《计算机系统结构》课程直播

2020. 4.7

请将ZOOM名称改为“姓名”；

听不到声音请及时调试声音设备；签到将在课结束后继续

讲课内容

1 单周期处理器的设计

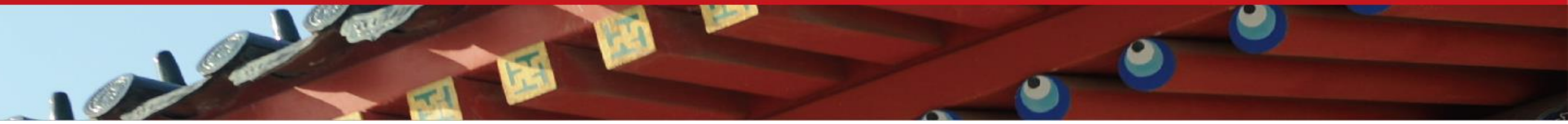
课堂讨论

- 解释为什么汇编程序在对下列汇编源程序中的beq指令进行汇编时会遇到问题，应该如何修改该程序段?

```
here:   beq  $s0,  $s2,  there
              ......
there:     addi  $s1,  $s0,  4
```

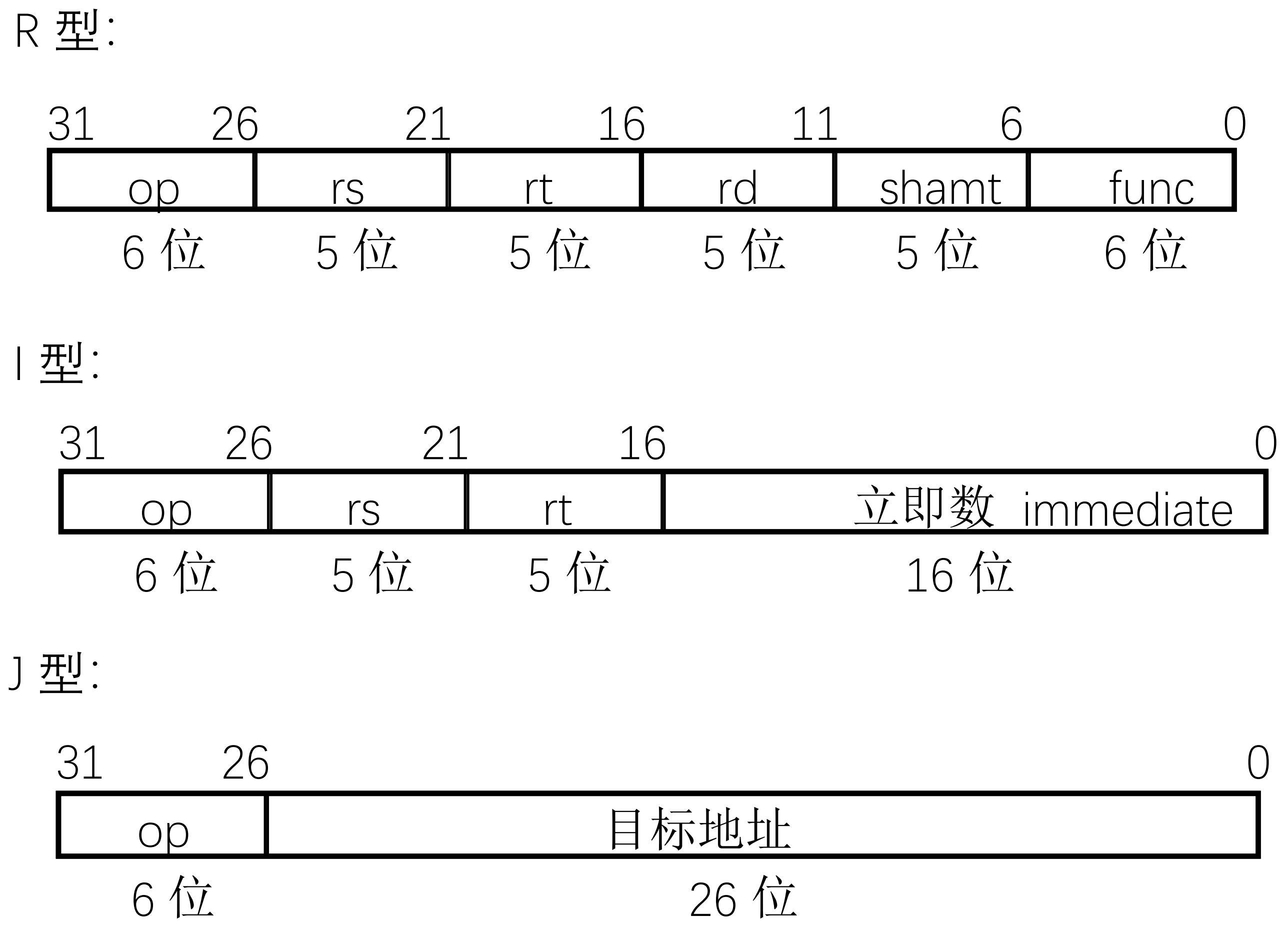
处理器设计

- 能实现几条简单指令的处理器
  - 算术逻辑运算指令：add, sub, ori
  - 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

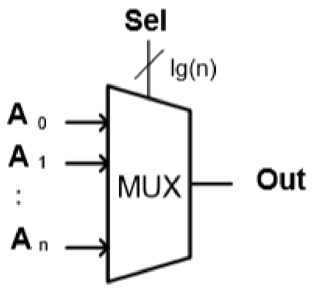


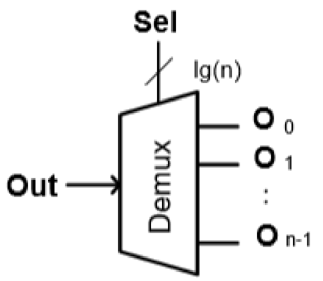
设计处理器的五个步骤

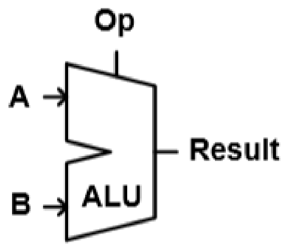
1. 分析指令系统， 得出对数据通路的需求
2. 选择数据通路上合适的组件
3. 连接组件构成数据通路
4. 分析每一条指令的实现，以确定控制信号
5. 集成控制信号，完成控制逻辑

Hardware Elements

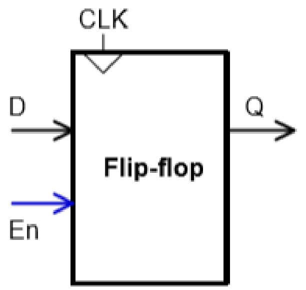
- Combinational logic:
  - Output is a function of the present input only：Out





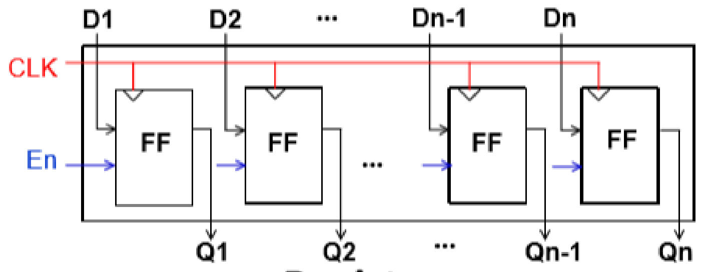


- Synchronous sequential logic
  - Edge-triggered: data is sampled at the clock edge



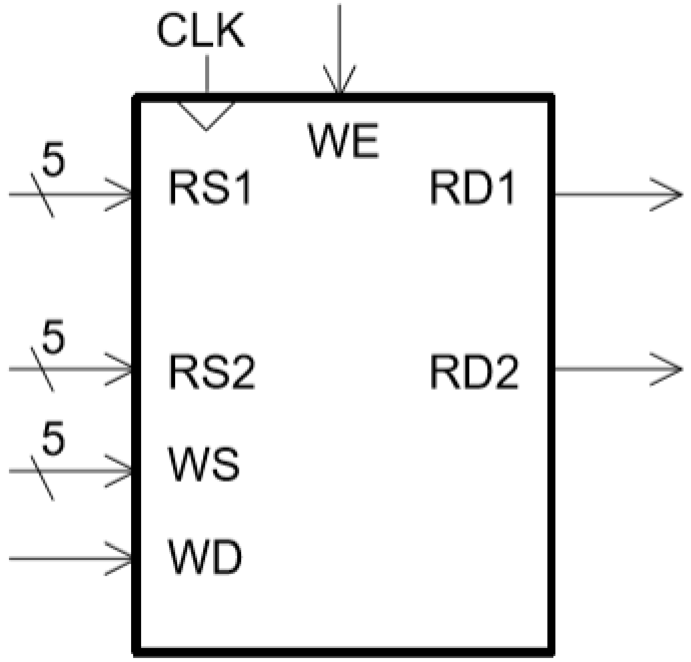
D	CLK	Q
0	Rising	0
1	Rising	1
-	0	Last Q
-	1	Last Q

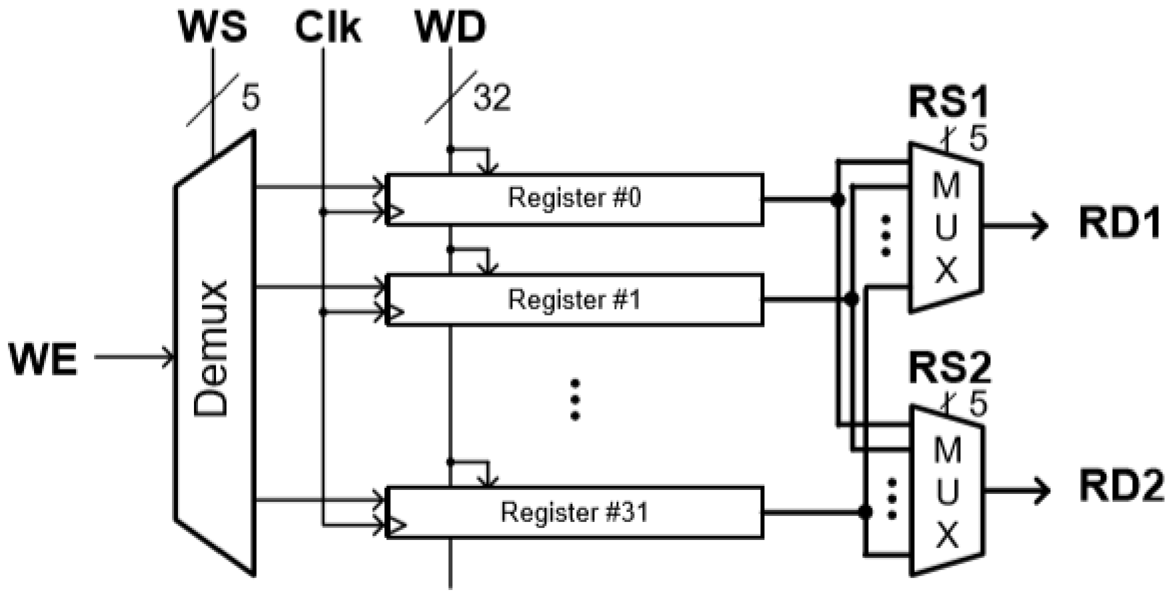
Latch



Register

Register Files

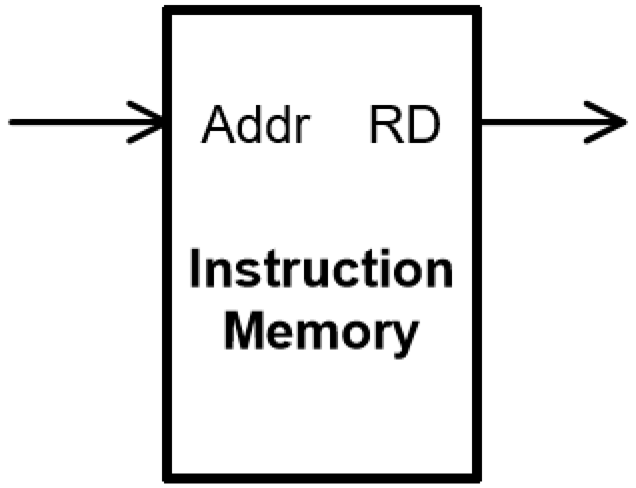


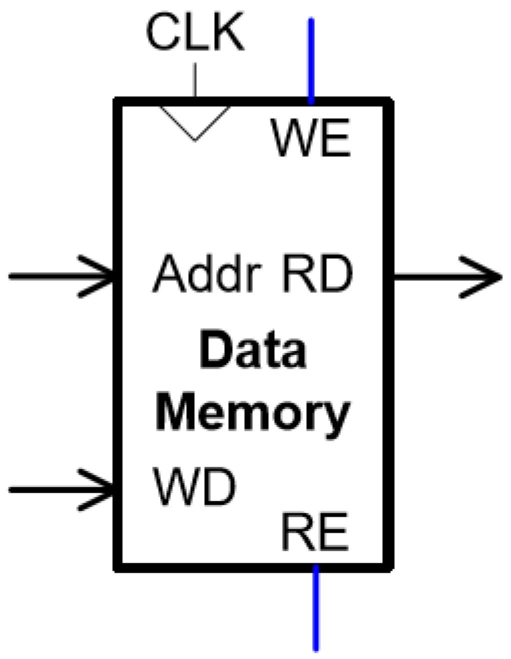


- The processor's general-purpose registers are stored in a structure called a register file

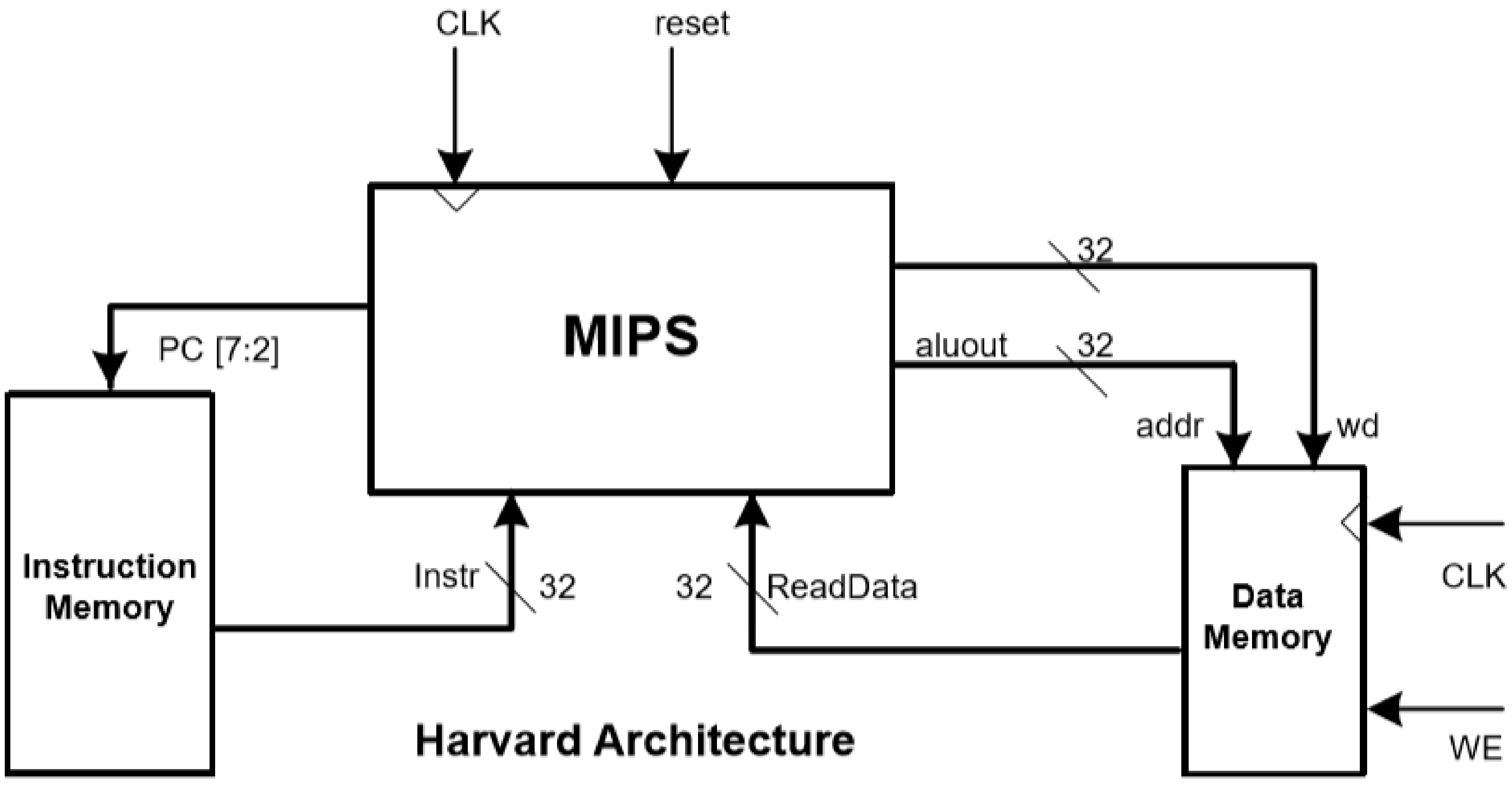
A Simple Memory Model

- a Read can be done any time (i.e. combinational logic)
- a Write is performed at the rising clock edge



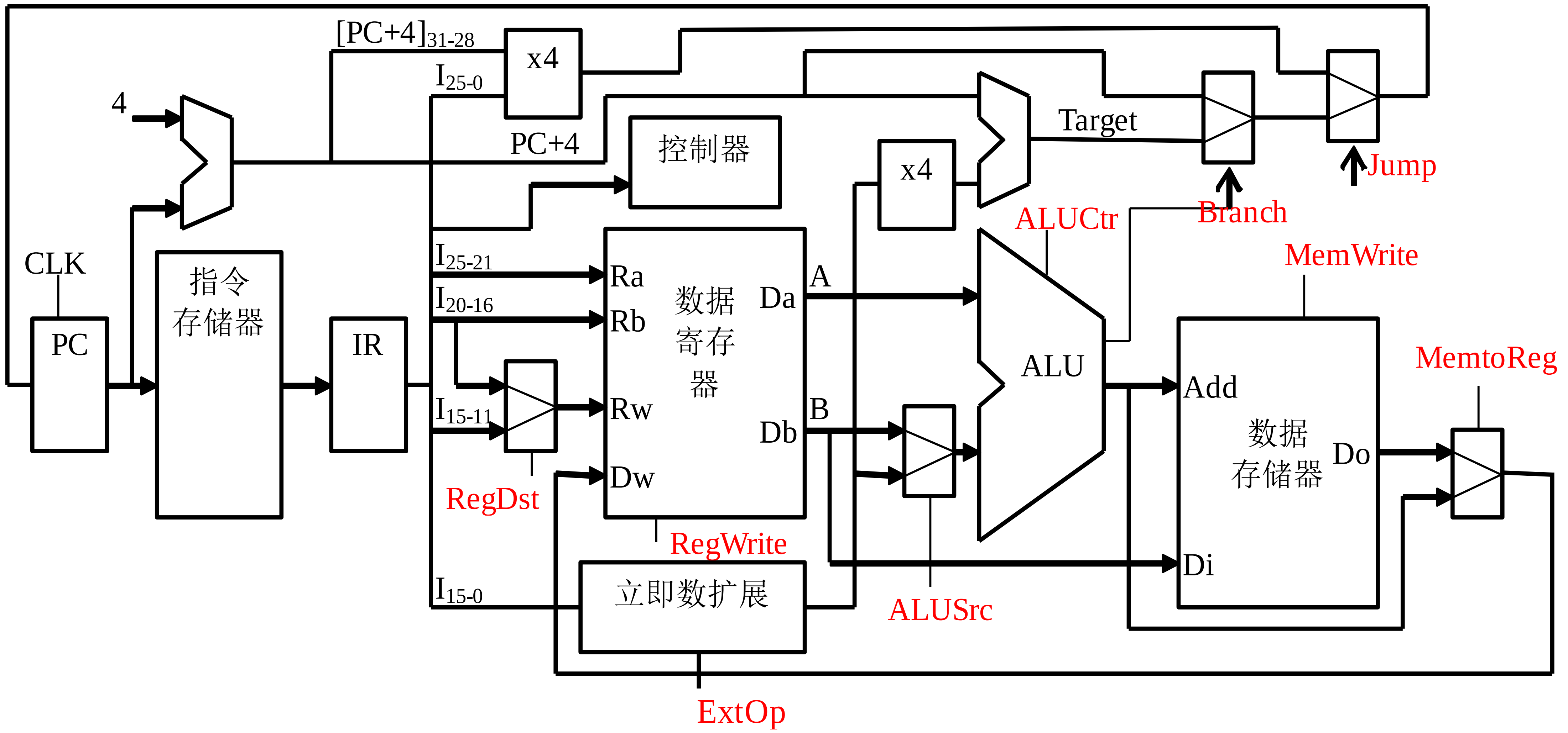


Harvard Architecture

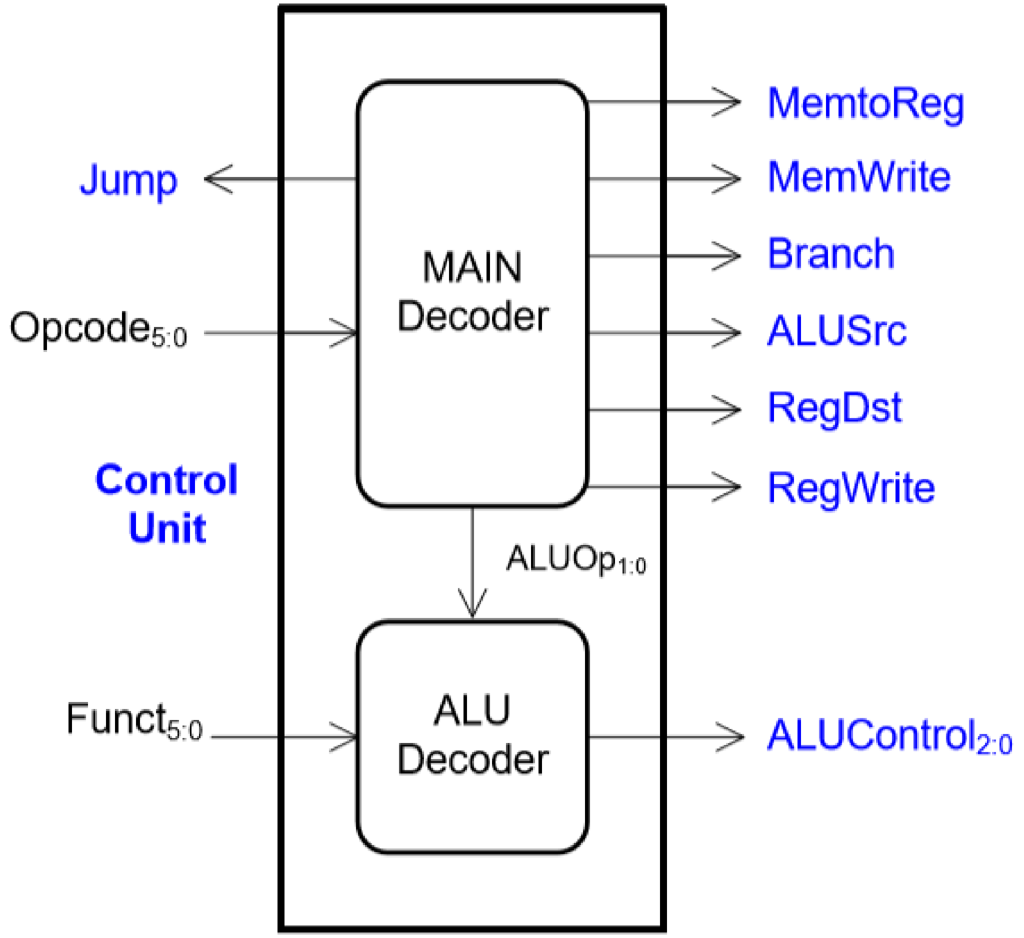


- Harvard Style: Physically separate storage and signal pathways for instructions and data
- Princeton Style: The same pathways (von Neumann Model)

完整的数据通路


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

Control Logic

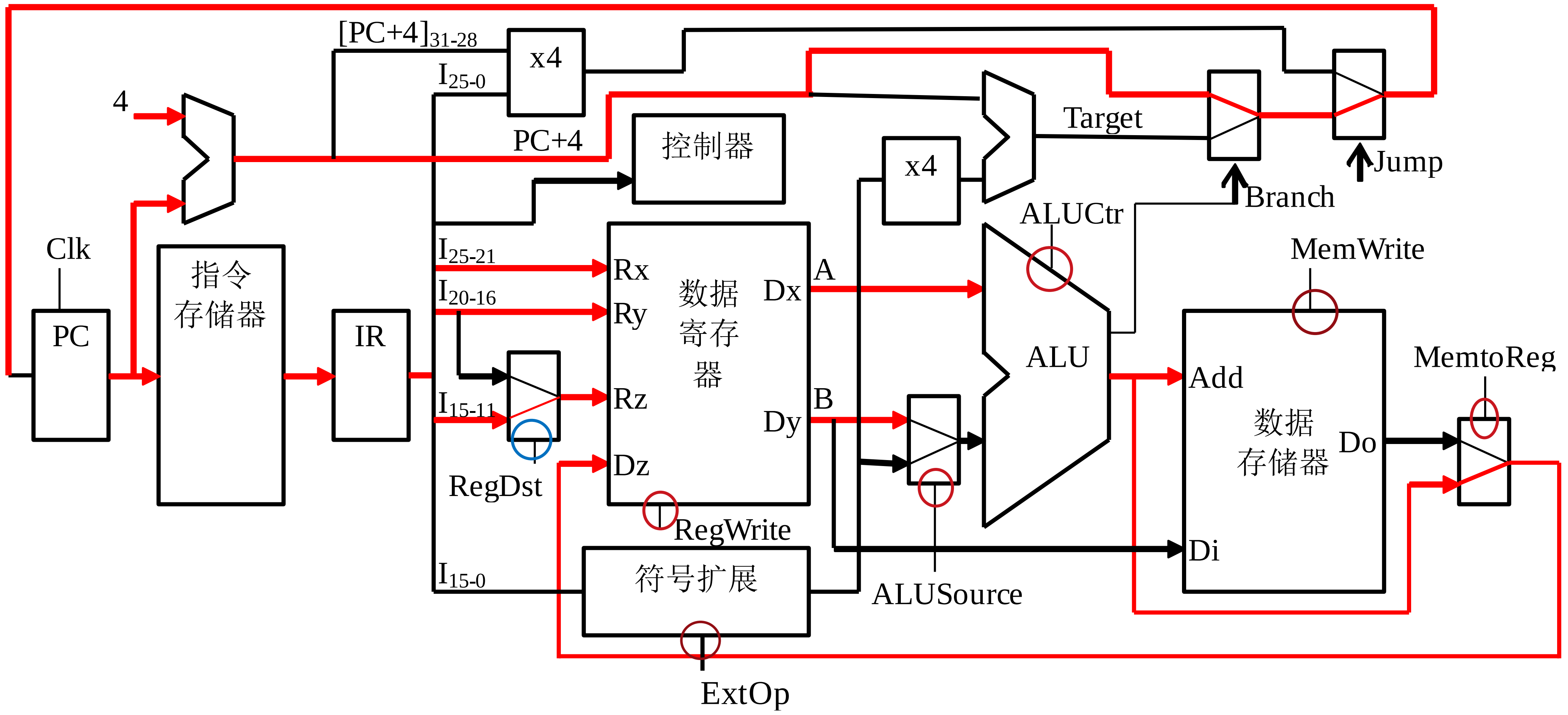


- The setting of the control lines is completely determined by the opcode fields of the instruction
- Focus on the design of the state machines to decode instructions and generate the sequence of control signals

指令　　　数据通路

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

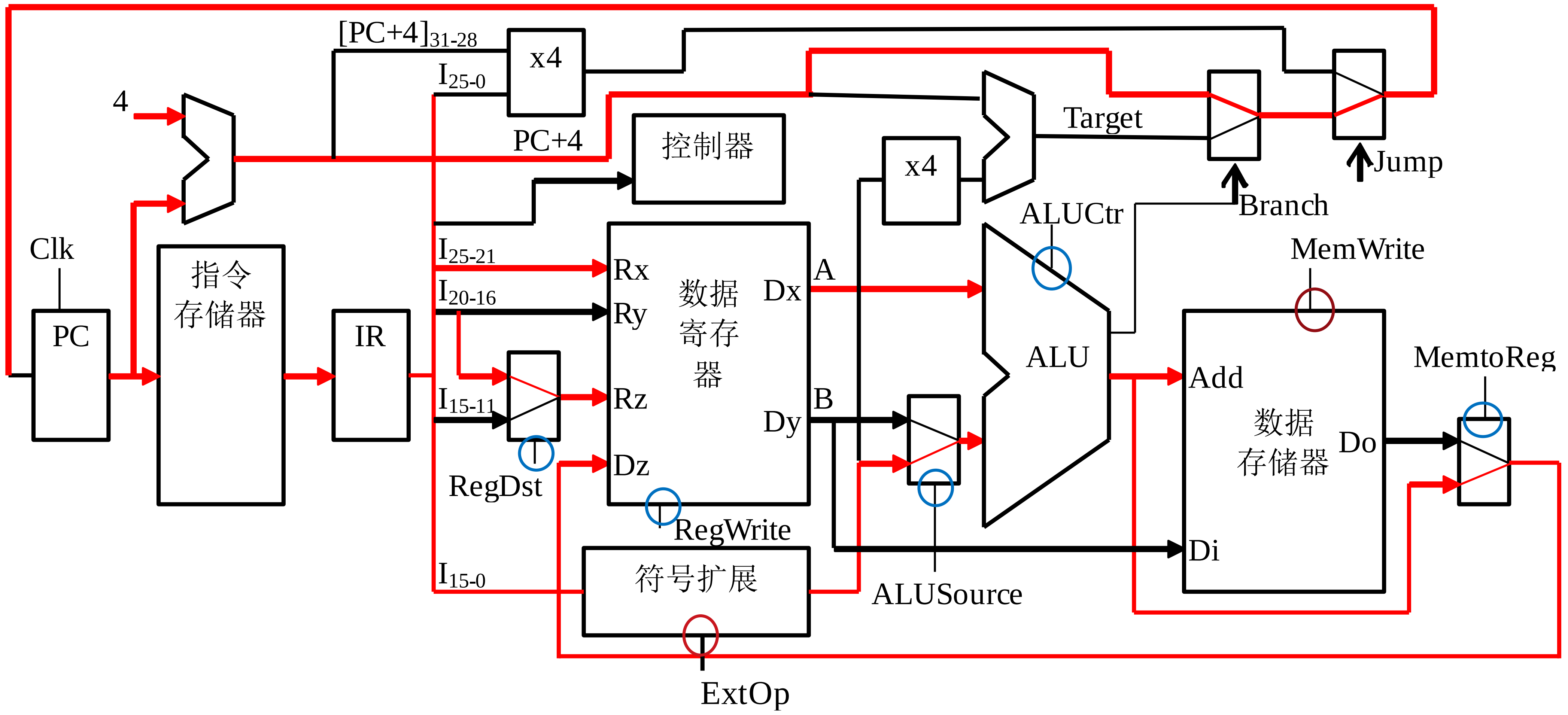
控制信号：Branch = 0 , Jump=0,　　ALUsrc = BusB , ALUctr = “add”, Extop=x, Memwrite=0, MemtoReg=ALU, RegDst = rd, RegWr=1



指令 数据通路

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

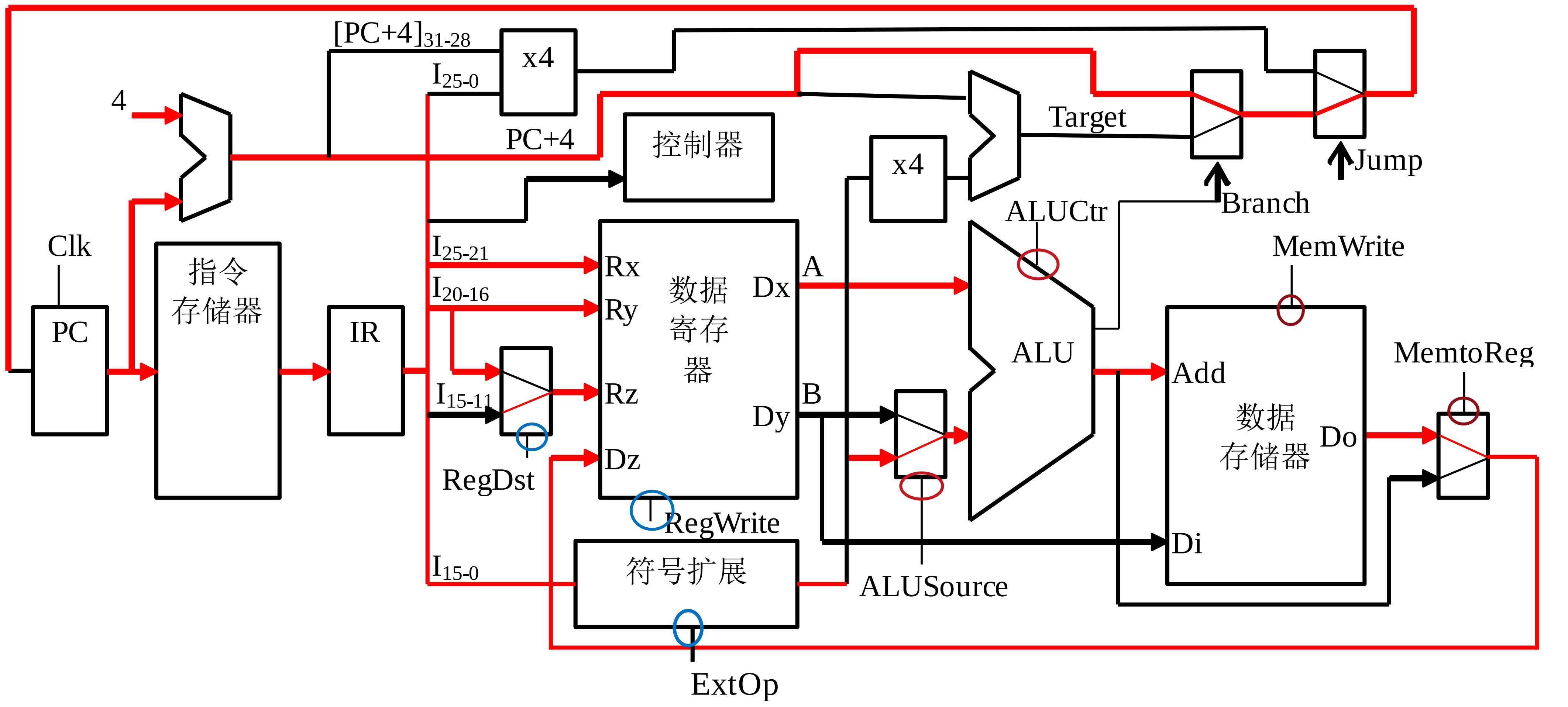
控制信号： **PC_source = 0, Jump=0,** **ALUsrc = Im, ALUCtr = “or”, Extop = “unSn，**
Memwite=0, MemtoReg=ALU, RegDst = rt, RegWr=1



指令　　　数据通路

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

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

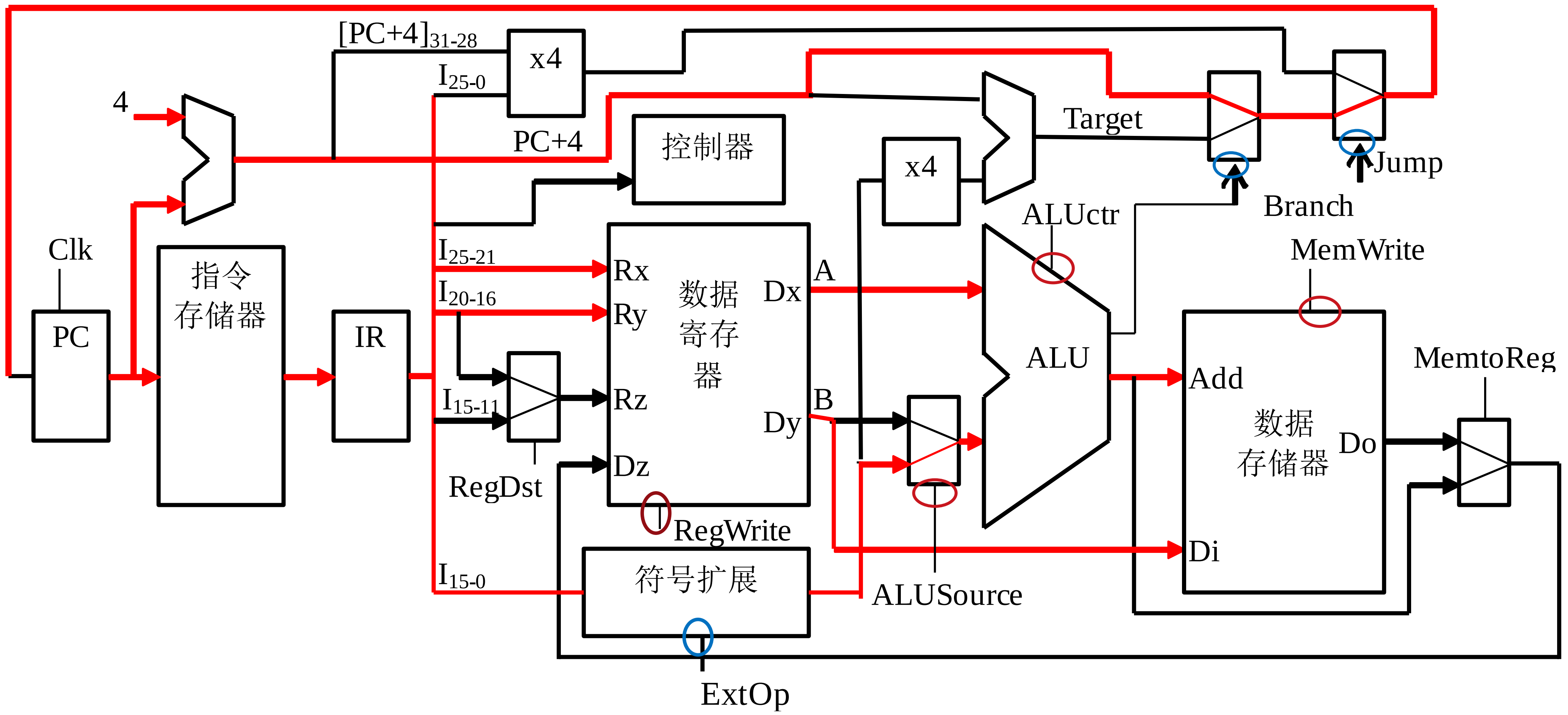


指令　　　数据通路

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”,

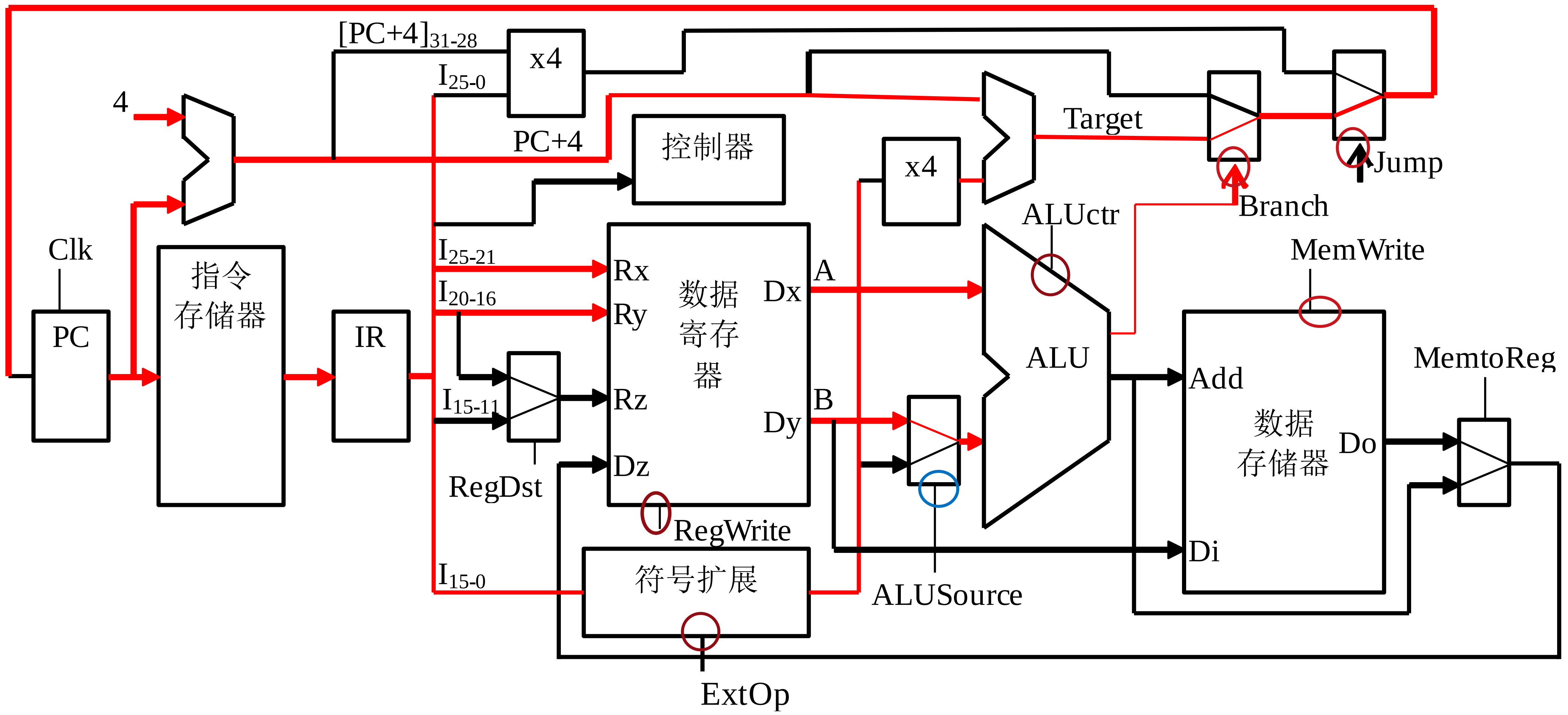
Memwrite=1, MemtoReg=x, RegDst = x, RegWr=0



指令 数据通路

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

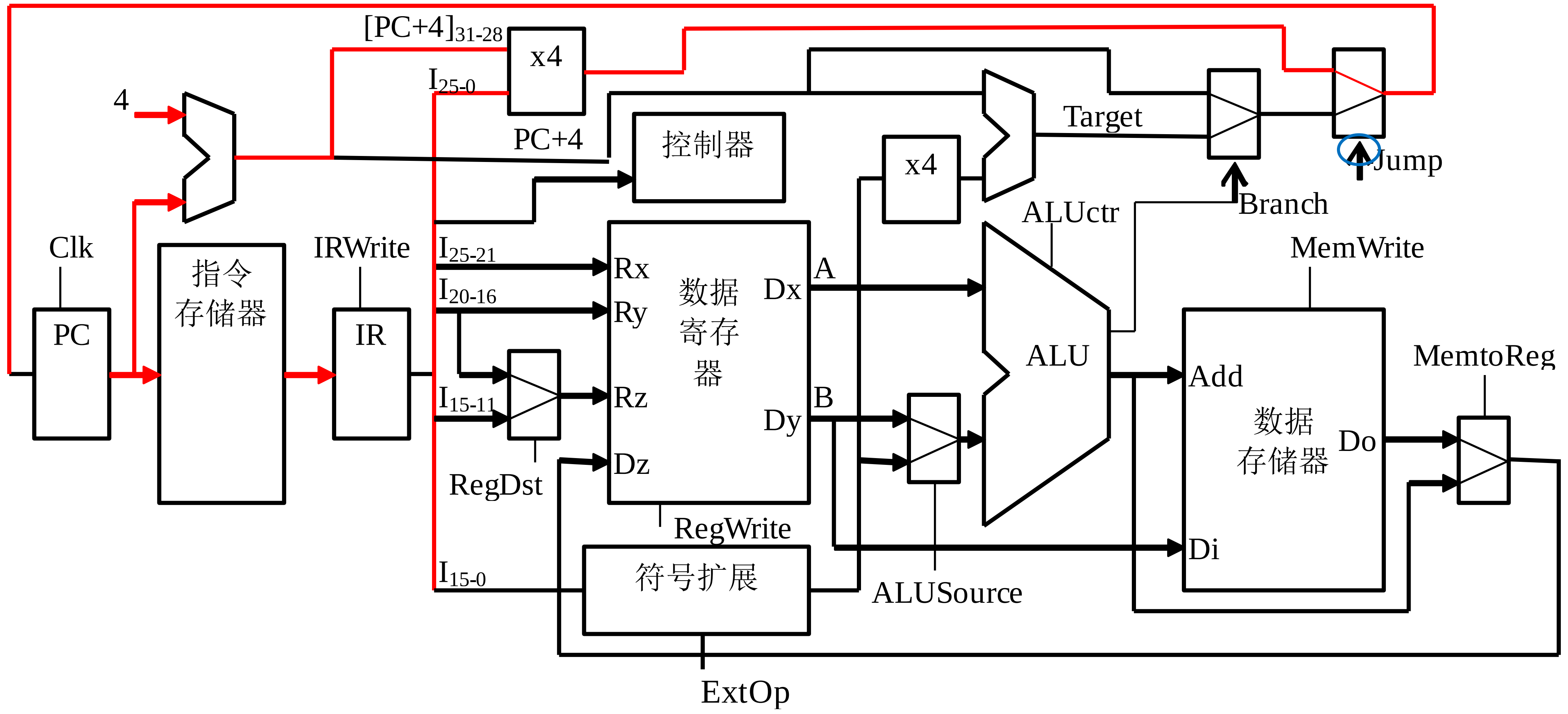
控制信号： ALUsrc = BusB , ALUctr = “sub” , Extop = “Sn”, Branch = “Br”, Jump=0, Memwrite=0, Regwrite=0， 其余=x



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

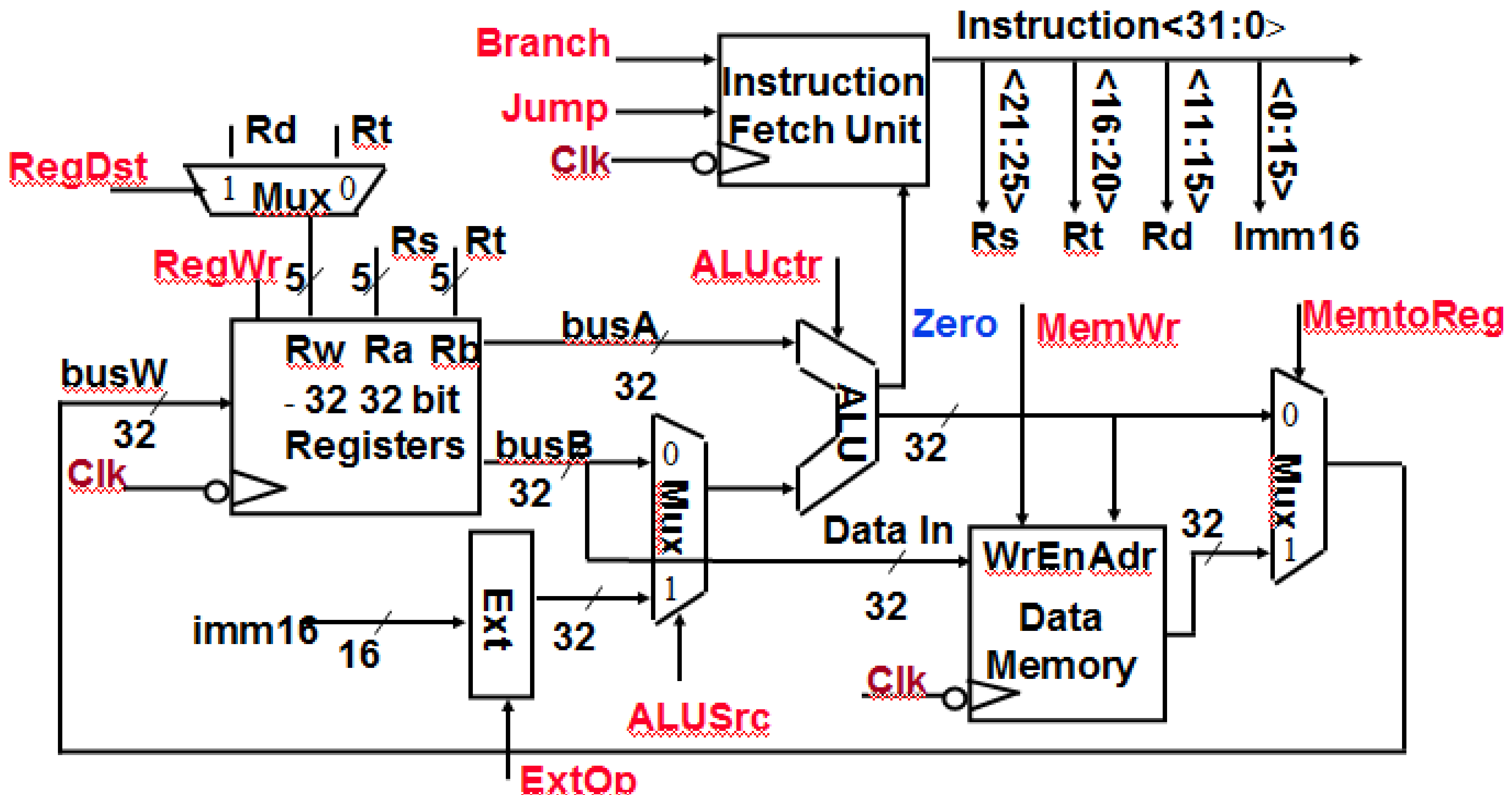
ALUsrc = BusB ,　Extop = “Sn”,　ALUctr = “sub” , Branch = “Br”, Jump=0,
Memwrite=0, Regwrite=0，　MemtoReg=x, RegDst = x,

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

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

课堂练习

- 假定图中单周期数据通路对应的控制逻辑发生错误，使得在任何情况下控制信号RegWr、RegDst、Branch、MemWr、ExtOp 总是为0，则哪些指令不能正确执行？为什么？



下一节

- 微程序控制器 vs 硬连线控制器

- 请预习到 5.5

再见