计算机组成原理

3.4 模型机CPU设计方法-4

2022/10/26

信息与软件工程学院
School of Information and Software Engineering

五、微程序控制方式

组合逻辑控制：

由逻辑电路产生控制信号；

微程序控制：

由微指令代码(μI)预存控制信号，经电路翻译代码后产生控制信号

五、微程序控制方式

设计思想

一条机器指令的执行分为若干步，将每一步操作所需的微命令按照固定格式进行编码，并存储成一条**微指令**，多条微指令构成一段**微程序**，这段微程序对应一条**机器指令**。

机器在执行指令过程中，每一步（时钟周期）取出一条微指令，经过译码后，产生一组控制信号（微命令），控制各个部件的操作。

控制存储器（Control memory,CM,控存）：只读存储器

存储可以经过译码产生控制信号的微程序（经过按照一定规则进行编制）。

注意：控存与主存的区别

五、微程序控制方式

微程序（ROM）存放地址

五、微程序控制方式

【例】

	分步	微命令序列	
机器指令	M→IR,	EMAR,R,SIR	微指令1
MOV R1,R0	PC+1→PC	PC→A,A+1,DM,...	
	R0→R1	R0→A,传A,DM,...	微指令2
	PC→MAR	PC→A,传A,DM, ...	微指令3

微程序

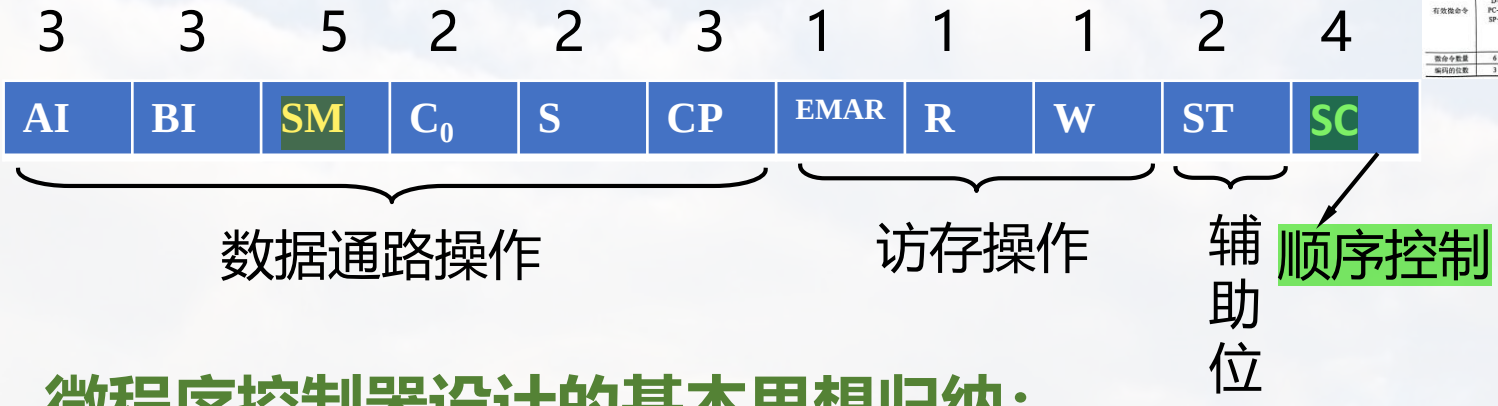
一条机器指令的执行对应一段微程序；

一段微程序可包含多条微指令；

一条微指令包含一步操作所需微命令

五、微程序控制方式

【例】模型机的微指令格式



微程序控制器设计的基本思想归纳：

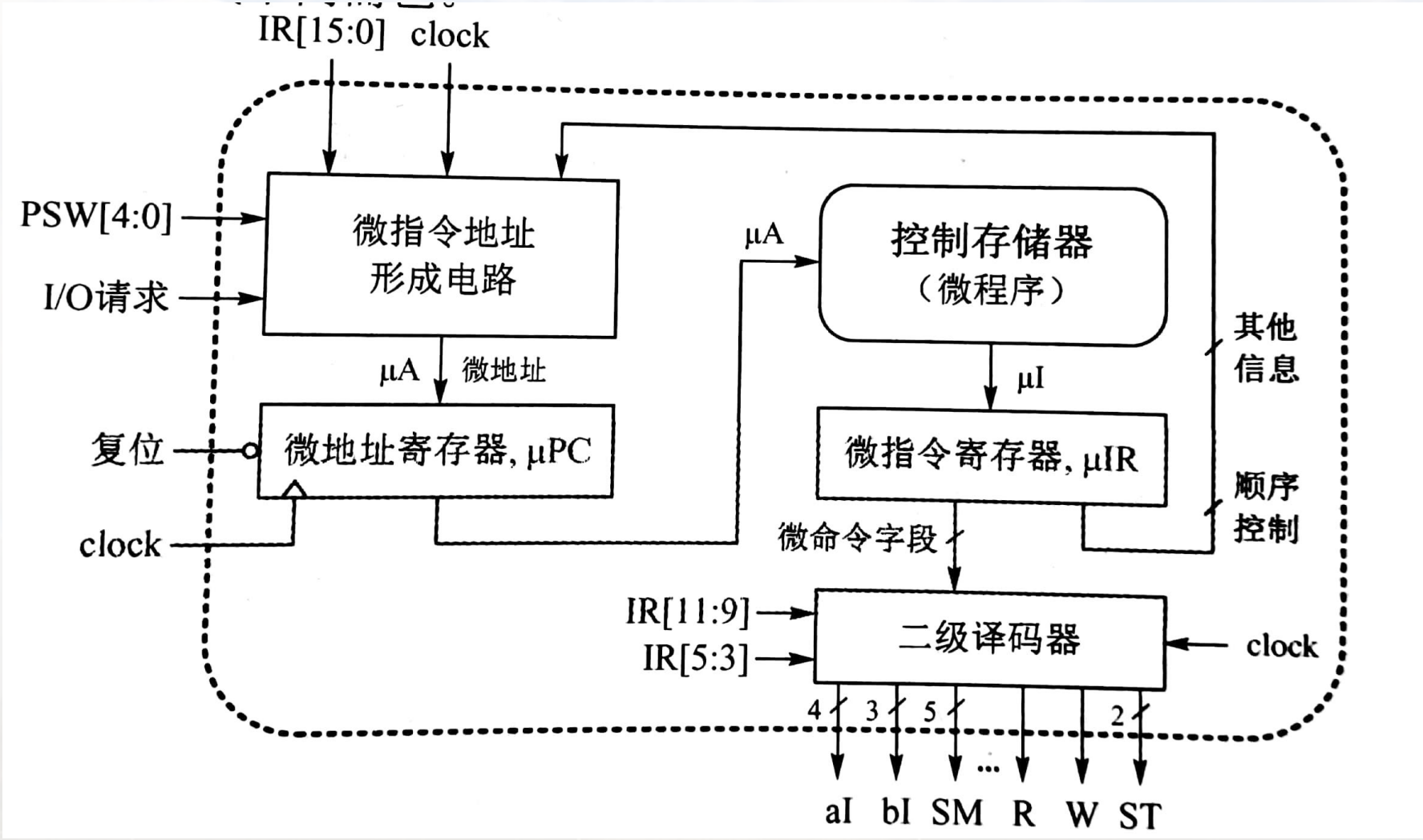
(1)时钟周期与微指令周期相同，每个微指令周期执行一条微指令，控制实现一步操作;

(2) 微程序事先存放在控制存储器中，执行机器指令时再取出，经译码形成相应的控制信号（微命令）。

(3) 若干微指令组成一段微程序，解释执行一条机器指令;

五、微程序控制方式

模型机微程序控制器框图



1. 原理、部件与工作过程

→ 微命令由预存储的代码翻译产生

→ 1条机器指令对应多条微指令 代码

（1）时序系统

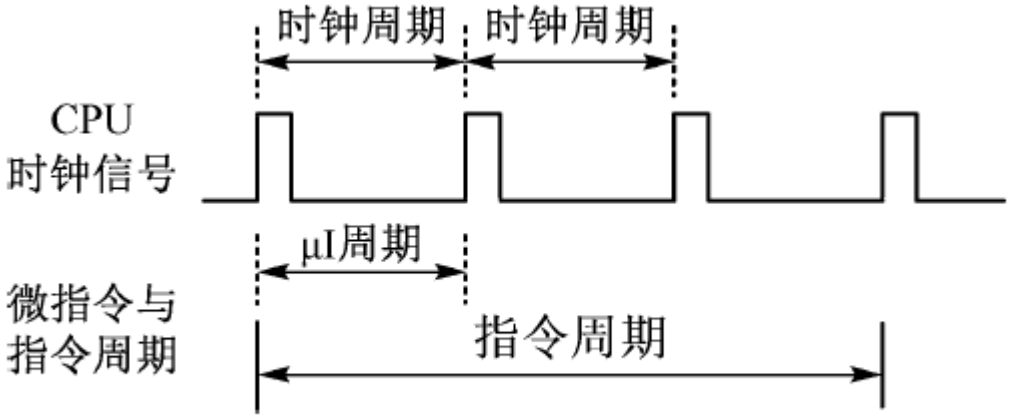


图 3-63　指令周期与微指令周期

组合逻辑方式下1个时钟周期完成的工作，现在由1个微指令周期控制完成。

五、微程序控制方式

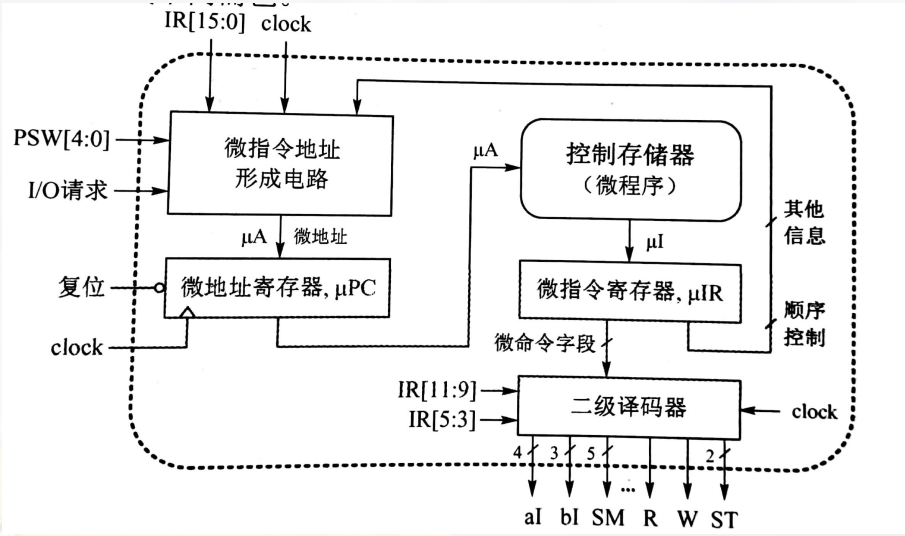
（2）控制系统的逻辑组成

①控制存储器CM

功能:存放微程序，(CM在CPU内部, 不是主存储器)。

②微指令寄存器uIR

功能:存放现行微指令。



微命令字段:
提供某一步操作所需的微命令。
(微操作控制字段)

微地址字段:
(顺序控制字段)

- 指明后续微地址的形成方式。
- 提供微地址的给定部分。

五、微程序控制方式

③微地址形成电路

功能:提供两种类型的微地址。

- 微程序入口地址: 由机器指令操作码形成。
- 后续微地址: 由微地址字段、现行微地址、运行状态等形成。

④微地址寄存器uPC

功能:存放下一条微指令在CM中的地址（微地址）

⑤微命令译码电路

功能:将微指令进行译码，输出指令所需要的控制信号（微命令）

五、微程序控制方式

(3) 微程序工作过程

① 取机器指令

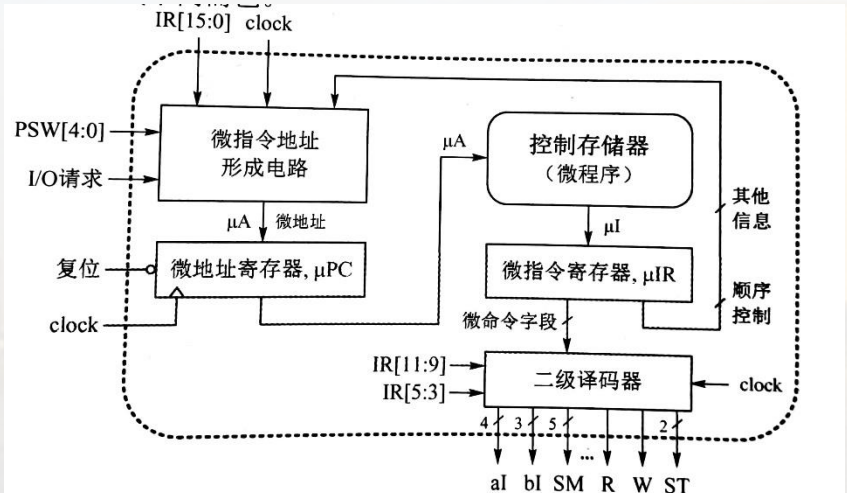
CM —取指微指令→ μIR —微命令字段→ 译码器 —微命令→ 主存 —机器指令→ IR

②转微程序入口

IR —操作码→ 微地址形成电路 —入口→ μPC → CM —首条微指令→ μIR

③执行首条微指令

μIR —微命令字段→ 译码器 —微命令→ 操作部件(运算器/寄存器/存储器部件等)



五、微程序控制方式

④取后续微指令

微地址字段
现行微地址
运行状态
→ 微地址形成电路 —后续微地址→ μPC → CM
CM —后续微指令→ μIR

⑤执行后续微指令：　同③

μIR —微命令字段→ 译码器 —微命令→ 操作部件（运算器/寄存器/存储器部件等）

⑥返回　微程序执行完，返回CM

（返回到存放取指微指令的固定单元00H）。

2. 微指令的编码方式与微地址形成

微指令的一般格式:

微命令字段n	.....	微命令字段1	地址字段

- 一条微指令包含多少微命令字段?
- 微命令字段如何编码?
- 如何指明后续微指令的地址?

五、微程序控制方式

1. 微指令的编码方式

(1) 直接控制法(不译法)

微命令按位给出 { 为1, 选用, 表示一种微命令
为0, 不选用, 不表示微命令

例. 某微指令

C0= { 0 进位初值为0
1 进位初值为1

R= { 0 不发读命令
1 发读命令

优点: 不需译码, 产生微命令的速度快;

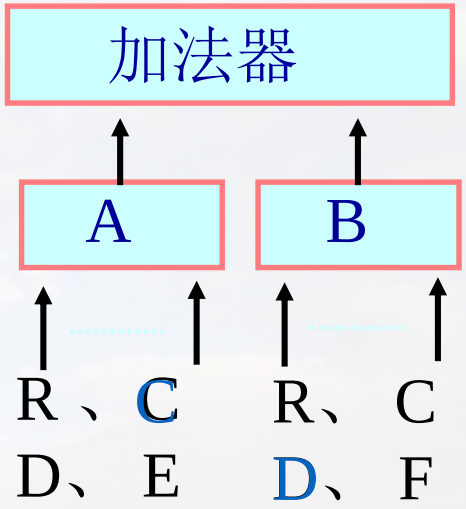
缺点: 信息的表示效率低。如果一条微指令命令字段有n位, 则只能表示n种微操作。

微指令中通常只有个别位采用直接控制法。

(2) 分段直接编译法

一条微指令分为多个微命令字段，每一字段的不同编码表示不同的微命令，即微命令由字段编码给出。

例：加法器A输入端的控制命令存放AI字段，
B输入端的控制命令存放BI字段。



AI: 000 不发命令
001 R→A
010 C→A
011 D→A
100 E→A

BI: 000 不发命令
001 R→B
010 C→B
011 D→B
100 F→B

分段直接编译法特点:
编码较简单;一条微指令能同时提供若干微命令，便于并行操作

(3) 分段间接编译法

微命令由本字段编码和其它字段解释共同给出。

① 设置解释位或解释字段

例.

解释位

$$C=\begin{cases}0 & \text{A为某类命令}\\ 1 & \text{A为常数}\end{cases}$$

② 分类编译

按功能类型将微指令分类，分别安排各类微指令格式和字段进行编码，并设置区分标志。

$$\text{微指令}\begin{cases}\text{CPU方式(触发器C=0或某字段=0)}\\ \text{I/O方式(触发器C=1或某字段=1)}\end{cases}$$

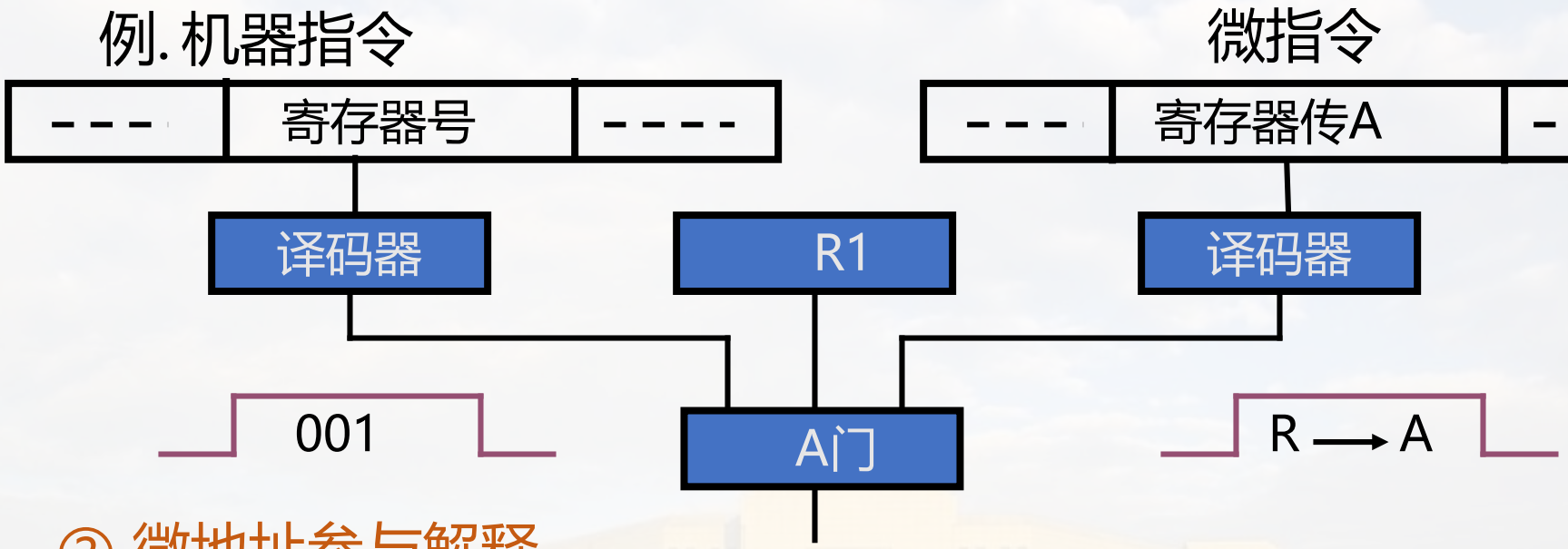
例.

解释位 或 触发器

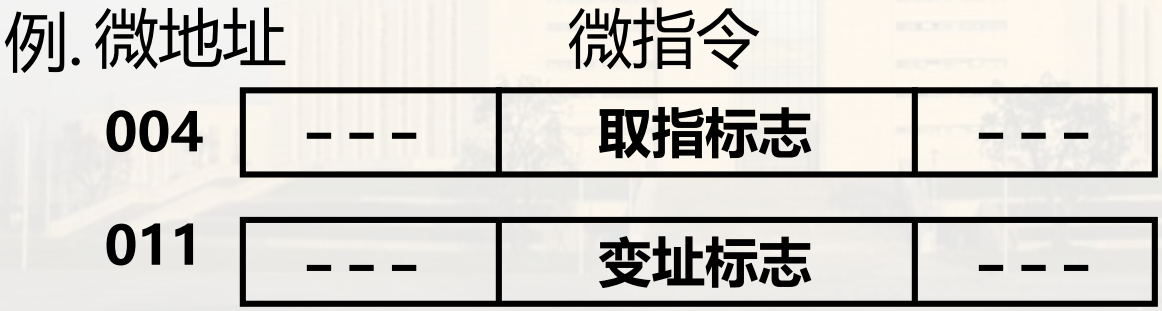
$$C=\begin{cases}0 & \text{A为CPU操作类命令}\\ 1 & \text{A为I/O操作类命令}\end{cases}$$

(4) 其他编码方法

① 微指令译码与机器指令译码复合控制



② 微地址参与解释



2. 微地址形成

关于公共入口问题:

不同机器指令存在一些公共的微操作, 如:

取指所需微操作、取操作数所需微操作等。

因此，不是将所有机器指令所需微操作顺序排列后存入微程序库(CM),

而是将公共操作构成微子程序存入CM, 并在需要执行该操作时, 转入该微子程序、或者形成若干分支。

(从指令流程图中也可看出，不同类型的指令执行时，存在许多相同的操作，在微程序库中，这些相同操作通常作为微子程序存在。)

关于微地址的类型与获得

- 取指令公共入口地址

  设计时约定(即硬件设计时设定，0#存储单元)

- 一条机器指令对应的微程序首地址

  ① 一级功能转移(指令类型)

  ② 二级功能转移（操作数寻址方式）

- 后继微指令地址(含顺序、转移、转子执行)

  ① 增量方式　② 断定方式

(1) 形成初始地址

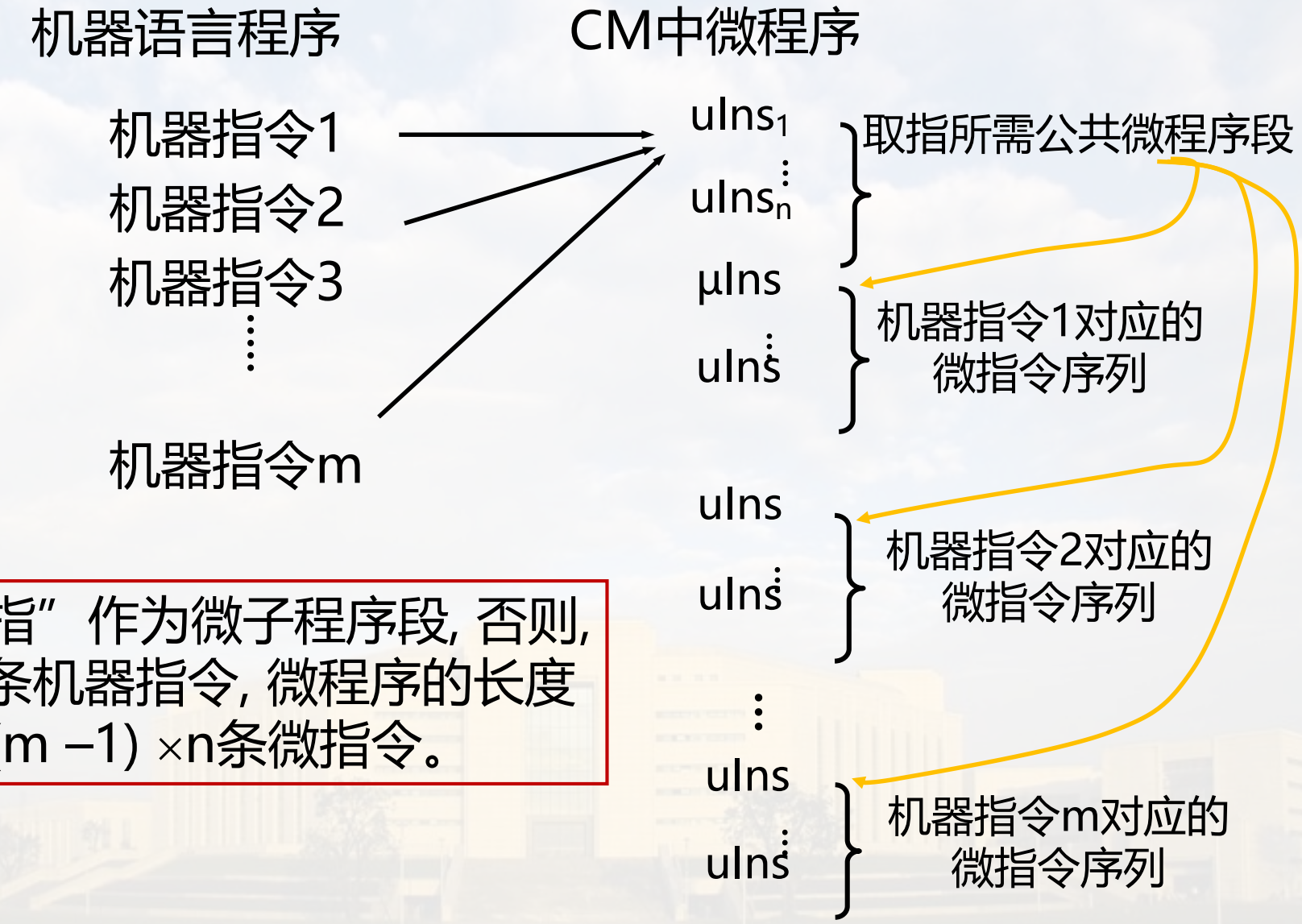
1) 取机器指令

所有指令的取指阶段的操作是相同的，因此将取指所需的所有微操作指令构成一段公共微程序(即微子程序)，并约定一个公共入口地址(比如从地址0开始)。

取指阶段结束后，经过指令译码，再根据指令类型转入不同指令各自对应的微程序段。

如下图所示:

五、微程序控制方式



将“取指”作为微子程序段，否则，上述m条机器指令，微程序的长度将增加(m –1) ×n条微指令。

2) 功能转移 (按指令功能转入对应的微程序段)

指令操作码 —功能转移→ 微程序入口

① 一级功能转移

各操作码的位置、位数固定, 一次转换成功。

入口地址= | 页号 | 操作码 |

例: 假设页号为00

机器指令(Ins_1)

0F(8位)	.....

入口地址=000FH → 0页

机器指令2 (Ins_2)

10(8位)	.....

入口地址=0010H

五、微程序控制方式

机器指令

Ins_1 → 000F

Ins_2 → 0010

⋮

问题:

指令系统中，操作码是连续编码，采用该地址形成方法，形成的微指令地址也是连续的，只能形成一条机器指令所对应的多条微命令的第一条微命令的地址。

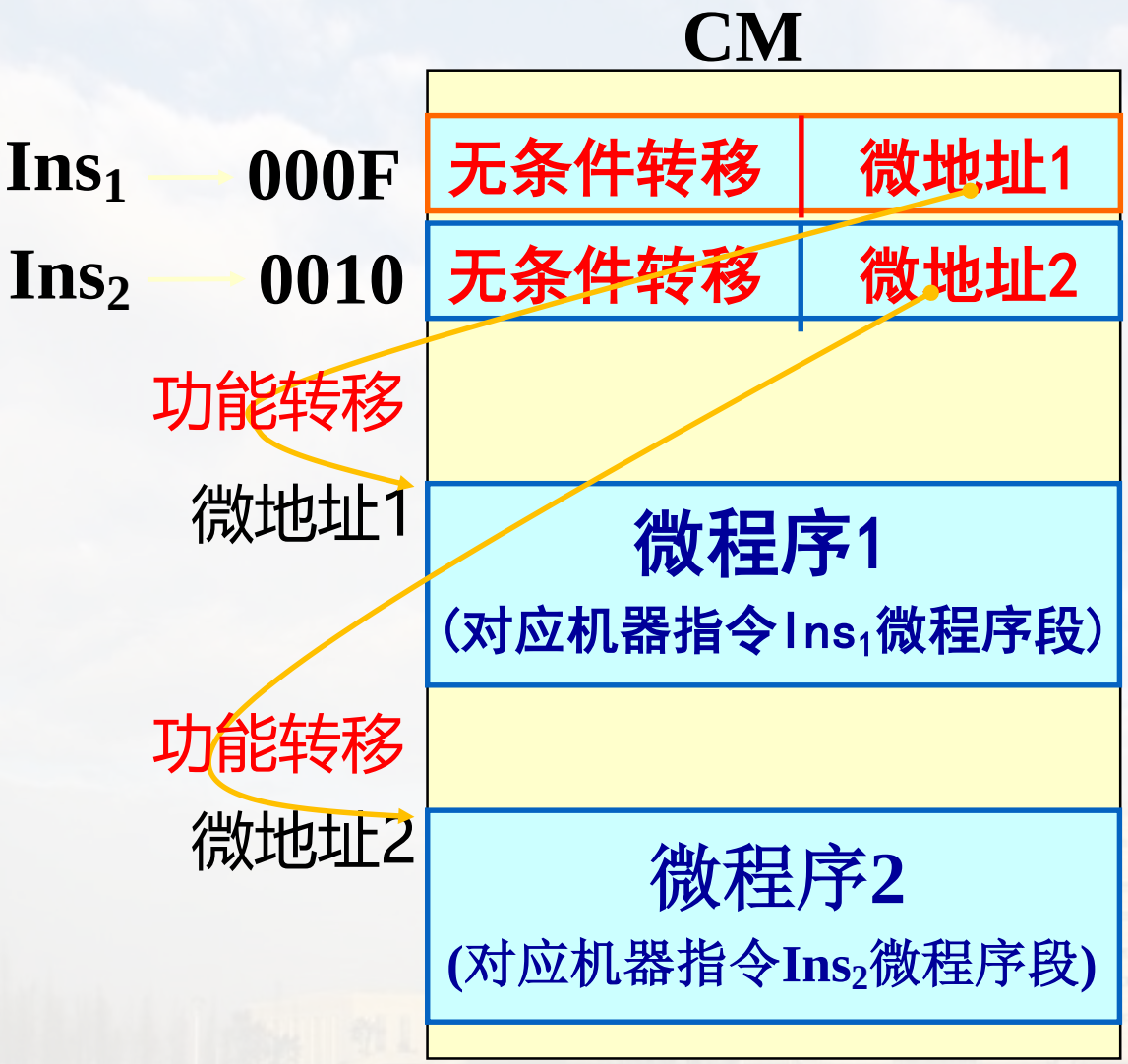
解决方法：采用再次转移:

CM

Ins_1对应的第一条微命令

Ins_2对应的第一条微命令

五、微程序控制方式



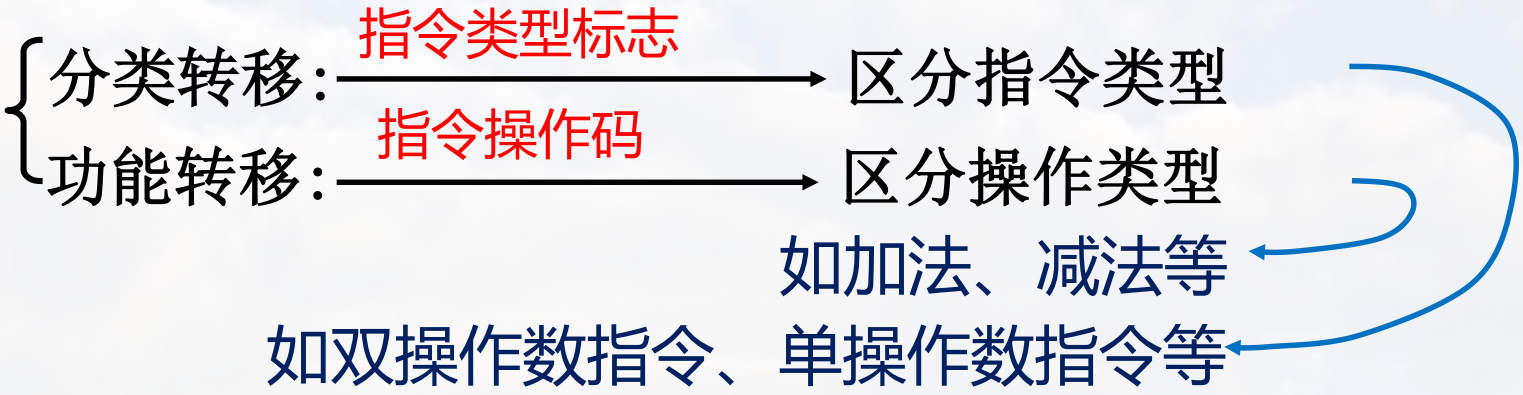
采用一级功能转移的前提条件:

各操作码的位置固定和位数固定

才能方便获得该操作码

才能获得定长地址码

② 二级功能转移(多级转移)



【例】某指令系统:

– 双操作数指令的操作码4位，其中:

高两位为00，作为双操作数指令类型标志;

– 单操作数指令的操作码6位，其中:

高两位为01，作为单操作数指令类型标志。

五、微程序控制方式

假设微ROM地址长度10位，转移过程可描述为:

根据指令类型(如双操作数指令、单操作数指令)转入一个固定地址

↓

给定一个地址的高位，操作码为地址低位,形成一个10位地址码进行功能转移

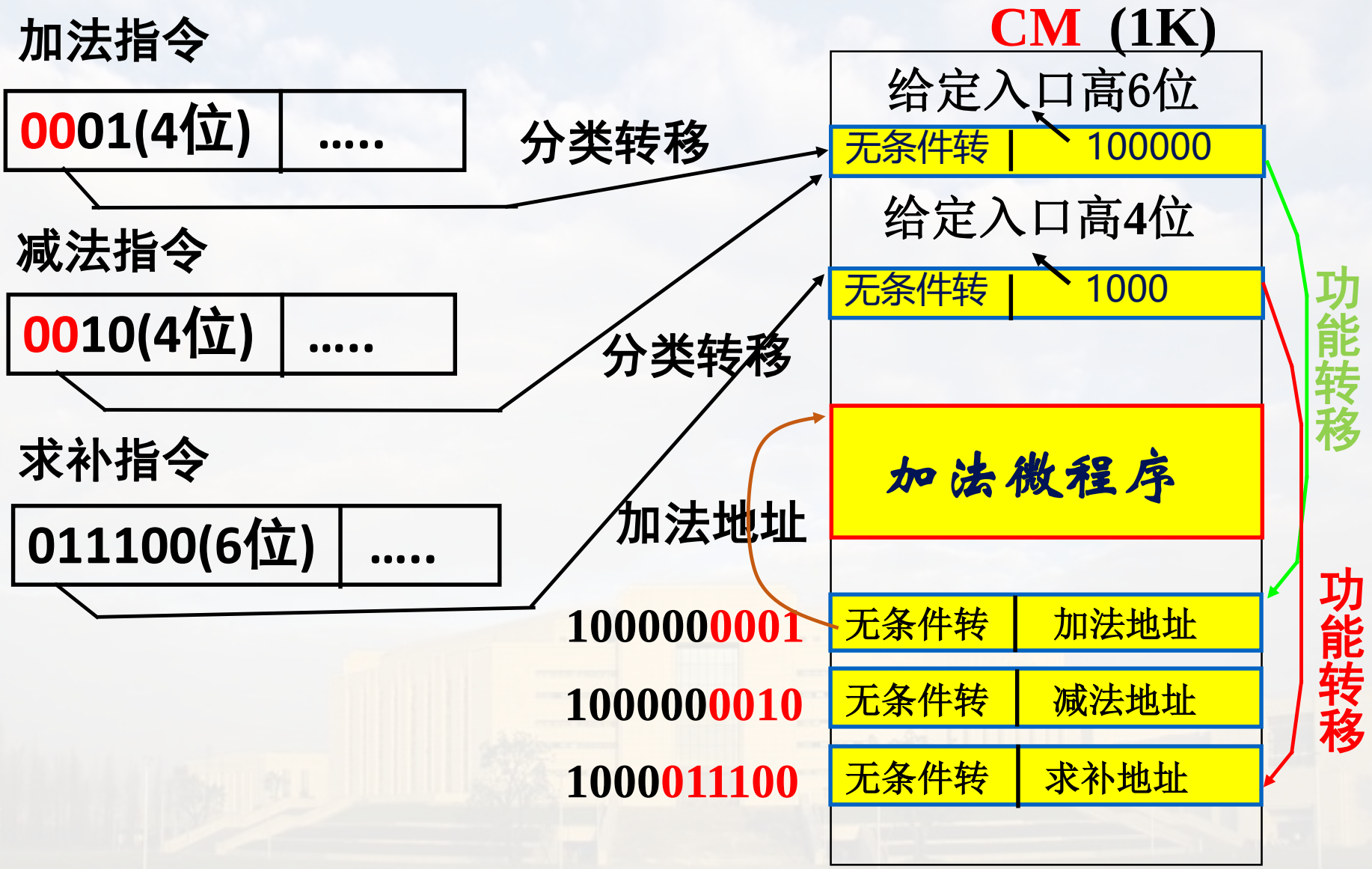
↓

再次转移到该指令所对应的微程序段

对双操作数指令，给定地址高6位

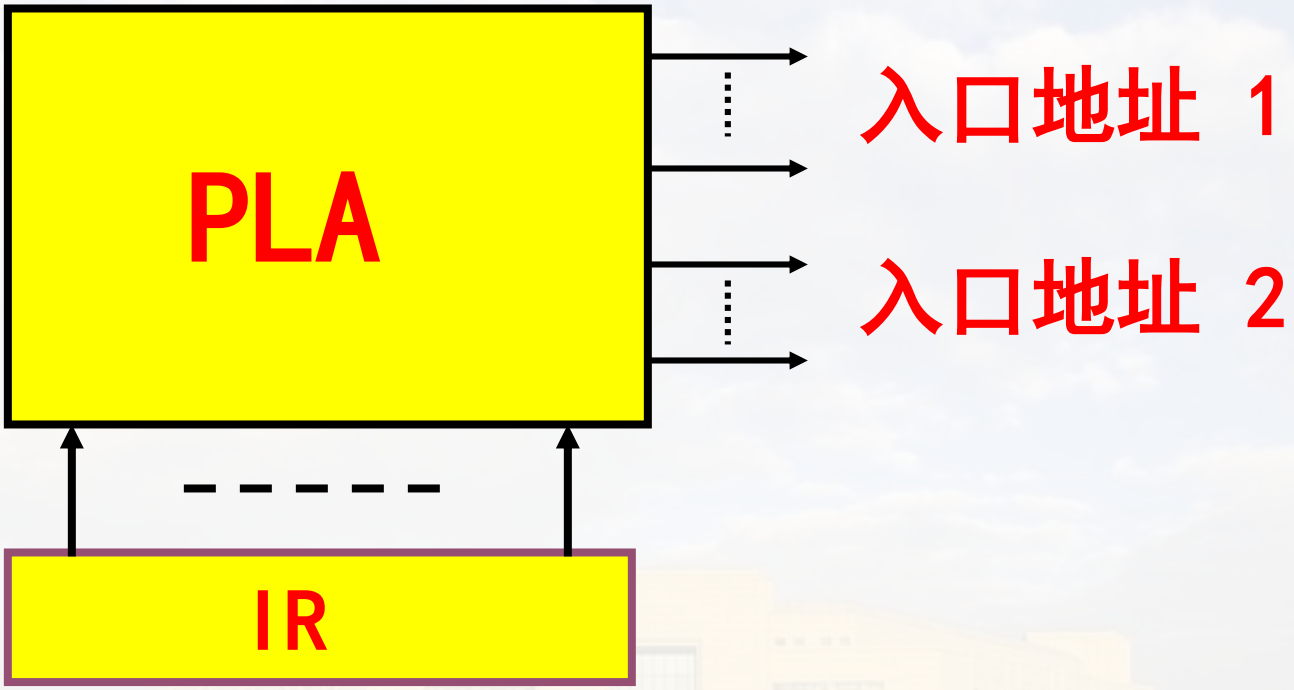
对单操作数指令，给定地址高4位

五、微程序控制方式


加法指令
0001(4位)
.....
减法指令
0010(4位)
.....
求补指令
011100(6位)
.....
分类转移
分类转移
加法地址
CM (1K)
给定入口高6位
无条件转
100000
给定入口高4位
无条件转
1000
加法微程序
1000000001
无条件转
加法地址
1000000010
无条件转
减法地址
1000011100
无条件转
求补地址
功能转移
功能转移

五、微程序控制方式

③用可编程逻辑阵列PLA实现功能转移



五、微程序控制方式

(2) 后续微地址的形成(增量方式和断定方式)

① 增量方式: 微指令某一字段提供后继地址

顺序: 现行微地址+1；

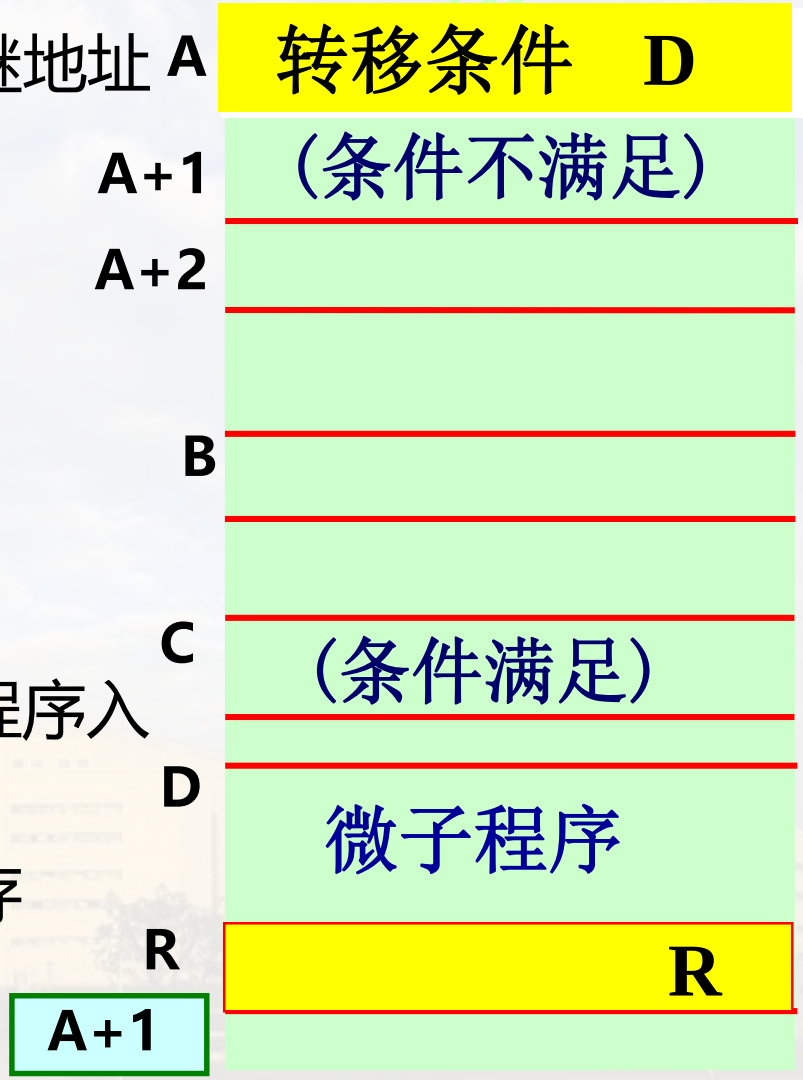
跳步: 现行微地址+2；

无条件转移: 现行微指令给出转移微地址;

条件转移: 现行微指令给出转移微地址和转移条件;

转微子程序: 现行微指令给出微子程序入口;

返回微主程序: 现行微指令给出寄存器号, 该寄存器的值为返回地址



五、微程序控制方式

② 断定方式

由直接给定地址和断定条件相结合形成微地址。

微指令直接给出后续地址中的某些位

检测某种条件或状态，形成后续地址的其余位

后续地址：$A_n \sim A_i$　$A_{i-1} \sim A_0$ → $A_nA_{n-1}\ldots A_1 \sim A_0$

一般格式：

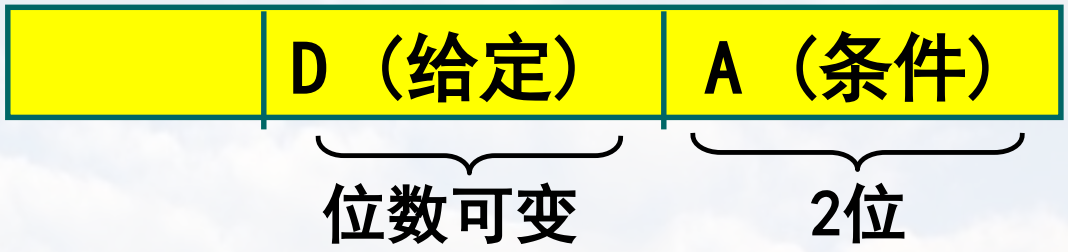
	给定部分	断定条件

给定后续微地址高位部分(非因变分量)

指明后续微地址低位部分的形成方式(因变分量)

五、微程序控制方式

例：微指令



假设微地址10位，约定：

A=
- 01 操作码作为微地址低4位，D给定高6位 → 可形成16路分支
- 10 源寻址方式编码作为微地址低3位，D给定高7位 → 可形成8路分支
- 11 目的寻址方式编码作为微地址低3位，D给定高7位 → 可形成8路分支

3、模型机微指令格式

按数据通路各段操作划分字段，同类操作中互斥的微命令放同一字段。(分段直接编译法)

（1）格式, 32位

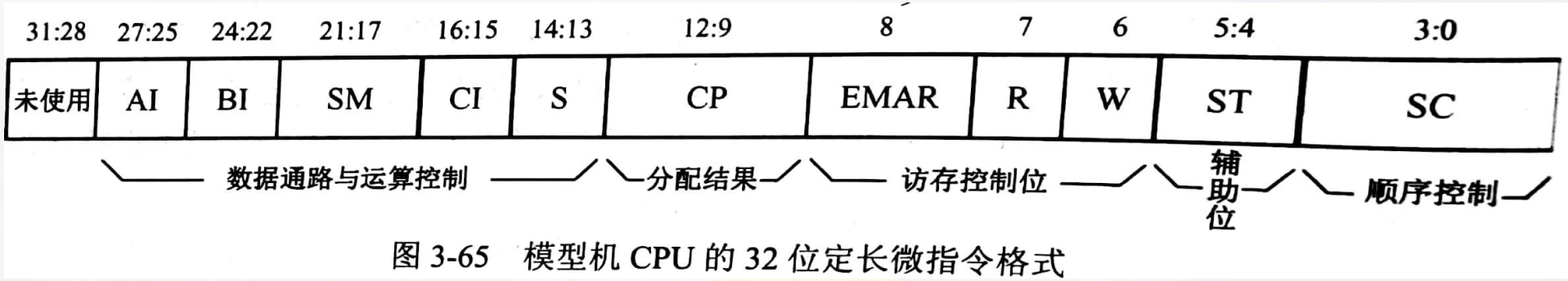


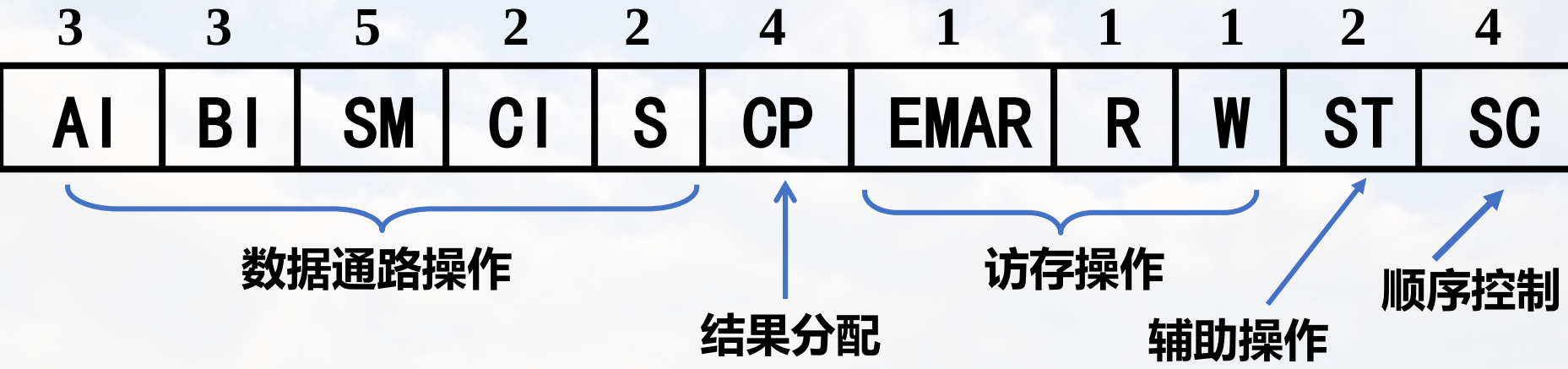
图 3-65　模型机 CPU 的 32 位定长微指令格式

— 微命令字段划分: 由3个部分组成

(1) 数据通路操作部分　　(2) 访存操作部分

(3) 辅助操作部分　　(4) 顺序控制部分(后续地址)

五、微程序控制方式



(2) 各字段功能

1) 数据通路操作

AI：A输入选择

R0～R3、SP、PC

000 无输入

001 Ri ⟶ A

010 C ⟶ A

011 D ⟶ A

100 SP ⟶ A

111 PC ⟶ A

五、微程序控制方式

BI： B输入选择

000 无输入

001 Ri→B

010 C→B

011 D→B

100 MDR→B

R0～R3、SP、PC

SM：ALU功能选择 (S3S2S1S0 M)

CI： 初始进位选择

S： 移位选择

0000 不输出脉冲

CP：结果分配

0001 CPRi (R0～R3、SP、PC)

0010 CPC

⋮

五、微程序控制方式

2）访存操作

EMAR、R、W

3）辅助操作 ST

00 无操作　　10 关中断

01 开中断　　11 SIR

4. 顺序控制 SC

指明微地址形成方式 { 增量 / 断定 } > 10种

0000 顺序执行

0001 无条件转移(D_{26}~D_{19}提供8位转移地址) 增量

断定：

0010 按操作码分支

0011 按操作码和目的寻址方式是寄存器型还是非寄存器型来断定, 实现分支。

0100 按转移条件满足与否和是否PC型寻址断定, 实现分支。

五、微程序控制方式

0101	按源操作数寻址方式断定实现分支	断定
0110	按目的操作数寻址方式断定实现分支	
0111	转微子程序(由D_{26}~D_{19}提供8位转移地址)	增量
1000	返回微主程序，由返回微地址寄存器提供返回地址	
1111	按照“转移寻址方式”断定，分支转移	断定

五、微程序控制方式

把从CM输出的微指令代码，转换成数据通路所需的直接控制信号。

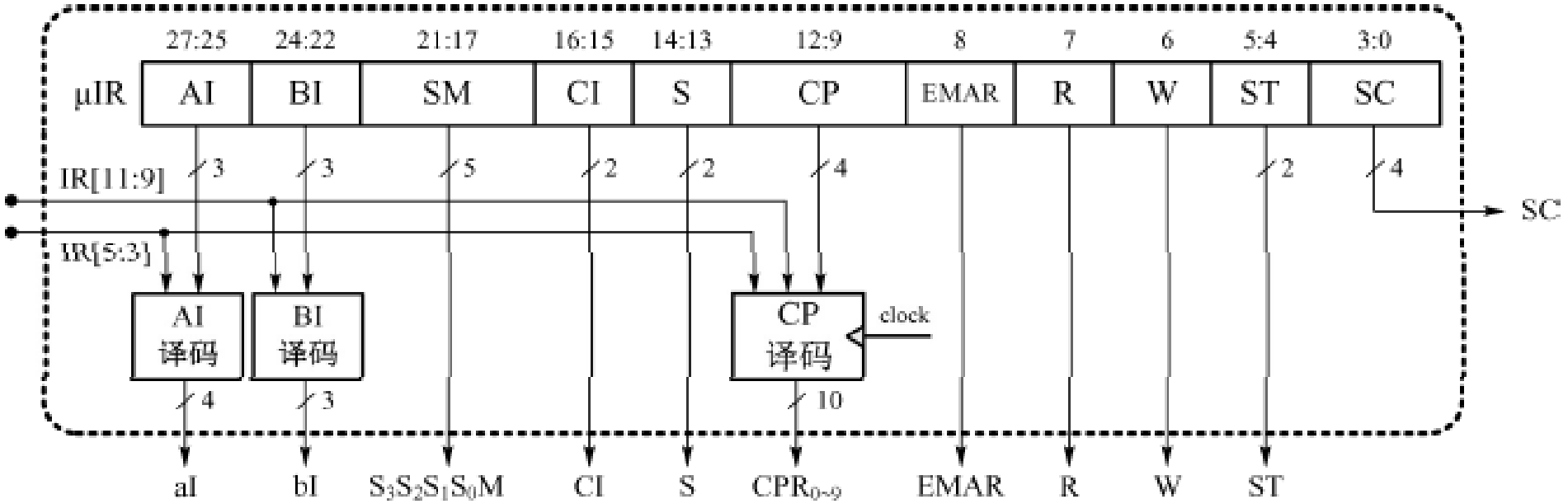


图 3-68　微命令二级译码电路的基本方案

微程序控制方式的优点

（1）用规整的存储逻辑结构代替了不规整的、复杂的硬连接逻辑，结构得到了简化，有利于设计自动化

（2）容易修改和扩展、灵活、通用性强

（3）可靠性较高，易于诊断和维护

（4）涉及访存，故速度慢，CPI会变大。

（5）性价比高。

微程序控制方式主要用于速度要求不高、功能较复杂的机器中，特别适合于系列机。

控制器设计小结

1、建立CPU整机概念 < 组成和结构
工作机制

(1) 逻辑组成

寄存器、ALU设置, 数据通路结构

(2) 工作机制

指令的执行过程:

{ 寄存器传送级: 各类指令的流程
微操作控制级: 微命令序列

- 拟定流程的关键: 熟练掌握数据通路结构

熟练掌握模型机寻址方式

2、基于模型机的控制器设计

(1) 组合逻辑设计

以下面四个方面为基础:

- 寄存器、ALU设置、数据通路结构
- 时序安排
- 各类指令的寻址方式和执行流程
- 微操作控制级 - 微命令序列

综合微操作信号、写出产生每一微操作信息的逻辑表达式, 简化表达式并用逻辑电路实现。

(2) 微程序设计

以下面四个方面为基础:

- 寄存器、ALU设置, 数据通路结构
- 时序安排
- 各类指令的寻址方式和执行流程
- 微操作控制级 - 微命令序列
  - 拟定微命令的格式以及编码方式;
  - 确定后续微地址形成方式;
  - 对所有指令执行过程所需微操作进行编码(包括后续地址或后继地址形成方式), 形成微指令。

3、有关的基本概念

(1) 微命令的产生方式

- 组合逻辑控制方式:基本思想 、优缺点、应用场合
- 微程序控制方式: 基本思想 、优缺点、应用场合

(2) 时序控制方式

同步控制方式: 定义、特点、应用场合

异步控制方式: 定义、特点、应用场合

同步控制的变形(半同步): 定义、特点、应用场合

综合应用题

综合应用题(本题共 20 分)

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。题七图是该 CPU 内部数据通路图。

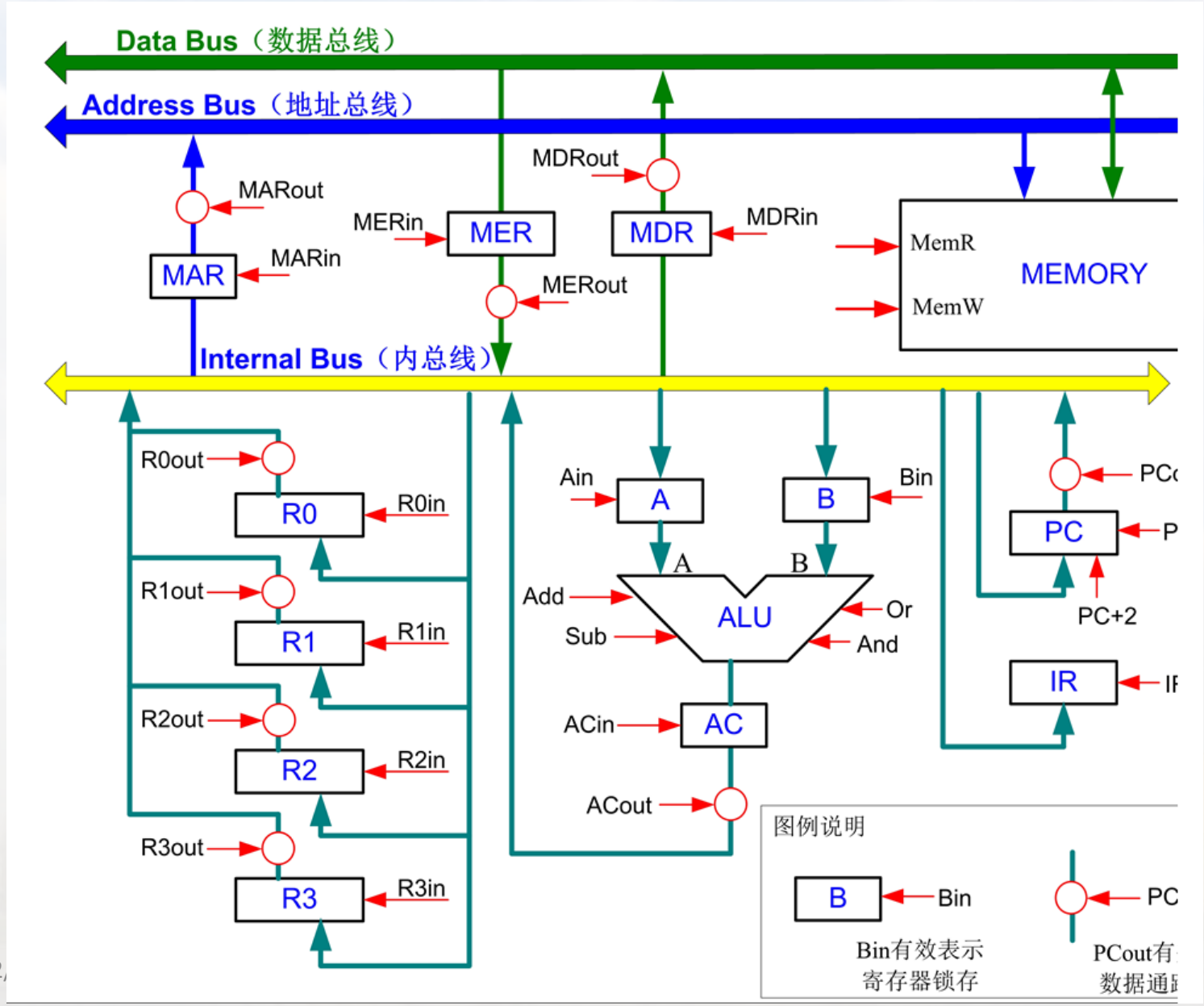
加法运算指令 *ADD R1，1000H(R2)*。其中源操作数 1000H(R2)是变址寻址，目的操作数 R1 是寄存器直接寻址，指令编码长度 32 位，指令编码格式如下：

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）。

综合应用题


Data Bus（数据总线）
Address Bus（地址总线）
Internal Bus（内总线）
MARout
MARin
MAR
MERin
MER
MEROut
MDROut
MDR
MDRin
MemR
MemW
MEMORY
R0out
R0in
R0
R1out
R1in
R1
R2out
R2in
R2
R3out
R3in
R3
Ain
A
B
Bin
Add
Sub
ALU
Or
And
ACin
AC
ACout
PC
PC+2
IR
图例说明
Bin有效表示
寄存器锁存
PCout有
数据通

综合应用题

	时钟周期	微操作（功能）	控制信号
取指令	T1	指令地址送 MAR （PC→MAR）	PCout，MARin
	T2	读指令送 MER （M →MER）	MARout，MemR，MERin
	T3	PC 调整，指令送 IR，译码 （PC + 2→PC，MER→IR）	PC+2；MERout，IRin
取位移量	T4		
	T5		
计算有效地址	T6		
读取源操作数	T7		
	T8		
执行指令	T9		
	T10		

谢谢观看

计算机组成原理

2022/10/26

信息与软件工程学院
School of Information and Software Engineering