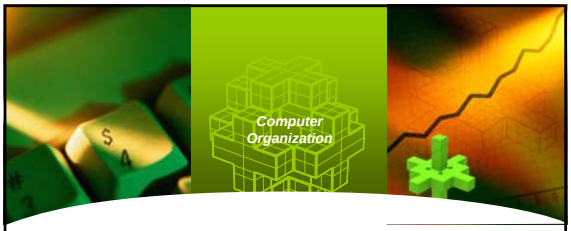


第 9 章　控制单元的功能

系统结构研究所 · 计算机组成原理

第 9 章　控制单元的功能

9.1　操作命令的分析

完成一条指令分 4 个工作周期

取指周期

间址周期

执行周期

中断周期

一、取指周期

PC ⟶ MAR ⟶ 地址线

1 ⟶ R　(启动主存读操作)

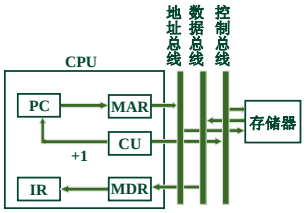
M (MAR) ⟶ MDR

（将MAR所指的主存单元的内容读至MDR）

MDR ⟶ IR

OP（IR）⟶ CU

(PC) + 1 ⟶ PC



二、间址周期

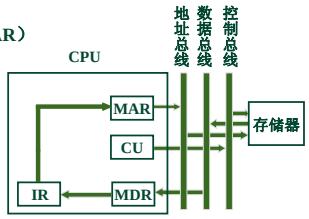
Ad (IR) ⟶ MAR

（指令形式地址 ⟶ MAR）

1 ⟶ R

M (MAR) ⟶ MDR

MDR ⟶ Ad (IR)



三、执行周期

1. 非访存指令

(1) CLA　清A　　0 ⟶ ACC

(2) COM　取反　　$\overline{ACC}$ ⟶ ACC

(3) SHR　算术右移　L(ACC) ⟶ R(ACC), (ACC_0 ⟶ ACC_0)

(4) CSL　循环左移　R(ACC) ⟶ L(ACC), (ACC_0 ⟶ ACC_n)

(5) STP　停机指令　0 ⟶ G　（运行标志触发器）

三、执行周期

9.1

2. 访存指令

(1) 加法指令 ADD X

Ad(IR) ⟶ MAR

1 ⟶ R

M(MAR) ⟶ MDR

(ACC) + (MDR) ⟶ ACC

(2) 存数指令 STA X

Ad(IR) ⟶ MAR

1 ⟶ W

ACC ⟶ MDR

MDR ⟶ M(MAR)

9.1

(3) 取数指令 LDA X

Ad (IR) ⟶ MAR

1 ⟶ R

M (MAR) ⟶ MDR

MDR ⟶ ACC

3. 转移指令

(1) 无条件转 JMP X

Ad (IR) ⟶ PC

(2) 条件转移 BAN X （负则转）

$A_0 \cdot Ad(IR) + \bar{A}_0 (PC) \longrightarrow PC$

(结果为负即A_0=1)

9.1

4. 三类指令的指令周期

非访存 指令周期 | 取指周期 | 执行周期

直接访存 指令周期 | 取指周期 | 执行周期

间接访存 指令周期 | 取指周期 | 间址周期 | 执行周期

转移 指令周期 | 取指周期 | 执行周期

间接转移 指令周期 | 取指周期 | 间址周期 | 执行周期

四、中断周期

9.1

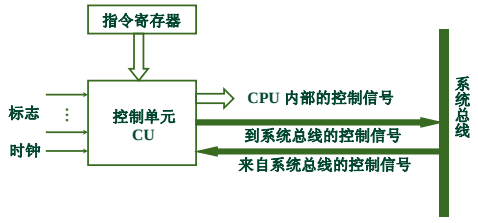
程序断点存入 “ 0 ” 地址	程序断点 进栈
0 ⟶ MAR	(SP) – 1 ⟶ MAR
1 ⟶ W	1 ⟶ W
PC ⟶ MDR	PC ⟶ MDR
MDR ⟶ M (MAR)	MDR ⟶ M (MAR)

中断识别程序入口地址 M ⟶ PC

0 ⟶ EINT（置“0”） 0 ⟶ EINT（置“0”）

9.2 控制单元的功能

一、控制单元的外特性



1. 输入信号

9.2

(1) 时钟

CU 受时钟控制

一个时钟脉冲

发一个操作命令或一组需同时执行的操作命令

(2) 指令寄存器 OP (IR) ⟶ CU

控制信号 与操作码有关

(3) 标志

CU 受标志控制

(4) 外来信号

如 INTR 中断请求

HRQ 总线请求

2. 输出信号 9.2

(1) CPU 内的各种控制信号

$R_i \longrightarrow R_j$

(PC) + 1 ⟶ PC

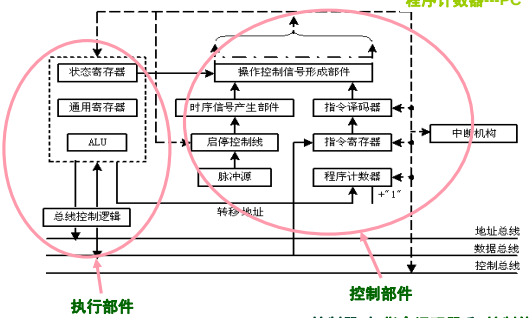
ALU　十、一、与、或 ……

(2) 送至控制总线的信号

$\overline{MREQ}$	访存控制信号
$\overline{IO/M}$	访 IO/存储器的控制信号
$\overline{RD}$	读命令
$\overline{WR}$	写命令
INTA	中断响应信号
HLDA	总线响应信号

CPU基本组成原理图

指令寄存器----IR
程序计数器---PC



执行部件

CPU 由 执行部件 和 控制部件组成

CPU 包含 数据通路 和 控制器

控制部件

控制器 由 指令译码器 和 控制信号形成部件 组成

- PC、IR
- 指令译码器ID：对操作进行译码，向控制器提供操作的特定信号。
- 时序发生器：产生各种时序信号
  - 脉冲源：通常由石英晶体振荡器构成。产生一定频率的脉冲信号作为整个机器的时钟脉冲，是机器周期和工作脉冲的基准信号。在机器刚加电时，产生总清信号(reset)
  - 启停线路：保证可靠地送出或封锁完整的时钟脉冲，控制时序信号的发生或停止，从而启动机器工作或停机。

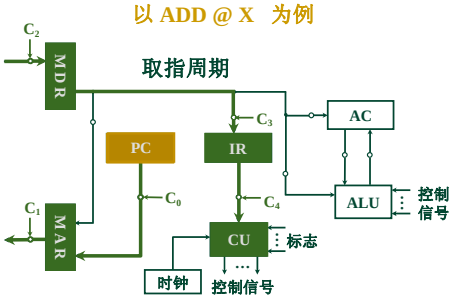
- 时序控制信号形成部件：当机器启动后，在CLK时钟作用下，根据当前正在执行的指令的需要，产生相应的时序控制信号，并根据被控功能部件的反馈信号调整时序控制信号。
- 状态寄存器（PSR）：存放PSW,PSW表明了系统基本状态，是控制程序执行的重要依据。

二、控制信号举例 9.2

1. 不采用 CPU 内部总线的方式

以 ADD @ X　为例

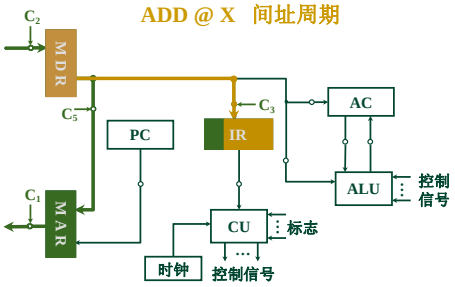
取指周期



二、控制信号举例 9.2

1. 不采用 CPU 内部总线的方式

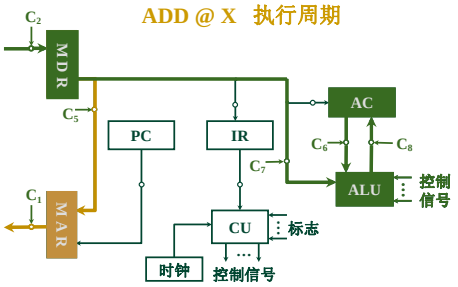
ADD @ X　间址周期



二、控制信号举例

1. 不采用 CPU 内部总线的方式

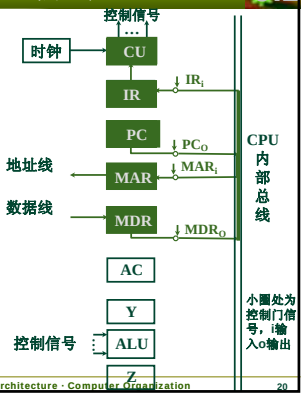
ADD @ X 执行周期



2. 采用 CPU 内部总线方式

(1) ADD @ X 取指周期

- PC⟶ MAR⟶ 地址线
  PC_O MAR_i
- CU 发读命令 1⟶R
- 数据线 ⟶ MDR
- MDR⟶ IR
  MDR_O IR_i
- OP（IR）⟶ CU
- （PC）+ 1⟶PC

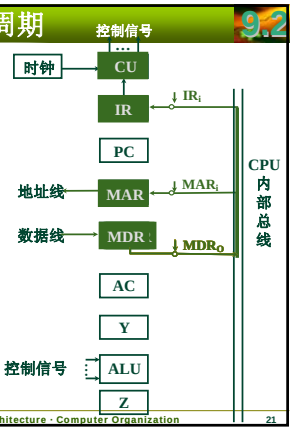


(2) ADD @ X 间址周期

形式地址 ⟶ MAR

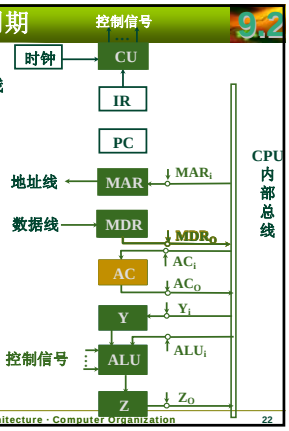
- MDR⟶MAR⟶ 地址线
  MDR_O MAR_i
- 1⟶ R
- 数据线 ⟶ MDR
- MDR⟶ IR
  MDR_O IR_i

有效地址 ⟶ Ad（IR）



(3) ADD @ X 执行周期

- MDR⟶ MAR ⟶ 地址线
  MDR_O MAR_i
- 1⟶ R
- 数据线 ⟶ MDR
- MDR⟶ Y ⟶ ALU
  MDR_O Y_i
- AC⟶ ALU
  AC_O ALU_i
- （AC）+（Y）⟶ Z
- Z⟶ AC
  Z_O AC_i

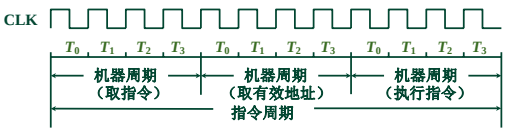


四、 CU的控制方式

产生不同微操作命令序列所用的时序控制方式

1. 同步控制方式

任一微操作均由 统一基准时标 的时序信号控制



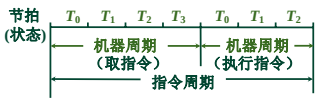
(1) 采用 定长 的机器周期

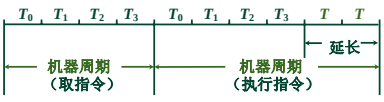
以 最长 的 微操作序列 和 最繁 的微操作作为 标准

机器周期内 节拍数相同

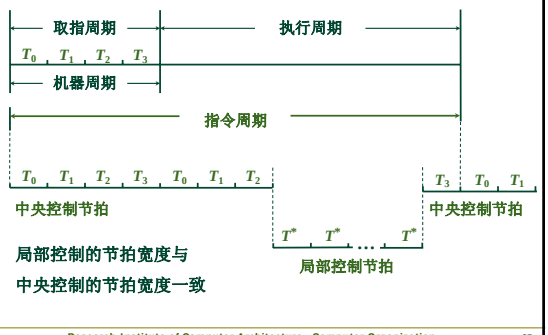
(2) 采用不定长的机器周期

机器周期内 节拍数不等





(3) 采用中央控制和局部控制相结合的方法



例. 设某机平均执行一条指令需要两次访问内存，平均需要3个CPU周期，每个CPU周期平均包含4个节拍周期。若机器主频为240MHz，问：
（1）若主存为“0等待”（即不需要插入等待周期），问执行一条指令的平均时间为多少？
（2）若每次访问内存需要插入2个等待周期，问执行一条指令的平均时间又是多少？

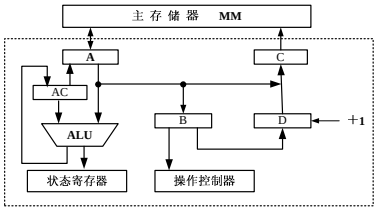
解：因为主频为240MHz，所以节拍周期=（1/240）μs每个
因为每个CPU周期平均包含4个节拍周期，所以：
CPU周期=节拍周期×4=4/240MHz=(1/60)μs
若访存不需要插入等待周期，则执行一条指令平均需要3个CPU周期，所以：
指令周期=3×CPU周期=3×（1/60）μs=(1/20)μs=0.05μs
机器平均速度=1/0.05μs=20 MIPS
（2）平均执行一条指令需要两次访问内存，每次访问内存需要插入2个等待周期，所以：
指令周期=0.05μs+2×(1/240)μs=(1/20)μs+(1/120)μs
=(7/120)μs
机器平均速度=120/7≈17MIPS

$$\frac{MIPS_1}{MIPS_2} = \frac{f_1}{f_2}$$

若某机主频为200MHZ，每个指令周期平均为2.5CPU周期，每个CPU周期平均包括2个主频周期，问：
(1)该机平均指令执行速度为多少MIPS?
(2)若主频不变，但每条指令平均包括5个CPU周期，每个CPU周期又包含4个主频周期，平均指令执行速度又为多少MIPS?
由此可得出什么结论?

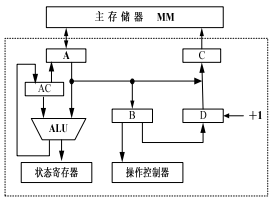
解：（1）主频为为200MHz，所以主频周期＝1/200MHz=0.005μs
每个指令周期平均为2.5CPU周期，每个CPU周期平均包括2个主频周期，所以一条指令的执行时间＝2.5×2×0.005μs＝0.025μs
该机平均指令执行速度＝1/0.025＝40MIPS。
（2）每条指令平均包括5个CPU周期，每个CPU周期又包含4个主频周期，所以一条指令的执行时间＝4×5×0.005μs＝0.1μs
该机平均指令执行速度＝1/0.1＝10MIPS
(3)说明指令的复杂程度会影响指令的平均执行速度。

例. CPU结构如图所示，其中包括一个累加寄存器AC、一个状态寄存器和其他四个寄存器，各部分之间的连线表示数据通路，箭头表示信息传送方向
（1）标明图中四个寄存器的名称



解：A为MDR，B为IR，C为MAR，D为PC

（2）简述取指令的数据通路
（3）简述完成指令LDA X的数据通路（X为内存地址，LDA的功能为（X）→(AC)）
（4）简述完成指令ADDY的数据通路（Y为内存地址，ADD功能为（AC）+（Y）→（AC））
（5）简述完成指令STA Z的数据通路（Z为内存地址，STA功能为（AC）→(Z)）



解：(2)取指：PC →MAR →MM →MDR →IR
(3) LDA X: X →MAR →MM →MDR →ALU →AC
(4) ADD Y: Y →MAR →MM →MDR →ALU →ADD →AC
(5) STA Z: Z →MAR , AC →MDR → MM

四、 CU的控制方式

2. 异步控制方式
无基准时标信号
无固定的周期节拍和严格的时钟同步
采用 应答方式

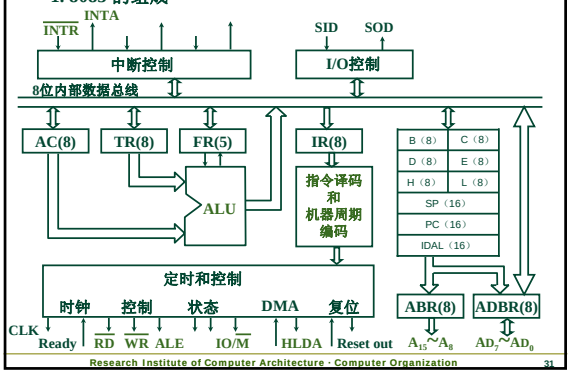
3. 联合控制方式
同步与异步相结合
大部分统一、小部分区别对待
如：取指同步、I/O异步

4. 人工控制方式
(1) Reset
(2) 连续 和 单条 指令执行转换开关
(3) 符合停机开关

五、多级时序系统实例分析

9.2

1. 8085 的组成



2. 8085 的外部引脚

9.2

(1) 地址和数据信号

A_{15}~A_8　AD_7~AD_0

SID　　SOD

(2) 定时和控制信号

入　X_1　X_2

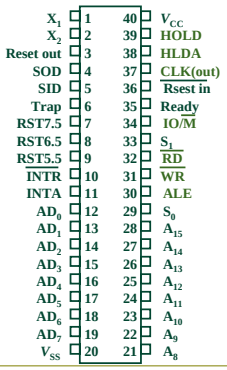
出　CLK　ALE　S_0　S_1

IO/$\overline{M}$　$\overline{RD}$　$\overline{WR}$

(3) 存储器和 I/O 初始化

入　HOLD　Ready

出　HLDA



(4) 与中断有关的信号

9.2

入　$\overline{INTR}$

出　INTA

Trap 重新启动中断

(5) CPU 初始化

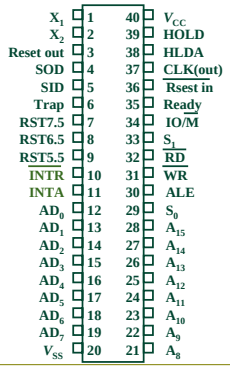
入　$\overline{Reset\ in}$

出　Reset out

(6) 电源和地

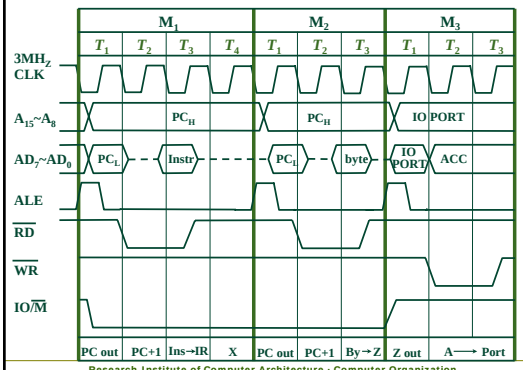
V_{CC}　+5 V

V_{SS}　地



3. 机器周期和节拍（状态）与控制信号的关系

9.2



小结

9.2

以一条输出指令（I/O 写）为例

机器周期 M_1　取指令操作码

机器周期 M_2　取设备地址

机器周期 M_3　执行 ACC 的内容写入设备

每个 控制 信号在 指定机器周期 的 指定节拍 *T* 时刻 发出

第10章　控制单元的设计

10.1　组合逻辑设计

10.2　微程序设计

控制器的类型

- 组合逻辑型：核心是微操作产生部件，用组合逻辑设计思想，以布尔代数为工具设计。输入信号来自指令译码器的输出、时序发生器的时序信号和程序运行结果特征及状态，输出为带有时间标志的微操作控制信号。
- 微程序控制型：将机器指令分解为基本微命令序列，用二进制码表示微命令，并编成微指令，多条微指令形成微程序。每种机器指令对应一段微程序，在制造CPU时固化在CPU中的一个控制存储器中（CS或CM）中。执行一条机器指令时，CPU依次从CS中取微指令，从而产生微命令。

基本概念

- 微命令：微程序控制计算机中的微操作控制信号。
- 微操作：控制器中执行部件接受微指令后所进行的操作，是指令序列中最基本、不可分割的动作。
  - 所有的微操作要在一个机器周期完成
  - 例如，一条加法指令要分成四步完成：取指令，计算地址，取数，加法运算，每一步要实现若干个微操作
- 微指令：在微程序控制的计算机中，同时发出的控制信号所执行的一组微操作称为微指令。
  - 是在机器的一个节拍中，一组实现一定操作功能的微命令
  - 将一条指令分解成若干条微指令，按次序执行微指令，即可实现指令的功能
- 微程序：由微指令组成的序列称为微程序。一个微程序的功能对应一条机器指令的功能。

基本概念

- 控制存储器：微程序存放于存储器中，由于该存储器主要存放控制命令（信号）和下一条执行的微指令地址（简称下址），因此被称为控制存储器。
  - 执行一条指令就是执行一段存放在控制存储器中的微程序。
  - 用ROM实现，因为一台计算机指令系统是固定的，所以微程序是固定的，所以控制存储器可以用ROM实现
- 微周期：执行一条微指令和取出下一条微指令所需时间
  - 通常一个微周期与一个CPU周期时间相等
- 相斥性微命令：不能在一个微周期出现的微命令
  - 如读命令和写命令
- 相容性微命令：能在一个微周期出现的微命令。

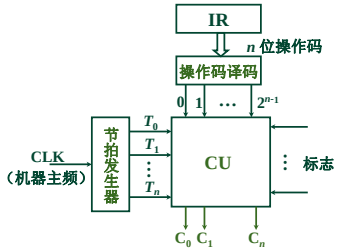
基本概念

- 微地址寄存器（CMAR,或μAR）用于存放控制存储器的读/写微指令地址。
- 微指令寄存器（ CMDR，或μIR）用于存放从控制存储器中读出的微指令

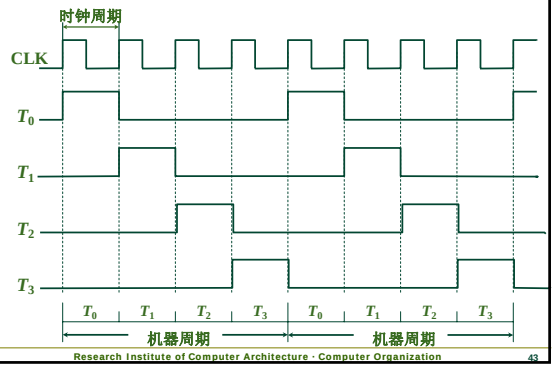
10.1　组合逻辑设计

一、组合逻辑控制单元框图

1. CU 外特性



2. 节拍信号

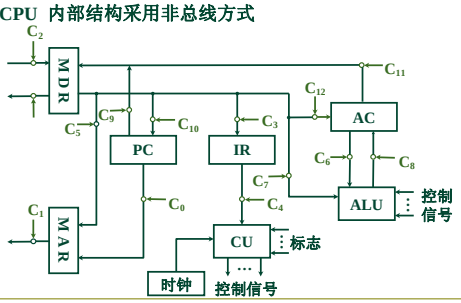


二、微操作的节拍安排

假设：采用 同步控制方式

一个 机器周期 内有 3个节拍（时钟周期）

CPU 内部结构采用非总线方式



1. 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

2. 取指周期 微操作的 节拍安排

T_0	PC ⟶ MAR	原则二
	1 ⟶ R	
T_1	M (MAR) ⟶ MDR	原则二
	(PC) + 1 ⟶ PC	
T_2	MDR ⟶ IR	原则三
	OP (IR) ⟶ ID	

3. 间址周期 微操作的 节拍安排

T_0	Ad (IR) ⟶ MAR
	1 ⟶ R
T_1	M (MAR) ⟶ MDR
T_2	MDR ⟶ Ad (IR)

4. 执行周期 微操作的 节拍安排

① CLA	T_0	
	T_1	
	T_2	0 ⟶ AC
② COM	T_0	
	T_1	
	T_2	$\overline{AC}$ ⟶ AC
③ SHR	T_0	
	T_1	
	T_2	L (AC) ⟶ R (AC)
		AC_0 ⟶ AC_0

④ CSL	T_0		
	T_1		
	T_2	R (AC) ⟶ L (AC)	AC_0 ⟶ AC_n
⑤ STP	T_0		
	T_1		
	T_2	0 ⟶ G	
⑥ ADD X	T_0	Ad (IR) ⟶ MAR	1 ⟶ R
	T_1	M (MAR) ⟶ MDR	
	T_2	(AC) + (MDR) ⟶ AC	
⑦ STA X	T_0	Ad (IR) ⟶ MAR	1 ⟶ W
	T_1	AC ⟶ MDR	
	T_2	MDR ⟶ M (MAR)	

⑧ LDA X

T_0 Ad (IR) ⟶ MAR　1 ⟶ R

T_1 M (MAR) ⟶ MDR

T_2 MDR ⟶ AC

⑨ JMP X

T_0

T_1

T_2 Ad (IR) ⟶ PC

⑩ BAN X

T_0

T_1

T_2 $A_0 \cdot Ad(IR) + \overline{A}_0 \cdot PC \longrightarrow PC$

5. 中断周期 微操作的 节拍安排

T_0 0 ⟶ MAR　　1 ⟶ W　硬件关中断

T_1 PC ⟶ MDR

T_2 MDR ⟶ M (MAR)　向量地址 ⟶ PC

中断隐指令完成

三、组合逻辑设计步骤

- 设计指令的操作码，确定指令长度是固定的还是变长的。
- 确定机器周期、节拍和时钟周期，确定机器周期是固定的还是可变长的。
- 根据指令功能和CPU的结构图，绘制每条指令的微操作流程图并综合成一个总的流程图。
- 给微操作流程图安排时序，确定每条指令所需的机器周期及在各机器周期需完成的操作，排出微操作时间表。
- 根据操作时间表写出微操作的逻辑表达式，即
  微操作＝周期·节拍·时钟脉冲·指令码·其他条件
- 根据微操作的表达式，画出组合逻辑电路

三、组合逻辑设计步骤

1. 列出操作时间表

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
FE 取指	T_0		PC⟶ MAR						
			1⟶ R						
	T_1		M(MAR) ⟶MDR						
			(PC) +1 ⟶PC						
	T_2		MDR⟶ IR						
			OP(IR) ⟶ID						
		I	1⟶ IND						
		$\overline{I}$	1⟶ EX						

间址特征

三、组合逻辑设计步骤

1. 列出操作时间表

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
IND 间址	T_0		Ad (IR)⟶MAR						
			1⟶ R						
	T_1		M(MAR) ⟶MDR						
	T_2		MDR⟶ Ad (IR)						
		$\overline{IND}$	1⟶ EX						

间址周期标志

三、组合逻辑设计步骤

1. 列出操作时间表

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
EX 执行	T_0		Ad (IR)⟶MAR						
			1 ⟶ R						
			1 ⟶ W						
	T_1		M(MAR) ⟶MDR						
			AC⟶ MDR						
	T_2		(AC)+(MDR) →AC						
			MDR⟶ M(MAR)						
			MDR ⟶AC						
			0⟶ AC						

三、组合逻辑设计步骤 10.1

1. 列出操作时间表

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
FE 取指	T_0		PC⟶ MAR	1	1	1	1	1	1
			1⟶ R	1	1	1	1	1	1
	T_1		M(MAR) ⟶MDR	1	1	1	1	1	1
			(PC) +1⟶ PC	1	1	1	1	1	1
	T_2		MDR⟶ IR	1	1	1	1	1	1
			OP(IR)⟶ ID	1	1	1	1	1	1
		I	1⟶ IND			1	1	1	1
		$\bar{I}$	1⟶ EX	1	1	1	1	1	1

三、组合逻辑设计步骤 10.1

1. 列出操作时间表

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
IND 间址	T_0		Ad (IR)⟶MAR			1	1	1	1
			1⟶ R			1	1	1	1
	T_1		M(MAR) ⟶MDR			1	1	1	1
	T_2		MDR⟶ Ad (IR)			1	1	1	1
		$\overline{IND}$	1⟶ EX			1	1	1	1

三、组合逻辑设计步骤 10.1

1. 列出操作时间表

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
EX 执行	T_0		Ad (IR)⟶MAR			1	1	1	
			1⟶ R			1		1	
			1 ⟶ W				1		
	T_1		M(MAR) ⟶MDR			1		1	
			AC⟶ MDR				1		
	T_2		(AC)+(MDR) →AC			1			
			MDR⟶ M(MAR)				1		
			MDR⟶AC					1	
			0⟶ AC	1					

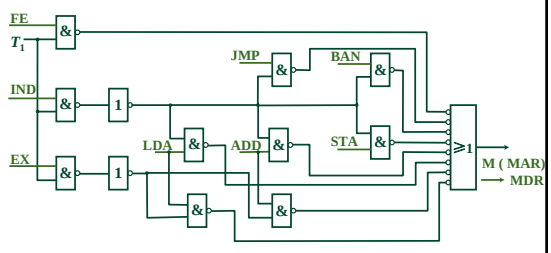
2. 写出微操作命令的最简表达式 10.1

$$M(MAR) \longrightarrow MDR$$

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

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

3. 画出逻辑图 10.1



特点
- 思路清晰，简单明了
- 庞杂，调试困难，修改困难
- 速度快 （RISC）

10.2 微程序设计

一、微程序设计思想的产生

1951 英国剑桥大学教授 Wilkes

完成一条机器指令：
- 微操作命令 1 — 微指令 1 10100000
- 微操作命令 2
- ⋮ — ⋮
- 微操作命令 n — 微指令 n 00010010

微指令 1 … 微指令 n 组成微程序

一条机器指令对应一个微程序

存入 ROM

存储逻辑

微指令的格式

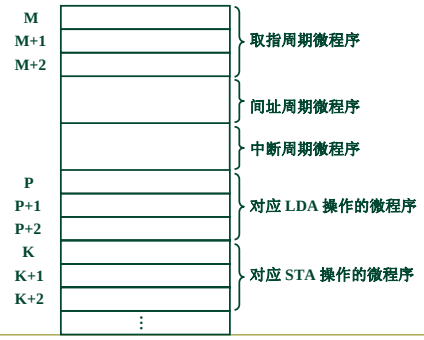
微指令：

控制字段	下址字段

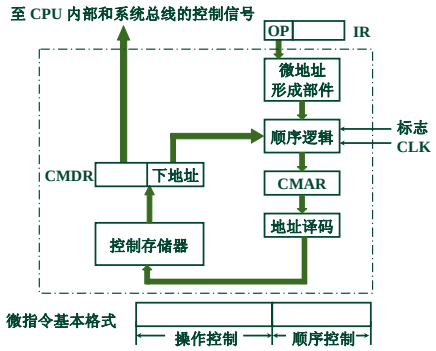
- 控制字段：操作控制，发出各种控制信号
- 下址字段：顺序控制，指出下调微指令地址，以控制微指令序列的执行顺序

二、微程序控制单元框图及工作原理 10.2

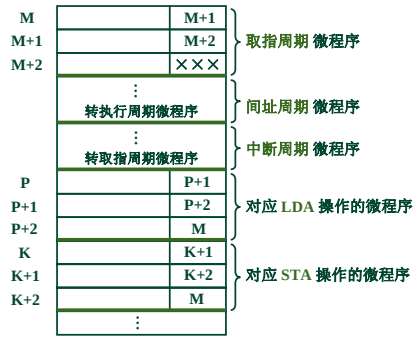
1. 机器指令对应的微程序



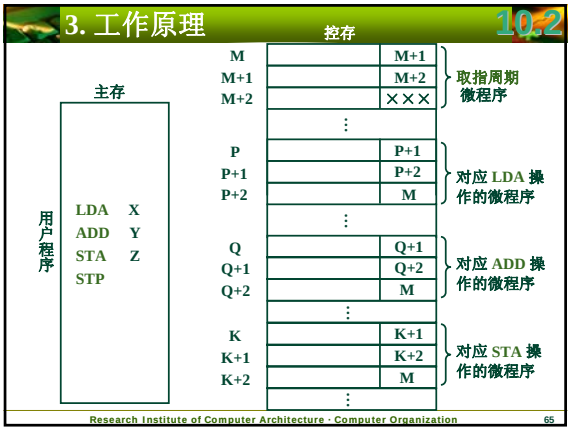
2. 微程序控制单元的基本框图 10.2



二、微程序控制单元框图及工作原理 10.2



3. 工作原理 10.2



3. 工作原理 10.2

(1) 取指阶段　　执行取指微程序

M ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

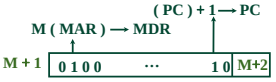
形成下条微指令地址　M + 1

Ad (CMDR) ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

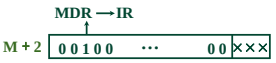
形成下条微指令地址　M + 2



Ad (CMDR) ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令



(2) 执行阶段　执行 LDA 微程序

10.2

OP (IR) ⟶ 微地址形成部件 ⟶ CMAR　(P ⟶ CMAR)

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

Ad (IR) ⟶ MAR　　1 ⟶ R

P | 0 0 0 1 … 0 0 1 | P+1

形成下条微指令地址 CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

M (MAR) ⟶ MDR

P + 1 | 0 1 0 0 … 0 | P+2

形成下条微指令地址 CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

MDR ⟶ AC

P + 2 | 0 0 0 0 0 0 1 … 0 | M

形成下条微指令地址 CMAR　(M ⟶ CMAR)

(3) 取指阶段　执行取指微程序

10.2

M ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

PC ⟶ MAR　　1 ⟶ R

M | 1 0 0 … 0 0 1 | M+1

⋮

全部微指令存在 CM 中，程序执行过程中 只需读出

关键

- 微指令的 操作控制字段如何形成微操作命令
- 微指令的 后续地址如何形成

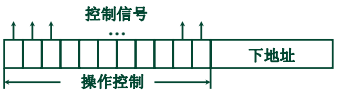
三、微指令的编码方式（控制方式）

10.2

1. 直接编码（直接控制）方式

在微指令的操作控制字段中，

每一位代表一个微操作命令



速度最快

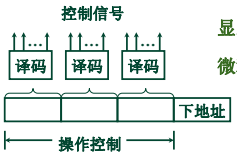
某位为 “1” 表示该控制信号有效

2. 字段直接编码方式

10.2

将微指令的控制字段分成若干 “段”，

每段经译码后发出控制信号



显式编码

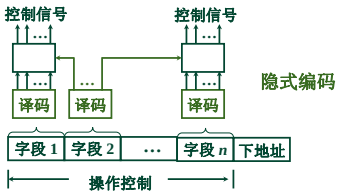
微程序执行速度较慢

每个字段中的命令是 互斥 的

缩短 了微指令 字长，增加 了译码 时间

10.2

3. 字段间接编码方式



隐式编码

4. 混合编码

直接编码和字段编码（直接和间接）混合使用

5. 其他

四、微指令序列地址的形成

10.2

1. 微指令的 下地址字段 指出(断定方式)

2. 根据机器指令的 操作码 形成：根据机器指令的操作码，由微地址形成部件形成对应该机器指令微程序的首地址。

3. 增量计数器（顺序地址）

(CMAR) + 1 ⟶ CMAR

4. 分支转移（转移指令）

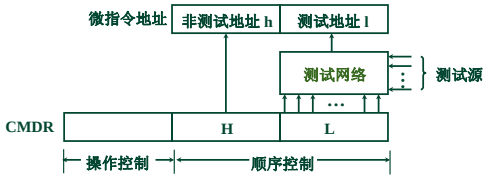
操作控制字段	转移方式	转移地址

转移方式　指明判别条件

转移地址　指明转移成功后的去向

四、微指令序列地址的形成 10.2

5. 通过测试网络



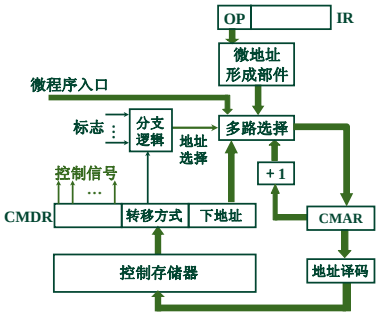
6. 由硬件产生微程序入口地址

加电后，第一条微指令地址　由专门 硬件 产生

中断周期　由 硬件 产生 中断周期微程序首地址

四、微指令序列地址的形成 10.2

7. 后续微指令地址形成方式原理图



五、微指令格式 10.2

1. 水平型微指令

一次能定义并执行多个并行操作

如　直接编码、字段直接编码、字段间接编码、
直接和字段混合编码

2. 垂直型微指令

类似机器指令操作码 的方式

由微操作码字段规定微指令的功能

3. 两种微指令格式的比较 10.2

(1) 水平型微指令比垂直型微指令 并行操作能力强，灵活性强

(2) 水平型微指令执行一条机器指令所要的微指令 数目少，速度快

(3) 水平型微指令 用较短的微程序结构换取较长的微指令结构

(4) 水平型微指令与机器指令 差别大

10.2

六、静态微程序设计和动态微程序设计

静态　微程序无须改变，采用 ROM

动态　通过 改变微指令 和 微程序 改变机器指令，
有利于仿真，采用 EPROM

七、毫微程序设计

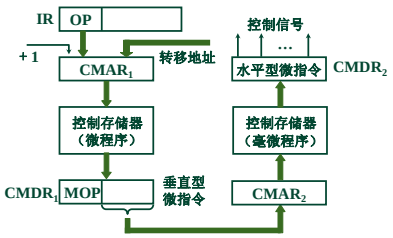
1. 毫微程序设计的基本概念

微程序设计　用 微程序解释机器指令

毫微程序设计　用 毫微程序解释微程序

毫微指令与微指令 的关系好比 微指令与机器指令 的关系

2. 毫微程序控制存储器的基本组成 10.2



八、串行微程序控制和并行微程序控制 10.2

串行 微程序控制

取第 i 条微指令	执行第 i 条微指令	取第 i+1 条微指令	执行第 i+1 条微指令

并行 微程序控制

取第 i 条微指令	执行第 i 条微指令		
	取第 i+1 条微指令	执行第 i+1 条微指令	
		取第 i+2 条微指令	执行第 i+2 条微指令

九、微程序设计举例 10.2

1. 写出对应机器指令的微操作及节拍安排

假设 CPU 结构与组合逻辑相同

(1) 取指阶段微操作分析　　3 条微指令

T_0　PC ⟶ MAR　　1 ⟶ R

T_1　M (MAR) ⟶ MDR　　(PC) + 1 ⟶ PC

T_2　MDR ⟶ IR　　OP (IR) ⟶ 微地址形成部件

若需考虑如何安排这条微指令？

则取指操作需 3 条微指令

Ad (CMDR) ⟶ CMAR

OP (IR) ⟶ 微地址形成部件 ⟶ CMAR

(2) 取指阶段的微操作及节拍安排 10.2

考虑到需要 形成后续微指令的地址

T_0　PC ⟶ MAR　　1 ⟶ R

T_1　Ad (CMDR) ⟶ CMAR

T_2　M (MAR) ⟶ MDR　(PC)+1 ⟶ PC

T_3　Ad (CMDR) ⟶ CMAR

T_4　MDR ⟶ IR　　OP (IR) ⟶ 微地址形成部件

T_5　OP (IR) ⟶ 微地址形成部件 ⟶ CMAR

(3) 执行阶段的微操作及节拍安排 10.2

考虑到需形成后续微指令的地址

取指微程序的入口地址 M
由微指令下地址字段指出

- 非访存指令

① CLA 指令

T_0　0 ⟶ AC

T_1　Ad (CMDR) ⟶ CMAR

② COM 指令

T_0　$\overline{AC}$ ⟶ AC

T_1　Ad (CMDR) ⟶ CMAR

10.2

③ SHR 指令

T_0　L (AC) ⟶ R (AC)　AC_0 ⟶ AC_0

T_1　Ad (CMDR) ⟶ CMAR

④ CSL 指令

T_0　R (AC) ⟶ L (AC)　AC_0 ⟶ AC_n

T_1　Ad (CMDR) ⟶ CMAR

⑤ STP 指令

T_0　0 ⟶ G

T_1　Ad (CMDR) ⟶ CMAR

• 访存指令 10.2

⑥ ADD 指令

T_0　Ad (IR) ⟶ MAR　　1 ⟶ R

T_1　Ad (CMDR) ⟶ CMAR

T_2　M (MAR) ⟶ MDR

T_3　Ad (CMDR) ⟶ CMAR

T_4　(AC) + (MDR) ⟶ AC

T_5　Ad (CMDR) ⟶ CMAR

⑦ STA 指令

T_0　Ad (IR) ⟶ MAR　　1 ⟶ W

T_1　Ad (CMDR) ⟶ CMAR

T_2　AC ⟶ MDR

T_3　Ad (CMDR) ⟶ CMAR

T_4　MDR ⟶ M (MAR)

T_5　Ad (CMDR) ⟶ CMAR

⑧ LDA 指令

T_0 Ad (IR) ⟶ MAR　　1 ⟶ R

T_1 Ad (CMDR) ⟶ CMAR

T_2 M (MAR) ⟶ MDR

T_3 Ad (CMDR) ⟶ CMAR

T_4 MDR ⟶ AC

T_5 Ad (CMDR) ⟶ CMAR

• 转移类指令

⑨ JMP 指令

T_0 Ad (IR) ⟶ PC

T_1 Ad (CMDR) ⟶ CMAR

⑩ BAN 指令

T_0 $A_0 \cdot Ad(IR) + \overline{A}_0 \cdot (PC)$ ⟶ PC

T_1 Ad (CMDR) ⟶ CMAR

全部微操作　20个

微指令　38条

2. 确定微指令格式

(1) 微指令的编码方式

采用直接控制

(2) 后续微指令的地址形成方式

由机器指令的操作码通过微地址形成部件形成

由微指令的下地址字段直接给出

(3) 微指令字长

由 20 个微操作

确定 操作控制字段　最少 20 位

由 38 条微指令

确定微指令的 下地址字段 为 6 位

微指令字长 可取 20 + 6 = 26 位

(4) 微指令字长的确定

38 条微指令中有 19 条

是关于后续微指令地址 ⟶ CMAR

其中 { 1 条　OP (IR) ⟶ 微地址形成部件 ⟶ CMAR
　　 { 18 条　Ad (CMDR) ⟶ CMAR

若用 Ad (CMDR) 直接送控存地址线

则 省去了输至 CMAR 的时间，省去了 CMAR

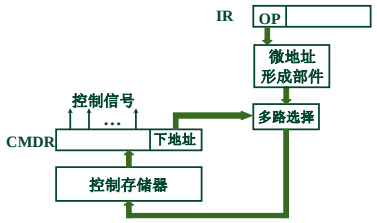
同理　OP (IR) ⟶ 微地址形成部件 ⟶ 控存地址线

可省去 19 条微指令，2 个微操作

38 − 19 = 19　　20 − 2 = 18

下地址字段最少取 5 位　　操作控制字段最少取 18 位

(5) 省去了 CMAR 的控制存储器



考虑留有一定的余量　取操作控制字段　18 位 ⟶ 24 位 } 共 30 位
　　下地址字段　5 位 ⟶ 6 位 }

(6) 定义微指令操作控制字段每一位的微操作

3. 编写微指令码点

微程序名称	微指令地址(八进制)	微指令（二进制代码） 操作控制字段 0	1	2	3	4	…	10	…	23	下地址字段 24	25	26	27	28	29
取指	00	1	1								0	0	0	0	0	1
	01			1	1						0	0	0	0	1	0
	02					1					×	×	×	×	×	×
CLA	03										0	0	0	0	0	0
COM	04										0	0	0	0	0	0
ADD	10		1								0	0	1	0	0	1
	11			1							0	0	1	0	1	0
	12										0	0	0	0	0	0
LDA	16		1					1			0	0	1	1	1	1
	17			1							0	1	0	0	0	0
	20										0	0	0	0	0	0

PC⟶MAR　PC+1⟶PC　MDR⟶IR　Ad(IR)⟶MAR　1⟶R　M(MAR)⟶MDR　1⟶R　Ad(IR)⟶MAR　M(MAR)⟶MDR


Computer
Organization
Thank You !
Computer Architecture Research Institute · Computer Organization