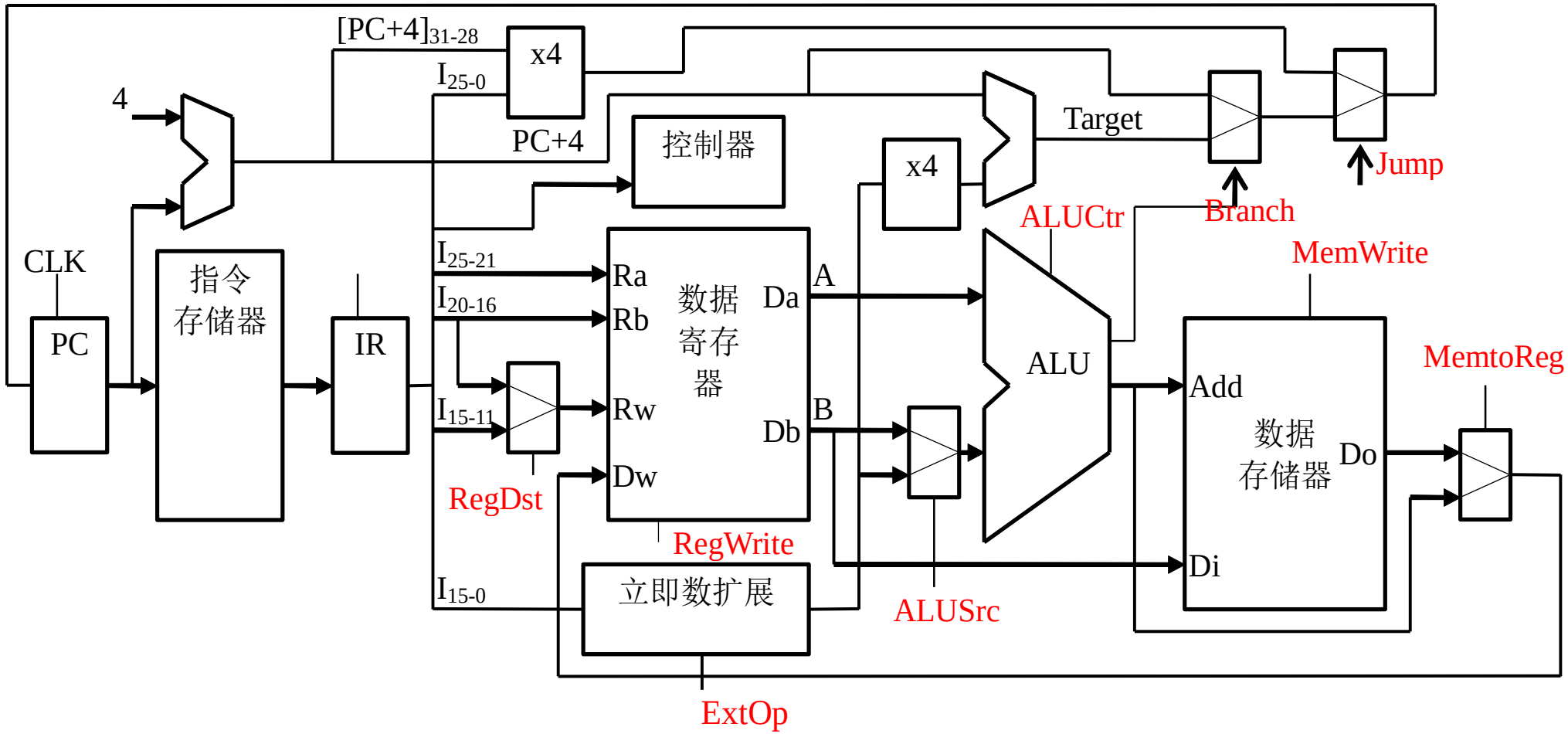
控制信号

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

单周期处理器



控制信号:

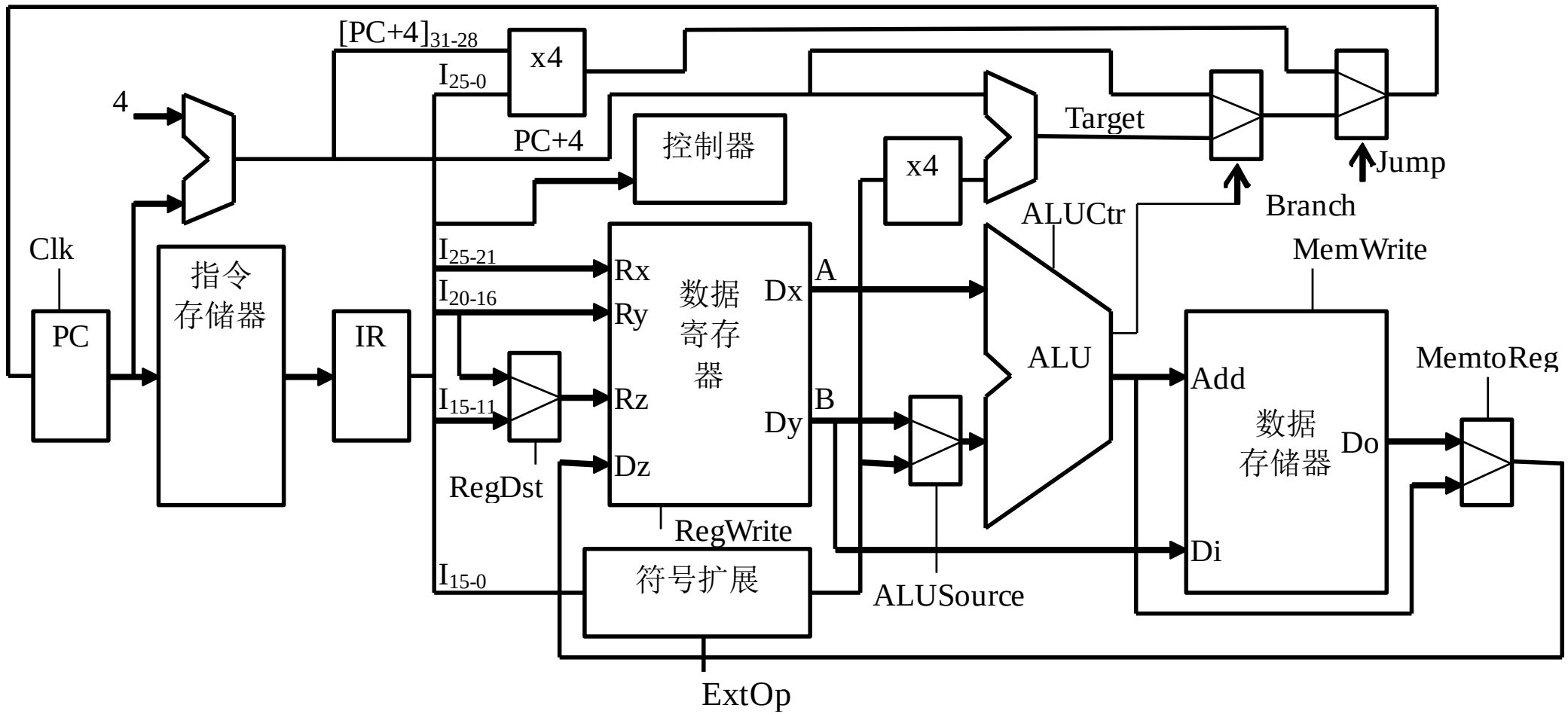
ALUCtr 运算操作码
ALUSrc ALU数据选择
ExtOp 无/带符号扩展

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

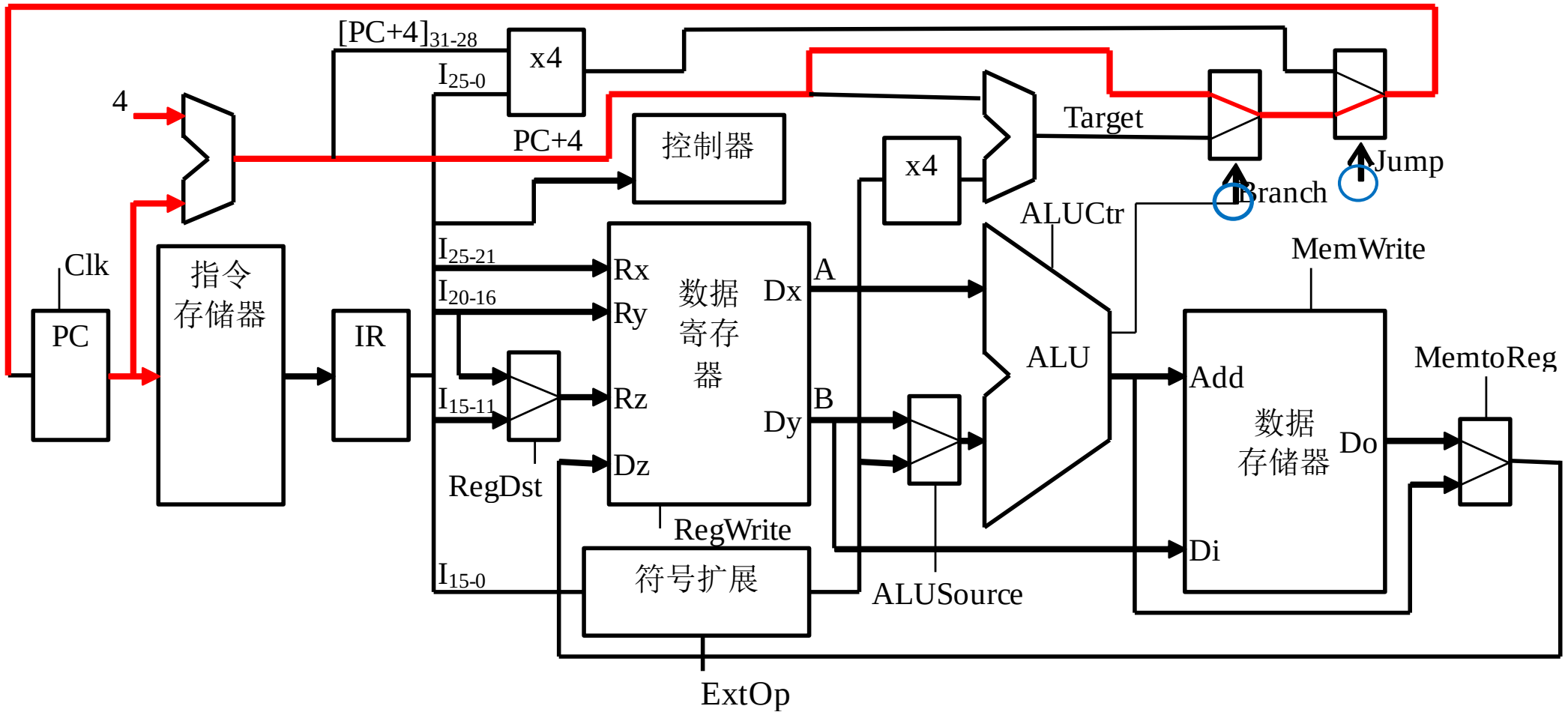
MemtoReg 写数据选择
RegWrite 数据寄存器写
RegDst 写寄存器选择

指令	数据通路	
ADD	**PC <– PC + 4**	**R[rd] <– R[rs] + R[rt];**

控制信号：



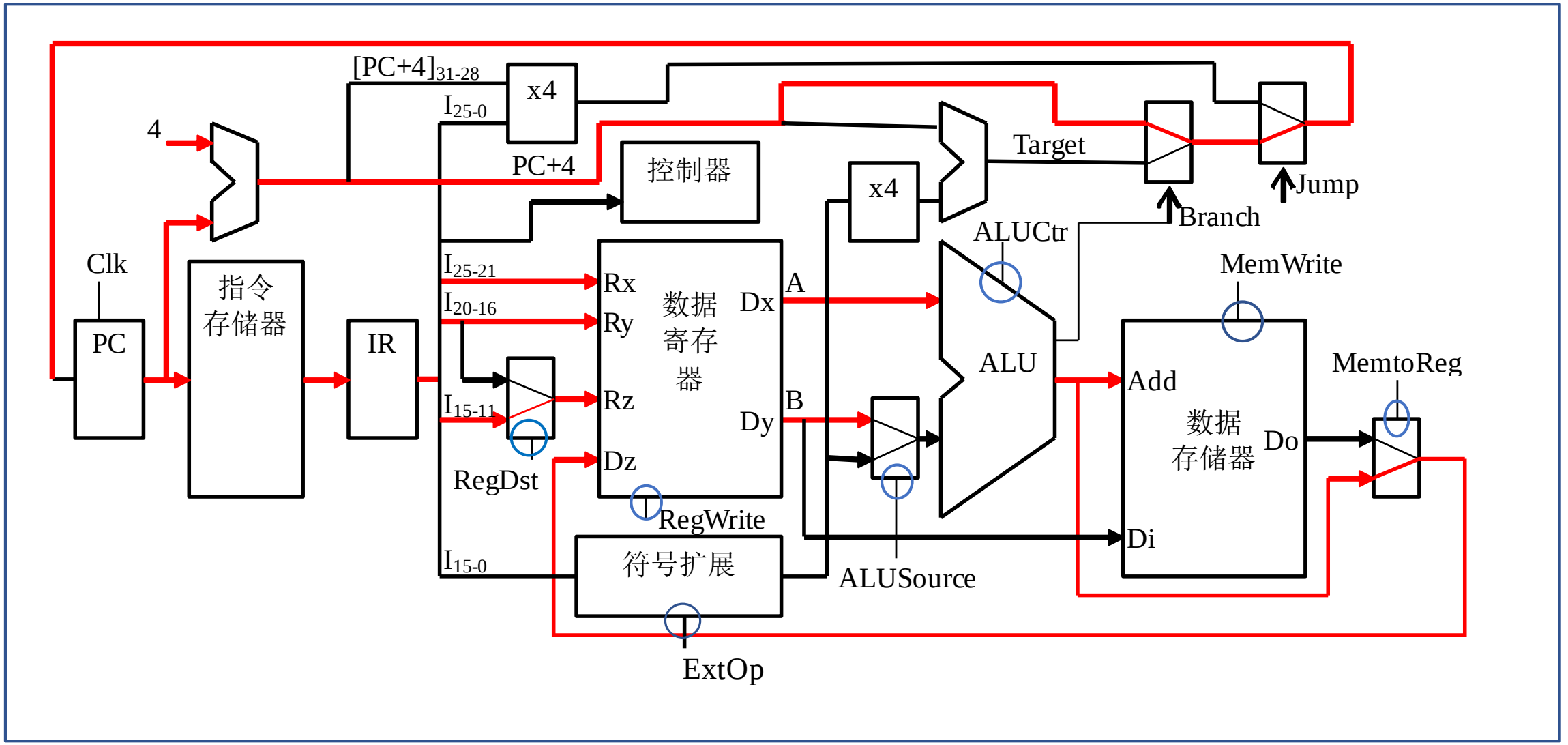
指令	数据通路	
ADD	PC <– PC + 4	R[rd] <– R[rs] + R[rt];
控制信号：	Branch = 0 , Jump=0,	ALUsrc = BusB , ALUctr = “add”, Extop=x, Memwrite=0, MemtoReg=ALU, RegDst = rd, RegWr=1



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

R[rd] <– R[rs] + R[rt];

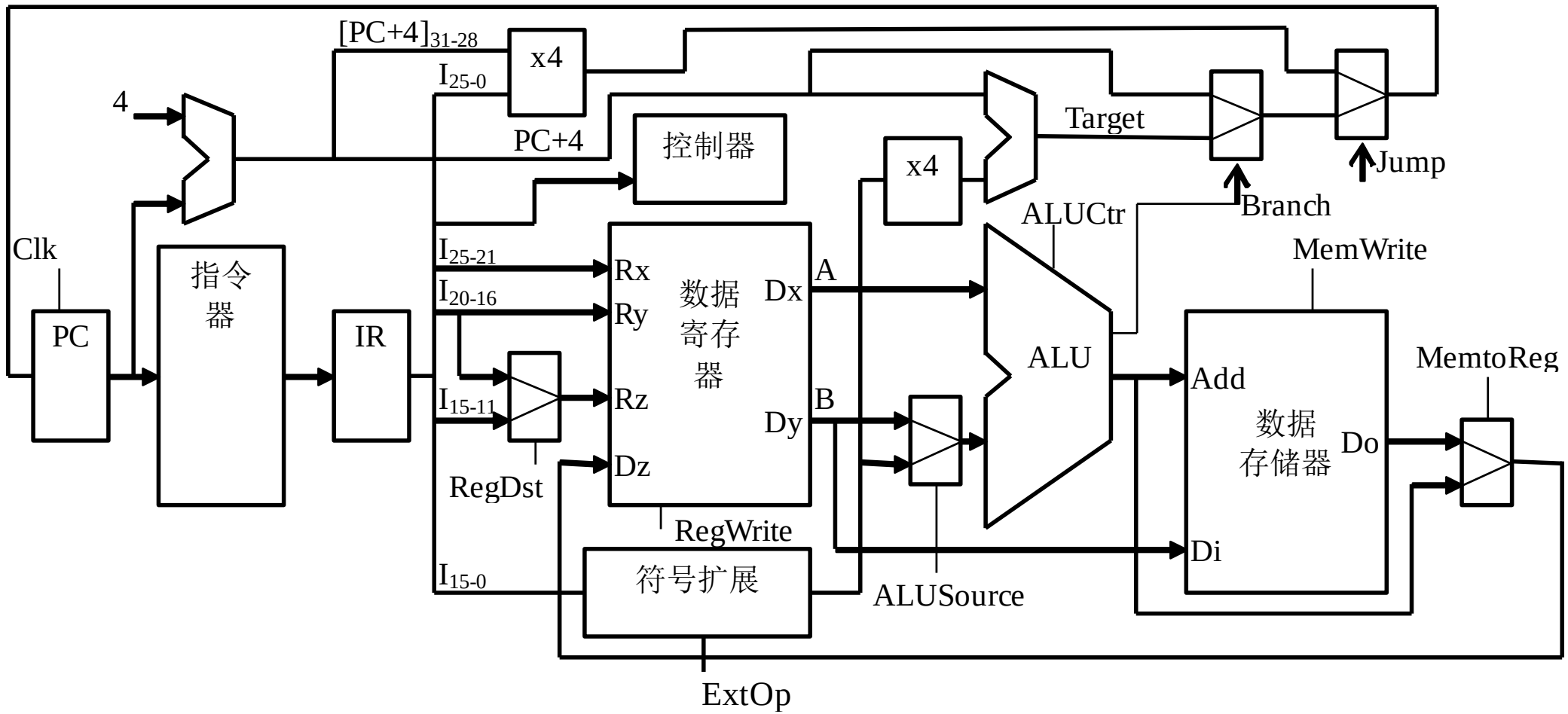
ALUsrc = BusB , ALUctr = “add”, Extop=x, Memwrite=0, MemtoReg=ALU, RegDst = rd, RegWr=1



指令　　数据通路

Ori　　PC <– PC + 4　　R[rt] <– R[rs] or unsign_ext(Imm16)];;

控制信号：



指令　　数据通路

Ori

控制信号：

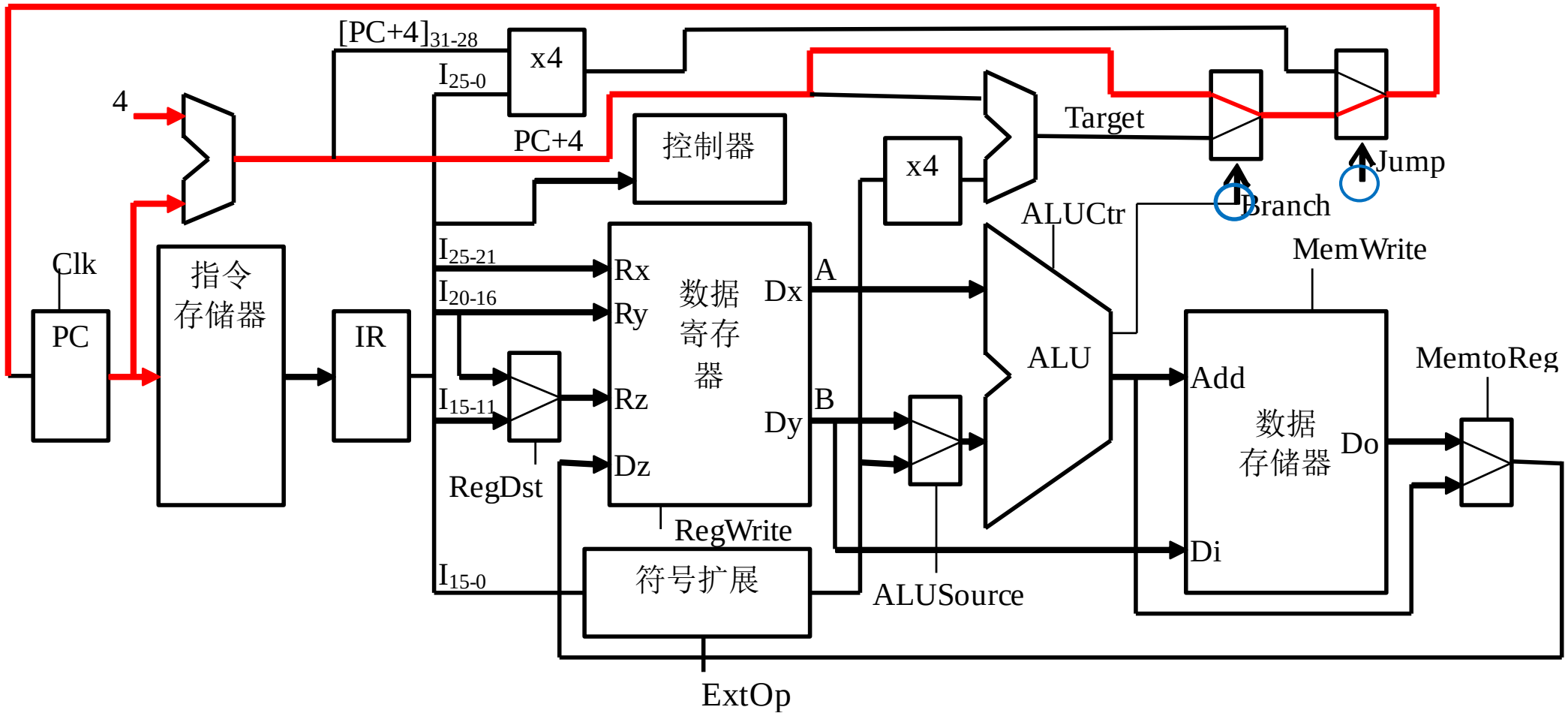
PC <– PC + 4

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

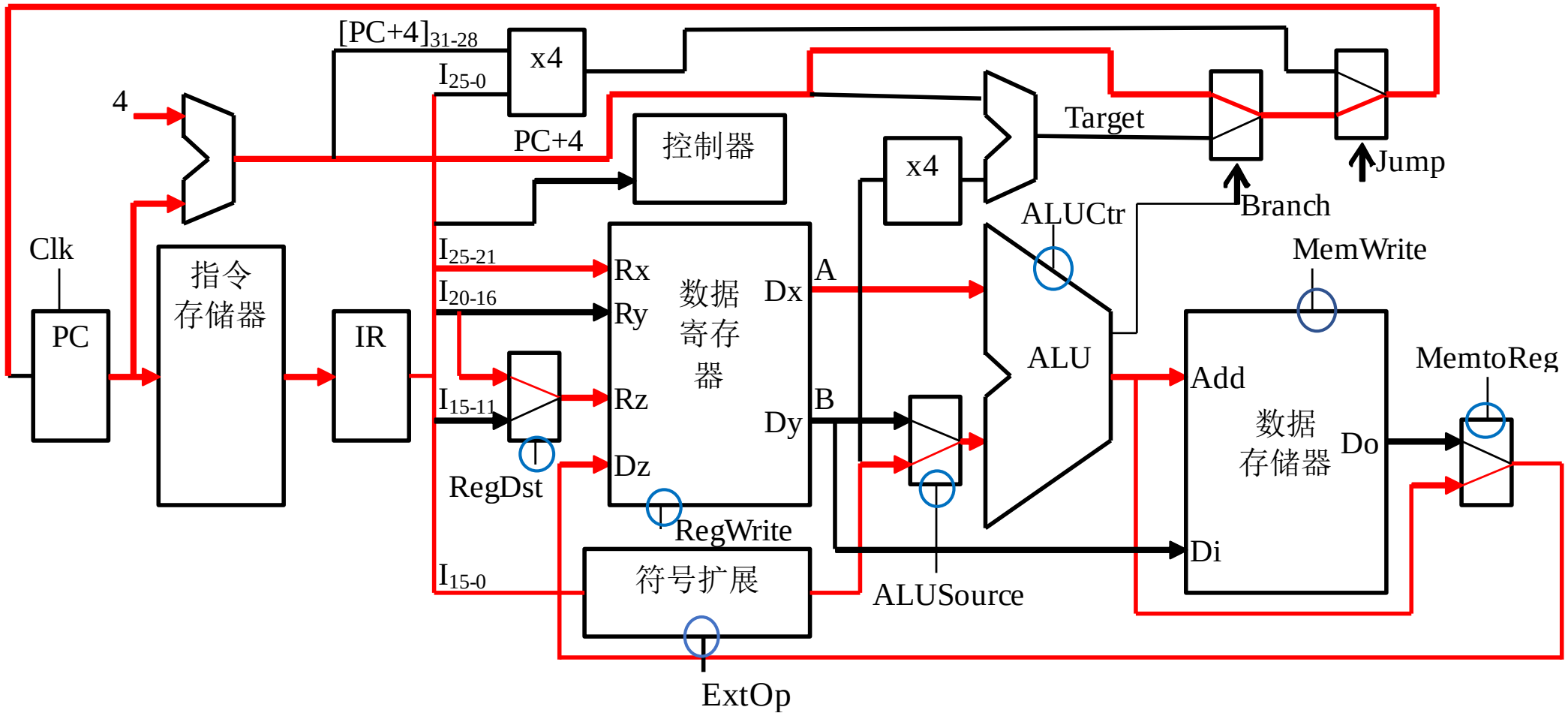


指令　　数据通路

Ori　　PC <– PC + 4

控制信号：　PC_source = 0, Jump=0,

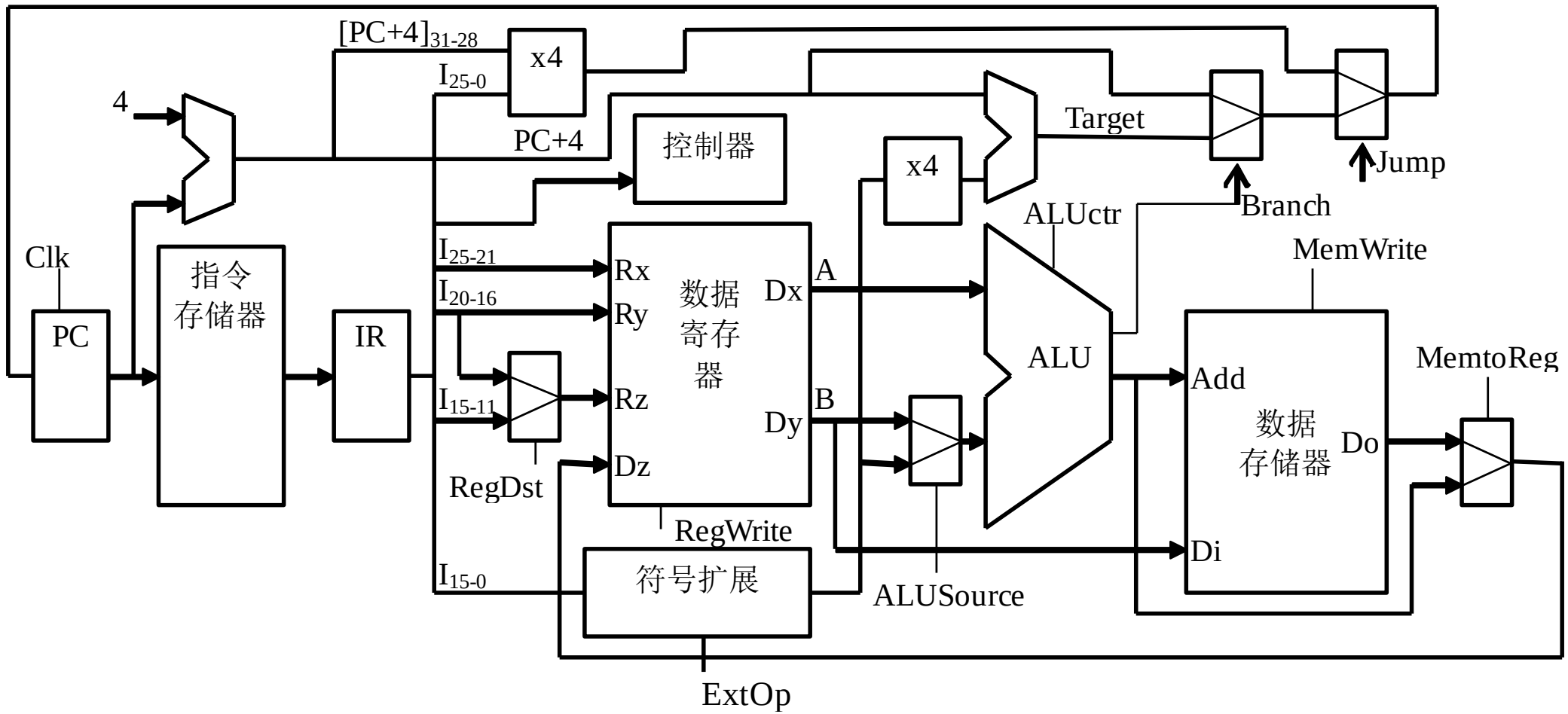
R[rt] <– R[rs] or unsign_ext(Imm16)];
ALUsrc = Im, ALUCtr = “or”, Extop = “unSn，
Memwite=0, MemtoReg=ALU, RegDst = rt, RegWr=1



指令　　数据通路

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

控制信号：



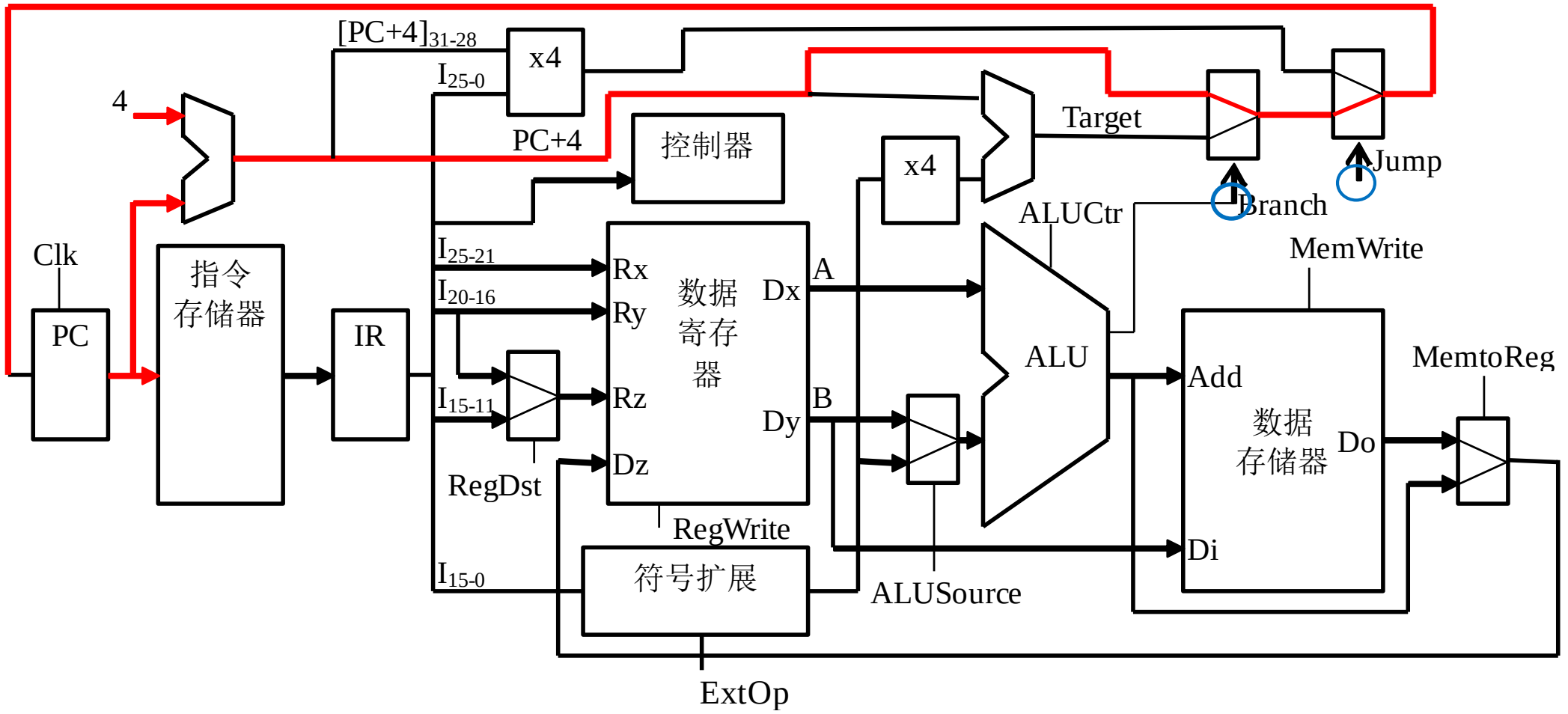
指令　　数据通路

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



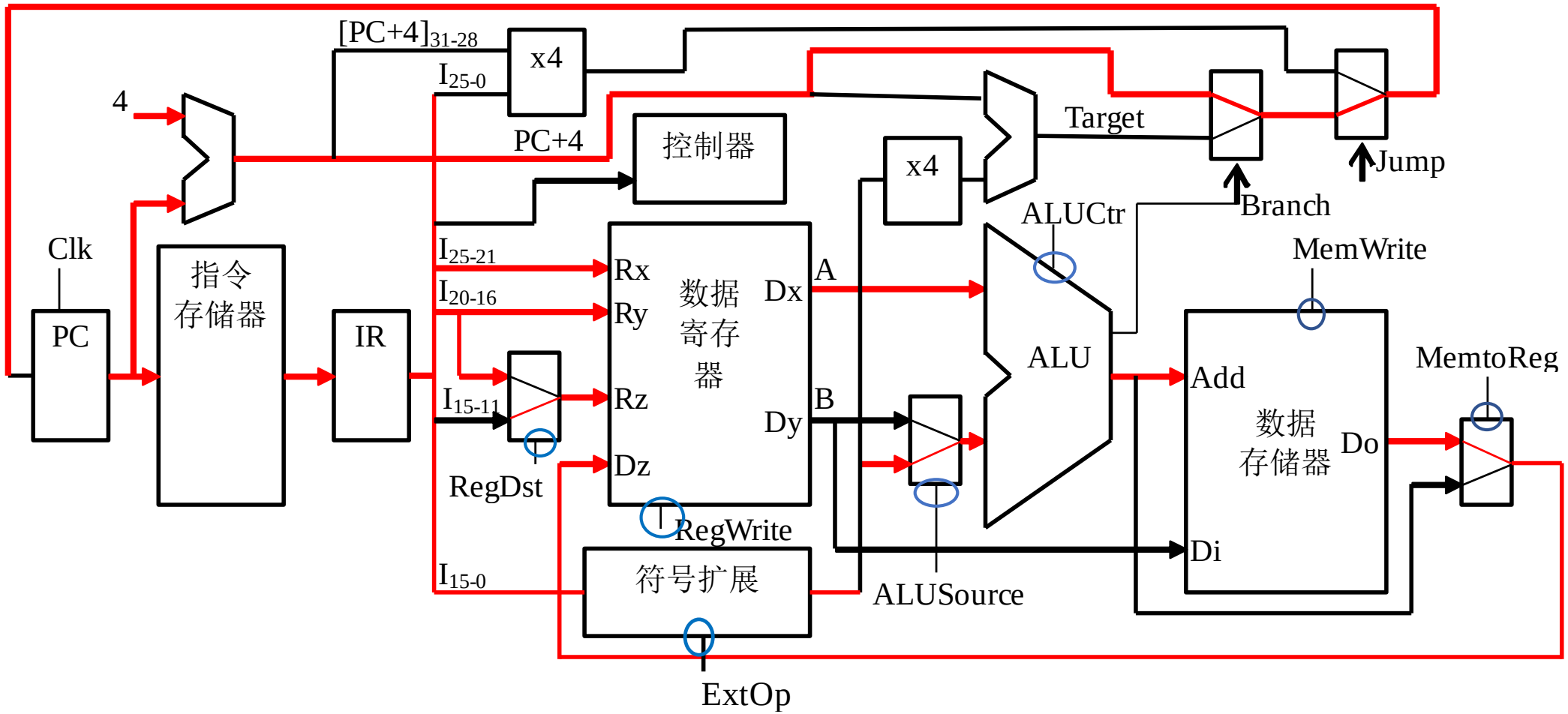
指令　　数据通路

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

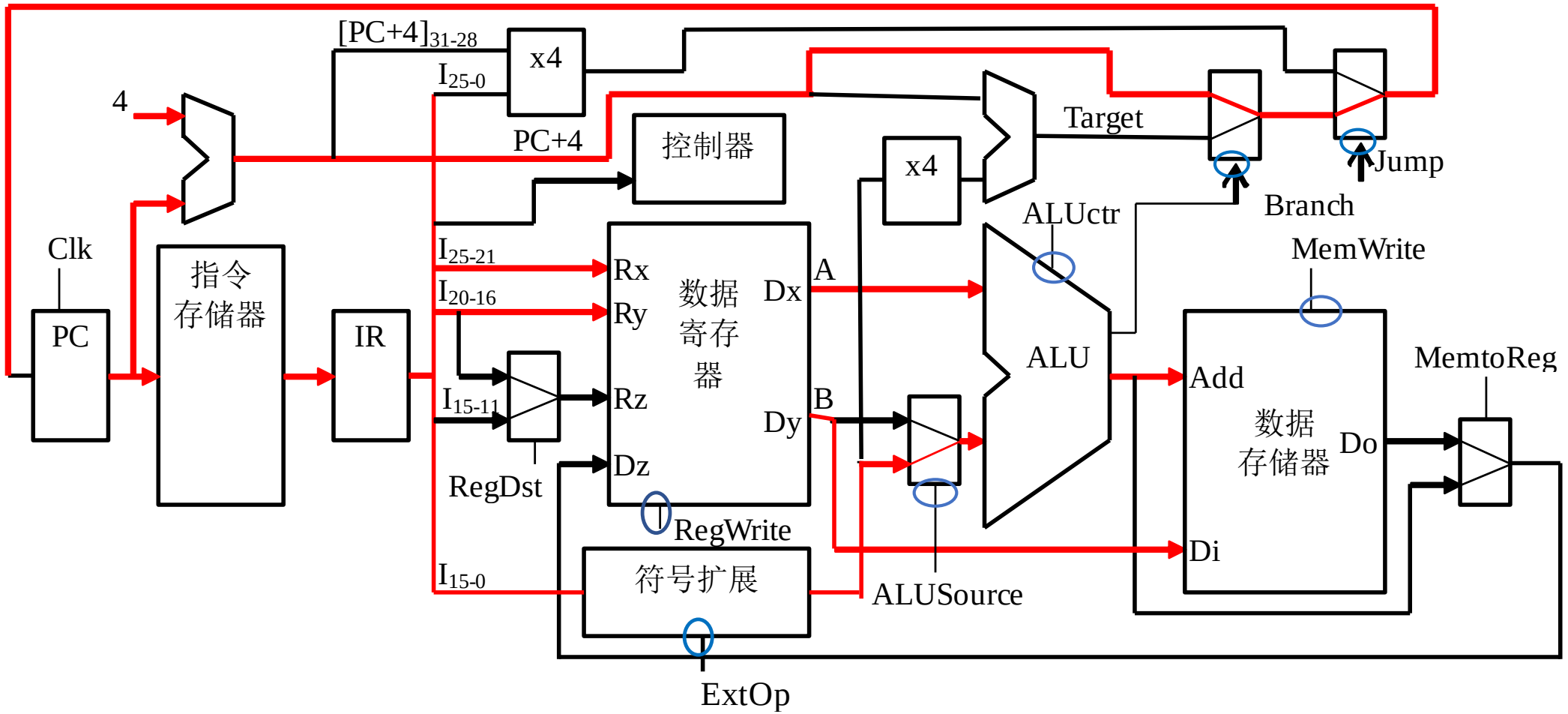


指令　　数据通路

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”,

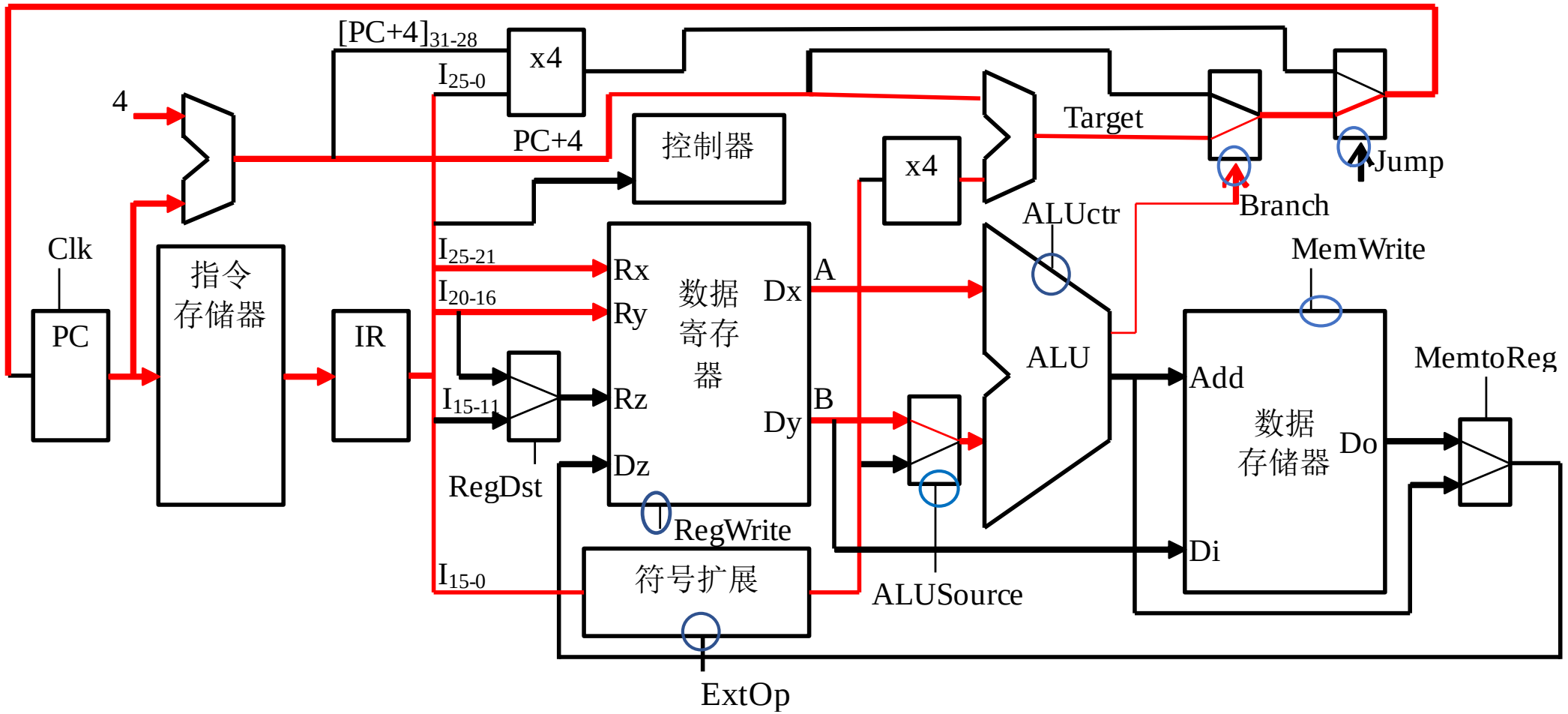
Memwrite=1, MemtoReg=x, RegDst = x, RegWr=0



指令　　数据通路

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

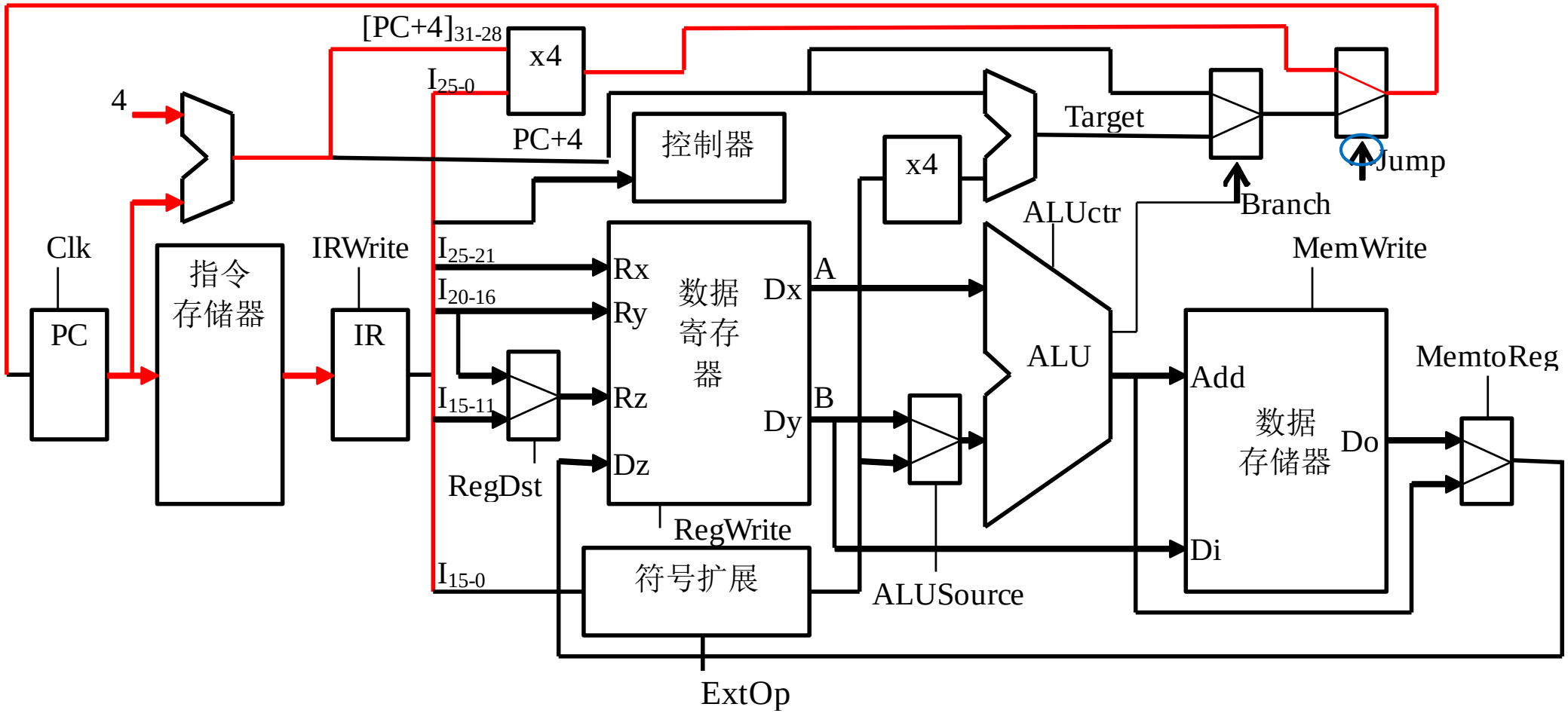
控制信号：　**ALUsrc = BusB , ALUctr = “sub” , Extop = “Sn”, Branch = “Br”, Jump=0, Memwrite=0, Regwrite=0，其余=x**



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

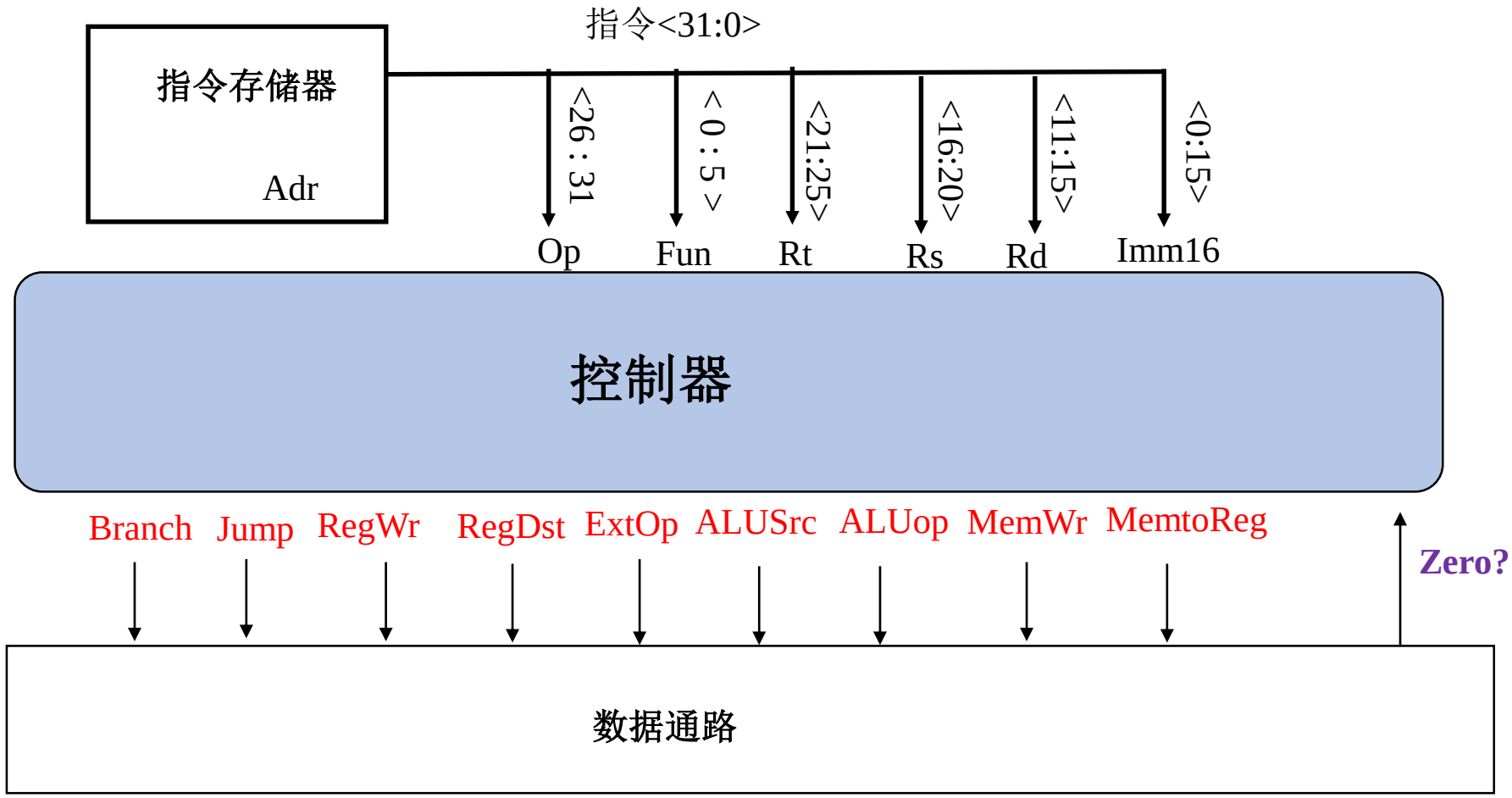
BEQ　　**if (R[rs] == R[rt]) then PC <– PC + sign_ext(Imm16)] || 00 else PC <– PC + 4**

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

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