3.5.3 单周期MIPS处理器

（控制单元设计）

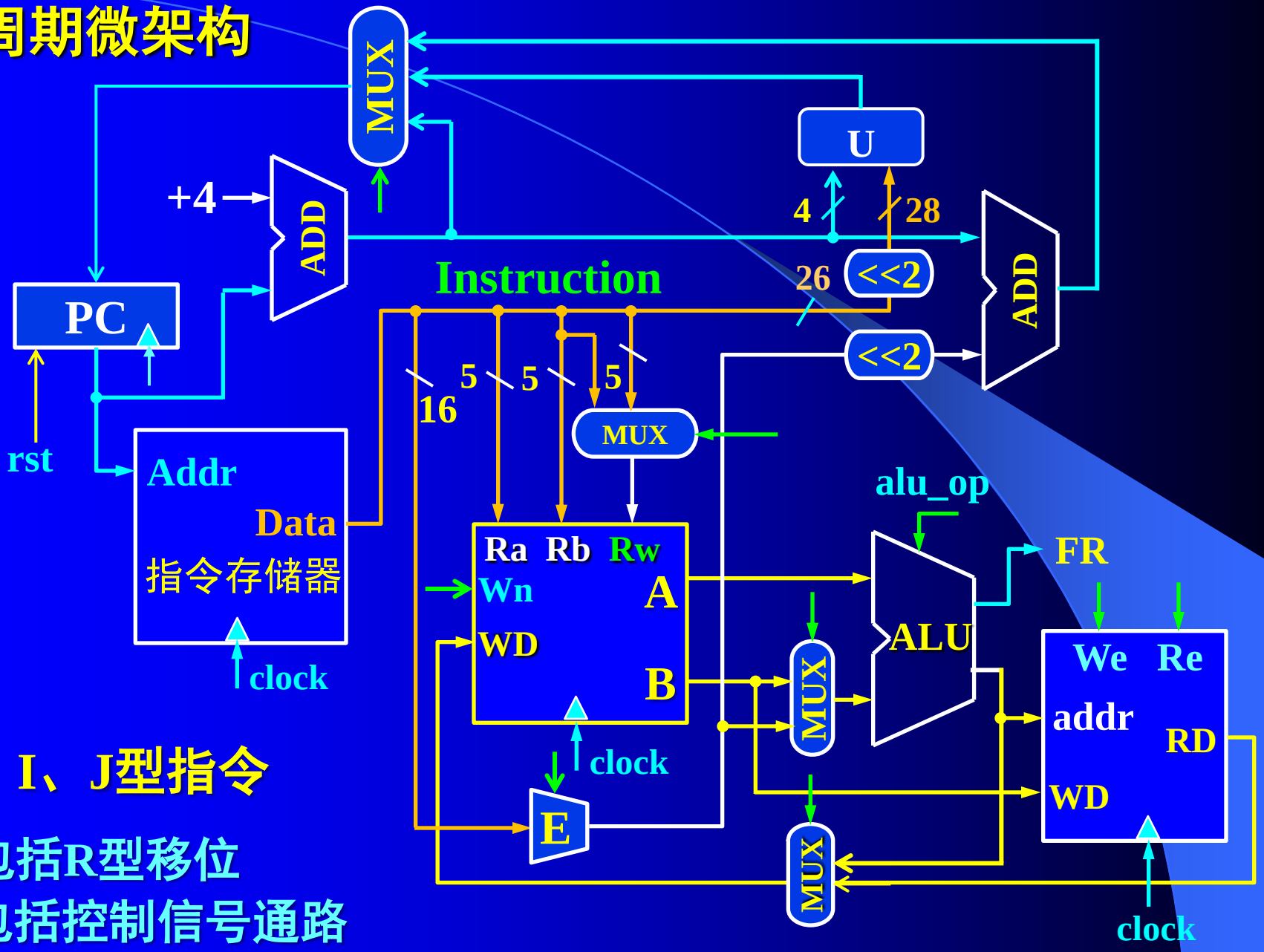
※CPU设计的主要任务

※拟定指令集 √

※数据通路设计 √

※控制器设计

单周期微架构



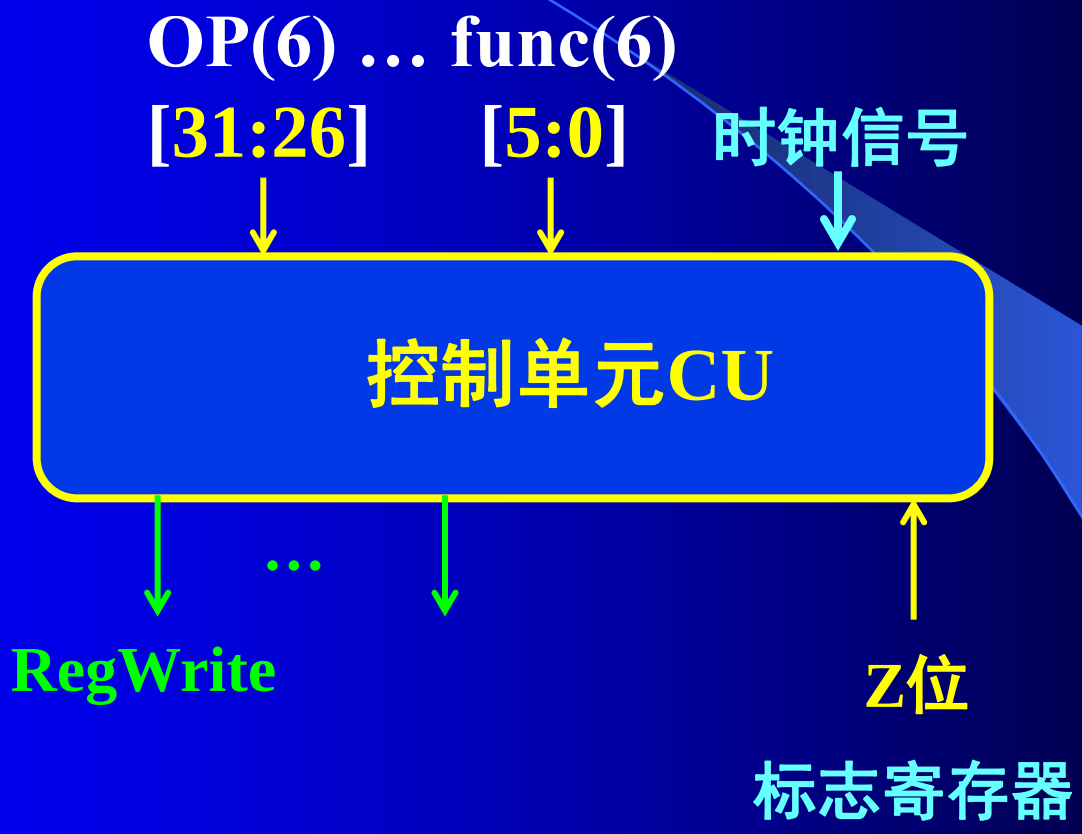
R、I、J型指令

未包括R型移位

未包括控制信号通路

➢ 控制信号（微命令）产生部件

指令:



如何恰当地产生这些控制信号？

有两种方式：

(1) 硬连线(hardwired), 基于组合逻辑。

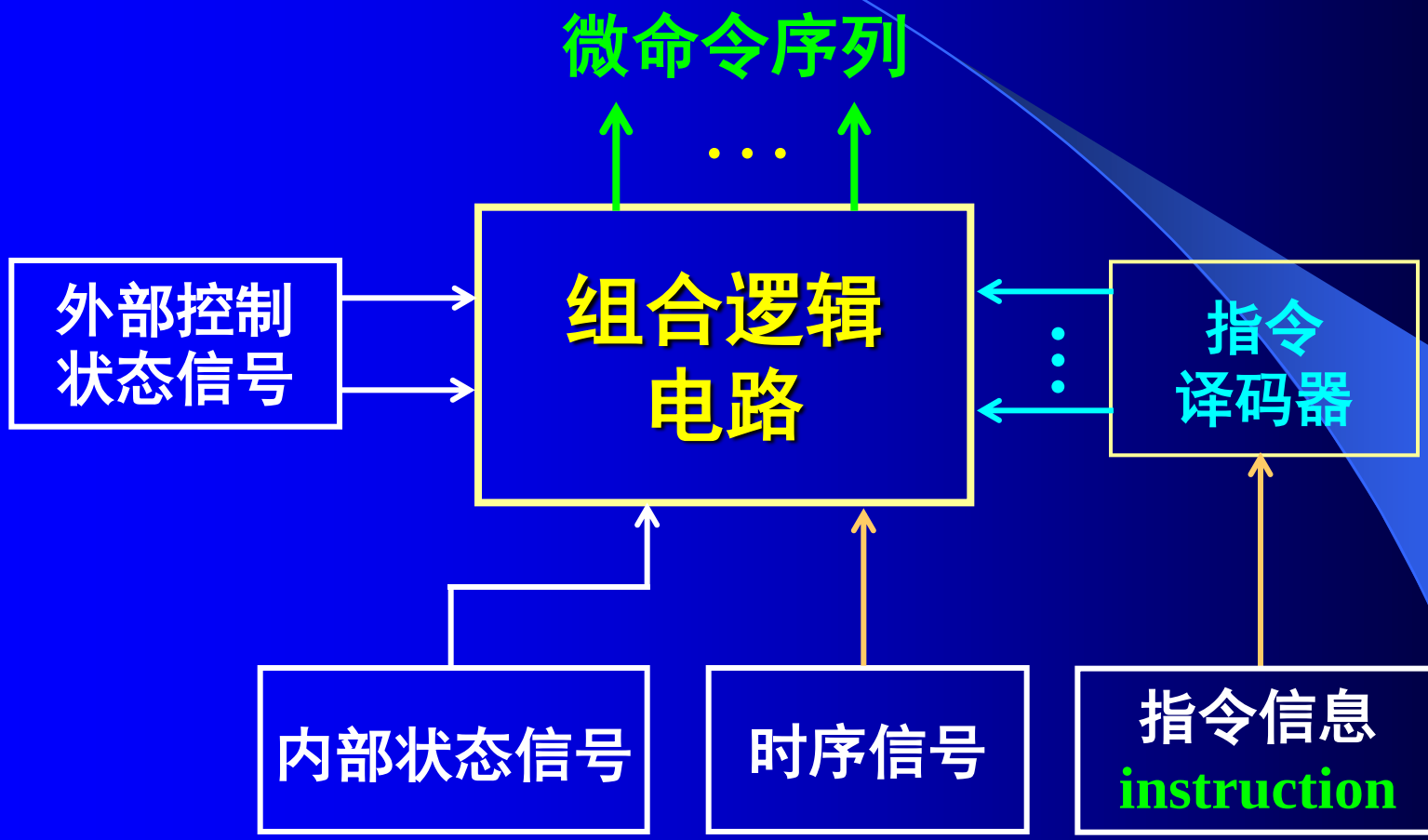
→ 硬连线控制器

→ 组合逻辑控制器

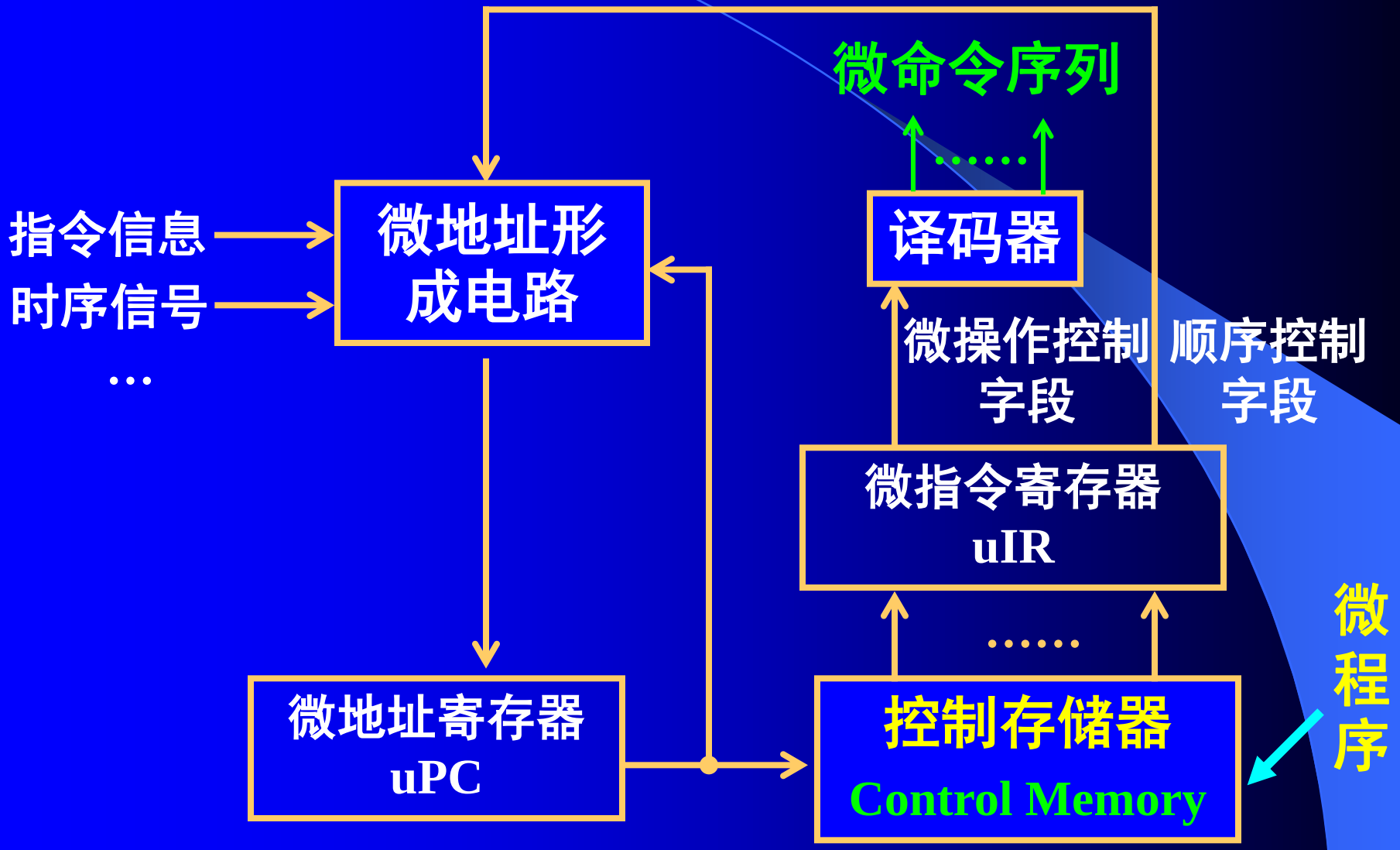
(2) 微程序(micro-programmed)，基于存储。

→ 微程序控制器

（1）组合逻辑控制的基本原理



（2）微程序控制的基本原理



组合逻辑方式的特点：

（1）控制信号的产生速度比微程序快

（2）设计不规整

（3）不容易修改或扩展

微程序控制方式的特点：

（1）用存储逻辑代替硬连线逻辑，结构规整；

（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）控制信号的产生比组合逻辑慢

3、单周期控制单元设计

※RISC，这里采用组合逻辑方式

设计步骤:

①基于数据通路，确定各指令的控制信号。

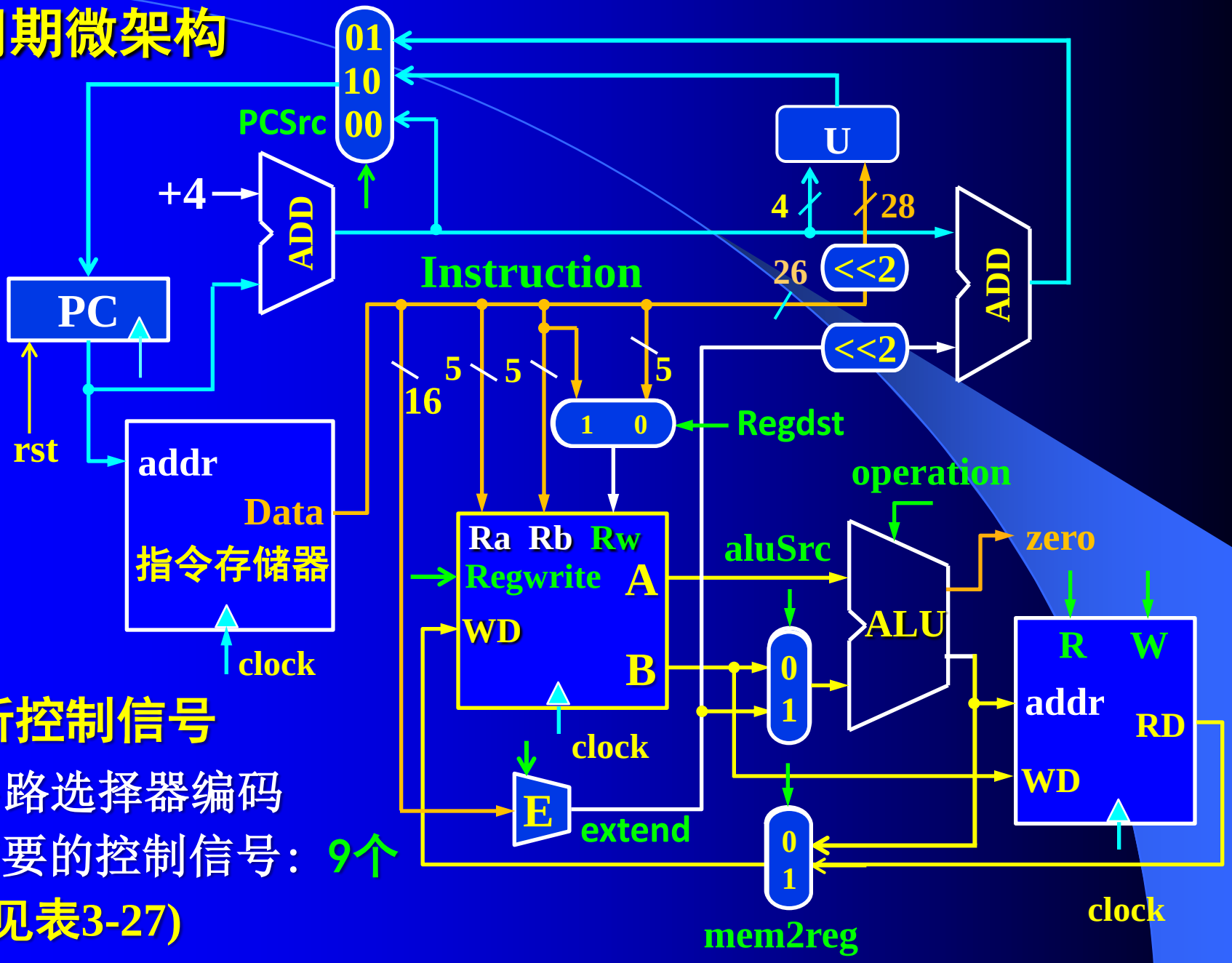
- 编码: 0、1、X(无关项)；
- 复杂信号特殊处理，如ALU的控制信号；

②构建控制信号的真值表。

③将真值表转换成控制信号的产生逻辑。

仿真、硬件实现

单周期微架构



分析控制信号

√ 多路选择器编码

√ 需要的控制信号：9个

(参见表3-27)

各控制信号的定义：

控制器输出	无效(=0)时含义	有效(=1)时含义
RegDst	选通rd端	选通rt端
RegWrite	寄堆置于读模式	寄堆置于写模式
AluSrc	选通寄堆的输出B	选通E(offset)
PCSrc[1:0]√	00选择PC+4，01选择分支地址，10选择跳转	
MemRead	两者都为0时，存储器禁用。	存储器置于读模式
MemWrite		存储器置于写模式
Mem2Reg	选通ALU的输出	选通存储器的输出
extend	执行零扩展	执行符号扩展
operation[3:0]√	参见ALU的控制代码	

7个一位的控制信号，2个多位的控制信号。

（1）控制单元的总体结构

PCSrc(2位)和operation(4位)比较复杂，需单独设计

对于PCSrc（00，01，10）：

取决于当前指令是否是beq和j，如果是beq则还需参考ALU输出的zero标志位

对于operation（4位代码）：

R型指令，则取决于OP和func字段；

I型指令，则只取决于OP字段；

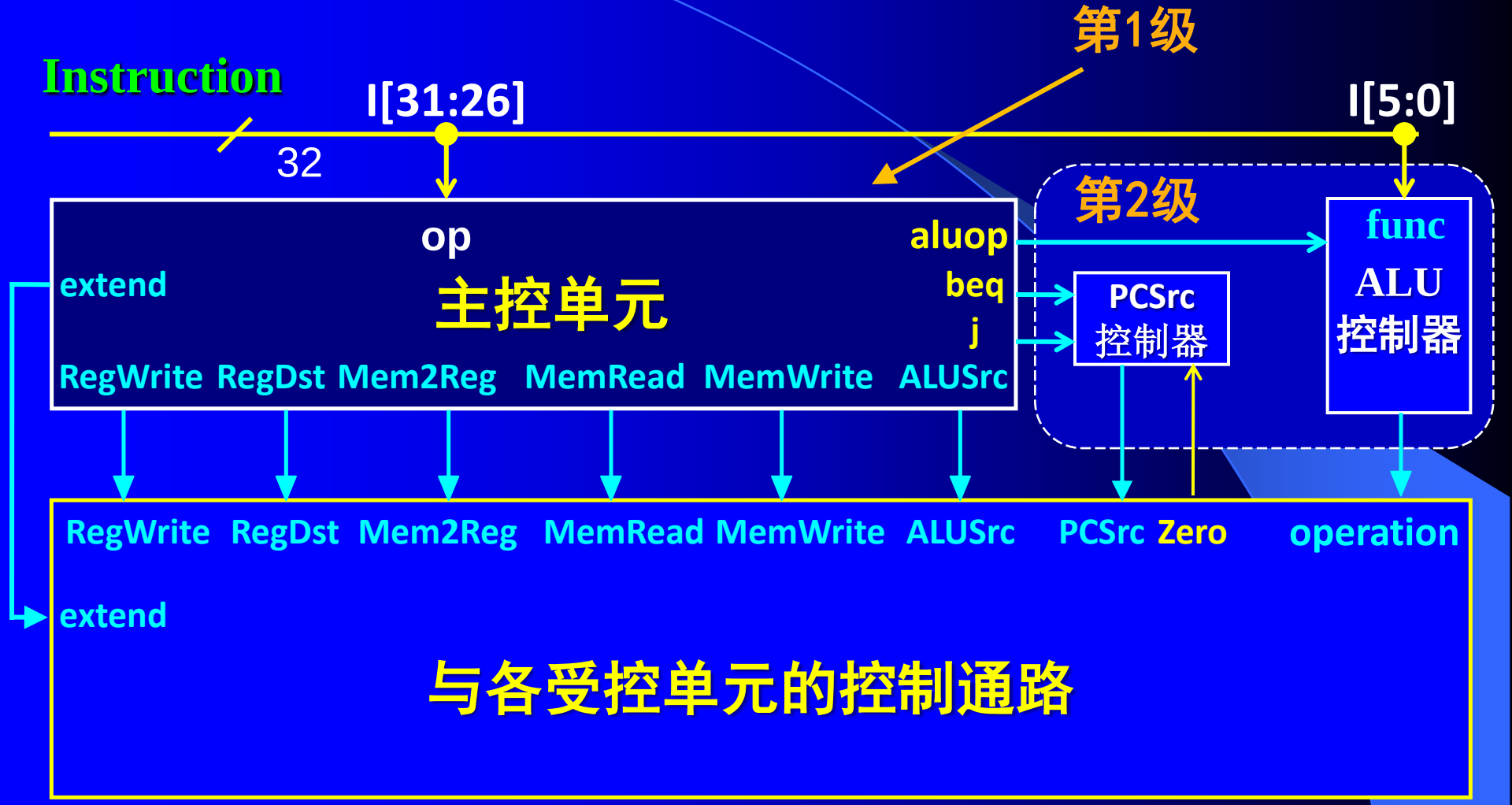
J型指令，不涉及到operation。

思路：采用分级控制方案。

主控单元→部分简单控制信号；

主控单元→中间信号→ 下级控制器→ 复杂控制信号；

控制单元采用两级控制模式



定义主-从接口： aluop，beq，j　编码？ 位数？

（2）设计ALU控制单元

主控单元输出aluop

$aluop = f(op)$

ALU控制码 (operation)	ALU功能
0000	AND（与）
0001	OR（或）
0010	ADD（加）
0110	SUB（减）
0111	小于则置1
1100	或非

ALU的输出逻辑复杂，需重点关注。

- ALU控制器的输出依赖于？
  - ✓ 指令类型(OP字段)
  - ✓ func字段(仅R型指令)
- 由此可知：
  - ✓ func只影响控制器输出的operation控制码

operation= *f* (aluop, func)

[设计思路]

先根据目标指令涉及到的ALU运算，定义aluop编码

再整理输入输出真值表

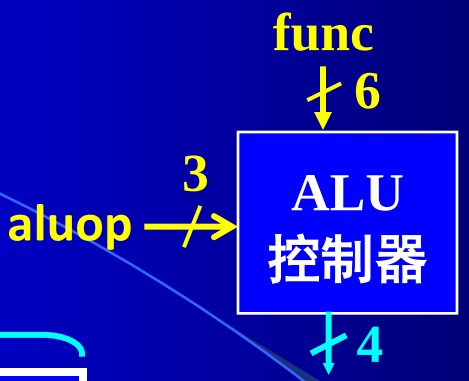
最后转换得到operation各位的产生逻辑

分析指令与ALU功能，编码aluop = *f* (op) 位数：3

指令类型	指令	func段	ALU功能	ALU控制码 (operation)	*aluop*
取数	lw	xxxxxx	加	0010	010
存数	sw	xxxxxx	加	0010	010
分支	beq	xxxxxx	减	0110	110
立即数加	addi	xxxxxx	加	0010	010
立即数与	andi	xxxxxx	与	0000	000
立即数或	ori	xxxxxx	或	0001	001
R-型	add	100000	加	0010	100
R-型	sub	100010	减	0110	100
R-型	and	100100	与	0000	100
R-型	or	100101	或	0001	100
J-型	j	xxxxxx	x	xxxx	xxx

ALU控制器的真值表

忽略无关项



指令j
与ALU无关

输入

aluop[2:0]			func[5:0]						operation			
[2]	[1]	[0]	[5]	[4]	[3]	[2]	[1]	[0]	[3]	[2]	[1]	[0]	
0	1	~~0~~	x	x	x	x	x	x	0	0	1	0	(+) lw,sw,addi
1	1	~~0~~	x	x	x	x	x	x	0	1	1	0	(-) beq
0	0	0	x	x	x	x	x	x	0	0	0	0	(&) andi
~~0~~	~~0~~	1	x	x	x	x	x	x	0	0	0	1	(|) ori
1	0	~~0~~	~~1~~	~~0~~	~~0~~	0	0	0	0	0	1	0	(+) add
1	0	~~0~~	~~1~~	~~0~~	~~0~~	0	1	0	0	1	1	0	(-) sub
1	0	~~0~~	~~1~~	~~0~~	~~0~~	1	0	0	0	0	0	0	(&) and
1	0	~~0~~	~~1~~	~~0~~	~~0~~	1	0	1	0	0	0	1	(|) or

根据真值表, 写出输入-输出逻辑式:

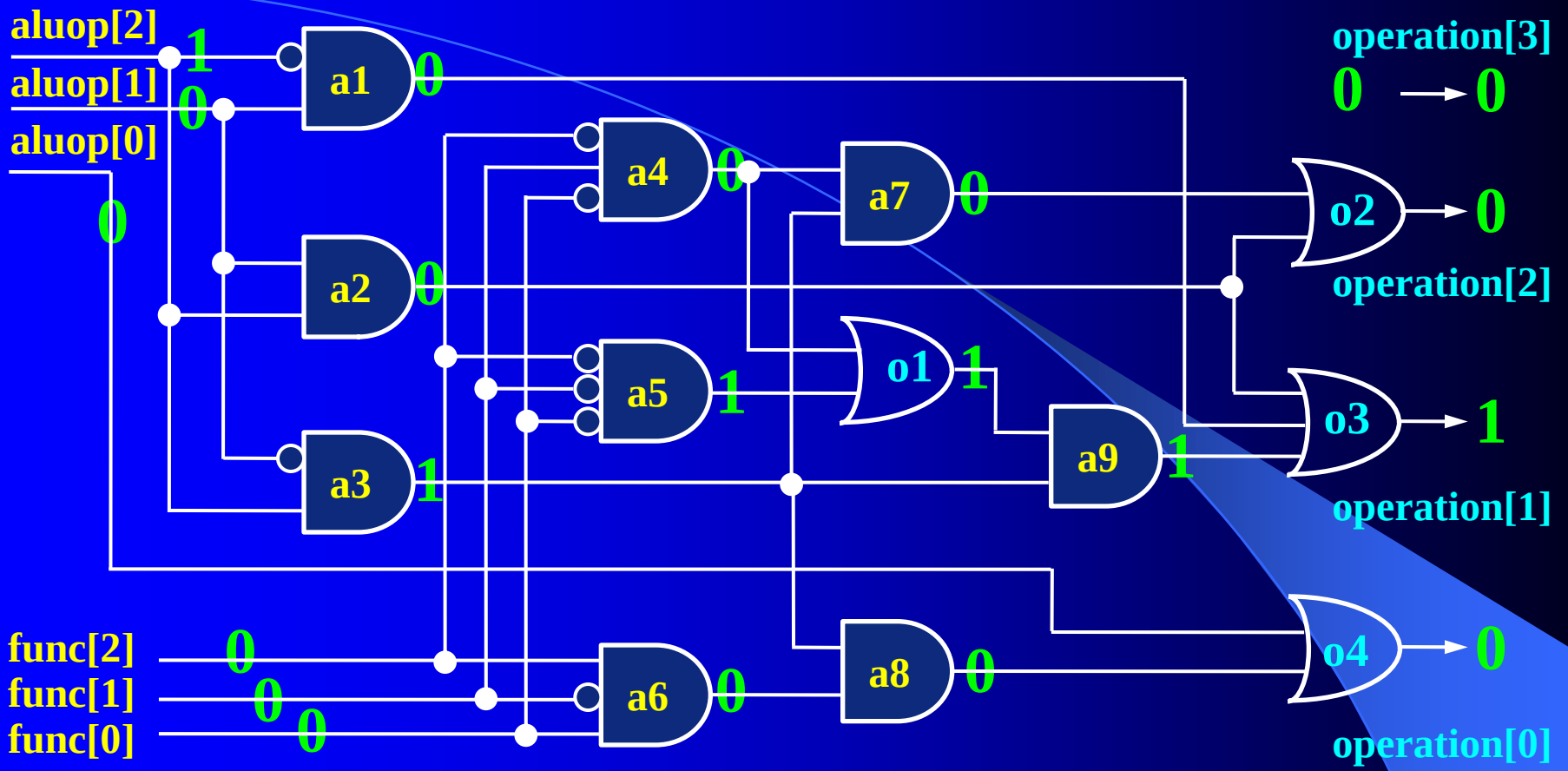
$operation[3] \equiv 0$

$operation[2]=aluop[2]aluop[1] + aluop[2]\overline{aluop[1]} \cdot \overline{f[2]f[1]f[0]}$

$operation[1]=\overline{aluop[2]}aluop[1]+aluop[2]aluop[1]+aluop[2]\overline{aluop[1]} \cdot (\overline{f[2]f[1]f[0]}+ f[2]\overline{f[1]}\overline{f[0]})$

$operation[0] = aluop[0]+aluop[2]\overline{aluop[1]} \cdot f[2]\overline{f[1]}f[0]$

可直接仿真，也可转换得到逻辑电路:



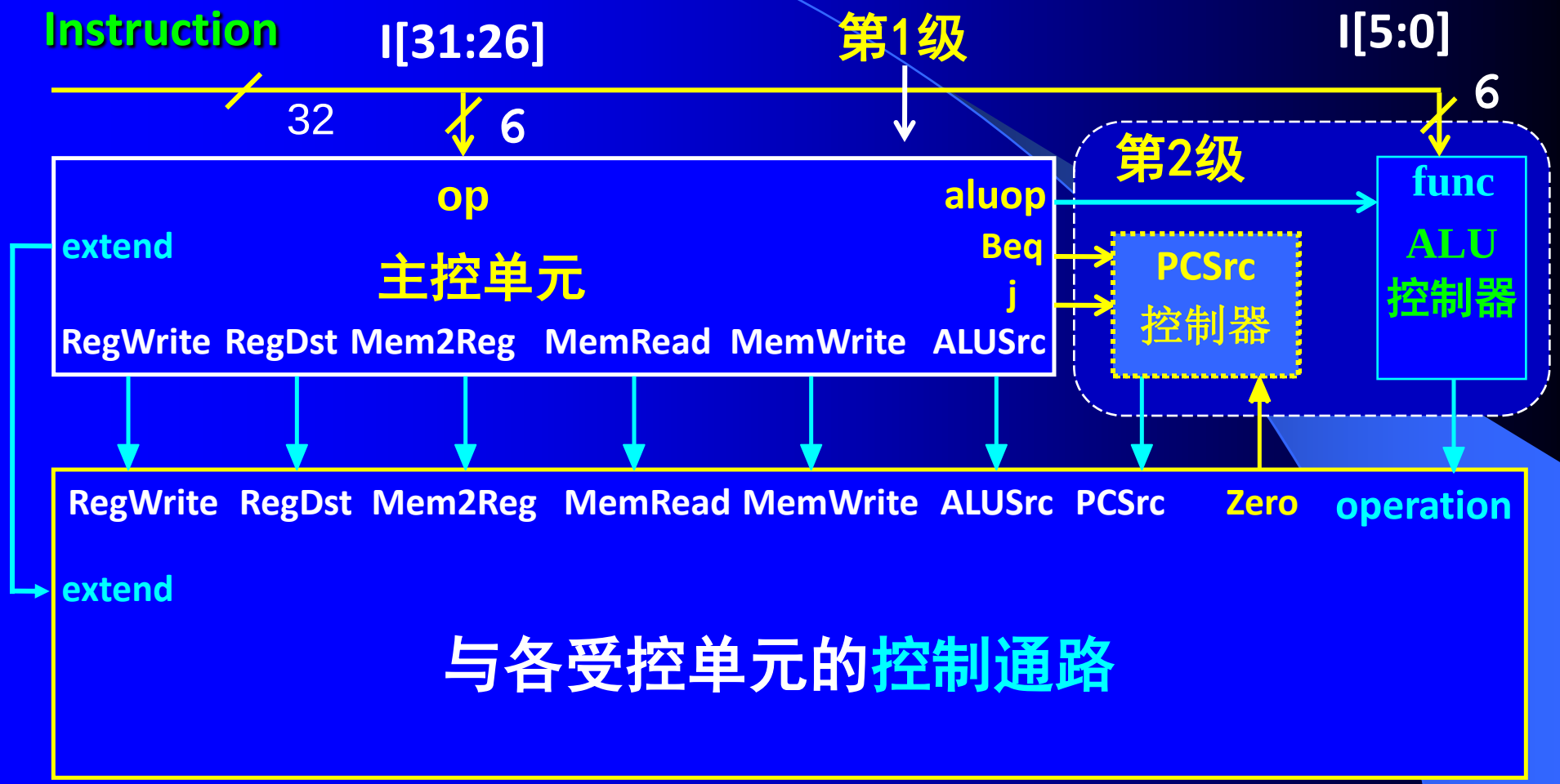
func[5:3]是无关项，不使用。

[验证] aluop[2:0]=100， func[2:0]=000时， operation=？

operation→0010（符合真值表）

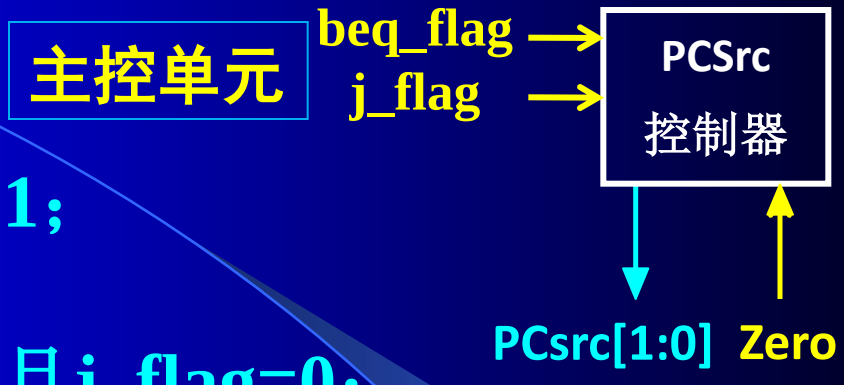
其它编码也能验证通过　alu控制器设计完成!

控制单元采用两级译码



→继续设计PCSrc控制器的组合逻辑

（3）PCSrc控制单元



方案分析：

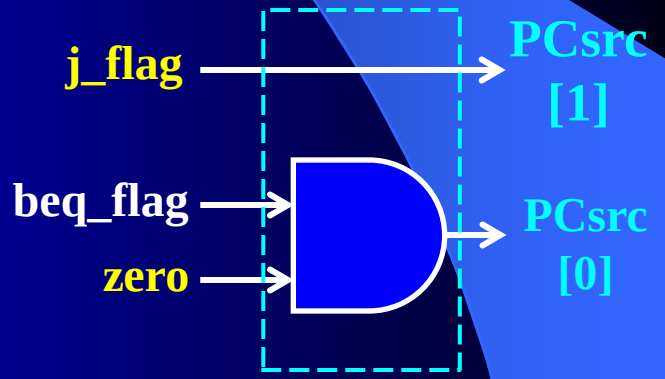
执行beq时，输出beq_flag=1；

执行j时，输出j_flag=1；

其余指令，输出beq_flag=0且j_flag=0；

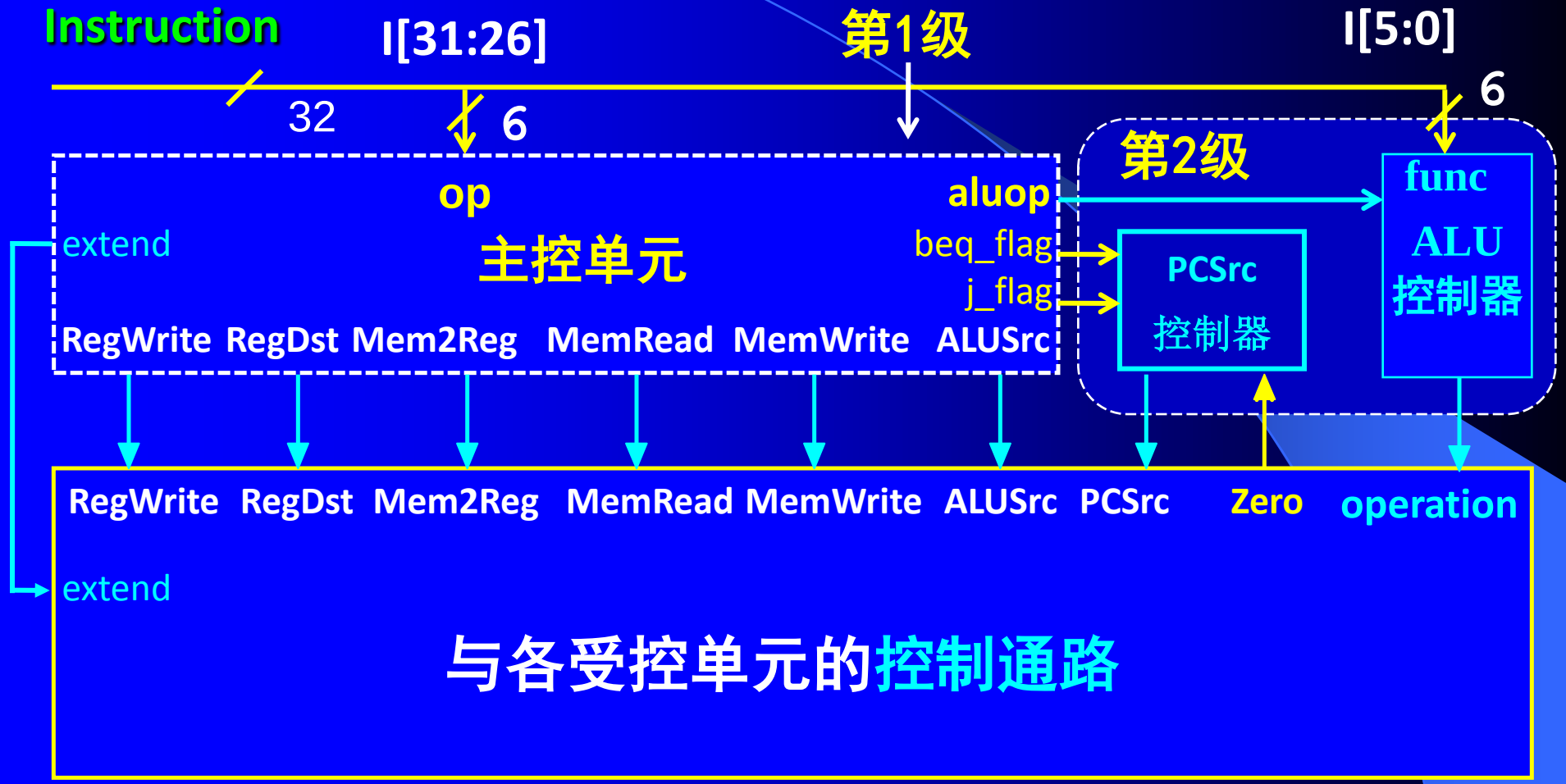
PCSrc控制器的“输入-输出”真值表如下：

指令	beg_flag	j_flag	zero	PCsrc [1]	PCsrc [0]
beq	1	x	0	0	0
	1	x	1	0	1
j	x	1	x	1	0
其它	0	0	x	0	0



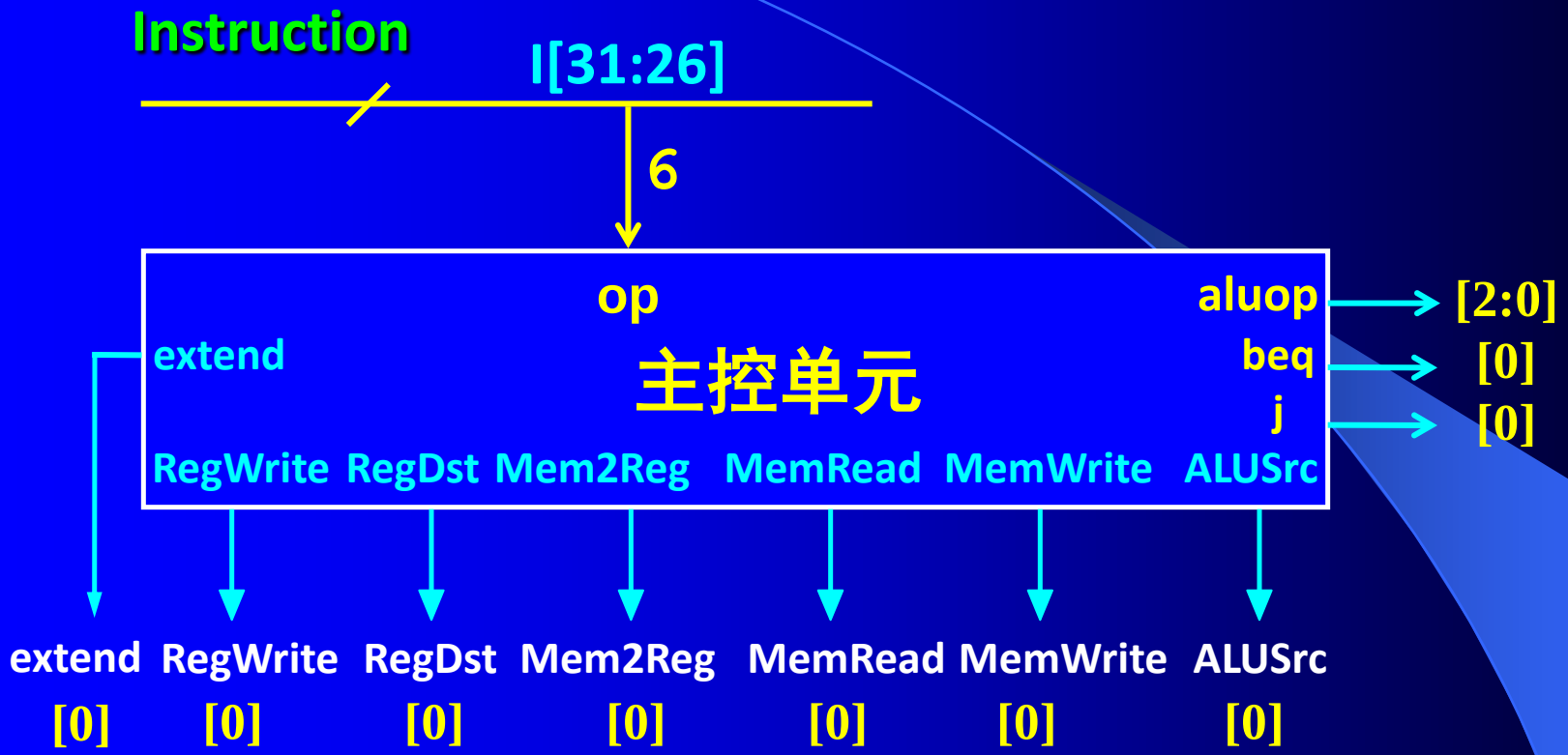
PCsrc[1]=j_flag； PCsrc[0]=beq_flag · zero（图3-85）

控制单元采用两级译码



→继续设计主控单元的组合逻辑

（4）主控单元的设计



主控单元根据1个输入，产生10个输出：

9个1位的控制信号 + 1个3位的控制信号

→整理主控单元的“输入-输出”真值表

主控单元的“输入-输出”真值表（表3-31）

	输入 OP[5:0]						RegDst	ALUSrc	Mem2Reg	RegWrite	MemWrite	MemRead	extend	aluop[2:0]			beq_flag	j_flag
	[5]	[4]	[3]	[2]	[1]	[0]								[2]	[1]	[0]		
add	0	0	0	0	0	~~0~~	0	0	0	1	0	0	x	1	0	0	0	0
sub	0	0	0	0	0	~~0~~	0	0	0	1	0	0	x	1	0	0	0	0
and	0	0	0	0	0	~~0~~	0	0	0	1	0	0	x	1	0	0	0	0
or	0	0	0	0	0	~~0~~	0	0	0	1	0	0	x	1	0	0	0	0
addi	~~0~~	~~0~~	1	0	0	~~0~~	1	1	0	1	0	0	1	0	1	0	0	0
andi	~~0~~	~~0~~	1	1	~~0~~	0	1	1	0	1	0	0	0	0	0	0	0	0
ori	~~0~~	~~0~~	1	1	~~0~~	1	1	1	0	1	0	0	0	0	0	1	0	0
lw	1	~~0~~	0	~~0~~	~~1~~	~~1~~	1	1	1	1	0	1	1	0	1	0	0	0
sw	1	~~0~~	1	~~0~~	~~1~~	~~1~~	x	1	x	0	1	0	1	0	1	0	0	0
beq	~~0~~	~~0~~	0	1	~~0~~	~~0~~	x	0	x	0	0	0	1	1	1	0	1	0
j	~~0~~	~~0~~	~~0~~	~~0~~	1	0	x	x	x	0	0	0	x	x	x	x	0	1

写出各输出信号的逻辑式：

输出10种共12位控制信号 ⇨ 共12个逻辑式

$regdst = op[3]\overline{op[2]}\,\overline{op[1]} + op[3]op[2]\overline{op[0]} + op[3]op[2]op[0] + op[5]\overline{op[3]}$

…

$aluop[0] = op[3]op[2]op[0]$

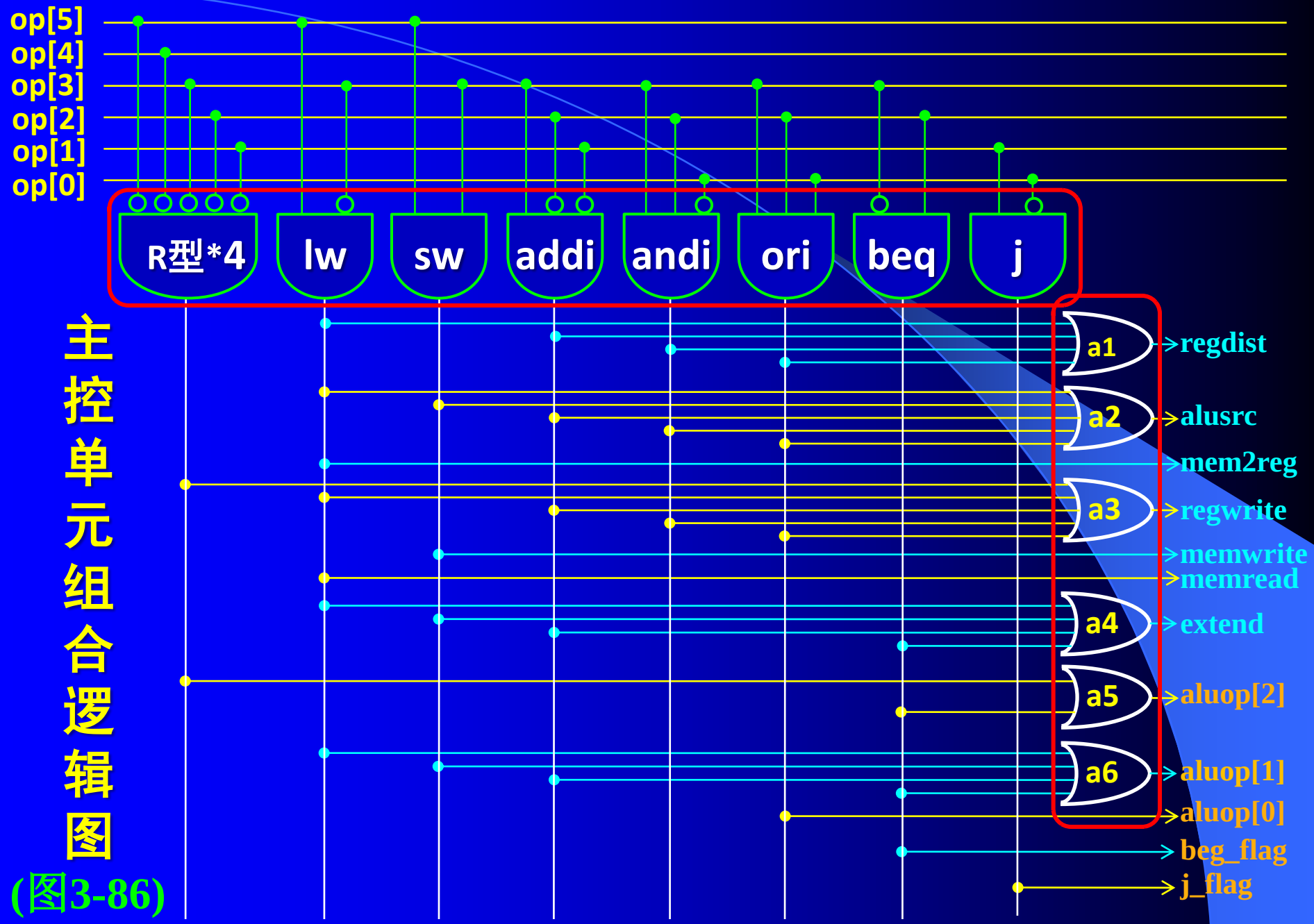
$beq\_flg = \overline{op[3]}op[2]$

$j\_flag = op[1]\overline{op[0]}$

根据各逻辑式，就可得到主控 单元的组合逻辑。
(参见教材图 3-86)

主控单元组合逻辑图

(图3-86)



完整的单周期CPU微架构