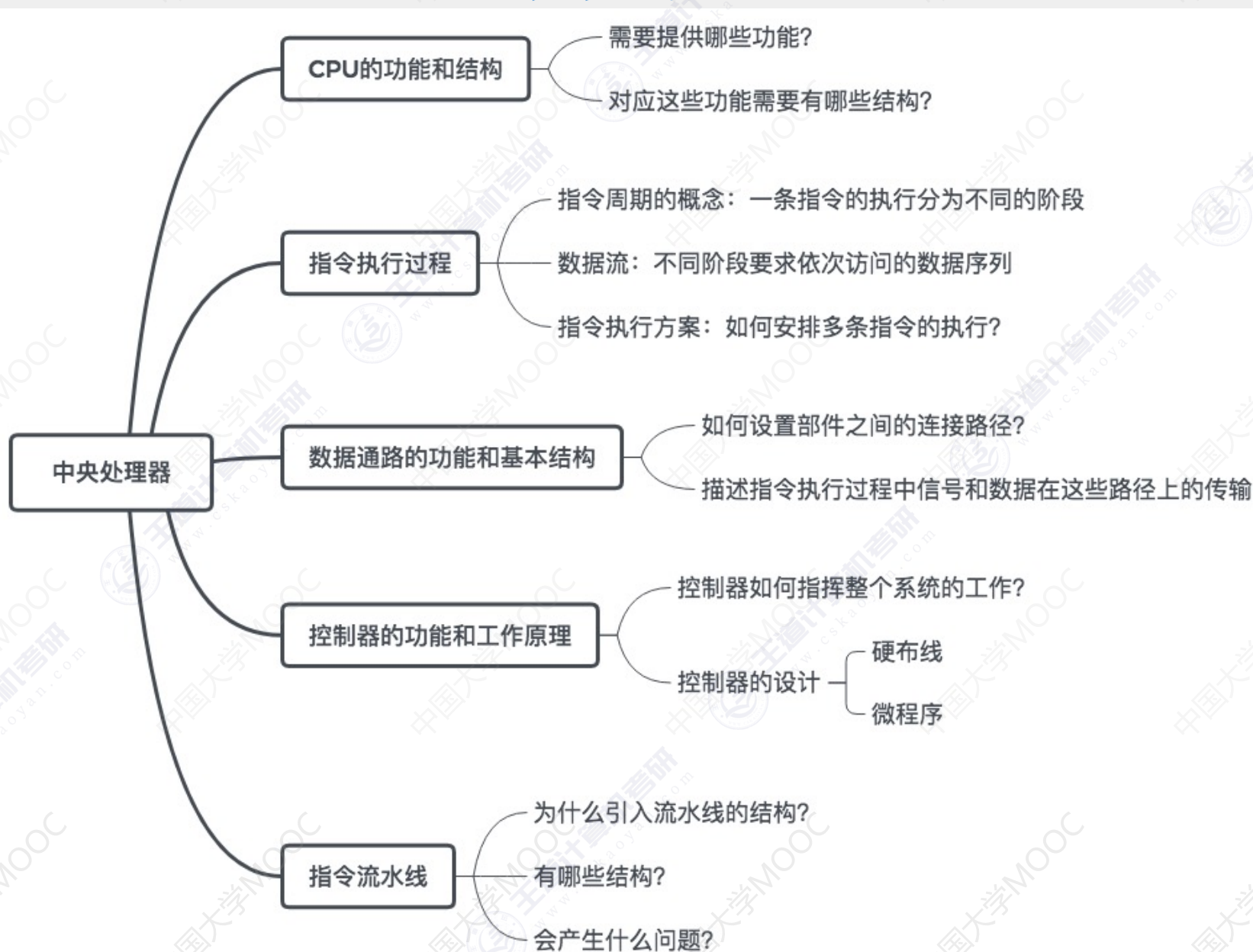


本节内容

# 控制器设计

## 硬布线控制器

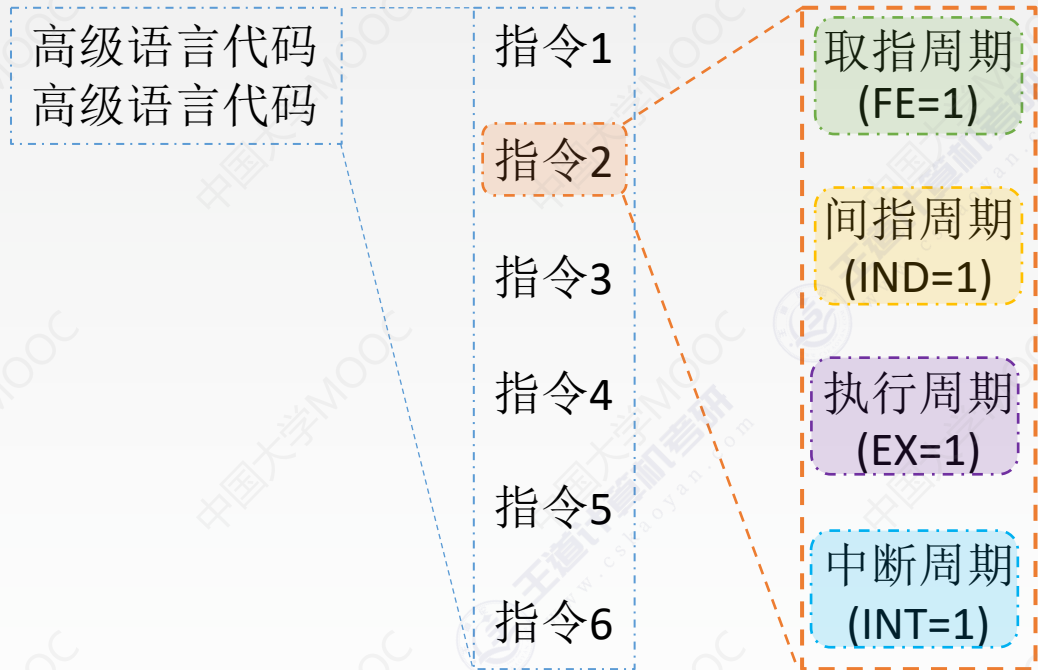
# 本章总览



根据 指令操作码、目前的机器周期、节拍信号、机器状态条件，即可确定现在这个节拍下应该发出哪些“微命令”

## 内容回顾

CU发出一个微命令，可完成对应微操作。  
如：微命令1使得  $PC_{out}$ 、 $MAR_{in}$  有效。  
完成对应的微操作1 ( $PC \rightarrow MAR$ )



$T_0$ : 微操作1、微操作2  
 $T_1$ : 微操作3  
 $T_2$ : 微操作4

一个节拍内可以并行完成多个“相容的”微操作

$T_0$ : 微操作5、微操作2  
 $T_1$ : 微操作6  
 $T_2$ : 微操作7

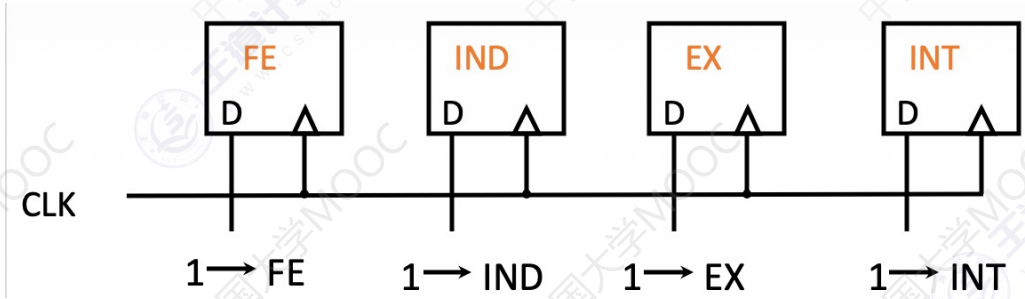
同一个微操作可能在不同指令的不同阶段被使用

$T_0$ :  
 $T_1$ : 微操作8  
 $T_2$ : 微操作9、微操作6

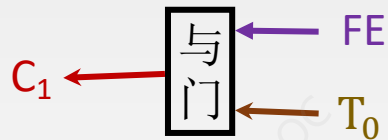
不同指令的执行周期所需节拍数各不相同。为了简化设计，选择定长的机器周期，以可能出现的最大节拍数为准（通常以访存所需节拍数作为参考）

$T_0$ :  
 $T_1$ : 微操作10  
 $T_2$ : 微操作11

若实际所需节拍数较少，可将微操作安排在机器周期末尾几个节拍上进行

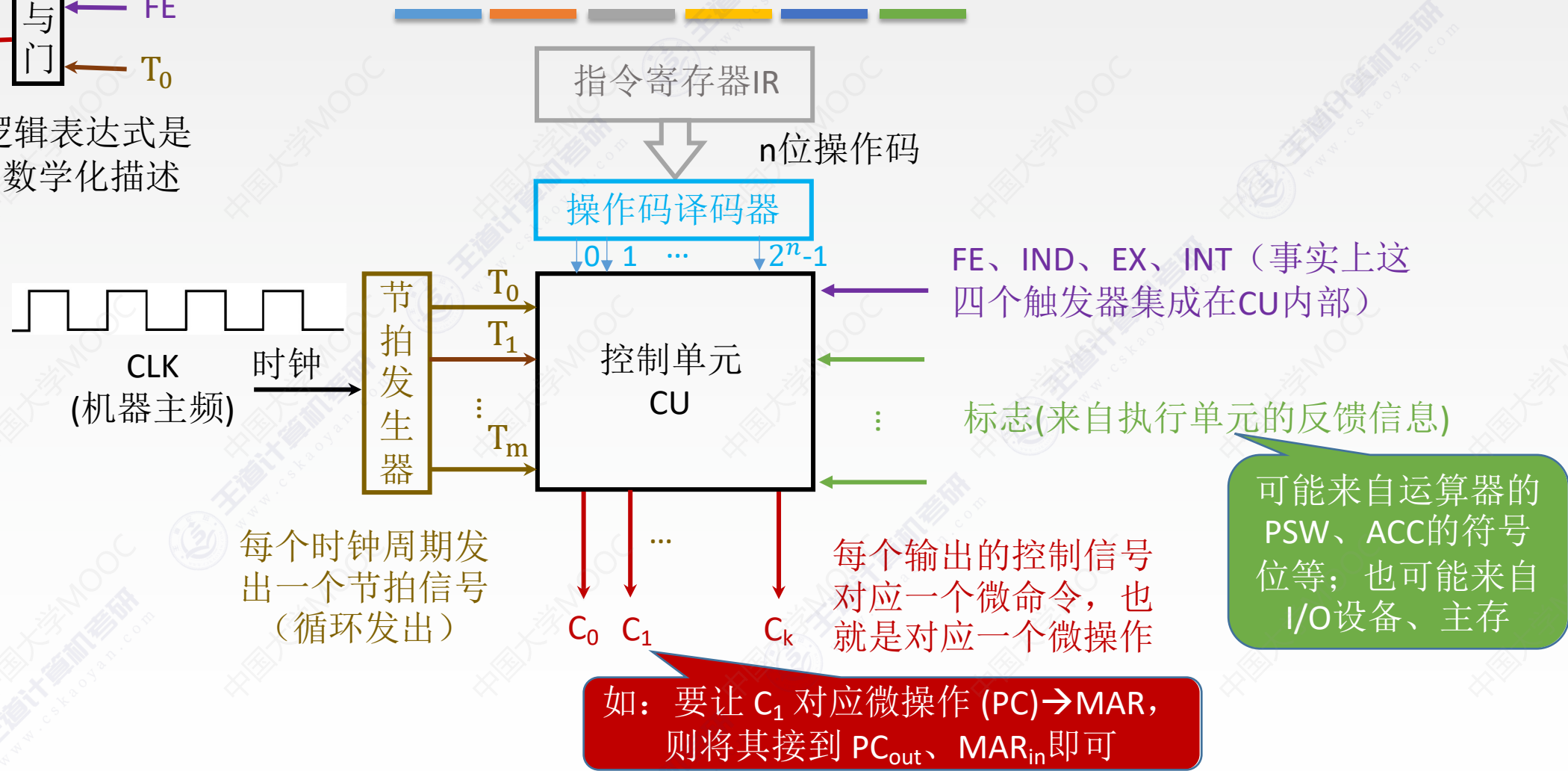


所有指令的取指周期、 $T_0$  节拍下一定要完成  $(PC) \rightarrow MAR$ 。则可知  $C_1 = FE \cdot T_0$



Tips: 逻辑表达式是电路的数字化描述

# 硬布线控制器



根据 指令操作码、目前的机器周期、节拍信号、机器状态条件，即可确定现在这个节拍下应该发出哪些“微命令”



用平静掩饰恐惧

颤抖吧！感受恐惧！

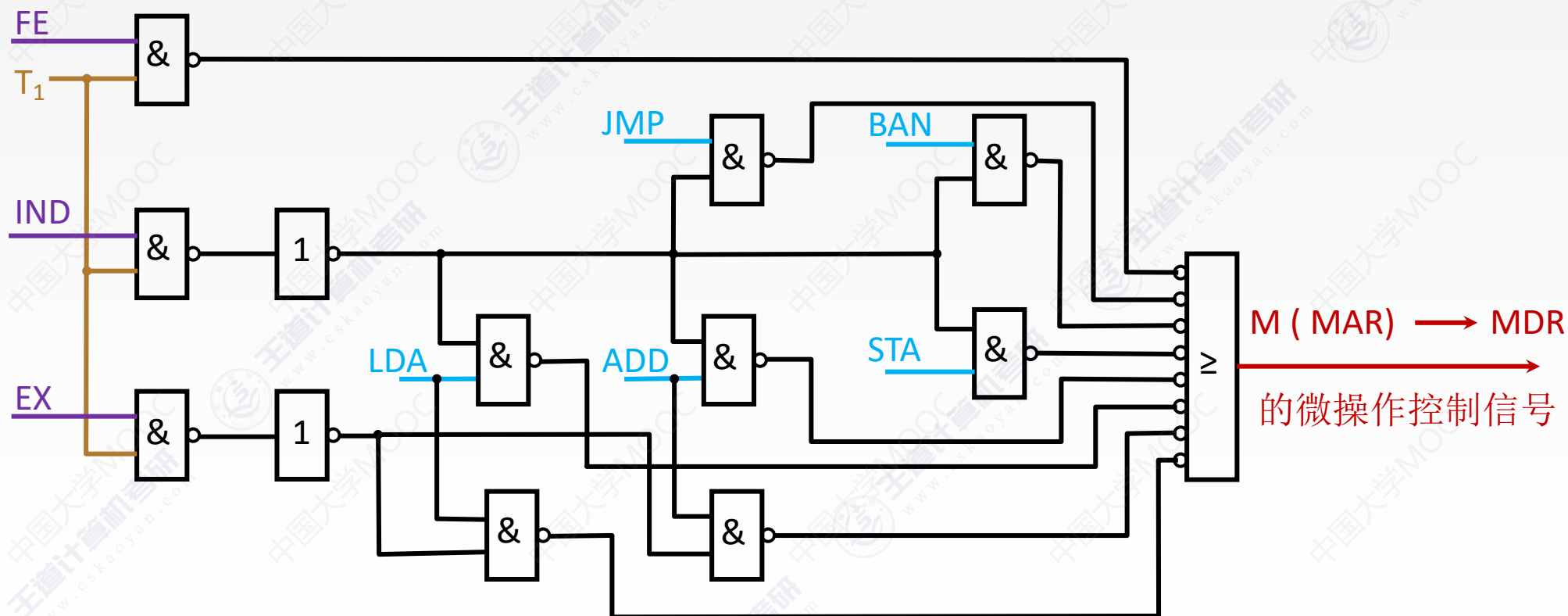


很紧张对吧？

注：一般不考电路，莫慌~

$M(MAR) \rightarrow MDR$  微操作命令的逻辑表达式：

$$FE \cdot T_1 + IND \cdot T_1 (ADD + STA + LDA + JMP + BAN) + EX \cdot T_1 (ADD + LDA)$$



根据 指令操作码、目前的机器周期、节拍信号、机器状态条件，即可确定  
现在这个节拍下应该发出哪些“微命令”



# 硬布线控制器的设计

设计步骤:

1. 分析每个阶段的微操作序列（取值、间址、执行、中断 四个阶段）

2. 选择CPU的控制方式

3. 安排微操作时序

4. 电路设计

确定哪些指令在什么阶段、在什么条件下会使用到的微操作

采用定长机器周期还是不定长机器周期？每个机器周期安排几个节拍？

如何用3个节拍完成整个机器周期内的所有微操作？

确定每个微操作命令的逻辑表达式，并用电路实现

假设采用同步控制方式（定长机器周期），一个机器周期内安排3个节拍。

安排，必须安排



注：中断周期内的微操作序列就不分析了，原理类似

## 分析每个阶段的微操作序列

取指周期（所有指令都一样）

$PC \rightarrow MAR$

$1 \rightarrow R$

$M(MAR) \rightarrow MDR$

$MDR \rightarrow IR$

$OP(IR) \rightarrow ID$

$(PC) + 1 \rightarrow PC$

注：ID 是指令译码器  
Instruction Decoder

间址周期（所有指令都一样）

$Ad(IR) \rightarrow MAR$

$1 \rightarrow R$

$M(MAR) \rightarrow MDR$

$MDR \rightarrow Ad(IR)$

执行周期（各不相同）

注：很多地方把  
ACC简写为AC

CLA

$0 \rightarrow AC$

clear ACC 指令  
ACC清零

LDA X

取数指令，  
把X所指内容  
取到ACC

$Ad(IR) \rightarrow MAR$

$1 \rightarrow R$

$M(MAR) \rightarrow MDR$

$MDR \rightarrow AC$

JMP X

无条件转移

$Ad(IR) \rightarrow PC$

负数符号位为1

BAN X

Branch ACC Negative

条件转移，当ACC为负时转移

$A_0 \bullet Ad(IR) + \overline{A_0} \bullet (PC) \rightarrow PC$

罗列出所有指令在各个阶段的微操作序列，就可以知道在什么情况下需要使用这个微操作

根据指令操作码、目前的机器周期、节拍信号、机器状态条件，即可确定现在这个节拍下应该发出哪些“微命令”

# 安排微操作时序的原则

原则一 微操作的 先后顺序不得 随意 更改

原则二 被控对象不同 的微操作

尽量安排在 一个节拍 内完成

原则三 占用 时间较短 的微操作

尽量 安排在 一个节拍 内完成

并允许有先后顺序



## 安排微操作时序-取指周期



原则一 微操作的 先后顺序不得 随意 更改

原则二 被控对象不同 的微操作

尽量安排在一个节拍 内完成

原则三 占用 时间较短 的微操作

尽量 安排在一个节拍 内完成

并允许有先后顺序

(1)  $PC \rightarrow MAR$

(2)  $1 \rightarrow R$

(3)  $M(MAR) \rightarrow MDR$

(4)  $MDR \rightarrow IR$

(5)  $OP(IR) \rightarrow ID$

(6)  $(PC) + 1 \rightarrow PC$

存储器空闲即可

在(1)之后

在(3)之后

在(4)之后

在(1)之后

## 安排微操作时序-取指周期

原则一 微操作的 先后顺序不得 随意 更改

原则二 被控对象不同 的微操作

尽量安排在一个节拍 内完成

原则三 占用 时间较短 的微操作

尽量 安排在一个节拍 内完成

并允许有先后顺序

$T_0$  (1)  $PC \rightarrow MAR$

$T_0$  (2)  $1 \rightarrow R$

存储器空闲即可

$T_1$  (3)  $M(MAR) \rightarrow MDR$

在(1)之后

$T_1$  (6)  $(PC) + 1 \rightarrow PC$

在(1)之后

$T_2$  (4)  $MDR \rightarrow IR$

在(3)之后

$T_2$  (5)  $OP(IR) \rightarrow ID$

在(4)之后

两个微操作占用时间较短，根据原则三安排在一个节拍

$M(MAR) \rightarrow MDR$  从主存取数据，用时较长，因此必须一个时钟周期才能保证微操作的完成

$MDR \rightarrow IR$  是CPU内部寄存器的数据传送，速度很快，因此在一个时钟周期内可以紧接着完成  $OP(IR) \rightarrow ID$ 。  
也就是可以一次同时发出两个微命令。

# 安排微操作时序-间址周期



原则一 微操作的 先后顺序不得 随意 更改

$T_0$  (1)  $Ad(IR) \rightarrow MAR$

原则二 被控对象不同 的微操作

$T_0$  (2)  $1 \rightarrow R$

尽量安排在一个节拍 内完成

$T_1$  (3)  $M(MAR) \rightarrow MDR$

原则三 占用 时间较短 的微操作

$T_2$  (4)  $MDR \rightarrow Ad(IR)$

尽量 安排在一个节拍 内完成

并允许有先后顺序

# 安排微操作时序-执行周期



原则一 微操作的 先后顺序不得 随意 更改

原则二 被控对象不同 的微操作

尽量安排在一个节拍 内完成

原则三 占用 时间较短 的微操作

尽量 安排在一个节拍 内完成

并允许有先后顺序

① CLA

clear

ACC清零

$T_0$

$T_1$

$T_2$

$0 \rightarrow AC$

② COM

complement

ACC取反

$T_0$

$T_1$

$T_2$

$\overline{AC} \rightarrow AC$

③ SHR

shift

算术右移

$T_0$

$T_1$

$T_2$

$T_2$

$L(AC) \rightarrow R(AC)$

$AC_0 \rightarrow AC_0$

④ CSL

cyclic shift

循环左移

$T_0$

$T_1$

$T_2$

$R(AC) \rightarrow L(AC), AC_0 \rightarrow AC_n$

⑤ STP

stop

停机

$T_0$

$T_1$

$T_2$

$0 \rightarrow G$

# 安排微操作时序-执行周期

## (1) 非访存指令

① CLA	$T_0$	
clear	$T_1$	
ACC清零	$T_2$	$0 \rightarrow AC$
② COM	$T_0$	
complement	$T_1$	
ACC取反	$T_2$	$\overline{AC} \rightarrow AC$
③ SHR	$T_0$	
shift	$T_1$	
算术右移	$T_2$	$L(AC) \rightarrow R(AC)$
	$T_2$	$AC_0 \rightarrow AC_0$
④ CSL	$T_0$	
cyclic shift	$T_1$	
循环左移	$T_2$	$R(AC) \rightarrow L(AC), AC_0 \rightarrow AC_n$
⑤ STP	$T_0$	
stop	$T_1$	
停机	$T_2$	$0 \rightarrow G$

## (2) 访存指令

⑥ ADD X	$T_0$	$Ad(IR) \rightarrow MAR, 1 \rightarrow R$
加法指令	$T_1$	$M(MAR) \rightarrow MDR$
隐含ACC	$T_2$	$(AC) + (MDR) \rightarrow AC$
⑦ STA X	$T_0$	$Ad(IR) \rightarrow MAR, 1 \rightarrow W$
存数指令	$T_1$	$AC \rightarrow MDR$
隐含ACC	$T_2$	$MDR \rightarrow M(MAR)$
⑧ LDA X	$T_0$	$Ad(IR) \rightarrow MAR, 1 \rightarrow R$
取数指令	$T_1$	$M(MAR) \rightarrow MDR$
隐含ACC	$T_2$	$MDR \rightarrow AC$

## (3) 转移指令

⑨ JMP X	$T_0$	
jump	$T_1$	
无条件转移	$T_2$	$Ad(IR) \rightarrow PC$
⑩ BAN X	$T_0$	
Branch ACC	$T_1$	
Negative	$T_2$	$A_0 \cdot Ad(IR) + \overline{A_0} \cdot (PC) \rightarrow PC$
条件转移		

# 安排微操作时序-中断周期



原则一 微操作的 先后顺序不得 随意 更改

原则二 被控对象不同 的微操作

尽量安排在一个节拍 内完成

原则三 占用 时间较短 的微操作

尽量 安排在一个节拍 内完成

并允许有先后顺序

$T_0$  (1)  $a \rightarrow \text{MAR}$

$T_0$  (2)  $1 \rightarrow W$

存储器空闲即可

$T_0$  (3)  $0 \rightarrow \text{EINT}$

硬件关中断

$T_1$  (4)  $(PC) \rightarrow \text{MDR}$

内部数据通路空闲即可

$T_2$  (5)  $\text{MDR} \rightarrow \text{M}(\text{MAR})$

在(3)之后

$T_2$  (6) 向量地址  $\rightarrow PC$

在(3)之后

这些操作由中断隐指令完成

注：中断隐指令不是一条指令，而是指一条指令的中断周期由硬件完成的一系列操作

中断周期的三个任务：

1. 保存断点
2. 形成中断服务程序的入口地址
3. 关中断

设计步骤：

1. 分析每个阶段的微操作序列
2. 选择CPU的控制方式
3. 安排微操作时序
4. 电路设计



# 组合逻辑设计



设计步骤:

1. 列出操作时间表

列出在取指、间址、执行、中断周期，T0、T1、T2 节拍内有可能用到的所有微操作

2. 写出微操作命令的最简表达式

3. 画出逻辑图

# 组合逻辑设计

设计步骤:

1. 列出操作时间表

非访存指令

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	SHR	CSL	STP	ADD	STA	LDA	JMP	BAN
FE 取指	T <sub>0</sub>		PC → MAR	1	1	1	1	1	1	1	1	1	1
			1 → R	1	1	1	1	1	1	1	1	1	1
	T <sub>1</sub>		M(MAR) → MDR	1	1	1	1	1	1	1	1	1	1
			(PC) + 1 → PC	1	1	1	1	1	1	1	1	1	1
	T <sub>2</sub>		MDR → IR	1	1	1	1	1	1	1	1	1	1
			OP(IR) → ID	1	1	1	1	1	1	1	1	1	1
		I	1 → IND						1	1	1	1	1
		$\bar{I}$	1 → EX	1	1	1	1	1	1	1	1	1	1

间址特征

# 组合逻辑设计

设计步骤:

1. 列出操作时间表

非访存指令

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	SHR	CSL	STP	ADD	STA	LDA	JMP	BAN
IND 间址	T <sub>0</sub>		Ad (IR) → MAR						1	1	1	1	1
			1 → R						1	1	1	1	1
	T <sub>1</sub>		M(MAR) → MDR						1	1	1	1	1
	T <sub>2</sub>		MDR → Ad (IR)						1	1	1	1	1
		$\overline{\text{IND}}$	1 → EX						1	1	1	1	1

间址周期标志

# 组合逻辑设计

设计步骤:

1. 列出操作时间表

2. 写出微操作命令的最简表达式

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	ADD	STA	LDA	JMP	BAN
EX 执行	T <sub>0</sub>		Ad (IR) → MAR			1	1	1		
			1 → R			1		1		
			1 → W				1			
	T <sub>1</sub>		M(MAR) → MDR			1		1		
			AC → MDR				1			
	T <sub>2</sub>		(AC)+(MDR) → AC			1				
			MDR → M(MAR)				1			
			MDR → AC					1		
			0 → AC	1						
			$\overline{AC} \rightarrow AC$		1					
			Ad(IR) → PC						1	
		A <sub>0</sub>	Ad(IR) → PC							1

## 微操作信号综合

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	SHR	CSL	STP	ADD	STA	LDA	JMP	BAN
FE 取指	T <sub>0</sub>		PC → MAR	1	1	1	1	1	1	1	1	1	1
			1 → R	1	1	1	1	1	1	1	1	1	1
	T <sub>1</sub>		M(MAR) → MDR	1	1	1	1	1	1	1	1	1	1
IND 间址	T <sub>1</sub>		M(MAR) → MDR						1	1	1	1	1
EX 执行	T <sub>1</sub>		1 → W							1			
			M(MAR) → MDR						1		1		

M(MAR) → MDR微操作命令的逻辑表达式:

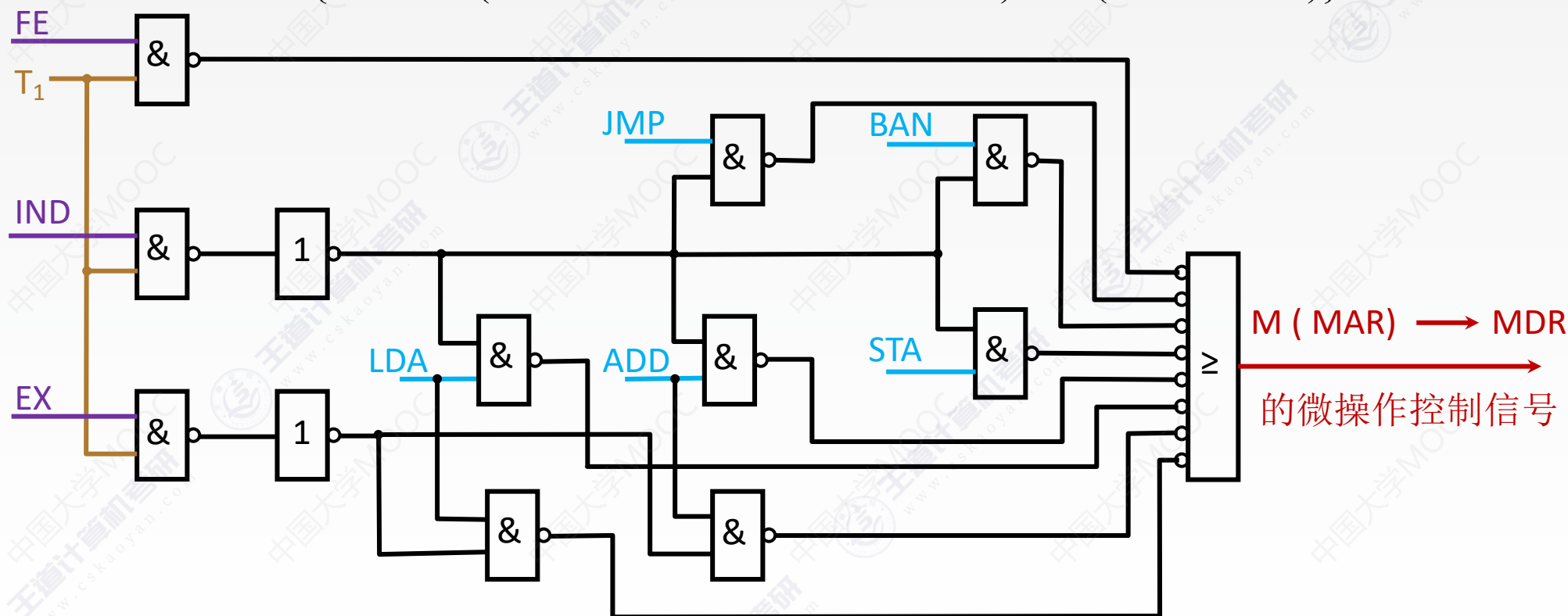
$$\begin{aligned}
 & \text{FE} \cdot T_1 + \text{IND} \cdot T_1 (\text{ADD} + \text{STA} + \text{LDA} + \text{JMP} + \text{BAN}) + \text{EX} \cdot T_1 (\text{ADD} + \text{LDA}) \\
 & = T_1 \{ \text{FE} + \text{IND} (\text{ADD} + \text{STA} + \text{LDA} + \text{JMP} + \text{BAN}) + \text{EX} (\text{ADD} + \text{LDA}) \}
 \end{aligned}$$

## 画出逻辑图

M (MAR) → MDR 微操作命令的逻辑表达式:

$$FE \cdot T_1 + IND \cdot T_1 (ADD + STA + LDA + JMP + BAN) + EX \cdot T_1 (ADD + LDA)$$

$$= T_1 \{ FE + IND (ADD + STA + LDA + JMP + BAN) + EX (ADD + LDA) \}$$



根据 指令操作码、目前的机器周期、节拍信号、机器状态条件，即可确定  
现在这个节拍下应该发出哪些“微命令”



# 硬布线控制器的设计



设计步骤:

1. 分析每个阶段的微操作序列
2. 选择CPU的控制方式
3. 安排微操作时序
4. 电路设计
  - (1) 列出操作时间表
  - (2) 写出微操作命令的最简表达式
  - (3) 画出逻辑图

硬布线控制器的特点:

指令越多, 设计和实现就越复杂, 因此一般用于 RISC (精简指令集系统)

如果扩充一条新的指令, 则控制器的设计就需要大改, 因此扩充指令较困难。

由于使用纯硬件实现控制, 因此执行速度很快。微操作控制信号由组合逻辑电路即时产生。



公众号：王道在线



b站：王道计算机教育



抖音：王道计算机考研