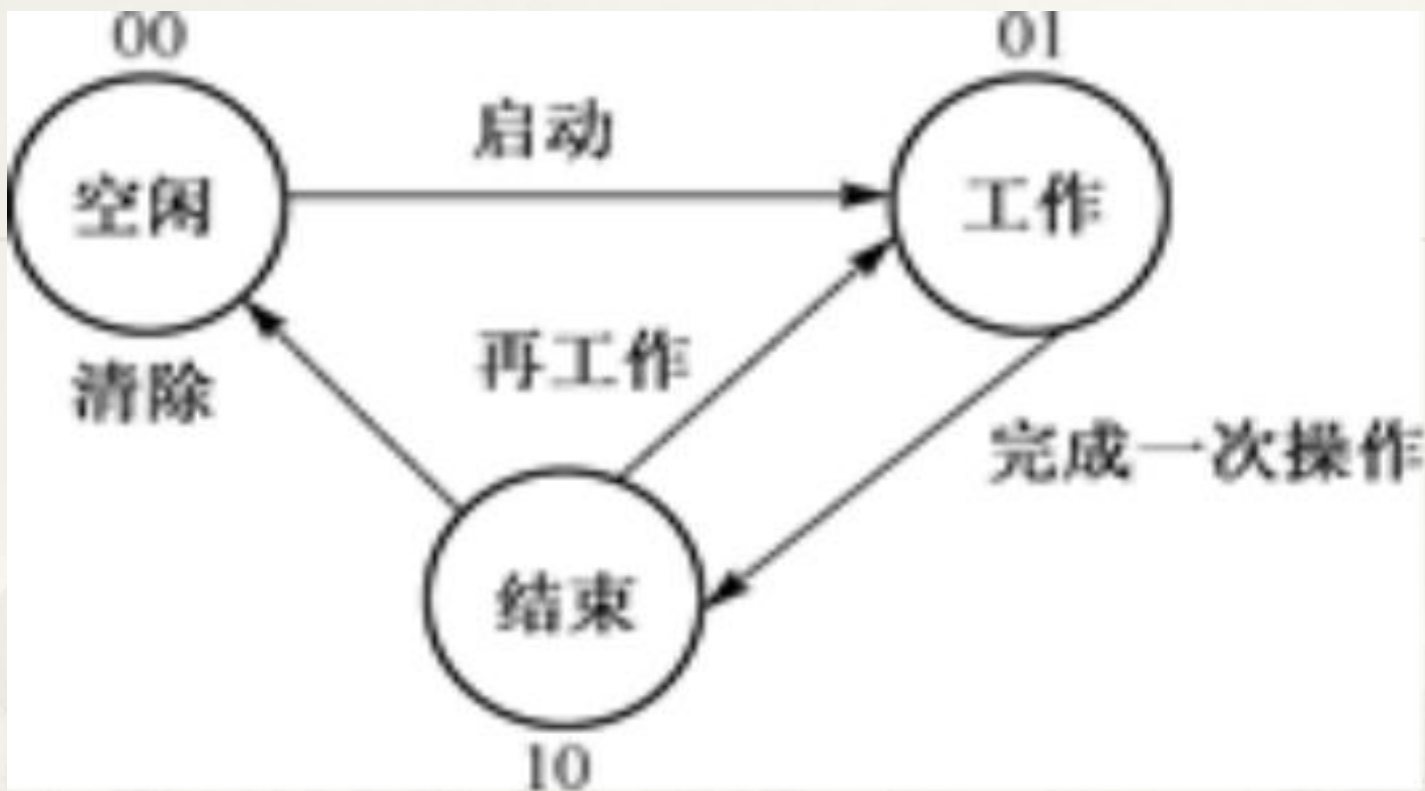
在外设接口的状态字中设置两位表状态:

空闲——外设不工作, 00

工作（忙）——外设置在执行操作, 01

结束（完成）——外设完成一次操作, 10

外设的状态转换如下:



四. 特点: CPU不能与外设并行地工作, 因而CPU利用率低, 并且CPU不能响应来自外部的随机请求. 因此, 只适用于低速外设。

5.4 DMA方式及接口

5.4.1 DMA方式及接口的基本概念

一. DMA方式定义:

DMA方式是依靠硬件直接在主存与外围设备之间进行的高速、批量、简单数据传送, 在传送过程中不需要CPU的干预。

二. DMA方式的实质、特点及典型应用

1. DMA方式的实质: 程序暂停

2. DMA方式的特点: 随机性

3. DMA方式的响应时机:一条指令完整结束后响应

4. DMA方式的典型应用:

- 1) DRAM芯片的异步刷新方式
- 2) 磁盘信息的读写

三. 几种控制方式的区别

1. 编写程序实现控制方式:程序控制传送、中断方式

硬件实现控制方式:DMA方式

2. 主机与I/O之间信息交换经过CPU的控制方式:程序控制传送、中断方式;

主机与I/O之间信息交换不经过CPU、只在主存与I/O之间的控制方式:DMA方式

3. 中断方式能处理随机复杂的事态, DMA方式只能处理简单的数据传送

四. DMA周期的流程:

ETi: 1→DMAT

与系统总线断开

DMAT: . 数据传送 (1个或n个总线周期)

.
1→FT

5.4.2 DMA方式硬件组织及传送方式

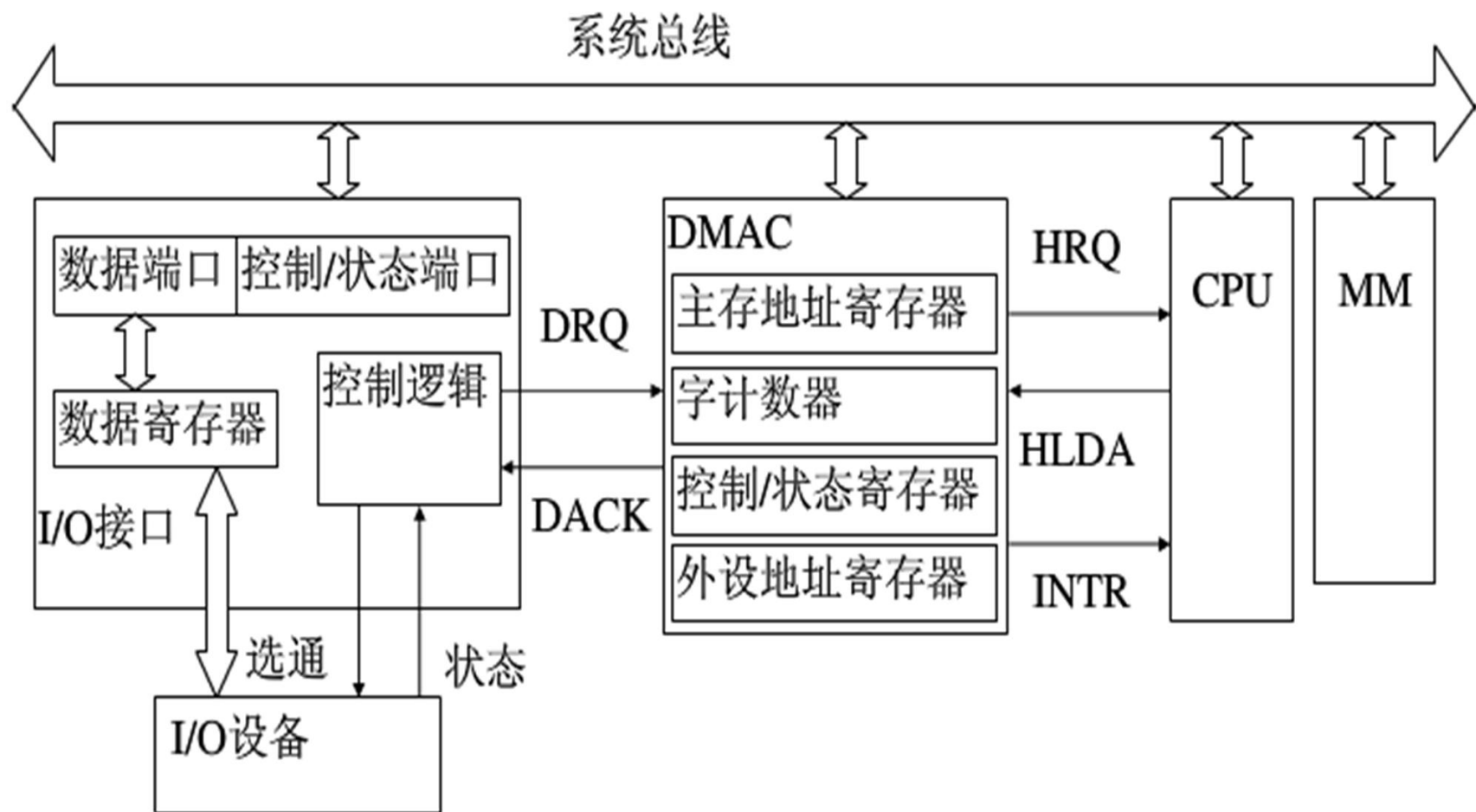
一. DMA方式硬件组织及传送操作方式

1. DMA方式硬件组织:

DMA接口: DMA控制器与接口 (分离)

DMA控制器: 负责申请、接管总线控制权, 发出传送命令和主存地址, 控制传送过程的起始。

接口: 实现连接和数据缓冲, 反映设备的特定要求。



DRQ: DMA请求

DACK: DMA响应

HRQ: 总线请求

HLDA: 总线响应

使用DMA方式进行数据传输时，主要有四种控制信号：

DRQ: DMA Request

DACK: DMA Acknowledgment

HRQ: Hold Request

HLDA: Hold Acknowledgment

每种控制信号的作用为：

DRQ：外设向DMAC提出的要进行DMA操作的申请信号。

DACK：DMAC向发出DRQ信号的外设的回应，表示收到请求和正在进行处理。

HRQ：DMAC向CPU发出的要求接管总线的请求信号。

HLDA：CPU向DMAC发出的允许接管总线的应答信号。

二、DMA传送方式

传送部件：M<====>I/O

方式

单字传送（1个总线周期）

成组连续传送（n个总线周期）

5.4.3 DMAC与接口的连接方式

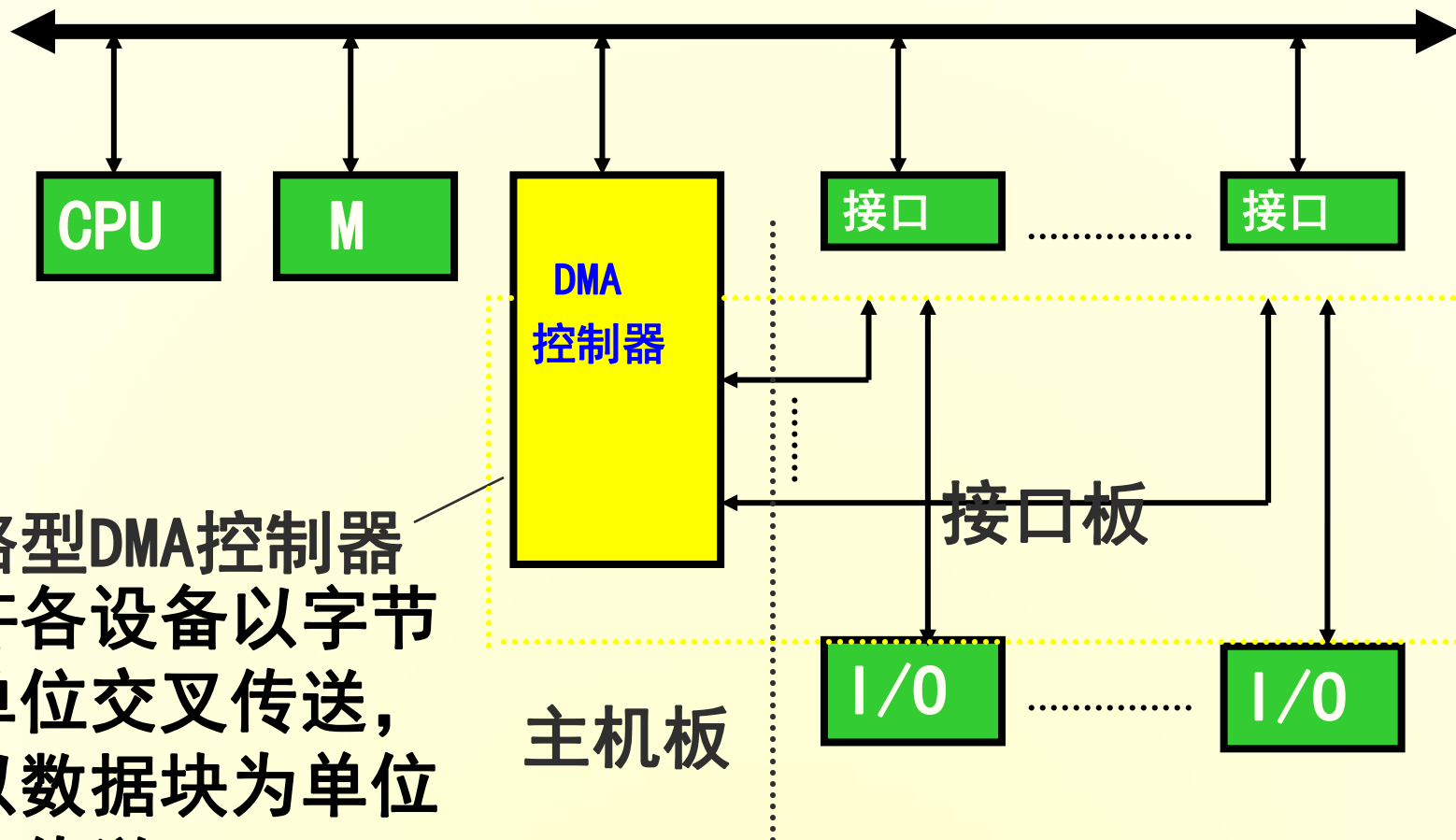
一、单通道型DMAC

二、选择型DMAC

三、多路型DMAC

DMA控制器与接口的连接（多路型）

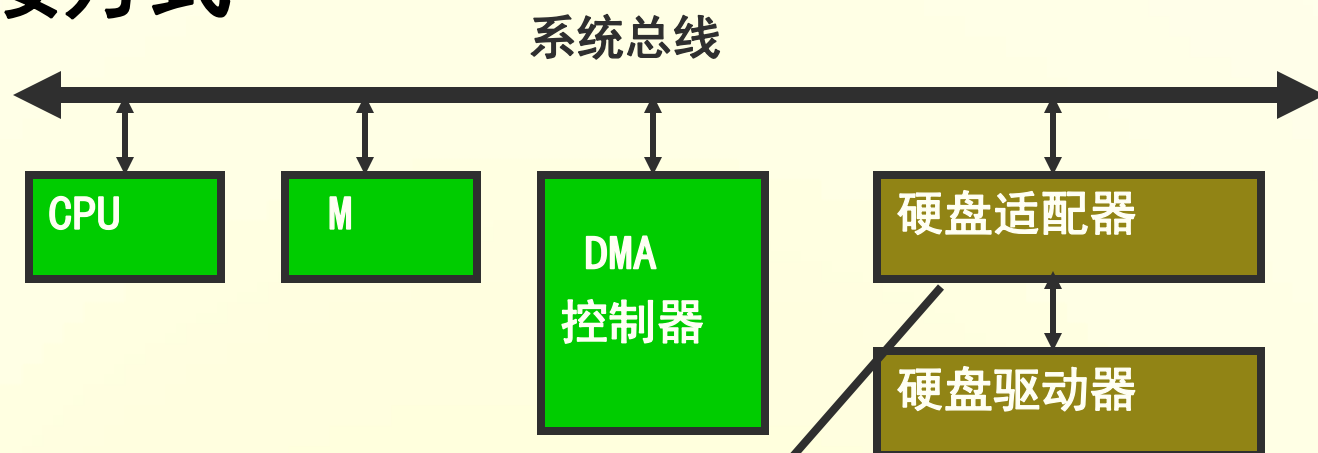
系统总线



多路型DMA控制器
允许各设备以字节
为单位交叉传送，
或以数据块为单位
成组传送。

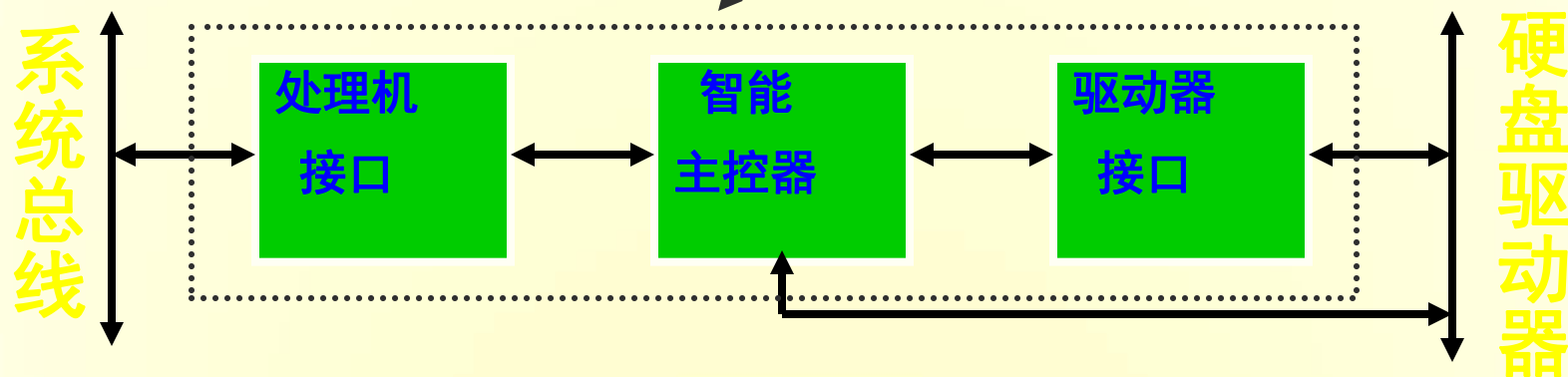
5.4.4 磁盘存储器接口（磁盘适配器）

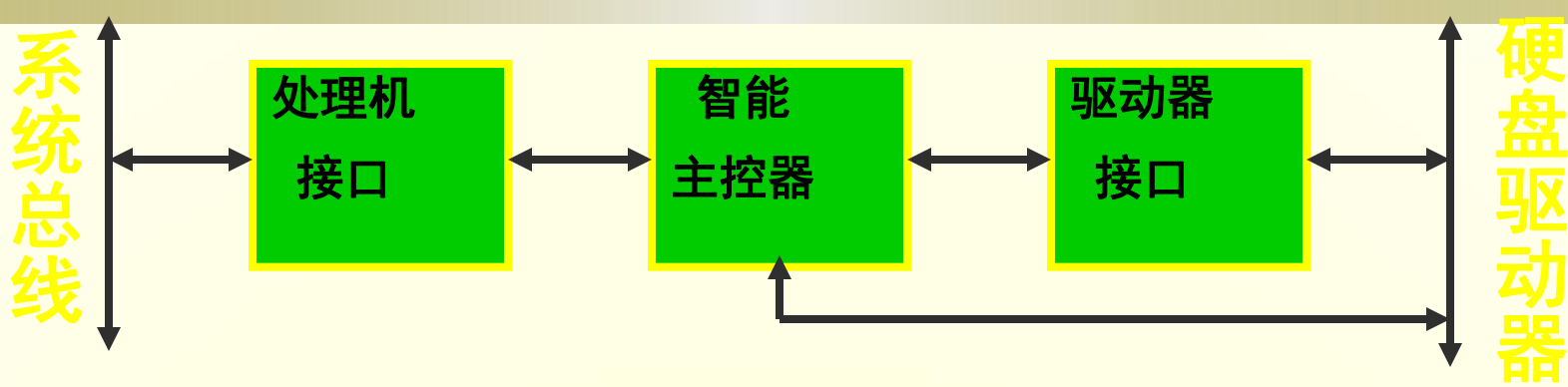
1. 系统连接方式



两级DMA控制器 { 主机板上DMA控制器: M ↔ 适配器
适配器内DMA控制器: 适配器 ↔ 驱动器

2. 硬盘适配器粗框





(1) 处理机接口 (面向系统总线一侧)

EPR0M控制逻辑: 放有硬盘驱动程序 (系统自检时被引入系统管理之下)。

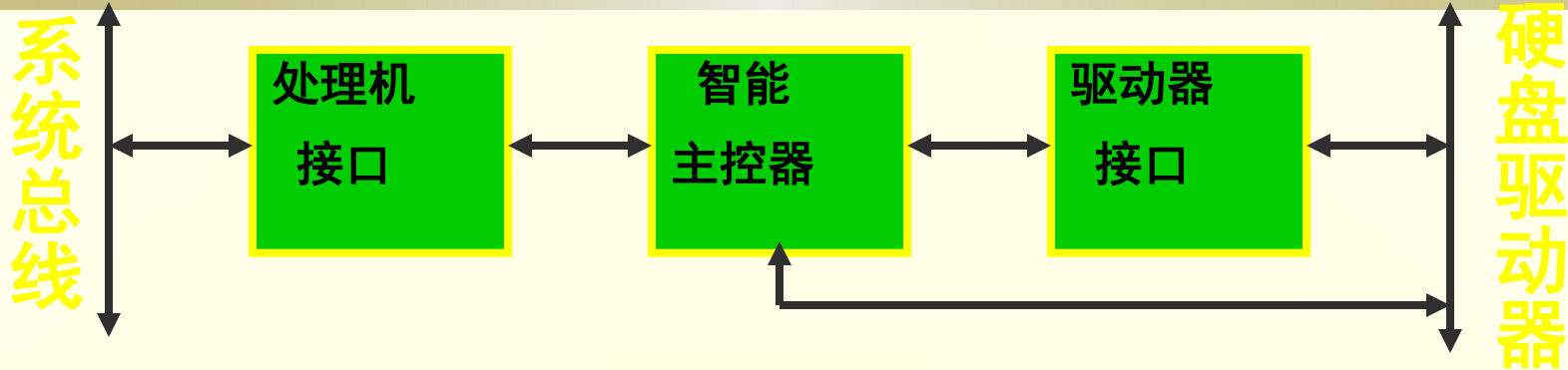
I/O端口控制逻辑: 接收CPU送来的端口地址、读/写命令, 访问处理机接口中的相应寄存器。

(2) 智能主控器

微处理器: 执行硬盘控制程序。

RAM: 扇区缓存 (存放二个扇区数据)。

ROM: 存放硬盘控制程序。



DMA控制器：控制主控RAM与驱动器之间的数据传送。

硬盘控制逻辑：控制串-并转换：

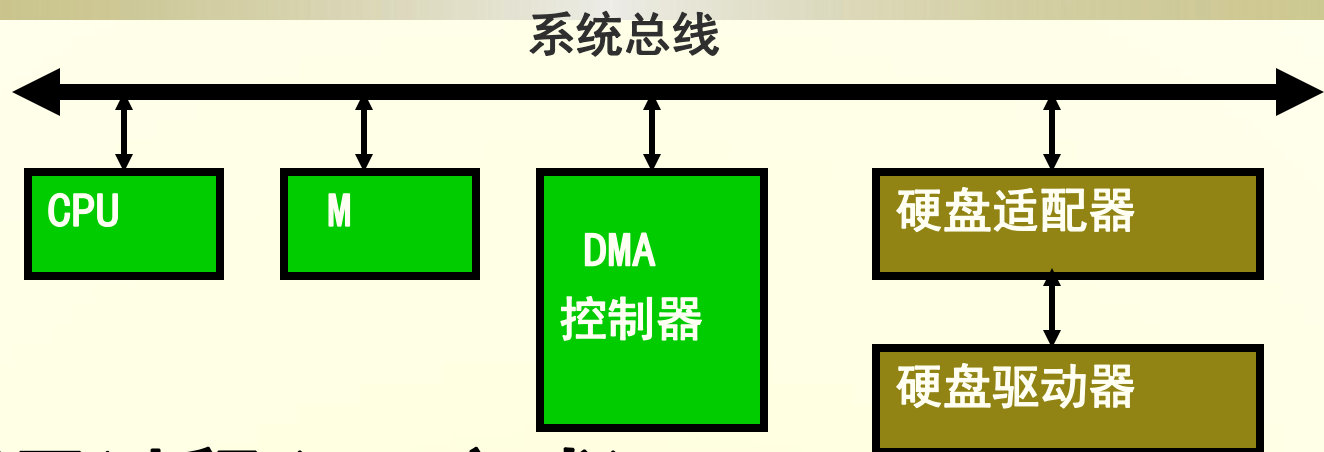
写盘：主控RAM 并-串 → 驱动器
读盘：驱动器 串-并 → 主控RAM

(3) 驱动器接口(面向设备一侧)

驱动器控制逻辑：向驱动器送出控制命令(驱动器选择、寻道方向选择、读、写……)。

驱动器状态逻辑：接收驱动器状态信息(选中、就绪、寻道完成……)。

传送串行数据。



3. 硬盘调用过程(DMA方式)

(1) CPU向适配器送出驱动器号、圆柱面号、磁头号、起始扇区号、扇区数等外设寻址信息；向DMA控制器送出传送方向、主存首址、交换量等信息。

(2) 适配器启动寻道，并用中断方式判寻道是否正确。

(不正确，重新寻道；正确，启动磁盘读/写。)

(3) 适配器准备好(读盘：主控RAM满一扇区；写盘：主控RAM空一扇区)，提出DMA请求。

(4) CPU响应，由DMA控制器控制总线，实现传送。

(5) 批量传送完毕，适配器申请中断。

(6) CPU响应，作善后处理。

第六章 输入/输出设备及接口

中央处理器（CPU）、主存储器（MM）构成计算机的主体，称为主机。主机以外的大部分硬件设备称为外围设备或输入/输出设备（I/O设备）。输入/输出接口（I/O接口）是主机与I/O设备之间的交接面，通过I/O接口可以实现主机与I/O设备之间的信息交流。

最常用的基本I/O设备：键盘、显示设备、打印设备、磁盘存储器、光驱设备等。

6.3 显示设备及接口

6.3.1 概述

显示系统 { 显示设备（硬件）
驱动程序（软件）

显示设备 { 控制器、接口 == 显示器适配卡（常称显卡）
显示器件

发光器件：外加电信号，发光器件将产生光辐射，从而发光。如：**CRT、LED、PDP、ELDP**等。

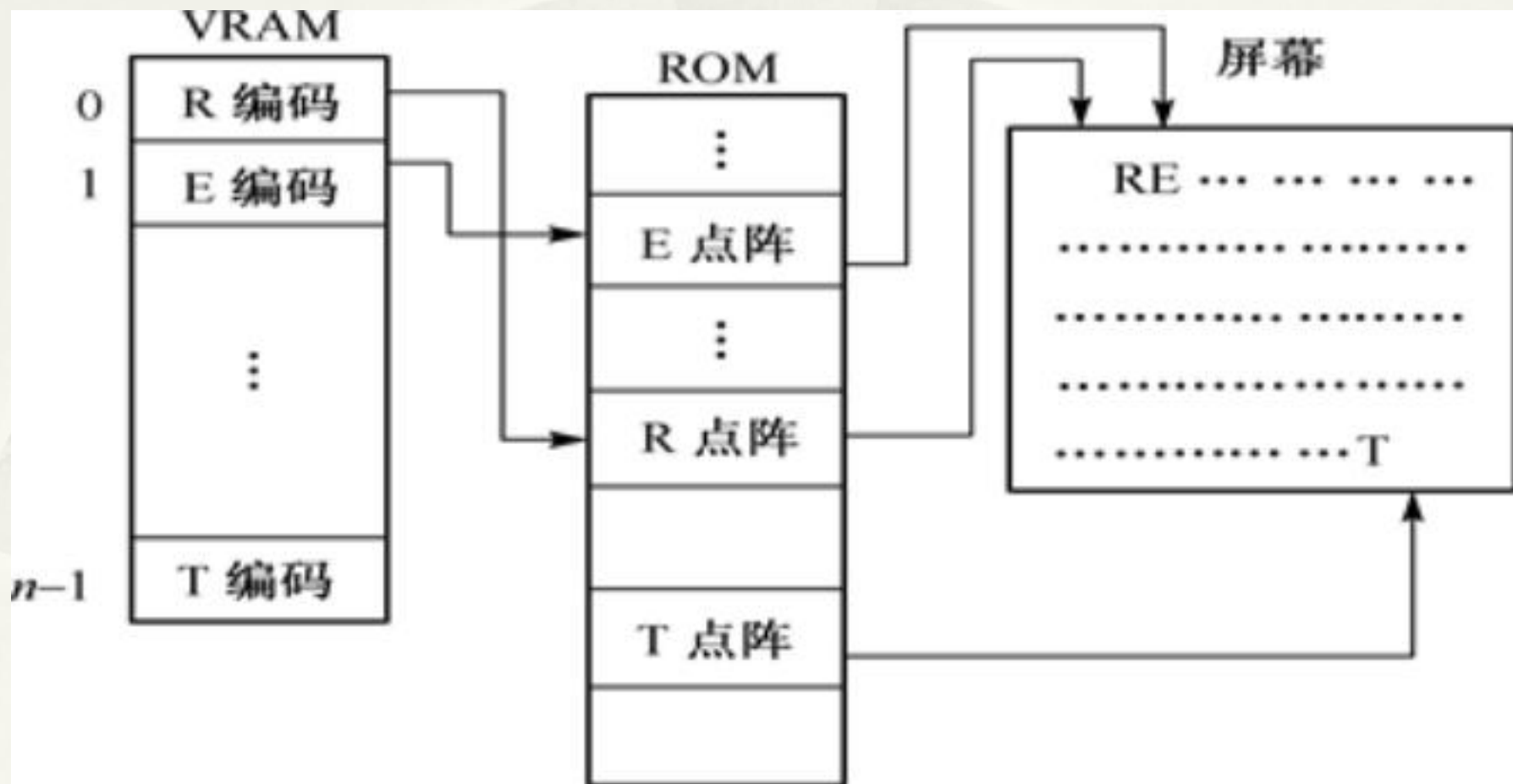
光调制器件：在外加电信号作用下，器件的局部区域的光特性发生变化，引起光线透过或反射。如：**LCD**。

6.3.2 显示技术中的相关术语

1. 图形：是指没有亮暗层次变化的线条图。
2. 图像：是指具有亮暗层次的图。
3. 分辨率：是指显示器所能表示的像素个数。
4. 灰度级：是指黑白显示器中所显示的像素点的亮暗差别，在彩色显示器中则表现为颜色的不同。
分辨率和灰度级决定了所显示图的质量。
5. 刷新：为了使人眼能看到稳定的图像显示，必须使电子束不断地重复扫描整个屏幕，这个过程叫做刷新。按人的视觉生理，刷新频率大于25次/秒时才不会感到闪烁。显示设备中通常选用每秒刷新70帧图像。
6. 扫描：电子束在荧光屏上按某种轨迹运动称为扫描，控制电子束扫描轨迹的电路叫扫描偏转电路。主要扫描方式有两种，随机扫描和光栅扫描。

6.3.3 屏幕显示与显示缓存间的对应关系

CRT显示工作原理图：如何将显存中的信息（字符编码/图形点代码）转换为字符/图形显示在屏幕上。





二、VRAM (Video RAM,独立显卡)

VRAM : 显示内容、属性内容两部分组成。

CRT显示器 { 字符方式 : 黑白显示器
图形方式 : 彩色显示器

基本显示缓存的容量和内容：

1.字符方式：

容量=行×列

内容：ASCII码或其它形式的编码

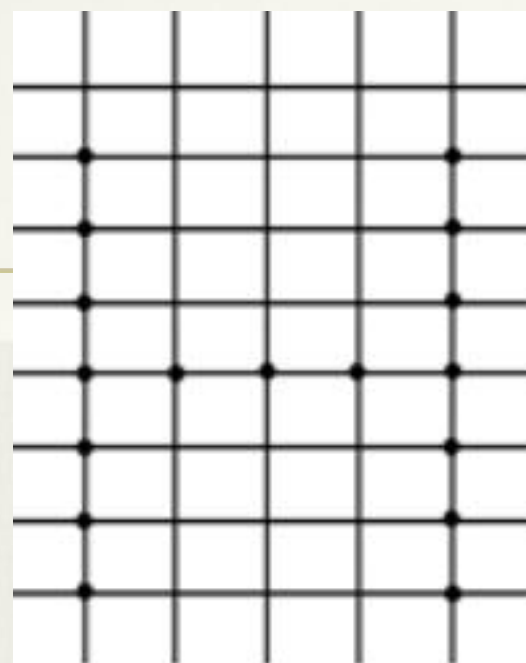
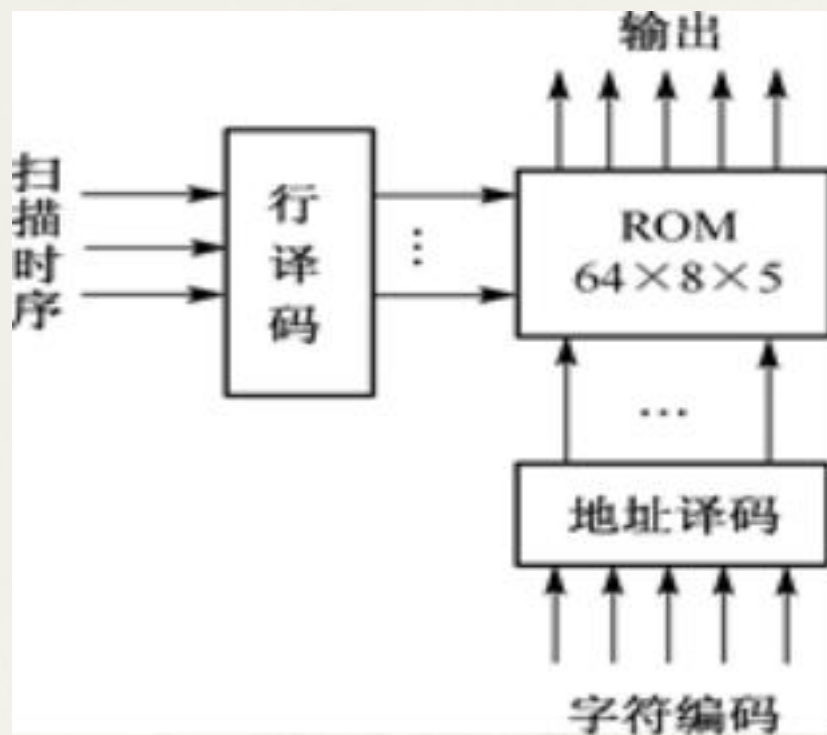
2.图形方式：

容量=（点×线）/8

内容：像素

三、字符发生器ROM

在CRT显示器中，用来产生字符点阵图形的器件称为字符发生器。它有专用的芯片，如APPLE-I所用的2513，采用5×8点阵，能产生64种字符的点阵信息。下图是2513芯片的逻辑框图。



(a) 点阵图

00000
10001
10001
10001
10001
11111
10001
10001
10001

(b) 点阵代码

1. 每个字符点阵容量 = 行容量占字节数 × 行数
 (若 $5 \leq \text{行容量} \leq 8$, 则行容量占字节数 = 1B ;
 若 $9 \leq \text{行容量} \leq 16$, 则行容量占字节数 = 2B)

字符发生器的容量 = 字符点阵容量 × 字符总数

2. 如何通过地址线访问字符发生器中的某个字符及取出字符点阵的各行代码？

通过高位地址访问字符发生器中的某个字符，低位地址取出字符点阵的行代码。

3. 屏幕上有多行，每行有多个字符，如何扫描实现？

电子束在进行全屏幕扫描时，是沿屏幕从左向右的方向扫描完一行光栅，再扫描第二行光栅。按照这种扫描方式，在显示字符时，并不是对一排的每个字符单独进行点阵扫描（即扫描完一个字符的各行点阵，再扫描同排另一字符的各行点阵），而是采用对一排的所有字符的点阵进行逐行依次扫描。

四、同步控制

不论字符显示还是图形显示，都要求行、场扫描和视频信号的发送在时间上要完全同步，即当电子束扫描到某字符或某像点的位置时，相应的视频信号必须同时输出。为此，在CRT显示器中设置了几个计数器，对显示器的主频脉冲进行分频，产生各种时序信号来控制对VRAM的访问、对CRT的水平扫描和垂直扫描，以及视频信号的产生等。

1. 字符方式：设置了点、字符、线、行四级同步计数器。
如：在PC机中最常用的显示规格为：每帧最大显示25行 \times 80列，字符点阵7 \times 9，字符区间9 \times 14。

字符计数

点计数

线计数

行计数

7

9

5

2

.....

0 行

⋮

24 行

.....

0 列

79 列

(1) 点计数分频 $(7+2) : 1$

对一个字符的一行点计数。

一次点计数循环访问一次VRAM、ROM。

(2) 字符计数分频 $(80+L) : 1$

对一帧的字符列计数。

一次字符计数循环发一次水平同步信号。

字符计数值提供VRAM列地址（低地址）。

(3) 线计数分频 $(9+5) : 1$

对一行字符的扫描线计数。

线计数值提供ROM低位地址。

(4) 行计数分频 $(25+M) : 1$

对一帧的字符行计数。

一次行计数循环发一次垂直同步信号。

行计数值提供VRAM行地址（高地址）。

字符显示的同步控制关系小结：

点计数一个循环，访问 VRAM一次以读取显示字符的编码，VRAM的地址根据行计数值与字符计数值决定；每读一次VRAM，就读一次字符发生器 ROM，由VRAM读出的字符编码产生ROM的高位地址，线计数值决定 ROM的低位地址；

每次从ROM中读出显示字符的一线7位，由点脉冲控制逐位显示7点（亮或暗）；

由于每条水平扫描线只能显示一行字符（可多达80个）的一线，所以上述访问 VRAM和ROM的过程需要重复9遍，才能显示完整的一行字符；

字符计数循环一次，发一次水平同步信号；
行计数循环一次，发一次垂直同步信号。

2. 图形方式：设置了点、字节、线三级同步计数器。

如：某彩色显示器的分辨率为：640点 \times 480线

(1) 点计数分频8：1

(2) 字节计数分频(80+L)：1

字节计数器又称为列计数器， $640/8=80$ (B)

(3) 线计数分频(480+M)：1