处理器的数据通路

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**

连接组件构成数据通路

（1）取指令

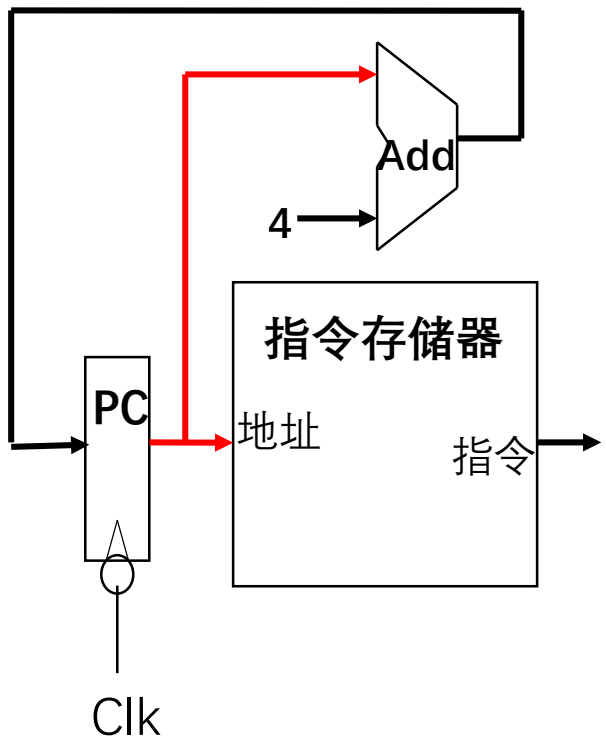
读指令存储器是一个组合电路实现

取指令步骤：

- 从指令存储器读指令
- 将PC值更新为顺序执行的下一条指令的地址
  - PC ← PC + 4

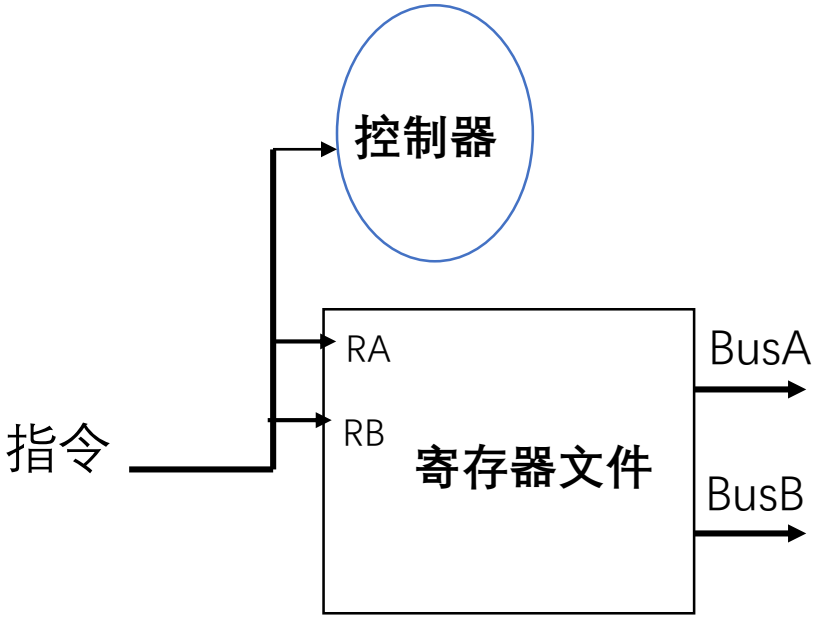
PC的状态更新是时序电路：

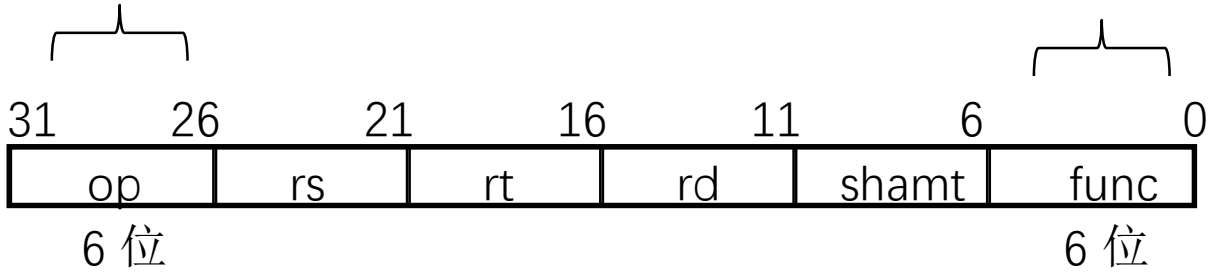
- 需要CLK边沿控制
- 每个时钟周期更新一次，



连接组件构成数据通路

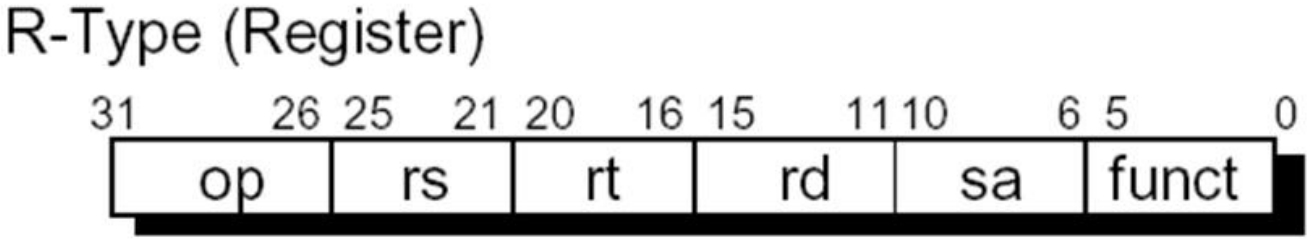
- （2）指令译码
  - 将指令的操作码（op)和功能(func)码作为控制器的输入端
  - 读寄存器
    - 指令中的寄存器地址连接到RA和RB
    - 从寄存器文件读，输出到BusA和BusB

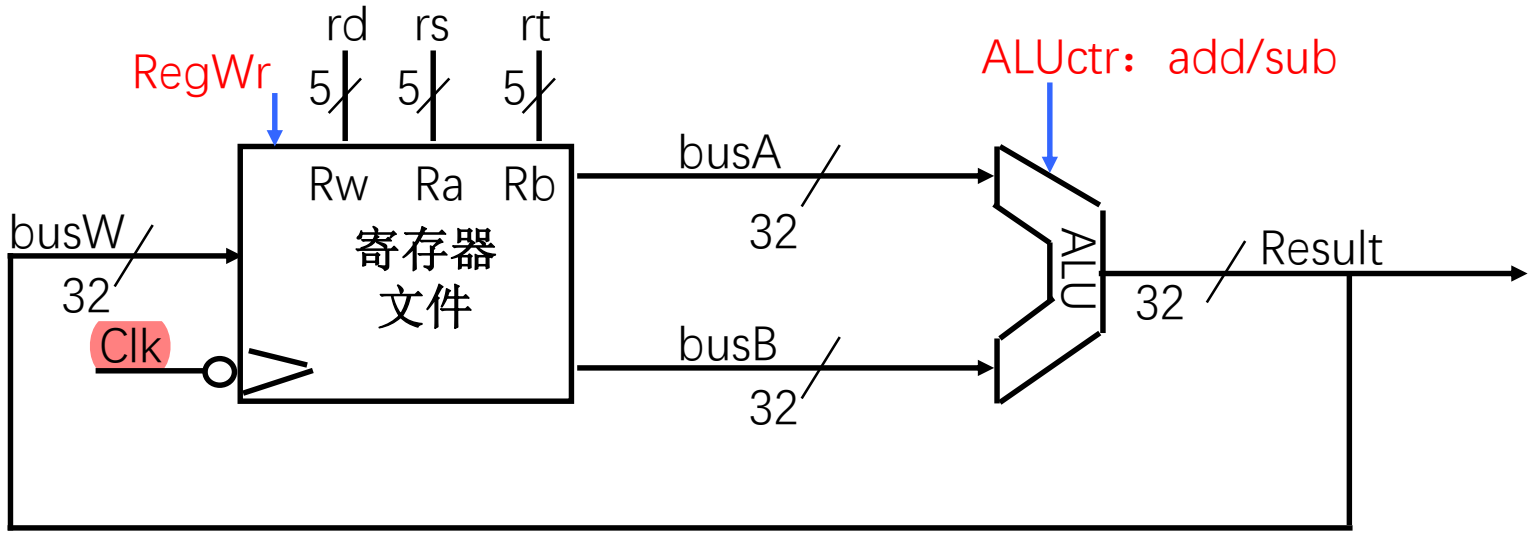




R型指令的数据通路

- add rd, rs, rt
- R[rd] ← R[rs] + R[rt]



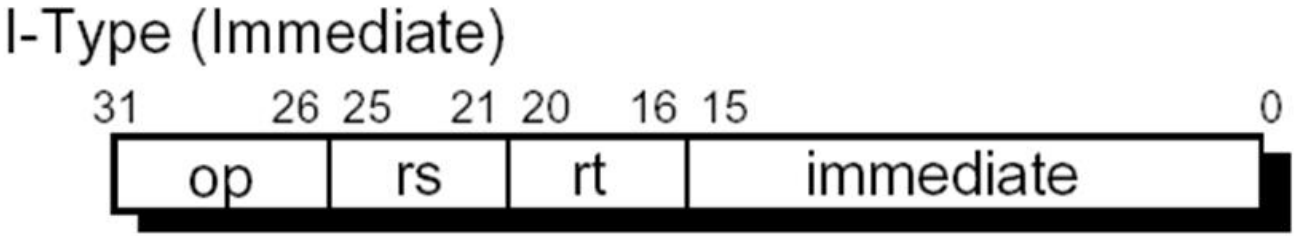


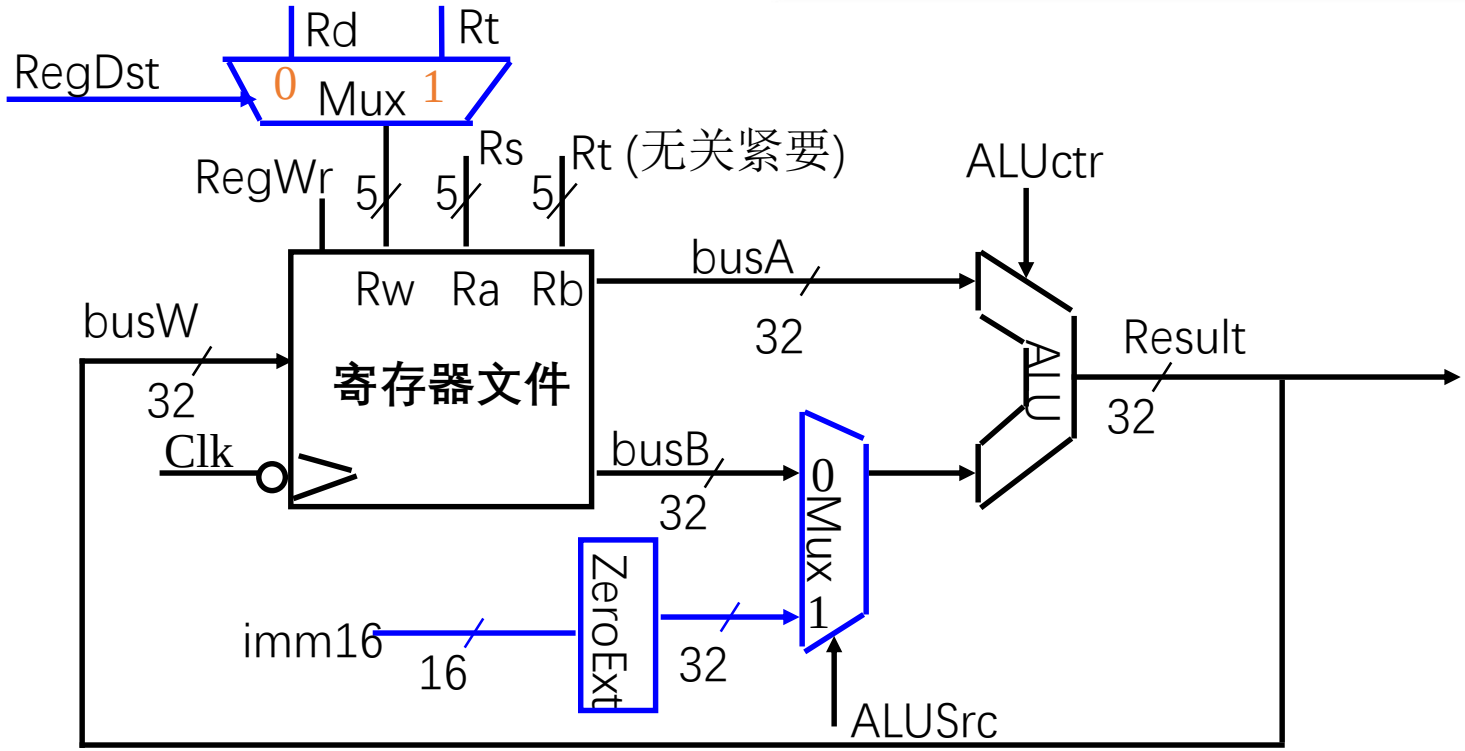
Ra, Rb, Rw 对应 rs, rt, rd

控制信号：ALUctr=add，RegWr=1

Ori 数据通路

- ori　rt, rs, imm16
- R[rt] ← R[rs] or ZeroExt[imm16]]

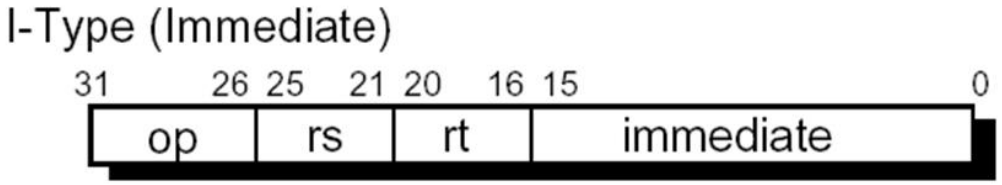


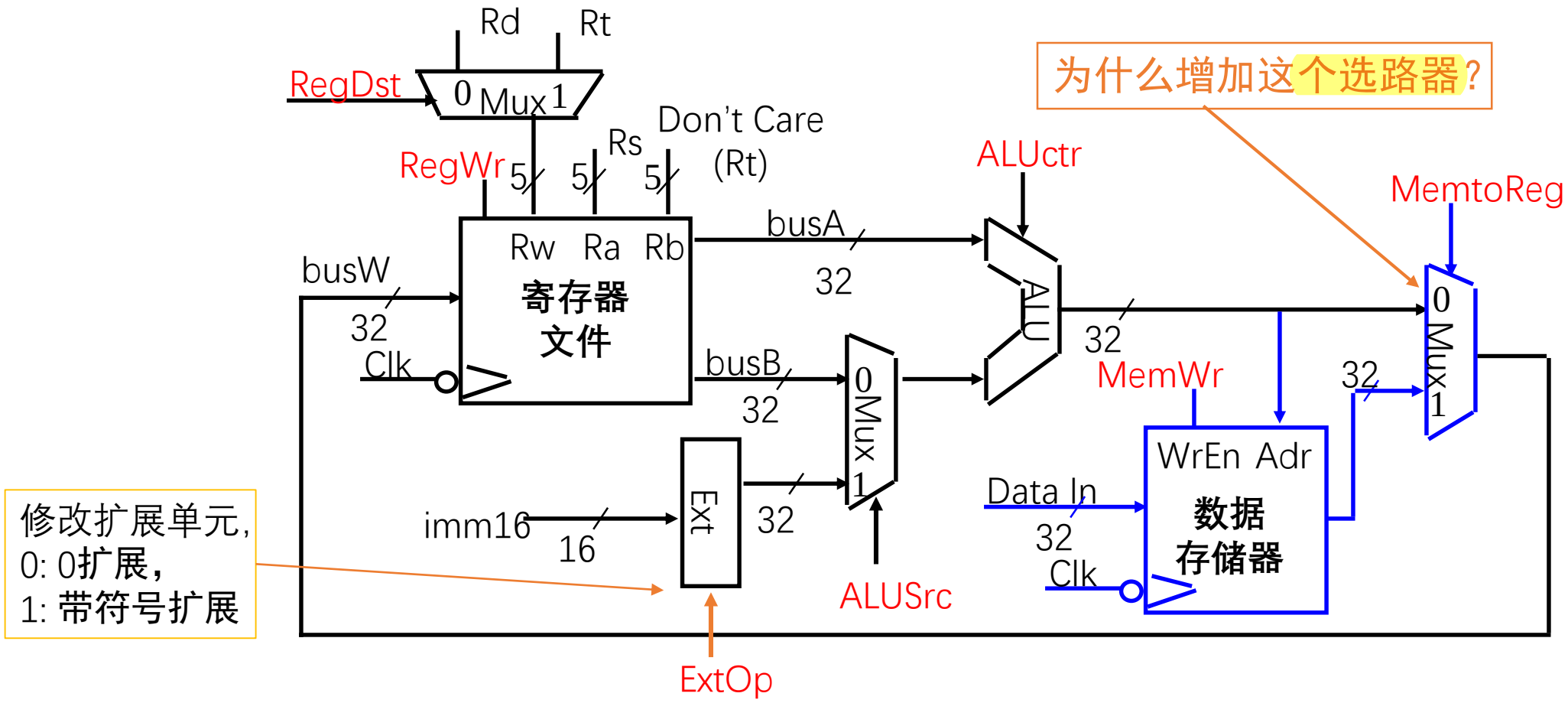


控制信号：ALUSrc=1; ALUctr=or ; RegWr=1 ; RegDst=1

Lw 数据通路

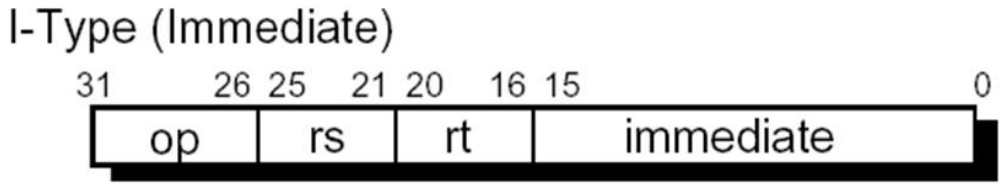
- lw rt, rs, imm16 R[rt] ← M[R[rs] + SignExt[imm16]]



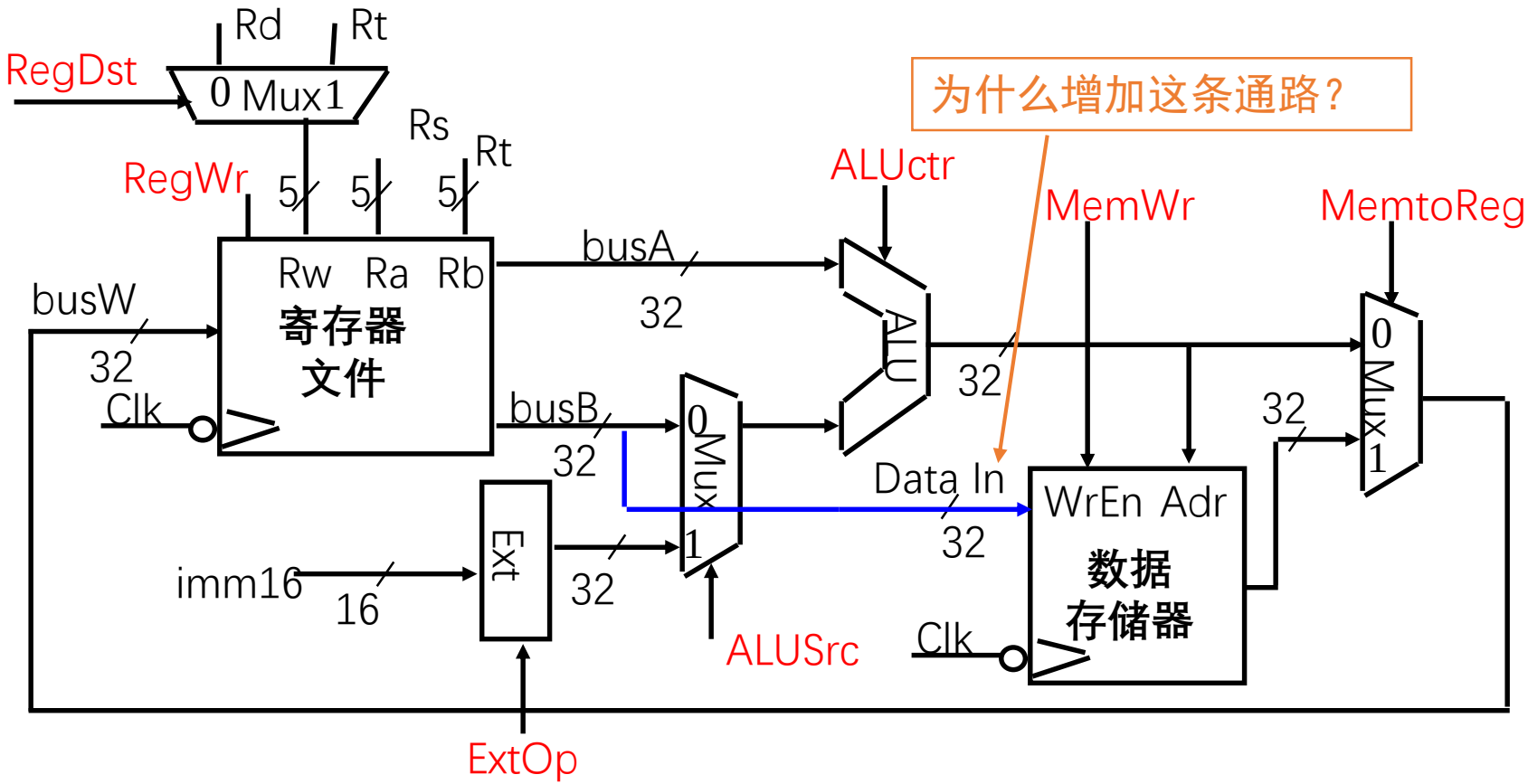


ALUctr=add, ALUSrc=1, ExtOp=1, , MemWr=0, RegDst=1, RegWr=1, MemtoReg=1

Sw 数据通路



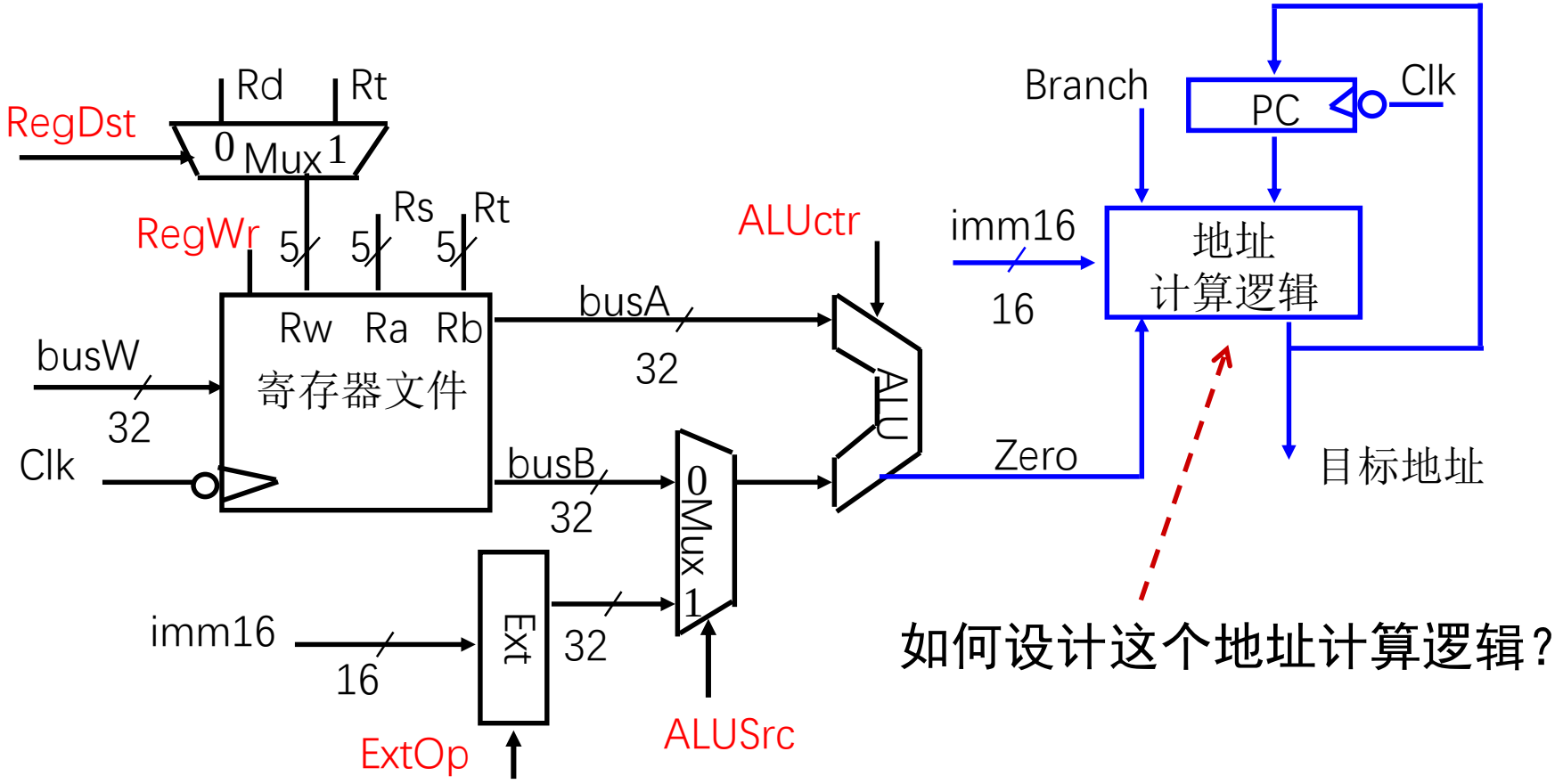
- :sw rt, rs, imm16： M[R[rs] + SignExt[imm16]]← R[rt]



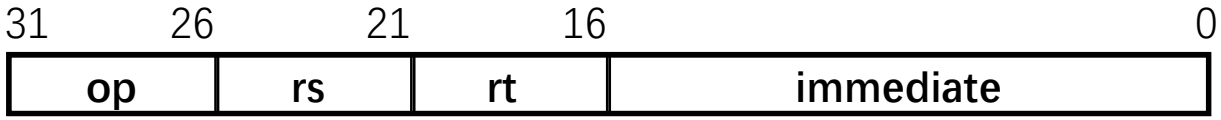
ALUctr=add, ALUSrc=1, ExtOp=1, MemWr=1, MemtoReg=x , RegDst=x, RegWr=0

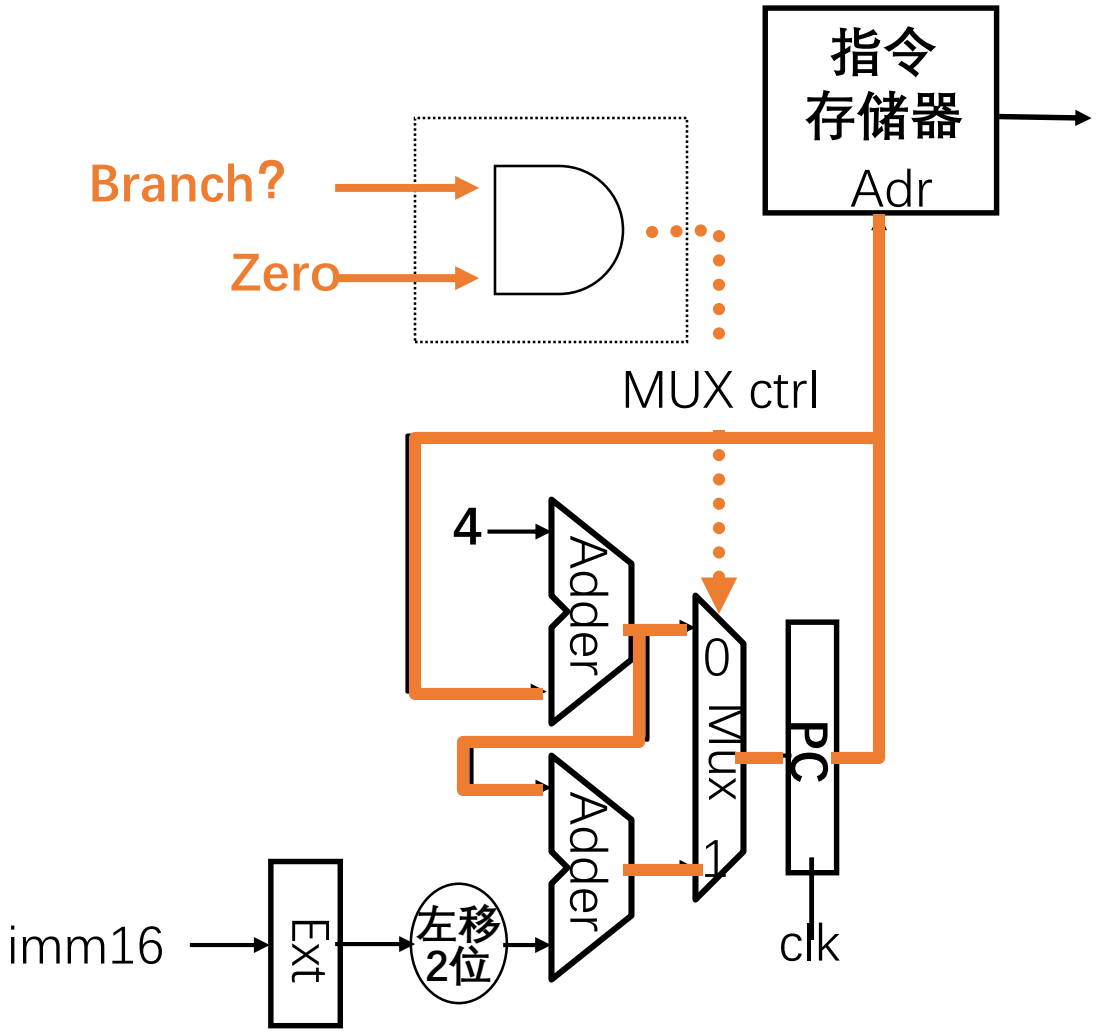
Beq 数据通路

- beq　rs, rt, imm16
- **if (R[rs] == R[rt]) then PC <– PC + sign_ext(Imm16)] || 00 else PC <– PC + 4**



RegDst=x, RegWr=0, ALUctr=sub, ExtOp=1, ALUSrc=0, MemWr=0, MemtoReg=x, Branch=1





if (Zero == 1) and (Branch ==1)
then PC = PC + 4 + SignExt[imm16]*4 ;
else PC = PC + 4

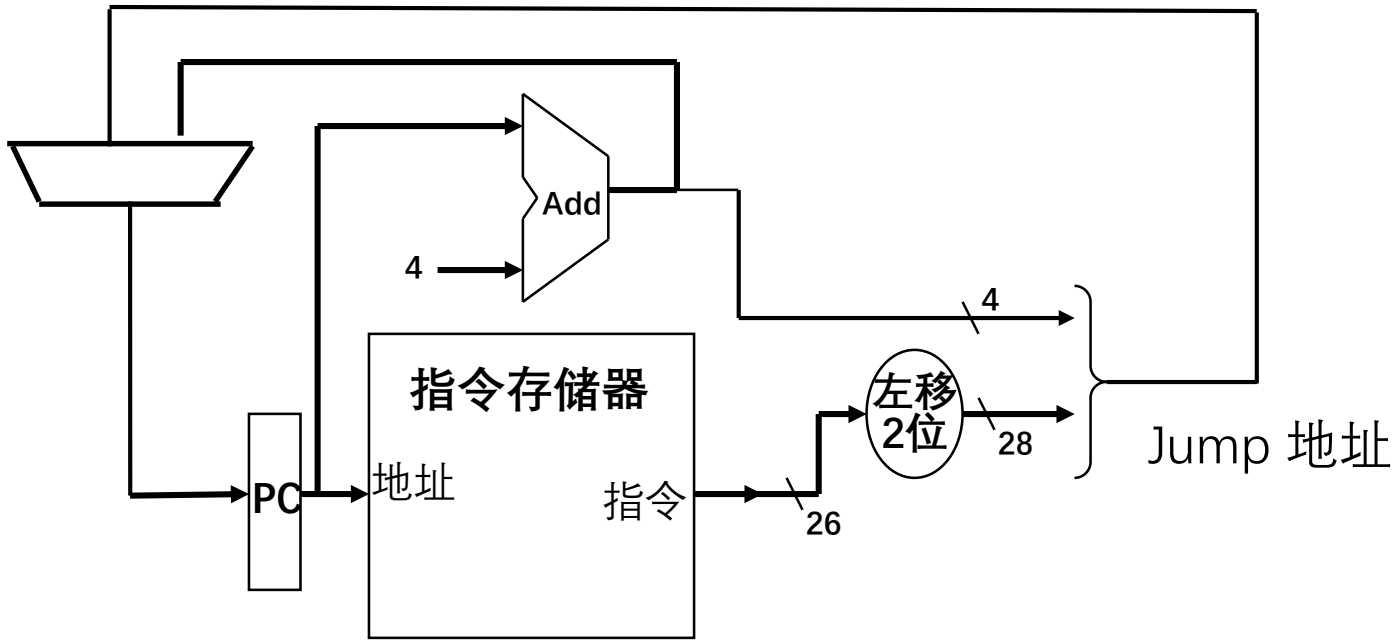
- MUX_ctrl （选路器控制逻辑）

Branch?	zero?	MUX
0	x	0
1	0	0
1	1	1

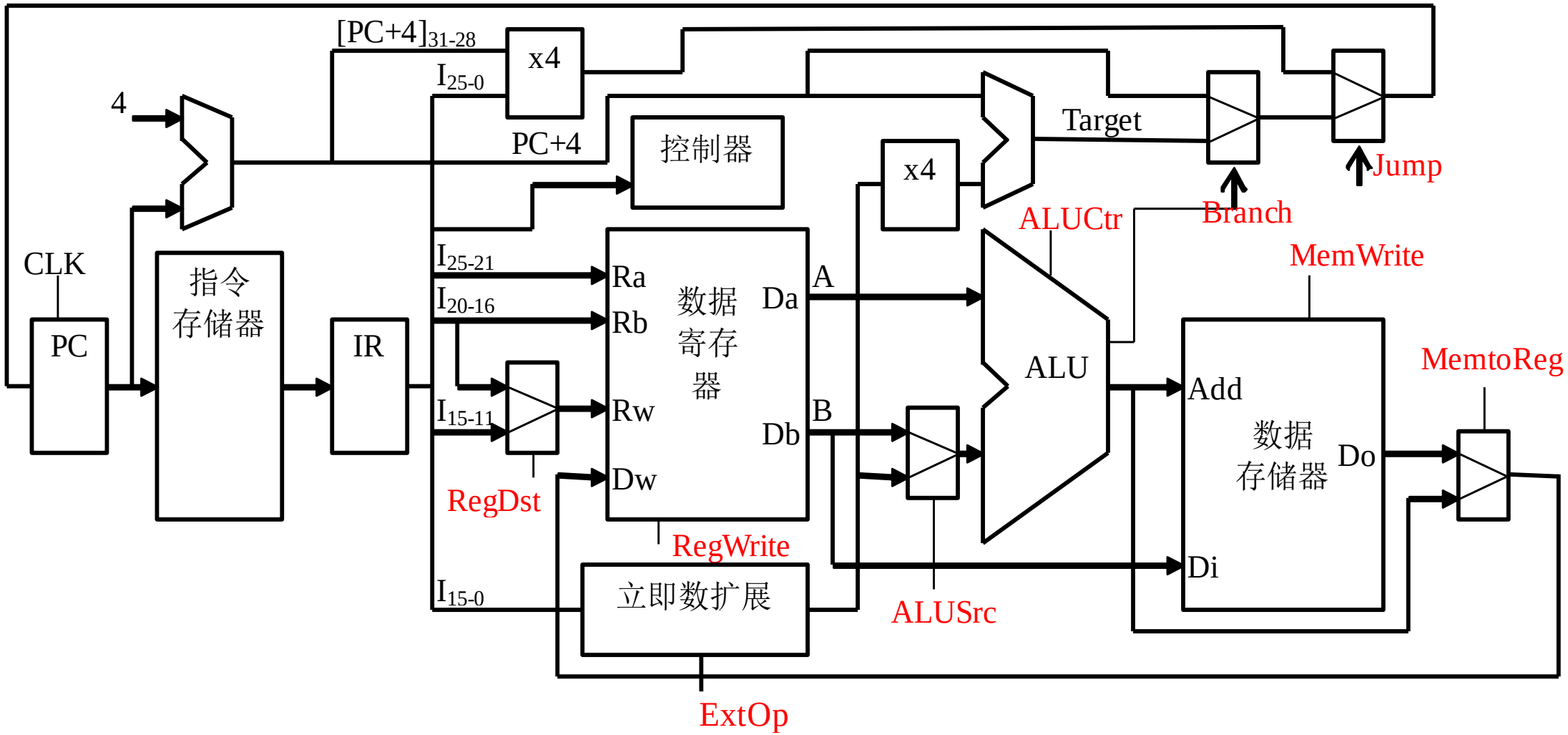
Jump 指令

0x02	26 位地址

- **JUMP： PC <– (PC +4[31-28], $I_{25\text{-}0}$) || 00**
- 指令中的最后26位左移2位后，与PC+4的前4位拼接



完整的数据通路


[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

小结

- **连接组件构成数据通路**
- **控制信号**