计算机组成原理与系统结构

第七章 控制器

http://jpkc.hdu.edu.cn/computer/zcyl/dzkjdx/

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

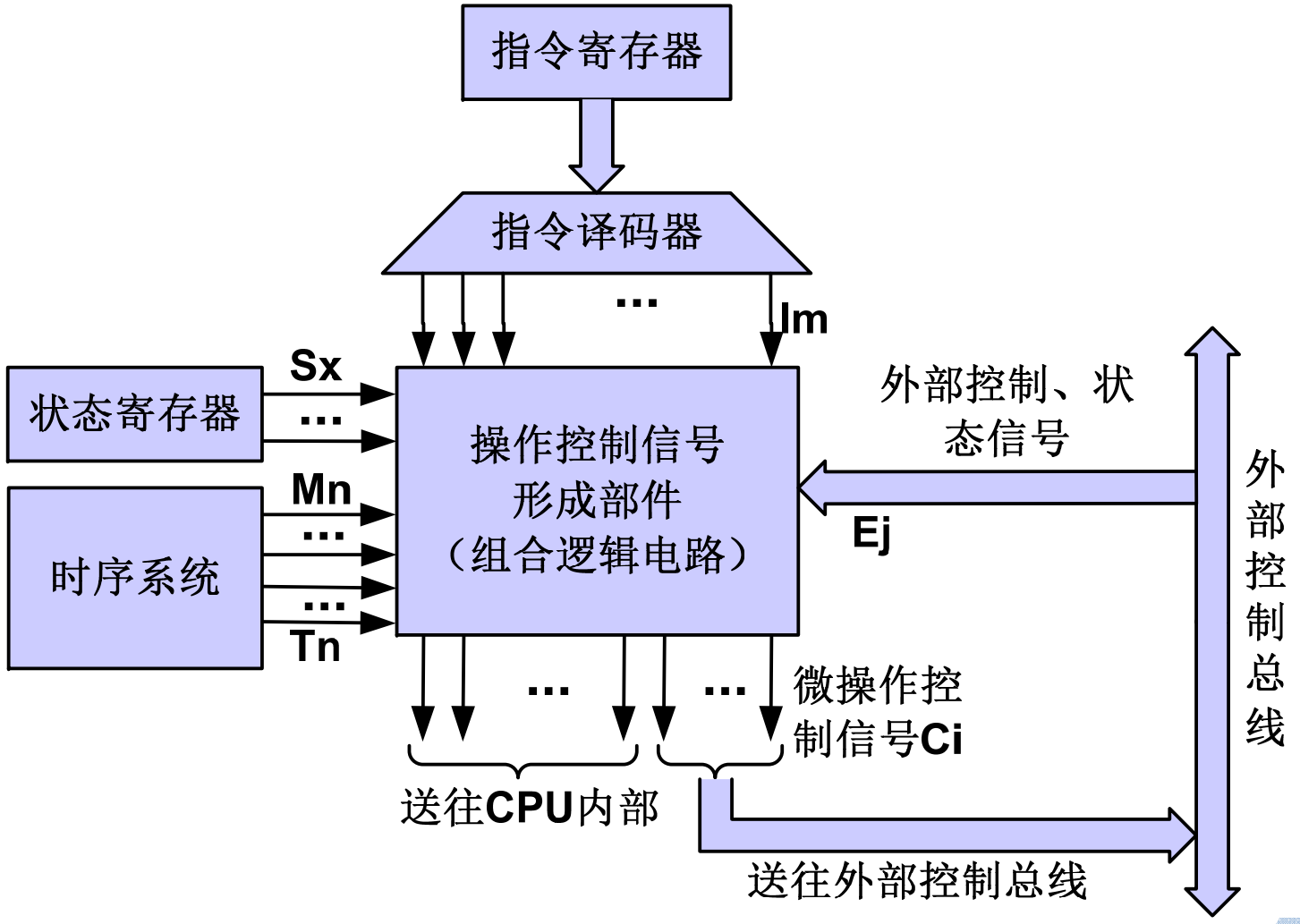
第七章　控制器

BACK

7.2 硬布线控制器

- 定义：控制器的操作控制信号形成部件是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。
- 基本原理：根据指令的功能、当前的时序及外部和内部的状态情况，按时间的顺序或者状态的转移发送一系列微操作控制信号。
- 特点：速度快，设计较为繁琐、不规整，修改、扩充较难。

7.2 硬布线控制器


指令寄存器
指令译码器
...
Im
Sx
...
状态寄存器
Mn
...
时序系统
...
Tn
操作控制信号
形成部件
（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

7.2 硬布线控制器

一 控制器的设计方法

二 硬布线控制器的结构与原理

三 多周期CPU的设计

四 硬布线控制器设计举例

五 硬布线控制器时序系统

一、控制器的设计方法

1、硬布线控制器的CPU设计步骤：

① 确定指令系统，包括每条指令的格式、功能和寻址方式， 分配操作码。

② 围绕着指令系统的实现，确定CPU的内部结构与数据通路，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，时序系统的构成。

③ 分析每条指令的执行过程，按机器周期顺序或者有限状态，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。

一、控制器的设计方法

2、微程序控制器的CPU设计步骤：

① 确定指令系统，包括每条指令的格式、功能和寻址方式，分配操作码。

② 围绕着指令系统的实现，确定CPU的内部结构和数据通路，包括运算器的功能和组成，微程序控制器的结构、组成及各部件的连接方式和数据通路，时序系统的构成。

③ 在以上基础上，分析每条指令的执行过程，画出指令系统的微程序流程图。

一、控制器的设计方法

2、微程序控制器的CPU的设计步骤：

④ 根据CPU的结构，写出每条微指令所发送的微操作控制信号序列。

⑤ 结合微程序控制器的结构、微操作控制信号序列和控制存储器容量，设计微指令格式。

⑥ 分配微程序流程图中各微指令的微地址，并编写微指令代码。

⑦ 将所有的微指令代码装入控制存储器的相应单元。

一、控制器的设计方法

3、需要注意两点：

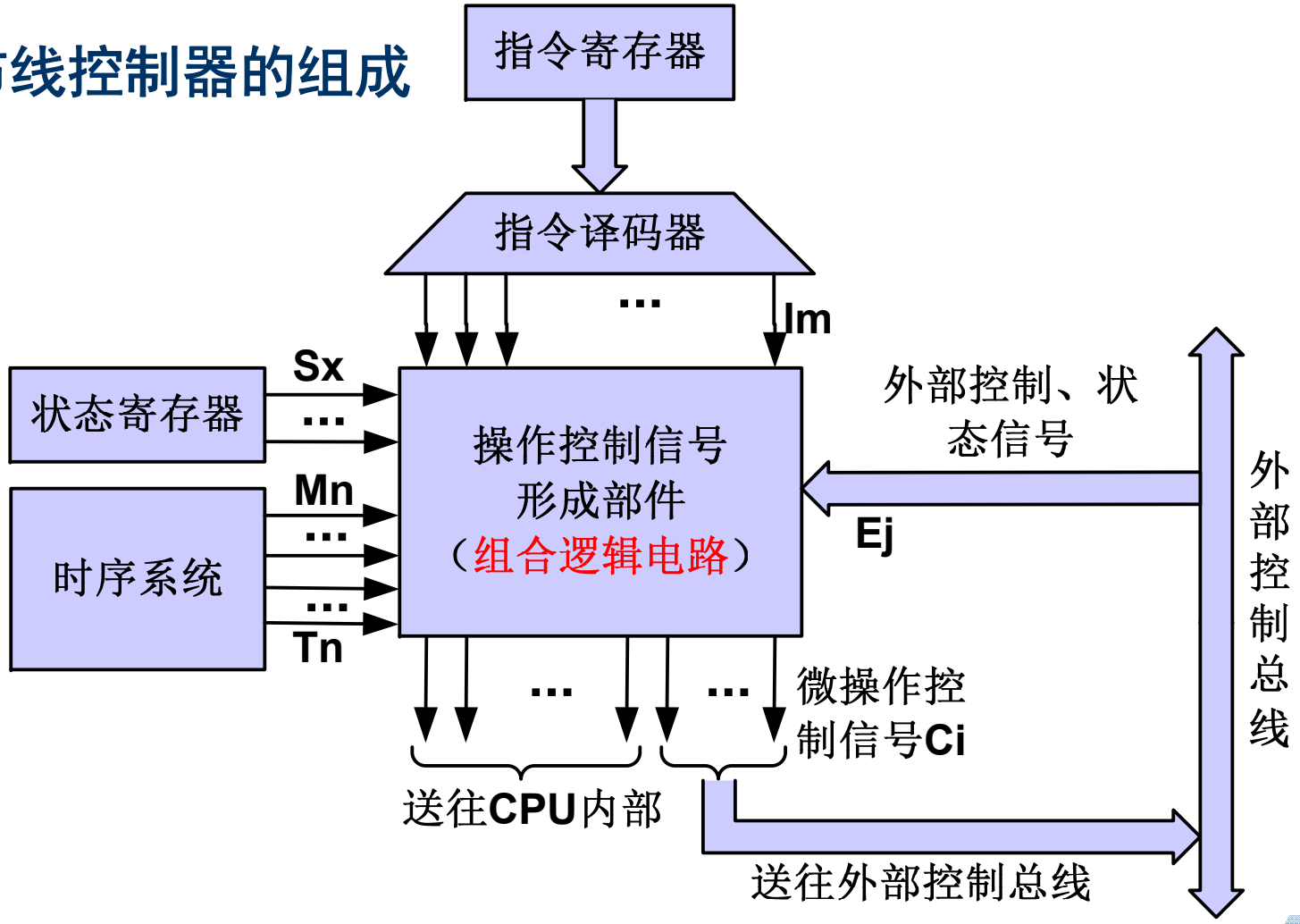
① 在一个CPU中，既可以使用硬布线控制器，也可以同时使用微程序控制器

② 在上述设计步骤中，不是单向线性的过程，而可能会交错进行

指令系统是软硬件的交接面，它既是硬件设计者的设计依据和设计目标，也是软件设计者控制计算机的惟一依据。

二、硬布线控制器的结构与原理

硬布线控制器的组成



二、硬布线控制器的结构与原理

- 组合逻辑电路的输入：
  - ① 指令译码器译码产生的指令信息I_m
  - ② 时序系统产生的机器周期信号M_n和节拍信号T_n：用于定序，相当于有限状态机中的状态序列；
  - ③ 状态寄存器的状态信号S_x
  - ④ 外部控制、状态信号E_j
- 输出：微操作控制信号Ci
  - 一部分为CPU外部控制信号：构成控制总线
  - 另一部分为CPU内部的微操作控制信号。

二、硬布线控制器的结构与原理

- 从逻辑函数的角度来看，微操作控制信号Ci是4种输入信号的函数：

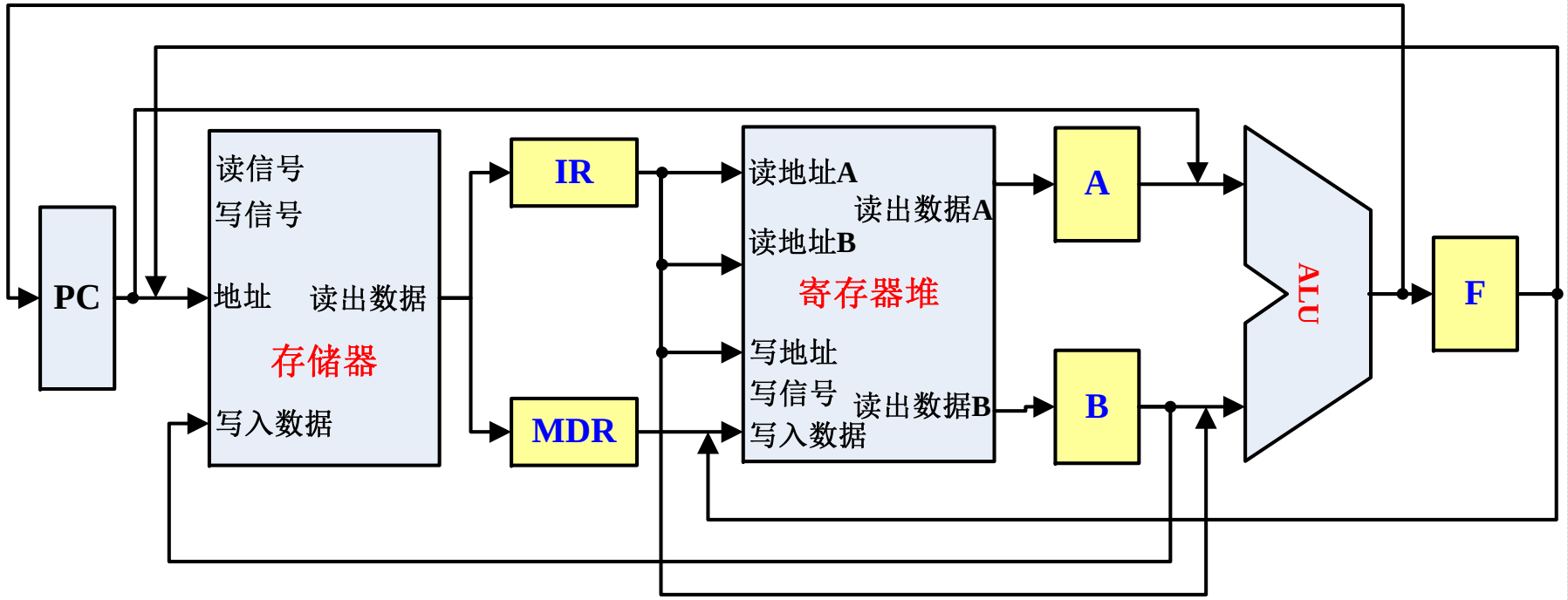
$$C_i=f_i（I_m，M_n，T_n，S_x，E_j）$$

- 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C_i的逻辑函数f_i的过程。

三、多周期CPU的设计

- 指令周期可包含多个时钟周期，每个时钟周期执行指令的一步操作。
- 特征：
  - 功能部件可在单指令周期中共享（多次使用）；
  - 不同指令所占用的时钟周期可以不同。

1、多周期CPU的数据通路



1、多周期CPU的数据通路

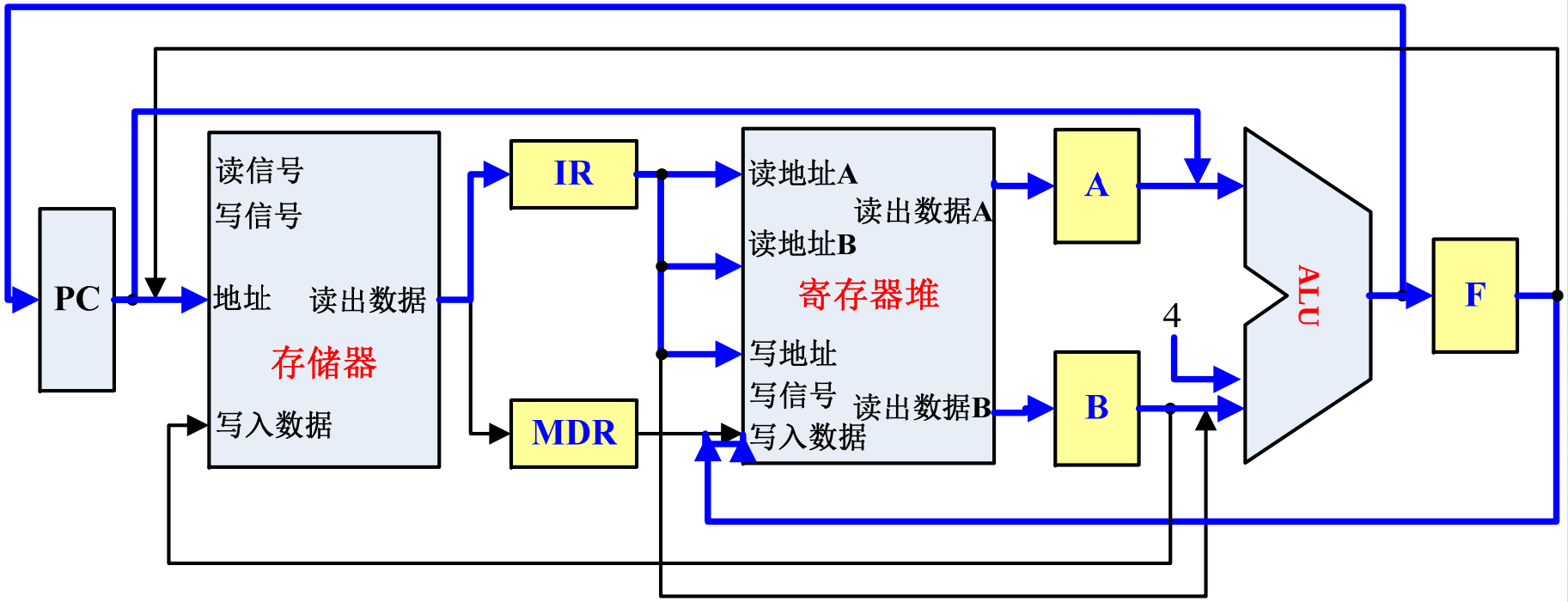
- 与单周期CPU的差异：
  - 指令存储器和数据存储器合二为一；
  - 节省了2个加法器：PC+4的加法器和相对转移地址加法器，使用ALU完成PC的自增和转移地址的计算；
  - 添加了5个专用、附加寄存器：
    - 指令寄存器IR：存储从主存读出的指令码；
    - 存储器数据寄存器MDR：存储从主存读出的数据；
    - 暂存器A和B：存储从寄存器堆读出的A口和B口数据；
    - 暂存器F：存储ALU的运算结果（数据或地址）
- 为什么要添加专用或附加寄存器？
  - 每个时钟周期下跳沿，均需将操作结果存入暂存器或者专用寄存器保存，以便下一时钟周期使用。

2、MIPS指令子集的执行过程

- （1）R型指令
- （2）I型访存指令
- （3）I型分支指令
- （4）J型跳转指令

（1）R型指令

❖R型指令数据通路：

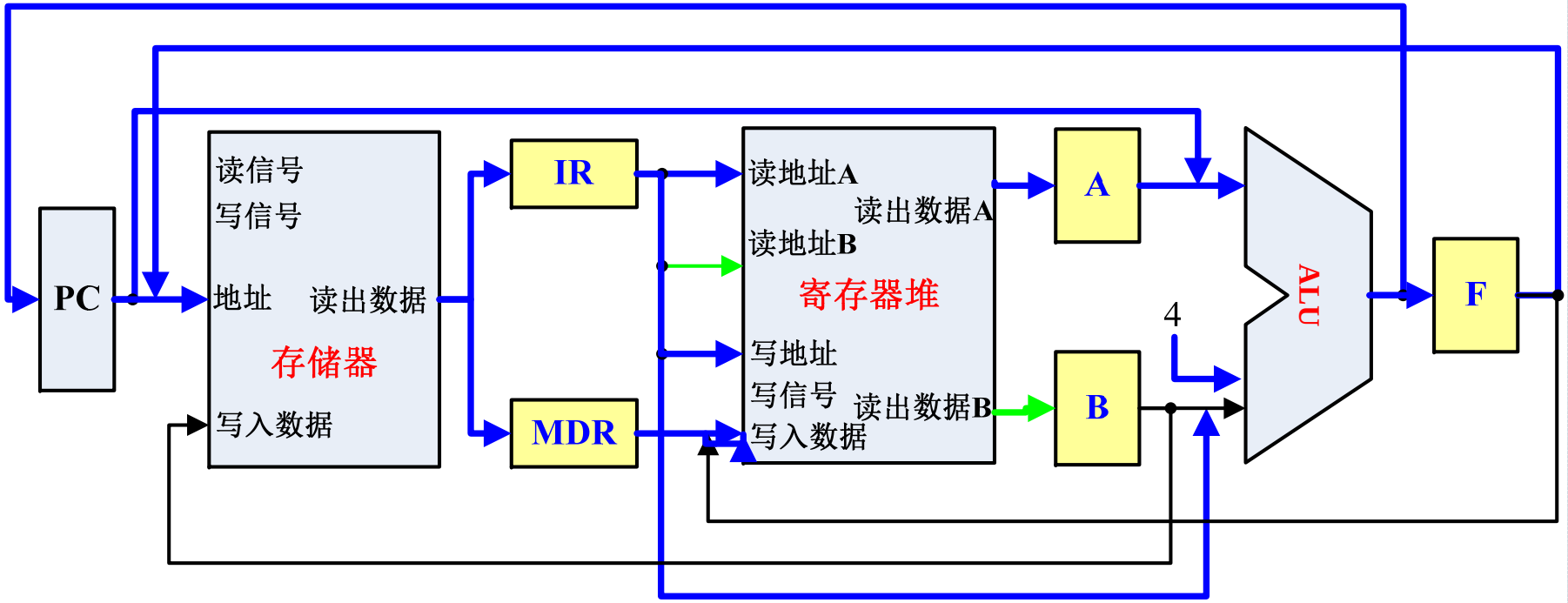


（1）R型指令

- R型指令执行过程：
- ①取指令，PC自增
  - Mem[PC]→IR，PC+4→PC
- ②读寄存器
  - Reg[rs]→A，Reg[rt]→B
- ③ALU运算
  - A （op）B→F
- ④写结果寄存器
  - F→ Reg[rd]

（2）I型访存指令

- 取数指令lw数据通路：

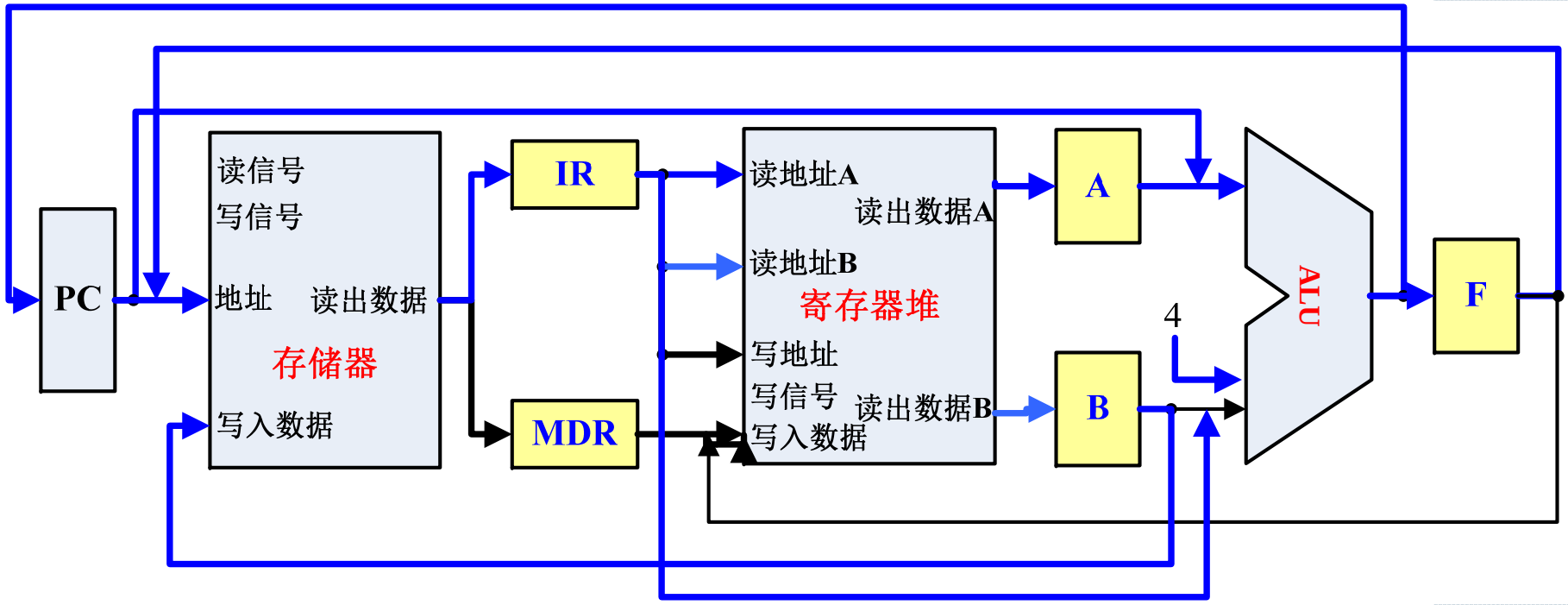


（2）I型访存指令

- 取数指令lw执行过程：
- ①取指令，PC自增
  - Mem[PC]→IR，PC+4→PC
- ②读寄存器
  - Reg[rs]→A，Reg[rt]→B
- ③计算有效地址EA
  - A + offset →F
- ④读存储器
  - Mem[F]→MDR
- ⑤写结果寄存器
  - MDR→ Reg[rt]

（2）I型访存指令

❖ 存数指令sw数据通路：

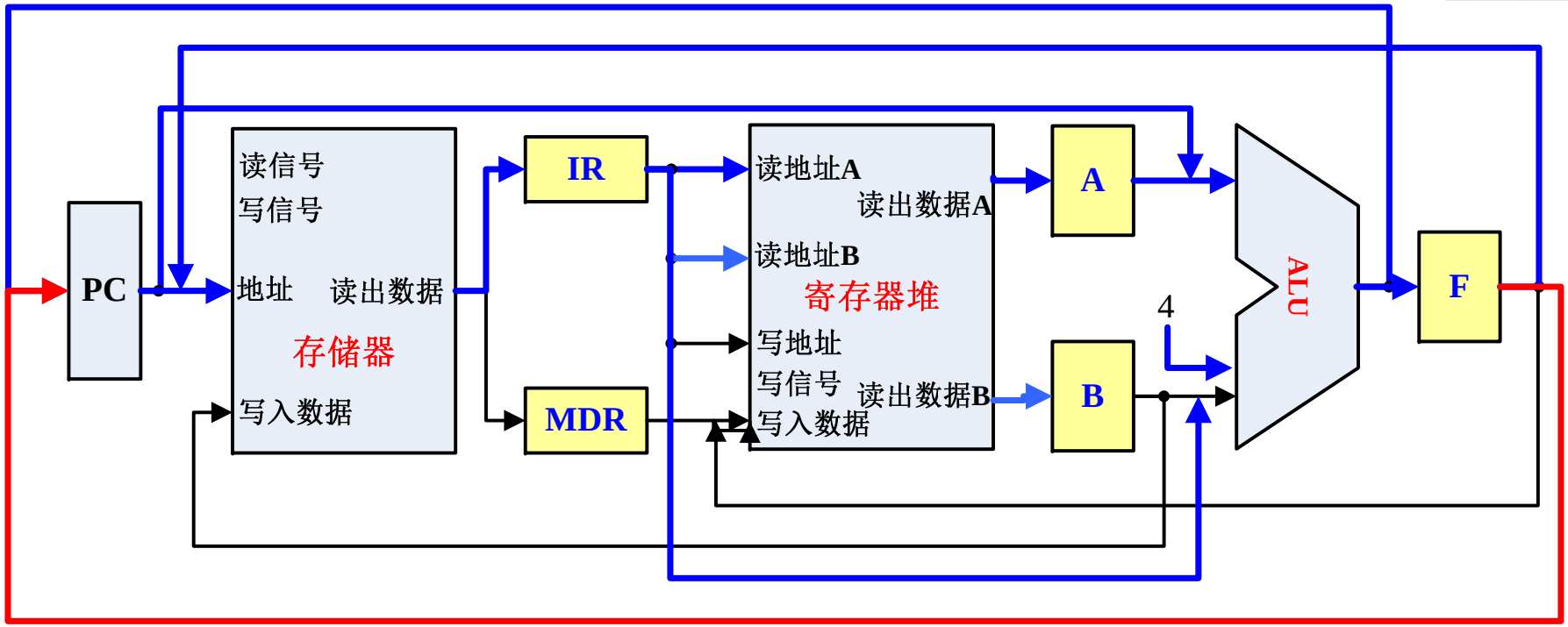


（2）I型访存指令

- 存数指令sw执行过程：
- ①取指令，PC自增
  - Mem[PC]→IR，PC+4→PC
- ②读寄存器
  - Reg[rs]→A，Reg[rt]→B
- ③计算有效地址EA
  - A + offset →F
- ④写存储器
  - B→Mem[F]

（3）I型分支指令

- 分支指令beq数据通路：
  - 红色线：新添加的数据通路

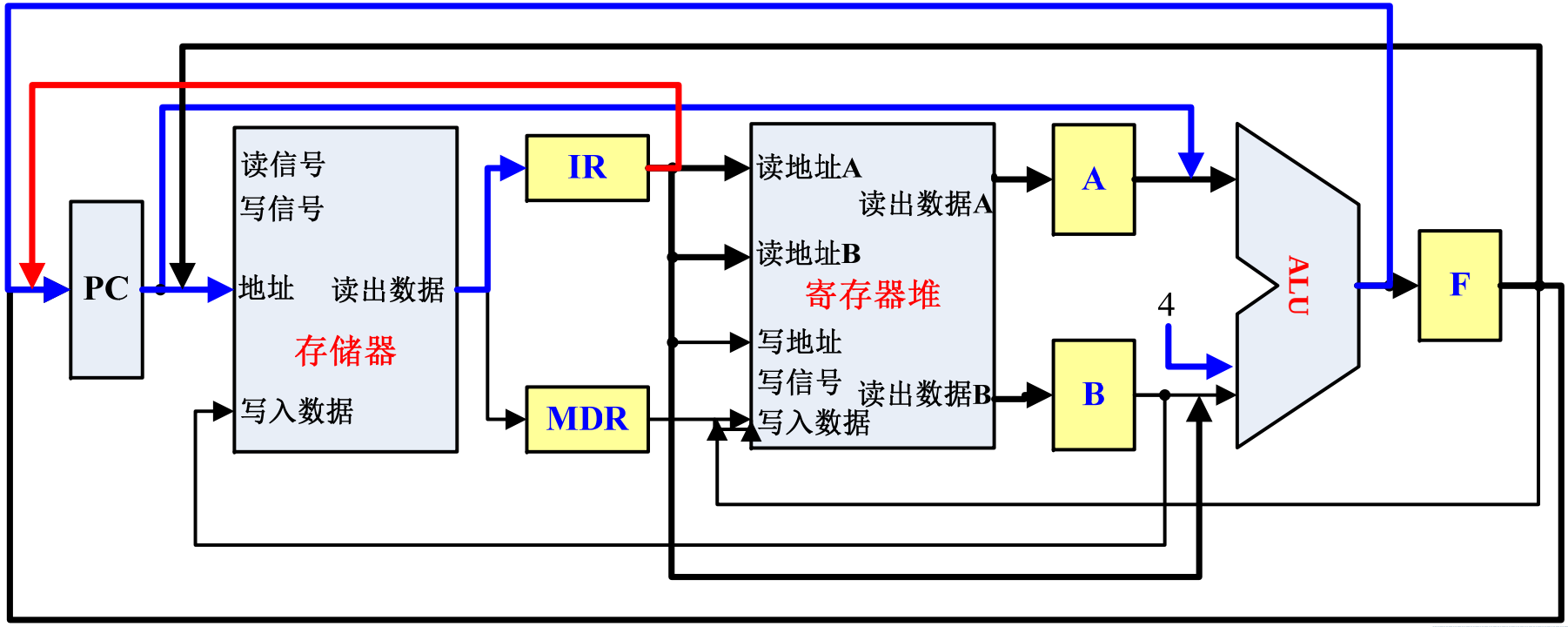


（3）I型分支指令

- 分支指令beq执行过程：
- ①取指令，PC自增
  - Mem[PC]→IR，PC+4→PC
- ②读寄存器和转移地址计算
  - Reg[rs]→A，Reg[rt]→B
  - PC+offset*4→F
- ③完成分支
  - A - B，产生ZF
  - zero=1，则F→PC
  - zero=0，空操作

（4）J型跳转指令

- 跳转指令J数据通路：
  - 红色线：新添加的数据通路

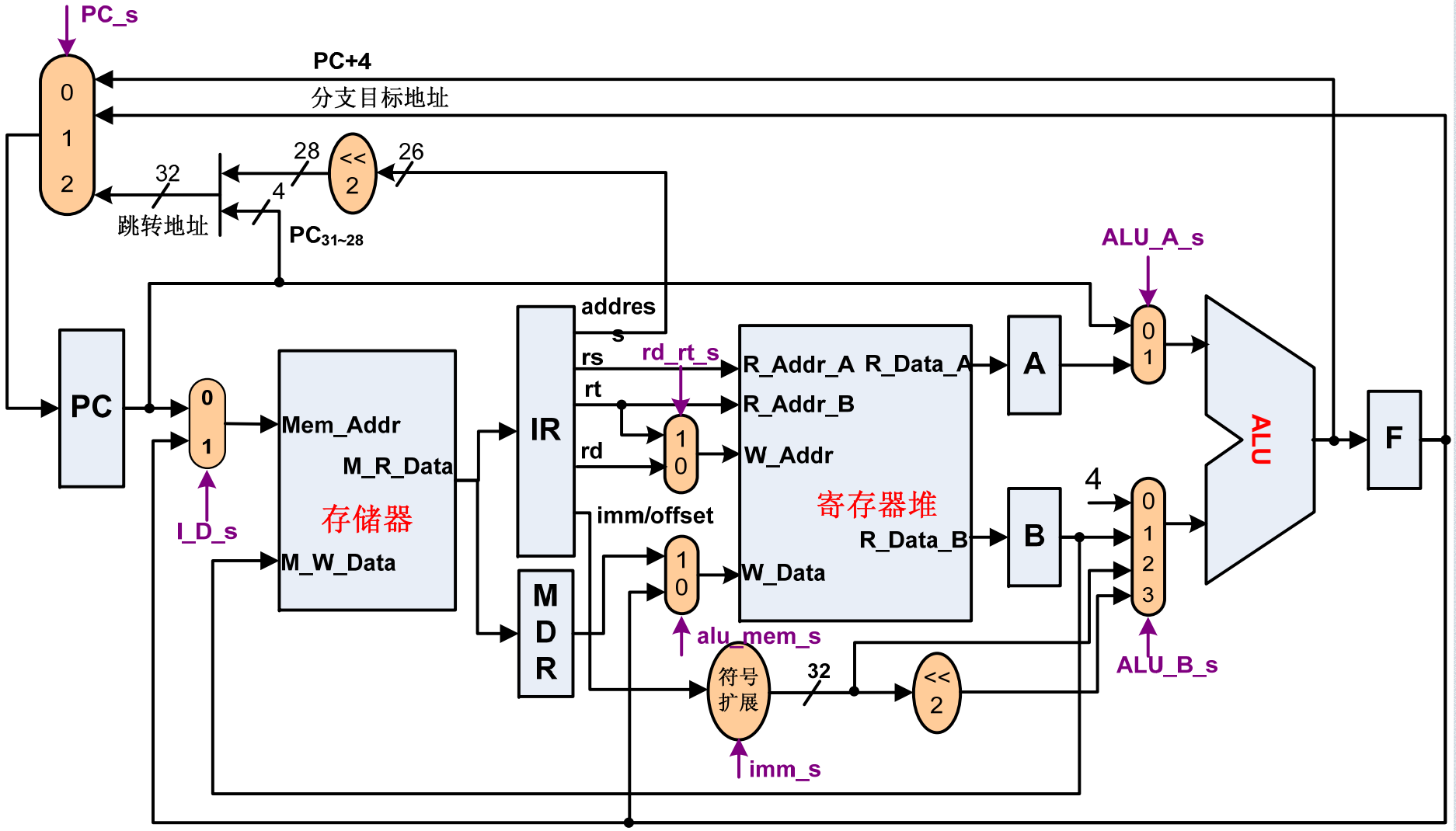


（4）J型跳转指令

- 跳转指令J执行过程：
- ①取指令，PC自增
  - Mem[PC]→IR，PC+4→PC
- ②完成跳转
  - {PC[31:28]，address，2‘b00}→PC

3、多周期CPU的实现

（1）细化数据通路的多路选择器：



3、多周期CPU的实现

❖ 多路选择器的控制信号： 图中紫色

信号	作用	值	选择操作
PC_s	选择更新PC的数据源	00	ALU的输出（PC+4）送PC
		01	F暂存器的内容（分支目标地址）送PC
		10	跳转目标地址送PC
I_D_s	选择主存地址来源	0	PC提供主存地址（指令地址）
		1	F暂存器提供主存地址（数据地址）
rd_rt_s	选择寄存器写地址	0	写入指令rd字段指定的寄存器
		1	写入指令rt字段指定的寄存器
alu_mem_s	选择寄存器写数据	0	选择暂存器F的内容写入寄存器
		1	选择MDR的内容写入寄存器

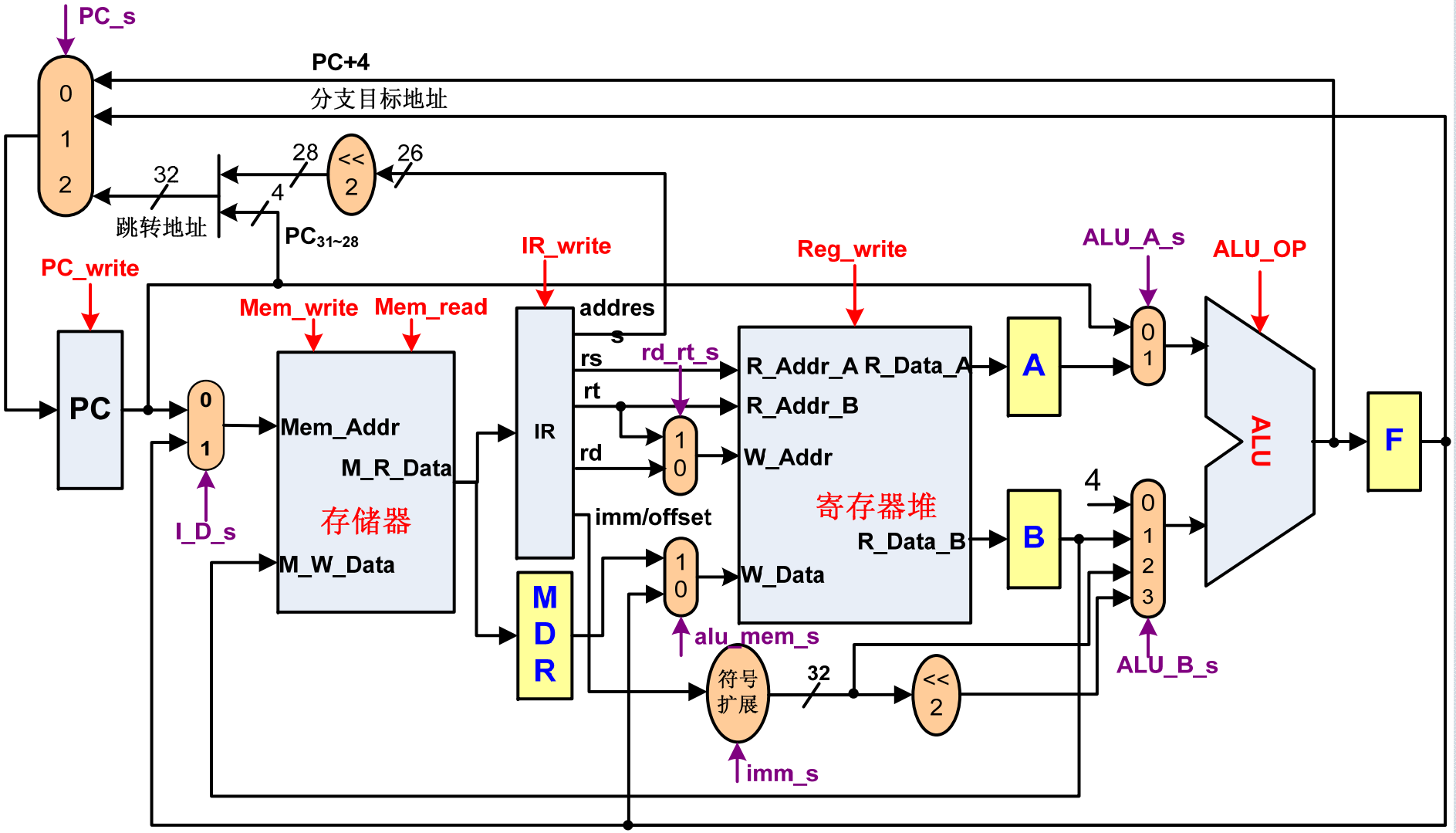
3、多周期CPU的实现

- 多路选择器的控制信号： 图中紫色

信号	作用	值	选择操作
imm_s	选择符号/无符号扩展	0	对指令的imm/offset字段进行无符号扩展
		1	对指令的imm/offset字段进行符号扩展
ALU_A_s	选择ALU的A操作数	0	PC的内容作为ALU的A操作数
		1	A暂存器的内容作为ALU的A操作数
ALU_B_s	选择ALU的B操作数	00	常数4作为ALU的B操作数
		01	暂存器B的内容作为ALU的B操作数
		10	指令的imm/offset字段扩展后的32位数作为ALU的B操作数
		11	指令的imm/offset字段扩展后的32位数又左移2为得到的数作为ALU的B操作数

3、多周期CPU的实现

❖ （2）添加部件的控制信号：读、写信号 红色



3、多周期CPU的实现

- 部件控制信号：均为高电平有效

部件	控制信号	作用	有效边沿
PC	PC_write	写入PC	指定周期下跳沿
存储器	Mem_write	存储器写操作	指定周期下跳沿
	Mem_read	存储器读操作	电平控制
IR	IR_write	写入IR	指定周期下跳沿
寄存器堆	Reg_write	写入寄存器	指定周期下跳沿
MDR	无	装入存储器读出数据	每个时钟周期下跳沿
A	无	装入rs寄存器读出数据	
B	无	装入rt寄存器读出数据	
F	无	装入ALU运算结果	

3、多周期CPU的实现

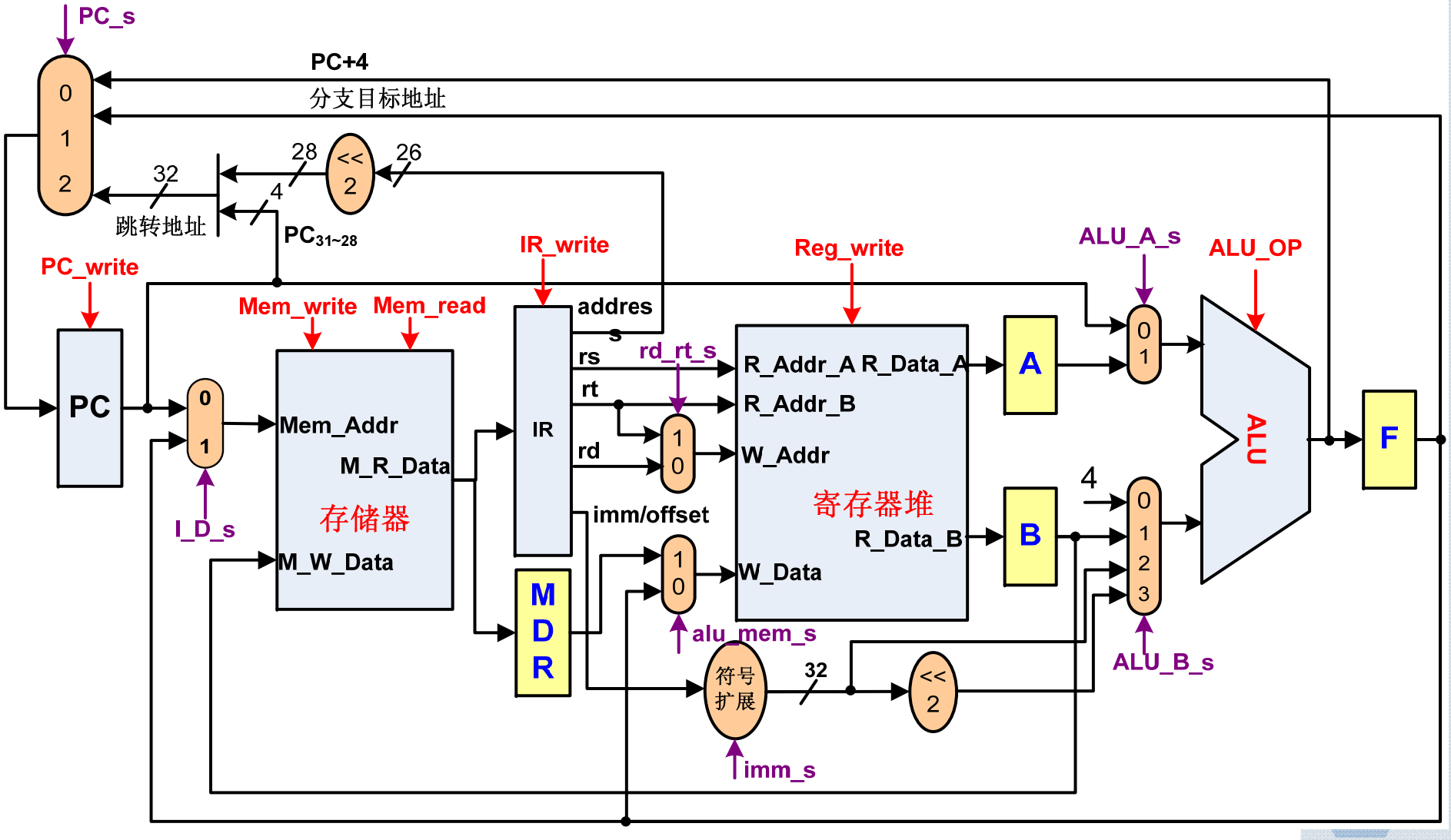
- 如何产生部件控制信号？
- 由操作控制信号形成部件产生。
- 其实现方法：有两种：
  - 组合逻辑电路：硬布线控制器；
    - 输入：IR的OP、func字段、ALU的zero标志、周期（状态）信号；
    - 输出：各个控制信号
  - 存储逻辑电路：微程序控制器；

四、硬布线控制器设计举例

- 目标：设计实现一个MIPS的多周期CPU
- 1、确定指令系统：
  - MIPS核心指令子集
    - R型指令（8条）；
    - I型访存指令（2条：lw、sw）；
    - I型分支指令（1条：beq）；
    - J型跳转指令（1条：J）；

四、硬布线控制器设计举例

❖2、确定系统结构与数据通路：同上节；



四、硬布线控制器设计举例

❖ 3、分析每条指令的执行过程，写出发送的微操作控制信号序列

（1）R型指令		
时钟周期	操作	发送控制信号
M0	Mem[PC]→IR, PC+4→PC	I_D_s=0,Mem_read,IR_write; ALU_A_s=0, ALU_B_s=00, PC_s=00, PC_write;
M1	Reg[rs]→A,Reg[rt]→B	无
M2	A （op）B→F	ALU_A_s=1, ALU_B_s=01, ALU_OP=***;
M3	F→ Reg[rd]	rd_rt_s=0,alu_mem_s=0, Reg_write;

四、硬布线控制器设计举例

❖ 3、分析每条指令的执行过程，写出发送的微操作控制信号序列

（2）I型访存指令：lw		
时钟周期	操作	发送控制信号
M0	Mem[PC]→IR, PC+4→PC	I_D_s=0,Mem_read,IR_write; ALU_A_s=0, ALU_B_s=00, PC_s=00, PC_write;
M1	Reg[rs]→A,Reg[rt]→B	无
M2	A + offset →F	ALU_A_s=1, ALU_B_s=10, imm_s=1,ALU_OP=100;
M3	Mem[F]→MDR	I_D_s=1,Mem_read;
M4	MDR→ Reg[rt]	rd_rt_s=1,alu_mem_s=1, Reg_write;

四、硬布线控制器设计举例

❖ 3、分析每条指令的执行过程，写出发送的微操作控制信号序列

（2）I型访存指令：sw		
时钟周期	操作	发送控制信号
M0	Mem[PC]→IR, PC+4→PC	I_D_s=0,Mem_read,IR_write; ALU_A_s=0, ALU_B_s=00, PC_s=00, PC_write;
M1	Reg[rs]→A,Reg[rt]→B	无
M2	A + offset →F	ALU_A_s=1, ALU_B_s=10, imm_s=1,ALU_OP=100;
M3	B → Mem[F]	I_D_s=1,Mem_write;

四、硬布线控制器设计举例

❖ 3、分析每条指令的执行过程，写出发送的微操作控制信号序列

（3）I型分支指令：beq		
时钟周期	操作	发送控制信号
M0	Mem[PC]→IR, PC+4→PC	I_D_s=0,Mem_read,IR_write; ALU_A_s=0, ALU_B_s=00, ALU_OP=100,PC_s=00, PC_write;
M1	Reg[rs]→A,Reg[rt]→B PC+offset*4→F	ALU_A_s=0, ALU_B_s=11, ALU_OP=100,imm_s=1;
M2	A - B，产生zero zero=1，则F→PC zero=0，空操作	ALU_A_s=1, ALU_B_s=01, ALU_OP=101; zero=1:PC_s=01,PC_write

四、硬布线控制器设计举例

- 3、分析每条指令的执行过程，写出发送的微操作控制信号序列

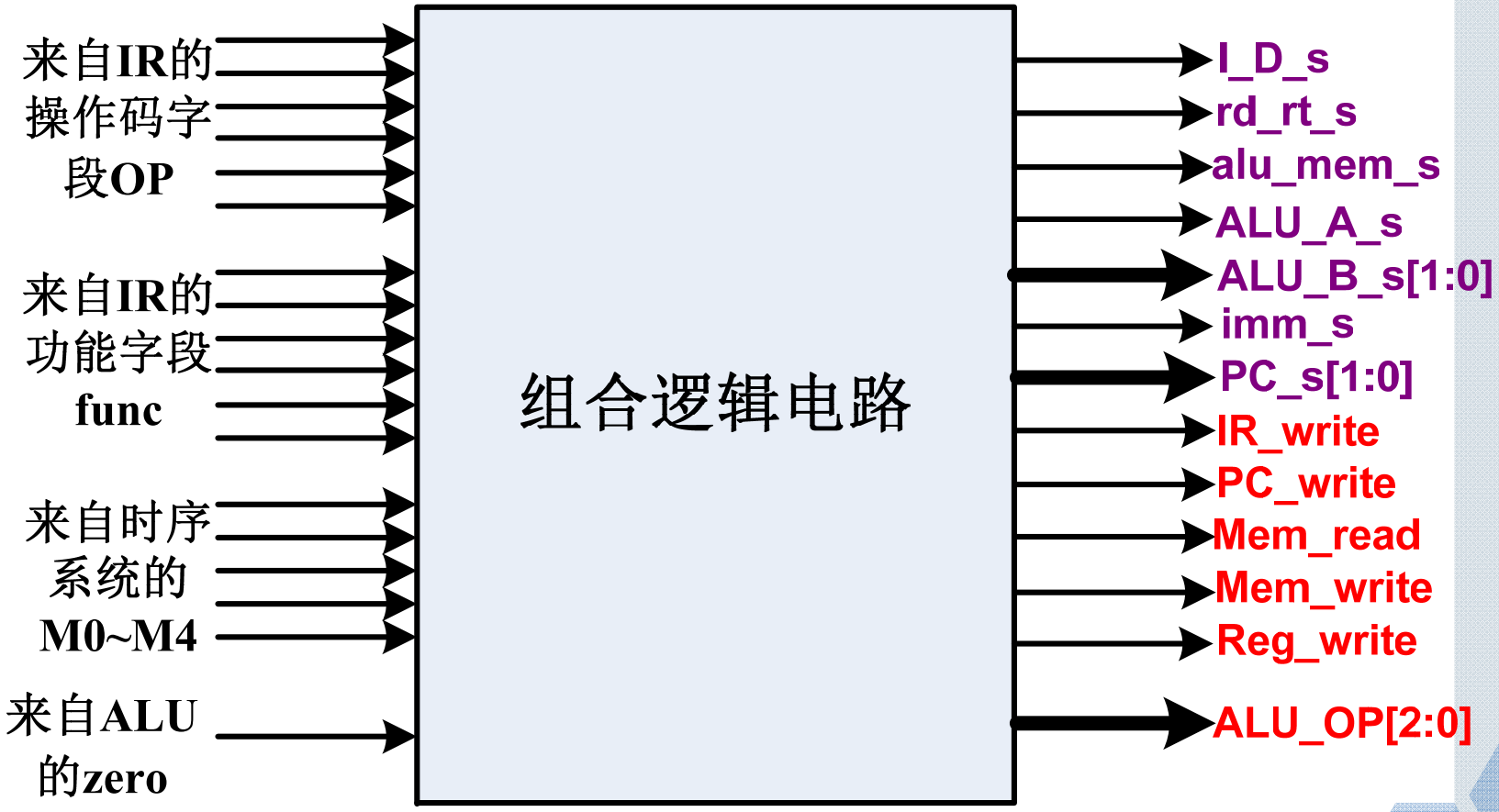
（4）J型跳转指令：J		
时钟周期	操作	发送控制信号
M0	Mem[PC]→IR, PC+4→PC	I_D_s=0,Mem_read,IR_write; ALU_A_s=0, ALU_B_s=00, ALU_OP=100,PC_s=00, PC_write;
M1	{PC[31:28],address, 2'b00}→PC	PC_s=10,PC_write

四、硬布线控制器设计举例

- 4、综合每个微操作控制信号的逻辑函数
- 输入变量：
  - 指令操作码：OP（IR[31:26]）
  - 功能码：func（ IR[5:0] ）
  - 时钟周期：M0~M4
  - ALU结果状态：zero
- 输出变量：各个控制信号

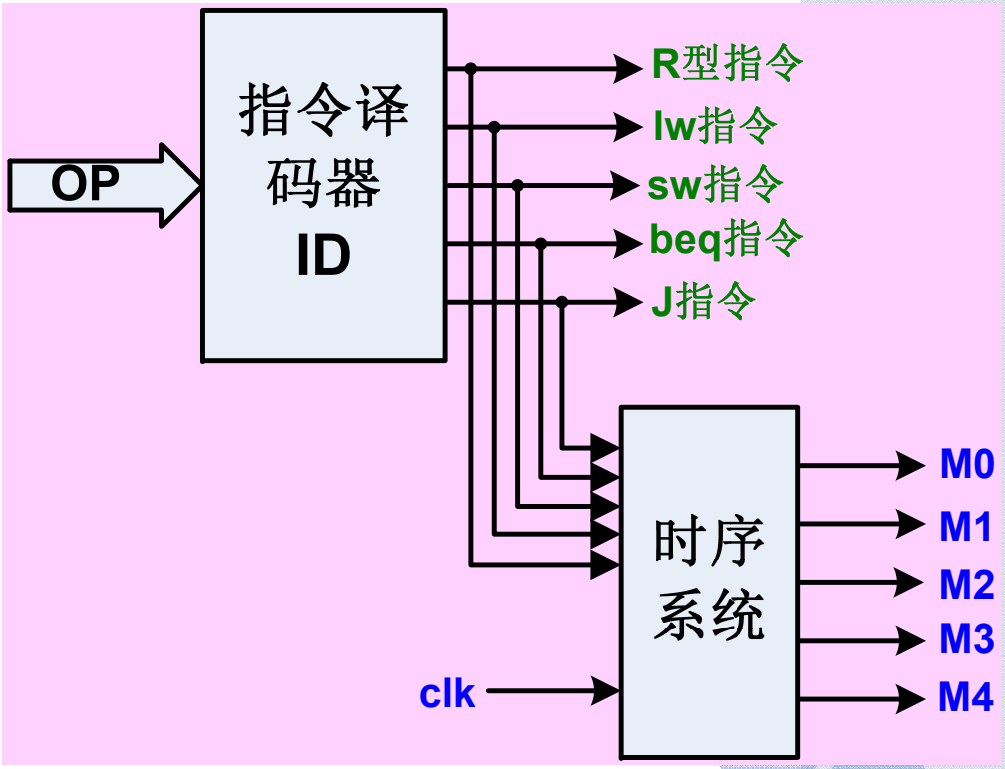
四、硬布线控制器设计举例

❖4、综合每个微操作控制信号的逻辑函数



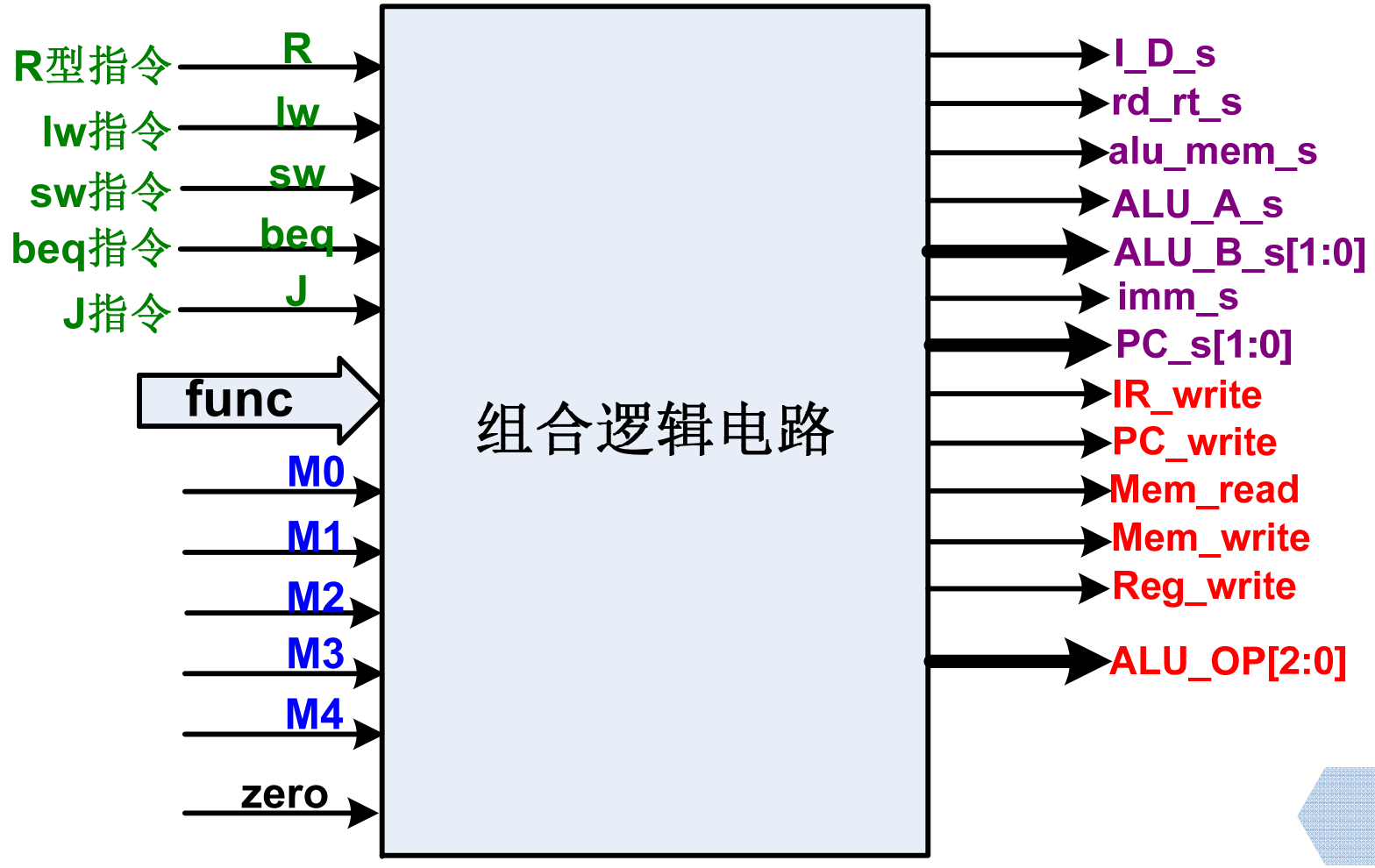
四、硬布线控制器设计举例

- 4、综合每个微操作控制信号的逻辑函数
- 为降低电路的复杂度，减少输入信号的个数，可以分块处理，譬如：
  - 单独设置指令译码器：
    - 输入：OP和func
    - 输出：指令信号



4、综合每个微操作控制信号的逻辑函数

- 控制单元的组合逻辑电路框图简化为：



4、综合每个微操作控制信号的逻辑函数

- （1）描述每个时钟周期发送的控制信号，方法是：

  指令·周期·状态：发送的控制信号

- M0： I_D_s=0,Mem_read,IR_write;ALU_A_s=0, ALU_B_s=00, PC_s=00, PC_write
- R·M1：无（不代表没有操作）
- R·M2：ALU_A_s=1, ALU_B_s=01, ALU_OP=***;
- R·M3：rd_rt_s=0,alu_mem_s=0, Reg_write

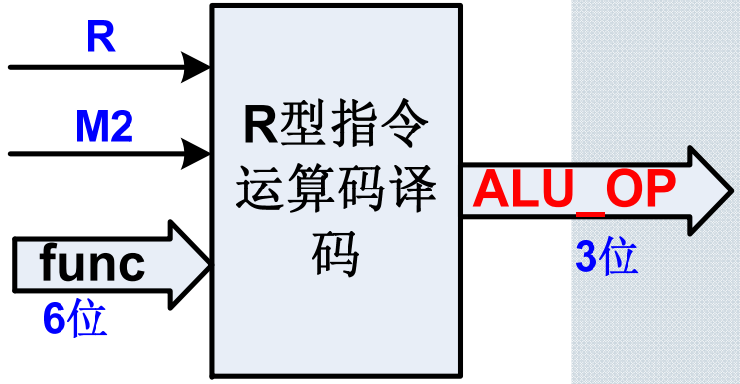
- M0周期时没有指令信号，是因为：
  - 不知道当前是何指令；
  - 所有指令在M0周期都执行相同的操作（公操作）；

根据func译码

4、综合每个微操作控制信号的逻辑函数

- R型指令的M2周期，ALU_OP要根据func字段译码，真值表如下：

指令	输入			输出
	R	M2	func	ALU_OP
add	1	1	100000	100
sub	1	1	100010	101
and	1	1	100100	000
or	1	1	100101	001
xor	1	1	100110	010
nor	1	1	100111	011
sltu	1	1	101011	110
sllv	1	1	000100	111



4、综合每个微操作控制信号的逻辑函数

- lw·M1：无（不代表没有操作）
- lw·M2： ALU_A_s=1, ALU_B_s=10, imm_s=1, ALU_OP=100
- lw·M3： I_D_s=1,Mem_read
- lw·M4： rd_rt_s=1,alu_mem_s=1, Reg_write

- sw·M1：无（不代表没有操作）
- sw·M2： ALU_A_s=1, ALU_B_s=10, imm_s=1, ALU_OP=100
- sw·M3： I_D_s=1,Mem_write

4、综合每个微操作控制信号的逻辑函数

- beq·M1： ALU_A_s=0, ALU_B_s=11, ALU_OP=100,imm_s=1
- beq·M2：ALU_A_s=1, ALU_B_s=01, ALU_OP=101;
- beq·M2·zero：PC_s=01,PC_write
- J·M1： PC_s=10,PC_write
- 罗列成表格：

条件	发送控制信号
M0	I_D_s=0,Mem_read,IR_write;ALU_A_s=0, ALU_B_s=00, PC_s=00, PC_write
R・M1	无（不代表没有操作）
R・M2	ALU_A_s=1, ALU_B_s=01, ALU_OP=***;
R・M3	rd_rt_s=0,alu_mem_s=0, Reg_write
lw・M1	无（不代表没有操作）
lw・M2	ALU_A_s=1, ALU_B_s=10, imm_s=1, ALU_OP=100
lw・M3	I_D_s=1,Mem_read
lw・M4	rd_rt_s=1,alu_mem_s=1, Reg_write
sw・M1	无（不代表没有操作）
sw・M2	ALU_A_s=1, ALU_B_s=10, imm_s=1, ALU_OP=100
sw・M3	I_D_s=1,Mem_write
beq・M1	ALU_A_s=0, ALU_B_s=11, ALU_OP=100,imm_s=1
beq・M2	ALU_A_s=1, ALU_B_s=01, ALU_OP=101
beq・M2・zero	PC_s=01,PC_write
J・M1	PC_s=10,PC_write

4、综合每个微操作控制信号的逻辑函数

- （2）逻辑函数综合：对每个控制信号，凡是“：”右边出现该信号（=1）的，将“：”左边的条件进行“或”运算；
- I_D_s = lw • M3 + sw • M3
- Mem_read = M0+ lw • M3
- IR_write = M0
- PC_write = M0+ beq • M2 • zero + J • M1
- Reg_write = R • M3 + lw • M4
- Mem_write = sw • M3
- ALU_A_s = R • M2 + lw • M2 + sw • M2 + beq • M2
- ALU_B_s[1] = lw • M2 + sw • M2 + beq • M1
- ALU_B_s[0] = R • M2 + beq • M1 + beq • M2
- ……

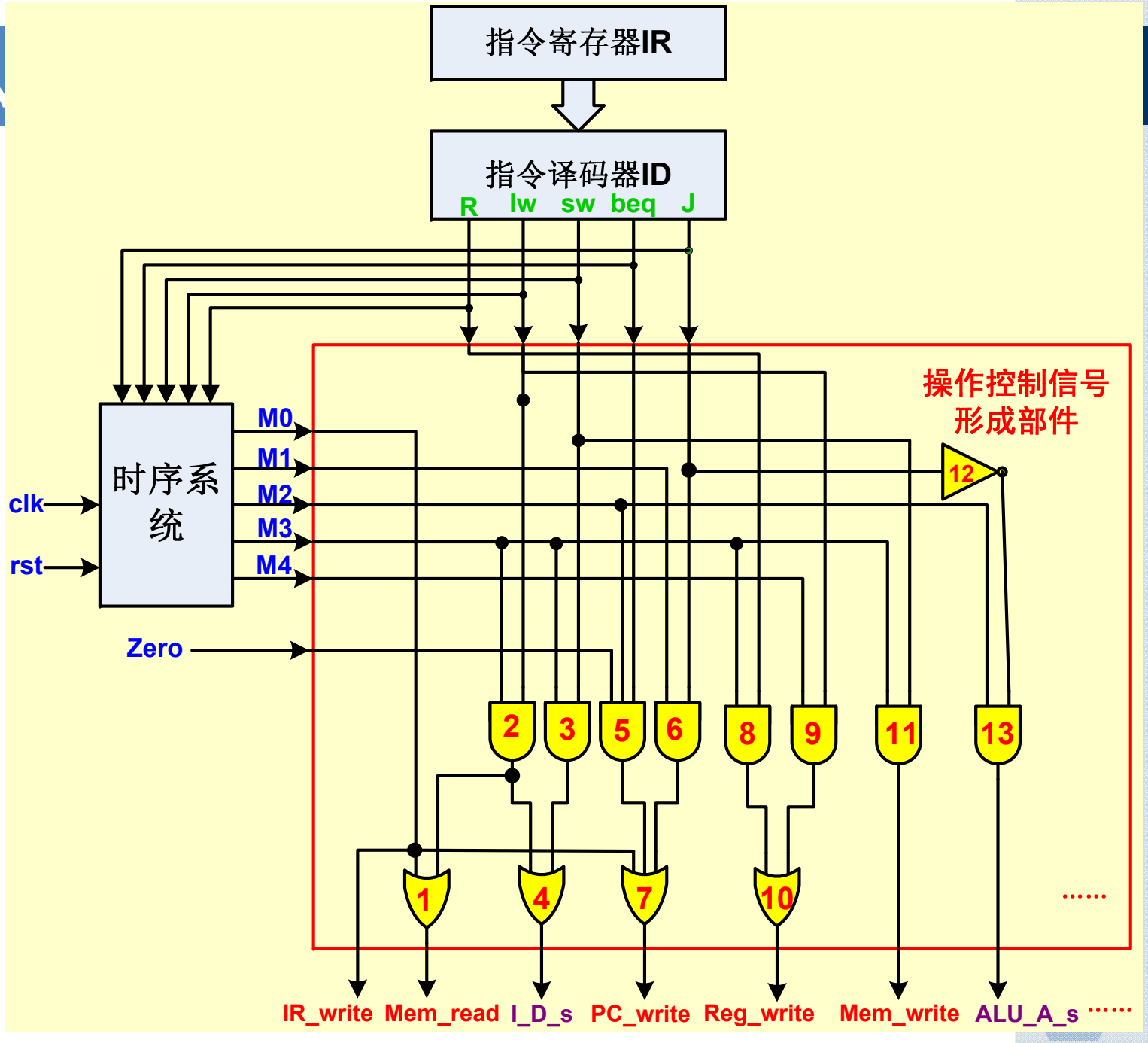
4、综合每个微操作控制信号的逻辑函数

- （3）优化和简化逻辑函数：
  - 从逻辑代数的角度化简；
  - 从指令系统的整体逻辑关系上来优化
- 譬如：
- ALU_A_s = R · M2 + lw · M2 + sw · M2 + beq · M2
- 优化后：
- ALU_A_s = （R + lw + sw + beq） · M2 = $\overline{J}$ · M2

- 即：将整个指令系统当做逻辑1来看待
- R + lw + sw + beq + J =1

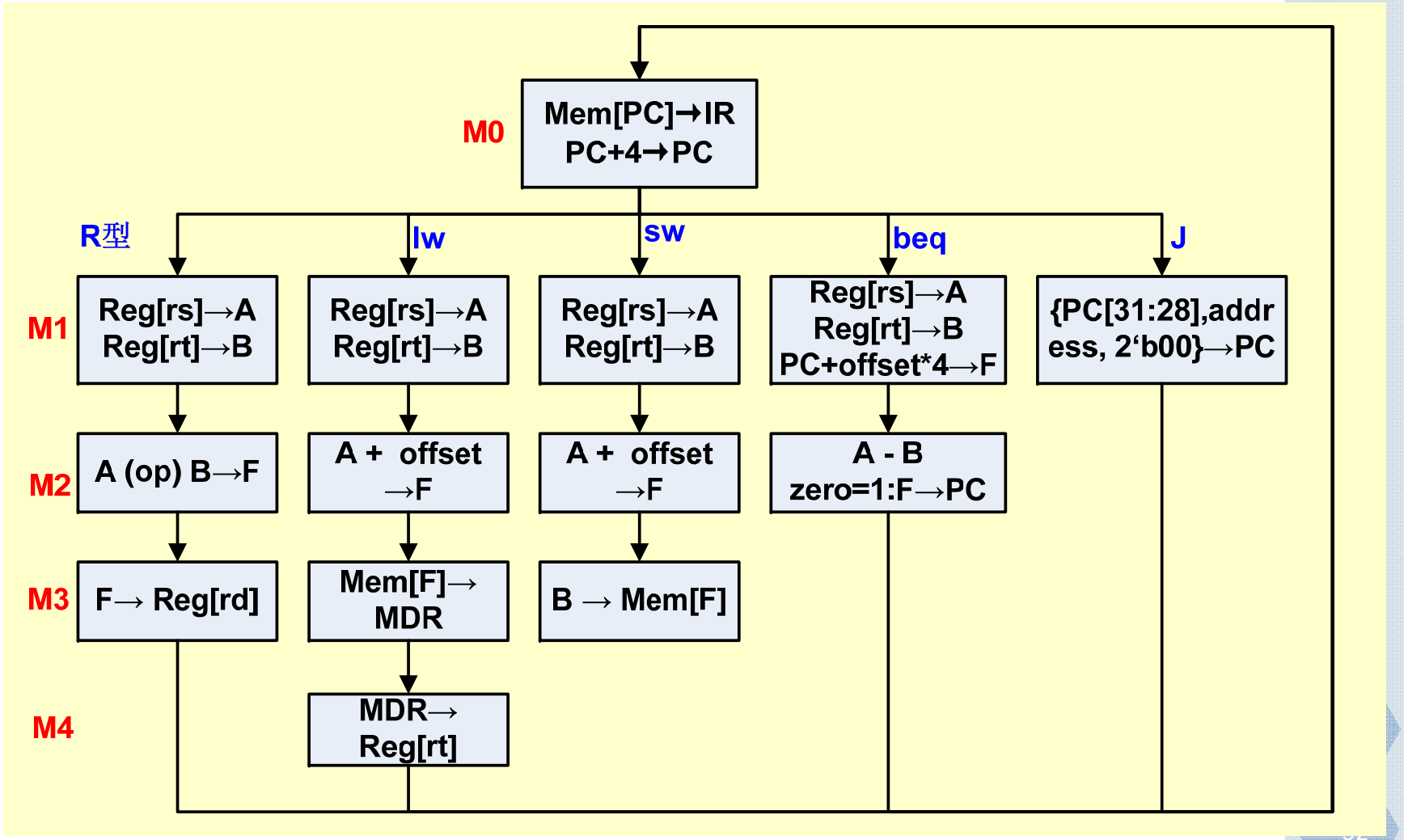
四、

❖ 5、逻辑实现


指令寄存器IR
指令译码器ID
R lw sw beq J
操作控制信号
形成部件
时序系统
clk
rst
M0
M1
M2
M3
M4
Zero
12
2
3
5
6
8
9
11
13
1
4
7
10
……
IR_write Mem_read I_D_s PC_write Reg_write Mem_write ALU_A_s ……

四、硬布线控制器设计举例

❖ 指令流程图



四、硬布线控制器设计举例

- 指令是如何被执行的呢？

① 开机上电后，硬件产生rst信号，该信号使得PC置初值，即为第一条指令在内存中的地址；同时，时序电路开始工作，机器周期计数器被清零，即产生的第一个周期信号为M0。

② M0信号送入操作控制信号形成部件后，由图中的逻辑可知，驱动1号或门和7号或门输出1，则信号IR_write、Mem_read、PC_write均有效(=1)，其他为0，这些信号发送到相应的部件后，也就执行了取指令、PC自增的操作：

Mem[PC]→IR,PC+4→PC；

四、硬布线控制器设计举例

③ M0周期结束后，指令译码器译码，指令信号线输出为有效，时序系统一定进入M1周期，此时M1=1：

- 如果是J指令，它驱动6号和7号门输出信号PC_write信号和PC_s=10，就执行了跳转操作；
- 如果是beq指令，则发送ALU_B_s=11, ALU_OP=100, imm_s=1，执行计算分支目标地址操作；
- 如果是其他指令，则不发送控制信号，但实际上执行了读rs、rt寄存器的操作。

四、硬布线控制器设计举例

④ 时序系统根据不同指令，决定下一个周期状态：

- J指令：下一周期为M0；
- R、sw指令：后续周期M2→M3 → M0
- lw指令：后续周期M2→M3 → M4 → M0
- beq指令：后续周期M2→ M0

❖ 在每个周期，不同指令发送不同的信号。

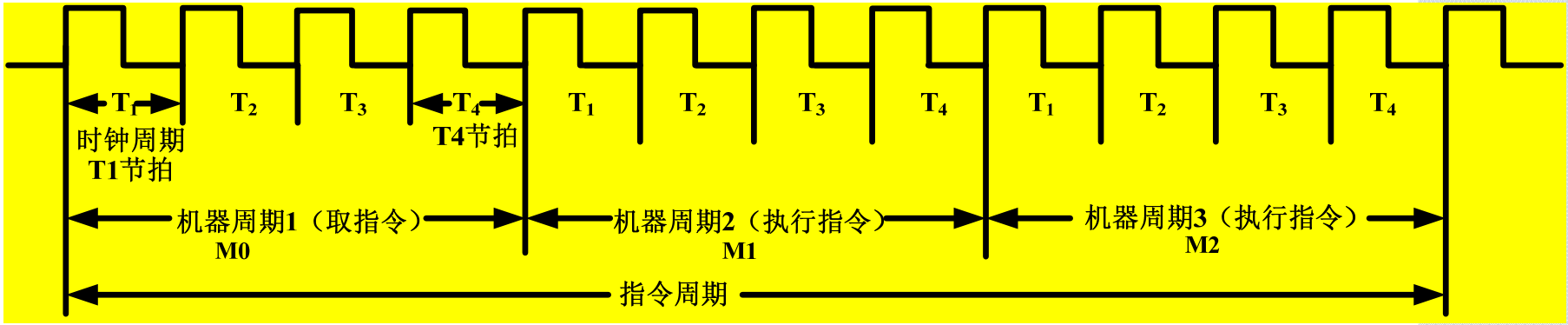
计算机工作的过程的就是循环往复地取指令、分析指令、执行指令的过程。

硬布线控制器的特点

- 微操作控制信号由组合逻辑电路即时产生。
- 硬布线控制器电路设计较为繁琐、不规整。
- 硬布线控制器非常不利于指令的修改和扩充。 现代微电子设计技术的自动化程度日益增高，极大地弥补了这个缺陷。
- 执行速度快，节省芯片面积。
- 硬布线控制器多应用于RISC系统。

五、硬布线控制器的时序系统

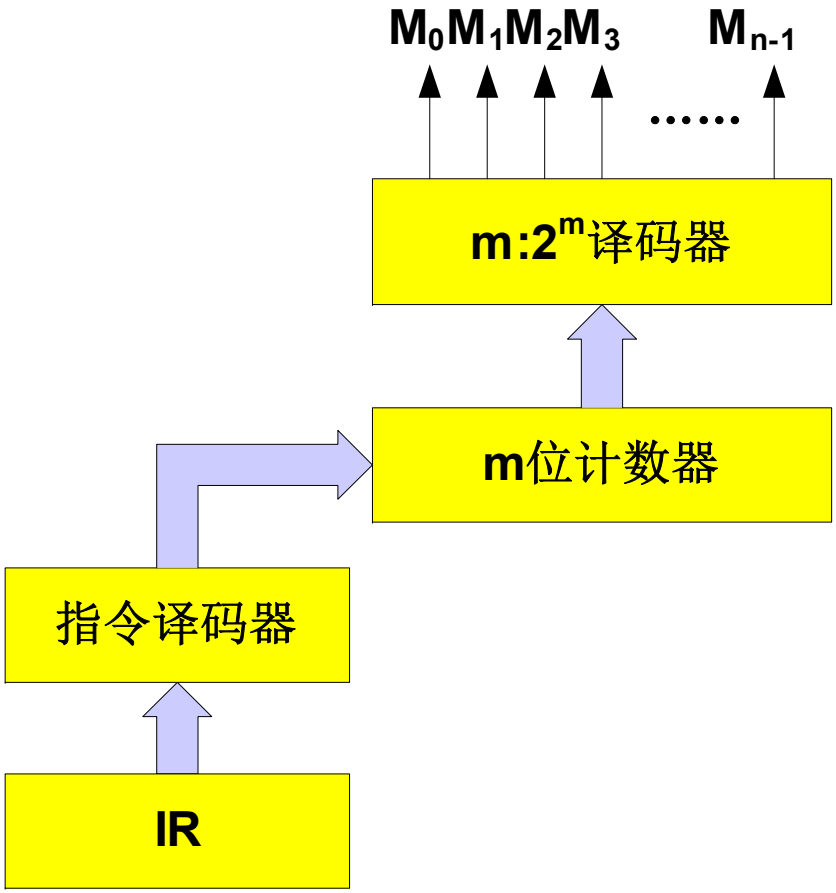
- 一般具有两级时序信号：
  - 机器周期：M
  - 节拍：T
- 前述举例：一个机器周期中包含一个时钟周期



指令周期、机器周期与节拍

五、硬布线控制器的时序系统

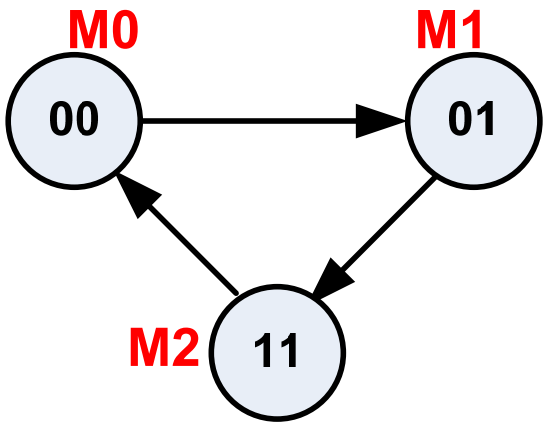
- 机器周期信号一般可以采用计数器输出译码方式产生。



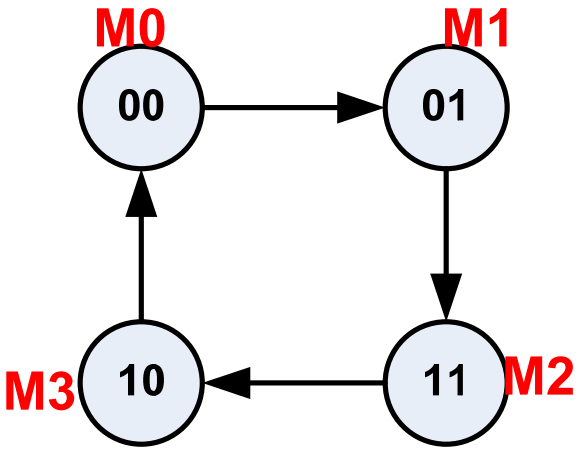
机器周期信号产生电路

五、硬布线控制器的时序系统

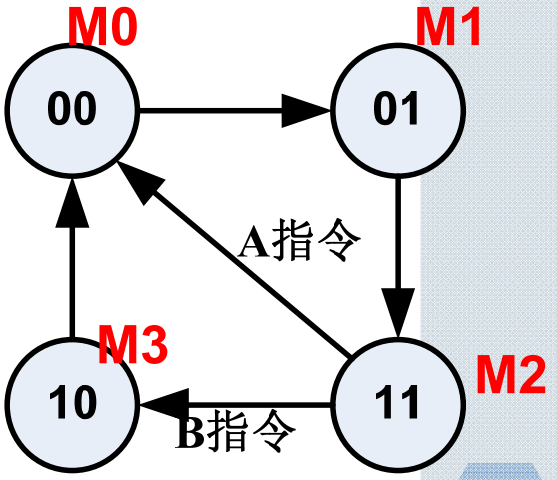
- 假设某机器的指令系统有两条指令：
  - 指令A包含3个机器周期：M0→M1→M2
  - 指令B包含4个机器周期：M0→M1→M2→M3
- 则需要一个2位计数器和一个2：4译码器
- 2位计数器的状态转移图：



（a）指令A



（b）指令B



（c）指令系统

五、硬布线控制器的时序系统

- 2位计数器的状态转移表：
  - Q_1，Q_2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	0	1	0	0	0	1
0	1	1	1	0	1	1	1
1	1	0	0	1	1	1	0
				1	0	0	0

五、硬布线控制器的时序系统

- 根据真值表列出计数器的输出表达式
- 对于指令A，其表达式为

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

- 对于指令B，其表达式为：

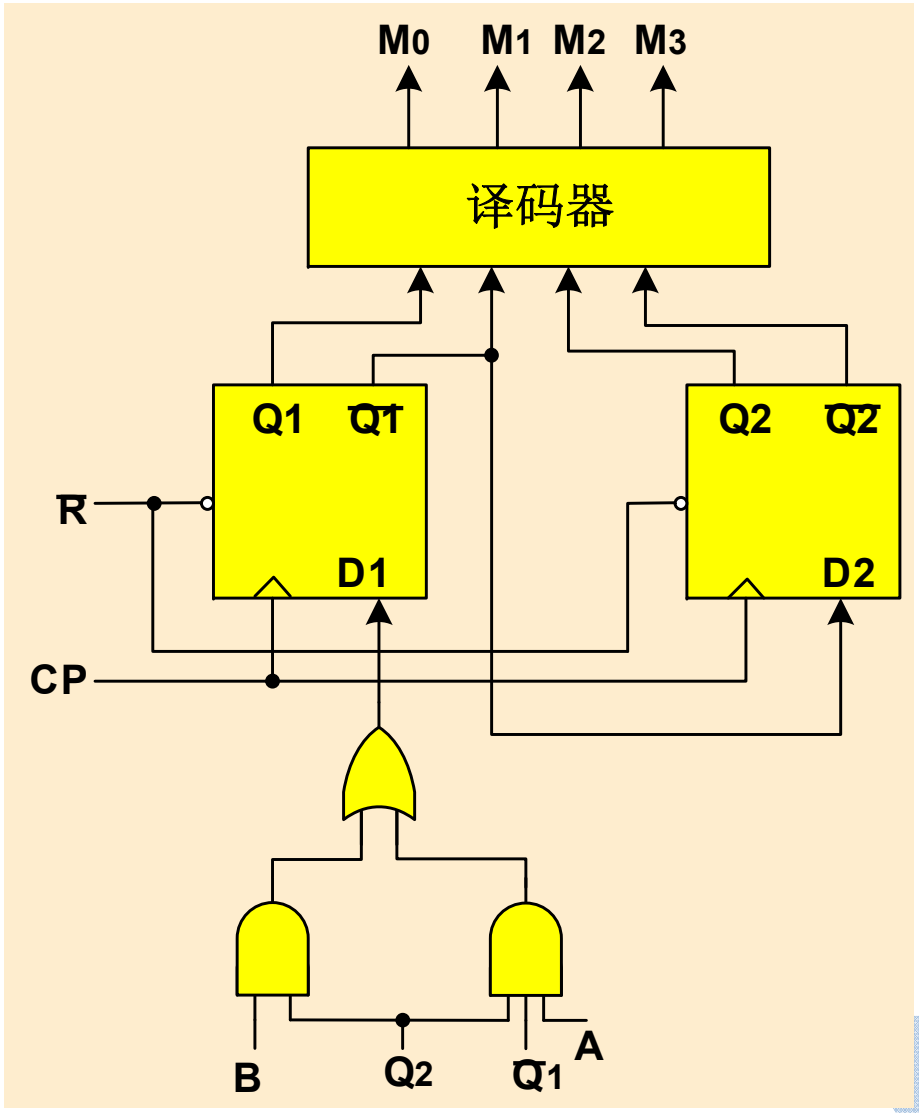
$$\begin{cases} Q_1^{'} = \overline{Q_1} Q_2 + Q_1 Q_2 = Q_2 \\ Q_2^{'} = \overline{Q_1}\,\overline{Q_2} + \overline{Q_1} Q_2 = \overline{Q_1} \end{cases}$$

所以：

$$\begin{cases} Q_1^{'} = A\,\overline{Q_1} Q_2 + B\,Q_2 \\ Q_2^{'} = (A+B)\,\overline{Q_1} \end{cases} \Rightarrow \begin{cases} Q_1^{'} = A\,\overline{Q_1} Q_2 + B\,Q_2 \\ Q_2^{'} = \overline{Q_1} \end{cases}$$

五、硬布线控制器的时序系统

- 当执行指令A时，顺序产生机器周期信号M0、M1、M2；
- 当执行指令B时，顺序产生机器周期信号M0、M1、M2、M3。



两条指令的机器周期产生电路

The End!