微程序理控制器举例

微程序控制器

微程序控制的基本思想，就是仿照通常的解题程序的方法，把操作控制信号编成所谓的“微指令”，存放到一个只读存储器里．当机器运行时，一条又一条地读出这些微指令，从而产生全机所需要的各种操作控制信号，使相应部件执行所规定的操作。

组合逻辑电路一经实现，不能变动其逻辑关系，必要时，必须改变其连线或重新设计。

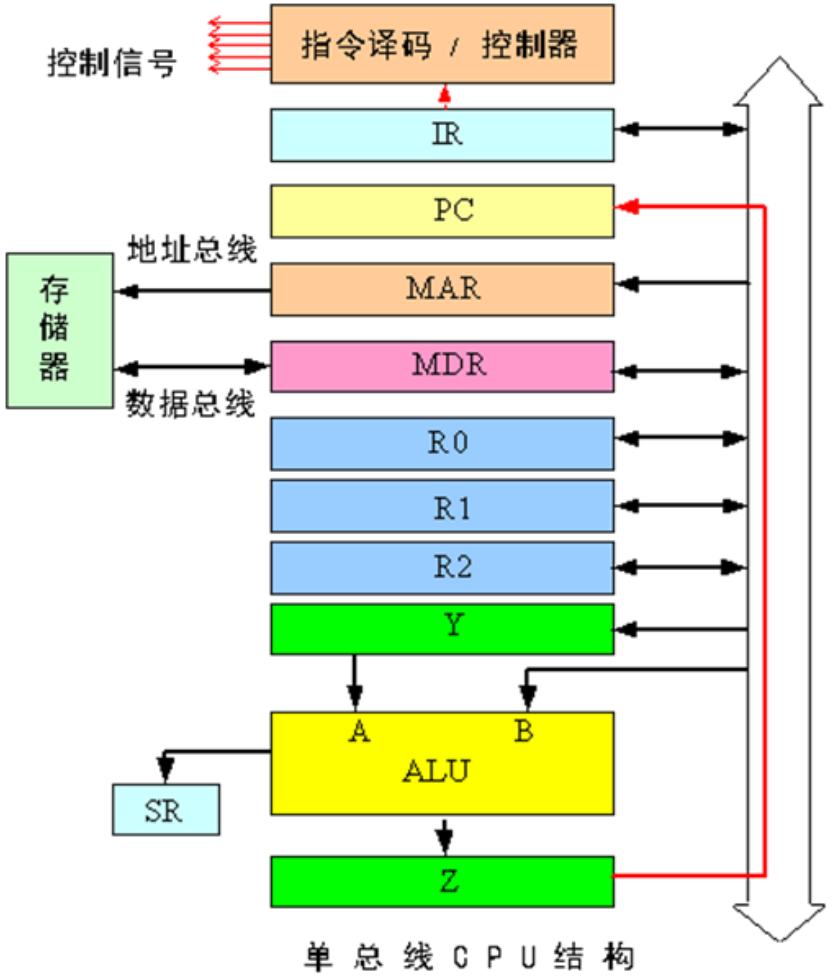
微程序控制方法：把指令执行所需要的所有控制信号存放在控制存储器中，需要时从这个存储器中读取，存储逻辑可以修改ROM存放的数据,从而修改逻辑功能,速度略慢,有一个寻址和读数据的过程.

微程序控制的特点：灵活性好，速度慢

微程序控制方式的历史简介

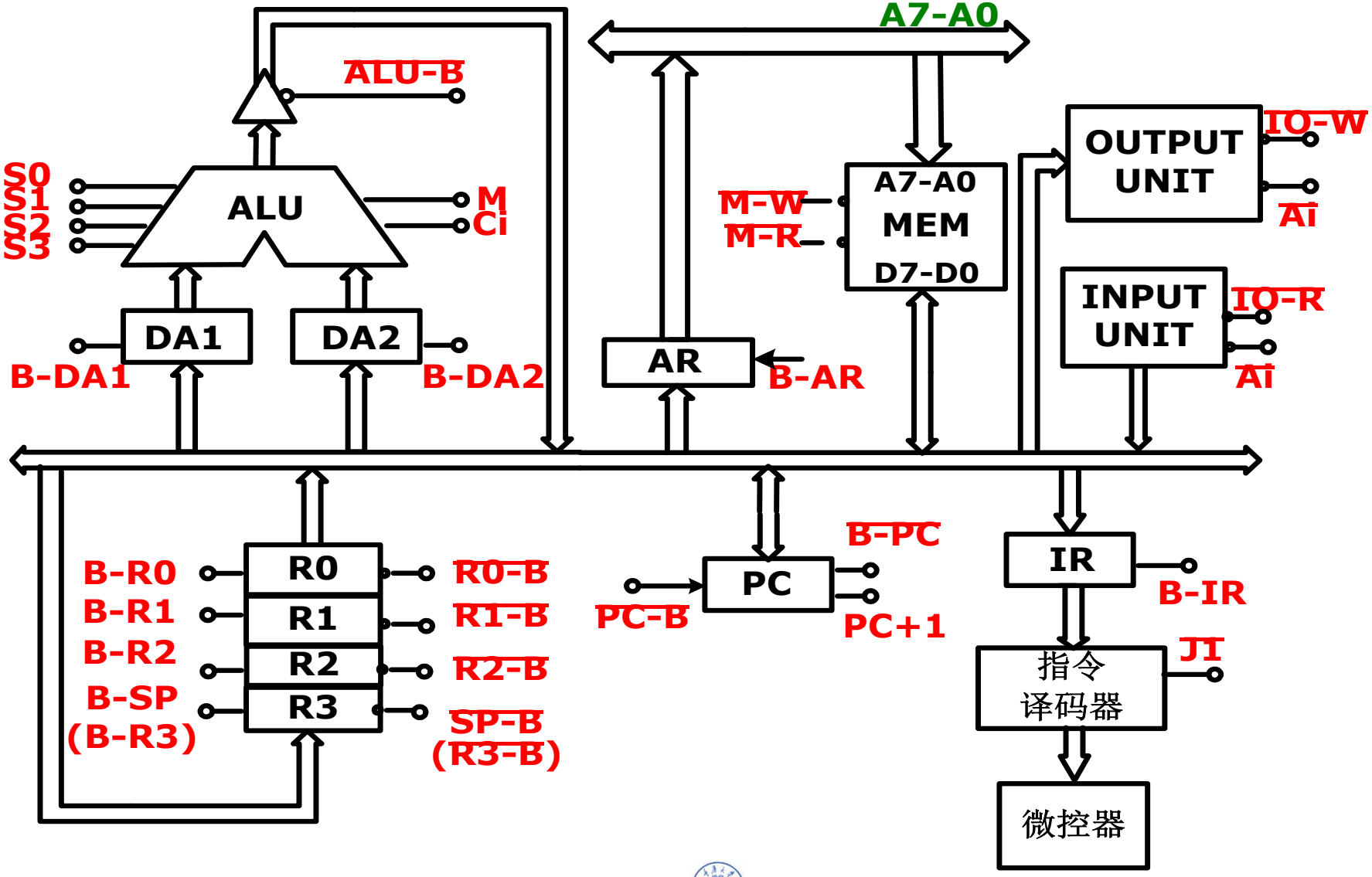
由莫里斯·威尔克斯（1967 图灵奖）。1946年10月，以冯·诺伊曼的EDVAC为蓝本设计建造了EDSAC。它使用了水银延迟线作存储器，穿孔纸带为输入设备和电传打字机为输出设备。EDSAC是第一台诺依漫机器结构的电子计算机。 在设计与制造EDSAC和EDSAC2的过程中，威尔克斯创造和发明了许多新的技术概念。诸如“变址”、“宏指令”、微程序、子例程及子例程库、高速缓冲存储器（Cache）等等，这些都对现代计算机的体系结构和程序设计技术产生了深远的影响。

单总线处理器—寄存器级抽象



单 总 线 C P U 结 构

简单模型机举例（单总线处理机的示例）



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

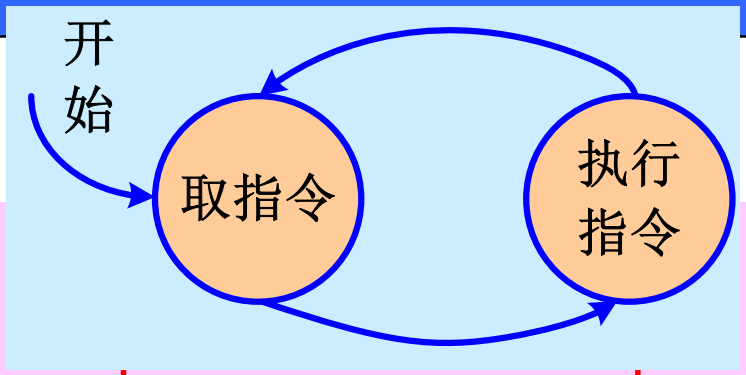
二、控制器的组成

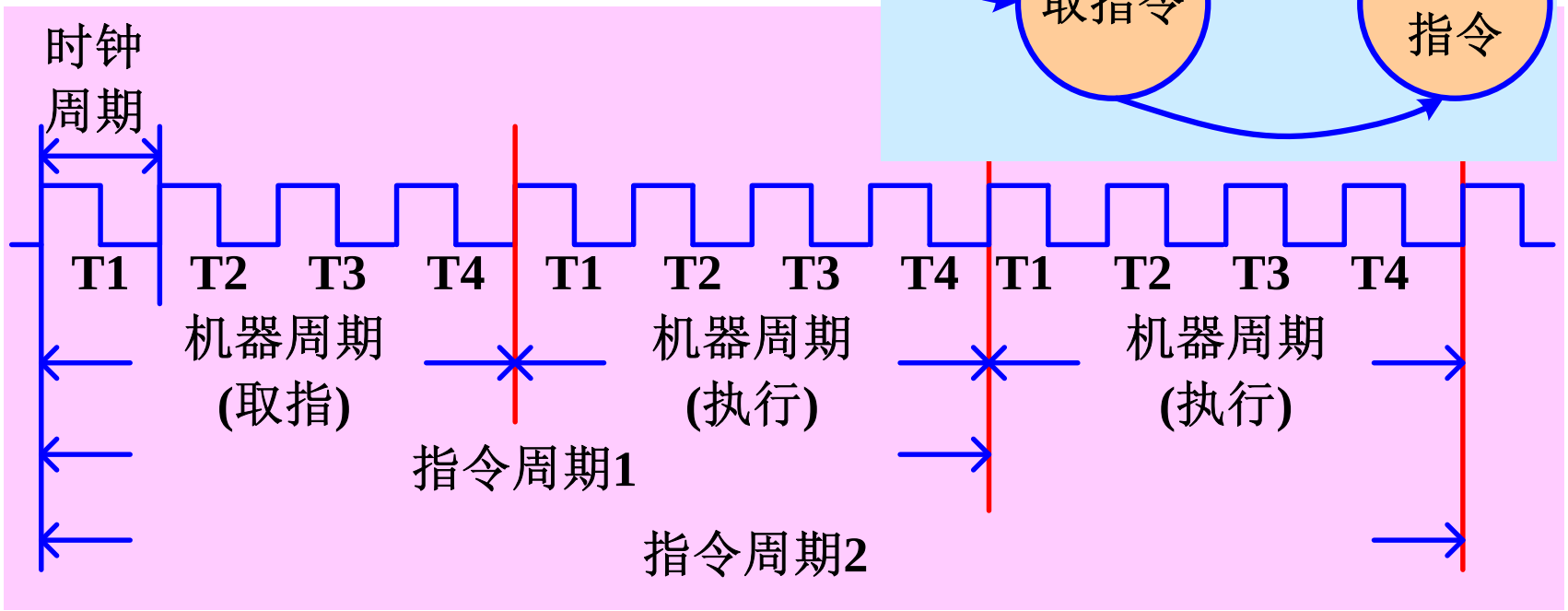
- **4. 时序信号产生器：**负责提供时钟信号和机器周期信号，以规定每个操作的时间。
  - 启停线路，负责控制时钟脉冲的送出与封锁，从而实现计算机的启动与停止。
- **5. 操作控制信号形成部件：**根据指令的操作码以及时序信号，产生取出指令和执行这条指令所需的各种操作控制信号，以便正确地建立数据通路，完成取出指令和执行指令的控制。
  - 操作控制信号形成部件采用组合逻辑电路的控制器，称作**硬布线控制器**；采用存储逻辑的称作**微程序控制器**。

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。

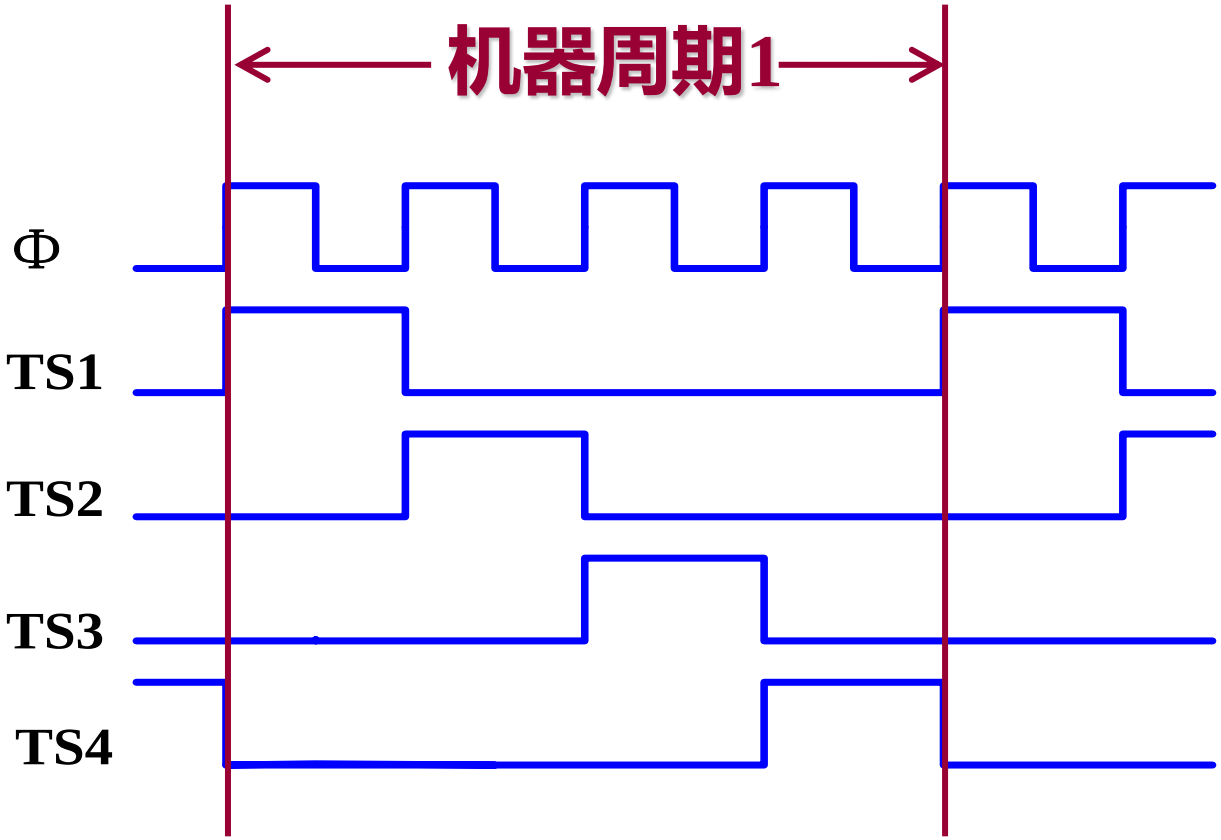
指令周期、机器周期、时钟周期的关系



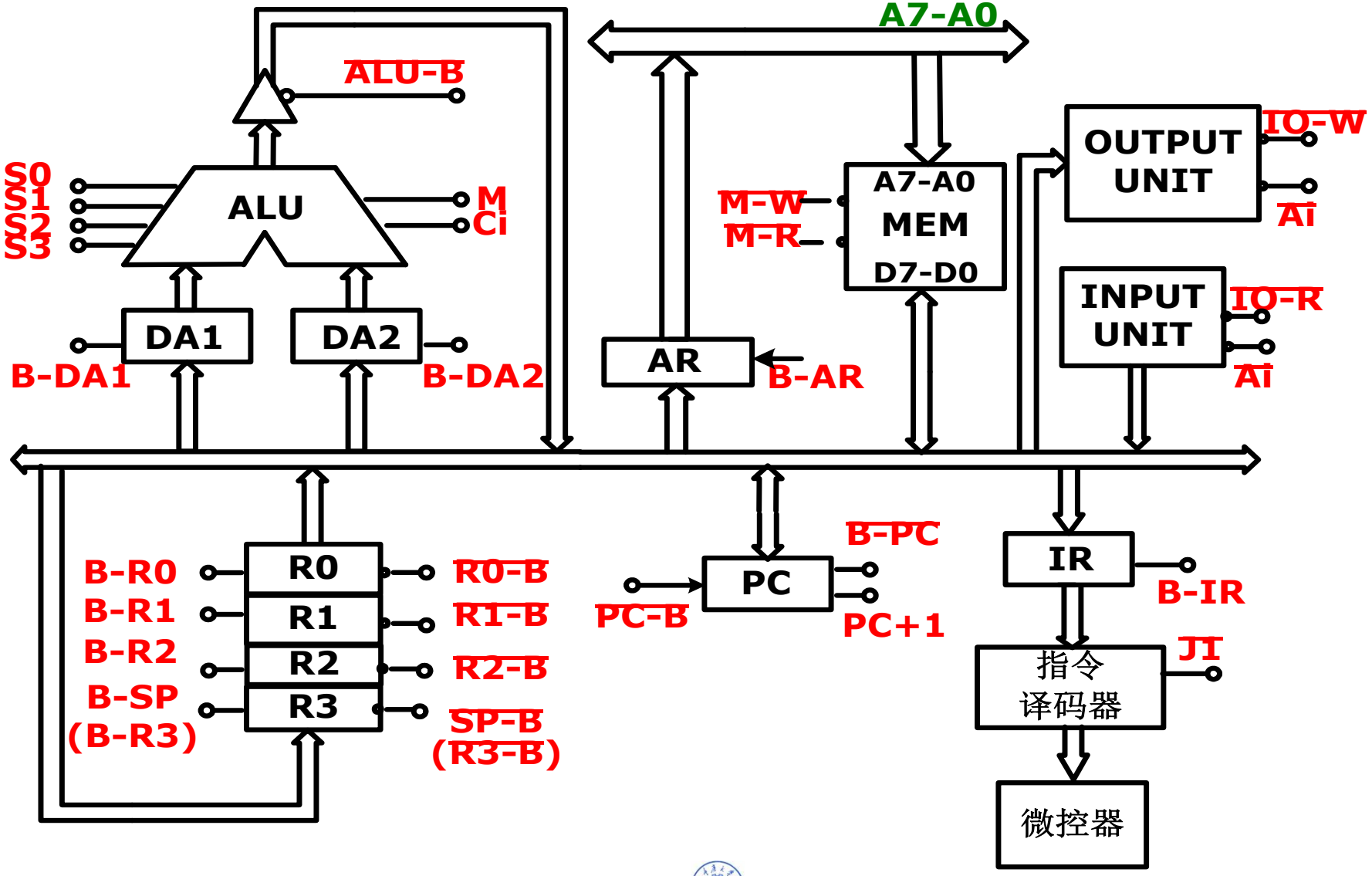


节拍信号TS1~TS4和时钟信号源Φ的关系

- 下例中，每四个节拍信号构成一个机器周期。



四 简单模型机举例


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
PC-B
PC+1
IR
B-IR
JI
指令
译码器
微控器

1 简单指令构成的例子程序

- 假定指令的长度为2字节，4个寄存器
- 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	$(R_0)+06H \rightarrow R_0$
05H	0000 0110（立即数）		
06H	1000 0000	JMP 04H	04H→PC
07H	0000 0100（转移地址）		

2 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：
- **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**

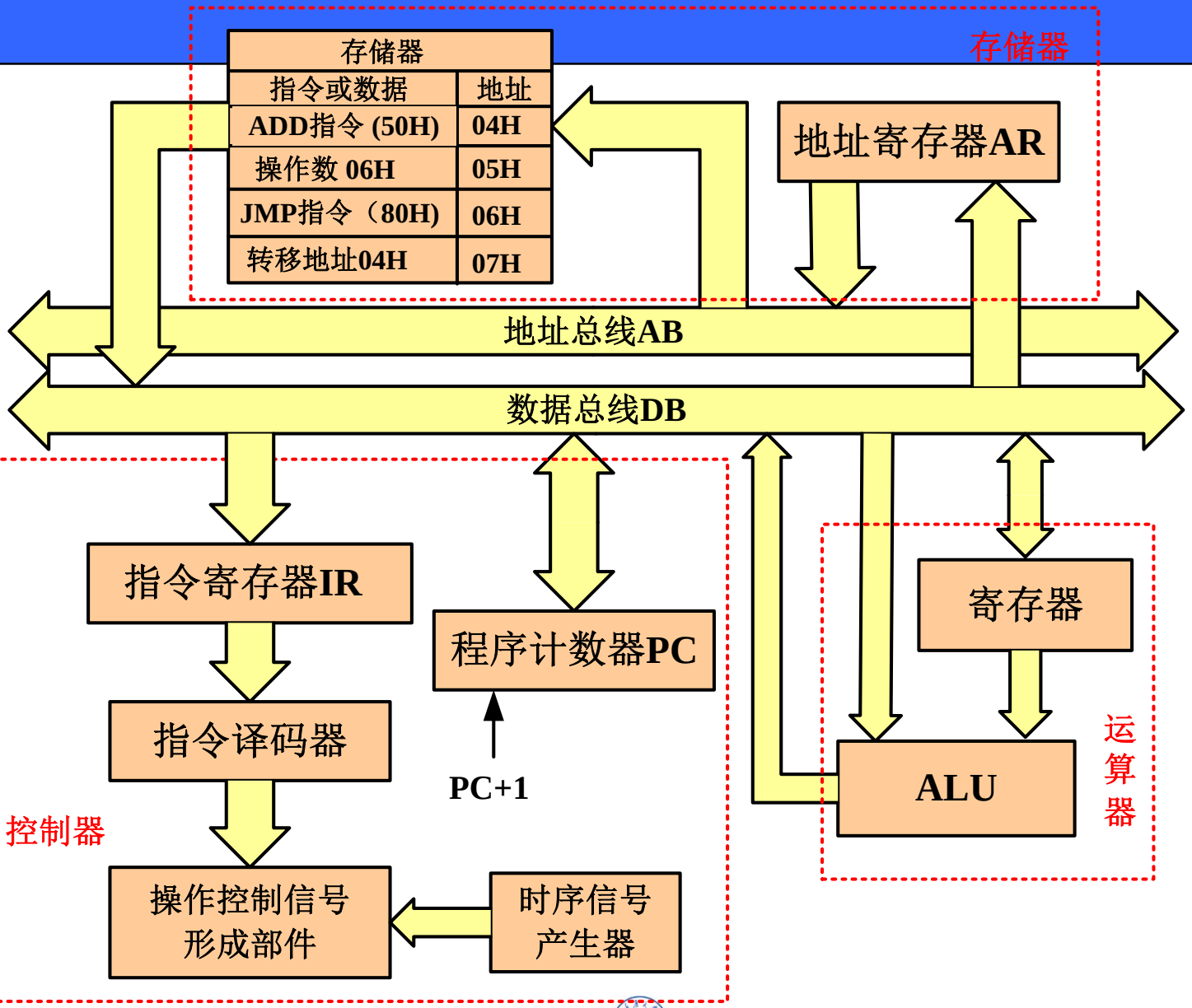
2指令执行过程概述

- （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# 。

执行指令

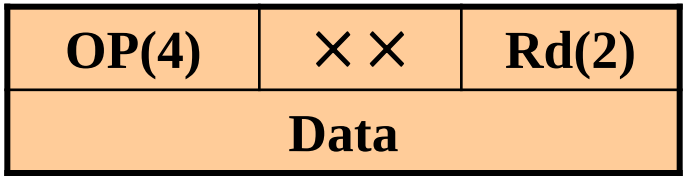
- 操作控制信号形成部件根据指令译码信息和时序周期信号，发出该指令所需的所有部件的有一定时序关系的控制信号序列，完成指令的执行。
- 执行指令的具体操作与指令的功能有很大的关系，不同的指令，其执行指令阶段也是不同的。

程序执行的初始状态


存储器
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
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

ADD 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：

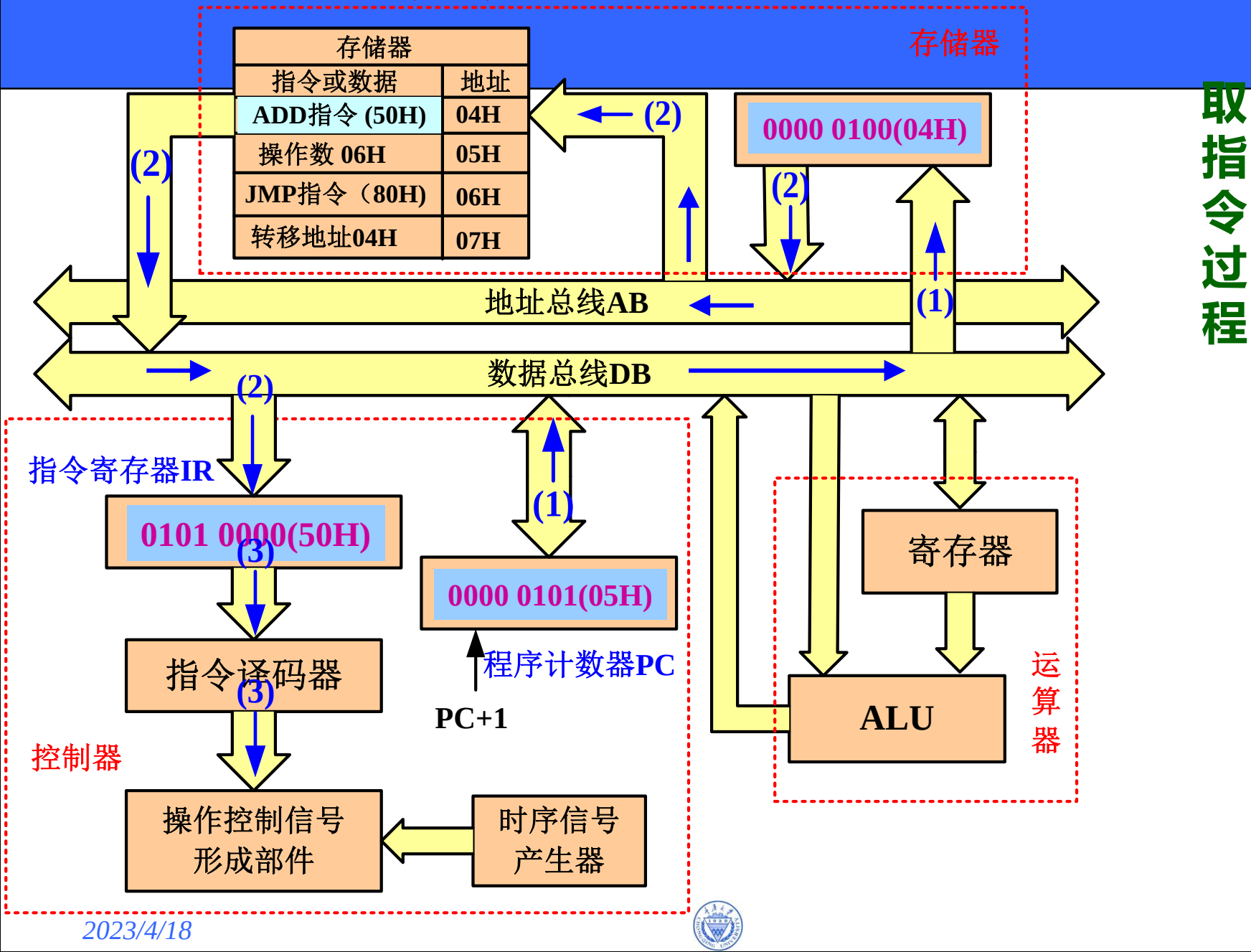


  - 执行过程：

ADD Rd, Data指令的执行过程

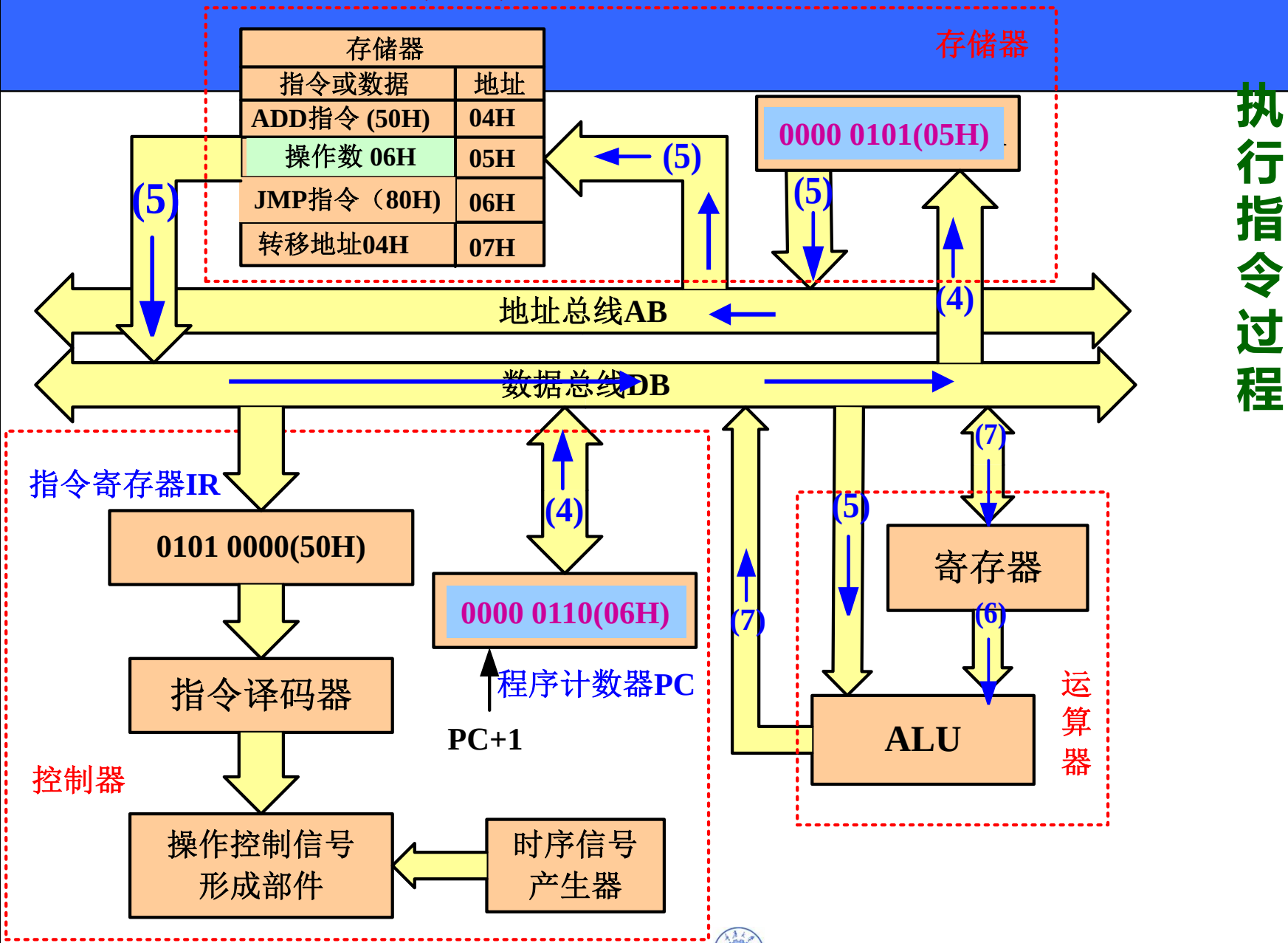
- **取指令：**
  - M1 (送存储器地址)：　　PC→AR, PC+1
  - M2 (读存储器)：　RAM→ IR
  - M3(指令译码）：　　J1#
- **执行指令：**
  - M4（取源操作数－送地址）：PC→AR, PC+1
  - M5（取源操作数－读）：RAM→ DA1
  - M6（取目的操作数）： Rd→DA2
  - M7（计算并置结果）：DA1+DA2→Rd

ADD R0, 06H；(R0)+06H→R0指令的运行过程



取指令过程

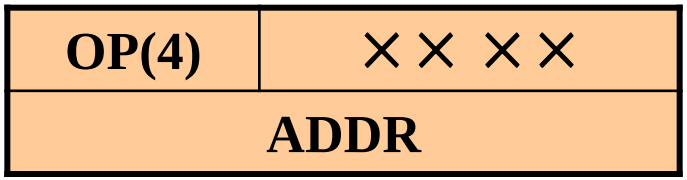
ADD R0, 06H；(R0)+06H→R0指令的运行过程



执行指令过程

JMP 指令的指令周期

- **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：

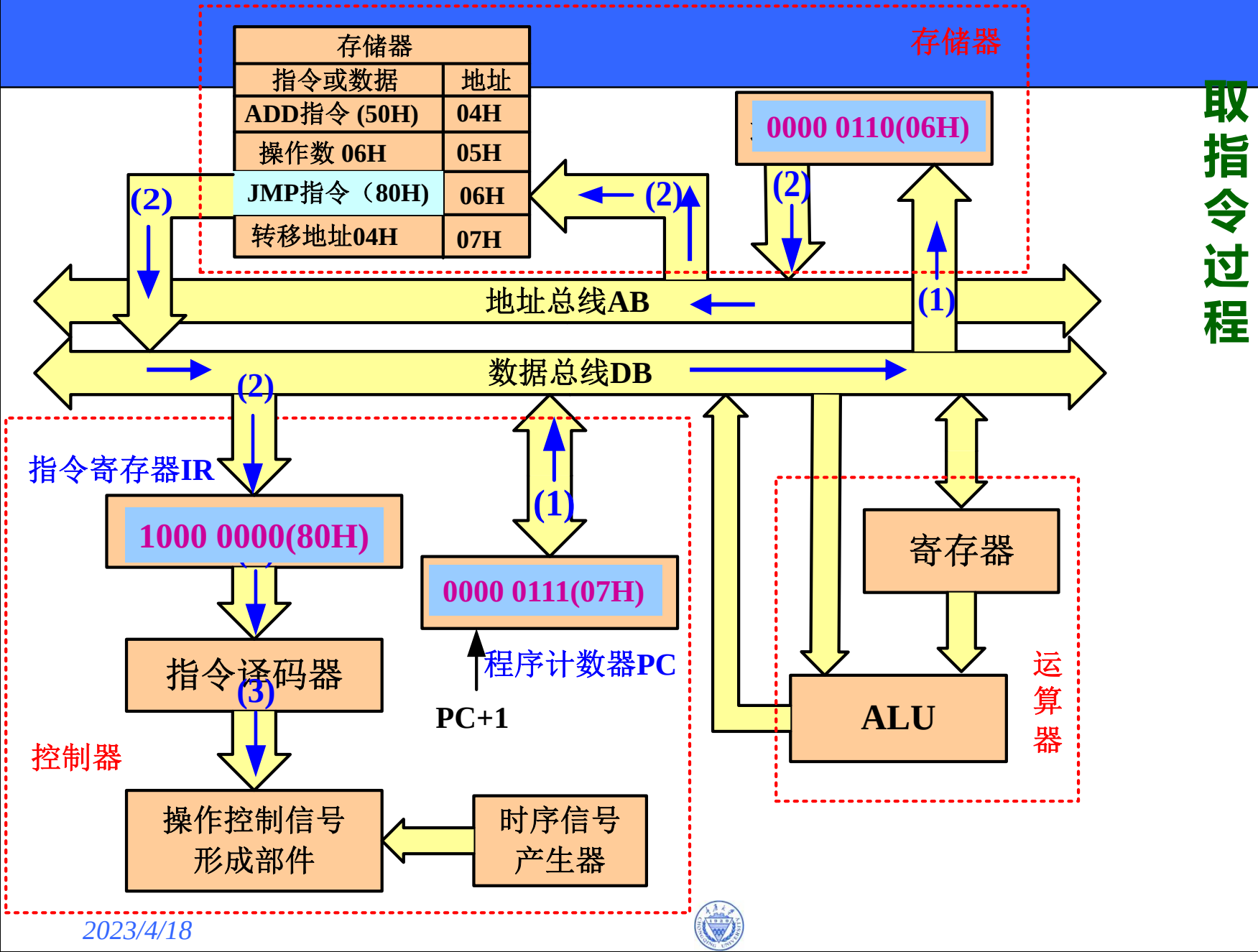


  - 执行过程：

JMP ADDR指令的执行过程

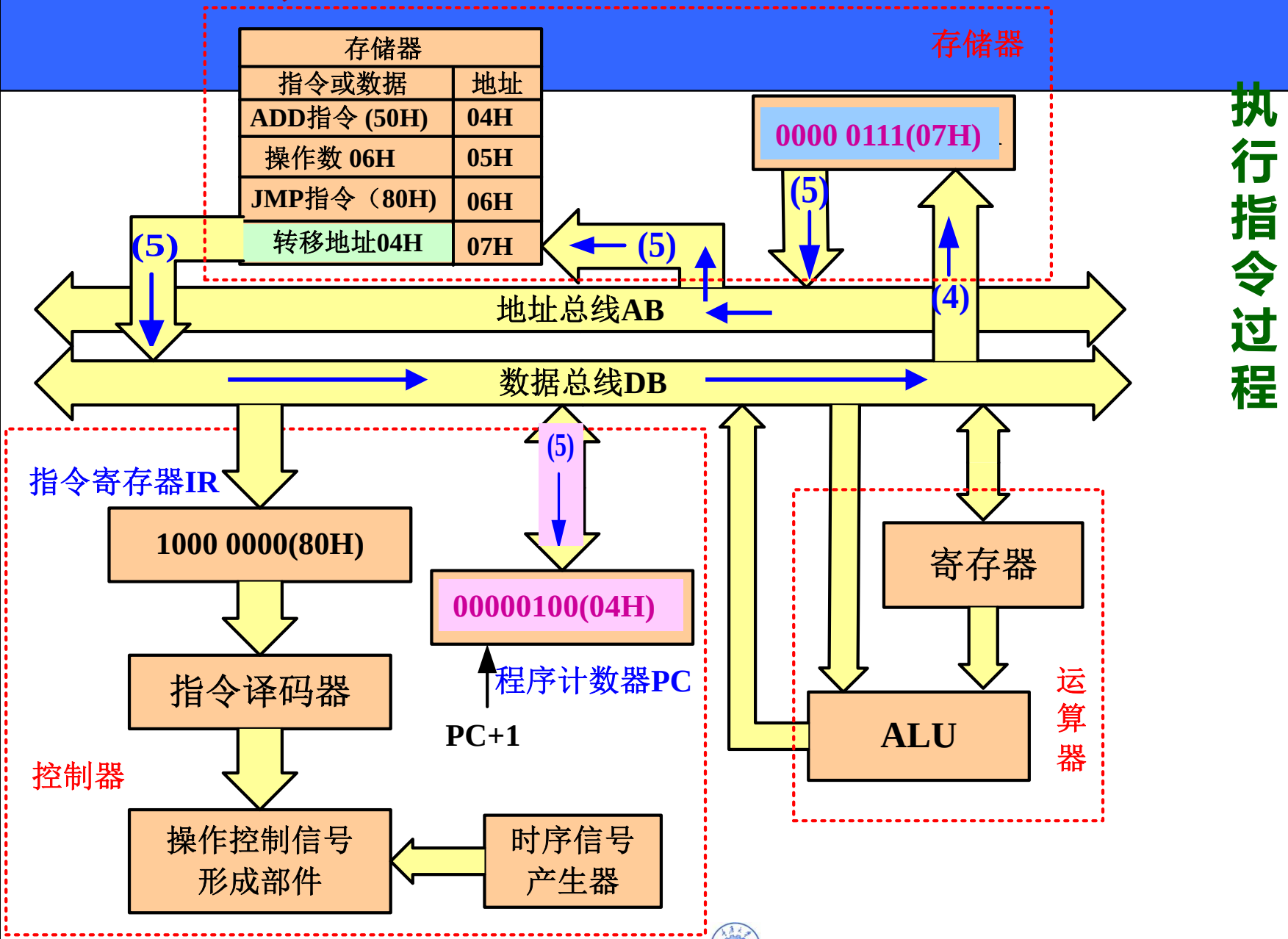
- **取指令：**
  - **M1 (送存储器地址)：PC→AR, PC+1**
  - **M2 (读存储器)：RAM→IR**
  - **M3(指令译码）：J1#**
- **执行指令：**
  - **M4（取操作数—送地址）：PC→AR, PC+1**
  - **M5（取操作数—读）：RAM→ PC**

JMP 04H；04H→PC指令的运行过程



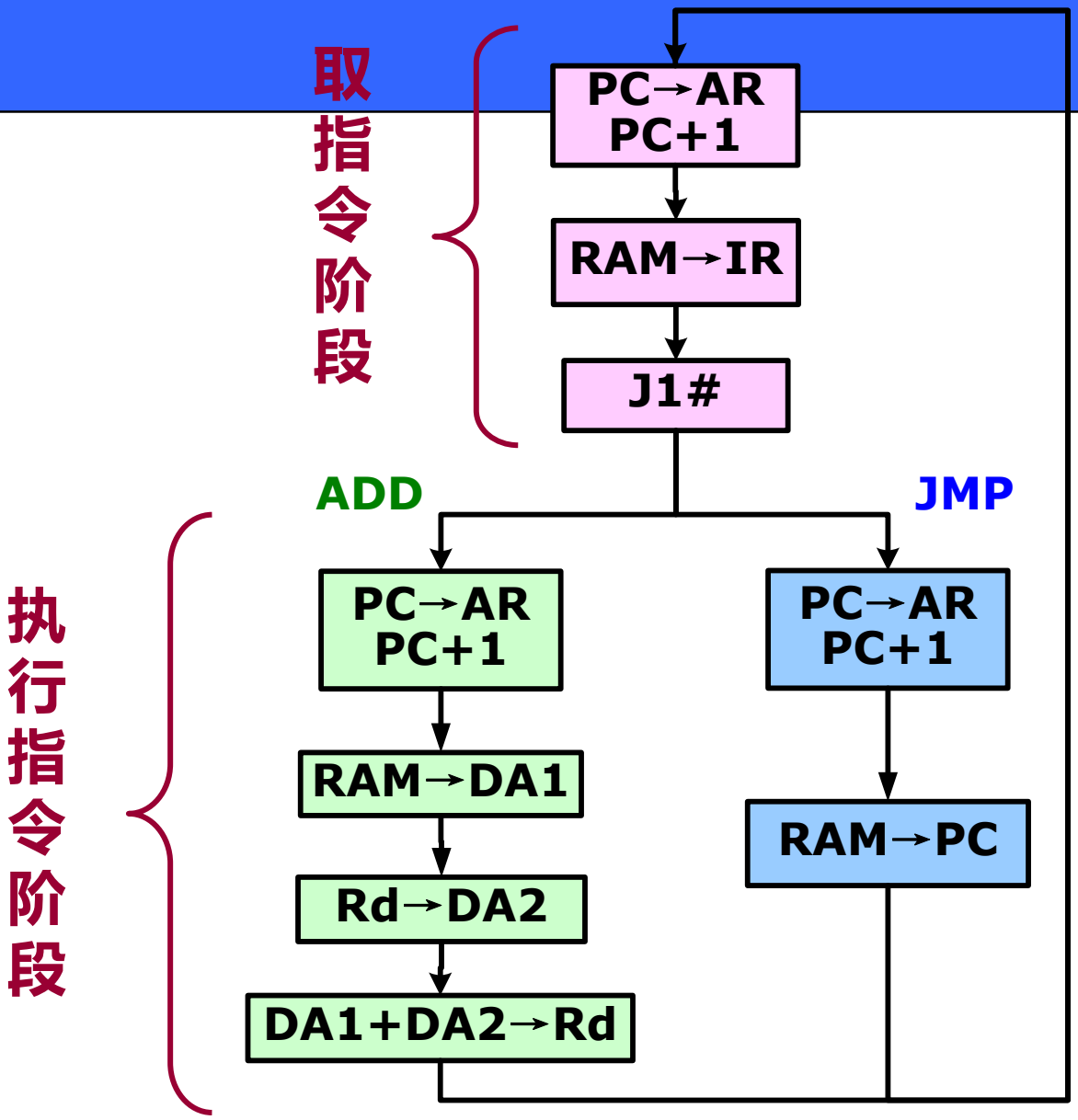
取指令过程

JMP 04H；04H→PC指令的运行过程



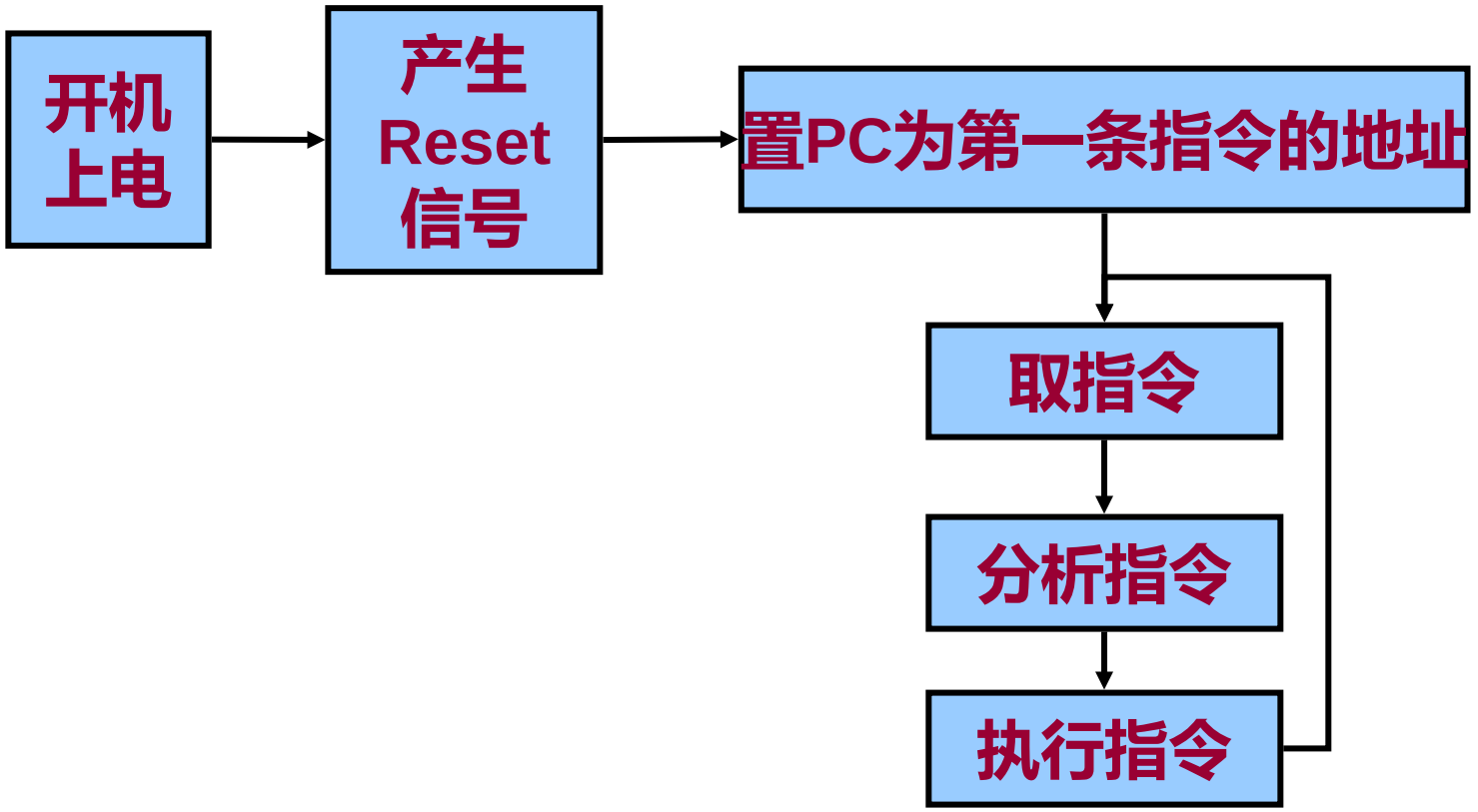
执行指令过程

指令执行的流程图


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

计算机的工作过程的实质

- 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。



五 微程序控制器

- 1、基本概念
- 2、微程序控制器的基本工作原理
- 3、微程序控制器的组成
- 4、微程序控制原理举例

1、 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。

1、 基本概念(续)

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。

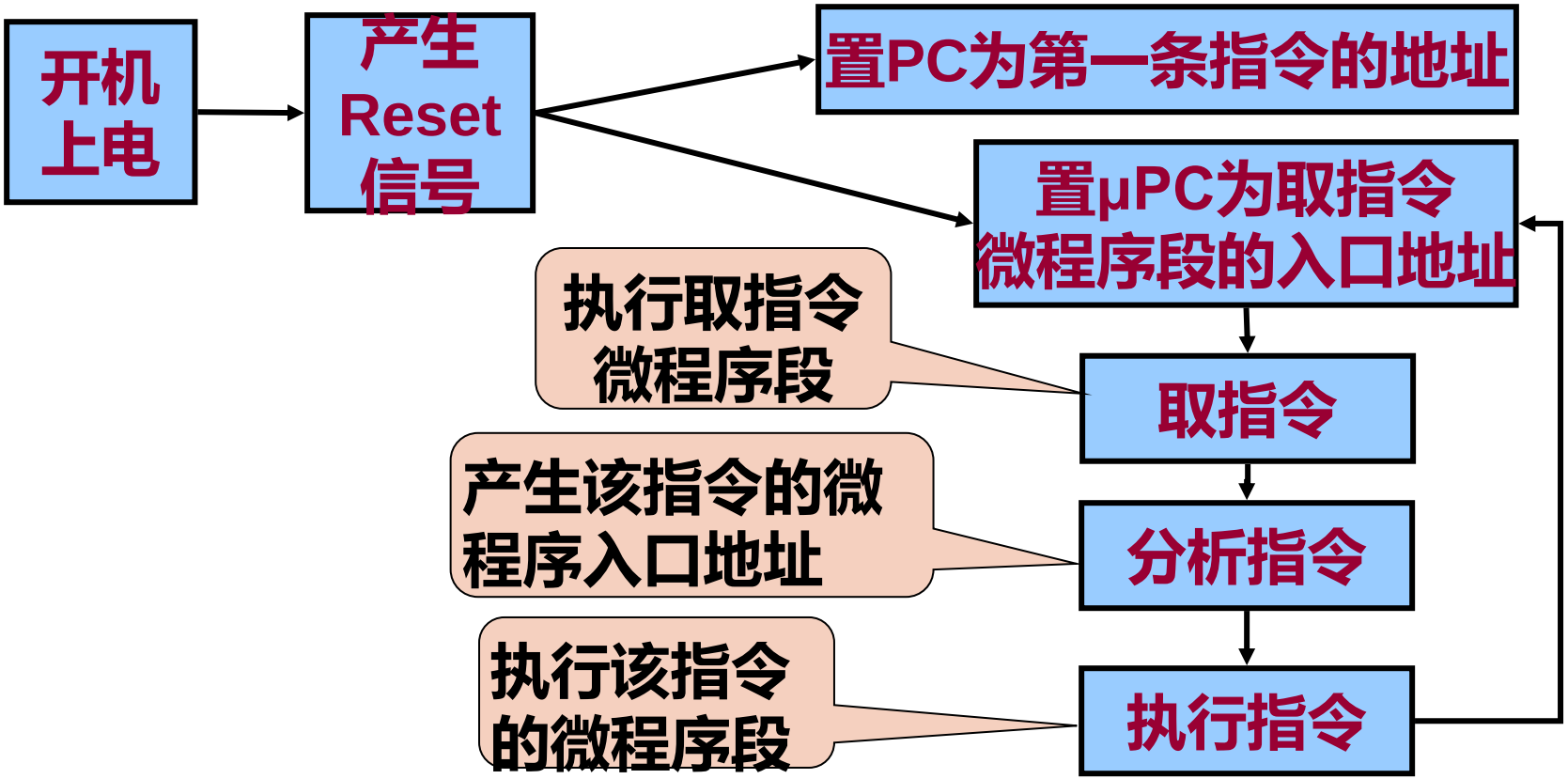
⑦ **微周期：** 指从控存中取出并执行一条微指令所需要的时间，一般**与一个机器周期相当**。

五 微程序控制器

- 1、基本概念
- 2、微程序控制器的基本工作原理
- 3、微程序控制器的组成
- 4、微程序控制原理举例

2、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：

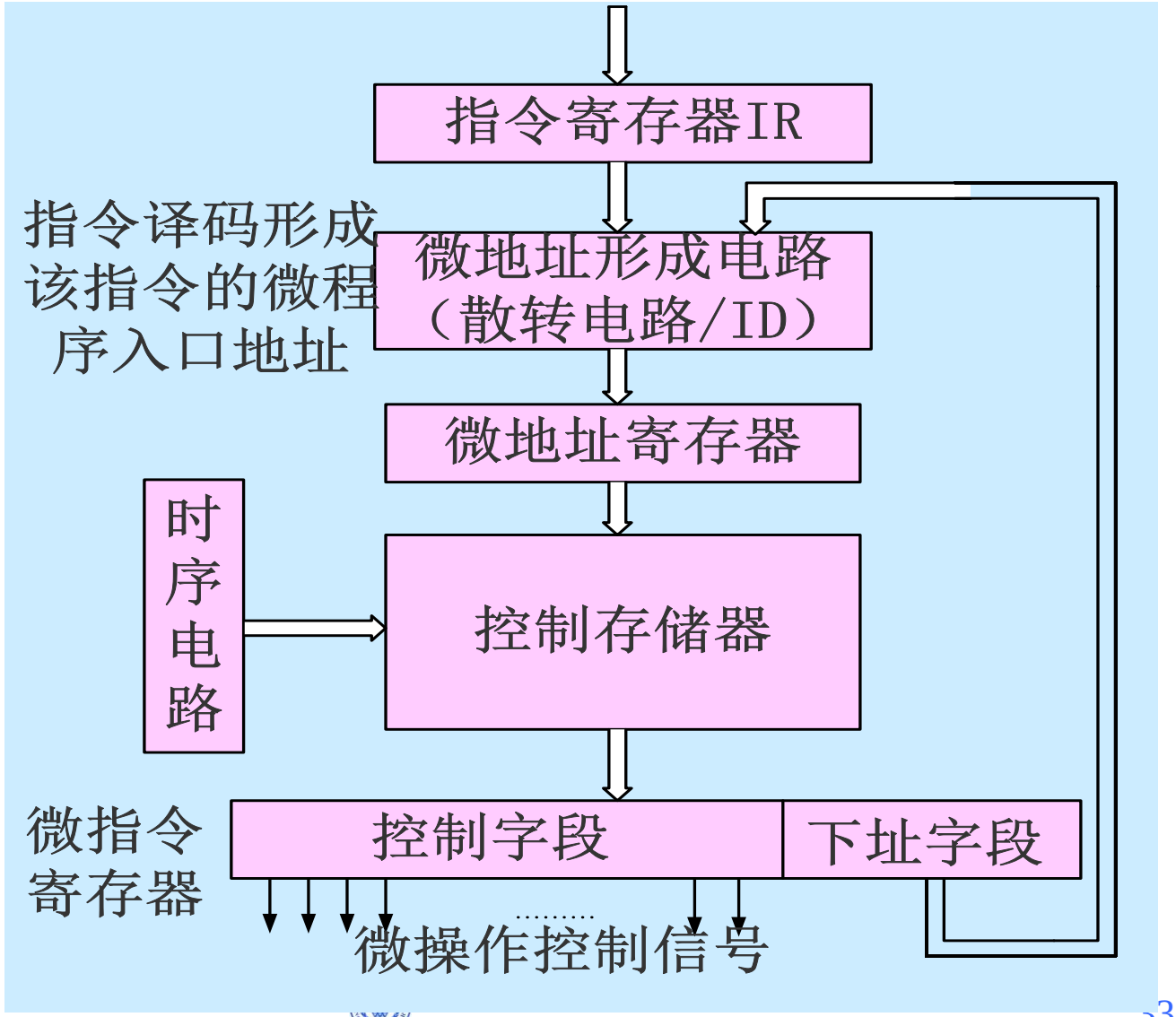


五 微程序控制器

- 1、基本概念
- 2、微程序控制器的基本工作原理
- 3、微程序控制器的组成
- 4、微程序控制原理举例

3、微程序控制器的组成

微程序控制器的组成框图



3 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。

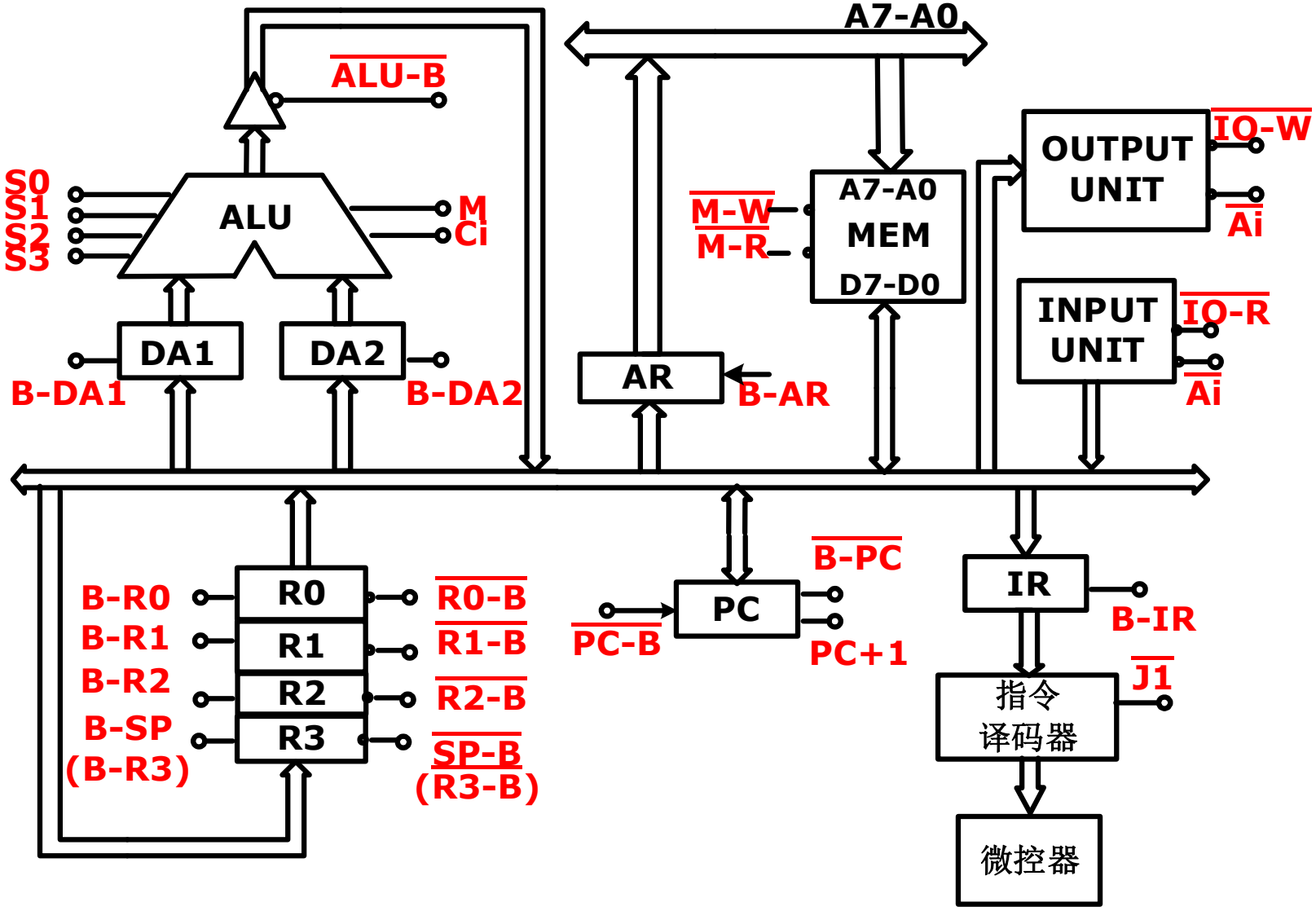
③ **微指令寄存器**:存放从控存中读出的微指令本身，又称μIR。其控制字段用于产生微操作控制信号，其下址则送至微地址形成电路，产生下一条微指令的地址。

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。**

4 微程序控制器

- 1、基本概念
- 2、微程序控制器的基本工作原理
- 3、微程序控制器的组成
- 4、微程序控制原理举例


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
J1
指令
译码器
微控器

4.0 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

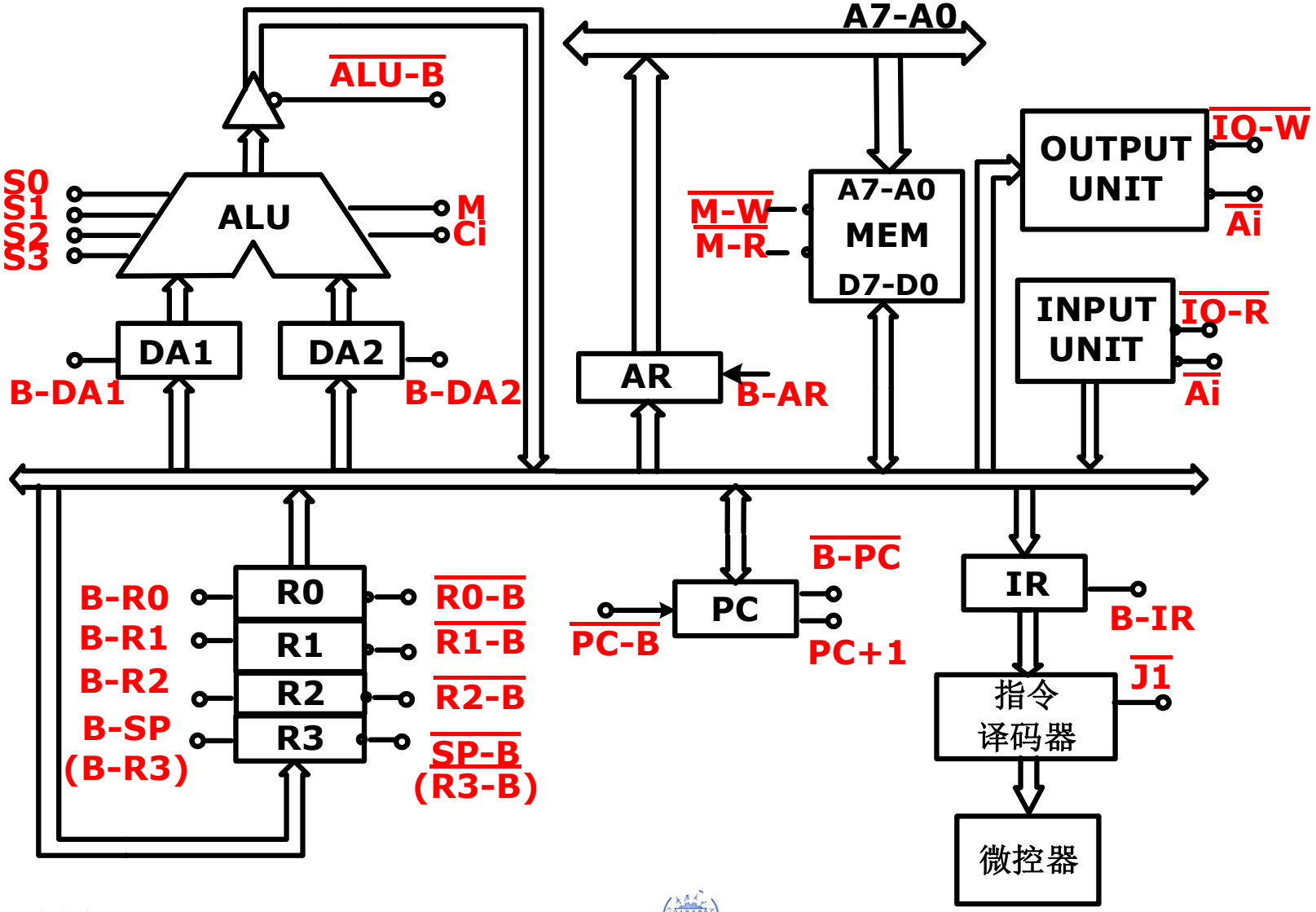
序号	控制信号	功能	序号	控制信号	功能
15	ALU-B #	运算器ALU内容送总线，低电平有效	22	R1-B #	R1内容送总线，低电平有效
16	Ci	ALU进位输入	23	R2-B #	R2内容送总线，低电平有效
17	B-R0	总线数据打入R0	24	R3-B #	R3内容送总线，低电平有效
18	B-R1	总线数据打入R1	25	I/O-W #	写（输出）I/O端口，低电平有效
19	B-R2	总线数据打入R2	26	I/O-R #	读（输入）I/O端口，低电平有效
20	B-R3	总线数据打入R3	27	Ai #	端口地址线
21	R0-B #	R0内容送总线，低电平有效	28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

4.1 模型计算机的基本

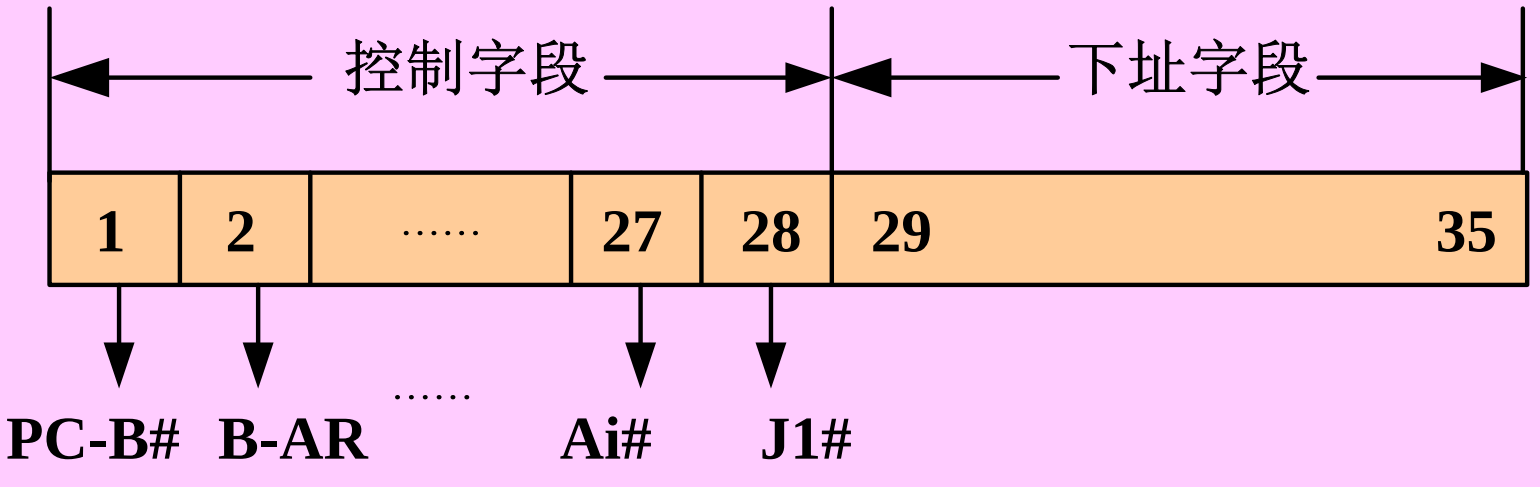
- **1、存储器读操作：分成两步：**
  - 送地址到总线，并打入地址寄存器AR；
  - 发送存储器读信号M-R# =0，启动存储器读操作，并将读出的数据从总线上接收至目的部件（例如某通用寄存器或者暂存器DA1、DA2）。
- **例如：取指令操作**
  - PC→AR，PC+1；
  - 发送M-R# =0，并RAM→IR。

4.1 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。


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
J1
指令
译码器
微控器

4.2 微指令格式-采用直接编码



- 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）。

4.3 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。

微程序设计举例—只包含2条指令

- 假设存放在存储器中的二条指令内容为：

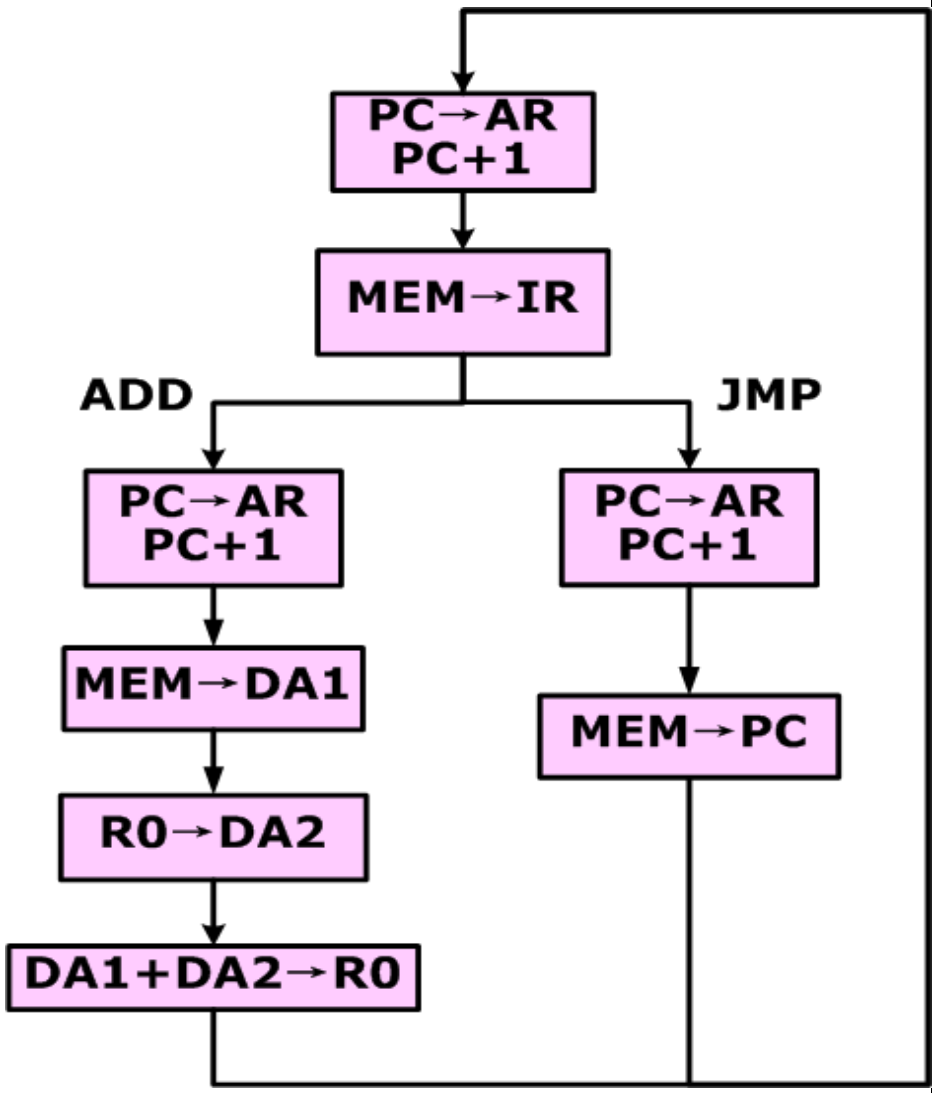
地址	机器码	助记符	功能
10H	0101 0000	Add R_0, 08H	(R_0)-08H→R_0
11H	0000 1000 (立即数)		
12H	1000 0000	JMP 10H	10H→PC
13H	0000 1010（转移地址）		

ADD 指令执行过程及控制信号

- 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

JMP 指令执行过程以及控制信号

- 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



指令执行时产生的微操作序列汇总

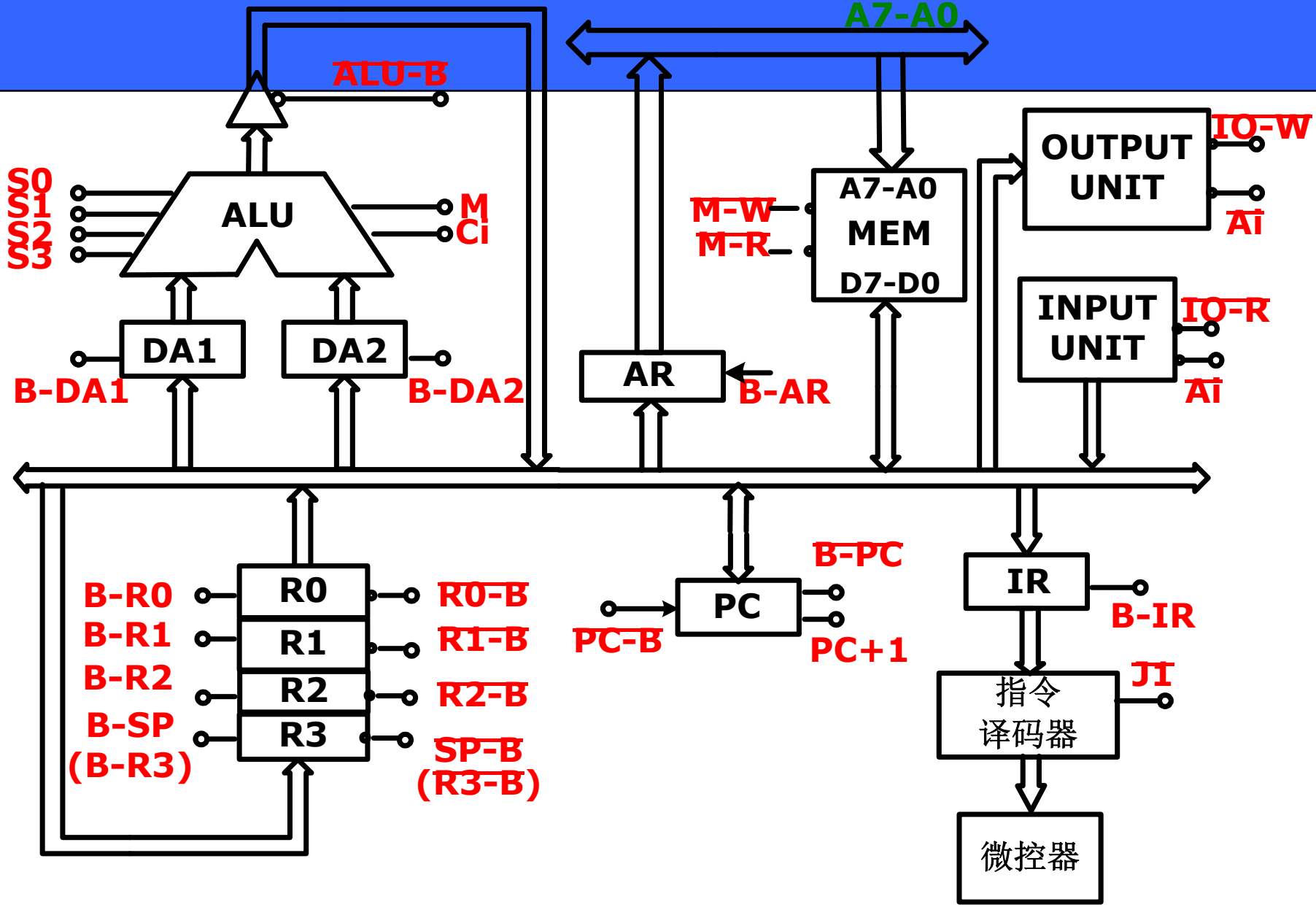
	序号	功能	发送控制信号
取指令	（1）	PC→AR,PC+1	PC-B#,B-AR,PC+1,
	（2）	MEM→IR	M-R#,B-IR,J1#
ADD	（1）	PC→AR,PC+1	PC-B#,B-AR,PC+1,
	（2）	MEM→DA1	M-R#,B-DA1
	（3）	R0→DA2	R0-B#,B-DA2
	（4）	DA1+DA2→R0	S3~S0,M,Ci1,ALU-B#,B-R0
JMP	（1）	PC→AR,PC+1	PC-B#,B-AR,PC+1,
	（2）	MEM→PC	M-R#,B-PC#

指令执行时产生的微操作序列汇总

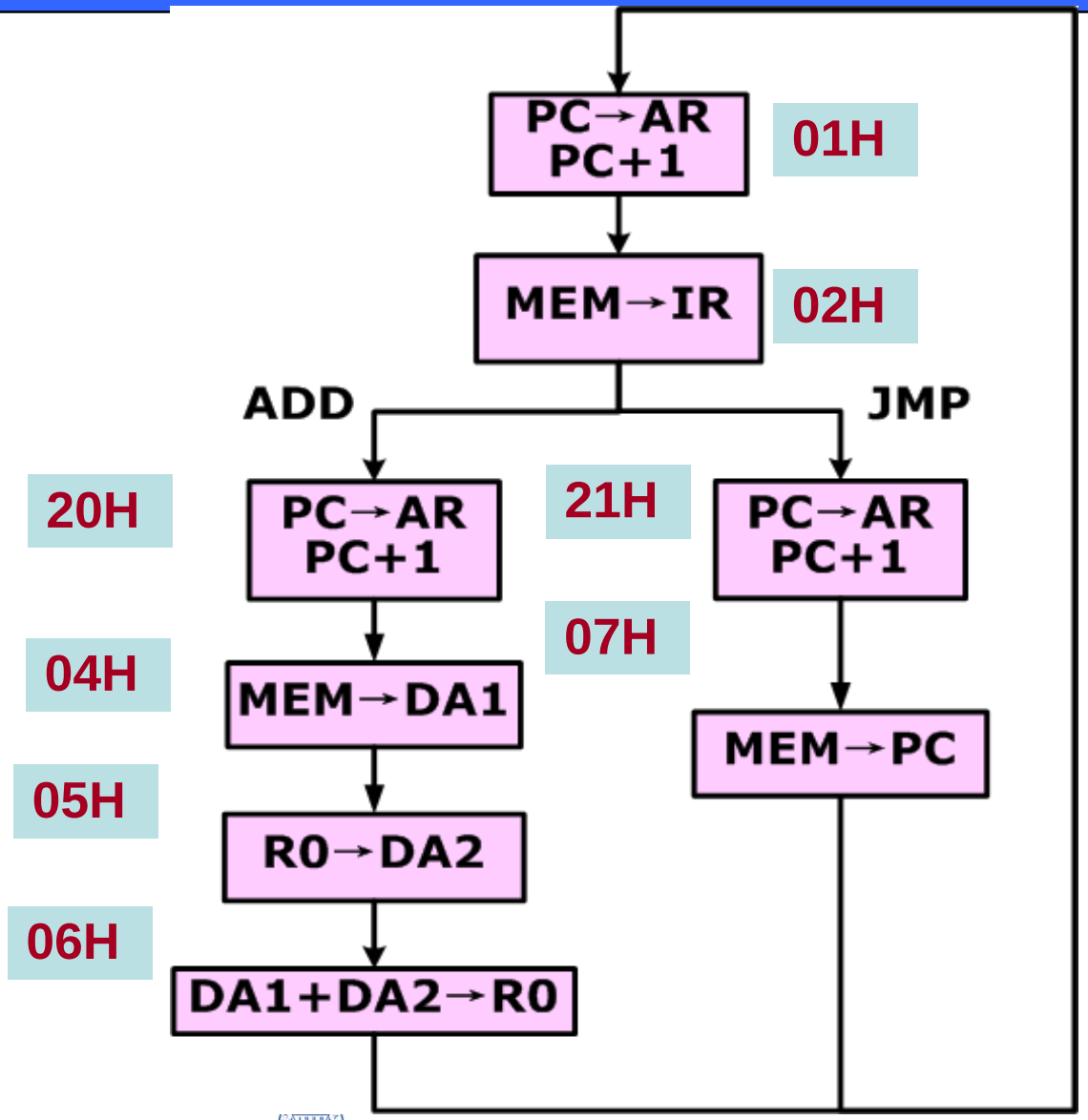
序号	微操作序列	发送控制信号
（1）	PC→AR,PC+1	PC-B#,B-AR,PC+1,
（2）	MEM→IR	M-R#,B-IR,J1#
（3）	PC→AR,PC+1	PC-B#,B-AR,PC+1,
（4）	MEM→DA1	M-R#,B-DA1
（5）	R0→DA2	R0-B#,B-DA2
（6）	DA1-DA2	S3~S0,M,Ci
（7）	ALU→R0	ALU-B#,B-R0

指令执行时产生的微操作序列汇总

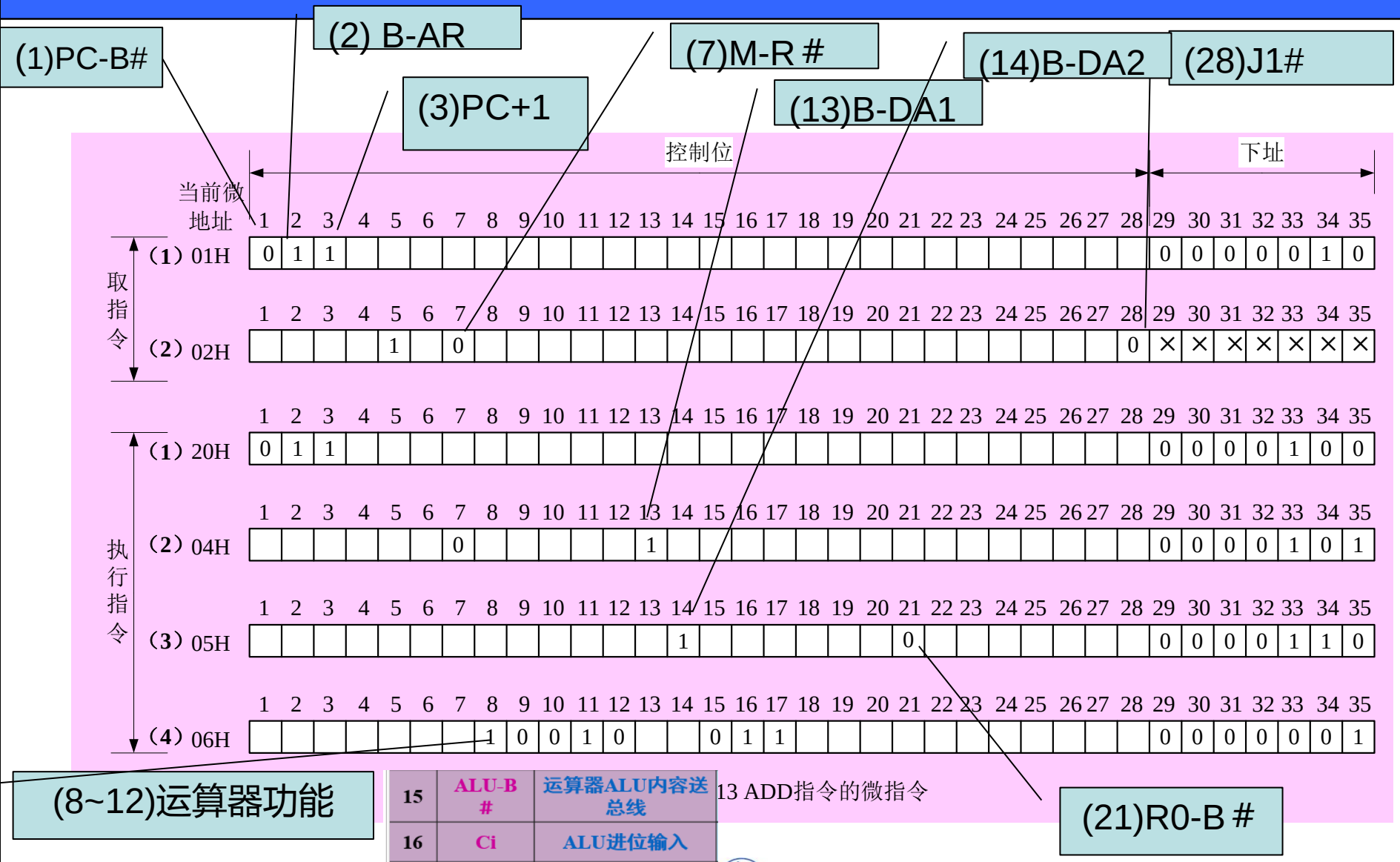
序号	微操作序列	发送控制信号
（1）	PC→AR,PC+1	PC-B#,B-AR,PC+1,
（2）	MEM→IR	M-R#,B-IR,J1#
（3）	PC→AR,PC+1	PC-B#,B-AR,PC+1,
（4）	MEM→PC	M-R#,B-PC#


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
PC-B
PC+1
IR
B-IR
JI
指令
译码器
微控器

为微指令分配微地址


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

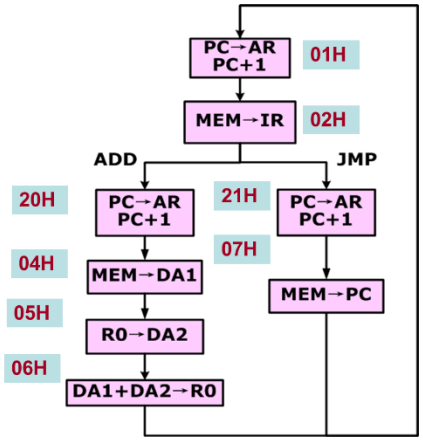
3、指令的微程序代码：ADD



15	ALU-B #	运算器ALU内容送总线
16	Ci	ALU进位输入
17	B-R0	总线数据打入R0

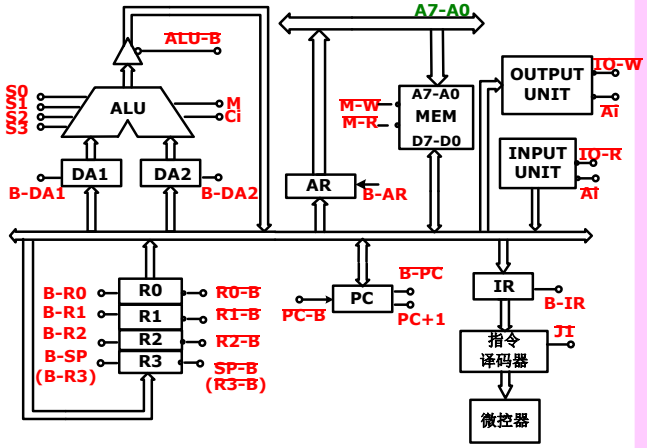
13 ADD指令的微指令

CM中加法指令对应的微程序示例



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

序号	控制信号	功能	序号	控制信号	功能
15	ALU-B5#	运算器ALU内容送总线,低电平有效	22	R1-B#	R1内容送总线，低电平有效
16	Ci	ALU进位输入	23	R2-B#	R2内容送总线，低电平有效
17	B-R0	总线数据打入R0	24	R3-B#	R3内容送总线，低电平有效
18	B-R1	总线数据打入R1	25	I/O-W#	写（输出）I/O端口，低电平有效
19	B-R2	总线数据打入R2	26	I/O-R#	读（输入）I/O端口，低电平有效
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，低电平有效	28	J1#	指令译码器译码



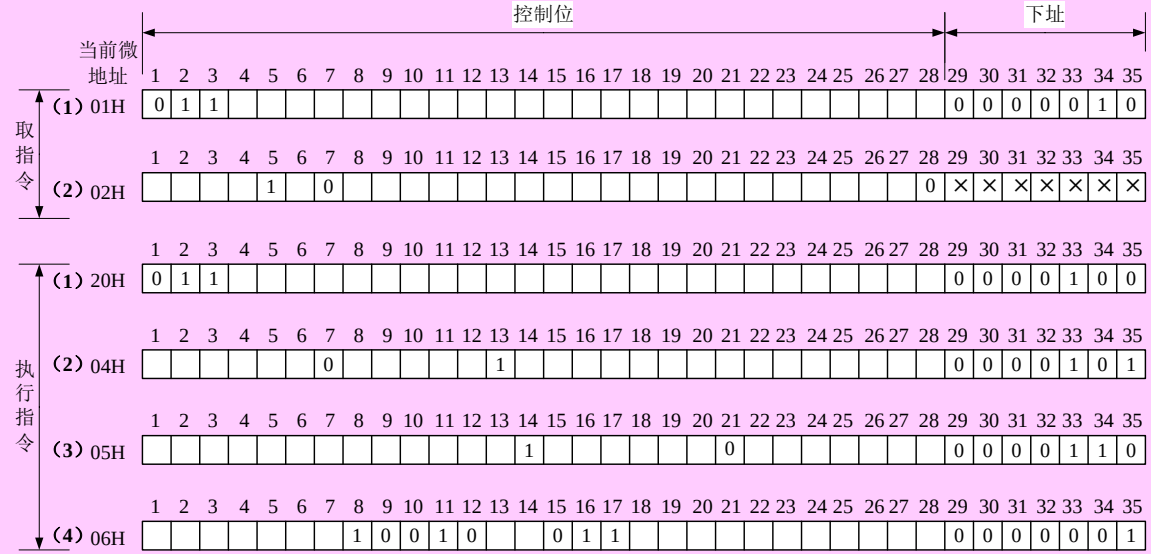
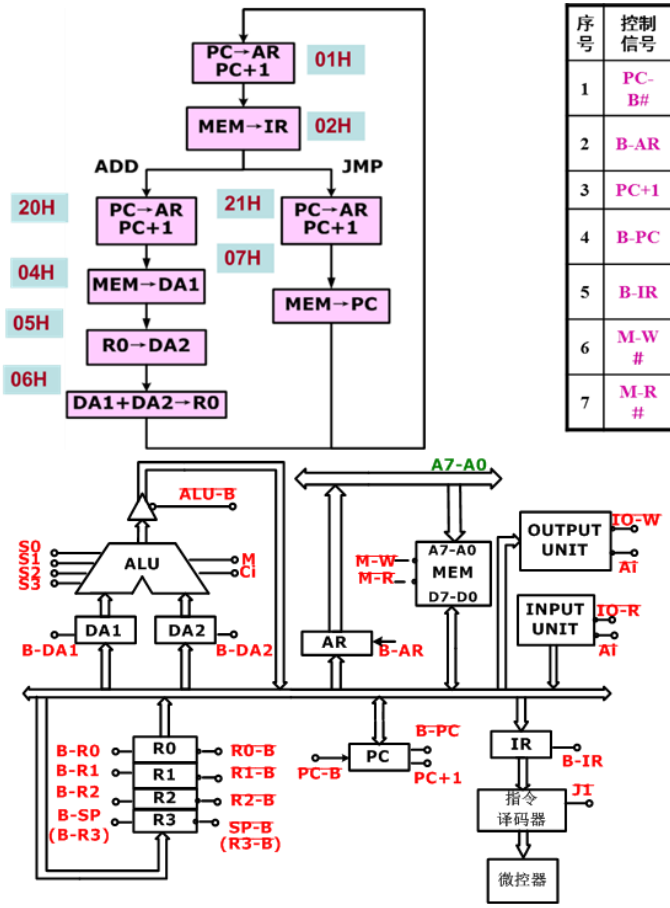


图7.13 ADD指令的微指令

主观题 10分

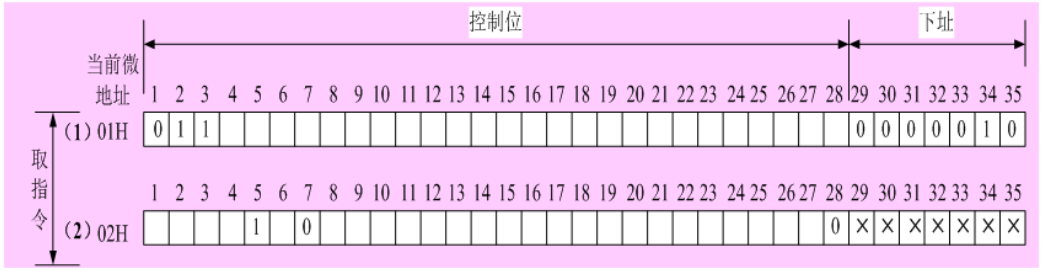
设置

请将无条件跳转指令JMP对应的微程序设计出来：假设其入口地址为21h



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

序号	控制信号	功能	序号	控制信号	功能
	ALU-B5#	运算器ALU内容送总线,低电平有效	22	R1-B#	R1内容送总线，低电平有效
16	Ci	ALU进位输入	23	R2-B#	R2内容送总线，低电平有效
17	B-R0	总线数据打入R0	24	R3-B#	R3内容送总线，低电平有效
18	B-R1	总线数据打入R1	25	I/O-W#	写（输出）I/O端口，低电平有效
19	B-R2	总线数据打入R2	26	I/O-R#	读（输入）I/O端口，低电平有效
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，低电平有效	28	J1#	指令译码器译码



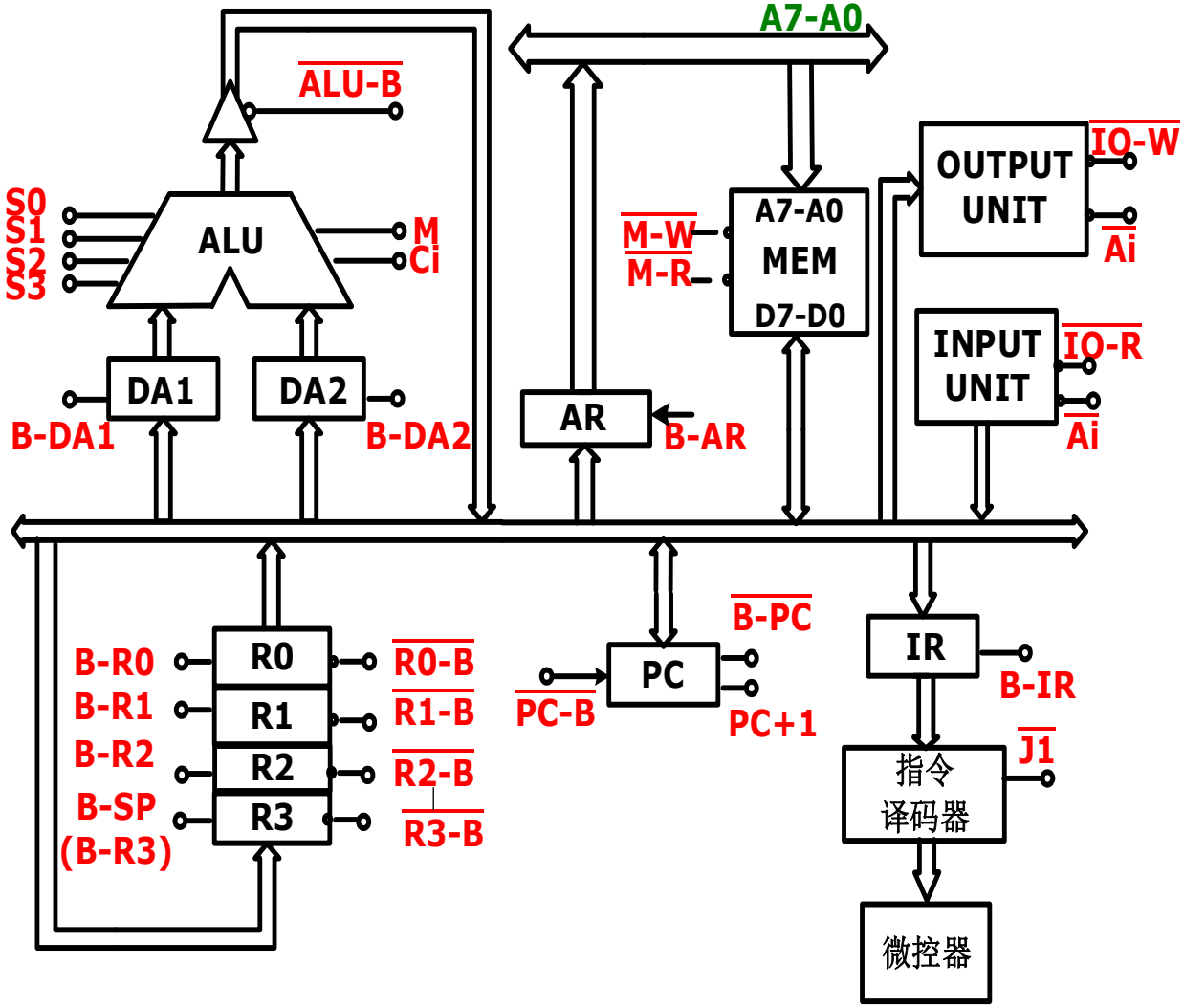
正常使用主观题需2.0以上版本雨课堂

作答

请大家思考：上述微程序控制器设计还存在什么问题?

1. 微程序控制器无法体现适应寄存器的变化

2. 无法处理按照条件的转移指令


A7-A0
ALU-B
S0
S1
S2
S3
ALU
M
Ci
M-W
M-R
A7-A0
MEM
D7-D0
OUTPUT
UNIT
IO-W
Ai
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
R3-B
B-PC
PC
PC-B
PC+1
IR
B-IR
J1
指令
译码器
微控器

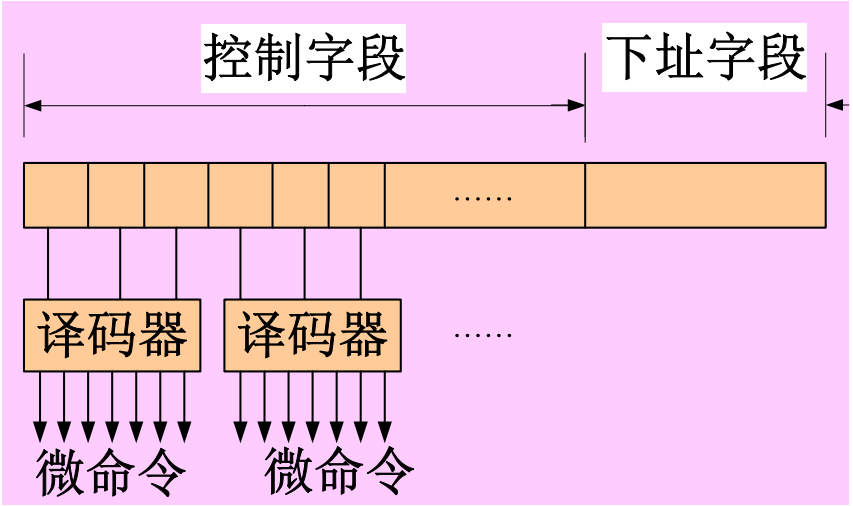
控制字段的编码方法

- **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

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



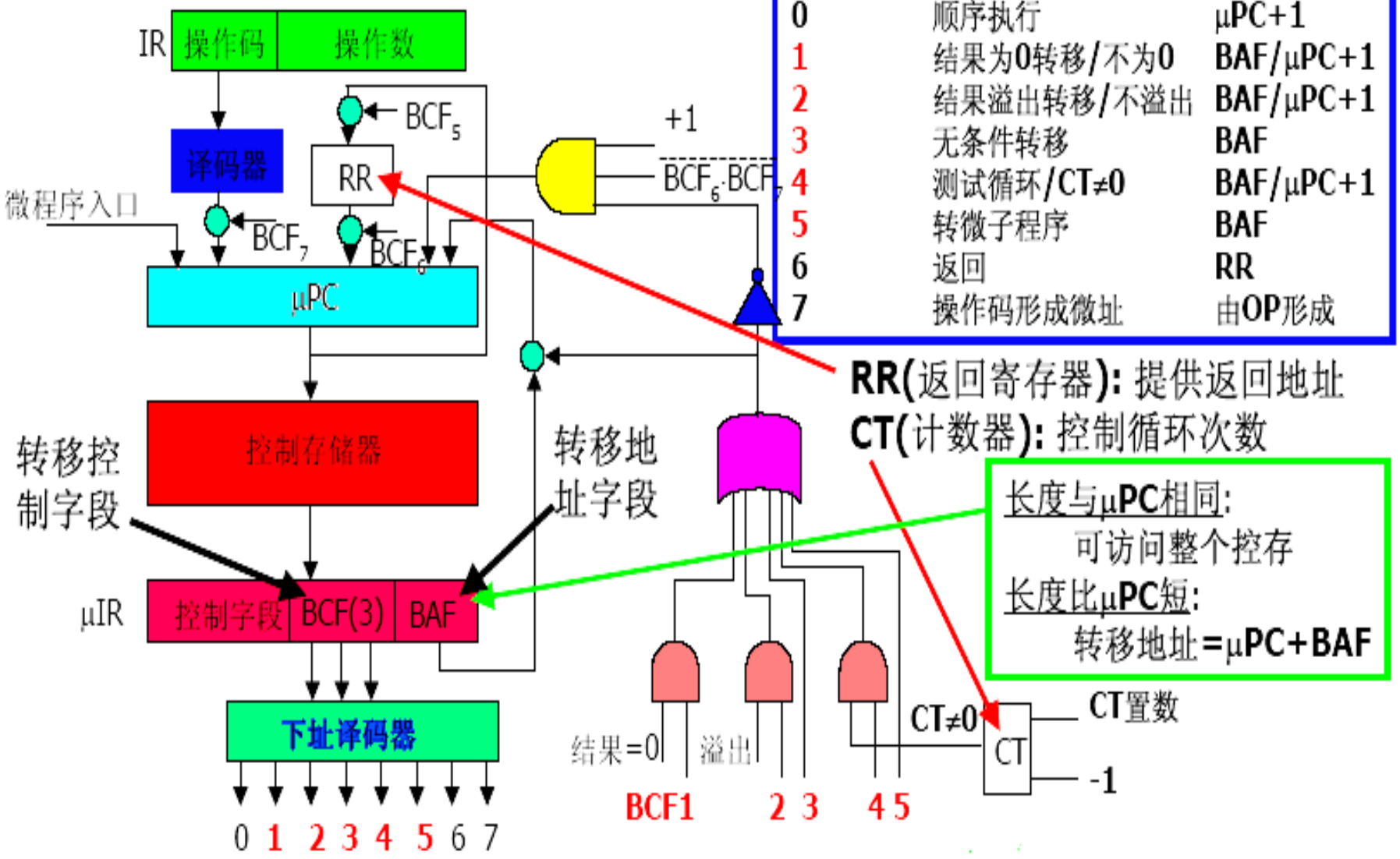
控制字段的编码方法

- **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

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。

后继地址的具体实现举例



BCF_i	名称	后继微地址
0	顺序执行	μPC+1
1	结果为0转移/不为0	BAF/μPC+1
2	结果溢出转移/不溢出	BAF/μPC+1
3	无条件转移	BAF
4	测试循环/CT≠0	BAF/μPC+1
5	转微子程序	BAF
6	返回	RR
7	操作码形成微址	由OP形成

RR(返回寄存器): 提供返回地址

CT(计数器): 控制循环次数

长度与μPC相同:
可访问整个控存

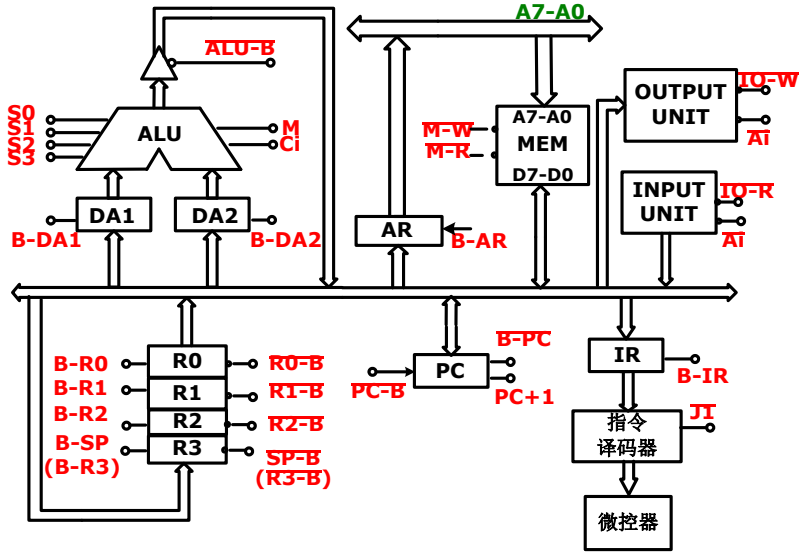
长度比μPC短:
转移地址=μPC+BAF

主观题 10分

设置

假定模型机有4个通用寄存器，地址总线为8位，指令采用双字节编码方式，请设计指令[JE #R0,#R1,NEXT] 和 JNE[JE #R0,#R1, NEXT]两条指令的格式，并设计出这两条指令对应的微操作序列，以及每个操作需要的控制信号以及其取值。

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）		



正常使用主观题需2.0以上版本雨课堂

作答

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。

控制器小结

- 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。

补充：控制器控制的方式

- 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

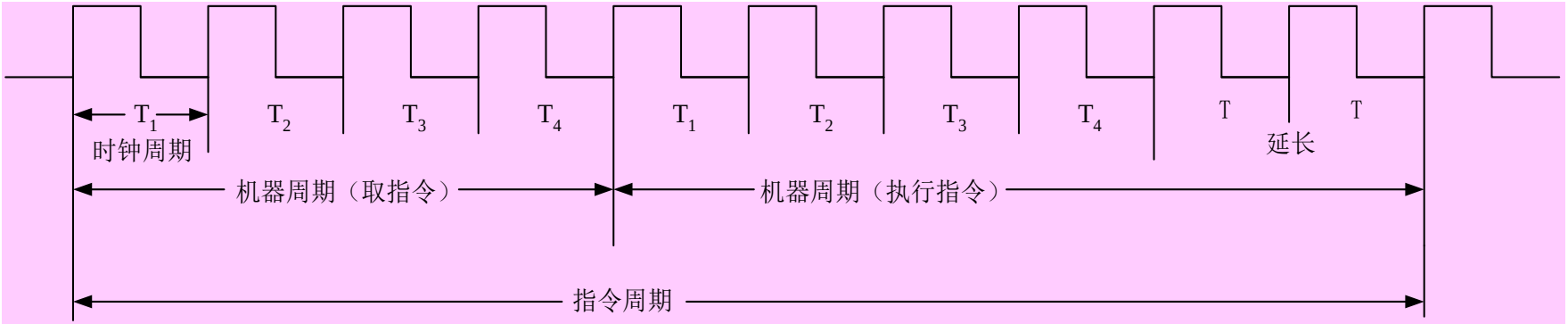


图7－8 延长机器周期的时序图

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。

4、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。

异常事件处理

异常和中断事件改变处理机正常指令的执行顺序

1、异常：指令执行过程中，由于操作非法和指令非法引起的事件。

2、陷阱：指陷阱指令，有些处理机用陷阱指令来实现操纵系统的调用。

3、中断：由外部**I/O**设备所引起。

异常事件处理包括两方面的内容：

1、异常事件的检测，当异常事件发生时，应能让处理机知道。

2、处理机应有相应的硬件机制，实现向异常事件处理程序的转移及处理完毕后回到用户程序。

如何保存PC？

1、保存PC到指定的通用寄存器中

2、保存PC到专门的寄存器中

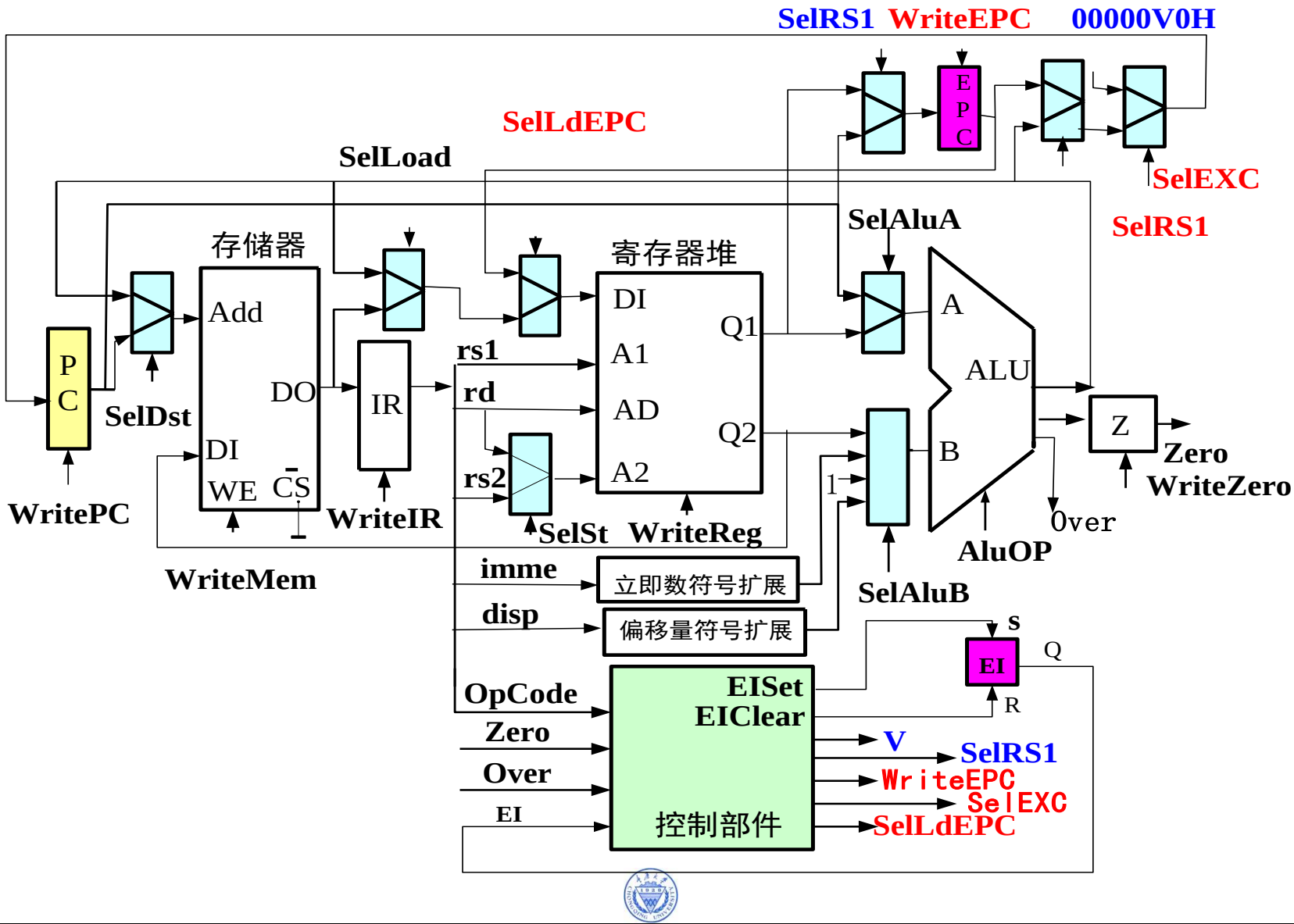
3、保存PC到存储器堆栈。

多周期处理机设计为例

新增加5条指令如下：

指令	意义
rdepc rd	读**EPC**的内容，写入寄存器**rd**
wrepc rs1	读寄存器**rs1**的内容，写入**EPC**
retex	从异常事件处理程序中返回
enai	开中断
disi	关中断

具有异常事件处理功能的处理机硬件结构



异常事件人口地址的产生方式：

地址低位0的个数取决于异常事件处理程序入口之间的间隔

向量的位数根据异常事件的种类而定

为灵活方便，可增加一个基址寄存器，异常事件处理程序可动态安排任何位置

考虑两种异常事件的“向量”

异常事件	向量V	解释
overflow	00000000B	结果溢出
invalid	00000001B	非法指令

基址为0，向量的位数8，两个异常事件处理程序入口之间的间隔为4条指令，异常事件人口地址的产生如下：

0000 0000 0000 0000 0000 V 0000 B

Concluding Remarks

- ISA influences design of datapath and control
- Datapath and control influence design of ISA
- Pipelining improves instruction throughput using parallelism
  - More instructions completed per second
  - Latency for each instruction not reduced
- Read Appendix E: A Survey of RISC Architectures for Desktop, Server, and Embedded Computers