CISC's CPU控制器设计

中科大11系
李曦

内容

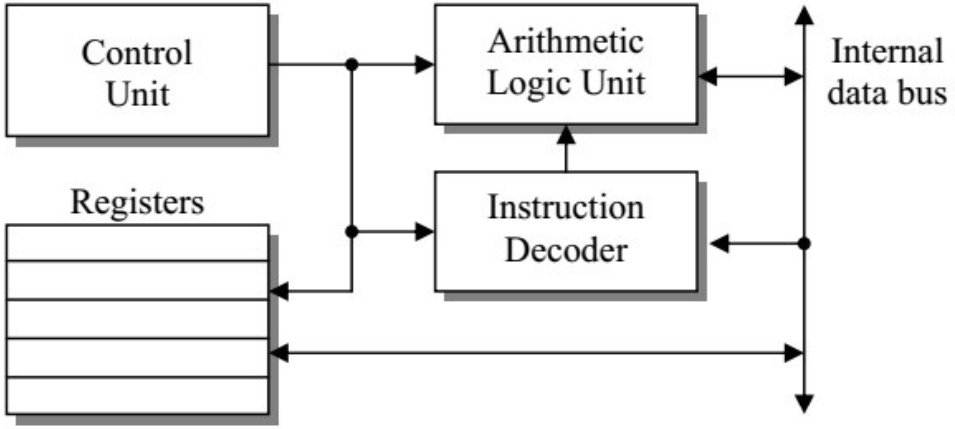
- **CISC's CPU**控制器设计：**A**模型
  - 多周期技术
    - 组合逻辑实现
    - 微程序控制

微程序控制Microprogrammed Control的概念和原理最早由英国剑桥大学的Maurice. V. Wilkes教授于**1951**年提出。1964，IBM System/360采用了此技术。

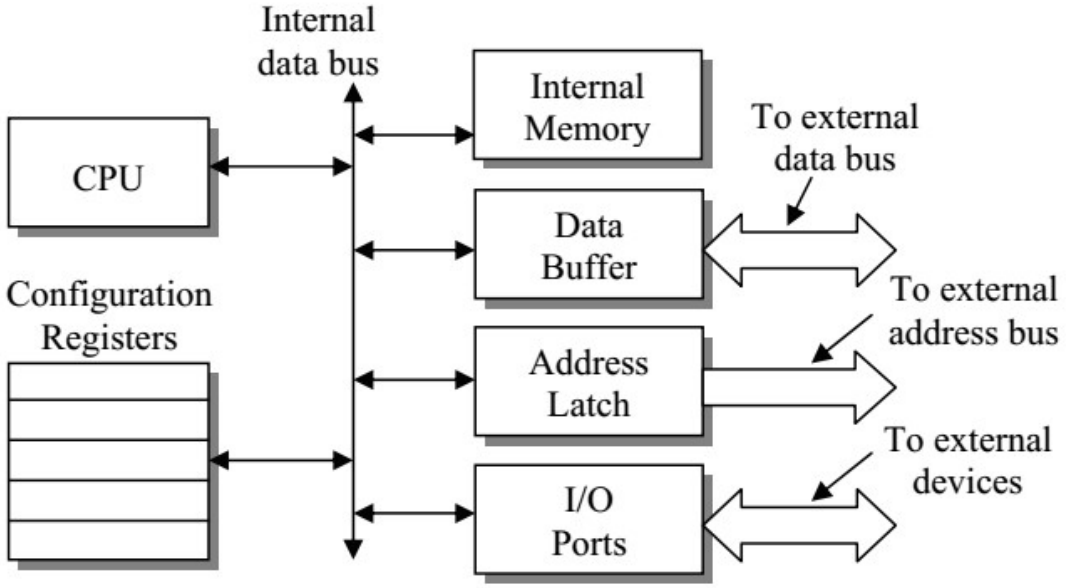
Wilkes教授：第二届图灵奖（1967）
根据冯·诺伊曼的EDVAC机设计方案，1949年在剑桥实现第一台存储程序式计算机EDSAC

functional requirements for a processor

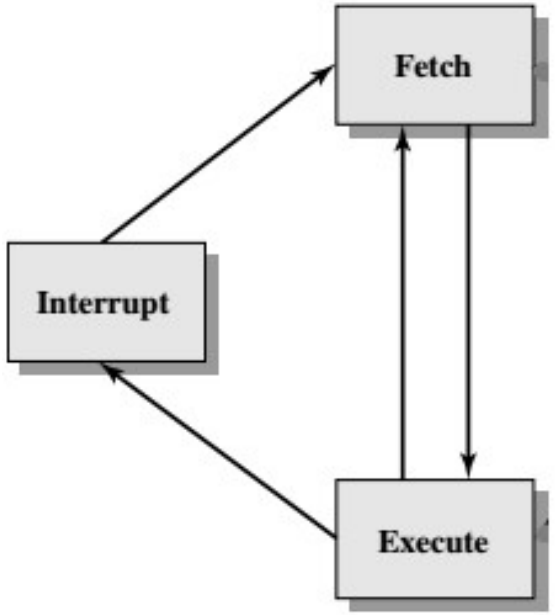
- Operations (opcodes)
- Addressing modes
- Registers

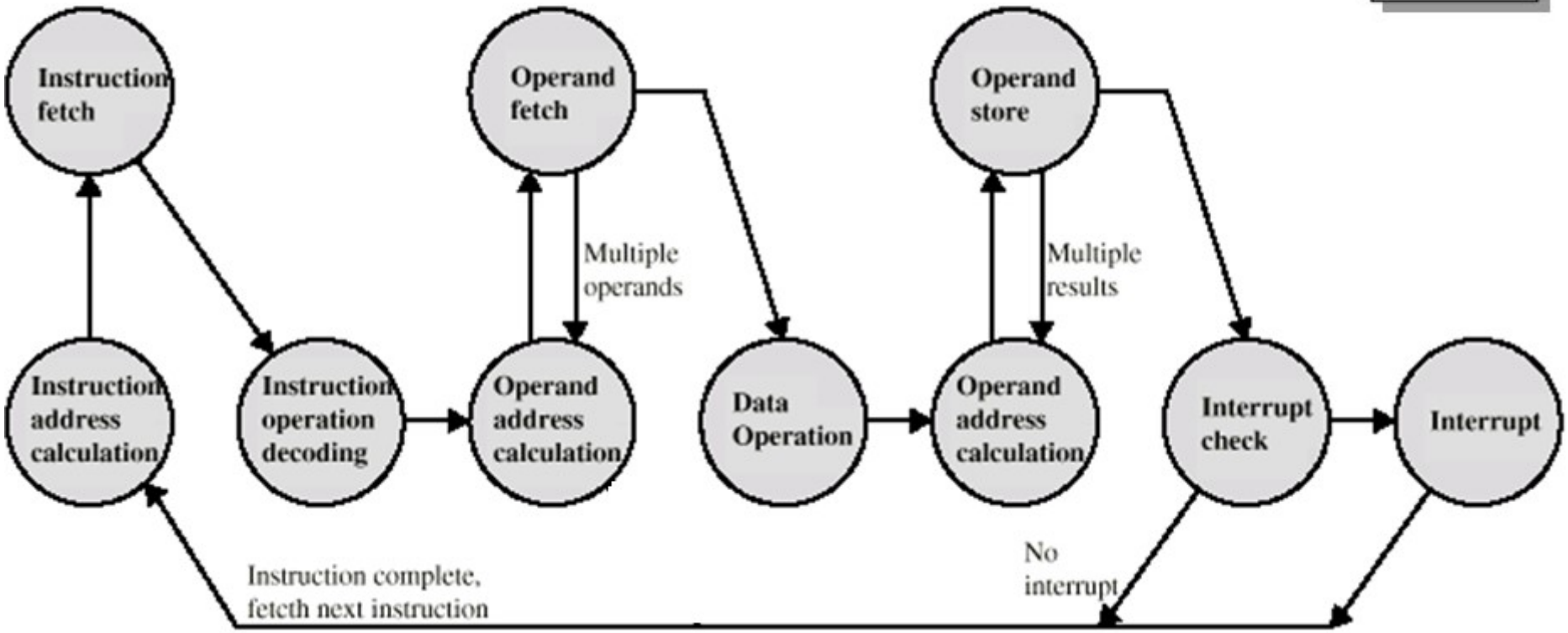


- Memory module interface
- I/O module interface
- Interrupts

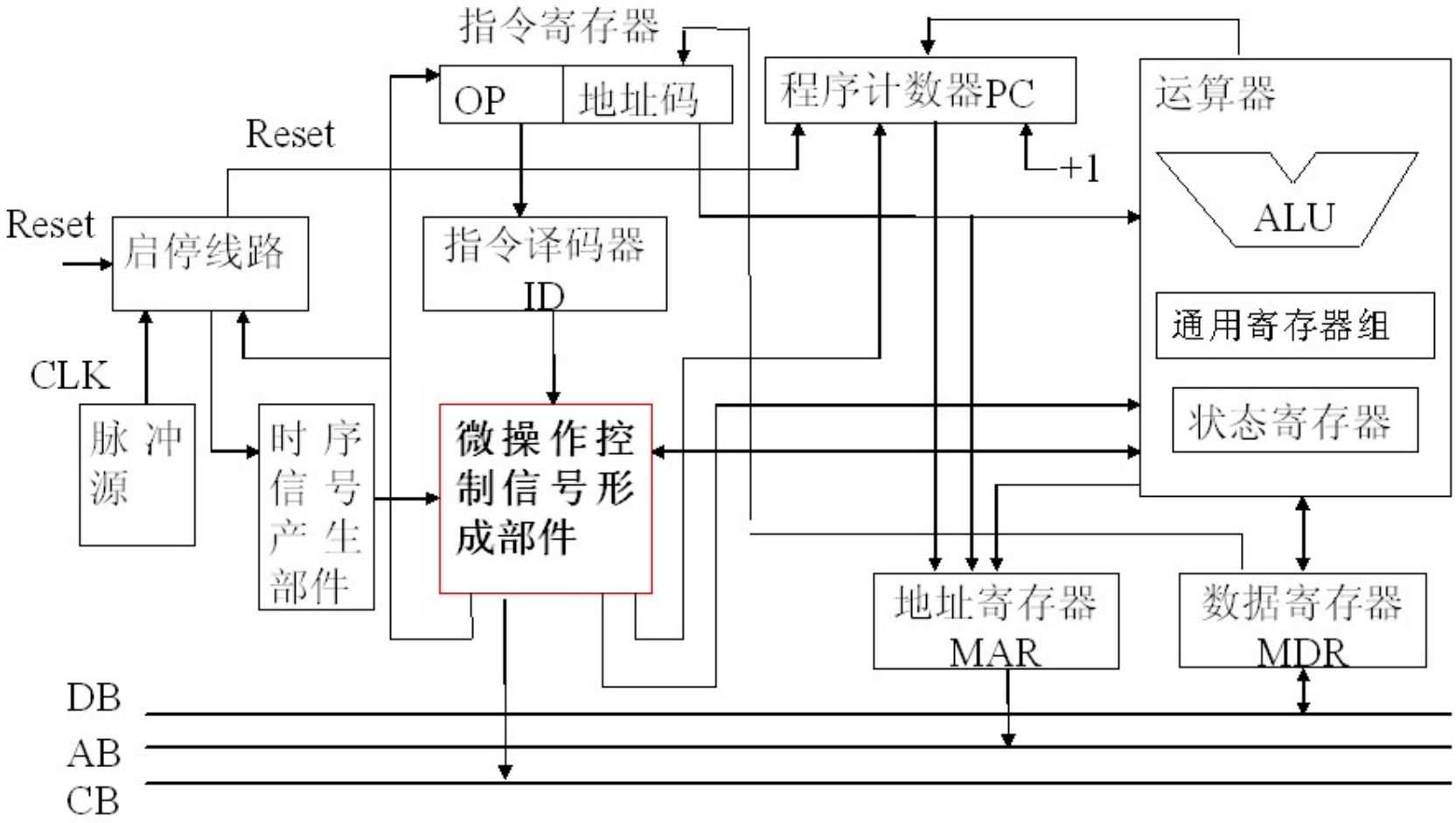


Instruction Cycle State Diagram



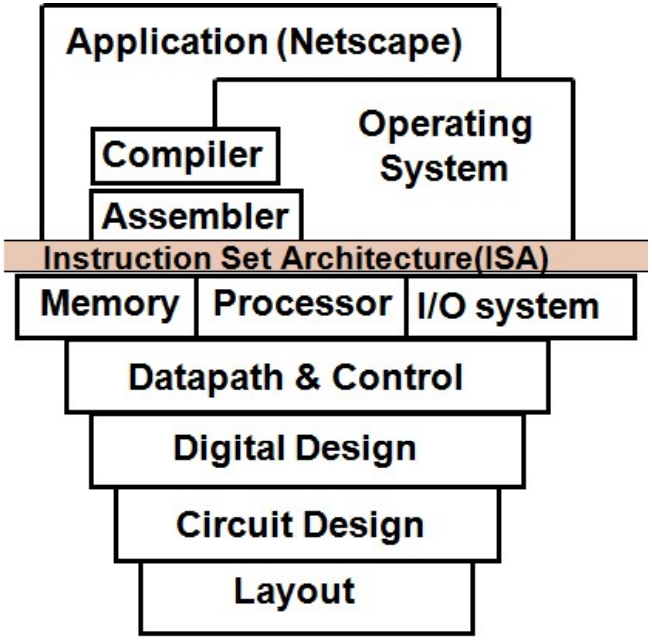


控制器设计


指令寄存器
OP
地址码
程序计数器PC
运算器
Reset
+1
ALU
Reset
启停线路
指令译码器
ID
通用寄存器组
CLK
脉冲源
时序信号产生部件
微操作控制信号形成部件
状态寄存器
地址寄存器
MAR
数据寄存器
MDR
DB
AB
CB

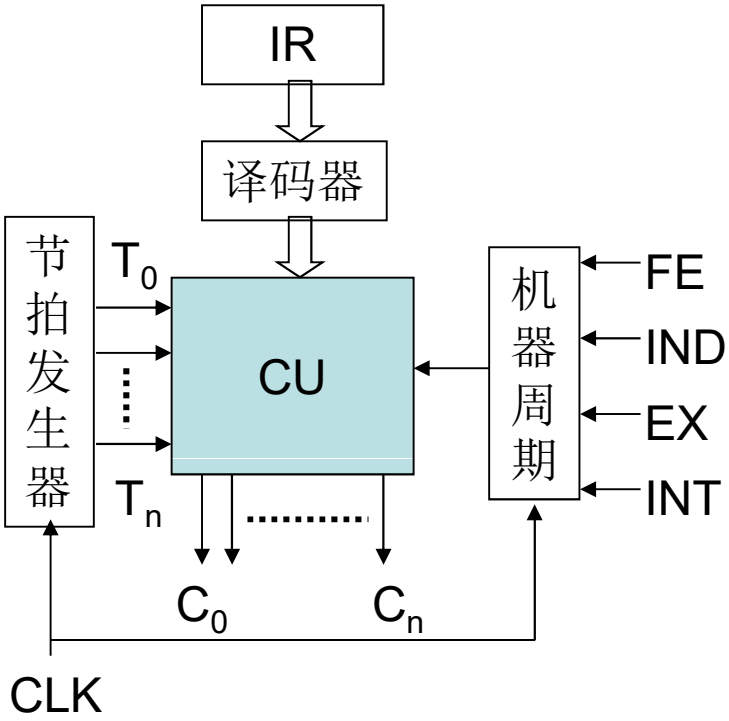
控制器设计思路

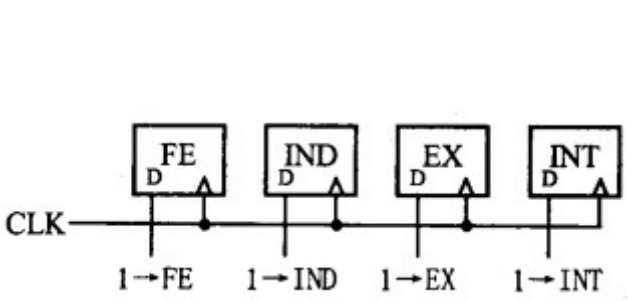
- 目标
  - 生成各个节拍所需的控制信号
- 步骤
  - 确定ISA
  - 确定处理器的微结构
    - 一个ISA可以有多种实现
  - 确定时序体系
    - 采用多级时序系统
  - 分析每条指令在各个机器周期的微操作
  - 将每条指令的各个微操作分配到各个节拍
  - 逻辑综合
    - 将所有指令在各个节拍的微操作进行综合，生成各个节拍所需的控制信号，据此产生控制逻辑

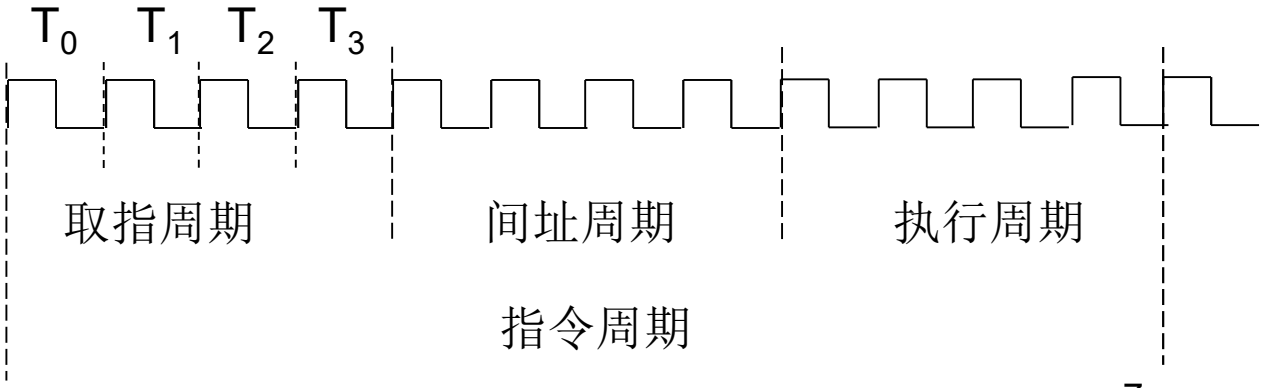


控制部件设计

- 任务：根据各个指令执行时在不同节拍的微操作要求，生成各个控制信号
- 指令译码
- 节拍发生器：在时钟控制下，使**CU**按规定的节拍发出控制信号
  - 节拍宽度：按照微操作的要求，满足信息沿数据通路从源寄存器传送到目的寄存器所需的时间
- 控制部件实现方式
  - 组合逻辑电路
  - 微程序设计





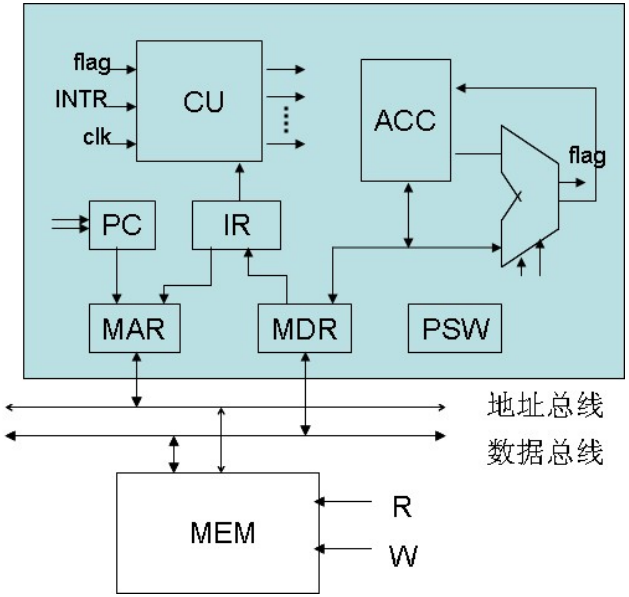


微操作的节拍安排原则

- 顺序性：遵守微操作的先后次序
  - 如先送地址，再发读写令
- 并行性：为了性能考虑，可以并行的微操作尽量安排在一个节拍内
  - 条件：**no race**
    - 针对不同对象的微操作
    - 无总线冲突
- 如果有些微操作占用的时间不长，尽量安排在一个节拍内完成
- **A**模型采用不定长指令周期、多机器周期、定长机器周期模式的实现模式

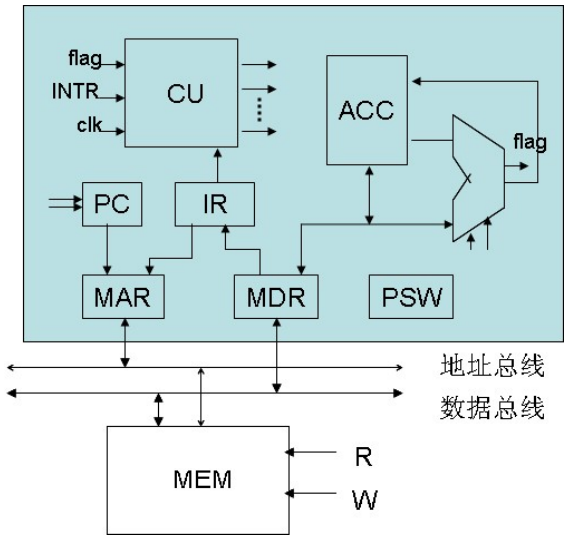
取指周期、间址周期

- 取指周期
  - **T_0: PC->MAR, 1->R**
  - **T_1: M(MAR)->MDR, PC+1->PC**
  - **T_2: MDR->IR, OP(IR)->ID**
- 间址周期
  - **T_0: Ad(IR)->MAR, 1->R**
  - **T_1: M(MAR)->MDR**
  - **T_2: MDR->Ad(IR)**



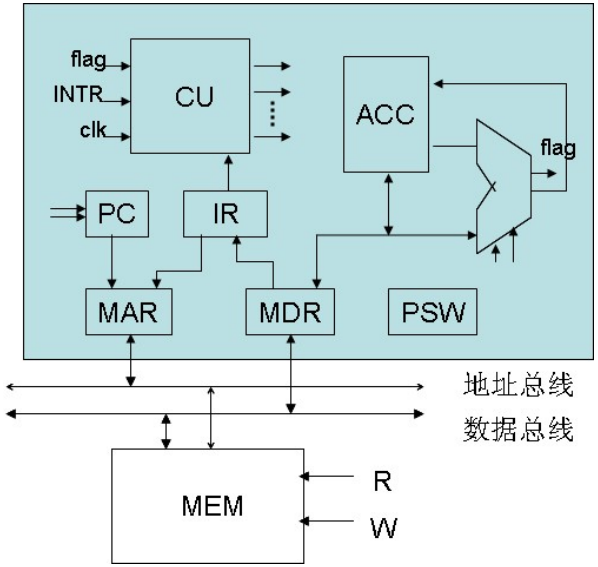
执行周期-非访存指令

- **CLA: 清ACC**
  - **T_0:**
  - **T_1:**
  - **T_2: 0->ACC**
    - 可以在任一节拍
- **COM: ACC取反**
  - **T_2: /ACC->ACC**
- **SHR: ACC算术右移一位, 符号位不变**
  - **T_2: L(ACC)->R(ACC), ACC_0->ACC_0**
- **CSL: ACC循环左移一位**
  - **T_2: R(ACC)->L(ACC), ACC_0->ACC_n**
- **STP：停机，标志触发器G置“0”**
  - **T_2: 0->G**

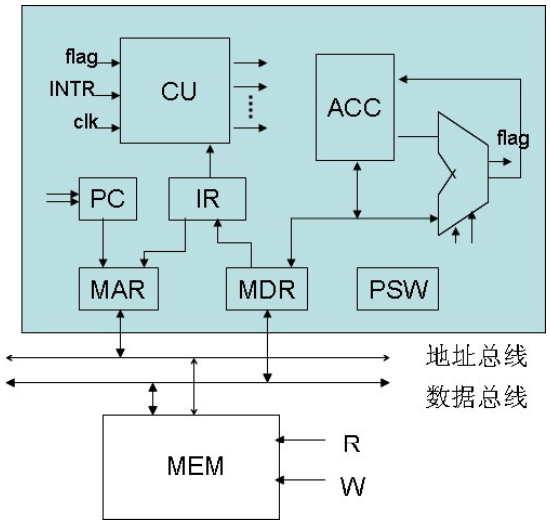


执行周期——访存指令

- **ADD X: ACC与地址X的内容相加，结果送ACC**
  - **T_0: Ad(IR)->MAR，1->R**
  - **T_1: M(MAR)->MDR**
  - **T_2: ACC+MDR->ACC**
    - **含ACC->ALU, MDR->ALU, ALU->ACC等**
- **STA X: 将ACC存入X中**
  - **T_0: Ad(IR)->MAR，1->W**
  - **T_1: ACC->MDR**
  - **T_2: MDR->M(MAR)**
- **LDA X: 将X中的内容读入ACC**
  - **T_0: Ad(IR)->MAR，1->R**
  - **T_1: M(MAR) ->MDR**
  - **T_2: MDR ->ACC**



执行周期——转移指令



- **JMP X:** 无条件转移至**X**
  - **T_0:**
  - **T_1:**
  - **T_2: Ad(IR)->PC**
- **BAN X:**
  - 如果前一条指令的执行结果为负（即**ACC**最高位为“**1**”）则转移至**X**
  - **T_0:**
  - **T_1:**
  - **T_2: ACC_0*Ad(IR)+/ACC_0* (PC)->PC**

组合逻辑实现Hardwired implementation

- 列出“微操作—节拍”总表
  - 因为每个微操作对应一个执行路径，也即表示了所需的控制信号（称为“命令”），因此该表表示的是各个节拍所需的控制信号。
- 写出生成每个微操作的逻辑表达式
  - 带时间约束关系
- 根据每个逻辑表达式生成对应的组合逻辑控制电路

示例

- 表10.1
  - I：指示间址否，由译码得到
    - 注意：支持直接寻址和间址寻址模式
  - IND：指示是否多级间址
  - INT周期的考虑
  - 注意：针对的是微操作而不是控制信号
- M(MAR)->MDR的逻辑表达式
  - M(MAR)->MDR = FE*T1 + IND*T1(ADD+STA+LDA+JMP+BAN) + EX*T1(ADD+LDA)
- 图10.3

10条指令的微操作命令时间表p402

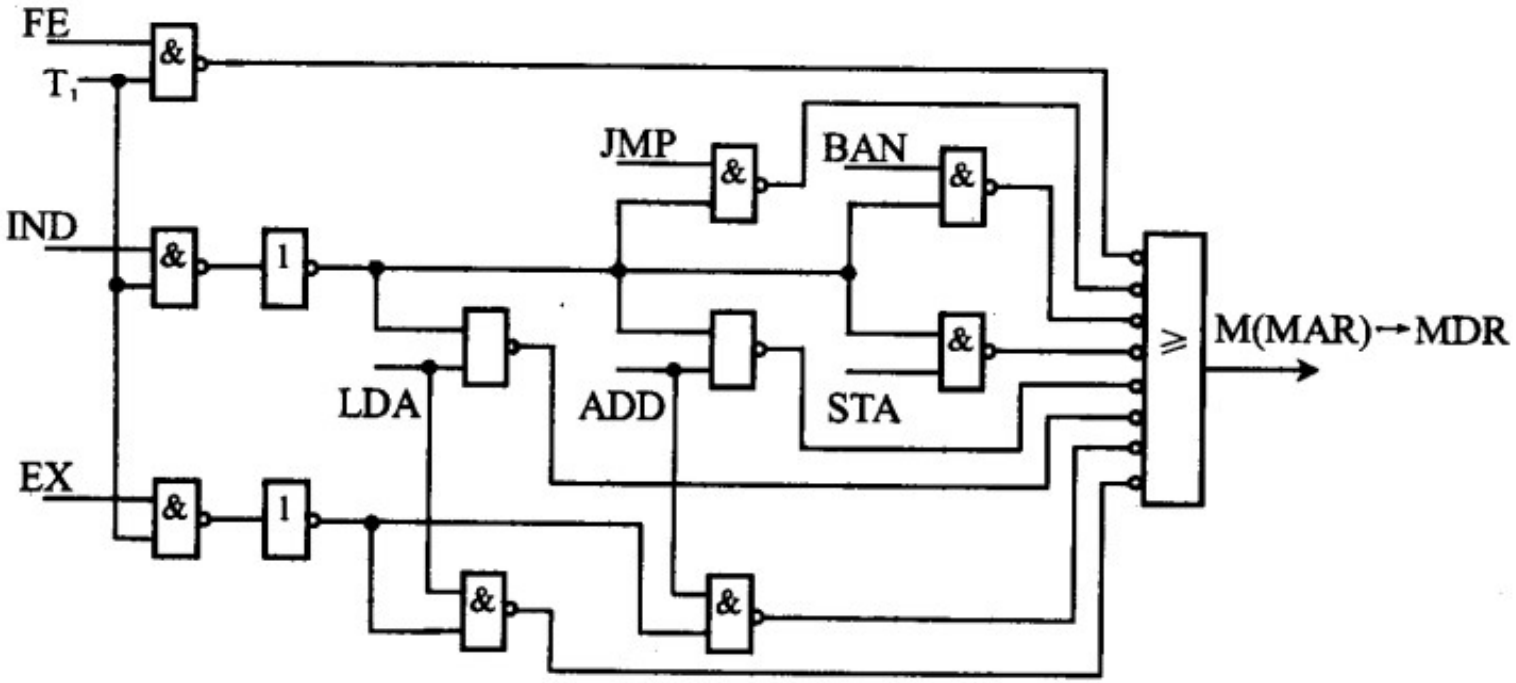
工作周期标记	节拍	状态条件	微操作命令信号	CAL	COM	SHR	CSL	STP	ADD	SAT	LDA	JMP	BAN
FE（取指）	T_0		PC→MAR	1	1	1	1	1	1	1	1	1	1
			1→R	1	1	1	1	1	1	1	1	1	1
	T_1		M(MAR)→MDR	1	1	1	1	1	1	1	1	1	1
			(PC)+1→PC	1	1	1	1	1	1	1	1	1	1
	T_2		MDR→IR	1	1	1	1	1	1	1	1	1	1
			OP(IR)→ID	1	1	1	1	1	1	1	1	1	1
		I	1→IND						1	1	1	1	1
		$\bar{I}$	1→EX	1	1	1	1	1	1	1	1	1	1
IND（间址）	T_0		Ad(IR) →MAR						1	1	1	1	1
			1→R						1	1	1	1	1
	T_1		M(MAR) →MDR						1	1	1	1	1
	T_2		MDR→Ad(IR)						1	1	1	1	1
		$\overline{IND}$	1→EX						1	1	1	1	1
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									
			$\overline{AC}$ →AC		1								
			R(AC) →L(AC)，AC_0不变			1							
			ρ^{-1}(AC)				1						
			Ad(IR)→PC									1	
		A_0	Ad(IR)→PC										1
			0→G					1					

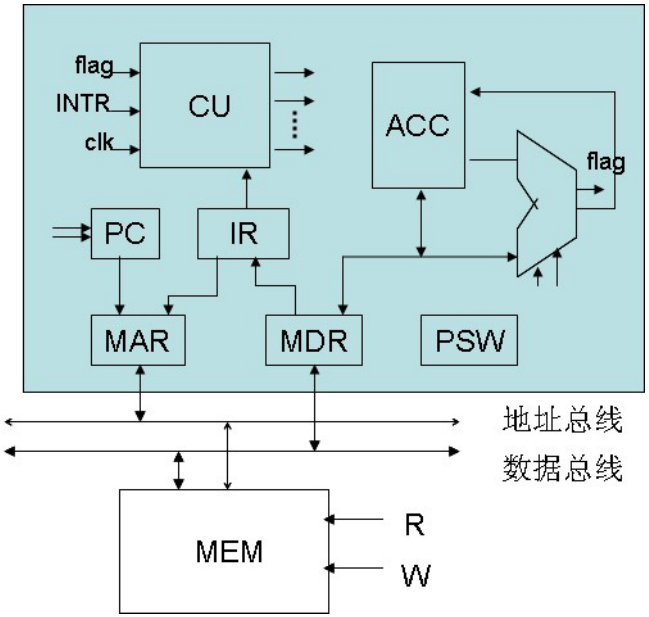
M(MAR)->MDR的逻辑表达式和逻辑图

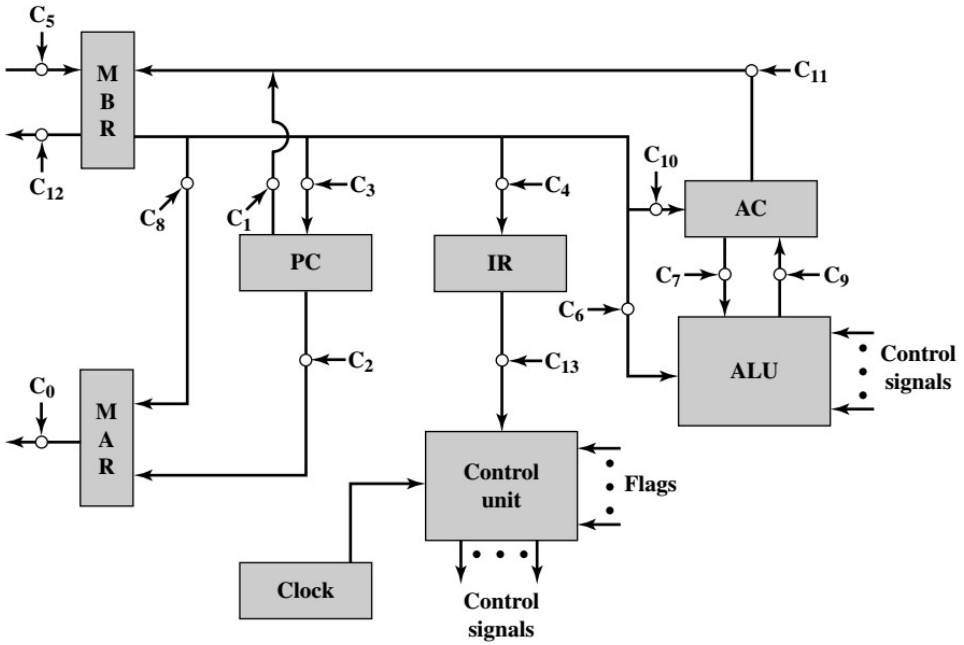
M（MAR）→MDR

＝FE·T1+IND·T1(ADD+STA+LDA+JMP+BAN)+EX·T1(ADD+LDA)

＝T1{FE+IND(ADD+STA+LDA+JMP+BAN)+EX（ADD+LDA）}



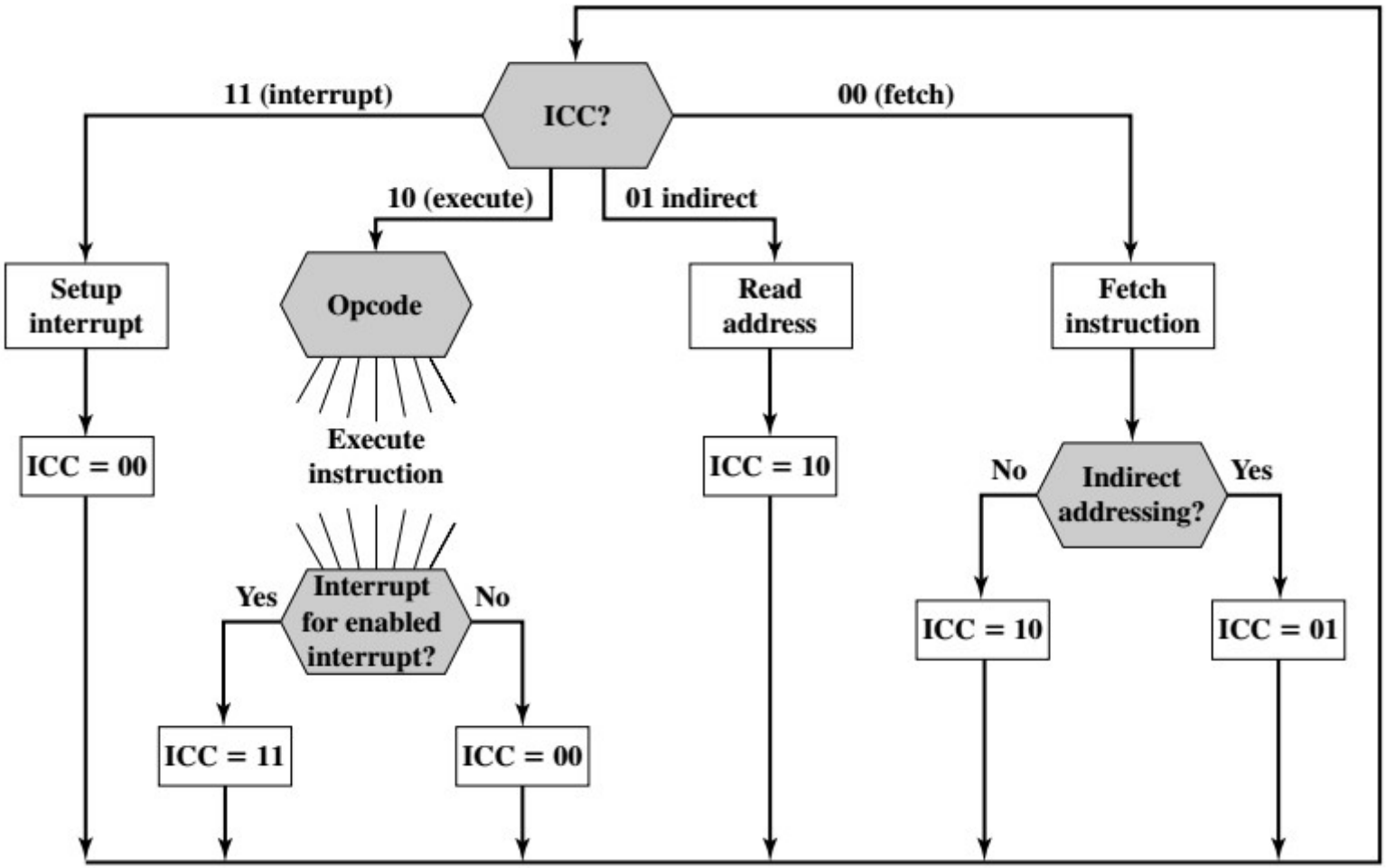




	Micro-operations	Active Control Signals
Fetch:	t_1: MAR ← (PC)	C_2
	t_2: MBR ← Memory PC ← (PC) + 1	C_5, C_R
	t_3: IR ← (MBR)	C_4
Indirect:	t_1: MAR ← (IR(Address))	C_8
	t_2: MBR ← Memory	C_5, C_R
	t_3: IR(Address) ← (MBR(Address))	C_4
Interrupt:	t_1: MBR ← (PC)	C_1
	t_2: MAR ← Save-address PC ← Routine-address	
	t_3: Memory ← (MBR)	C_{12}, C_W

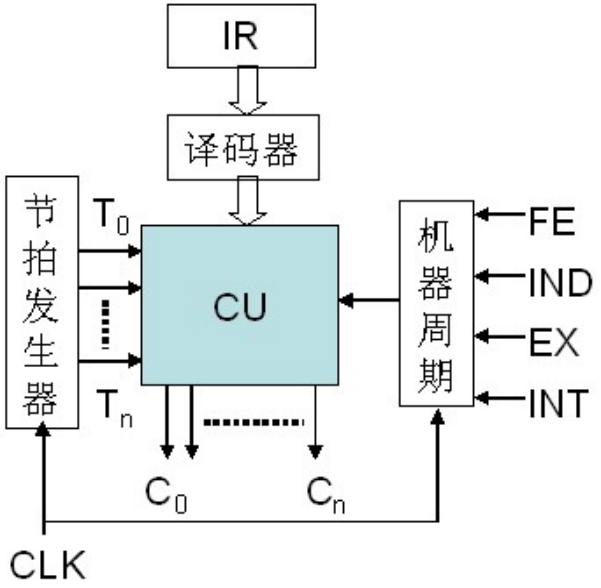
instruction cycle code (ICC)

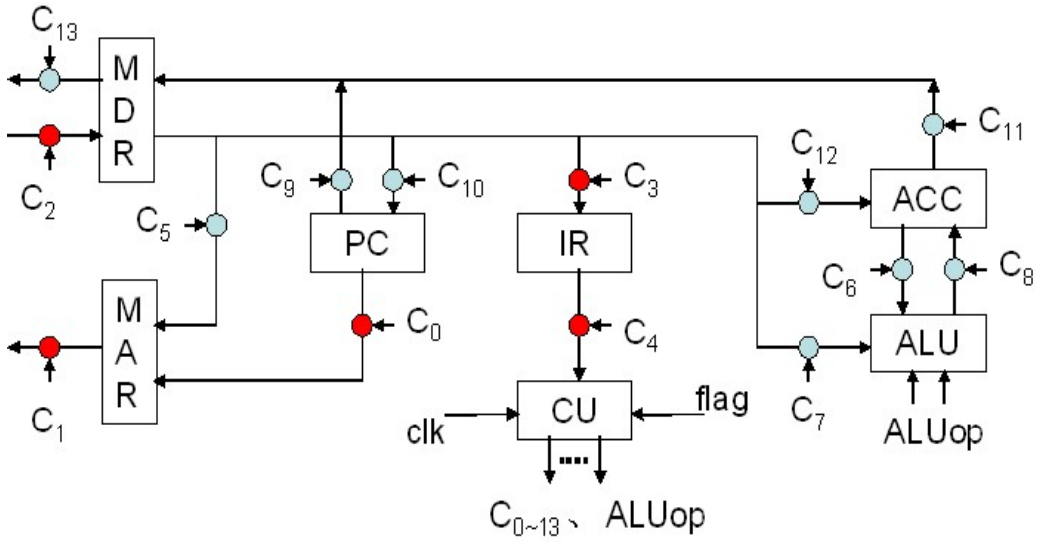
- Flowchart for Instruction Cycle

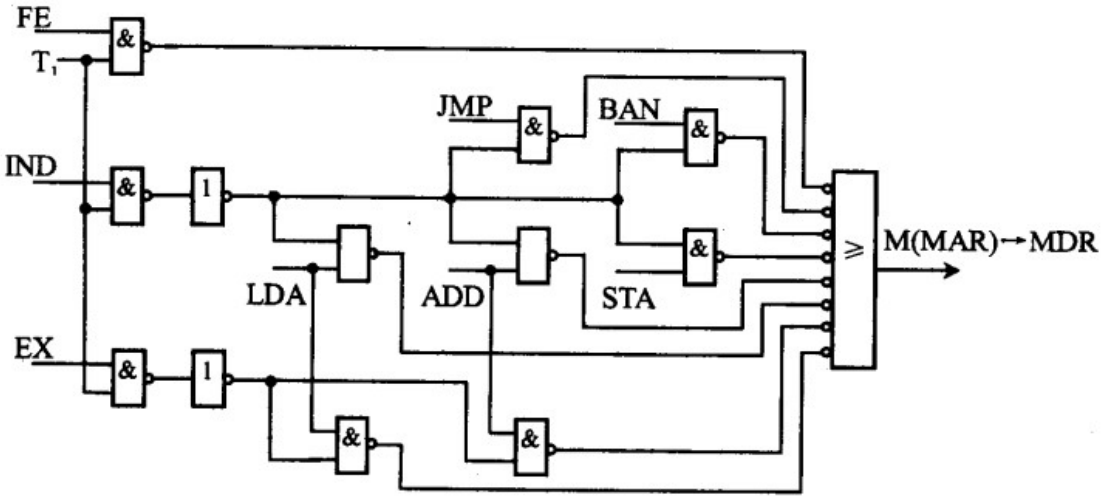


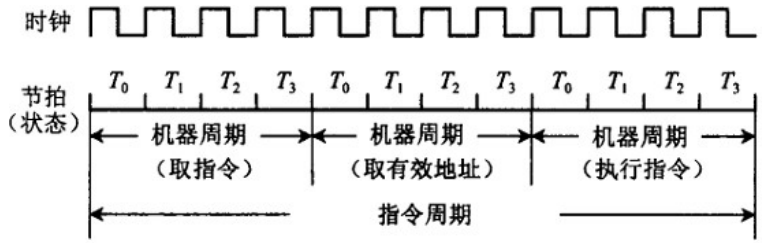
CPU周期ICC：
00: Fetch
01: Indirect
10: Execute
11: Interrupt

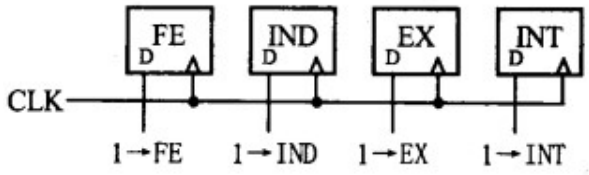
正确的时间，正确的控制


IR
译码器
节拍发生器
CU
机器周期
T_0
T_n
C_0
C_n
FE
IND
EX
INT
CLK


C_{13}
MDR
C_2
C_5
MAR
C_1
C_9
C_{10}
PC
C_0
C_3
IR
C_4
CU
clk
flag
$C_{0\sim13}$、ALUop
C_{12}
C_{11}
ACC
C_6
C_8
ALU
C_7
ALUop


FE
T_1
IND
EX
JMP
BAN
LDA
ADD
STA
M(MAR)→MDR


时钟
节拍（状态）
T_0 T_1 T_2 T_3
机器周期（取指令）
机器周期（取有效地址）
机器周期（执行指令）
指令周期

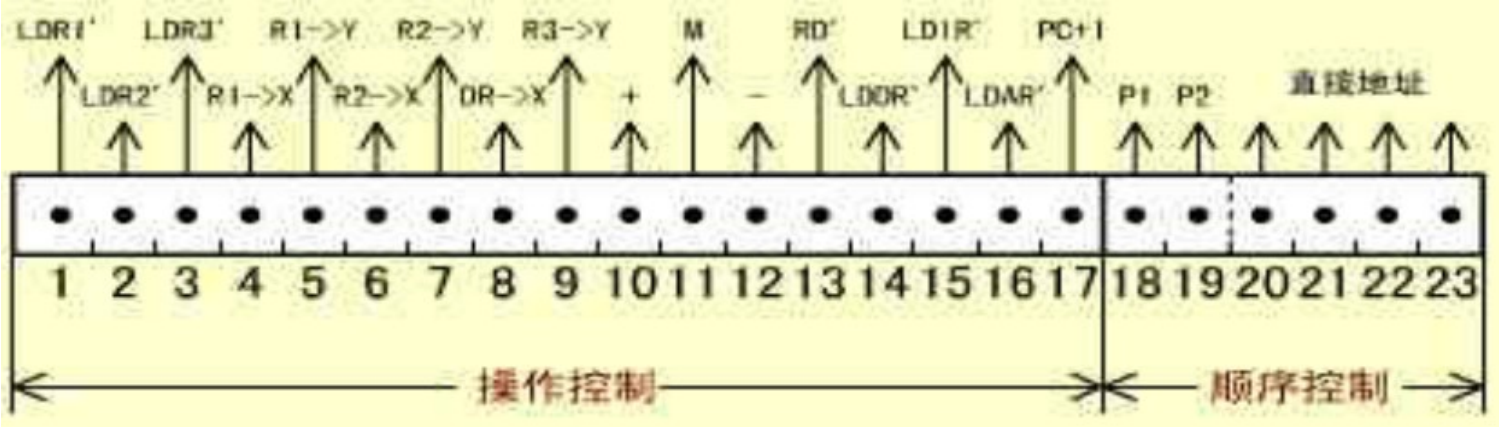

FE
IND
EX
INT
CLK
1→FE
1→IND
1→EX
1→INT

组合逻辑实现说明

- 思路清晰，电路规模大
- 书中给出的只是设计思路，并不是真正的实现。
  - 因为实现的是微操作，而不是微操作完成所需的控制（称“微操作命令”或“微指令”——见微程序设计），而**CU**输出的应该是控制。
  - 如**M(MAR)->MDR**所需的控制为C_1、C_2、

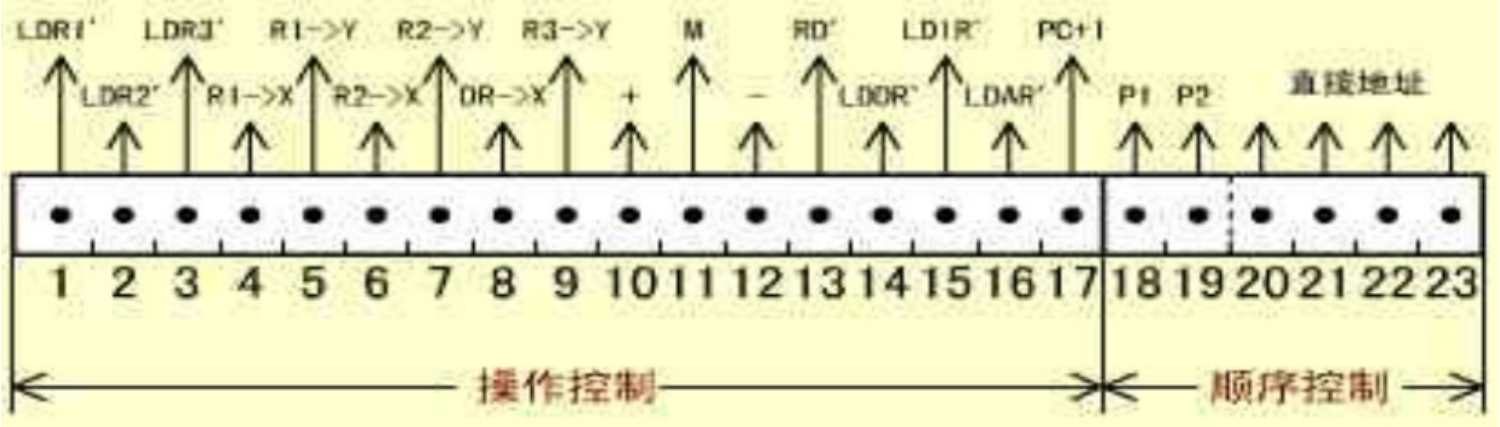
Motivation：微指令、微命令

- 取指周期
  - T_0: PC->MAR, 1->R
  - T_1: M(MAR)->MDR, PC+1->PC
  - T_2: MDR->IR, OP(IR)->ID
- 间址周期
  - T_0: Ad(IR)->MAR, 1->R
  - T_1: M(MAR)->MDR
  - T_2: MDR->Ad(IR)



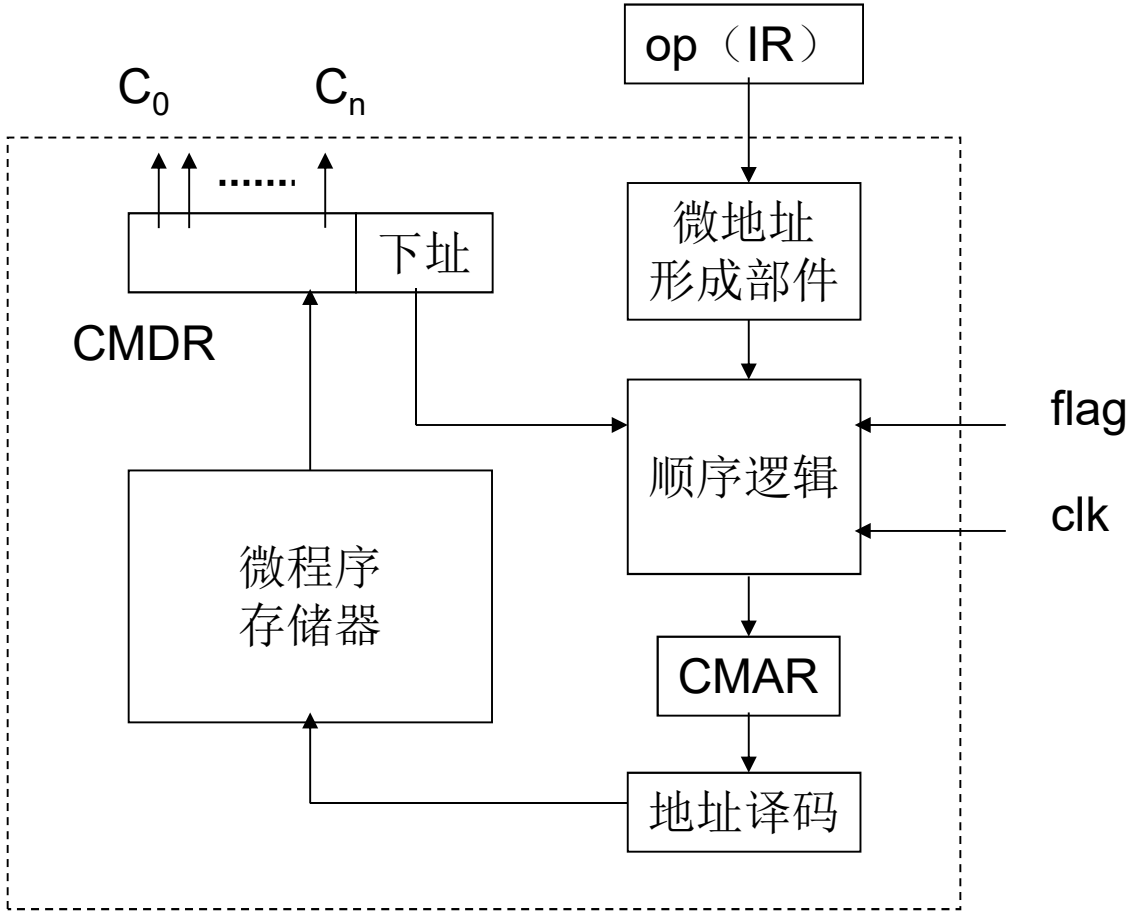
微程序

- 用一个微程序实现一条机器指令
  - 微程序由多条微指令组成
    - 与节拍对应
  - 一条微指令对应一个或多个微操作命令，称“微命令”，即实现微操作的控制信号。
    - 微指令的编码方式
  - 微程序存储在“控存”中
- 微指令的格式



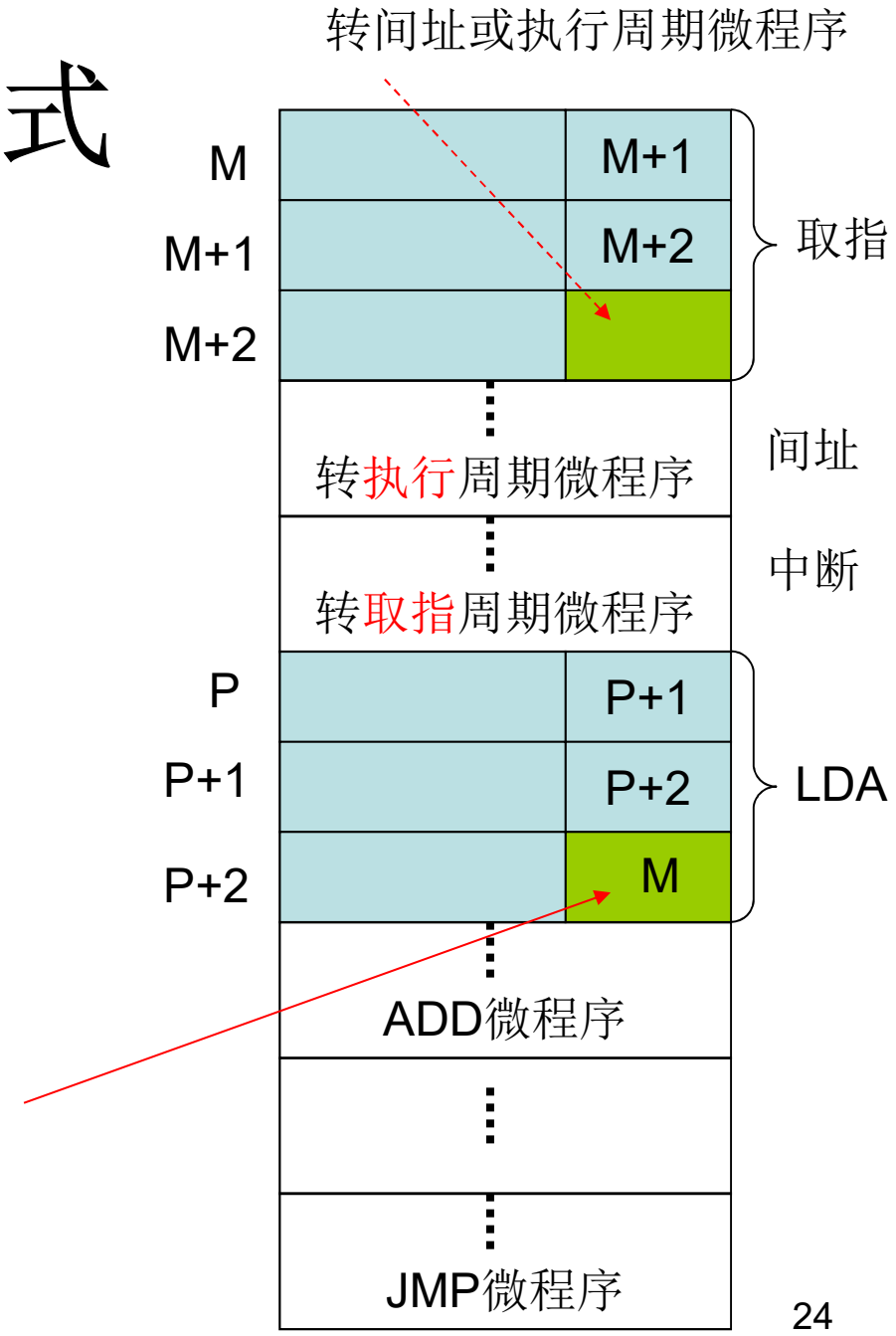
微程序控制部件的结构

- 微地址形成部件
  - 形成微程序的首址
- 顺序逻辑
  - 形成下一条微指令的地址（计数器）
- 微程序存储器（控存）
  - 存储微程序



微程序的存储形式

- “取指”、“间址”、“中断”微程序等三个微程序所有指令共用，“执行”各个指令不同
  - 微程序个数为**3＋X**

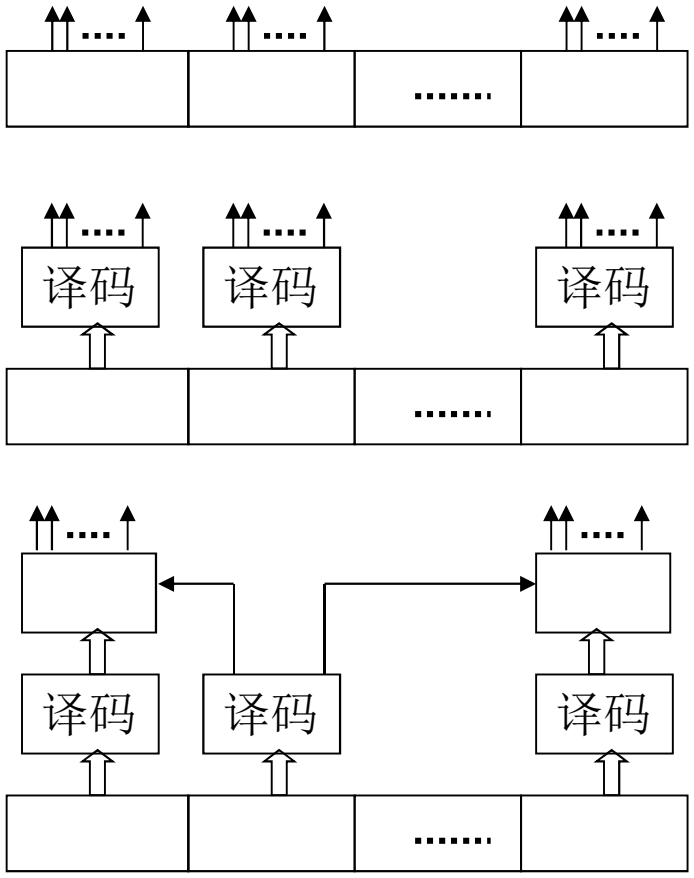


微指令的格式

- 水平微指令
  - 直接编码方式、字段直接编码方式、字段间接译码方式都是水平微指令
  - 并行性好
- 垂直微指令
  - 类似于“指令”，含操作码和操作数、操作方式等。
  - 优点：缩短指令字长

水平微指令的编码方式

- 指微命令的表示方式
  - 直接编码方式
    - 每个微命令一位
    - 直观，但微指令字长太长（几百位），控存容量太大
  - 字段直接编码方式
    - 将命令域划分成多个可以并发的字段，通过译码生成微命令
      - 采用较少的位表示较多的微命令
  - 字段间接编码方式
    - 命令域中的某些字段专用于其他字段的控制



垂直型微指令示例

微操作码	地址码		其他		微指令类别及功能
012	3～7	8～12	13 15		
000	源寄存器	目的寄存器	其他控制		传送型微指令
001	ALU 左输入	ALU 右输入	ALU		运算控制型微指令。按 ALU 字段所规定的功能执行，其结果送暂存器
010	寄存器	移位次数	移位方式		移位控制型微指令。按移位方式对寄存器中的数据移位
011	寄存器	存储器	读写	其他	访存微指令。完成存储器和寄存器之间的传送
100	D			S	无条件转移微指令。D 为微指令的目的地址
101	D		测试条件		条件转移微指令。最低 4 位为测试条件
110 111					可定义 I/O 或其他操作。第 3～15 位可根据需要定义各种微命令。

微指令序列地址的形成

- 直接由下址字段给出后续微指令的地址
- 根据机器指令的操作码形成
  - 微地址形成部件根据操作码形成微程序入口首址
- 增量计数
  - 对一个微程序中的连续微指令**CMAR+1->CMAR**
- 分支转移
  - 根据条件转向不同地址
- 硬件逻辑形成微程序的入口地址
  - 中断、间址

微程序设计方法

- 静态微程序设计
  - 每条机器指令的微程序是预先设计好的。
- 动态微程序设计
  - 可以通过改变微指令和微程序而改变机器的指令系统——“可重构（**reconfigurable**）体系”
- 毫微程序设计
  - 进一步细化，提高并行性
  - **2**级：第一级为垂直微指令，第二级为水平微指令

微程序控制方法

- 微指令的执行分两个阶段：取指，执行
- 串行微程序控制
  - 取指，执行
- 并行微程序控制
  - **2**级流水
  - “结果相关”问题：延迟一个周期

(a) 串行操作

(b) 并行操作

微程序设计示例

- 微程序设计步骤
  - 微操作节拍分配
  - 确定微指令格式与编码
  - 编写每条微指令
- 示例：
  - **10**条指令
  - 本例只考虑取指周期和执行周期，不考虑间址和中断

微程序节拍分配—取指阶段

- 取指周期
  - **T_0: PC->MAR, 1->R**
  - **T_1: Ad(CMDR)->CMAR; /*下址*/**
  - **T_2: M(MAR)->MDR, PC+1->PC**
  - **T_3: Ad(CMDR)->CMAR; /*下址*/**
  - **T_4: MDR->IR, OP(IR)->ID**
  - **T_5: OP(IR)->CMAR; /*“执行” 微程序首址*/**
- 执行一条微指令需要两个时钟周期
- 下址形成方式：直接由下址字段给出

微程序节拍分配—执行阶段

- 访存指令**ADD**
  - **T_0: Ad(IR)->MAR, 1->R**
  - **T_1: Ad(CMDR)->CMAR; /*下址*/**
  - **T_2: M(MAR)->MDR**
  - **T_3: Ad(CMDR)->CMAR; /*下址*/**
  - **T_4: ACC+MDR->ACC**
  - **T_5: Ad(CMDR)->CMAR; /*“取指”微程序首址*/**

微指令格式确定

- 本例共20个微操作
  - 见表
- “取指”＋“执行”微程序长度为38条微指令
- 任务：确定编码方式，下址形成方式，微指令字长
  - 编码方式：采用直接编码方式
    - 20位控制字段，6位下址字段（可以64条微指令）
  - 下址形成方式：指令操作码＋下址字段
  - 指令字长20＋6＝26位

编写微指令码点

- 下表列出了对应**10**条机器指令的微指令码点。表中空格中“**0**”缺省为空。

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	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
取指	00	1	1																											1
	01			1	1																								1	
	02					1																				x	x	x	x	x	x
CLA	03						1																							
COM	04							1																						
SMR	05								1																					
CSL	06									1																				
STP	07										1																			
ADD	10		1									1																1			1
	11			1																								1		1	
	12												1																		
STA	13											1		1														1	1		
	14														1													1	1		1
	15															1															
LDA	16		1									1																1	1	1	1
	17			1																							1				
	20																1														
JMP	21																	1													
BAN	22																		1												

本例的优化

- 本例共**20**个微操作，其中含一个“下址生成微操作”和一个“执行微程序首址生成微操作”
- “取指”＋“执行”微程序长度为**38**条微指令（或周期），其中有**19**条微指令（或周期）是为了得到下址
- 如果直接将**Ad(CMDR)**送到控存地址线，并使用**MUX**控制下址的来源，则可省略**2**条微操作“**Ad(CMDR)->CMAR**”和“**OP(IR)->CMAR**”
  - 也即节省了**19**个时钟周期

小结

- **CU**设计步骤：正确时间产生正确控制
  - 组合逻辑实现
  - 微程序控制（简化硬件设计复杂度）
    - 基本概念：微指令、微程序，下址，控存，水平微指令、垂直微指令

- 作业：**10.2**、**10.3**、**10.9**


Thank You