第四章　控制器

第四章　控制器

- 4.1 控制器的组成及指令的执行
- 4.2 控制方式和时序的产生
- 4.3 微程序控制器
- 4.4 微程序控制器及其微程序设计举例
- 4.5 硬布线控制器

4.1 控制器的组成及指令的执行

- 一、控制器的功能
- 二、控制器的组成
- 三、指令周期
- 四、指令的执行过程

一、控制器的功能

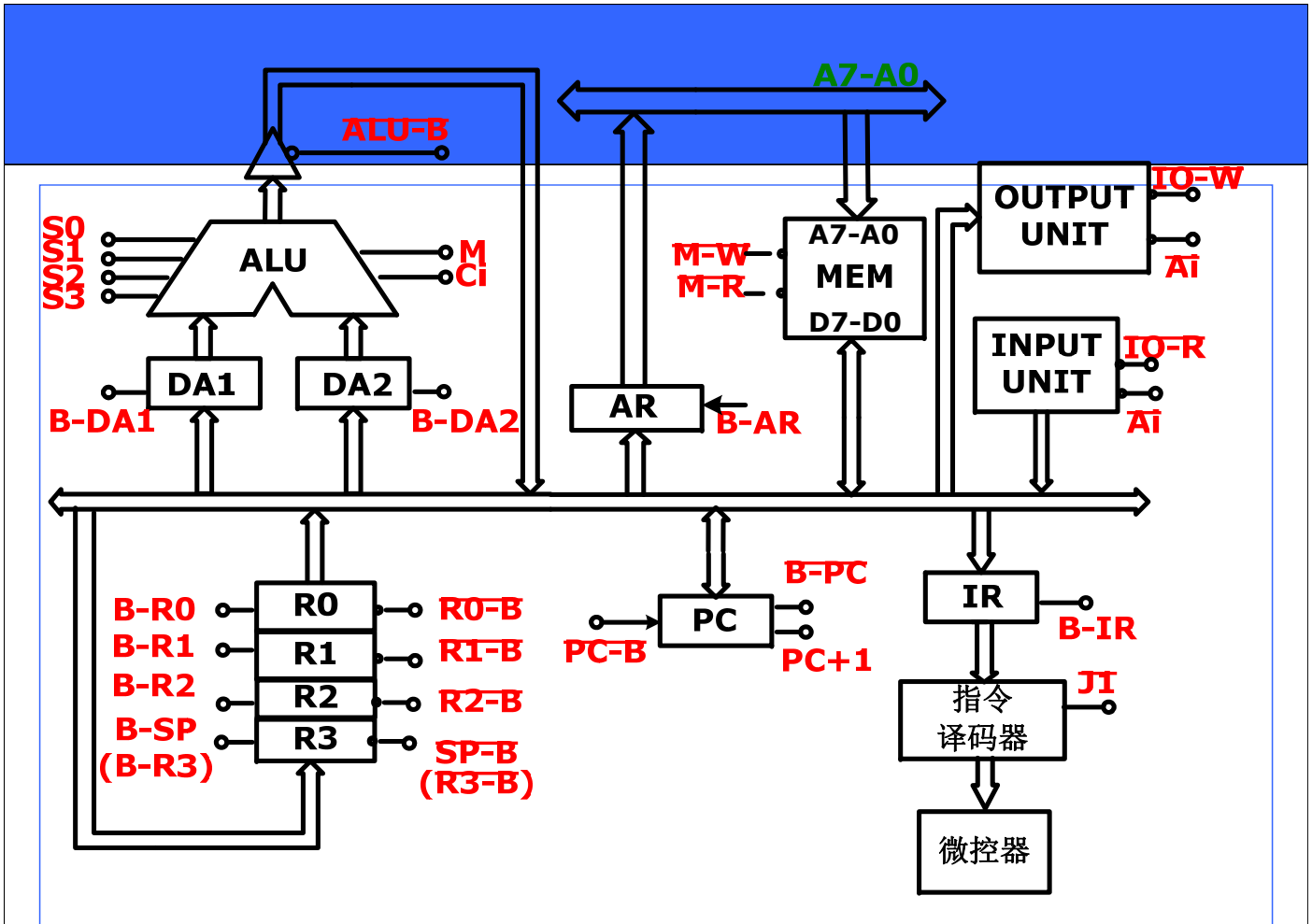
- 1、取指令：从内存取出指令（码）送CPU。
- 2、分析指令：对指令码进行分析译码，判断其功能、操作数寻址方式等。
- 3、执行指令：根据指令分析的结果，执行计算操作数地址、取操作数、运算等操作。
- 4、中断处理和响应特殊请求。
- 计算机工作的过程，就是循环往复的取指令、分析指令、执行指令的过程。

二、控制器的组成

- 1．程序计数器（PC）：存放指令的地址（当前指令或者下一条指令地址）；
  - 当指令顺序执行时，由PC+1产生下一条指令的地址；
  - 当遇到转移指令时，转移地址→PC作为下一条指令的地址。
- 2．指令寄存器（IR）：存放当前指令的指令码
- 3．指令译码器：对指令寄存器中的指令操作码字段进行译码。
  - 译码器的输出信号送入操作控制信号形成部件，产生该指令所需要的有一定时序关系的操作控制信号序列

二、控制器的组成

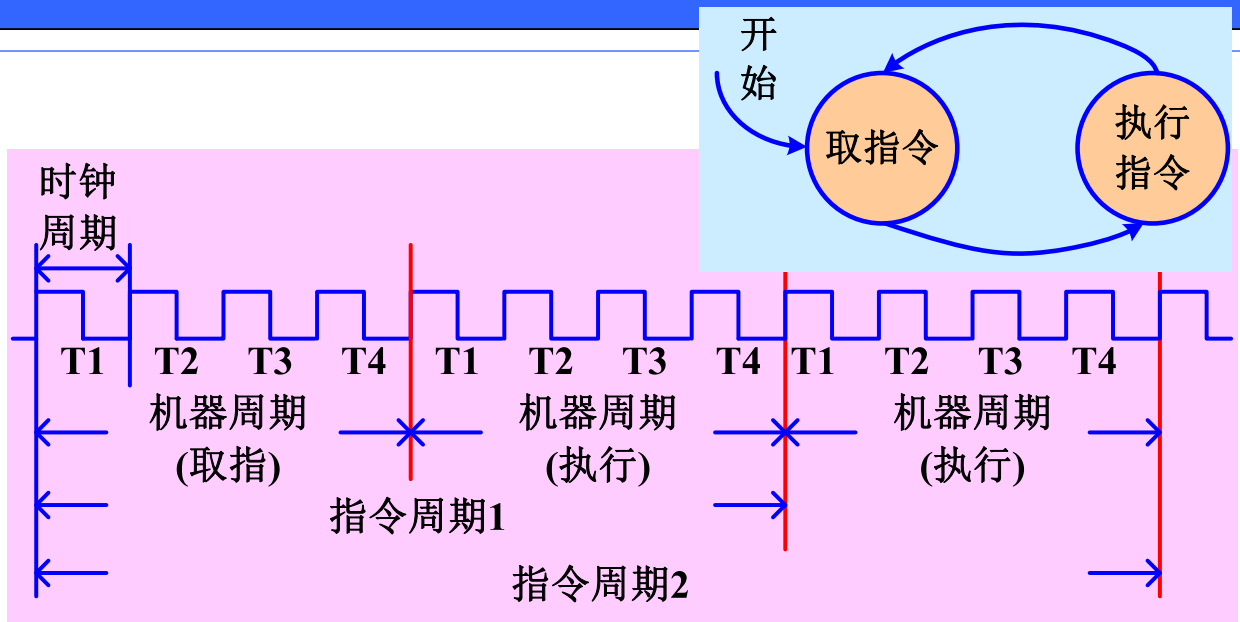
- **4．时序信号产生器：**负责提供时钟信号和机器周期信号，以规定每个操作的时间。
  - 启停线路，负责控制时钟脉冲的送出与封锁，从而实现计算机的启动与停止。
- **5．操作控制信号形成部件：**根据指令的操作码以及时序信号，产生取出指令和执行这条指令所需的各种操作控制信号，以便正确地建立数据通路，完成取出指令和执行指令的控制。
  - 操作控制信号形成部件采用组合逻辑电路的控制器，称作硬布线控制器；采用存储逻辑的称作微程序控制器。


A7-A0
ALU-B
S0
S1
S2
S3
ALU
M
Ci
DA1
DA2
B-DA1
B-DA2
AR
B-AR
M-W
M-R
A7-A0
MEM
D7-D0
OUTPUT
UNIT
IO-W
Ai
INPUT
UNIT
IO-R
Ai
B-R0
B-R1
B-R2
B-SP
(B-R3)
R0
R1
R2
R3
R0-B
R1-B
R2-B
SP-B
(R3-B)
PC-B
PC
B-PC
PC+1
IR
B-IR
JI
指令
译码器
微控器

三、指令周期

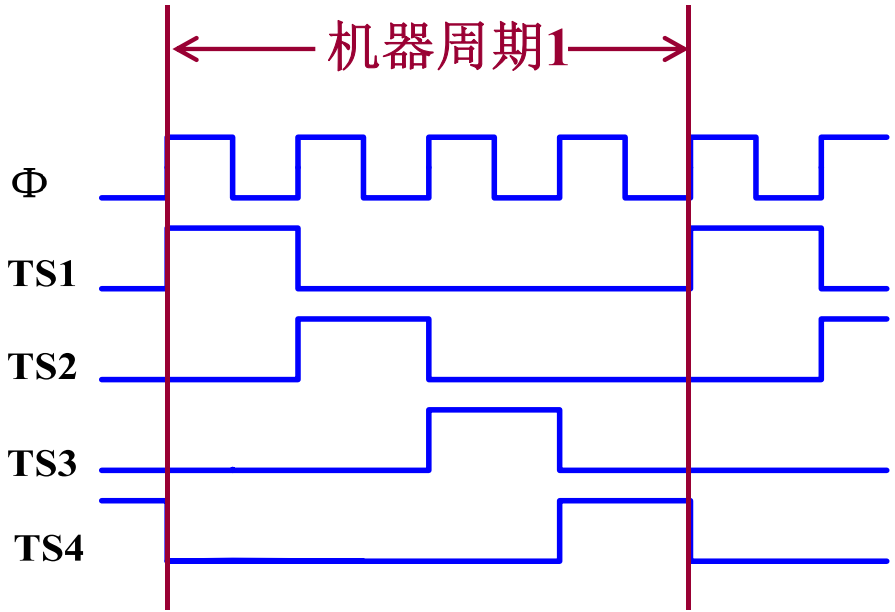
- **指令周期**：是指计算机从内存取出一条指令并完成该指令的执行所需要的时间。
  - 不同指令的指令周期是不相同的。
  - 一个指令周期可能由若干个机器周期组成。
- **机器周期**：又称为CPU周期，用于完成1次内存的操作（读或写访问）或者1次ALU的运算，或者1次总线传送
  - 一般规定为CPU与内存交换1次信息（读或写内存）所需要的时间。
  - 一个机器周期的功能需要多个时钟周期完成。
- **时钟周期**：又称为节拍周期，是指CPU执行一个微操作命令（即控制信号）的最小时间单位，也即T周期。

指令周期、机器周期、时钟周期的关系



节拍信号TS1~TS4和时钟信号源Φ的关系

- 下例中，每四个节拍信号构成一个机器周期。



四、指令的执行过程

- （一）指令执行过程概述
- （二）典型指令的执行过程
- （三）计算机的工作过程

（一）指令执行过程概述

- 一条指令的执行过程包括取指令、执行指令两大阶段：
- 1、取指令
  - （1）送指令地址：当前指令的地址由程序计数器PC指出，PC的内容送到地址寄存器AR，同时PC的内容递增以指向下一条指令的地址；即PC→AR, PC+1
  - （2）读取指令：AR的输出通过地址总线送到存储器的地址端，指明指令所在的地址单元，控制器发出读控制信号，控制从存储器中读出这条指令；该指令通过数据总线送到指令寄存器IR；即RAM→ IR

（一）指令执行过程概述

- （3）指令译码：由指令译码器对IR中的指令其进行分析译码；指令译码器首先判断该指令是什么指令，然后将判断结果信息传递给操作控制信号形成部件；即J1# 。

2、执行指令

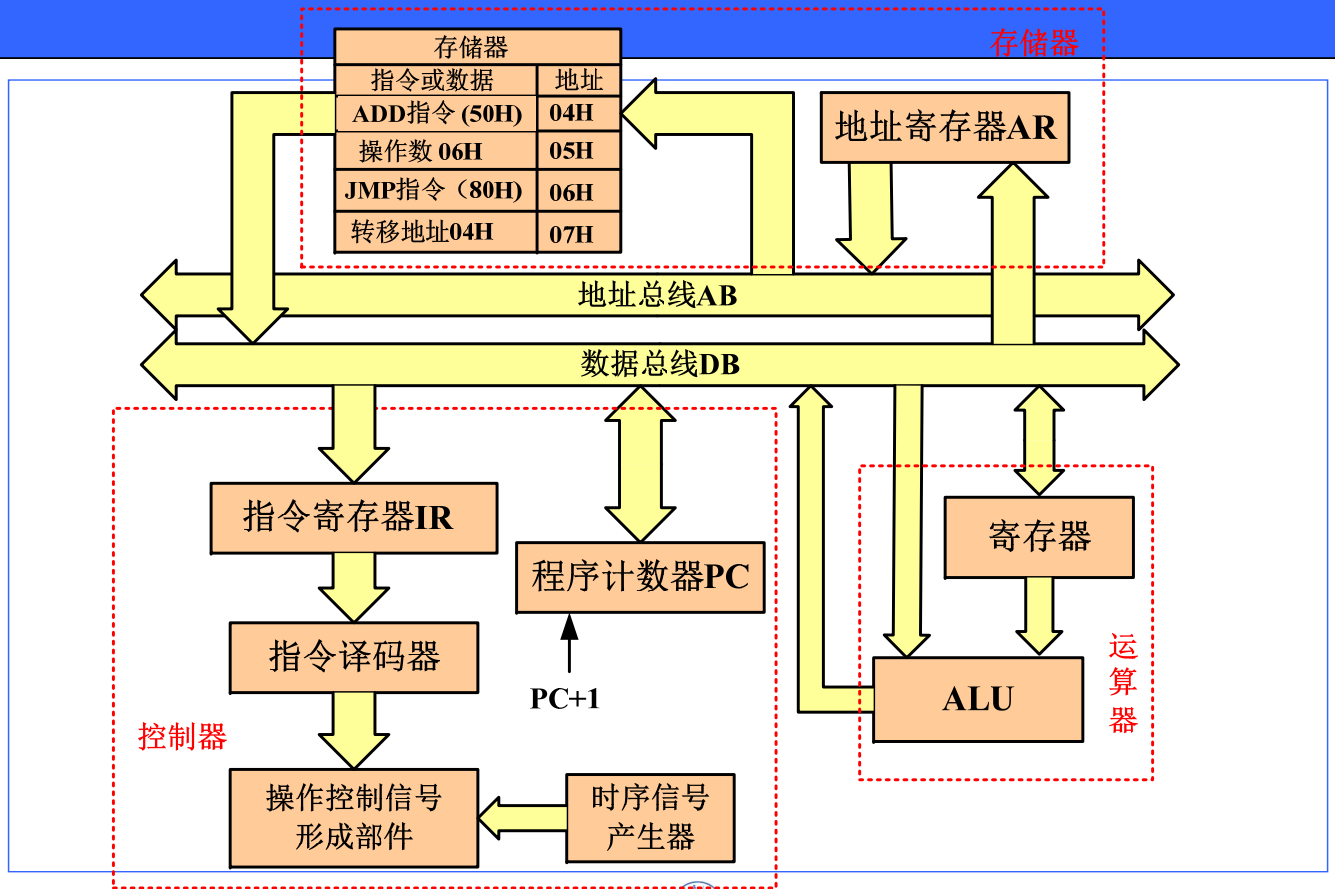
- 操作控制信号形成部件根据指令译码信息和时序周期信号，发出该指令所需的所有部件的有一定时序关系的控制信号序列，完成指令的执行。
- 执行指令的具体操作与指令的功能有很大的关系，不同的指令，其执行指令阶段也是不同的。

（二）典型指令的执行过程

- 指令的执行过程举例：假设存放在存储器中的二条指令内容为：

地址	机器码	助记符	功能
04H	0101 0000	ADD R_0, 06H	(R_0)+06H→R_0
05H	0000 0110（立即数）		
06H	1000 0000	JMP 04H	04H→PC
07H	0000 0100（转移地址）		

模型计算机的系统结构


存储器
指令或数据
地址
ADD指令 (50H)
04H
操作数 06H
05H
JMP指令（80H)
06H
转移地址04H
07H
存储器
地址寄存器AR
地址总线AB
数据总线DB
指令寄存器IR
程序计数器PC
指令译码器
PC+1
控制器
操作控制信号
形成部件
时序信号
产生器
寄存器
ALU
运
算
器

典型指令的指令周期

- 1、ADD Rd, Data; (Rd)+Data→Rd
  - 加法指令：寄存器＋立即数存入寄存器
  - 寻址方式：源操作数为立即数寻址，目的操作数为寄存器（直接）寻址
  - 指令格式：

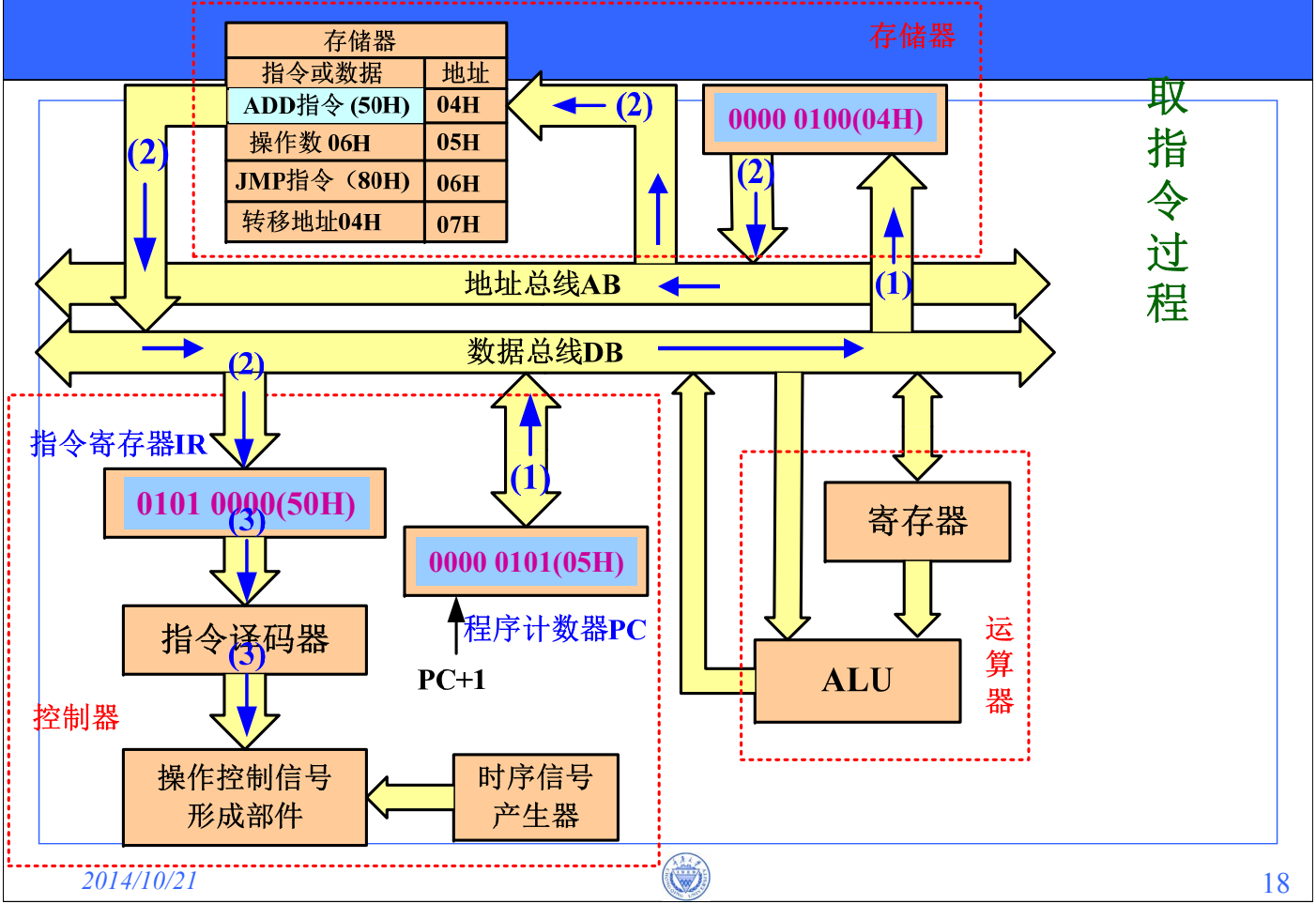
OP(4)	××	Rd(2)
Data		

  - 执行过程：

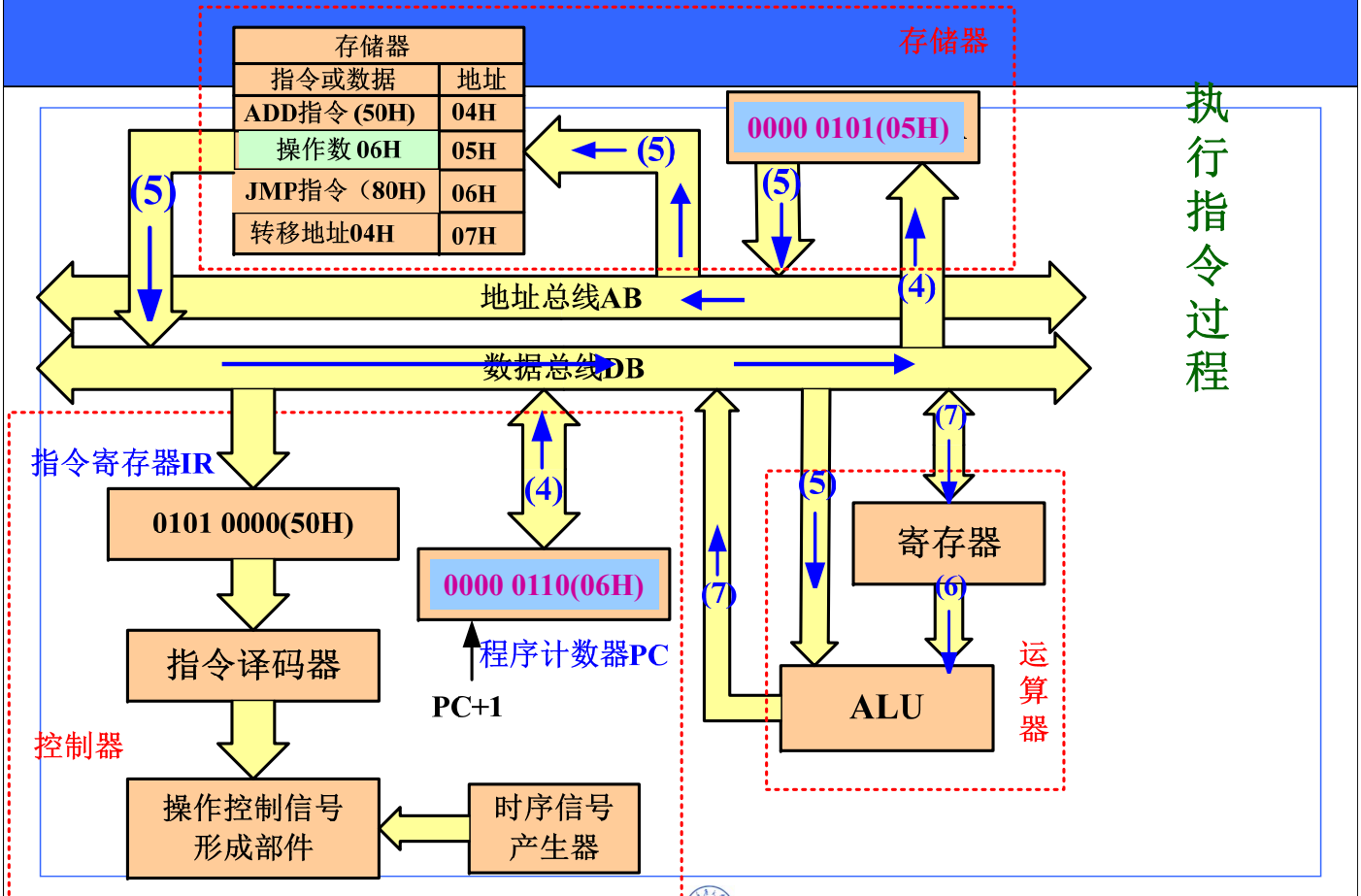
ADD Rd, Data指令的执行过程

- 取指令：
  - M1 (送存储器地址)：　　PC→AR, PC+1
  - M2 (读存储器)：　RAM→ IR
  - M3(指令译码）：　　J1#
- 执行指令：
  - M4（取源操作数－送地址）：PC→AR, PC+1
  - M5（取源操作数－读）：RAM→ DA1
  - M6（取目的操作数）：　Rd→DA2
  - M7（计算并置结果）：DA1+DA2→Rd

ADD R0, 06H；(R0)+06H→R0指令的运行过程



ADD R0, 06H；(R0)+06H→R0指令的运行过程



典型指令的指令周期

2、JMP ADDR; ADDR→PC

- 跳转指令：从当前指令跳转到目标处执行
- 寻址方式：单操作数指令，操作数为直接转移地址，直接寻址
- 指令格式：

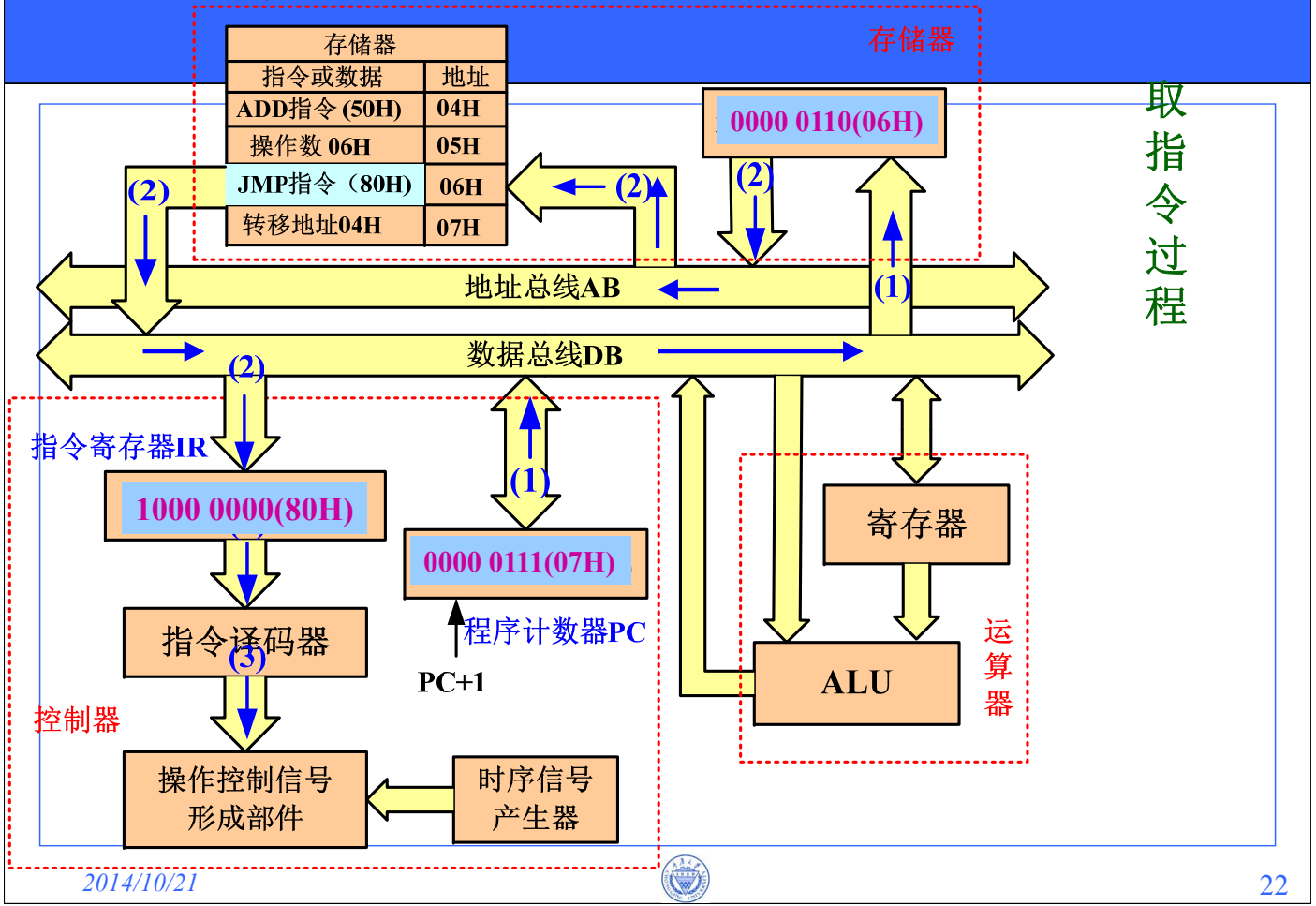
OP(4)	×× ××
ADDR	

- 执行过程：

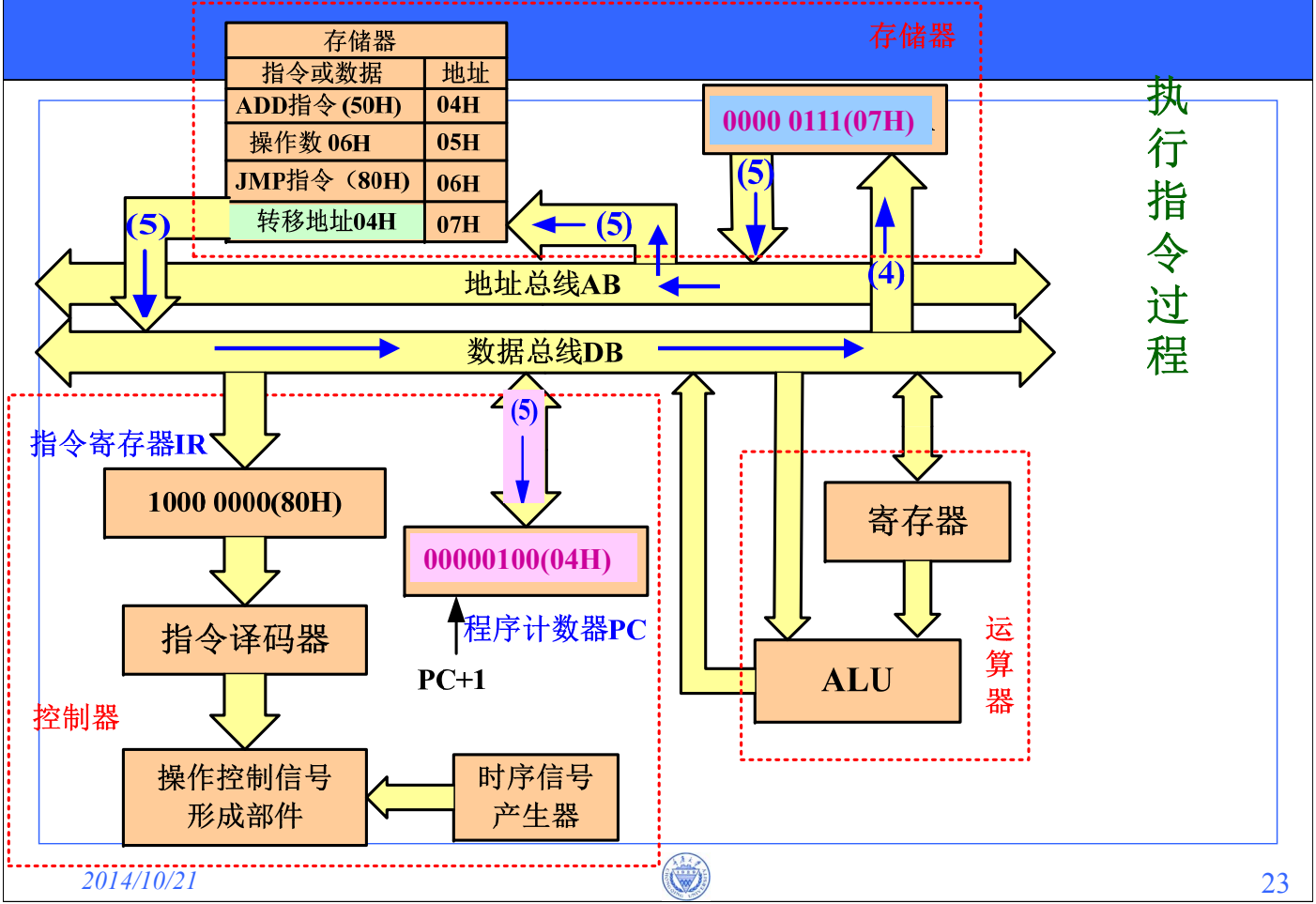
JMP ADDR指令的执行过程

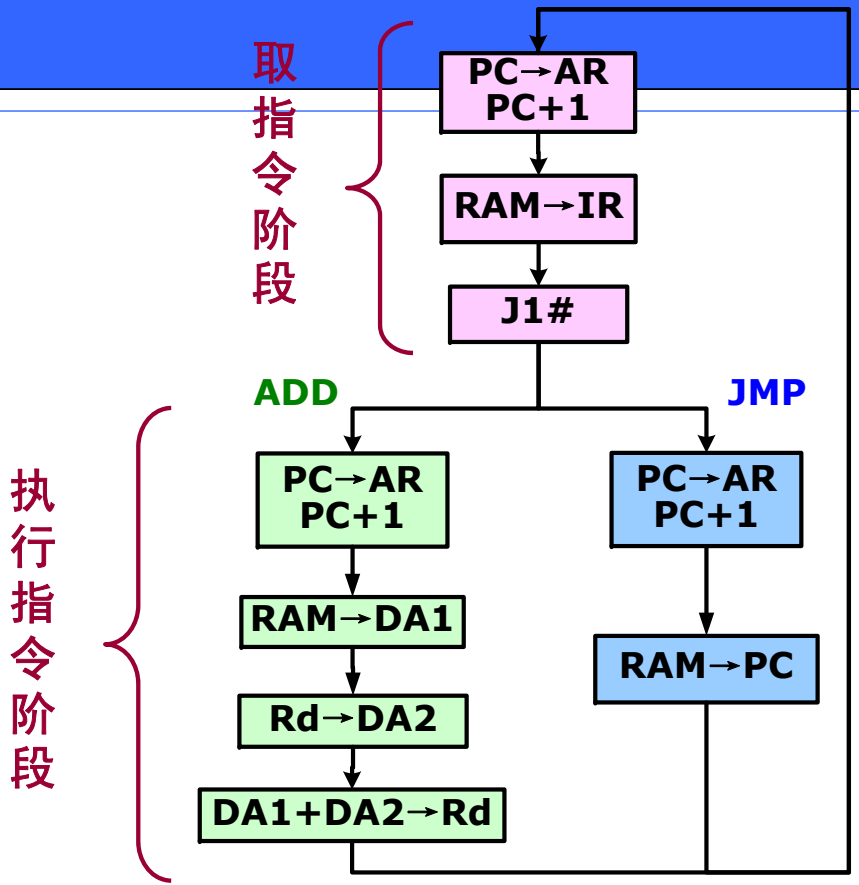
- 取指令：
  - M1 (送存储器地址)：PC→AR, PC+1
  - M2 (读存储器)：　　RAM→IR
  - M3(指令译码）：　　J1#
- 执行指令：
  - M4（取操作数－送地址）：PC→AR, PC+1
  - M5（取操作数－读）：RAM→ PC

JMP 04H；04H→PC指令的运行过程



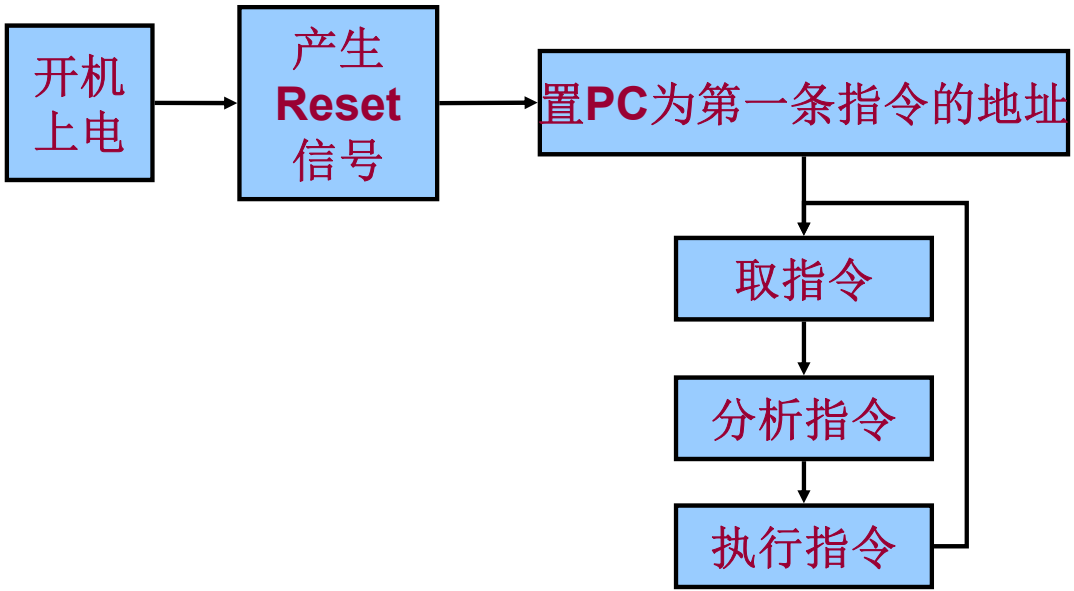
JMP 04H；04H→PC指令的运行过程




取指令阶段
PC→AR
PC+1
RAM→IR
J1#
ADD
JMP
执行指令阶段
PC→AR
PC+1
RAM→DA1
Rd→DA2
DA1+DA2→Rd
PC→AR
PC+1
RAM→PC

（三）计算机的工作过程

- 计算机的工作过程即是循环往复的取指令、分析指令、执行指令的过程。



4.2 控制方式和时序的产生

- 一、控制方式
- 二、时序脉冲发生器和启停控制

一、控制方式

- 控制方式定义：形成控制微操作控制信号序列的时序控制信号的方法。
- 可以分为三种：
- 1、同步控制方式
- 2、异步控制方式
- 3、联合控制方式

1、同步控制方式

- 以微操作序列最长的指令为标准，确定控制微操作运行的时钟周期数（节拍数）。
- 控制器产生统一的、顺序固定的、周而复始的节拍电位（机器周期信号）和节拍脉冲（时钟周期信号）；
- 微操作序列短的指令可空着几个节拍不用。也就是说，采用相同的机器周期数和相同的节拍脉冲来形成每条指令的操作控制信号序列，因此每条指令的执行所用的时间都是相同的。
- 同步控制方式的优点是电路简单，缺点是运行速度慢。

2、异步控制方式

- 每条指令需要多少节拍，就产生多少节拍；当指令执行完毕，发出回答信号；控制器收到回答信号时，才开始下条指令的执行。
- 执行不同指令所需的时间完全由实际需要确定，不尽相同。
- 每个机器周期内的节拍数可以不一样，通常把大多数微操作安排在一个较短的机器周期内完成，而对某些复杂的微操作，采用延长机器周期或增加节拍数的办法来解决。
- 异步控制方式的优点是运行速度快，其缺点是控制电路比较复杂。

2、异步控制方式

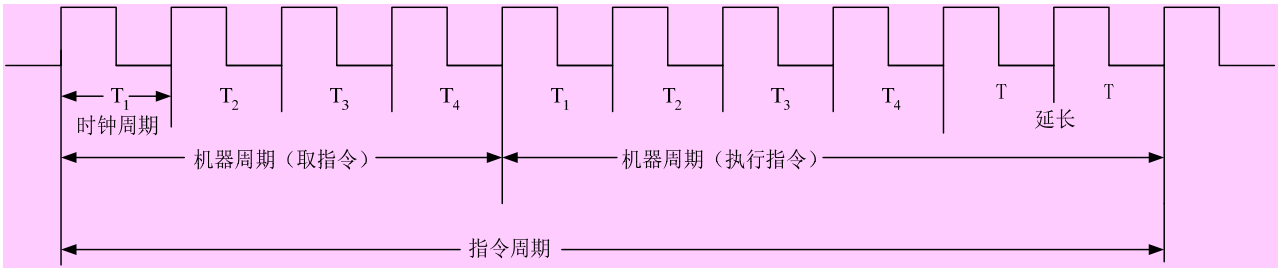


图7—8 延长机器周期的时序图

3、联合控制方式

- 把同步控制方式和异步控制方式结合使用。
- 大部分指令安排在统一的机器周期内完成，即同步控制；而将较少数特殊指令，或微操作序列过长或过短，或微操作时间难以确定的，采用异步控制来完成。
- 联合控制方式的优点是能保证一定的运行速度，其缺点是控制电路设计相对比较复杂。

二、时序脉冲发生器和启停控制

- 时序脉冲发生器就是根据时钟产生一定频率的节拍脉冲信号作为整个机器工作的时序信号；
- 启停控制电路是保证在适当的时刻准确可靠地开启或封锁计算机工作时钟，以控制微操作命令序列的产生或停止，从而启动或停止计算机的运行。
- 通常用访问一次主存取指或取数据的时间来作为机器周期的基本时间。
- 机器周期确定后，每一机器周期的节拍与时钟数、机器的主频也就基本确定了。
- 控制器的时钟输入实际上是节拍脉冲序列，其频率即为机器的主频。

第四章　控制器

- 4.1 控制器的组成及指令的执行
- 4.2 控制方式和时序的产生
- 4.3 微程序控制器
- 4.4 微程序控制器及其微程序设计举例
- 4.5 硬布线控制器

4.3 微程序控制器

- 一、基本概念
- 二、微程序控制器的基本工作原理
- 三、微程序控制器的组成
- 四、微程序控制原理举例
- 五、微程序设计技术

一、 基本概念

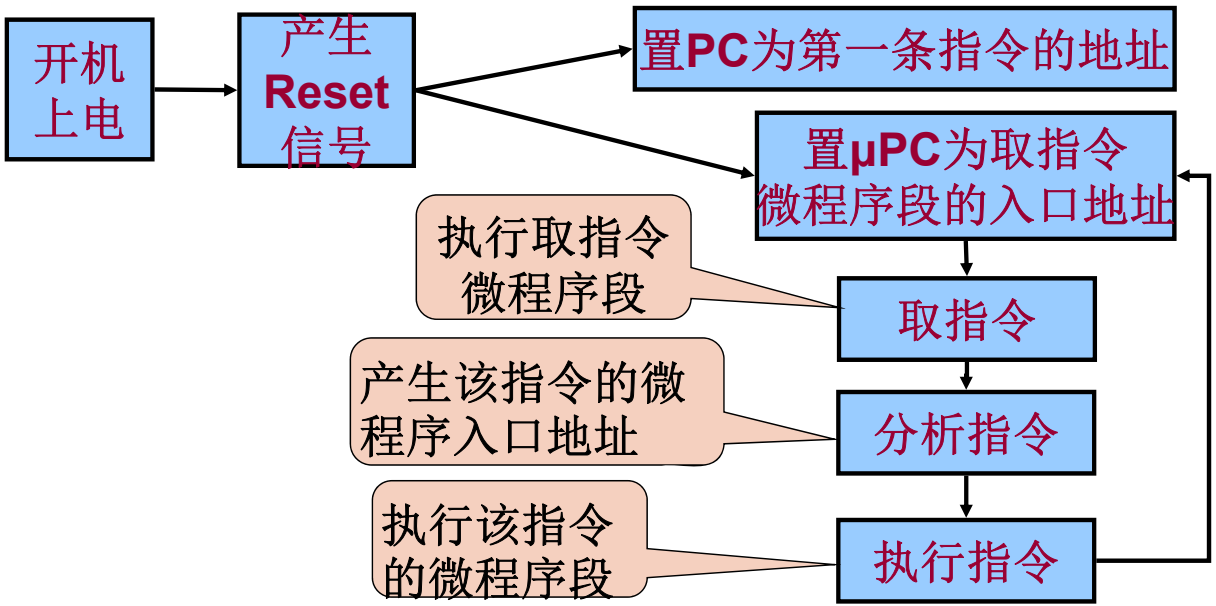
1. 微操作：指令执行时必须完成的基本操作。例如，PC→AR，PC+1→ PC，RAM→IR。
2. 微命令：是组成微指令的最小单位，也就是控制微操作实现的控制信号。一般用于控制数据通路上门的打开/关闭，或者功能选择。
3. 微指令：是一组微命令的集合，用于完成一个功能相对完整的操作。
4. 微程序：微指令的有序集合，用于实现机器指令的功能。

一、 基本概念

5. **控制存储器：**简称控存，用于存放所有指令的微程序，其中一个存储单元存放一条微指令。一般为ROM。
6. **微地址：**微指令在控存中的地址。
7. **微周期：**指从控存中取出并执行一条微指令所需要的时间，一般与一个机器周期相当。

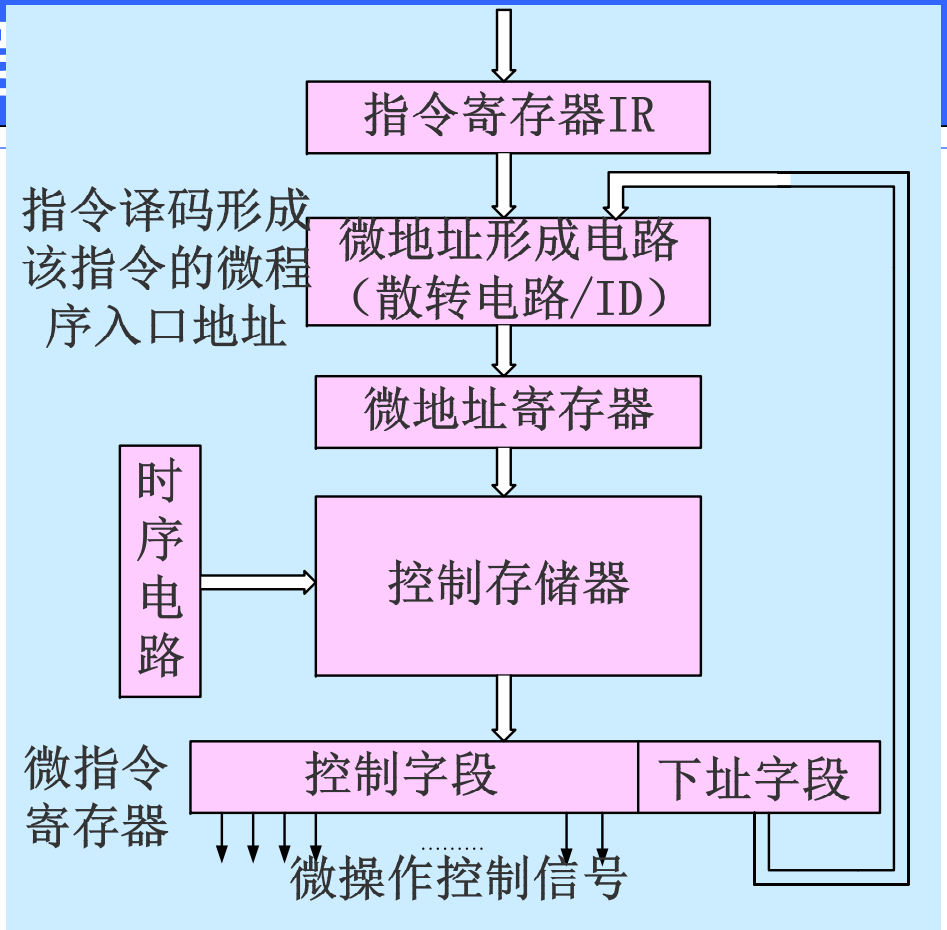
二、微程序控制器的基本工作原理

- 一条机器指令由一段微程序来解释实现 。
- 微程序控制的计算机工作过程：



三、微程

微程序控制器的组成框图

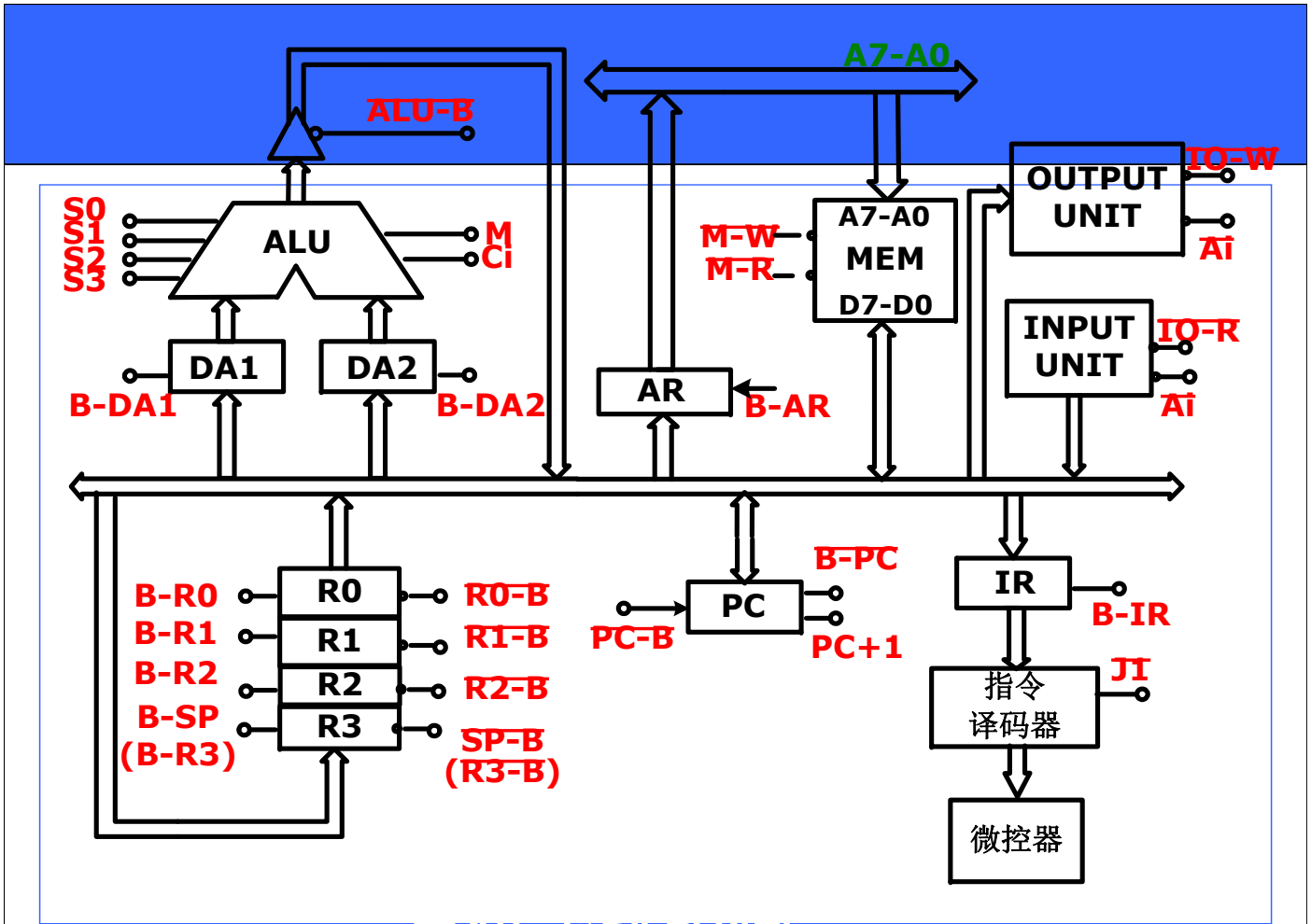


微程序控制器的构成部件

1. 控制存储器:简称控存、CM，用于存放微程序，一般由ROM构成。
2. 微地址寄存器:存放要访问的控存中的微指令的地址，又称μAR 、CMAR。
3. 微指令寄存器:存放从控存中读出的微指令本身，又称μIR。其控制字段用于产生微操作控制信号，其下址则送至微地址形成电路，产生下一条微指令的地址。
4. 微地址形成电路:用于产生下一条微指令的地址。包含了指令译码器。
5. 微程序控制器中ID的作用是将指令寄存器中的操作码OP转换成该指令的微程序入口地址。

四、微程序控制原理举例

- （一）模型计算机系统结构
- （二）模型计算机数据通路
- （三）模型计算机控制信号
- （四）微指令格式
- （五）微程序设计举例


A7-A0
ALU-B
IO-W
OUTPUT UNIT
Ai
S0
S1
S2
S3
ALU
M
Ci
M-W
M-R
A7-A0
MEM
D7-D0
INPUT UNIT
IO-R
Ai
DA1
DA2
B-DA1
B-DA2
AR
B-AR
B-R0
B-R1
B-R2
B-SP
(B-R3)
R0
R1
R2
R3
R0-B
R1-B
R2-B
SP-B
(R3-B)
PC-B
PC
B-PC
PC+1
IR
B-IR
JI
指令
译码器
微控器

（二）模型计算机数据通路

- 1、存储器读操作：分成两步：
  - 送地址到总线，并打入地址寄存器AR；
  - 发送存储器读信号M-R# =0，启动存储器读操作，并将读出的数据从总线上接收至目的部件（例如某通用寄存器或者暂存器DA1、DA2）。
- 例如：取指令操作
  - PC→AR，PC+1；
  - 发送M-R# =0 ，并RAM→IR。

（二）模型计算机数据通路

- **2、存储器写操作：分成两步：**
  - 送地址到总线，并打入地址寄存器AR；
  - 送数据到总线，并发送存储器写信号M-W# =0，启动存储器写操作。
- **3、运算器的运算操作：分成三步：**
  - 送第一个数据到总线，并打入ALU暂存器DA1/DA2
  - 送第二个数据到总线，且打入ALU暂存器DA2/DA1
  - 发送运算器功能选择信号S3～S0、M、Ci，控制ALU进行某种运算，并打开ALU输出三态门(ALU-B#=0)，将总线上运算结果送目的部件。

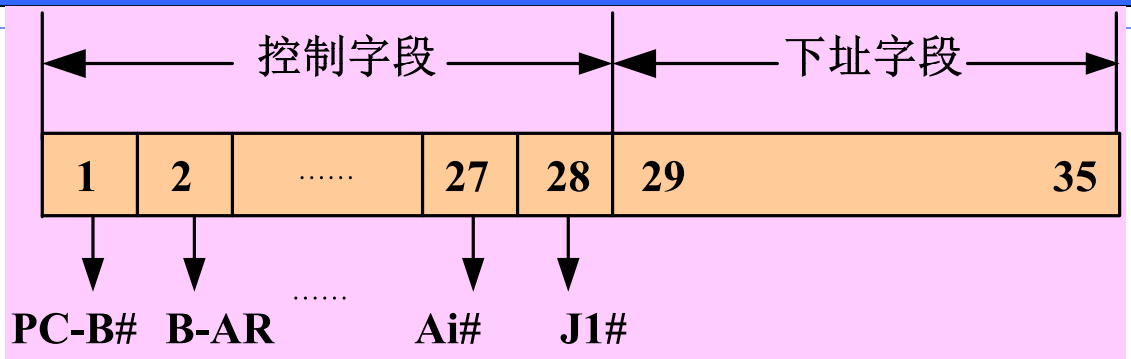
（三）模型计算机控制信号

序号	控制信号	功能	序号	控制信号	功能
1	PC-B#	指令地址送总线	8	S3	S3- S0选择ALU16种运算之一
2	B-AR	总线数据打入AR	9	S2	
3	PC+1	程序计数器+1	10	S1	
4	B-PC	总线数据打入PC	11	S0	
5	B-IR	总线数据打入IR	12	M	选择逻辑运算(1)和算数运算(0)
6	M-W #	存储器写	13	B-DA1	总线数据打入暂存器DA1
7	M-R #	存储器读	14	B-DA2	总线数据打入暂存器DA2

（三）模型计算机控制信号

序号	控制信号	功能	序号	控制信号	功能
15	ALU-B #	运算器ALU内容送总线	22	R1-B #	R1内容送总线
16	Ci	ALU进位输入	23	R2-B #	R2内容送总线
17	B-R0	总线数据打入R0	24	R3-B #	R3内容送总线
18	B-R1	总线数据打入R1	25	I/O-W #	写（输出）I/O端口
19	B-R2	总线数据打入R2	26	I/O-R #	读（输入）I/O端口
20	B-R3	总线数据打入R3	27	Ai #	端口地址线
21	R0-B #	R0内容送总线	28	J1#	指令译码器译码

（四）微指令格式



- 微指令的控制字段28位，一位表示一个微命令。
- 微指令的下址字段指出下一条微指令的地址，该模型机的控制存储器地址是7位，表示最多有128个单元，每个单元（28+7=35位）。

（五）微程序设计举例

- 微程序设计步骤：

1) 根据数据通路，写出每条指令的执行过程，画出微程序流程图。
2) 写出每条微指令所发出的微操作控制信号。
3) 按照微指令格式，编写每条微指令的代码。
4) 对照指令的执行流程图，分配微指令的地址
5) 将写好的微指令按分配好的微地址装入控制存储器。

（五）微程序设计举例

- 假设存放在存储器中的二条指令内容为：

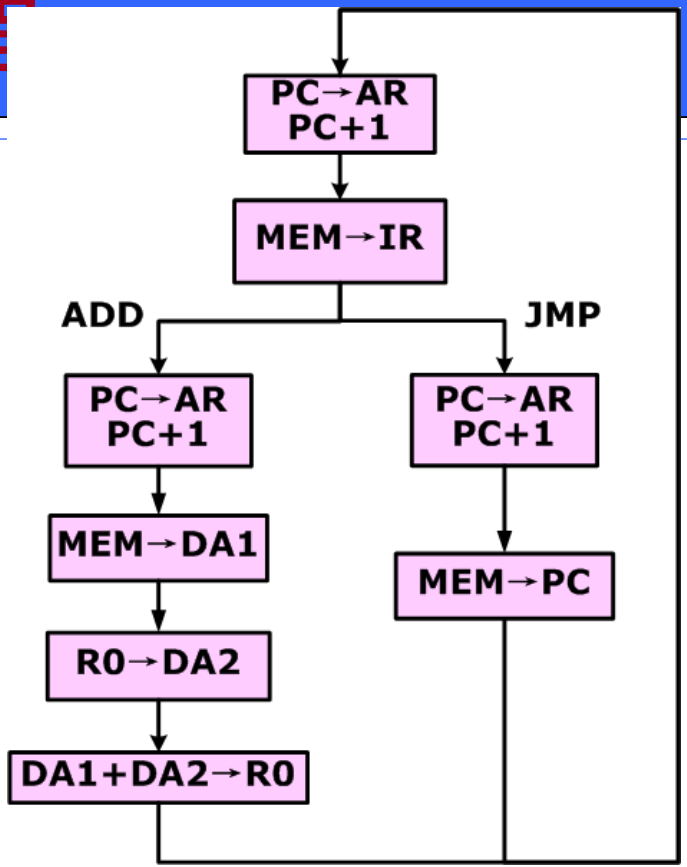
地址	机器码	助记符	功能
04H	0101 0000	ADD R_0, 06H	$(R_0)+06H \rightarrow R_0$
05H	0000 0110（立即数）		
06H	1000 0000	JMP 04H	04H→PC
07H	0000 0100（转移地址）		

1、指令执行过程

- ADD R_0, 06H
- 取指令：
  - M1 (送存储器地址)：PC→AR, PC+1
  - M2 (读存储器，指令译码)：　　MEM→ IR,J1#
- 执行指令：
  - M3（取源操作数－送地址）：PC→AR, PC+1
  - M4（取源操作数－读）：MEM→ DA1
  - M5（取目的操作数）： R0→DA2
  - M6（计算并置结果）：DA1+DA2→Rd

1、指令执行过程

- JMP ADDR
- 取指令：
  - M1 (送存储器地址)：PC→AR, PC+1
  - M2 (读存储器,指令译码)：RAM→IR, J1#
- 执行指令：
  - M4（取操作数—送地址）：PC→AR, PC+1
  - M5（取操作数—读）：RAM→ PC



2、指令执行时产生的微操作序列

	序号	功能	发送控制信号
取指令	（1）	PC→AR,PC+1	PC-B#,B-AR,PC+1,
	（2）	MEM→IR	M-R#,B-IR,J1#
ADD	（1）	PC→AR,PC+1	PC-B#,B-AR,PC+1,
	（2）	MEM→DA1	M-R#,B-DA1
	（3）	R0→DA2	R0-B#,B-DA2
	（4）	DA1+DA2→R0	S3~S0,M,Ci=100101,ALU-B#,B-R0
JMP	（1）	PC→AR,PC+1	PC-B#,B-AR,PC+1,
	（2）	MEM→PC	M-R#,B-PC#

3、指令的微程序代码：ADD

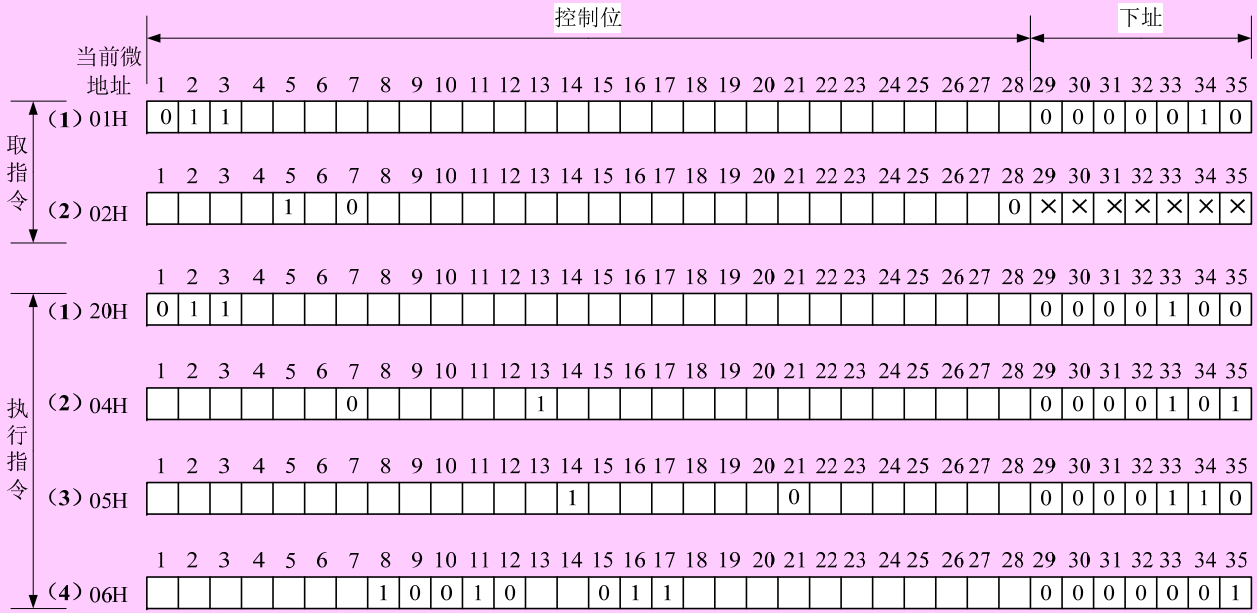


图7.13 ADD指令的微指令

3、指令的微程序代码：JMP

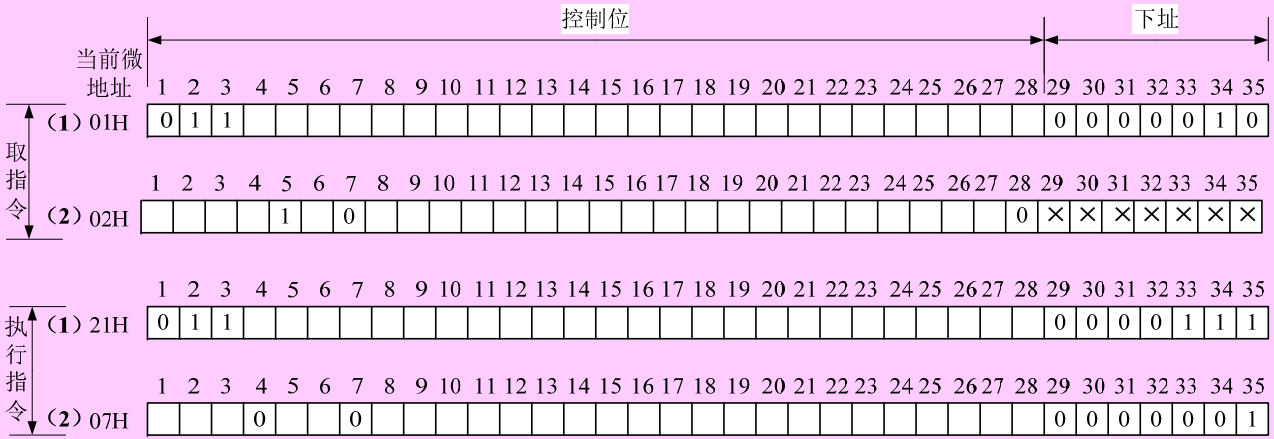
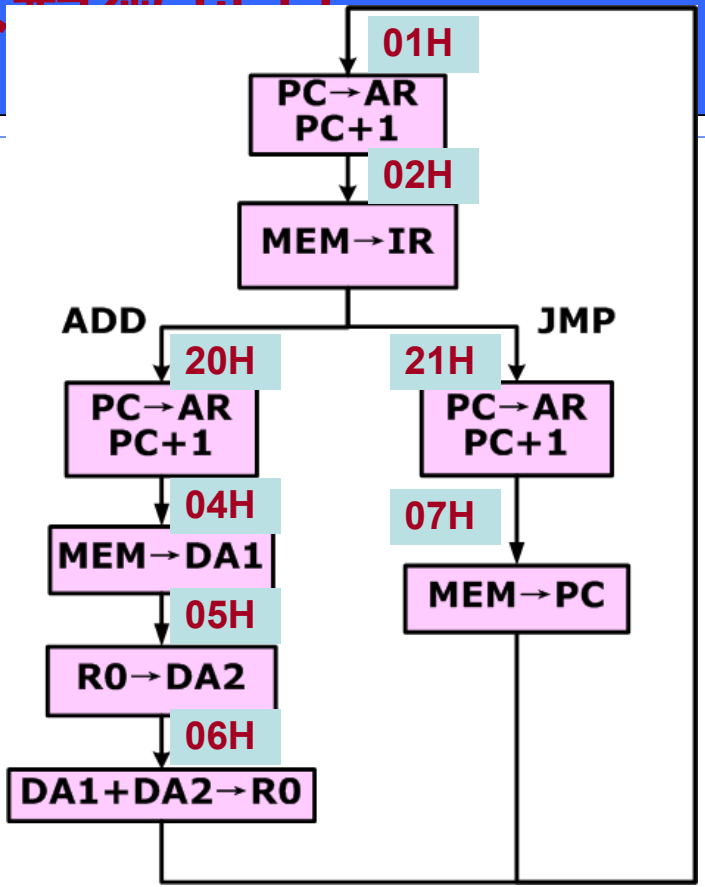


图7.14　JMP指令的微指令

4、为微指令分


01H
PC→AR
PC+1
02H
MEM→IR
ADD
JMP
20H
21H
PC→AR
PC+1
PC→AR
PC+1
04H
07H
MEM→DA1
MEM→PC
05H
R0→DA2
06H
DA1+DA2→R0

五、微程序设计技术

- 采用微程序设计的目的：
  - 缩短微指令字长；
  - 减少控制存储器的容量；
  - 微程序的执行速度；
  - 易于微指令的修改；
  - 增加微程序设计的灵活性。
- 微指令由两部分构成：
  - 控制字段：指出微指令的功能，表示要发送的微命令
  - 下址字段：用于指出下一条微指令的地址。

五、微程序设计技术

- （一）控制字段的编码方法
- （二）下址字段的设计方法
- （三）微指令格式的类型
- （四）控制存储器和动态微程序设计

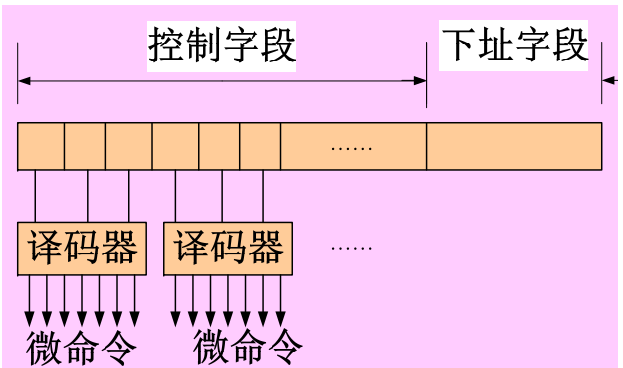
（一）控制字段的编码方法

- **1、直接控制法：**微指令的控制字段中，每一位代表一个微命令（控制信号），在设计微指令时，如果要发出某个微命令则将控制字段中对应位置有效信号，即打开对应控制门。
  - **优点：**无需译码，执行速度快；微程序较短。
  - **缺点：**微指令字长很长，占用控存容量大。
- **2、全译码方式：**将所有的控制信号进行编码，作为控制字段。在执行微指令时，译码产生各个微命令。
  - **优点：**微指令字长很短。
  - **缺点：**并行操作能力弱，微程序很长，执行速度慢

（一）控制字段的编码方法

- **3、字段直接编译法：**将控制字段分成若干段，每段通过编码/译码对应到各个控制信号。
- 优点：并行操作能力较强，字长较短
- 分段原则是：**相斥性微命令分在同一字段内，相容性微命令分在不同字段内**。前者可以提高信息位的利用率，缩短微指令字长；后者有利于实现并行操作，加快指令的执行速度。

  - **相斥性微命令：**指在同一个微周期中**不可能**同时出现的微命令。
  - **相容性微命令：**指在同一个微周期中可以同时出现的微命令



（一）控制字段的编码方法

- **4、字段间接编译法：**某字段的编码含意，除了其本身的编码外，还需要由另一字段来加以解释。也就是说，某一字段所产生的微命令，是和另一字段的代码联合定义出来的。
  - 优点：进一步缩短微指令字长

（二）下址字段的设计方法

- 微指令中下址字段的结构和后继微指令地址的形成方法有关，又称为微程序流的控制。
- 后继微指令地址获得方法有三种：
  a) 根据机器指令操作码产生该指令对应的微程序入口地址（通过指令译码散转）；
  b) 由下址字段指出下一微地址，或顺序＋1；
  c) 根据上一条微指令执行结果来判断微指令转移还是顺序执行微指令，即实现微程序分支。

（二）下址字段的设计方法

- 1、微程序入口地址的产生
- 2、后继微地址的产生

1、微程序入口地址的产生

- 根据机器指令的操作码转移到其对应的微程序入口地址，通常称为操作码映射。
- 这是一种多分支情况，也就是指令译码，通常采用以下两种方法来实现这种转移。
  - 映射ROM：简称MAPROM，该ROM中存放的是指令的微程序入口地址。以指令的操作码为地址，访问该MAPROM单元，读出的内容即是该指令的微程序入口地址。
  - 逻辑电路：以指令的操作码作为输入变量，通过组合逻辑电路，产生的输出变量即为指令的微程序入口地址。

2、后继微地址的产生

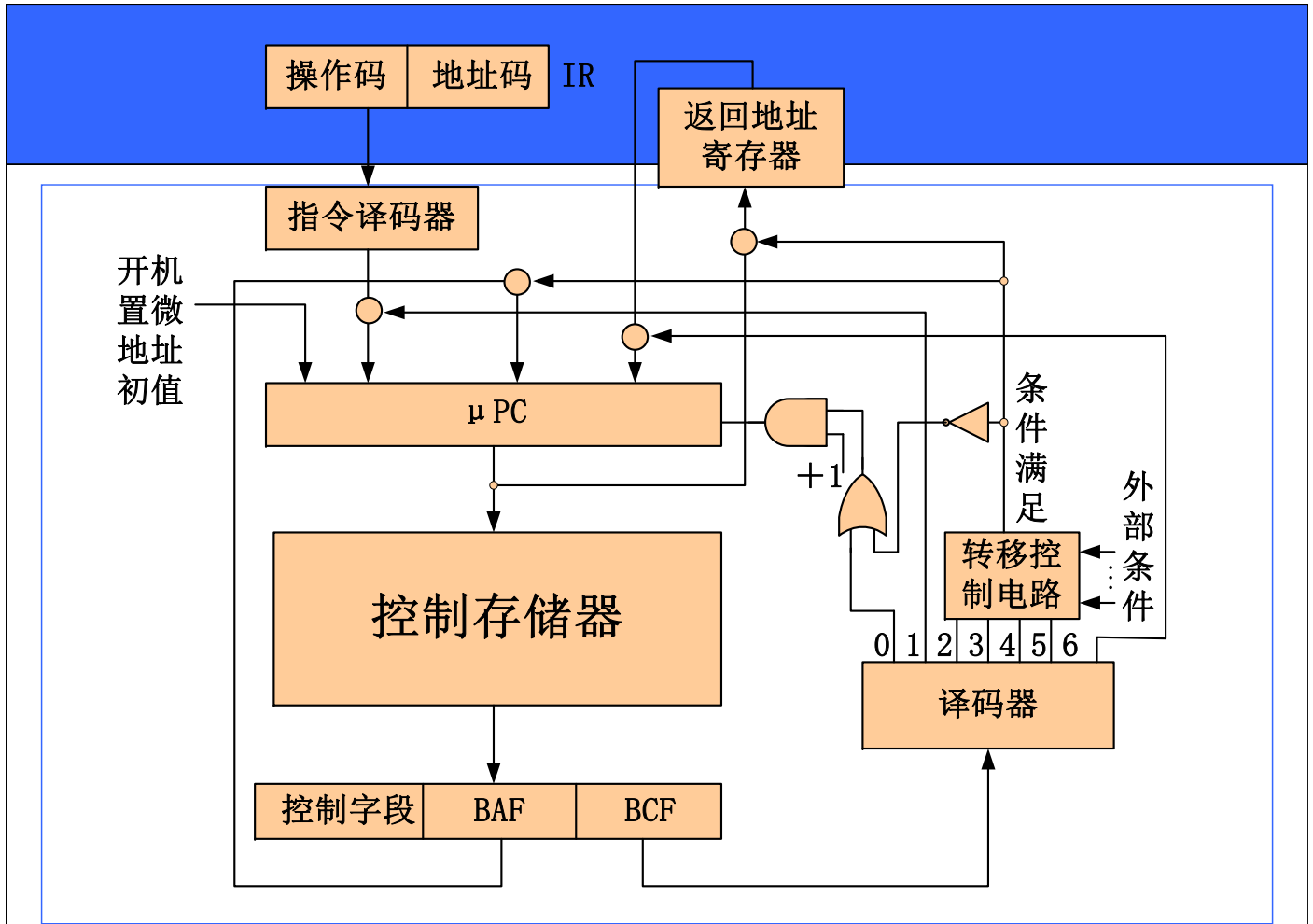
- 每条微指令执行完毕，都必须根据要求产生后继微指令地址。方法有两种：
- （1）计数器方式：在微程序控制器单元中设置一个微程序计数器μPC，用于保存后继微指令地址。
  - 在顺序执行微指令时，后继微指令地址由现行微地址1（即μPC ＋1）来产生。
  - 遇到转移时，由微指令给出转移地址，使微程序按新的微地址顺序执行。
  - 微指令格式：

控制字段	BAF	BCF

（1）计数器方式

- BAF：转移地址字段，用于给出微指令转移
  BAF一般位数少，将它送到μPC的若干低位
  地址。
- BCF：转移控制字段，用来确定后继微地址
  件转移。
  - 当条件成立时，微程序转移，将BAF送μPC
    一条微指令（μPC＋1）。
  - 转移控制字段BCF应当能够定义各种后继微
  - 假设，在微程序中有顺序执行、无条件转移
    循环、转微子程序和微子程序返回6种情况，
    相应的微程序入口地址，则转移控制字段B
    种情况。

- 优点是微指令字较短，便于编写微程序，后继微地址产生机构比较简单；
- 缺点是执行速度低，微程序在控制存储器中的物理分配不方便，需合理安派、调整


操作码
地址码
IR
返回地址
寄存器
指令译码器
开机
置微
地址
初值
μPC
+1
条
件
满
足
外
部
条
件
转移控
制电路
控制存储器
0 1 2 3 4 5 6
译码器
控制字段
BAF
BCF

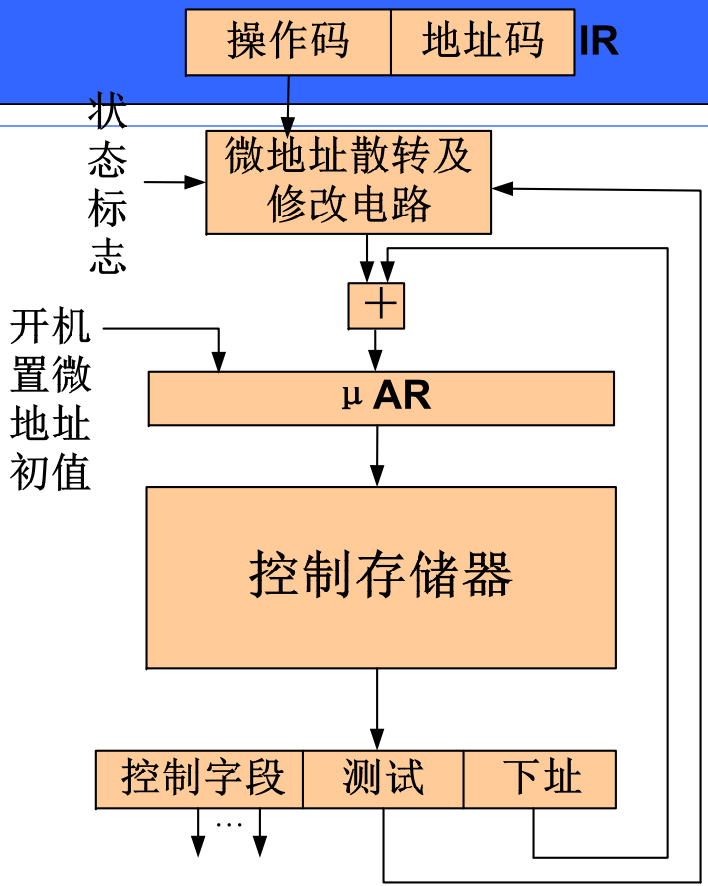
（2）判定方式（下址字段法）

- 微指令格式：

控制字段	判别测试字段	下

- 微指令格式中必须设置一个下址字段，
要执行的微指令地址，所以也称为下址
- 当微程序不产生分支时，后继微指令地
的下址字段给出；
- 当微程序出现分支时，按判别测试字段
逻辑电路来形成后继微地址。
- 由于每一条微指令至少都是一条无条件
此不必设置专门的转移微指令。

- 优点是可以实现快速多路分支，以提高微程序的执行速度，微程序在控制存储器的物理分配方便，微程序设计灵活；
- 缺点是微指令字长，形成后继微地址的结构比较复杂


操作码
地址码
IR
状态标志
微地址散转及修改电路
+
开机置微地址初值
μAR
控制存储器
控制字段
测试
下址
…

（三）微指令格式的类型

- 不同的计算机有不同的微指令格式，但一般分为水平型微指令和垂直型微指令两种类型。

（1）水平型微指令

- 微指令字采用长格式，也就是控制字段采用直接控制法和字段直接编码法，使一条微指令能控制数据通路中多个功能部件并行操作。
- 水平型微指令的优点是：一条微指令可同时发送多个微命令，微指令执行效率高，速度快，较灵活，并行操作能力强；水平型微指令构成的微程序较短。它的主要缺点是：微指令字长较长，明显地增加了控制存储器的横向容量。

（三）微指令格式的类型

（2）垂直型微指令

- 控制字段采用完全编码的方法，将一套微命令代码化构成微指令。就像计算机机器指令一样，它由微操作码、源地址和目标地址以及其他附带信息构成
- 垂直型微指令和机器指令一样分成多种类型的微指令，所有微指令构成一个微指令系统。
- 主要特点：微指令字采用短格式，每条微指令只能控制一二个微操作，并行控制能力差。但由于微指令和机器指令格式相类似，对于用户来说，垂直型微指令比较直观，容易掌握和便于使用。微指令字短，减少了横向控制存储器的容量；但微程序长，影响了执行的速度。

（四）控制存储器和动态微程序设计

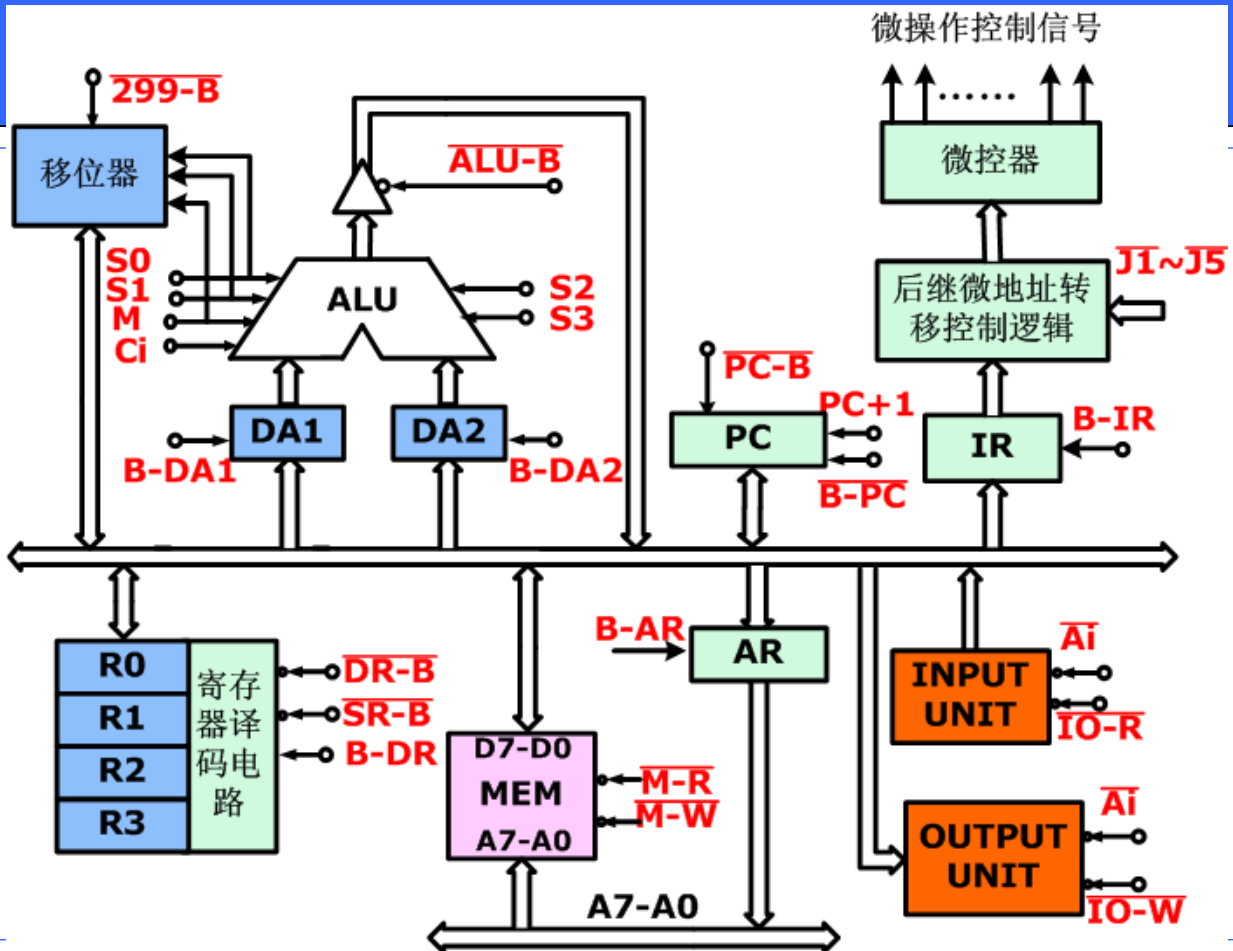
- 控制存储器：一般由ROM构成，因为指令系统一般是固定的，微程序是解释执行指令的，因此微程序一般也是固定的，所以使用只读存储器来存放微程序。
- 动态微程序设计：在一台微程序控制的计算机中，假如能根据用户的要求改变微程序，那么这台机器就具有动态微程序设计功能。
  - 具有动态微程序设计功能的控制器的CM必须是可改写的存储器，如RAM或者E^2PROM。
  - 动态微程序设计可以通过修改微程序来实现不同的指令系统，或者实现指令系统的扩充或调整。
  - 动态微程序设计的目的是使计算机能更灵活、更有效地适应于各种不同的应用场合。

第四章　控制器

- 4.1 控制器的组成及指令的执行
- 4.2 控制方式和时序的产生
- 4.3 微程序控制器
- 4.4 微程序控制器及其微程序设计举例
- 4.5 硬布线控制器

4.4 微程序控制器及其微程序设计举例

- 一、模型机系统组成
- 二、模型机微程序控制器组成
- 三、模型机微程序设计


微操作控制信号
299-B
移位器
ALU-B
微控器
S0
S1
M
Ci
ALU
S2
S3
后继微地址转
移控制逻辑
J1~J5
PC-B
PC+1
PC
IR
B-IR
B-PC
DA1
DA2
B-DA1
B-DA2
B-AR
AR
Ai
INPUT
UNIT
IO-R
R0
R1
R2
R3
寄存
器译
码电
路
DR-B
SR-B
B-DR
D7-D0
MEM
A7-A0
M-R
M-W
Ai
OUTPUT
UNIT
IO-W
A7-A0

第七章　控制器

- 4.1 控制器的组成及指令的执行
- 4.2 控制方式和时序的产生
- 4.3 微程序控制器
- 4.4 微程序控制器及其微程序设计举例
- 4.5 硬布线控制器

4.4 微程序控制器及其微程序设计举例

- 一、模型机系统组成
- 二、模型机微程序控制器组成
- 三、模型机微程序设计

三、模型机微程序设计

- （一）模型机微程序设计的步骤
- （二）微程序流程图的编写
- （三）微地址及下址字段的分配
- （四）微指令代码的编写

（一）模型机微程序设计的步骤

- 模型机微程序设计的步骤：

1. 设计实验模型机的结构和数据通路；
2. 设计指令的功能、格式（包括指令码）及寻址方式
3. 在以上的基础上，编写微程序流程图
4. 根据指令码和转移方式J1＃～J4＃，分配微地址及下址字段
5. 根据微指令格式，编写微指令代码；

（二） 微程序流程图的编写

1. 机器指令的功能由微程序完成，一条机器指令对应着一段微程序。
每条指令的微程序都包含三部分：
   - 取指令微程序段
   - 根据操作码散转至微程序入口的微指令
   - 该机器指令的独立微程序段

- 每一条指令的前两部分都相同，称作公操作，不同的是第三部分的独立微程序段，取决于该机器指令的寻址方式和功能，用于实现的指令规定的特殊功能。
- 例如：取指令及散转的公共微程序段为：
  a) PC→AR，PC＋1
  b) RAM→IR
  c) J1＃散转至微程序入口。

（二） 微程序流程图的编写(续1)

2. 每条微指令可以实现：

- 总线上的一个数据传送：例如PC→AR
- 进行运算器的一个运算：例如DA1+DA2→DR
- 启动存储器的一个读/写：例如RAM→DR。

- 按照每条微指令的功能，可以将各种操作归纳为三种：
  1) 通用寄存器之间的传送操作
  2) 存储器访问操作
  3) 运算器的运算操作

（二） 微程序流程图的编写(续2)

3. 对于带寻址方式码MOD的指令(格式二)，微程序段至少经过两个散转：第1次为J1#散转，分辨出寻址方式并计算出有效地址，第2次为J2#散转，分辨出指令并实现其功能。

4. 编写指令的微程序流程图，不仅数据通路要可行，还要考虑微码编写是否可行。

   - 例如，OUT [PORT],[ADDR]指令，功能为将存储器单元ADDR中的数据输出至输出部件LED显示。
   - 其微程序中，要进行存储器的读操作，从模型机框图上看，读出的数据在总线上，可以直接送数码管显示，即一条微指令RAM→LED即可；但其微码的编写却不可行，因为存储器的读操作要求微码FUNC FS=0100，但LED显示要求微码FUNC FS=0110，冲突；因此，必须用两条微指令实现：RAM→DA1；DA1→LED。

（二） 微程序流程图的编写(续3)

- 又例如，取指令及散转的公共微程序段为：
  a) PC→AR，PC＋1
  b) RAM→IR
  c) J1＃散转至微程序入口
- 在电路原理上，J1#信号可以在b）微指令一起发送，但由于J1＃信号与M-R＃信号被编码在同一字段，因此必须分为两条微指令完成。

5. 对于同一条指令，可能存在不同的微程序流程图，但均能实现指令的功能。

1)通用寄存器之间的传送操作

- 通过源寄存器内容送总线，而目的寄存器将总线上数据打入来实现。
- 例如：
- 指令MOV DR，SR ： 功能为将源寄存器SR的内容送目的寄存器DR。
  - 其微程序段为一条微指令：
  - SR→DR。

2)存储器访问操作

- 读访问操作：通过以下两条微指令实现：
  - a) 送地址到总线，并打入地址寄存器AR；
  - b) 启动存储器读操作，并将读出的数据从总线上接收至目的部件。
- 例如，指令MOV DR，@SR：功能为将源寄存器SR所指示的存储器地址单元的内容送目的寄存器DR，即源操作数是寄存器间接寻址。其微程序段为以下两条：
  - SR→AR
  - RAM→DR
- 取指令也是一种典型的存储器读访问操作。

2)存储器访问操作

- 写访问操作：通过以下两条微指令实现：
  a) 送地址到总线，并打入地址寄存器AR；
  b) 送数据到总线，启动存储器写操作。
- 例如，指令MOV @DR，SR：功能为将源寄存器SR的内容写至目的寄存器DR所指示的存储器地址单元，即目的操作数是寄存器间接寻址。其微程序段为以下两条：
  - SR→AR
  - DR→RAM

3)运算器的运算操作

- 运算器的运算操作：通过三条微指令实现
  - a) 送第一个数据到暂存器DA1（或者DA2）；
  - b) 送第二个数据到暂存器DA2（或者DA1）；
  - c) 选择ALU运算功能并进行运算，结果送目的部件；
- 例如，指令ADD SR，DR，功能为将源寄存器SR的内容与目的寄存器DR的内容相加，并送DR。其微程序段为三条微指令：
  - SR→DA1
  - DR→DA2。
  - DA1+DA2→DR。

（三）微地址及下址字段的分配

1、首先要分配每条机器指令的微程序入口地址。步骤如下：
   - 确定发送J1#信号的那条微指令的下址字段。
   - 确定指令的操作码。
   - 根据J1#转移的规则确定每条指令的微程序入口地址。
2、如果流程图中还有J2#～J5#等转移，则接下来必须确定它们的各分支的入口微地址。
3、在确定了上述有特殊要求的微指令的固定地址后，对于其他的顺序执行的微指令，只需直接按前后顺序随意编排即可。
4、每条指令的微程序段的最后一条微指令的下址字段一定是取指令微程序段的首地址。

（四）微指令代码的编写

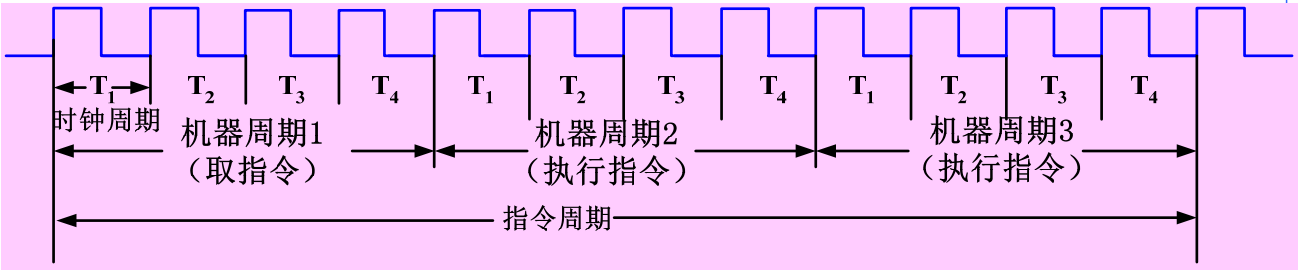
- 根据写好的微程序流程图，参照数据通路，首先排列出每条微指令必须发送的微操作控制信号，然后，对照微指令格式，写出这些微操作控制信号对应的微代码。

4.5 硬布线控制器

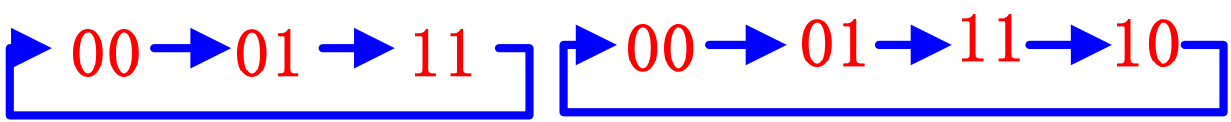
- 一、时序系统
- 二、硬布线控制器的结构
- 三、硬布线控制器的设计方法
- 四、硬布线控制器与微程序控制器的比较

一、时序系统

- 三级时序：指令周期—机器周期—节拍



- 例如：采用计数器输出译码方式产生机器周期信号假设某机器的指令系统有两条指令：
- 指令A包含三个机器周期，计数器的计数变化状态
- 指令B包含四个机器周期，计数器的变化状态



机器周期信号产生举例

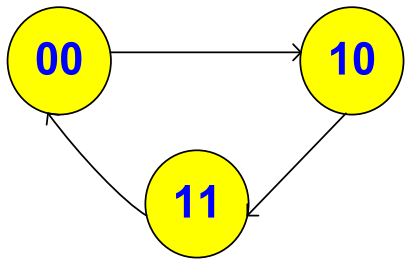
- 产生两条指令所需的机器周期信号时的计数器状态表
  - 其中Q1，Q2表示当前周期计数器状态输出，Q'1，Q'2表示下一个周期计数器状态输出

指令A				指令B			
Q_1	Q_2	Q'_1	Q'_2	Q_1	Q_2	Q'_1	Q'_2
0	0	1	0	0	0	0	1
1	0	1	1	0	1	1	1
1	1	0	0	1	1	1	0
				1	0	0	0

机器周期信号产生举例

- 根据表7－7的真值表列出计数器的输出表达式，对于指令A，其表达式为

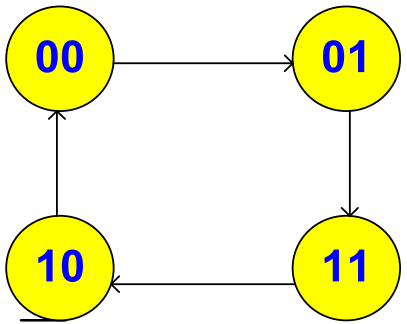
指令A



$$Q_1^{'} = \overline{Q_1}\,\overline{Q_2} + Q_1\overline{Q_2} = \overline{Q_2}$$

$$Q_2^{'} = Q_1\overline{Q_2}$$

指令B



$$Q_1^{'} = \overline{Q_1}Q_2 + Q_1Q_2 = Q_2$$

$$Q_2^{'} = \overline{Q_1}\,\overline{Q_2} + \overline{Q_1}Q_2 = \overline{Q}_1$$

$$Q_1^{'} = \text{指令}A \cdot \overline{Q_2} + \text{指令}B \cdot Q_2$$

$$Q_2^{'} = \text{指令}A \cdot Q_1 \cdot \overline{Q_2} + \text{指令}B \cdot \overline{Q_1}$$

机器周期信号产生举例

- 根据表达式画出逻辑电路图
- 当执行指令A时，产生机器周期信号M0、M1、M2；
- 而当执行指令B时，产生机器周期信号M0、M1、M2、M3。

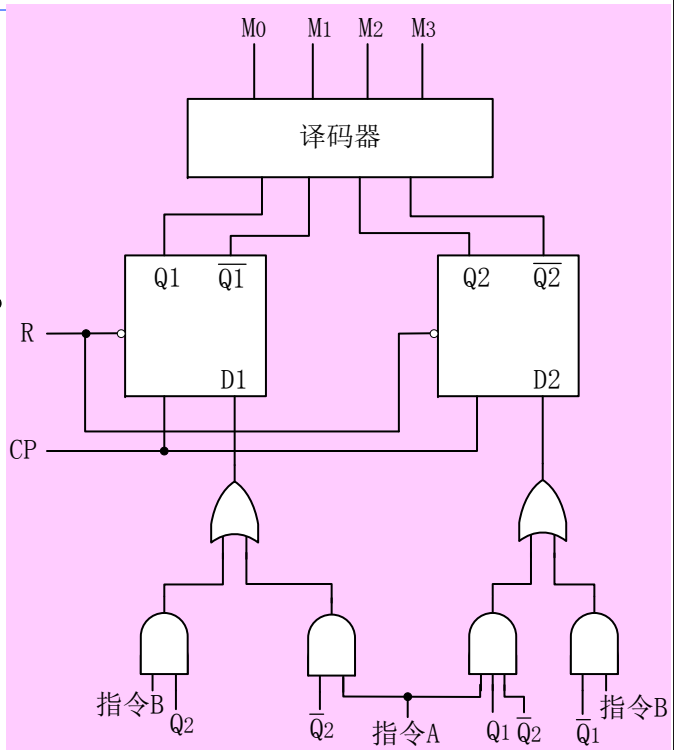
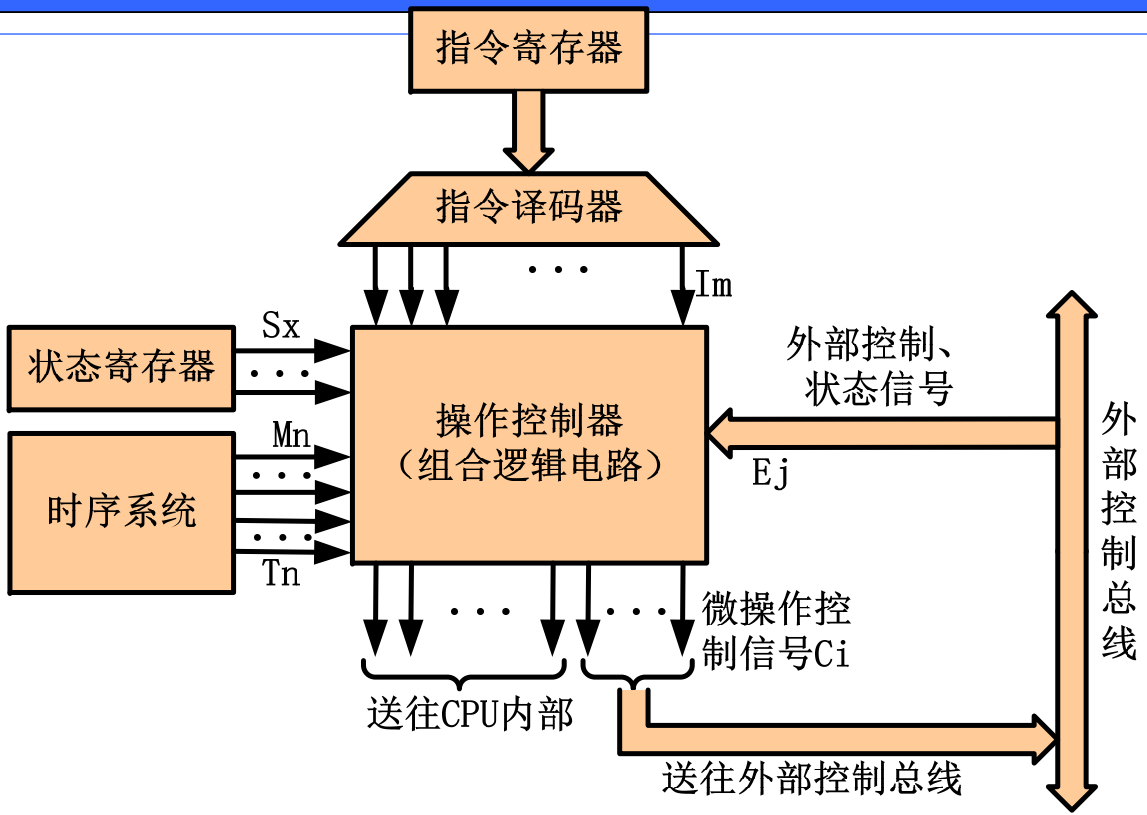


图7－29 两条指令的机器周期产生电路图

二、硬布线控制器的结构


指令寄存器
指令译码器
Im
Sx
状态寄存器
Mn
时序系统
Tn
操作控制器
（组合逻辑电路）
外部控制、
状态信号
Ej
外部控制总线
微操作控制信号Ci
送往CPU内部
送往外部控制总线

二、硬布线控制器的结构

- **基本原理：**是根据指令的要求、当前的时序及外部和内部的状态情况，按时间的顺序发送一系列微操作控制信号。它由复杂的组合逻辑门电路和一些触发器构成，因此又称为组合逻辑控制器，或常规逻辑控制器。
- 从逻辑函数的角度来看，输出微操作控制信号Ci是4种输入信号的函数：Ci=fi（Im，Mn，Tn，Sx，Ej）
- 设计硬布线控制器的过程，也就是求出每个微操作控制信号Ci的逻辑函数fi的过程。

二、硬布线控制器的结构

- 1、指令信息Im用于指出当前是哪一条指令的指令周期。
- 2、机器周期信号Mn和时钟周期信号Tn指出当前处于哪一个机器周期和哪一个节拍。
- 3、状态信号Sx指出运算器的结果状态及机器内部的其他状态，以决定某些操作信号是否发送。
- 4、外部控制、状态信号Ej指出和传递CPU外部各部件的状态和控制信号
- 5、微操作控制信号Ci：一部分送到CPU外部构成系统总线的控制总线；另一部分则送到CPU内部供使用。

三、硬布线控制器的设计方法

- 硬布线控制器的设计步骤：
  - 1、确定指令系统，包括指令系统中每条指令的格式、功能和寻址方式。
  - 2、围绕着指令系统的实现，确定CPU的内部结构，包括运算器的功能和组成，控制器的组成及它们的连接方式和数据通路，同时也确定时序系统的构成。
  - 3、分析每条指令的执行过程，按机器周期顺序，写出所必需发送的微操作控制信号序列。
  - 4、综合每个微操作控制信号的逻辑函数，化简和优化。
  - 5、用逻辑电路实现。

硬布线控制器设计举例

- 例如：设计实现ADD和JMP指令 的硬布线控制器。
- ADD 和JMP指令均为二字节指令。
- ADD指令的第一个字是操作码和寄存器地址，第二个字是立即数。
- JMP指令第一个字是操作码，第二个字是转移的直接地址。

表4—1 存放在存储器中的二条指令内容

指令地址	指令机器码	助记符
0000 0100	0101 0000	ADD R_0, 06H 立即数
0000 0101	0000 0110	
0000 0110	1000 0000	JMP 04H 转移地址
0000 0111	0000 0100	

1、首先列出ADD和JMP指令的执行过程

- **ADD指令：**
  - M0： PC→AR，PC+1→PC；（取指令地址）
  - M1： RAM→IR, J1#；（取指令并译码）
  - ADD·M2： PC→AR，PC+1→PC；（取指令第二字地址）
  - ADD·M3： RAM→ALU；（取数据）
  - ADD·M4： Ri→ALU；（送寄存器数据）
  - ADD·M5： ALU（+）→Ri ；（计算并存结果）

1、首先列出ADD和JMP指令的执行过程

- **JMP指令：**
  - **M0： PC→AR，PC+1→PC；（取指令地址）**
  - **M1： RAM→IR,J1#；（取指令并译码）**
  - **JMP·M2： PC→AR，PC+1→PC；（取指令第二字地址）**
  - **JMP·M3： RAM→ALU；（取转移地址）**
  - **JMP·M4： ALU→PC；（执行转移）**

2、写出每个机器周期所必需发送的微操作控制信号序列

- 取指令公操作：
  - M0： PC→IB，IB→AR，PC+1；
  - M1： MEMR#（内存读），IB→IR,J1#；
- ADD指令：
  - ADD·M2： PC→IB，IB→AR，PC+1；
  - ADD·M3： MEMR#，IB→ALU；
  - ADD·M4： Ri→IB，IB→ALU；
  - ADD·M5： ALU（F=A+B），IB→Ri ；
- JMP指令：
  - JMP·M2： PC→IB，IB→AR，PC+1；
  - JMP·M3： MEMR#，IB→ALU；
  - JMP·M4： ALU（F=A），IB→PC ；

3、综合所有的微操作控制信号

- 即对于某一个微操作控制信号，将上述列表中，凡是在冒号“：”右边出现该信号的机器周期，把其左边的条件（与项）作为一个或项，全部进行或运算，即得到该微操作控制信号的逻辑函数。
  - PC→IB= （M0+ ADD·M2+ JMP·M2+……）
  - IB→AR= （M0+ ADD·M2+ JMP·M2+……）
  - PC+1 = （M0+ ADD·
  - MEMR=（M1+ADD·
  - IB→IR = （M1+……
  - IB→ALU= ADD·M3+
  - Ri→IB = ADD·M4+ .
  - RAM→ALU = ADD·

  - ALU（+）→Ri =ADD·M5+……
  - ALU→PC = JMP·M4+……
  - ALU（F=A+B）= ADD·M5+……
  - IB→Ri = ADD·M5+……
  - ALU（F=A）= JMP·M4+……
  - IB→PC = JMP·M4+……
  - ……

•若某个微操作控制信号必须在某个机器周期内的Tn时刻有效，则该信号表达式还要与上Tn时钟周期信号

4、优化、简化、实现

- 对逻辑函数优化和简化，使得逻辑电路最简，用硬件电路实现

四、硬布线控制器与微程序控制器的比较

比较内容	微程序控制器	硬布线控制器
工作原理	微操作控制信号事先以微程序的形式存放在控存中，执行指令时读出即可	微操作控制信号由组合逻辑电路根据当前的指令码、状态和时序，即时产生
执行速度	慢	快
规整性	较规整	繁琐、不规整
应用场合	CISC CPU	RISC CPU
易扩充性	易扩充修改	困难

小结

- 控制器是计算机硬件的核心部件，是根据机器指令产生执行指令时全机所需要的操作控制信号，协调控制计算机各个部件有序工作。掌握重点：
- 控制器的功能：取指令、执行指令、分析指令
- 控制器的组成：
  - 专用寄存器：PC、IR、AR、DR、PSW
  - 指令译码器ID
  - 时序系统
  - 时序信号产生器（操作控制器）
- 指令的执行过程：由取指令阶段和执行阶段构成，取指令阶段的操作是公共的；而执行阶段的操作由指令操作码决定。

小结

- 指令周期、机器周期、时钟周期的概念
- 控制器有两种设计方法：
  - 硬布线控制器：它是将指令执行时的各个机器周期的微操作信号用时序逻辑电路来实现，硬布线控制器速度快，但设计复杂繁琐，适合于RISC结构。
  - 微程序控制器：它是将机器指令根据其执行步骤分成若干条微指令，指令执行时从控制存储器中依次取出这些微指令，发出指令所需要的全部微操作控制信号，从而完成指令的执行。微程序控制器相对硬布线控制器速度慢，但设计比较规整，易于实现指令系统修改，适合于CISC结构。

敬请批评指正

谢　谢