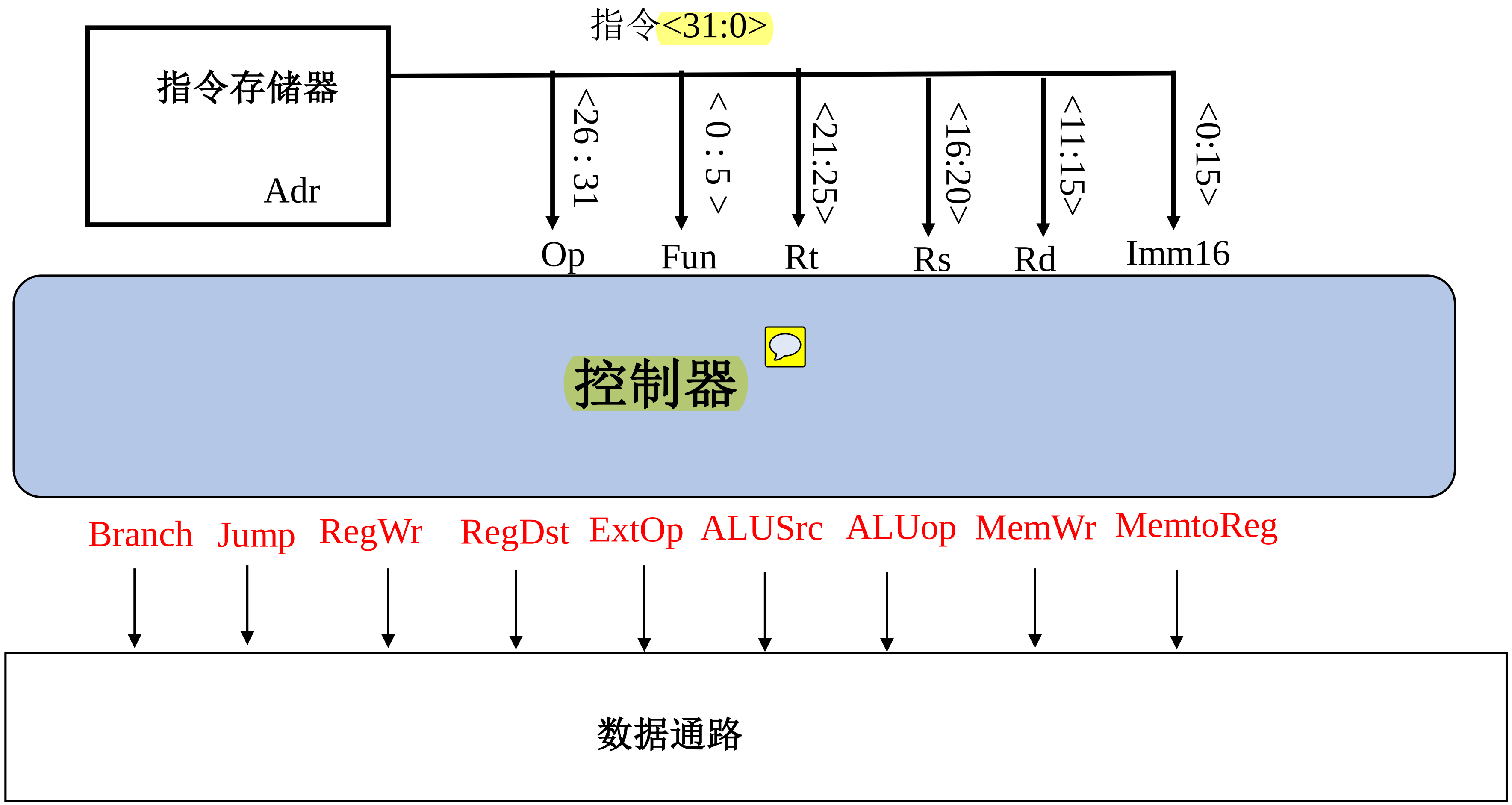
# 控制逻辑

# 处理器设计的五个步骤

1. 分析指令，得出对数据通路的需求
2. 选择数据通路上合适的组件
3. 连接组件构成数据通路
4. 分析每一条指令的实现，以确定控制信号
5. **集成控制信号，完成控制逻辑**

# 集成控制信号


指令<31:0>
指令存储器
Adr
<26 : 31
< 0 : 5 >
<21:25>
<16:20>
<11:15>
<0:15>
Op
Fun
Rt
Rs
Rd
Imm16
控制器
Branch
Jump
RegWr
RegDst
ExtOp
ALUSrc
ALUop
MemWr
MemtoReg
数据通路

# 专用通路结构控制信号总结

指令　　数据通路和控制信号：

ADD　R[rd] <– R[rs] + R[rt];　　PC <– PC + 4

Branch = 0 , Jump=0, ALUsrc = BusB , Extop=x, ALUctr = “add”,
Memwrite=0, MemtoReg=ALU, RegDst = rd, RegWr=1

Ori　R[rd] <– R[rs] or R[rt];　　PC <– PC + 4

PC_source = 0, Jump=0, Extop = “Sn”, ALUsrc = Im, ALUCtr = “or”,
Memwite=0, MemtoReg=ALU, RegDst = rt, RegWr=1

LOAD　R[rt] <– MEM[ R[rs] + sign_ext(Imm16)];　　PC <– PC + 4

Branch = 0 , Jump=0, Extop = “Sn”, ALUsrc = Im, ALUctr= “add”,
Memwrite=0, MemtoReg=Mem, RegDst = rt, RegWr=1

STORE　MEM[ R[rs] + sign_ext(Imm16)] <– R[rs];　　PC <– PC + 4

Branch = 0 , Jump=0, Extop = “Sn”, ALUsrc = Im, ALUctr = “add”,
Memwrite=1, MemtoReg=x, RegDst = x, RegWr=0

BEQ　if ( R[rs] == R[rt] ) then PC <– PC + sign_ext(Imm16)] || 00 else PC <– PC + 4

ALUsrc = BusB , Extop = “Sn”, ALUctr = “sub” , Branch = “Br”, Jump=0,
Memwrite=0, Regwrite=0， MemtoReg=x, RegDst = x,

JUMP　PC <– (PC +4[31-28], $I_{25\text{-}0}$ ) || 00

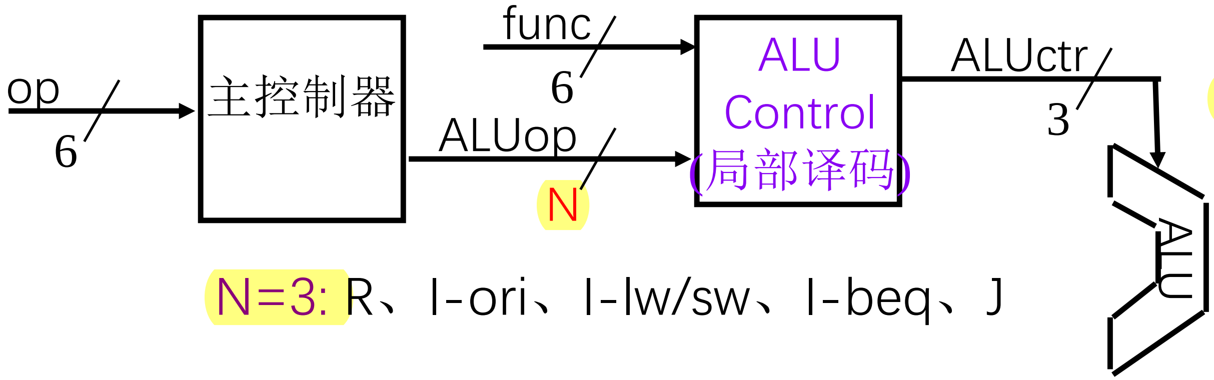
Jump=1, Memwrite=0, Regwrite=0， 其余=x

# 控制信号总结

| func | 10 0000 | 10 0010 | 无关项 | | | | |
|---|---|---|---|---|---|---|---|
| op | 00 0000 | 00 0000 | 00 1101 | 10 0011 | 10 1011 | 00 0100 | 00 0010 |
| | add | sub | ori | lw | sw | beq | jump |
| RegDst | 1 | 1 | 0 | 0 | x | x | x |
| ALUSrc | 0 | 0 | 1 | 1 | 1 | 0 | x |
| MemtoReg | 0 | 0 | 0 | 1 | x | x | x |
| RegWrite | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| MemWrite | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Branch | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Jump | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| ExtOp | x | x | 0 | 1 | 1 | 1 | x |
| ALUctr<2:0> | Add | Subtr | Or | Add | Add | Subtr | xxx |

# ALU Control的局部译码

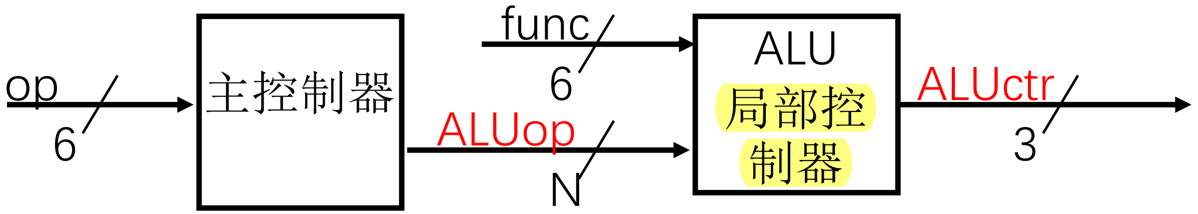
| op | 00 0000 | 00 1101 | 10 0011 | 10 1011 | 00 0100 | 00 0010 |
|---|---|---|---|---|---|---|
| | R-type | ori | lw | sw | beq | jump |
| RegDst | 1 | 0 | 0 | x | x | x |
| ALUSrc | 0 | 1 | 1 | 1 | 0 | x |
| MemtoReg | 0 | 0 | 1 | x | x | x |
| RegWrite | 1 | 1 | 1 | 0 | 0 | 0 |
| MemWrite | 0 | 0 | 0 | 1 | 0 | 0 |
| Branch | 0 | 0 | 0 | 0 | 1 | 0 |
| Jump | 0 | 0 | 0 | 0 | 0 | 1 |
| ExtOp | x | 0 | 1 | 1 | x | x |
| ALUctr | Add/Subtr | Or | Add | Add | Subtr | xxx |



N=3: R、I-ori、I-lw/sw、I-beq、J

ALUctr 由ALUop 和 func 决定

# ALUop的编码

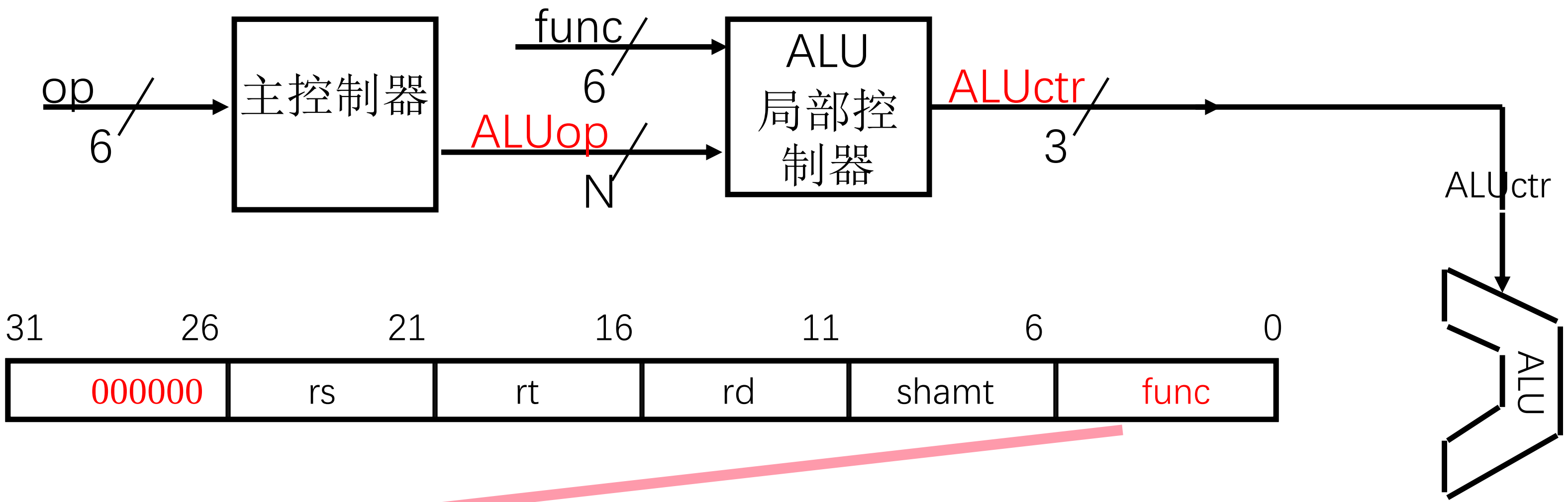


对 ALUop 编码(N=3）

| | R-type | ori | lw | sw | beq | jump |
|---|---|---|---|---|---|---|
| ALUop (Symbolic) | “R-type” | Or | Add | Add | Subtr | xxx |
| ALUop<2:0> | 1 xx | 0 10 | 0 00 | 0 00 | 0x1 | xxx |

ALUop 也可以只用两位（N=2）：
J-xx , R:11, ori:10, beq:01, lw/sw:00,

# ALUctr的编码



| | 31 | 26 | 21 | 16 | 11 | 6 | 0 |
|---|---|---|---|---|---|---|---|
| R-type | 000000 | rs | rt | rd | shamt | func | |

| func<5:0> | 指令的操作 |
|---|---|
| 10 0000 | add |
| 10 0010 | subtract |
| 10 0100 | and |
| 10 0101 | or |
| 10 1010 | set-on-less-than |

| ALUctr<2:0> | ALU 运算 |
|---|---|
| 000 | Add |
| 001 | Subtract |
| 100 | And |
| 101 | Or |
| 010 | Subtract |

# ALUctr 的真值表

R型指令由
func决定ALUctr

非R型指令由
ALUop决定ALUctr

| ALUop<br>(符号) | R-型 | ori | lw | sw | beq |
|---|---|---|---|---|---|
|  | “R-type” | Or | Add | Add | Subtr |
| ALUop<2:0> | 1 00 | 0 10 | 0 00 | 0 00 | 0 x1 |

| funct<3:0> | 指令的运算操作 |
|---|---|
| 0000 | add |
| 0010 | subtract |
| 0100 | and |
| 0101 | or |
| 1010 | set-on-less-than |

| ALUop |  |  | func |  |  |  | ALU<br>运算操作 | ALUctr |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| bit2 | bit1 | bit0 | bit<3> | bit<2> | bit<1> | bit<0> |  | bit<2> | bit<1> | bit<0> |
| 0 | 0 | 0 | x | x | x | x | Add | 0 | 0 | 0 |
| 0 | x | 1 | x | x | x | x | Subtract | 0 | 0 | 1 |
| 0 | 1 | 0 | x | x | x | x | Or | 1 | 1 | 0 |
| 1 | x | x | 0 | 0 | 0 | 0 | Add | 0 | 0 | 0 |
| 1 | x | x | 0 | 0 | 1 | 0 | Subtract | 0 | 0 | 1 |
| 1 | x | x | 0 | 1 | 0 | 0 | And | 0 | 1 | 0 |
| 1 | x | x | 0 | 1 | 0 | 1 | Or | 1 | 1 | 0 |
| 1 | x | x | 1 | 0 | 1 | 0 | Subtract | 0 | 0 | 1 |

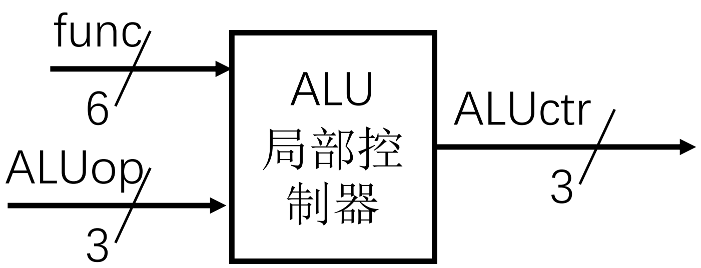
# ALUctr<0> 的逻辑表达式

ALUctr[0]=1 所在的行

| ALUop | | | func | | | | |
|---|---|---|---|---|---|---|---|
| bit<2> | bit<1> | bit<0> | bit<3> | bit<2> | bit<1> | bit<0> | ALUctr<0> |
| 0 | x | 1 | x | x | x | x | 1 |
| 1 | x | x | 0 | 0 | 1 | 0 | 1 |
| 1 | x | x | 1 | 0 | 1 | 0 | 1 |

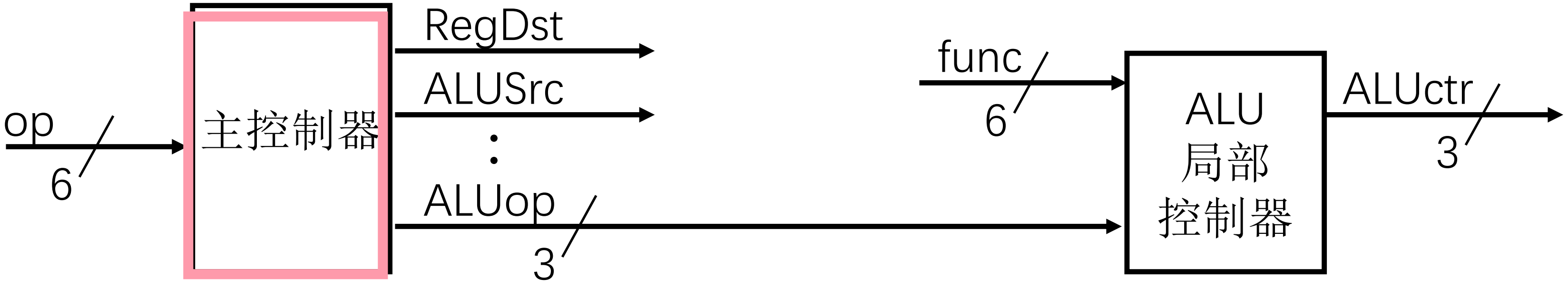
ALUctr<0> = !ALUop<2> & ALUop<0> +
ALUop<2> & !func<2> & func<1> & !func<0>

# ALU Control 控制信号汇总



- ALUctr<0> = !ALUop<2> & ALUop<0> +
  ALUop<2> & !func<2> & func<1> & !func<0>
- ALUctr<1> = !ALUop<2> & ALUop<1> & !ALUop<0> +
  ALUop<2> & !func<3> & func<2> & !func<1>
- ALUctr<2> = !ALUop<2> & ALUop<1> & !ALUop<0> +
  ALUop<2> & !func<3> & func<2> & !func<1> & func<0>

# Main Control的真值表



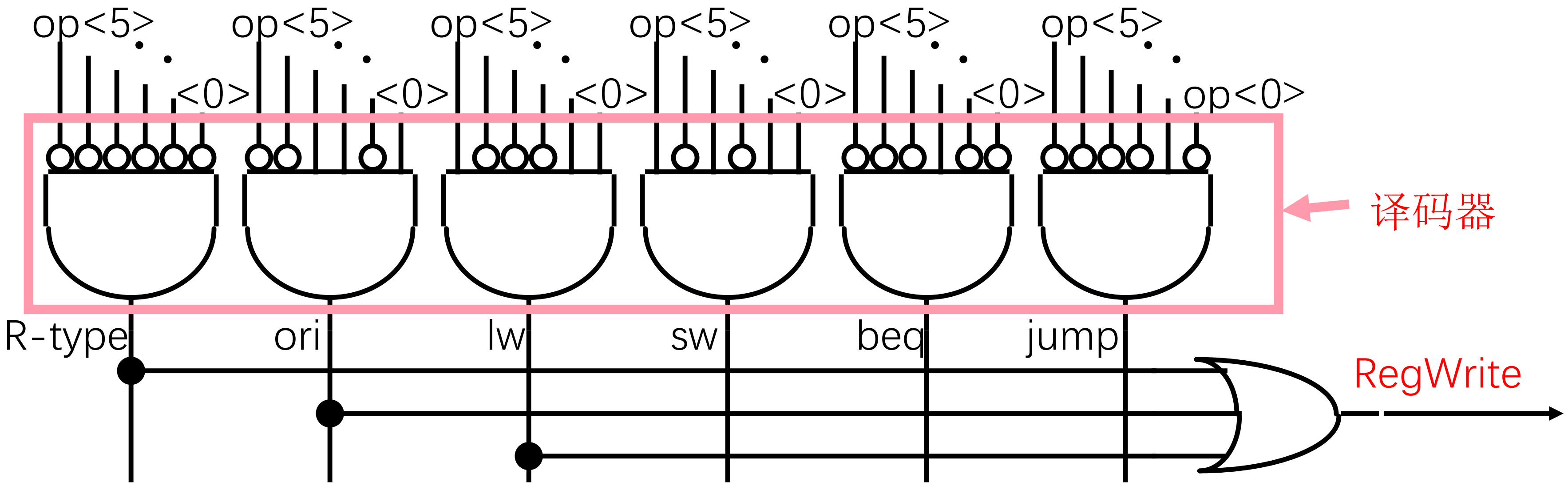
主控制器的输入

| op | 00 0000 | 00 1101 | 10 0011 | 10 1011 | 00 0100 | 00 0010 |
|---|---|---|---|---|---|---|
|  | R-type | ori | lw | sw | beq | jump |
| RegDst | 1 | 0 | 0 | x | x | x |
| ALUSrc | 0 | 1 | 1 | 1 | 0 | x |
| MemtoReg | 0 | 0 | 1 | x | x | x |
| RegWrite | 1 | 1 | 1 | 0 | 0 | 0 |
| MemWrite | 0 | 0 | 0 | 1 | 0 | 0 |
| Branch | 0 | 0 | 0 | 0 | 1 | 0 |
| Jump | 0 | 0 | 0 | 0 | 0 | 1 |
| ExtOp | x | 0 | 1 | 1 | x | x |
| ALUop (符号) | “R-型” | Or | Add | Add | Subtr | xxx |
| ALUop <2> | 1 | 0 | 0 | 0 | 0 | x |
| ALUop <1> | x | 1 | 0 | 0 | x | x |
| ALUop <0> | x | 0 | 0 | 0 | 1 | x |

主控制器的输出

# RegWrite 信号的真值表

| op | 00 0000 | 00 1101 | 10 0011 | 10 1011 | 00 0100 | 00 0010 |
|---|---|---|---|---|---|---|
|  | R-type | ori | lw | sw | beq | jump |
| RegWrite | 1 | 1 | 1 | 0 | 0 | 0 |



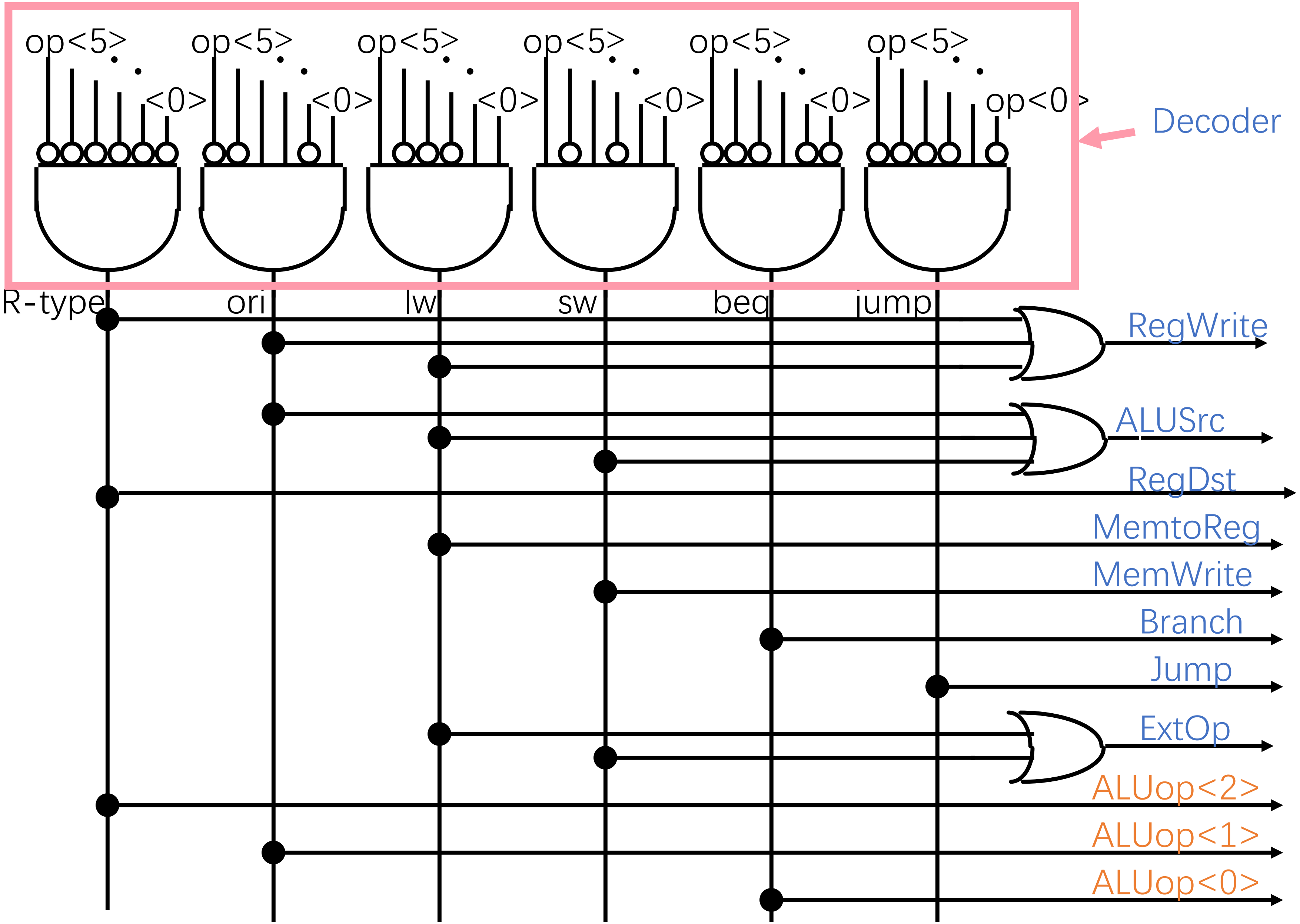
RegWrite = R型 + ori + lw

= !op<5> & !op<4> & !op<3> & !op<2> & !op<1> & !op<0> (R型)

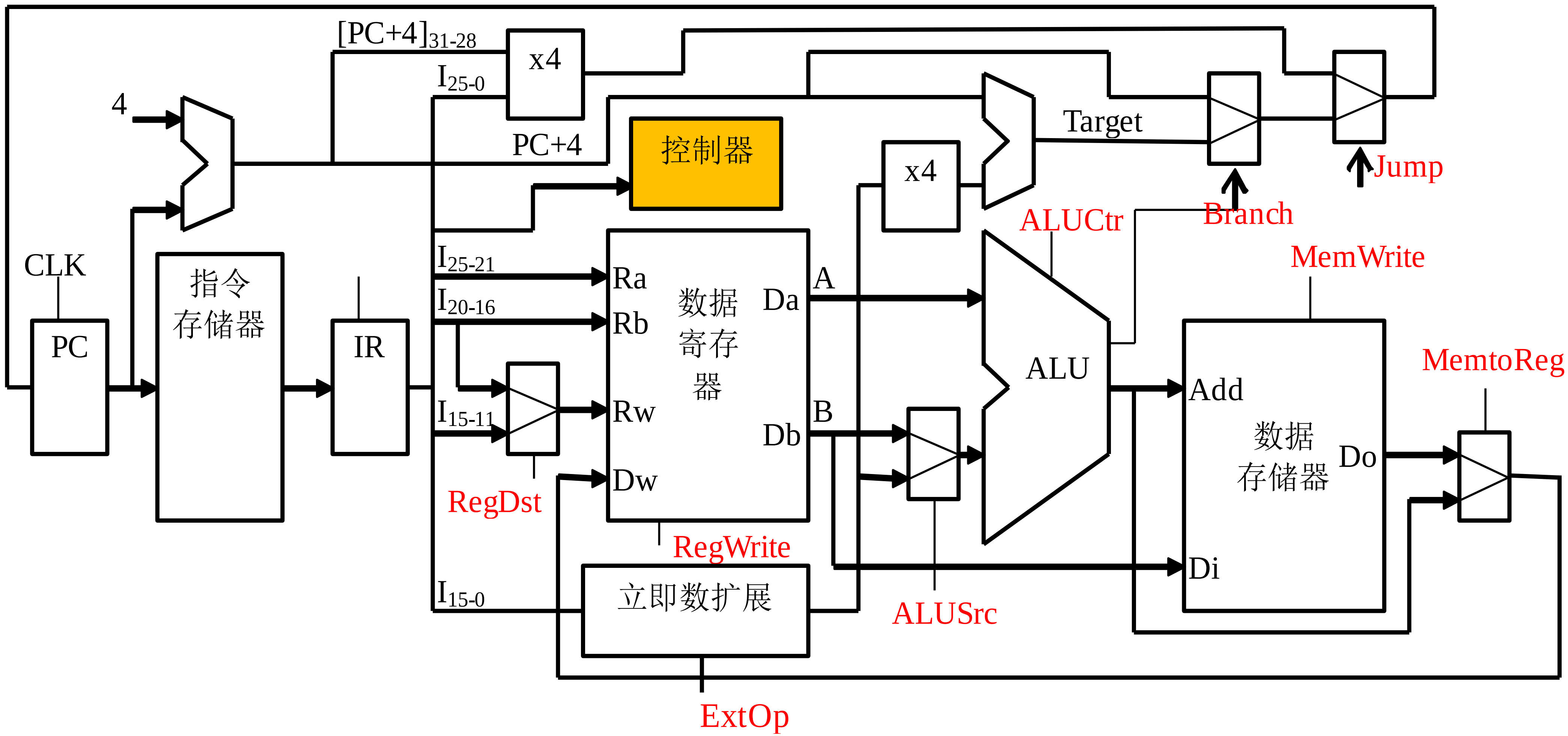
+ !op<5> & !op<4> & op<3> & op<2> & !op<1> & op<0> (ori)

+ op<5> & !op<4> & !op<3> & !op<2> & op<1> & op<0> (lw)

# 其他控制信号真值表


op<5>
<0>
op<0>
Decoder
R-type
ori
lw
sw
beq
jump
RegWrite
ALUSrc
RegDst
MemtoReg
MemWrite
Branch
Jump
ExtOp
ALUop<2>
ALUop<1>
ALUop<0>

# 小结


[PC+4]31-28
x4
I25-0
4
PC+4
控制器
Target
Jump
x4
Branch
ALUCtr
MemWrite
CLK
PC
指令存储器
IR
I25-21
I20-16
Ra
Rb
数据寄存器
Da
Db
A
B
ALU
I15-11
Rw
Dw
RegDst
Add
数据存储器
Do
Di
MemtoReg
RegWrite
立即数扩展
I15-0
ALUSrc
ExtOp