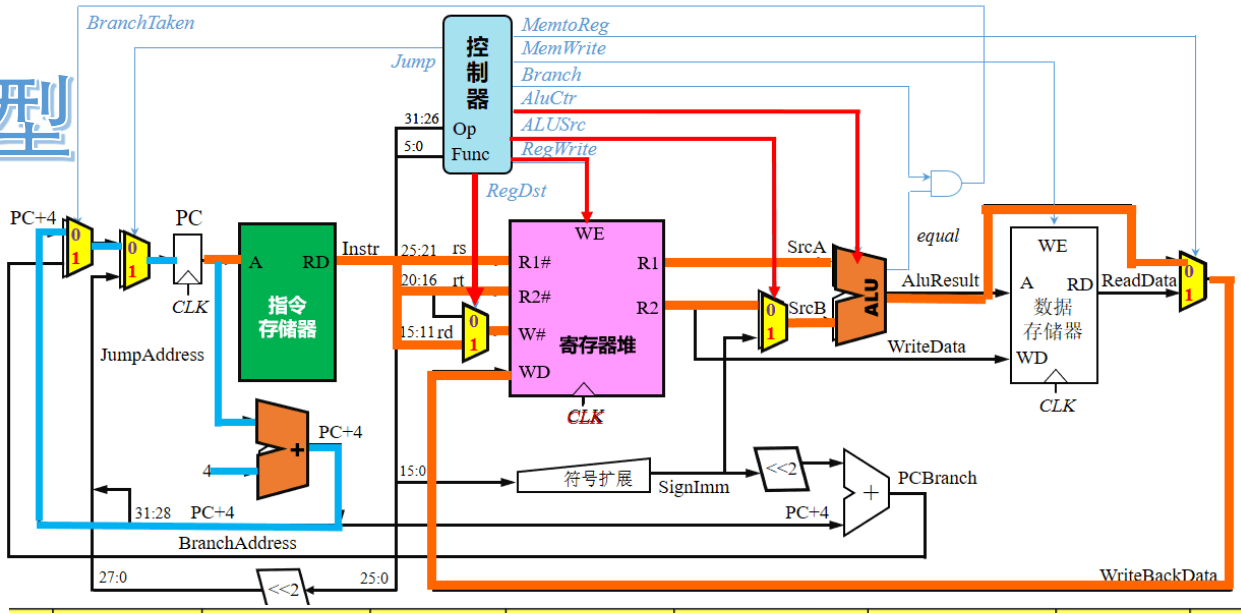
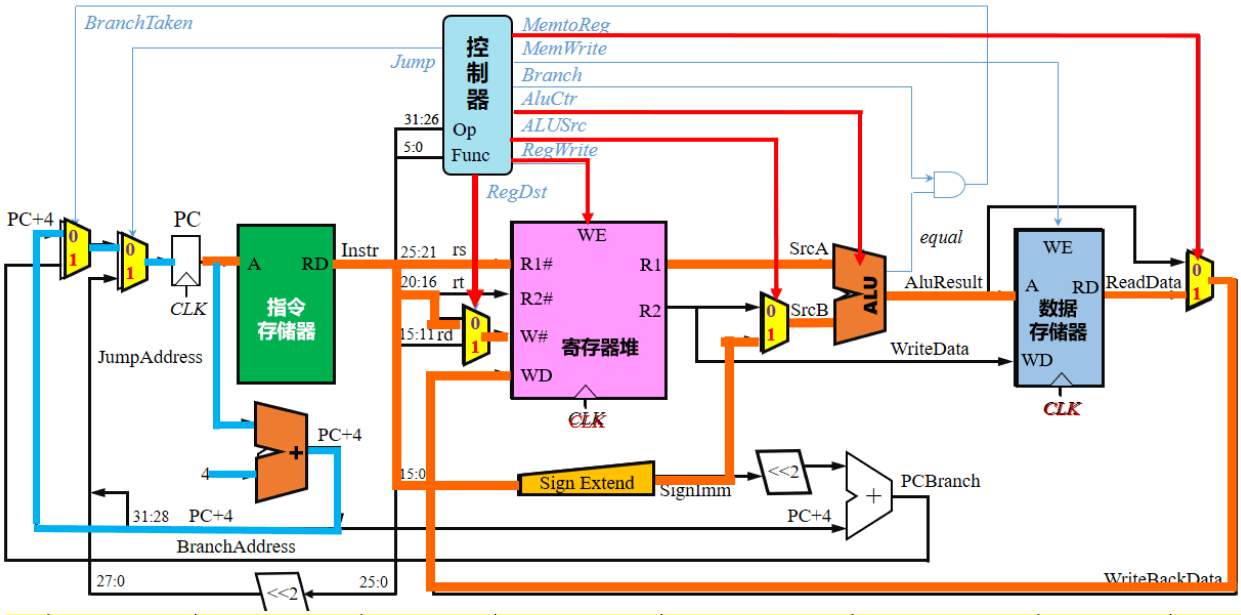
名	RegDst	RegWrite	ALUSrc	ALUCtr	MemtoReg	MemWrite	Branch	Jump
	1	1	0	add/sub/or/and/slt	0	0	0	0

R 型



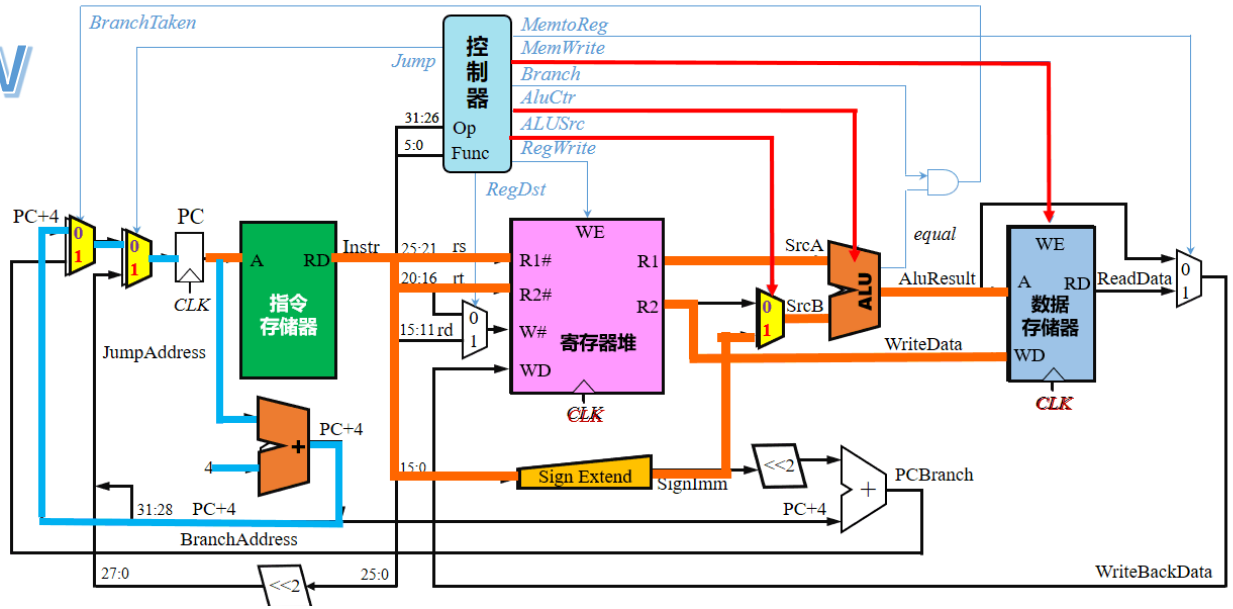
名	RegDst	RegWrite	ALUSrc	ALUCtr	MemtoReg	MemWrite	Branch	Jump
	0	1	1	add	1	0	0	0

lw

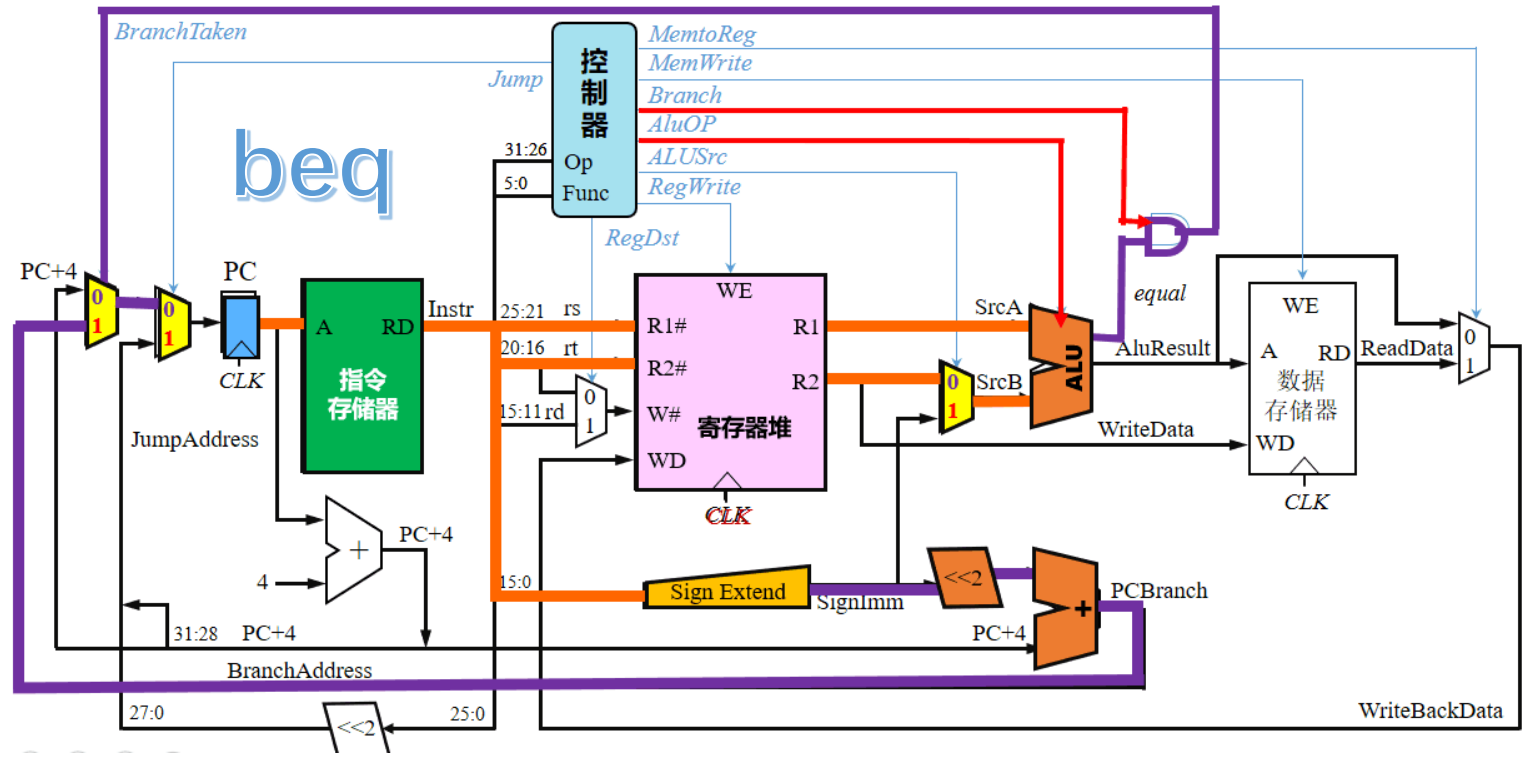


名	RegDst	RegWrite	ALUSrc	ALUCtr	MemtoReg	MemWrite	Branch	Jump
	X	0	1	add	X	1	0	0

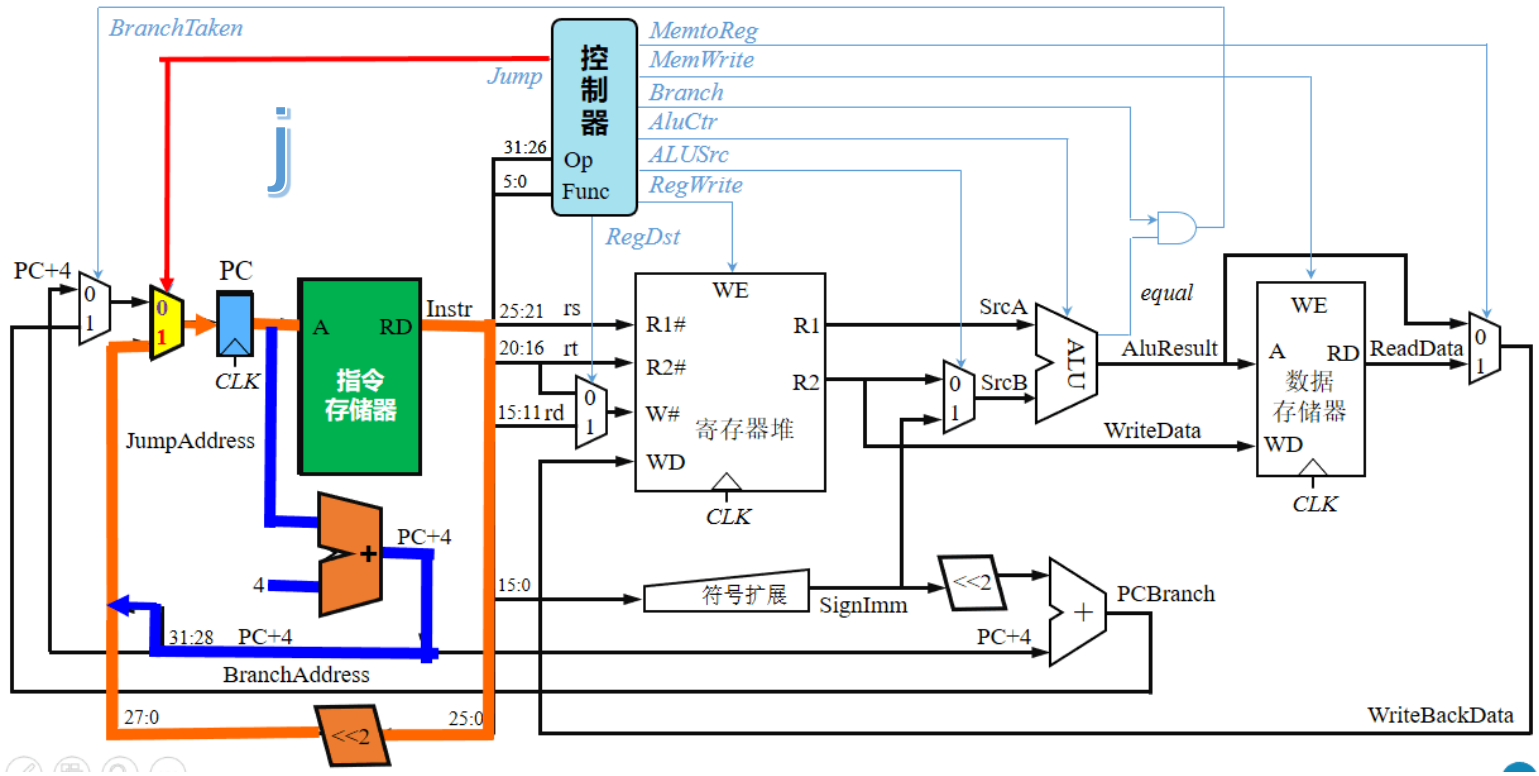
SW



名	RegDst	RegWrite	ALUSrc	ALUCtr	MemtoReg	MemWrite	Branch	Jump
出	X	0	0	sub	X	0	1	0

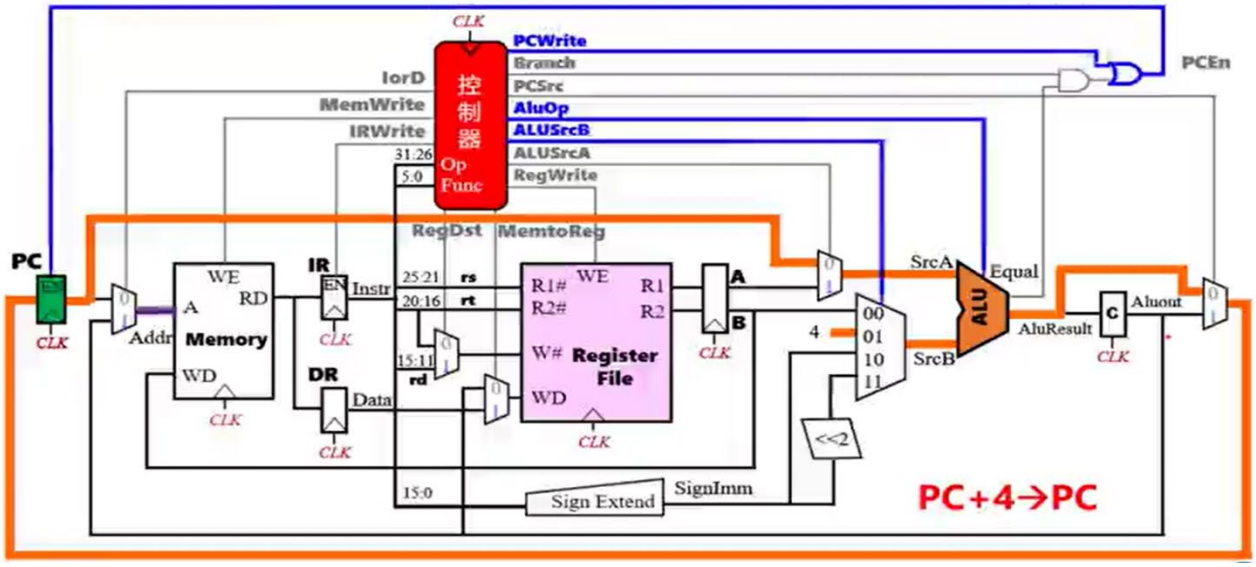


名	RegDst	RegWrite	ALUSrc	ALUCtr	MemtoReg	MemWrite	Branch	Jump
出	X	0	X	X	X	0	0	1



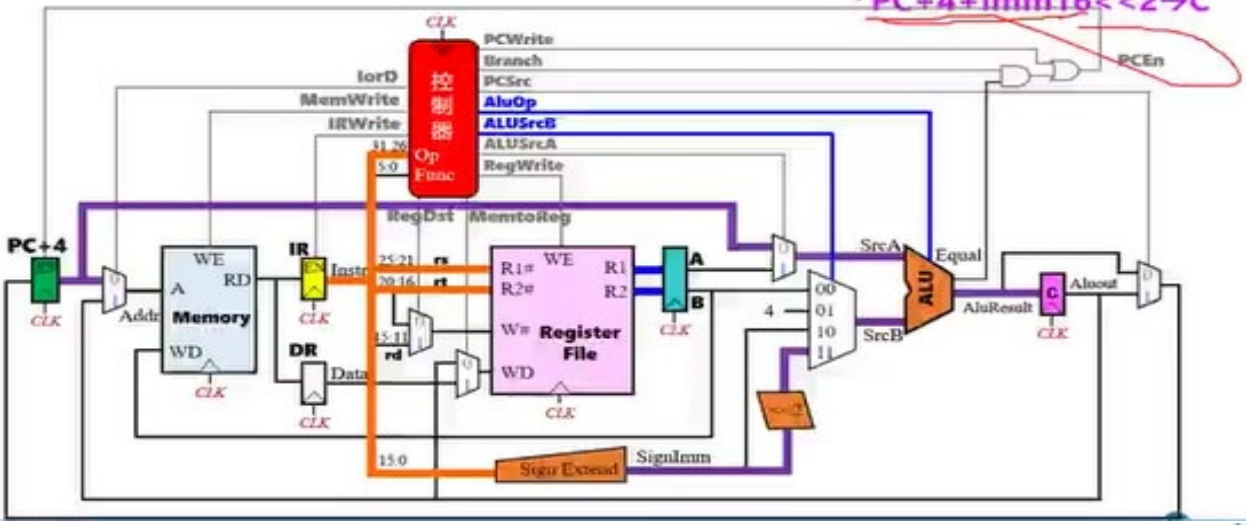
③ 取指令阶段T1的数据通路

并行执行：M[PC]→IR
PC+4→PC

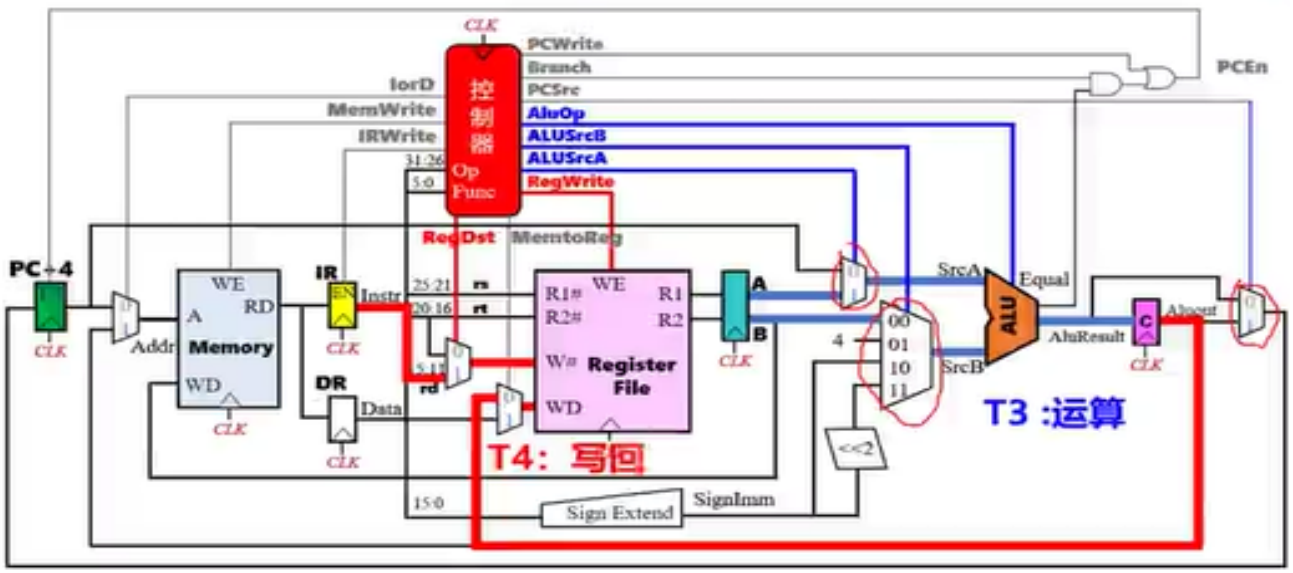


④ 译码/取数阶段T2的数据通路

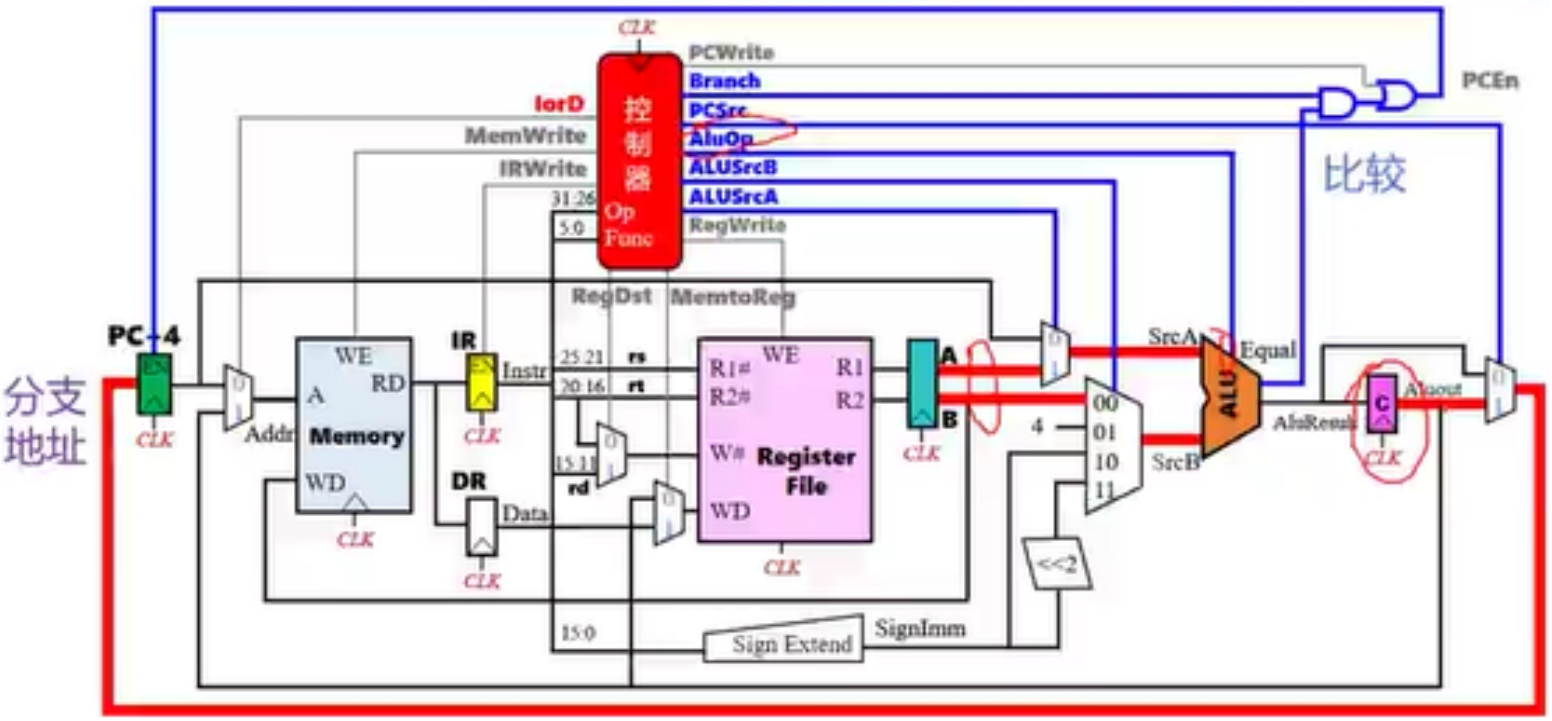
- 指令译码；
- 并行执行：
  - ✓ 取寄存器操作数→A、B
  - ✓ 提前计算分支目标地址
    PC+4+Imm16<<2→C



⑤ R型指令执行阶段T3~T4的数据通路



⑦ Beq指令执行阶段T3的数据通路



⑥ LW指令执行阶段T3~T5的数据通路

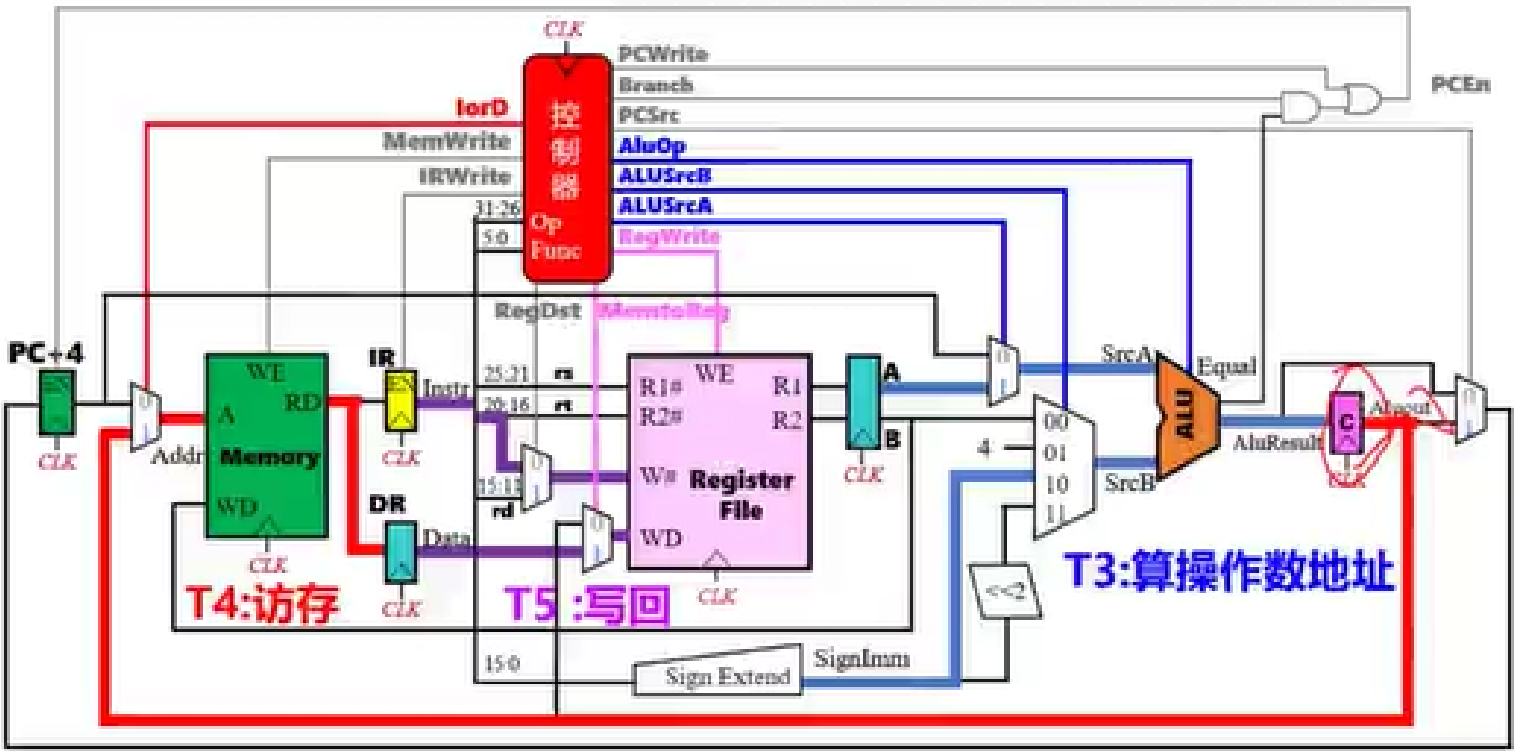


表 5.7　MIPS 寄存器功能说明

编号	助记符	英文全称	功能描述
$0	$zero	zero	恒零值，可用 0 号寄存器参与的加法指令实现 MOV 指令
$1	$at	Assembler Temp	汇编器保留寄存器，常用作伪指令的中间变量
$2 ～ 3	$v0 ～ $v1	Value	存储子程序的非浮点返回值
$4 ～ 7	$a0 ～ $a3	Argument	用于存储子程序调用的前 4 个非浮点参数
$8 ～ 15	$t0 ～ $t7	Temporaies	临时变量，调用者保存寄存器，在子程序中可直接使用
$16 ～ 23	$s0 ～ $s7	Saved Registers	通用寄存器，被调用者保存寄存器，在子程序中使用时必须先压栈保存原值，使用后应出栈恢复原值
$24 ～ 25	$t8 ～ $t9	Temporaies	临时变量，属性同 $t0 ～ $t7
$26 ～ 27	$k0 ～ $k1	Kernel Reserved	操作系统内核保留寄存器，用于进行中断处理
$28	$gp	Global Pointer	全局指针
$29	$sp	Stack Pointer	栈指针，指向栈顶
$30	$fp/$s8	Frame Pointer	帧指针，用于过程调用，也可以当作 $s8 使用
$31	$ra	Return Address	子程序返回地址

funct	指令助记符	指令功能描述	备注
00	sll rd,rt,shamt	R[rd]=R[rt]<<shamt	逻辑左移，注意 rs 字段未使用
02	srl rd,rt,shamt	R[rd]=R[rt]>>shamt	逻辑右移，注意 rs 字段未使用
03	sra rd,rt,shamt	R[rd]=R[rt]>>shamt	算术右移，注意 rs 字段未使用
04	sllv rd,rt,rs	R[rd]=R[rt]<<R[rs]	可变左移
08	jr rs	PC=R[rs]	R[rs] 值应是 4 的倍数，字对齐
09	jalr rs	R[31]=PC+8 PC=R[rs]	子程序调用
12	syscall	系统调用	无操作数
16	mfhi rd	R[rd]=HI	取 HI 寄存器的值，mflo 取 LO
17	mthi rs	HI=R[rs]	存 HI 寄存器的值，mtlo 存 LO
24	mult rs,rt	{HI,LO}=R[rs]*R[rt]	有符号乘，64 位结果送入 HI、LO 寄存器
32	add rd,rs,rt	R[rd]=R[rs]+R[rt]	溢出时发生异常，且不修改 R[rd]
34	sub rd,rs,rt	R[rd]=R[rs]-R[rt]	溢出时发生异常，且不修改 R[rd]
36	and rd,rs,rt	R[rd]=R[rs]&R[rt]	逻辑与
37	or rd,rs,rt	R[rd]=R[rs]|R[rt]	逻辑或
42	slt rd,rs,rt	R[rd]=(R[rs]<R[rt])?1:0	小于置位指令，有符号比较

04	beq rs,rt,imm	if(R[rs]==R[rt]) PC=PC+4+imm<<2	条件分支（相等跳转）
05	bne rs,rt,imm	if(R[rs]!=R[rt]) PC=PC+4+imm<<2	条件分支（不等跳转）
08	addi rt,rs,imm	R[rt]=R[rs]+imm	立即数加，溢出发生异常
10	slti rt,rs,imm	R[rt]=(R[rs]<imm)?1:0	小于置位指令，有符号比较
12	andi rt,rs,imm	R[rt]=R[rs]&imm	立即数逻辑与指令
15	lui rt,imm	R[rt]=imm<<16	加载立即数指令
35	lw rt,imm(rs)	R[rt]=M[R[rs]+imm]	取数指令，类似指令还有 lb、lh、lbu 等
43	sw rt,imm($rs)	M[R[rs]+imm]=R[rt]	存数指令，类似指令还有 sb、sh

02	j address	PC ← $\{(PC+4)_{31:28}$,address,00$\}$	无条件分支
03	jal address	R[31] ← PC+8（无延迟槽 +4） PC ← $\{(PC+4)_{31:28}$,address,00$\}$	子程序调用指令

表 7.5　采用 BTB 后流水线运行的情况

指令在 BTB 中	预测情况	下条指令地址	实际情况	预测情况	流水停顿周期
命中	预测跳转	分支目标地址	跳转	预测成功	0
命中	预测跳转	分支目标地址	未跳转	预测失败	2
命中	预测不跳转	顺序地址	跳转	预测失败	2
命中	预测不跳转	顺序地址	未跳转	预测成功	0
缺失		顺序地址	跳转		2
缺失		顺序地址	未跳转		0

$$T_{\min\_clk} = \max\left(T_{if\_max}, T_{id\_max}, T_{ex\_max}, T_{ex\_max}, T_{mem\_max}, T_{wb\_max}\right)$$

表 7.6　流水线各功能段关键延迟

功能段	标识	功能段延迟	65nm CMOS 工艺实际值
IF	T_{if_max}	$T_{clk\_to\_q} + T_{mem} + T_{setup}$	300ps = 30 + 250 + 20
ID	T_{id_max}	$2(T_{clk\_to\_q} + T_{RF\_read} + T_{setup})$	400ps = 2 × (30 + 150 + 20)
EX	T_{ex_max}	$T_{clk\_to\_q} + 2T_{mux} + T_{alu} + T_{setup}$	300ps = 30 + 2 × 25 + 200 + 20
MEM	T_{mem_max}	$T_{clk\_to\_q} + T_{mem} + T_{setup}$	300ps = 30 + 250 + 20
WB	T_{wb_max}	$T_{clk\_to\_q} + T_{mux} + T_{setup}$	75ps = 30 + 25 + 20

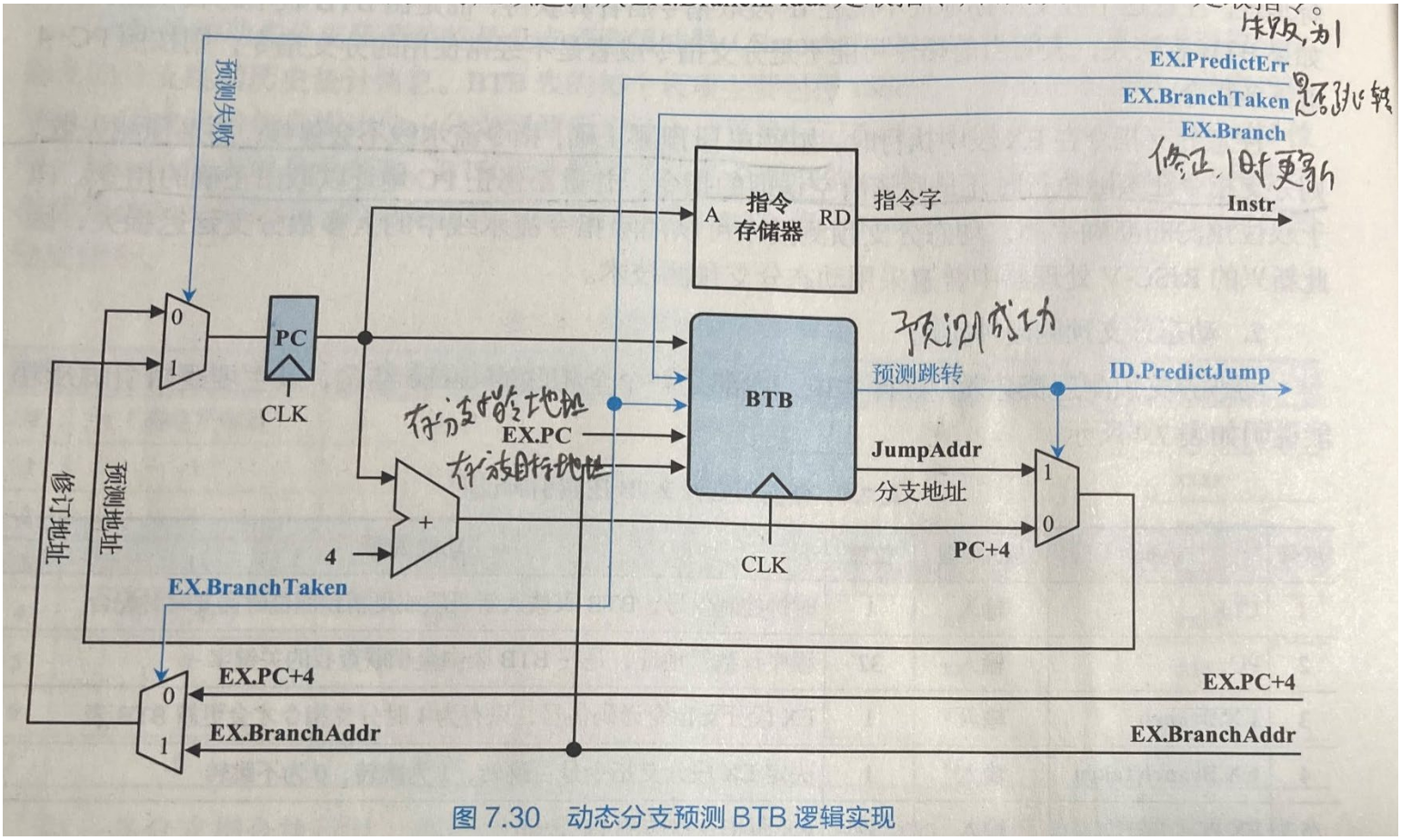


图 7.30　动态分支预测 BTB 逻辑实现