MIPS 单周期处理器设计

（实验课相关要求：lab5）

处理器设计目标

- 能实现几条简单指令的处理器

  算术逻辑运算指令: **add, sub, ori ,** **（实验要求多三条：** and , or, slt **）**

  访问存储器: **lw, sw**

  控制流指令: **beq, j**

MIPS指令的子集

- 加法与减法
  - add rd, rs, rt
  - sub rd, rs, rt
- OR 立即数:
  - ori rt, rs, imm16
- 读内存 和 写内存
  - lw rt, rs, imm16
  - sw rt, rs, imm16
- 条件转移:
  - beq rs, rt, imm16
- 无条件转移:
  - j target

rs或Imm → rt

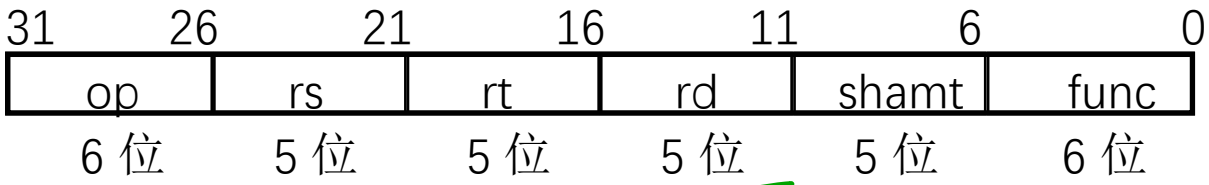
字节偏移

PC+4+Imm×4

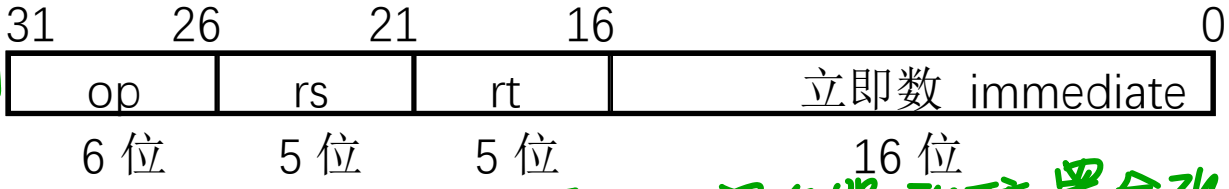
RISC-V:

PC+Imm×2

R 型:



I 型:

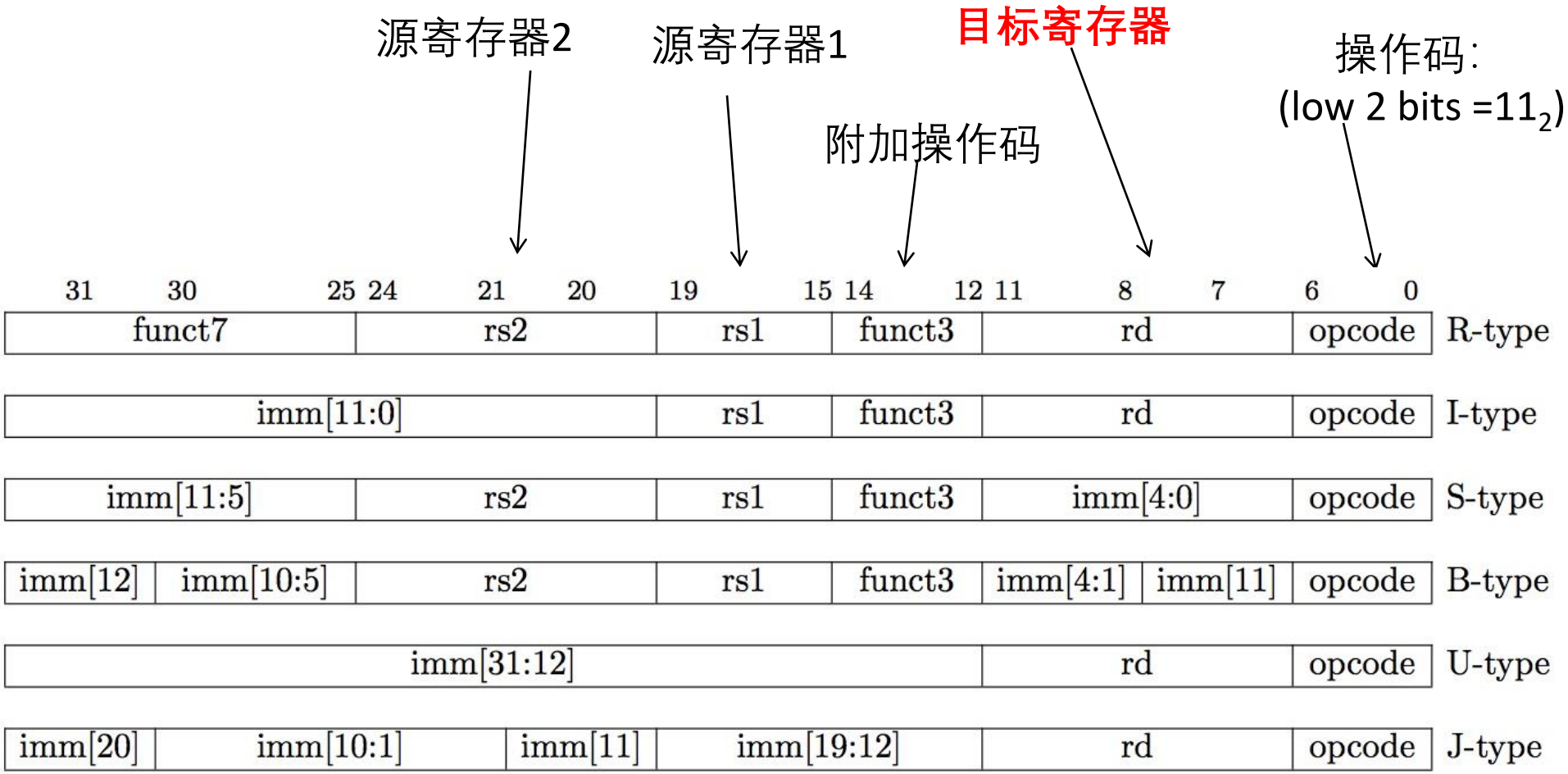


目标寄存器的位置会改变

J 型:

4位被

对比：RV32I 指令格式



分析各条指令的数据通路

指令的执行过程

取指令

指令译码

指令执行

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

ORi　　R[rt] <– R[rs] | zero_ext(Imm16);　　PC <– PC + 4

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

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

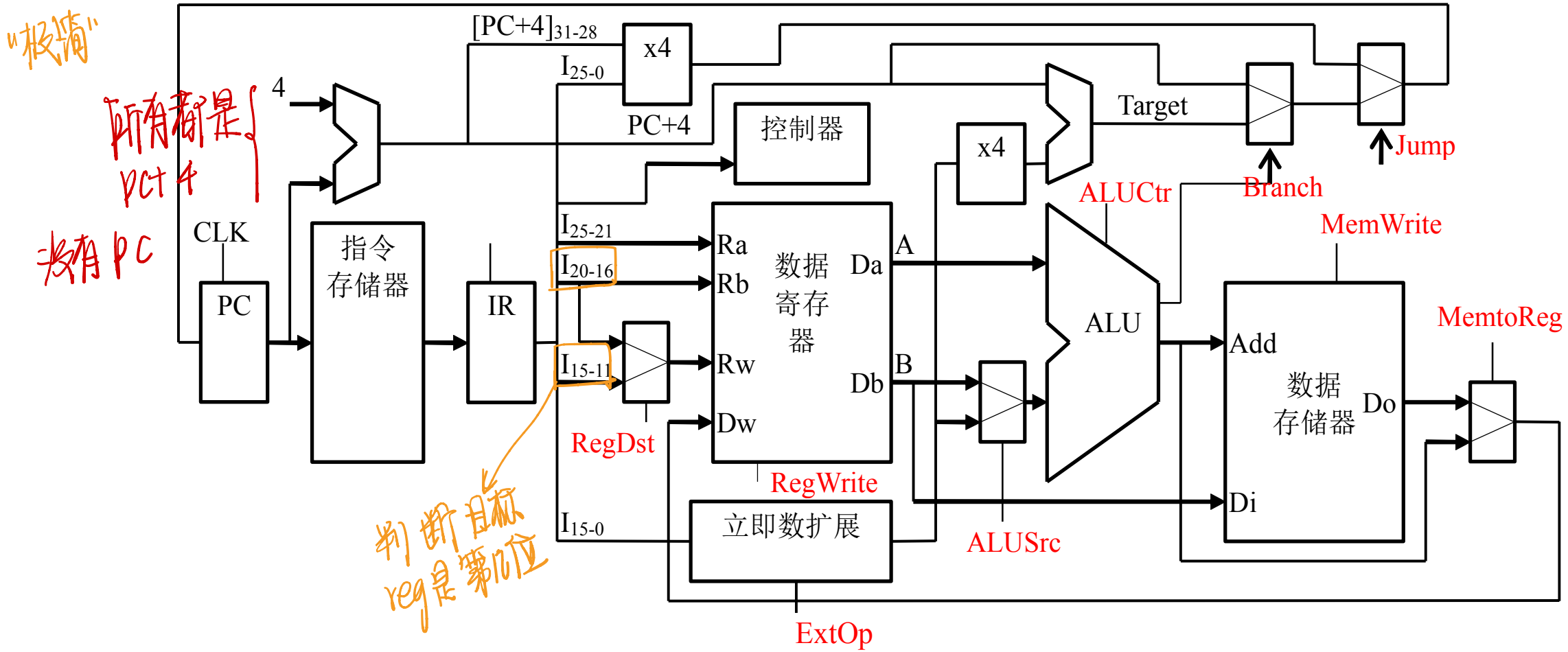
BEQ　　if (R[rs] == R[rt]) then　PC <– **PC + 4 +sign_ext(Imm16)] || 00**

else　PC <– PC + 4

JUMP　　PC **<– (PC +4[31-28], I25-0) || 00**

字节偏移量

单周期处理器



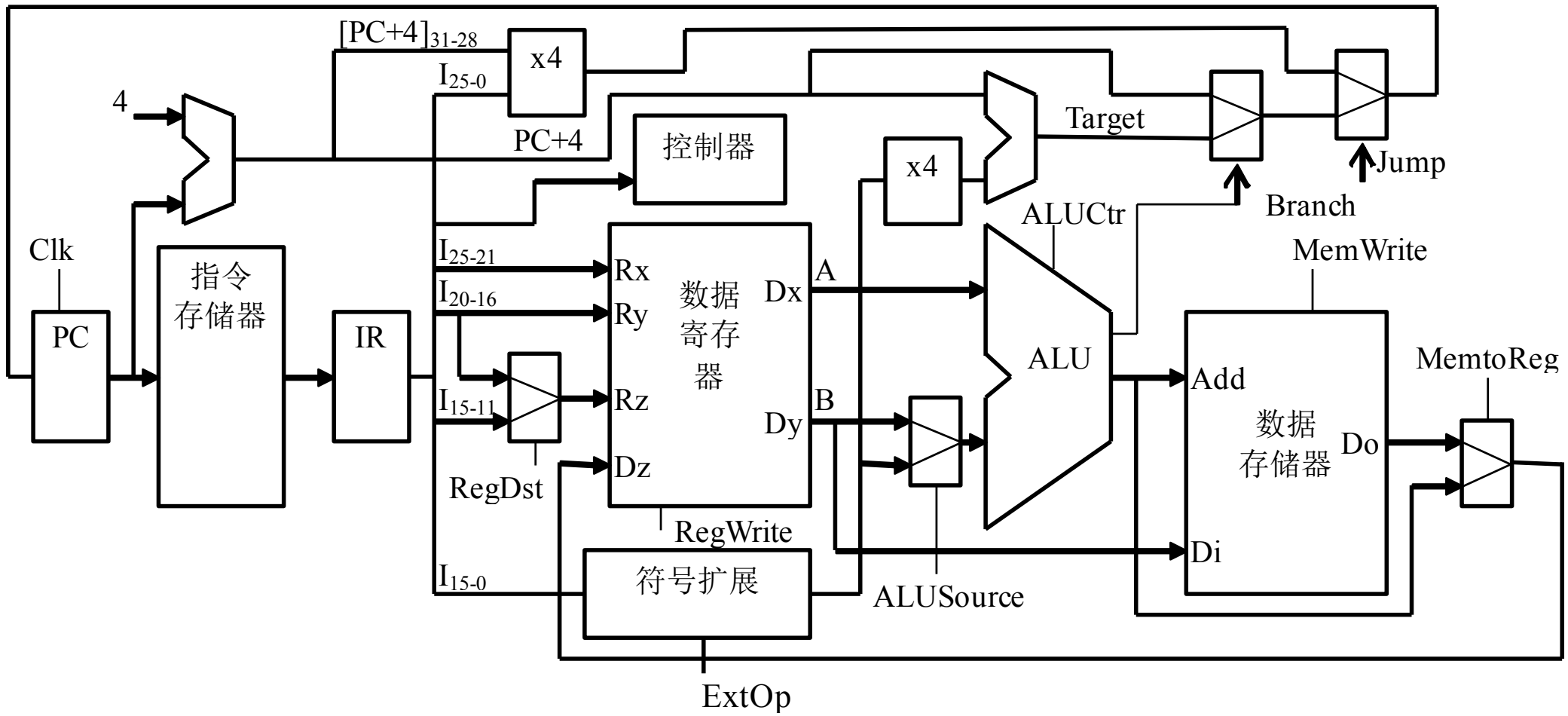
控制信号:

ALUCtr	运算操作码
ALUSrc	ALU数据选择
ExtOp	无/带符号扩展
Branch	是否为条件转移指令
Jump	是否为无条件转移指令
MemWrite	存储器写
MemtoReg	写数据选择
RegWrite	数据寄存器写
RegDst	写寄存器选择

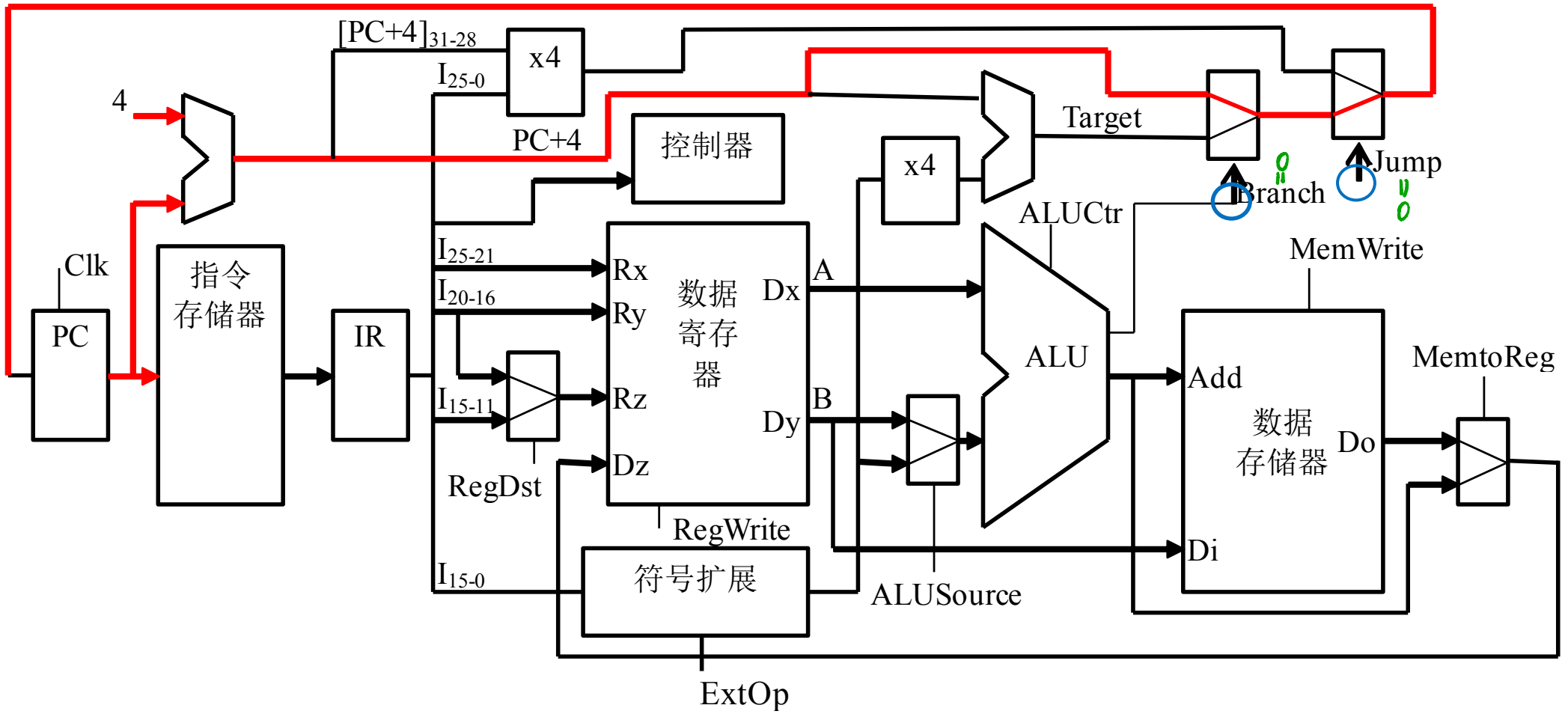
指令　　数据通路

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

控制信号：



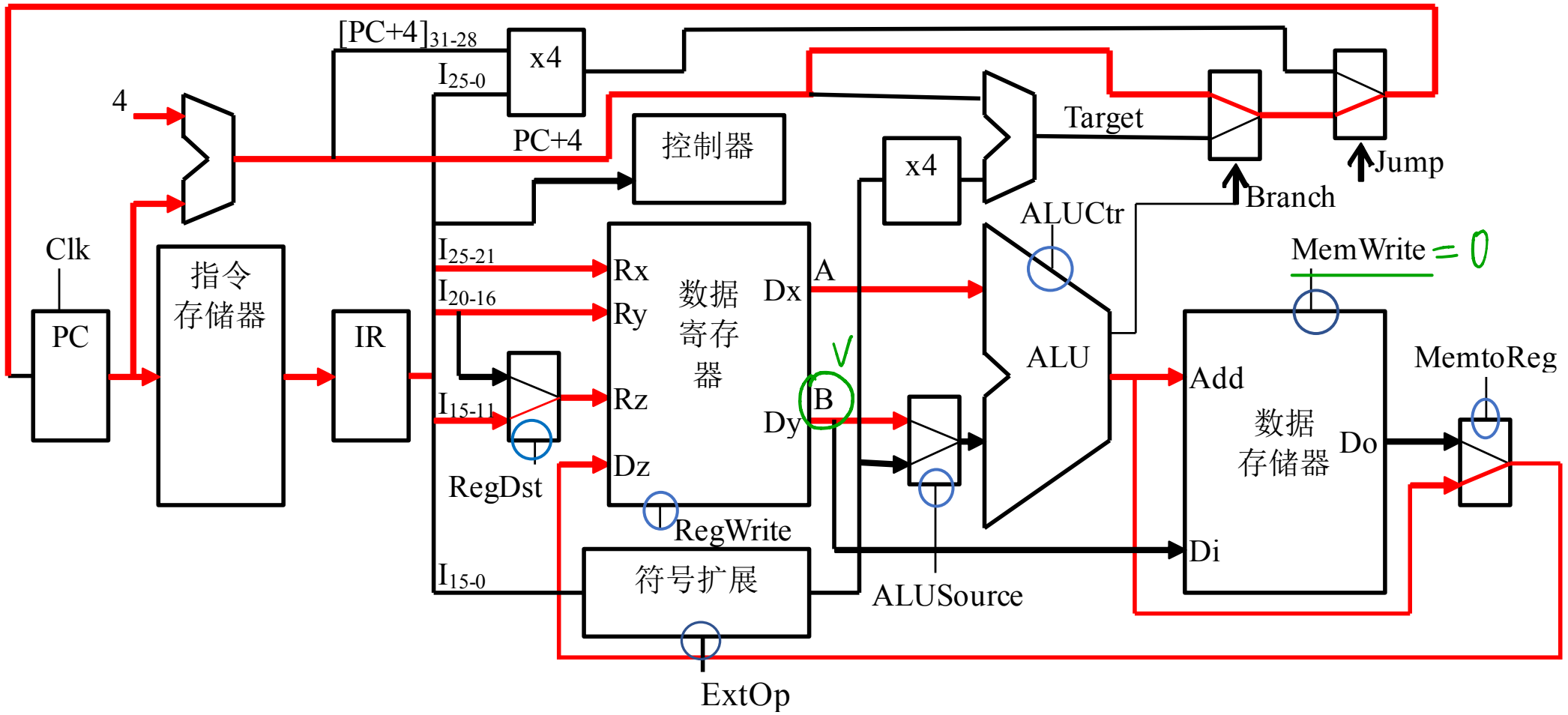
指令	数据通路	
ADD	PC <– PC + 4	R[rd] <– R[rs] + R[rt];
控制信号：	Branch = 0 , Jump=0,	ALUsrc = BusB , ALUctr = “add”, Extop=x, Memwrite=0, MemtoReg=ALU, RegDst = rd, RegWr=1



指令	数据通路
ADD	PC <– PC + 4
控制信号：	Branch = 0 , Jump=0,

R[rd] <– R[rs] + R[rt];

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

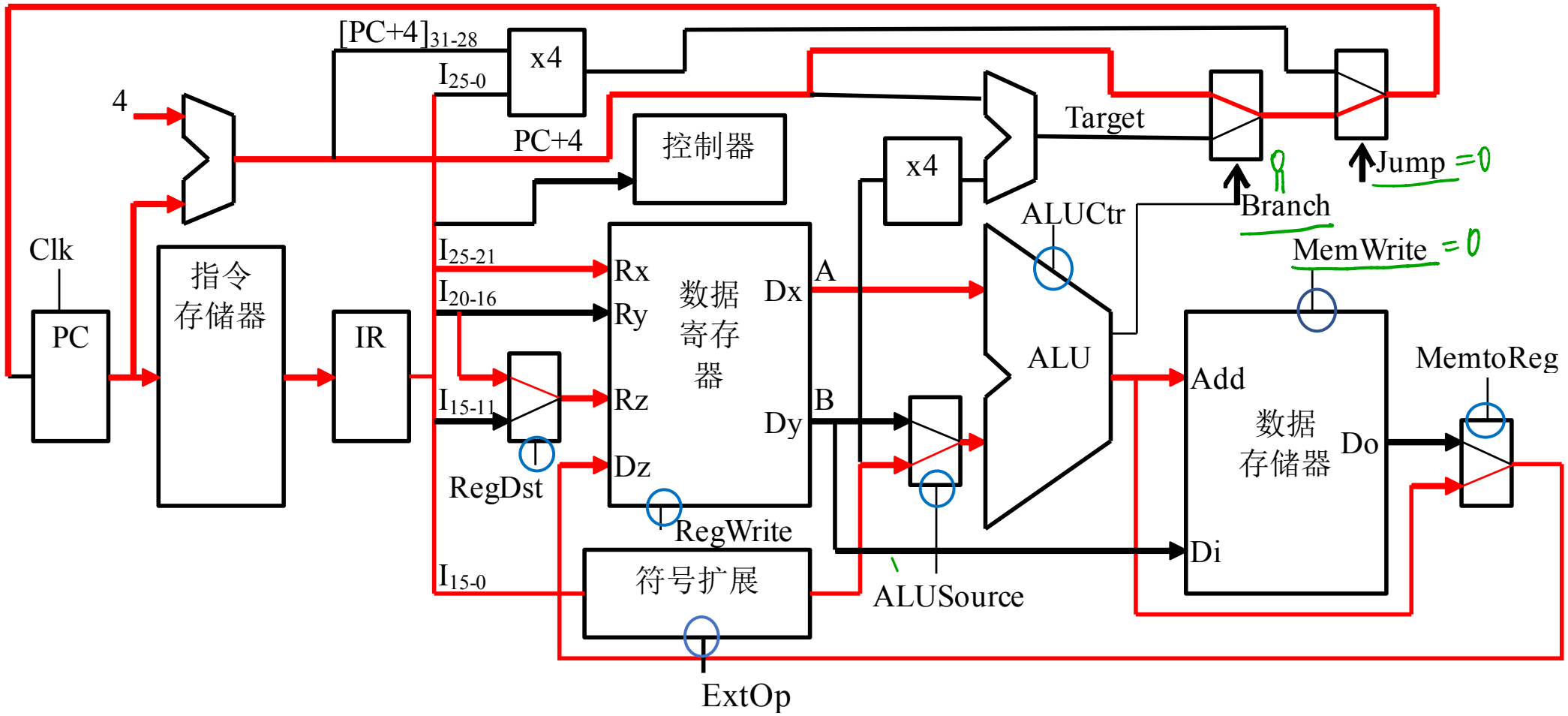


指令　　数据通路

Ori　　PC <– PC + 4

控制信号：　PC_source = 0, Jump=0,

R[rt] <– R[rs] or unsign_ext(Imm16)];
ALUsrc = Im, ALUCtr = “or”, Extop = “unSn，
Memwite=0, MemtoReg=ALU, RegDst = rt, RegWr=1



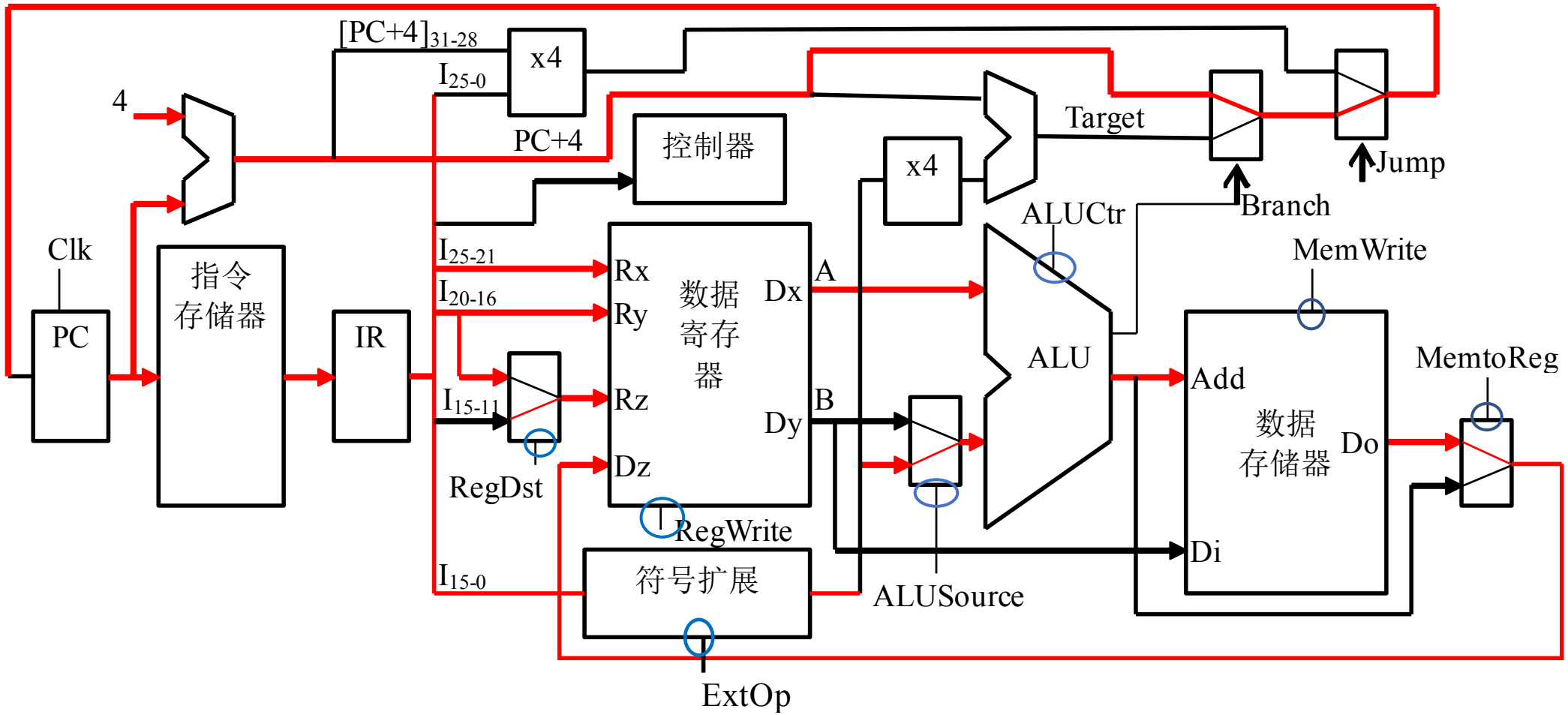
指令　　数据通路

LOAD　　PC <– PC + 4 ，

控制信号：　Branch = 0 , Jump=0,

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

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

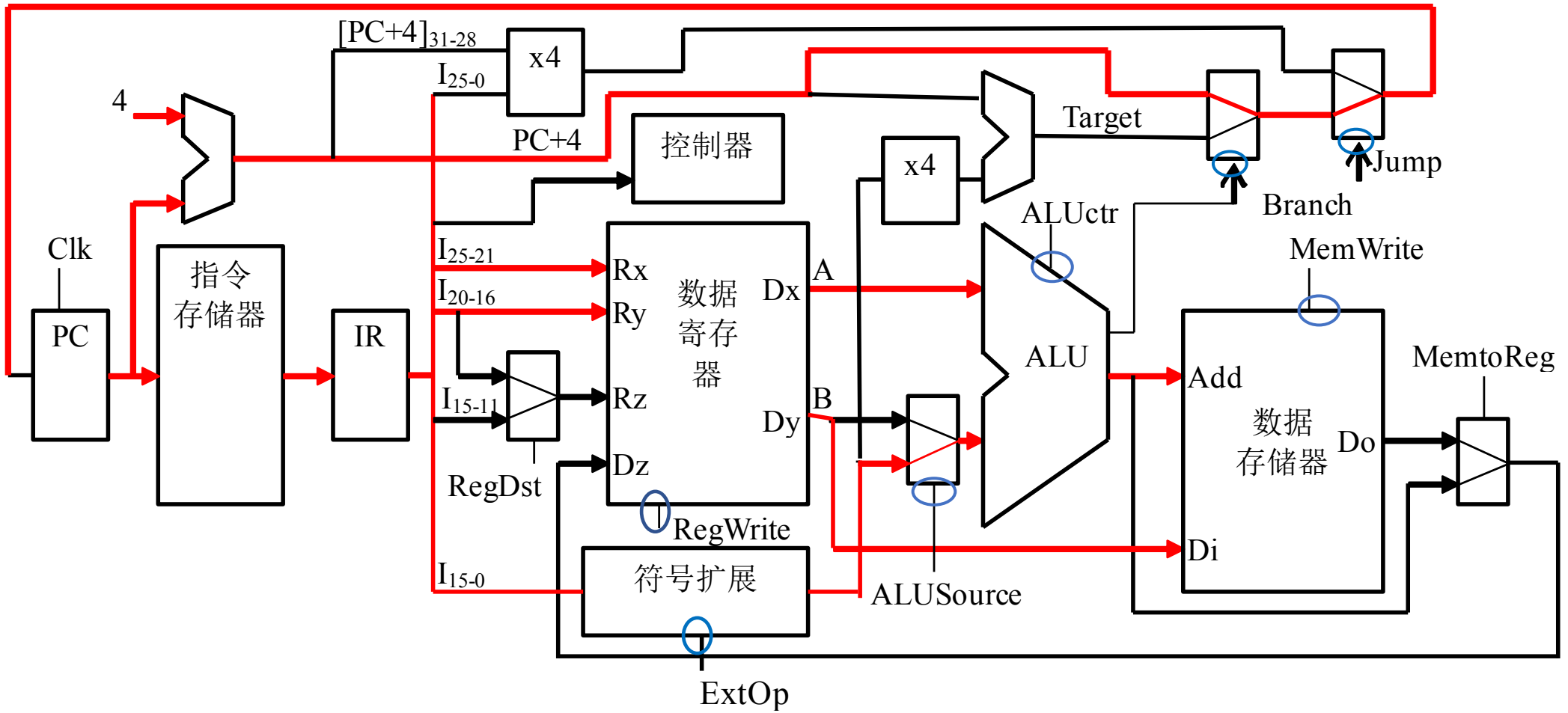


指令　　数据通路

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

控制信号：　Branch = 0 , Jump=0,　　ALUsrc = Im, ALUctr = “add”, Extop = “Sn”,

Memwrite=1, MemtoReg=x, RegDst = x, RegWr=0

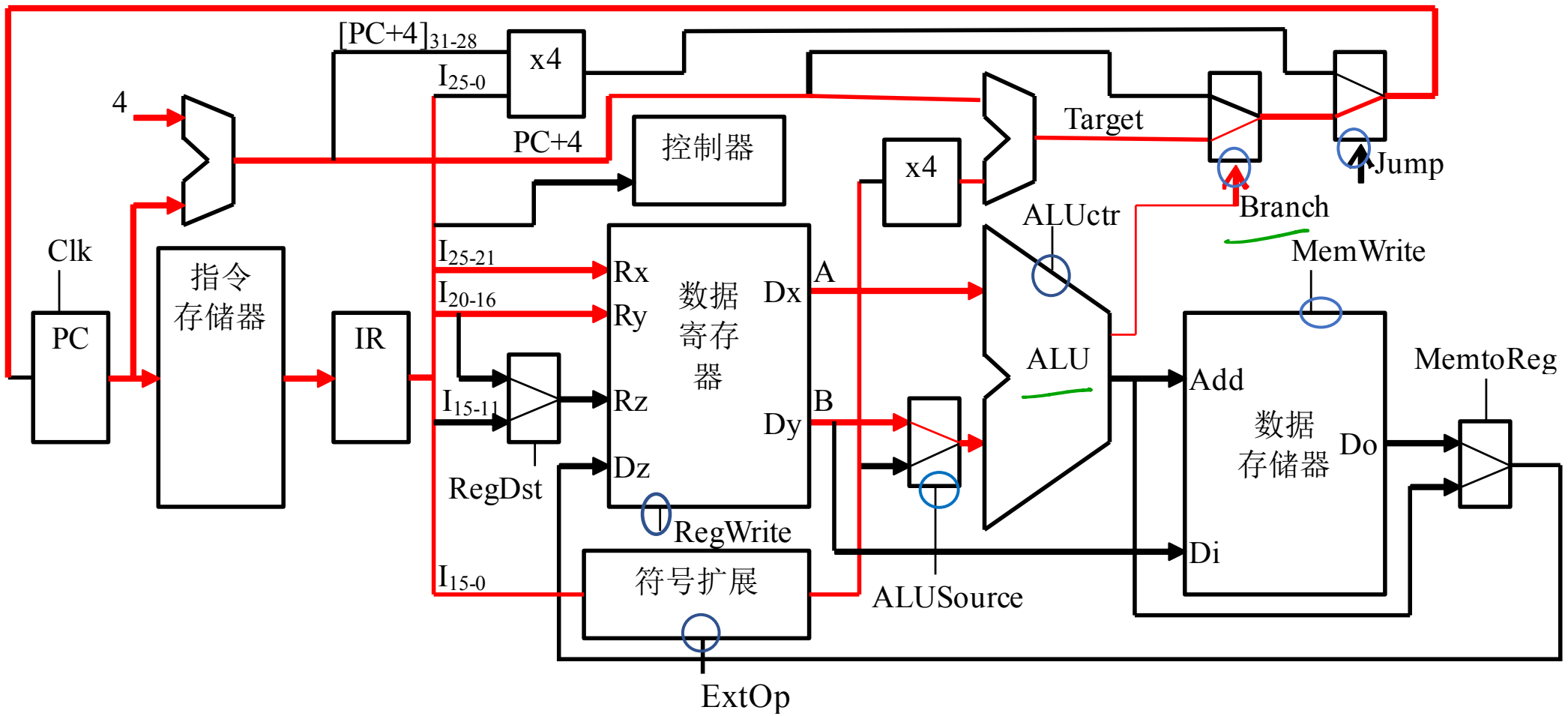


指令　　数据通路

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

控制信号：　ALUsrc = BusB , ALUctr = “sub” , Extop = “Sn”, Branch = “Br”, Jump=0, Memwrite=0, Regwrite=0，其余=x

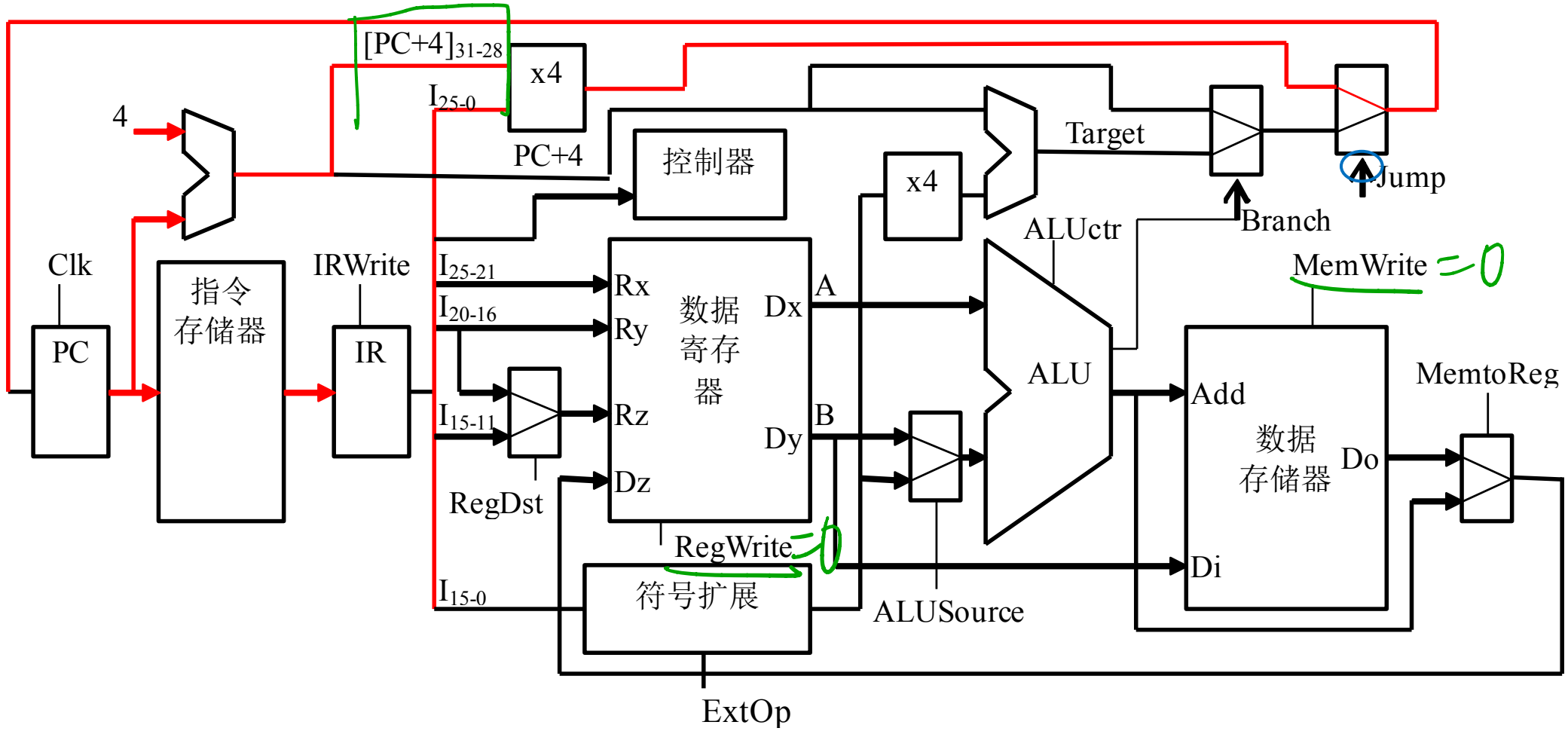
根据ALU算出Branch再改变PC的值



指令　　数据通路

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

控制信号：　Branch=0, Jump=1, Memwrite=0, Regwrite=0，其余=x



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**

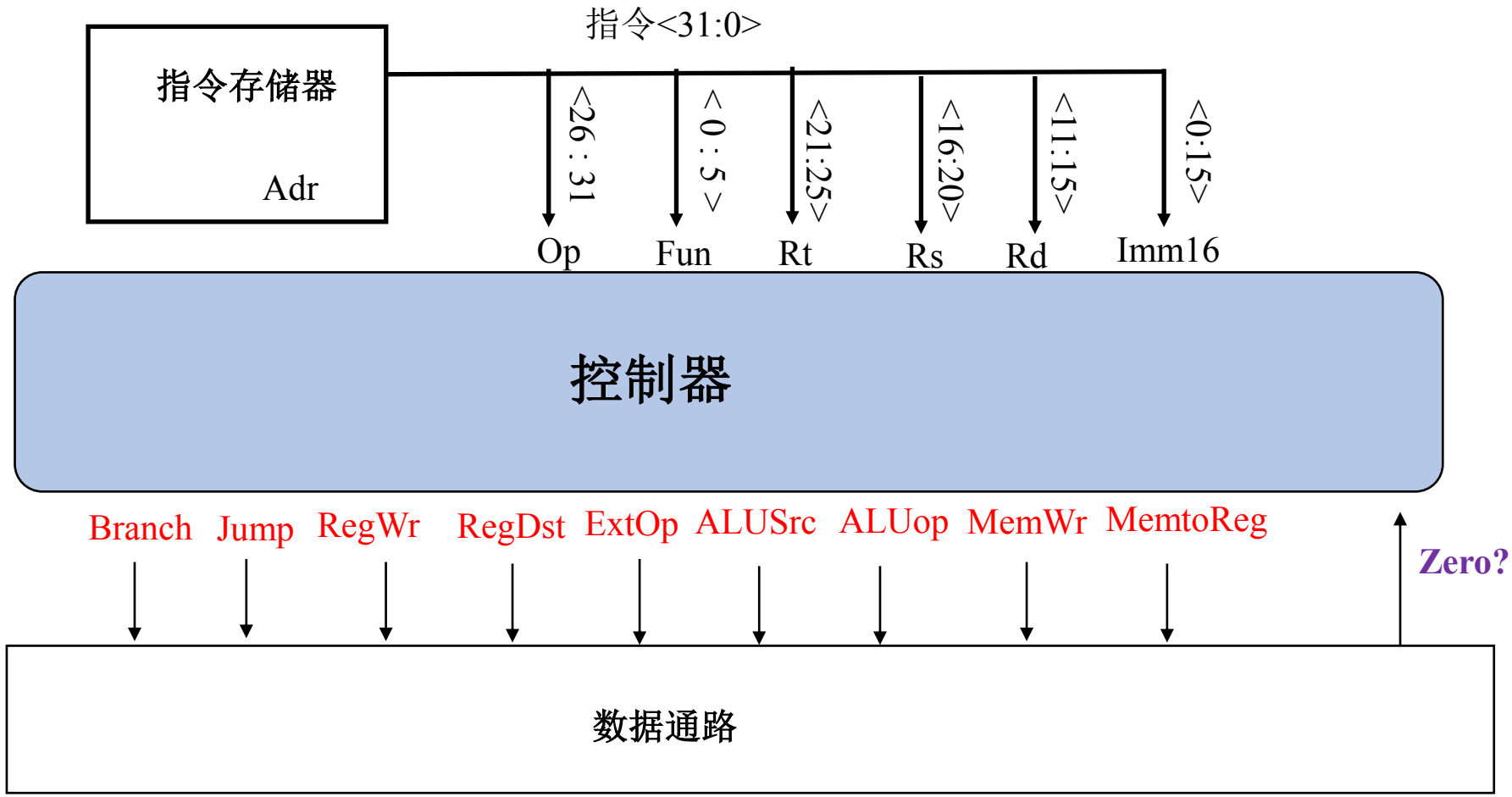
Branch=0, Jump=1, Memwrite=0, Regwrite=0，其余=x

控制信号总结

输入高6位与低6位

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

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
Zero?
数据通路

实验指导书对应内容

以主控制模块为例，以下代码为实例化 Ctr （mainCtr 就是模块名），并连接其端口。其中 INST 是定义好的指令存储器输出的连接信号，其它信号线在 3.1.3 中已完成定义。

```
100     Ctr mainCtr(
101             .opcode(INST[31:26]),
102             .regDst(REG_DST),
103             .jump(JUMP),
104             .branch(BRANCH),
105             .memRead(MEM_READ),
106             .memToReg(MEM_TO_REG),
107             .aLUOp(ALU_OP),
108             .memWrite(MEM_WRITE),
109             .aLUSrc(ALU_SRC),
110             .regWrite(REG_WRITE));
```

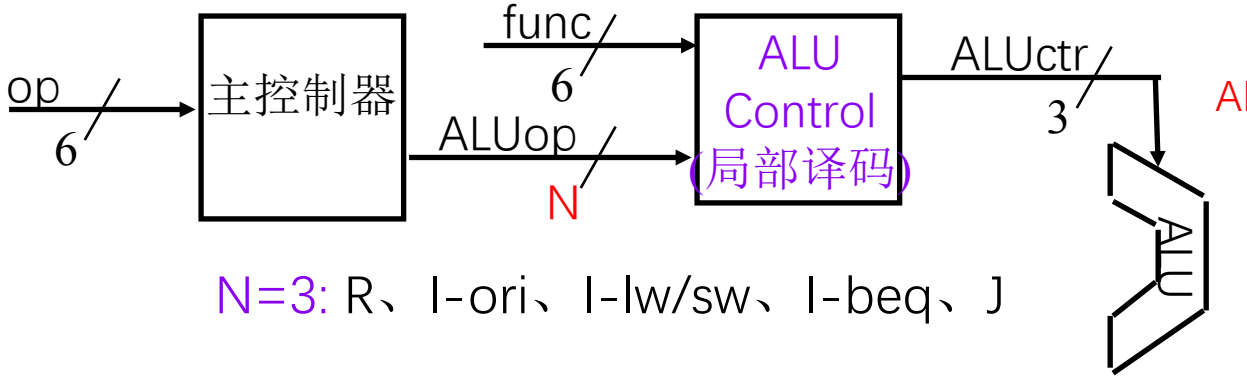
实例化 Ctr

```
21  module Ctr(opcode, regDst, jump, branch, memRead, memToReg, aLUOp, memWrite, aLUSrc, regWrite);
```

Ctr 模块定义

ALU Control的局部译码

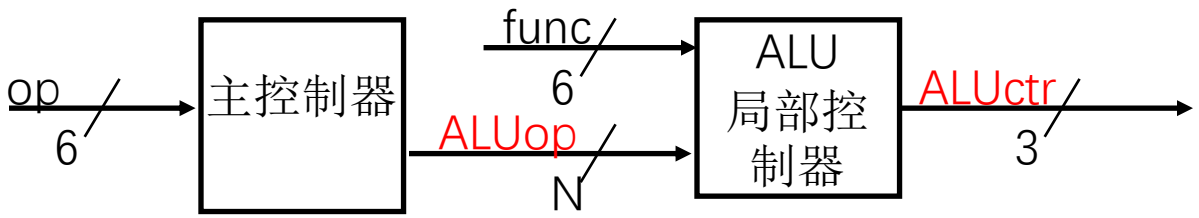
op	00 0000	00 1101	10 0011	10 1011	00 0100	00 0010
	R-type	ori	lw	sw	beq	jump
RegDst	1	0	0	x	x	x
ALUSrc	0	1	1	1	0	x
MemtoReg	0	0	1	x	x	x
RegWrite	1	1	1	0	0	0
MemWrite	0	0	0	1	0	0
Branch	0	0	0	0	1	0
Jump	0	0	0	0	0	1
ExtOp	x	0	1	1	x	x
ALUctr	Add/Subtr	Or	Add	Add	Subtr	xxx



ALUctr 由ALUop 和 func 决定

N=3: R、I-ori、I-lw/sw、I-beq、J

ALUop的编码

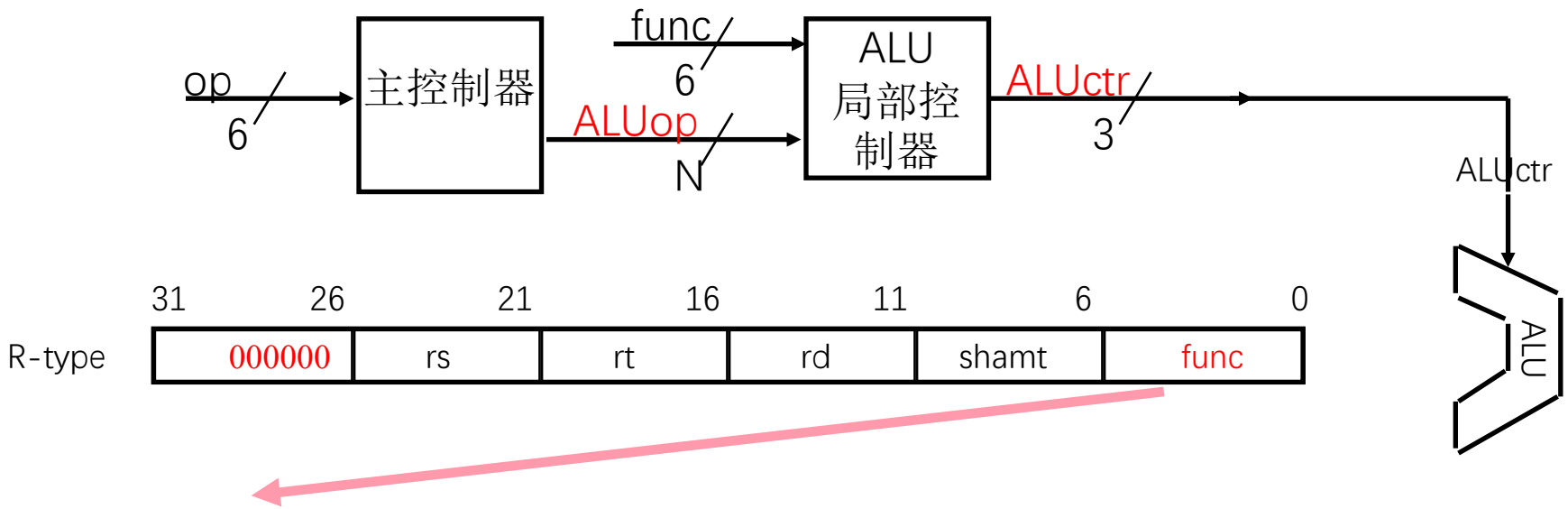


对 ALUop 编码(N=3）

	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



func<5:0>	指令的操作	ALUctr<2:0>	ALU 运算
10 0000	add	000	Add
10 0010	subtract	001	Subtract
10 0100	and	100	And
10 0101	or	101	Or
10 1010	set-on-less-than	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 (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 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

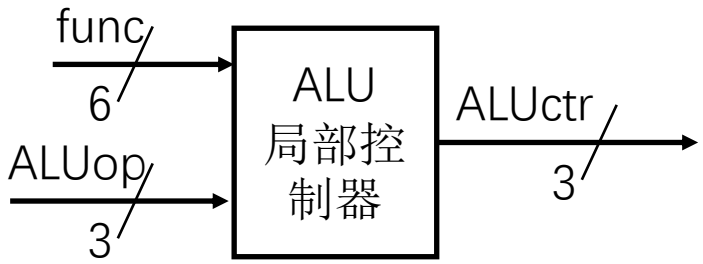
ALUctr[0]=1 所在的行

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> +

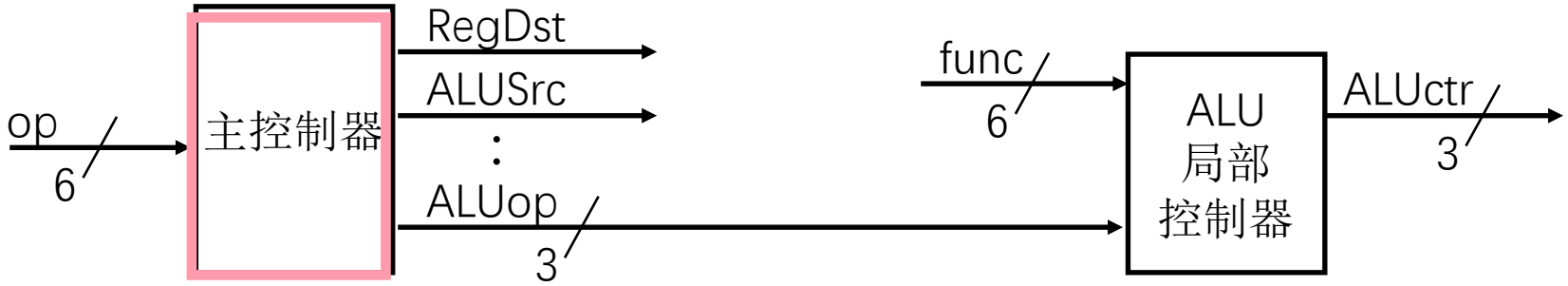
ALUop<2> & !func<2> & func<1> & !func<0>

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



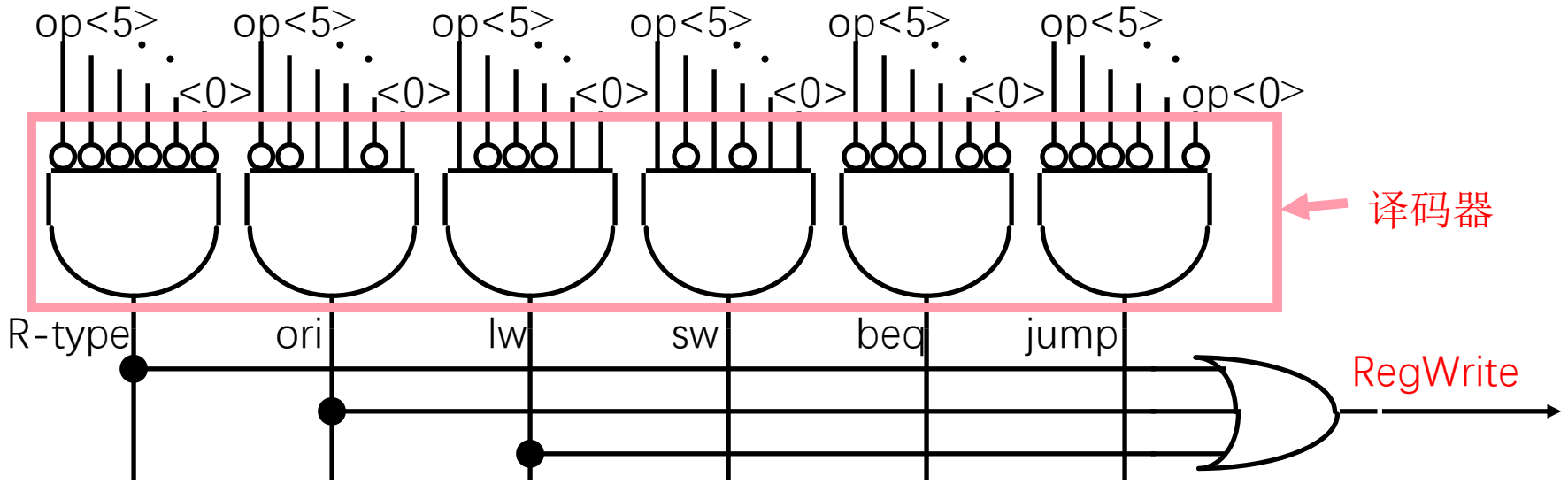
主控制器的输入

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



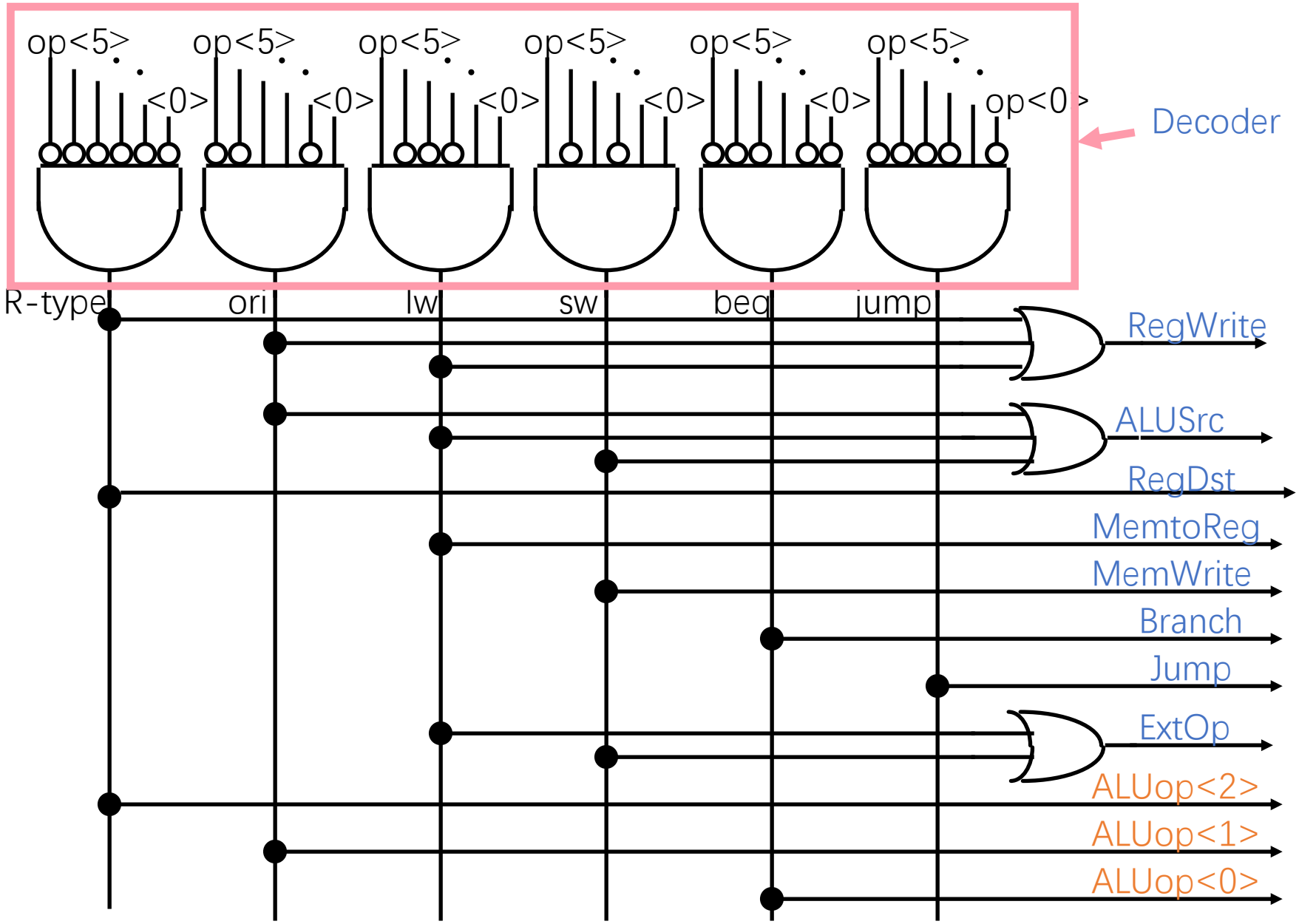
RegWrite = R型 + ori + lw

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

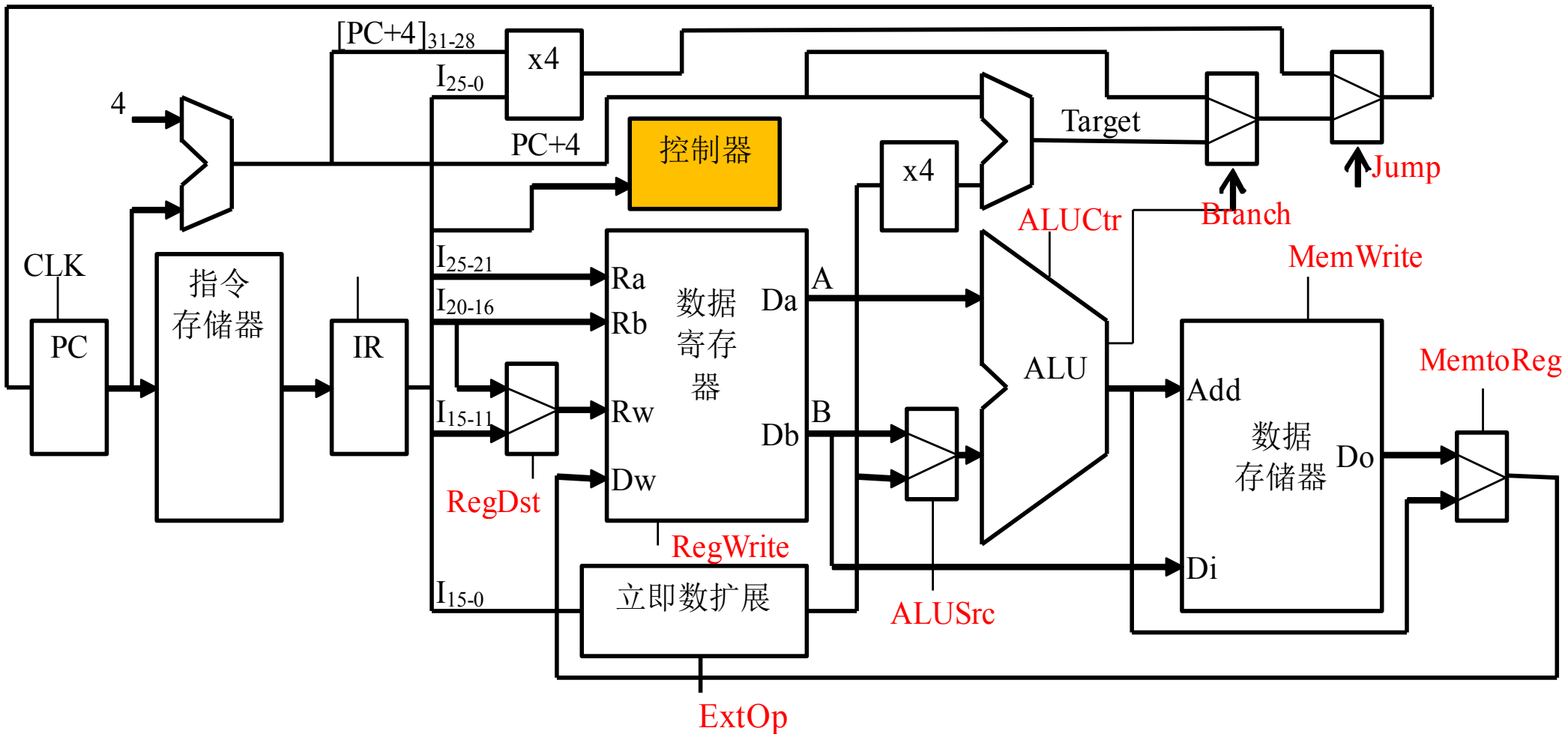
+ !op<5> & !op<4> & op<3> & op<2> & !op<1> & op<0> (ori)

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

思考题

在MIPS 或 RISC-V 指令集中需要增加一条swap指令，可以使用软件方式用若干条已有指令来实现伪指令，也可以通过改动硬件来实现。

（1）写出用伪指令方式实现“swap $rs, $rt”时的指令序列

（2）假定用硬件实现时会使一条指令的执行时间增加10%，则swap指令在程序中占多大的比例才值得用硬件方式来实现

(1) xor $rs, $rs, $rt
xor $rt, $rs, $rt
xor $rs $rs , $rt

(2) X% 节省(1-X)%.
X75%.

设计处理器的五个步骤

1. 分析指令系统，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

MIPS处理器设计教材、参考视频

- 教材：
  - 计算机组成与设计：软硬件接口 （第四版 MIPS版）第四章
  - canvas提供电子版

- 视频资料
  - Canvas 课程主页：提供资料模块
  - 或直接点击：
    https://vshare.sjtu.edu.cn/play/df5093cdbb1b889b8ec057354afda117

MIPS指令集参考视频

- Canvas 课程：补充资料模块

- 或直接点击：
- 邓倩妮 4.1 MIPS指令系统简介 (sjtu.edu.cn)
- 邓倩妮 4.2 MIPS 控制流指令 (sjtu.edu.cn)
- 邓倩妮 4.2 MIPS 控制流指令 (sjtu.edu.cn)