郑州大学

计算机与人工智能学院

硬件技术课程设计

（计算机科技与技术专业）

张大伟

iedwzhang@zzu.edu.cn

18736002097（手机微信同号）

2023-4

硬件课程设计介绍

- 教学目标
  - 使学生能够对计算机硬件技术有深入的研究
  - 具有一定的计算机硬件设计或应用能力
  - 具有计算机底层软硬件技术相结合的编程开发能力
- 设计任务——基于单/多周期MIPS指令系统的CPU设计与仿真
  - MIPS单周期CPU设计
  - MIPS多周期CPU微程序控制器设计
  - MIPS指令集汇编语言设计
  - 仿真测试

硬件课程设计介绍

■ 课程设计安排

□准备阶段：布置任务、集中讲解（第8周周四下午14：00—15：40，集中讲解）

□实验阶段：查阅资料、方案设计、动手实践、及时汇报实验进展、存在的问题，独立完成课程设计内容。（第9-10周，线上答疑时间：周四上午9：00—10：30）必须参加，随机提问

□验收阶段：成果演示、检查验收、学生答辩。（第11周周四全天，线上验收答辩，上午8：30开始，每人3—5分钟，按班级学号依次进行）必须参加，在线演示+问题回答

□验收结束一周后提交课程设计报告，包括课程设计概述、总体方案设计、详细设计与实现、实验过程与调试、设计总结与心得等。要求在适当位置增加相应的专业知识、电路原理设计图、数据通路图、控制器设计逻辑表、程序设计进行说明。（第12周周五提交课程设计报告）

硬件课程设计介绍

课程进度安排

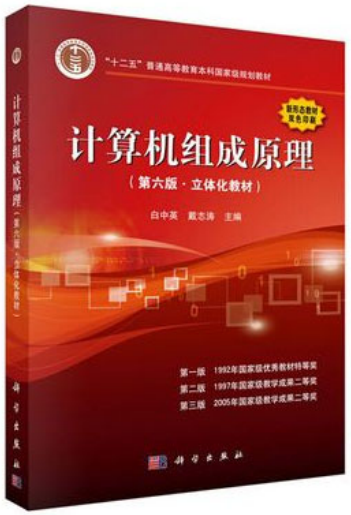
- 查阅资料：单周期指令CPU结构，Logisim使用方法，8条MIPS指令格式及功能，MARS使用方法，参考书籍/视频课程。（1-3天）
- 实验阶段1：单周期指令CPU结构数据通路绘制，控制器设计，MIPS排序程序设计、调试与仿真。（4-5天）
- 查阅资料：多周期指令CPU结构，微程序控制器原理，微程序控制器指令格式与设计，参考书籍/视频课程（6-7天）
- 实验阶段2：多周期指令CPU结构数据通路绘制、微指令与微程序控制器设计、MIPS指令调试与仿真。（8-9天）
- 答辩验收（10天）

硬件课程设计介绍

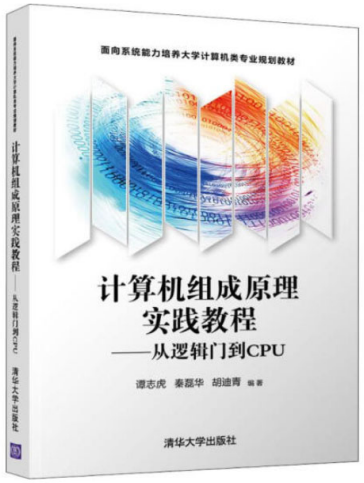
- 课程设计所需主要仪器
  - 基于Windows个人计算机（PC机），在虚拟平台上完成设计任务。
- 考核标准
  - 总成绩＝课程设计过程与结果成绩（70%）+课程设计报告（30%）
  - 其中，课程设计过程与结果成绩（70%）＝难度系数（10%）＋面试验收（线上演示及答辩40%）＋平时表现（线上抽查课程设计进度20%）
- 备注：答辩结束一周后，每人提供课程设计报告pdf与电路原理图circ，以“学号-名字”命名。

硬件课程设计介绍

■ 参考教材及学习资源



第四章：指令系统
第五章：中央处理器



第五章：MIPS处理器设计实验

中国大学生慕课App

回顾计算机组成原理实验（Logisim和MARS）

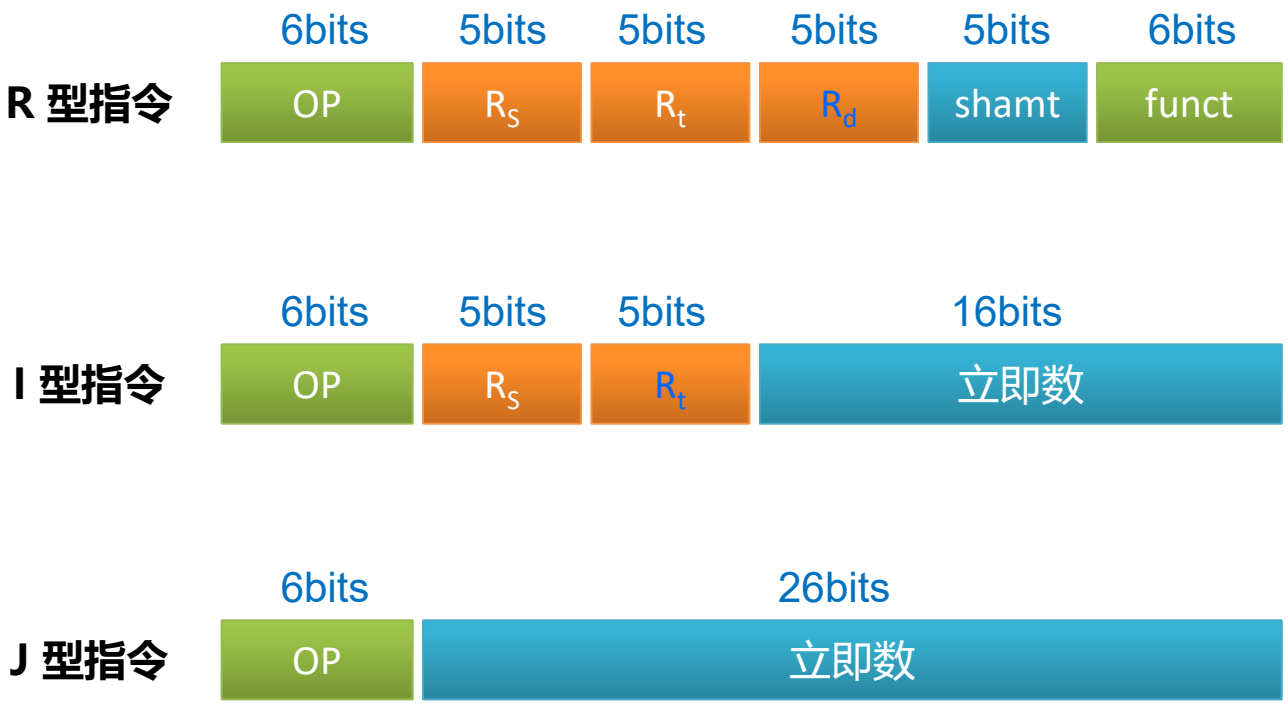
学习资料

- 链接：https://pan.baidu.com/s/1crw5tGa8xZDdg5bPa3MsEQ 提取码：2bqe

MIPS 单/多周期 CPU设计

- 定长指令周期：单周期实现
  - **所有指令均在一个时钟周期内完成， CPI=1**
  - **性能取决于最慢的指令，时钟周期过长**
- 变长指令周期：多周期实现
  - **缩短时钟周期，复用器件或数据通路**
  - **可支持流水操作，提升性能**

MIPS指令格式


6bits
5bits
5bits
5bits
5bits
6bits
R 型指令
OP
R_S
R_t
R_d
shamt
funct
6bits
5bits
5bits
16bits
I 型指令
OP
R_S
R_t
立即数
6bits
26bits
J 型指令
OP
立即数

R型指令格式

R 型指令

6bits	5bits	5bits	5bits	5bits	6bits
OP	R_s	R_t	R_d	shamt	funct

- `add $s1,$s2,$s3`

0	18	19	17	0	32

- `sub $s0,$s1,$s2`

0	17	18	16	0	34

- `sll $s0,$s1,2`

0	0	17	16	2	0

I型指令格式

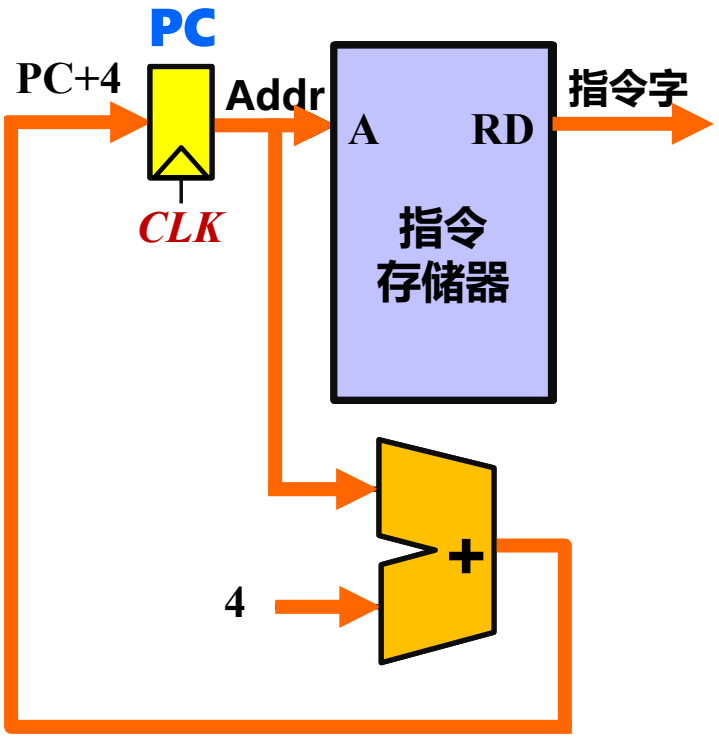
I 型指令

- `addi $s1,$s2,200`

- `lw $s1,300($s2)`

- `beq $s1,$s2,400`

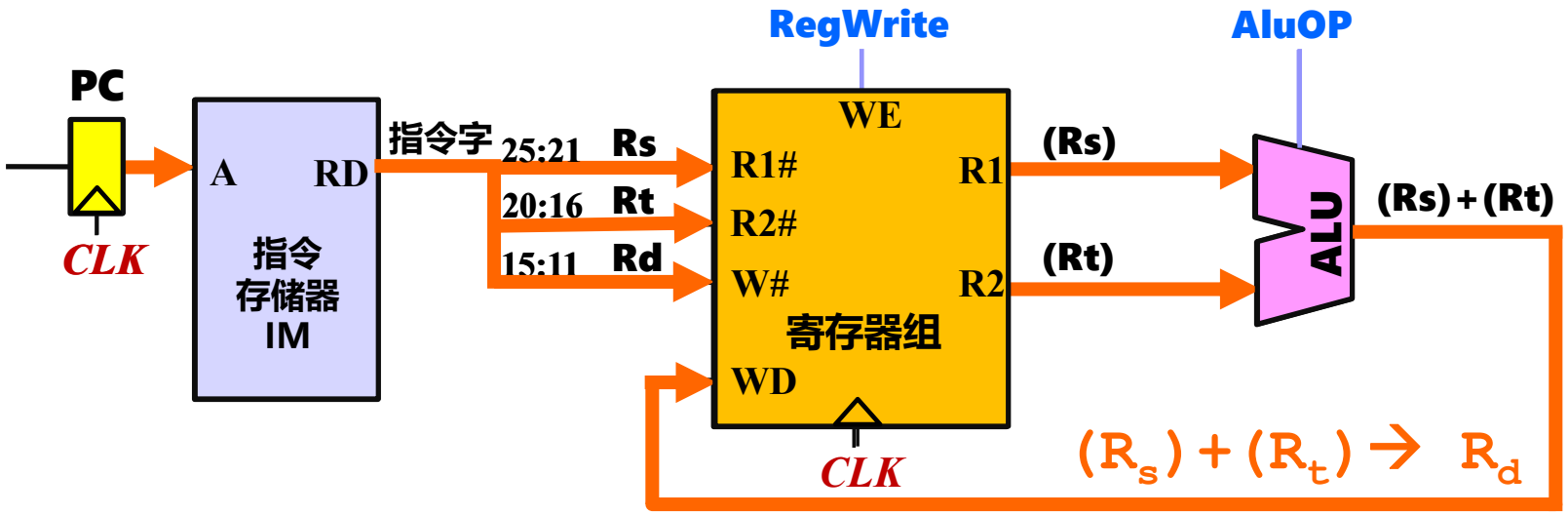
取指令数据通路



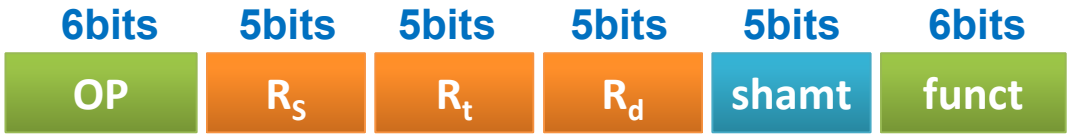
Mem[PC++] → IR

- 单周期不能设置AR，DR，IR寄存器
- 程序和数据分开存放---哈佛结构
  - 指令存储器　数据存储器
  - 指令cache　数据cache
- 运算器和PC累加器分离

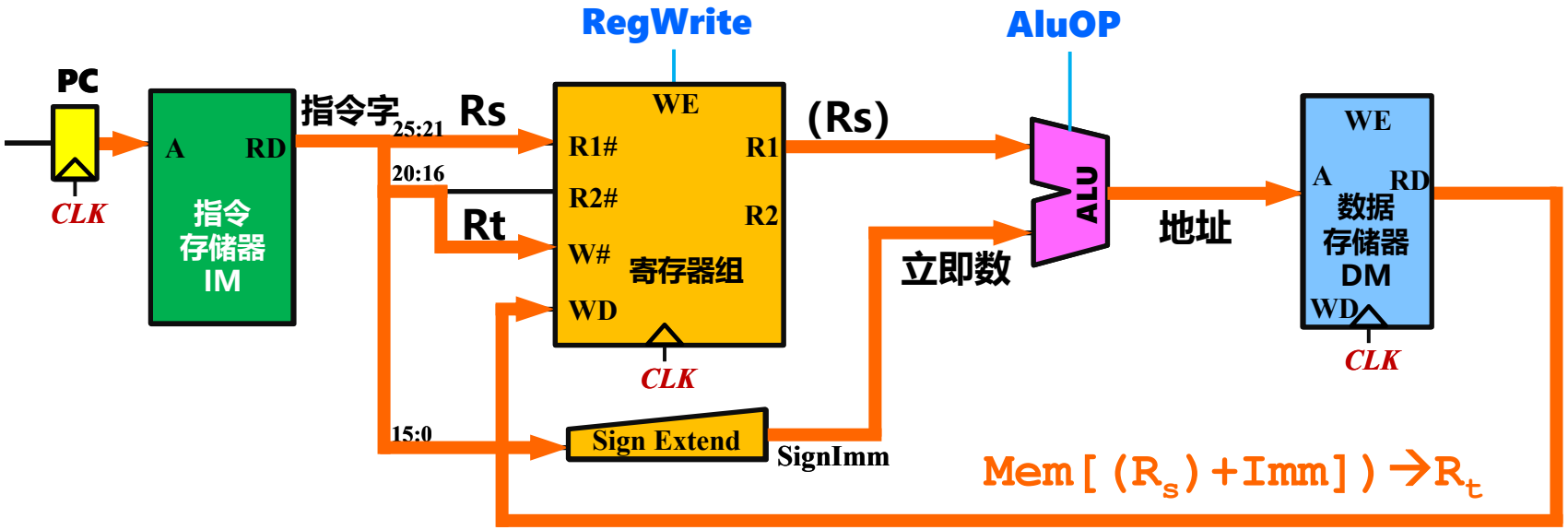
R型指令数据通路




```
add $s0,$s1,$s2
```

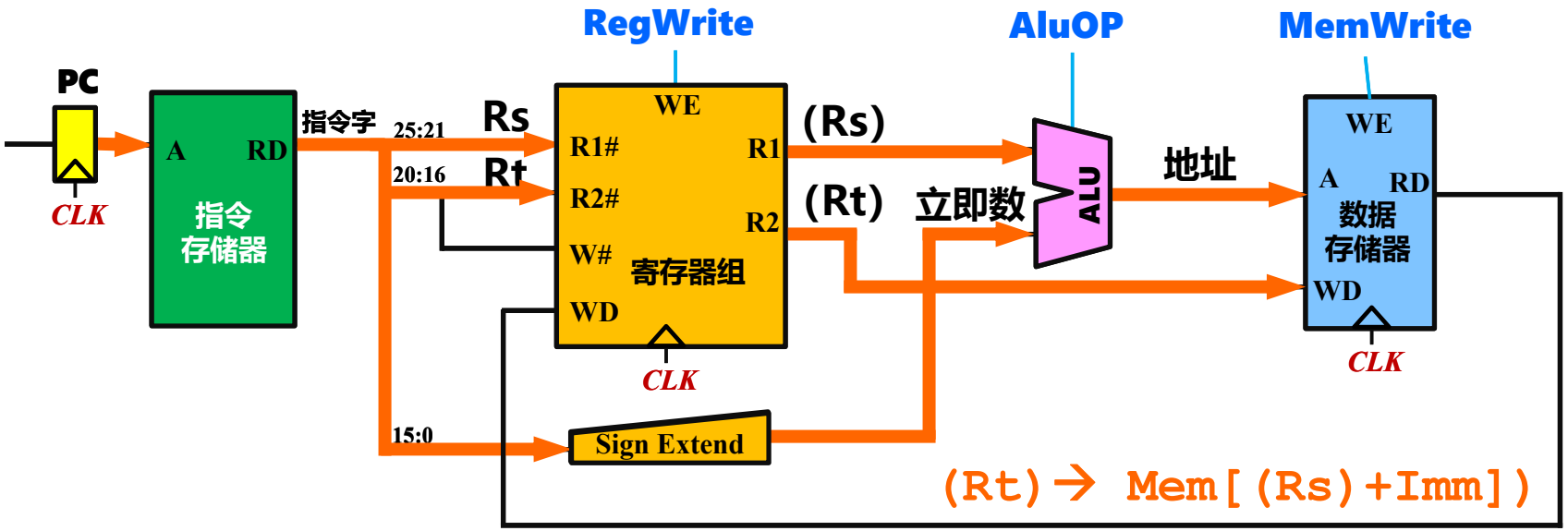


Lw指令数据通路



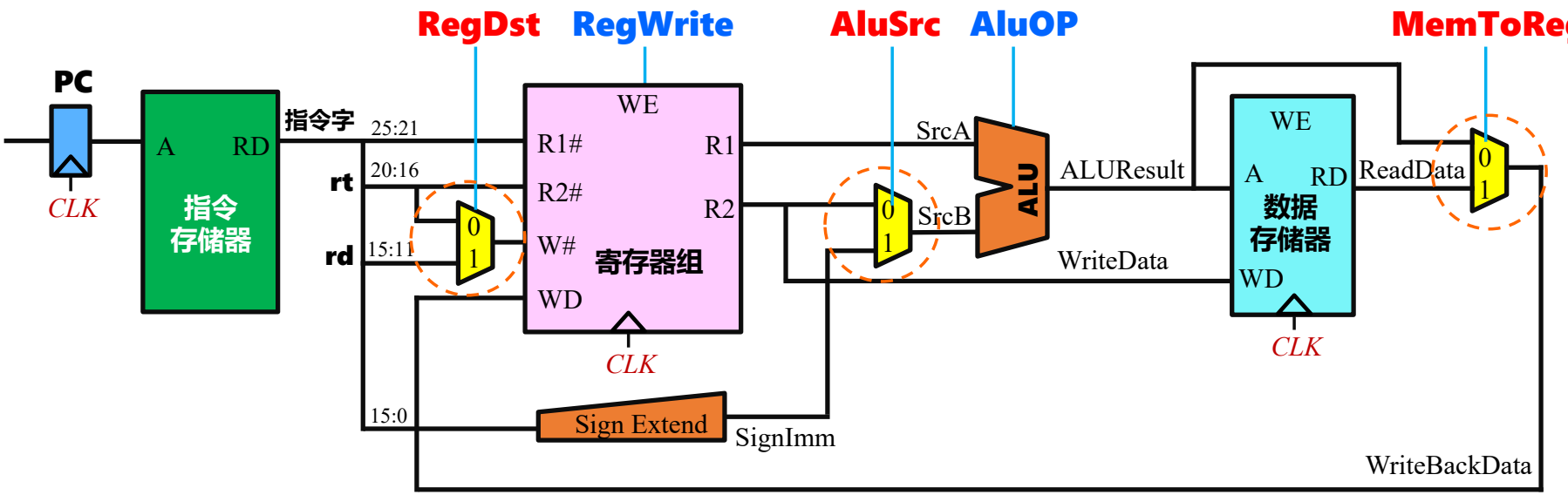

```
lw $s0,32($s1)
```

sw指令数据通路



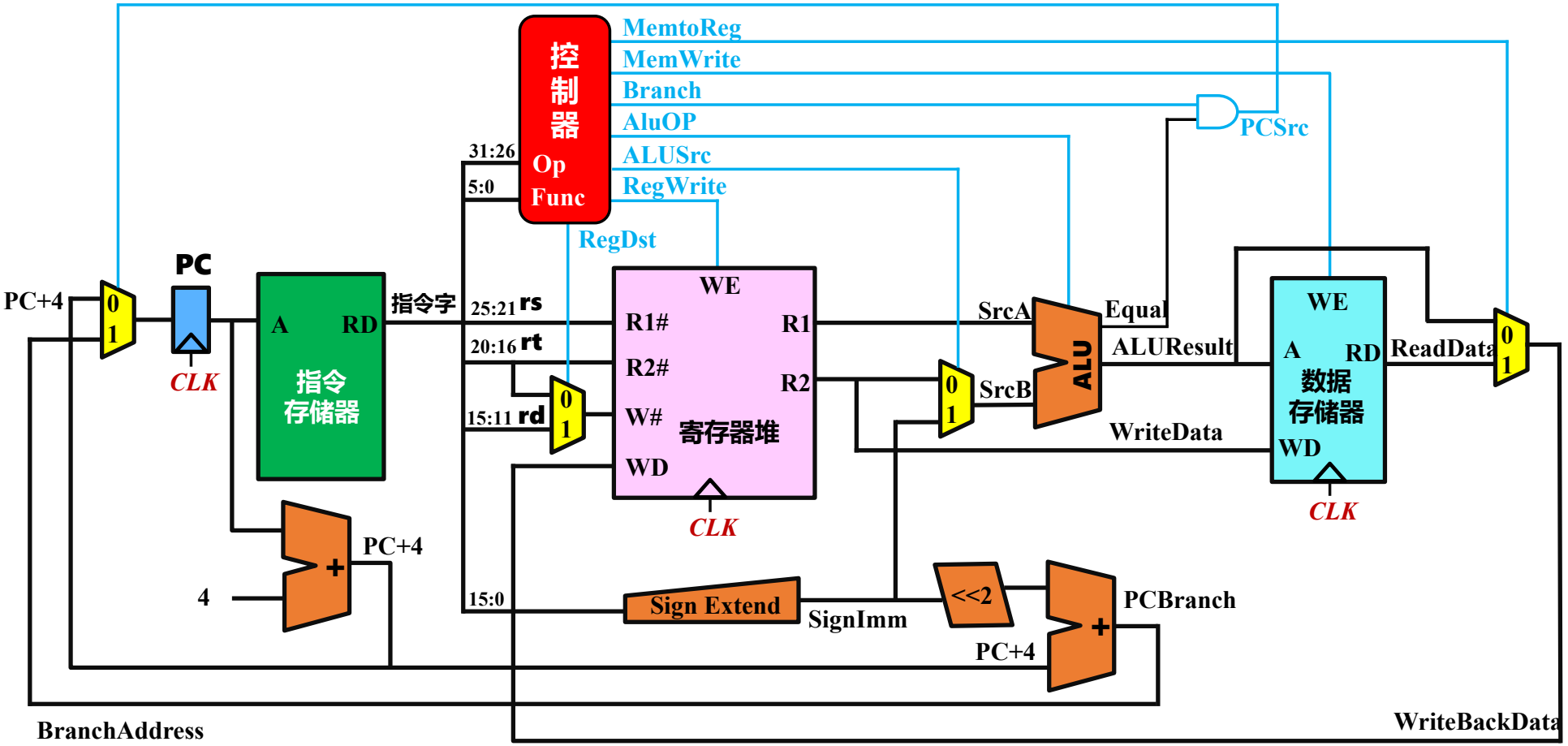
- `sw $s0,32($s1)`

数据通路综合

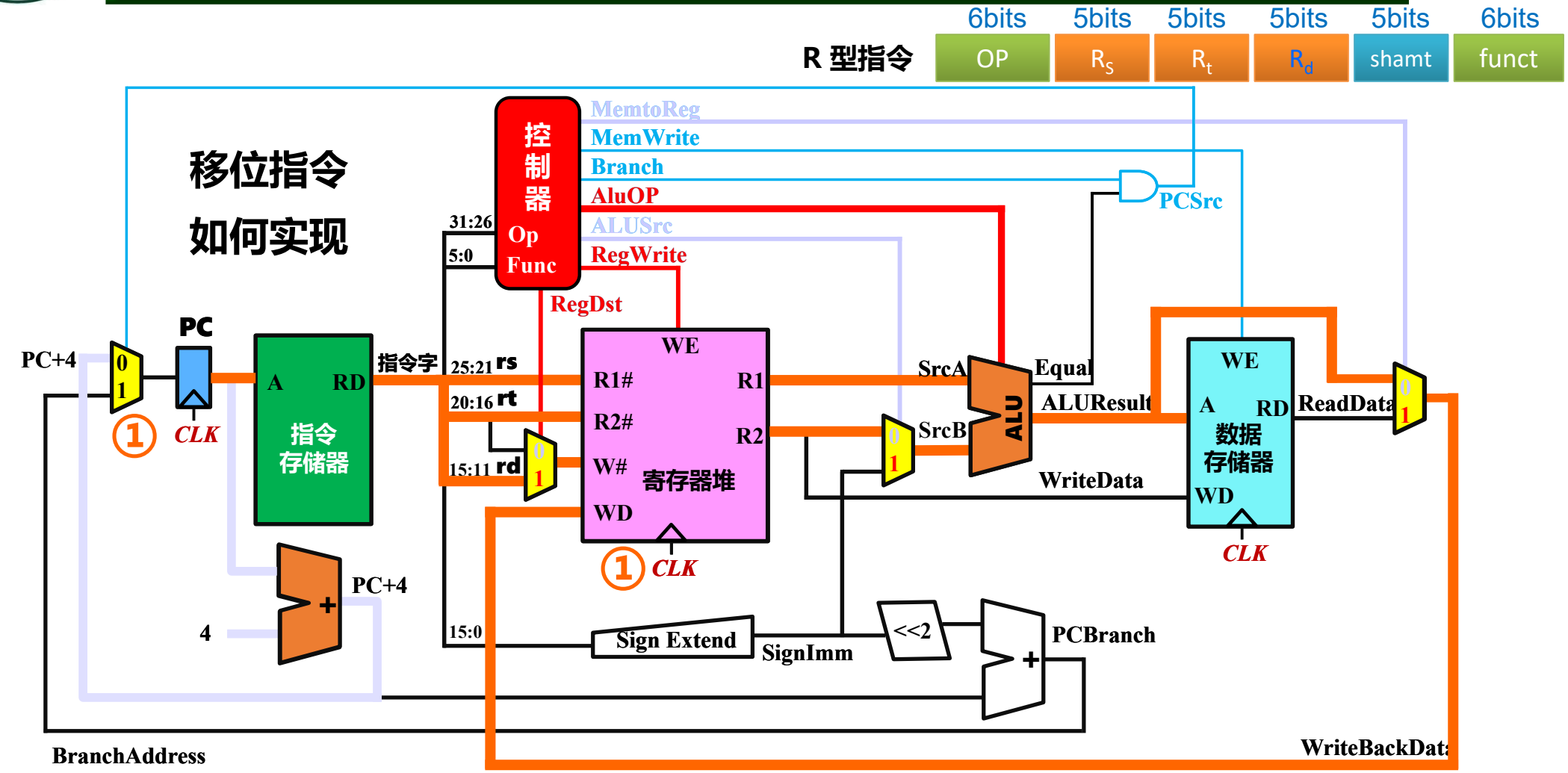


凡是有多个输入来源的，增加MUX，引入控点

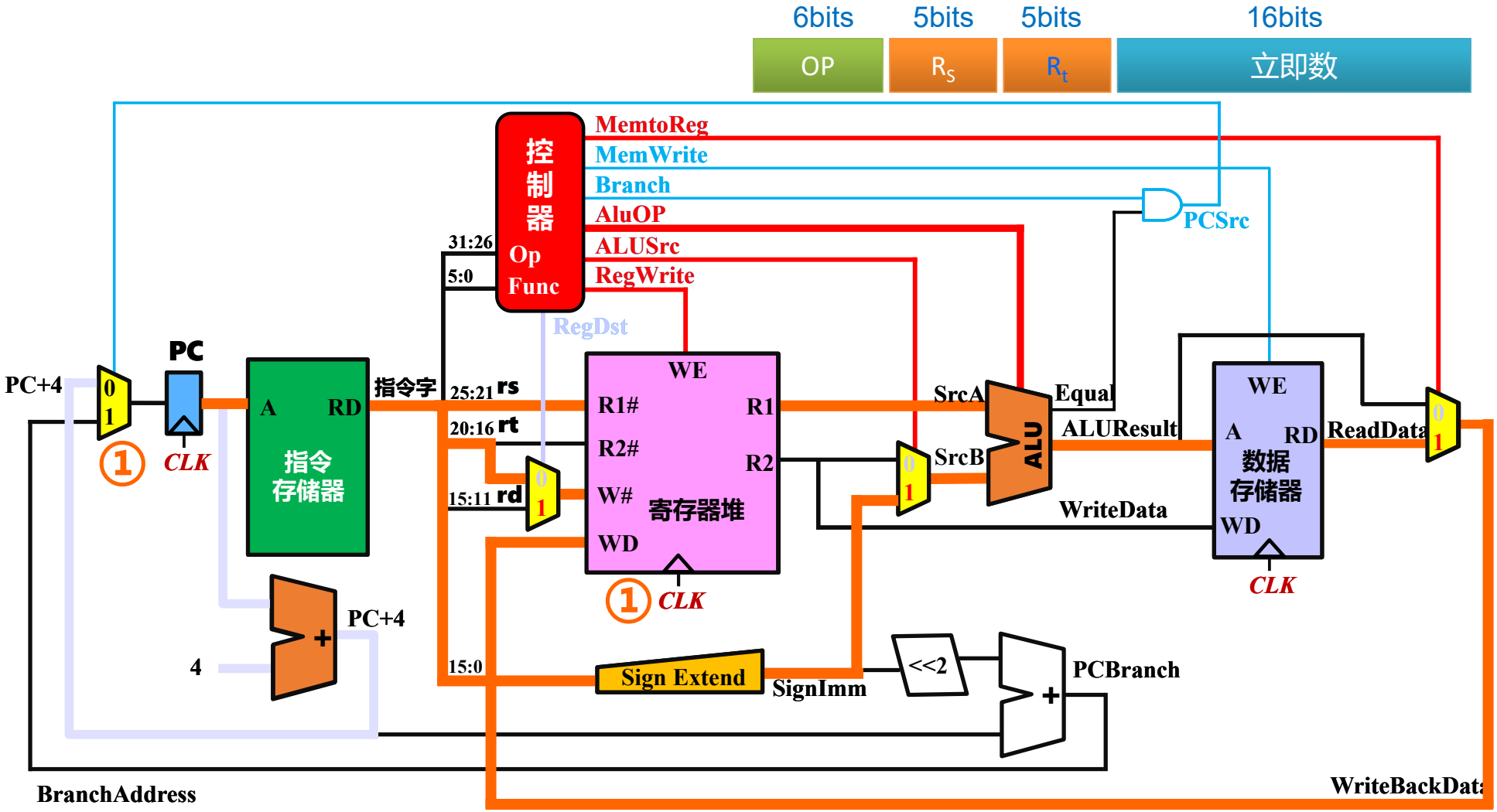
单周期MIPS数据通路



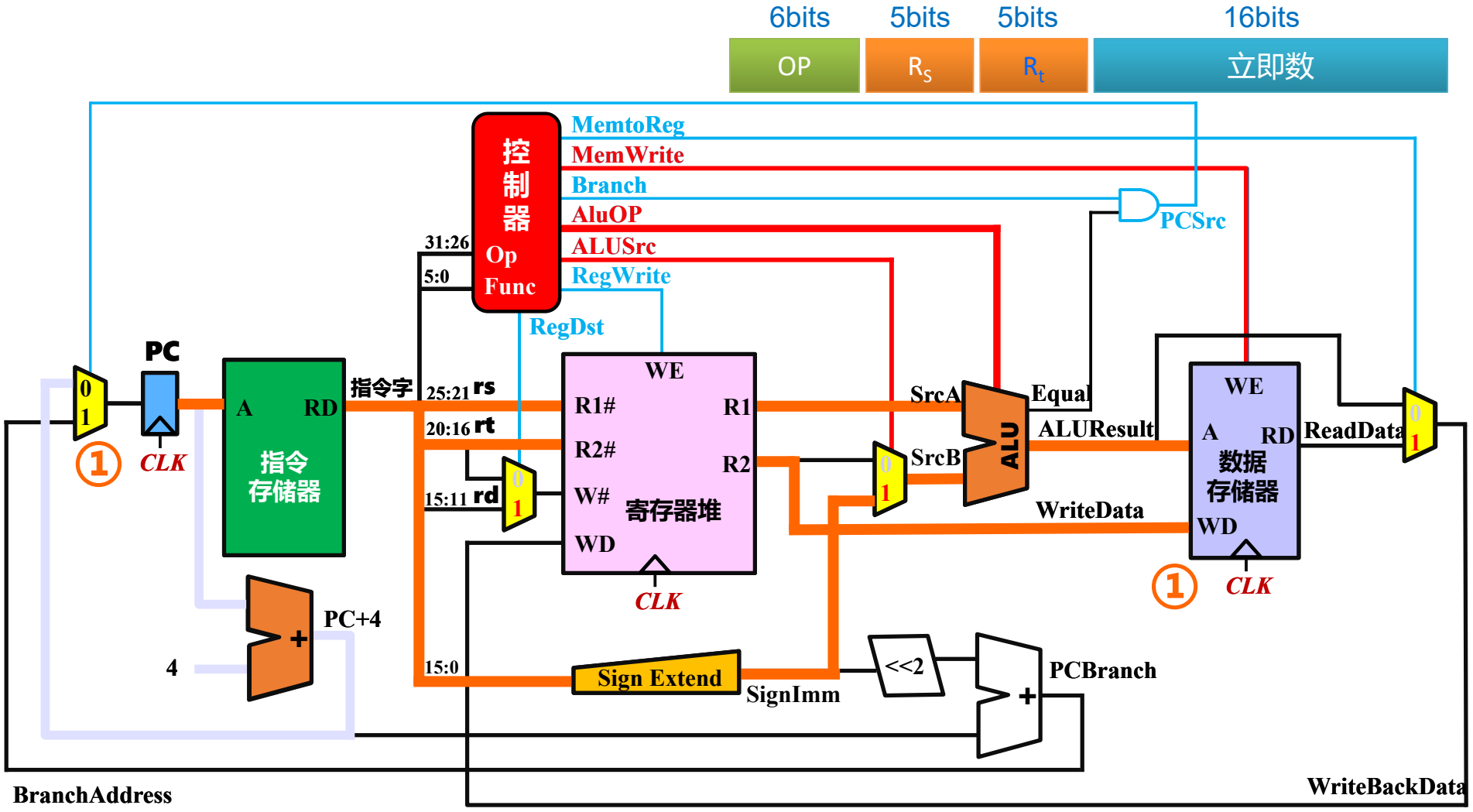
R型指令数据通路建立过程



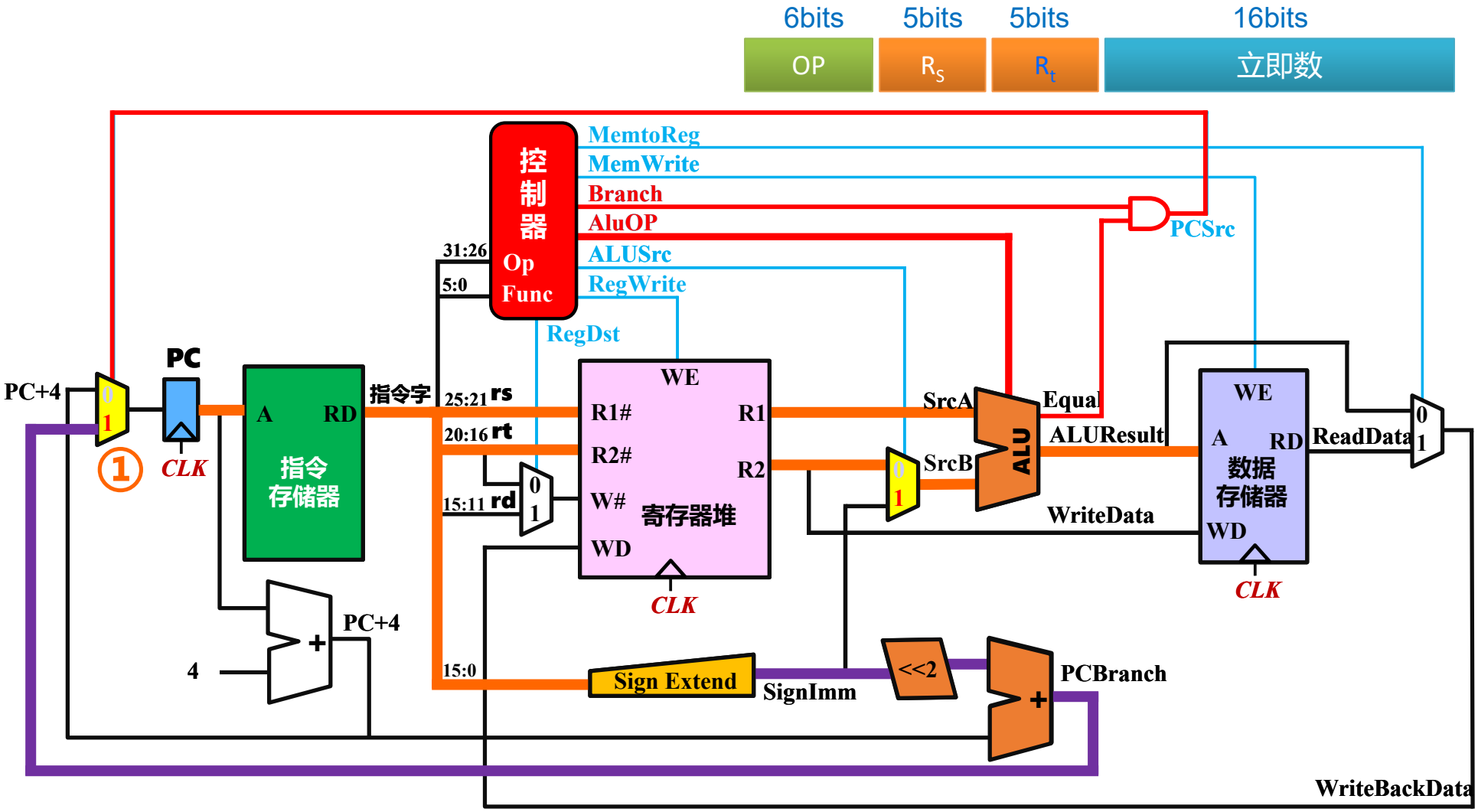
LW指令数据通路建立过程



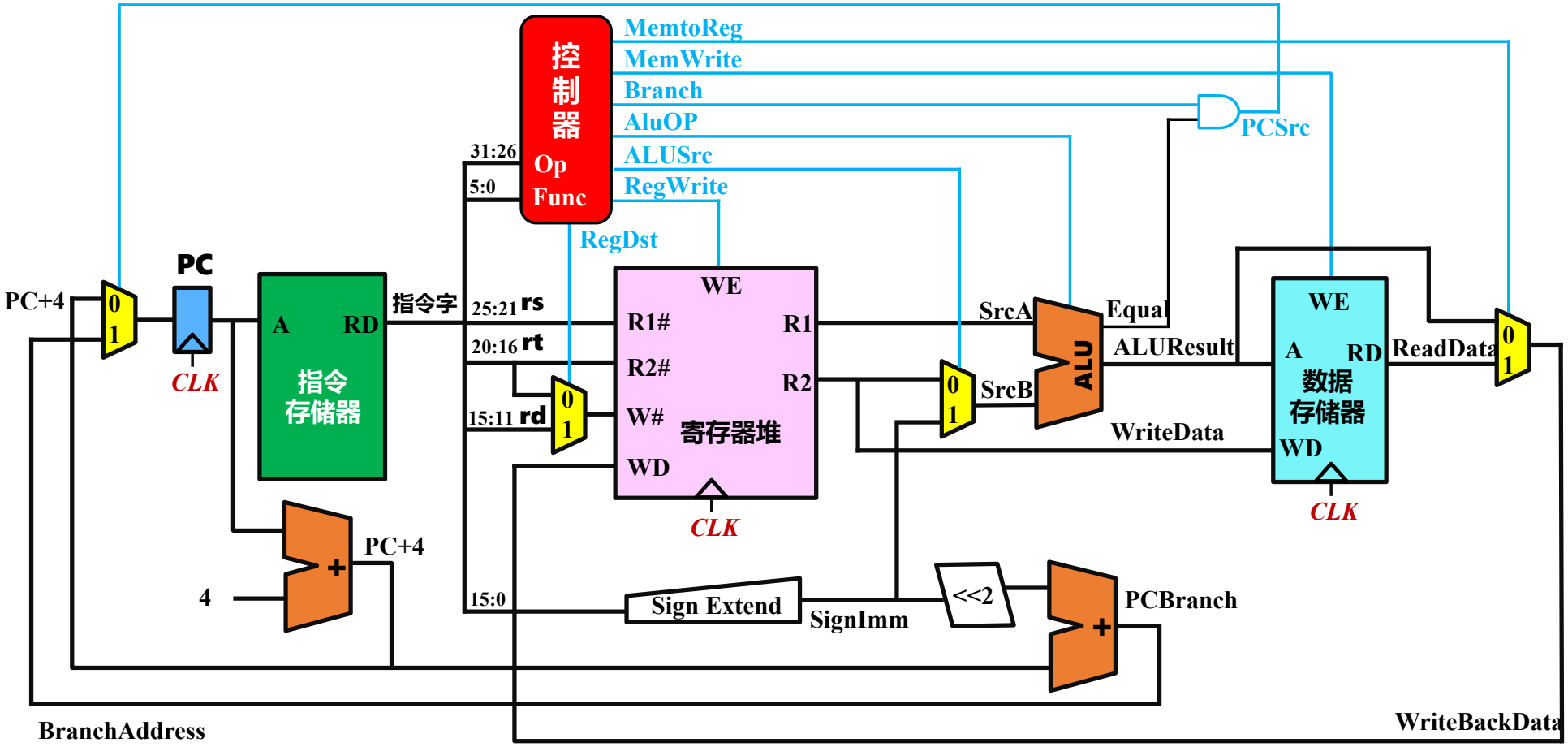
SW指令数据通路建立过程



Beq指令数据通路建立

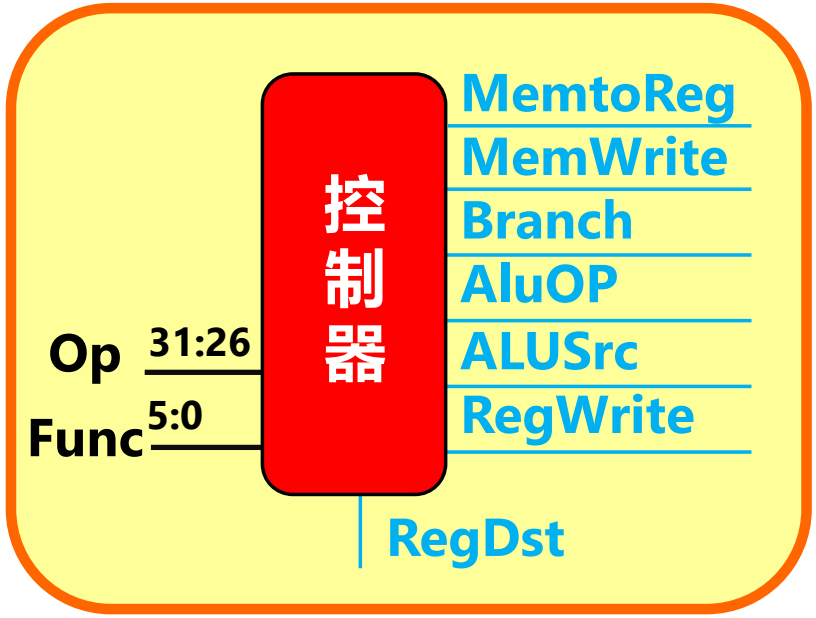


J型指令数据通路建立？


控制器
Op
Func
MemtoReg
MemWrite
Branch
AluOP
ALUSrc
RegWrite
RegDst
PCSrc
31:26
5:0
PC
PC+4
CLK
A
RD
指令存储器
指令字
25:21 rs
20:16 rt
15:11 rd
WE
R1#
R2#
W#
WD
R1
R2
寄存器堆
CLK
SrcA
SrcB
ALU
Equal
ALUResult
WriteData
WE
A
RD
数据存储器
WD
CLK
ReadData
PC+4
4
15:0
Sign Extend
SignImm
<<2
PCBranch
BranchAddress
WriteBackData

单周期MIPS控制器设计

- 输入信号
  - 指令字Opcode，Func字段（12位）
- 输出信号
  - 多路选择器选择信号
  - 寄存器写使能信号
  - 内存访问控制信号
  - 运算器控制信号，指令译码信号
- 纯组合逻辑电路，无时序逻辑



MIPS单周期CPU设计目的

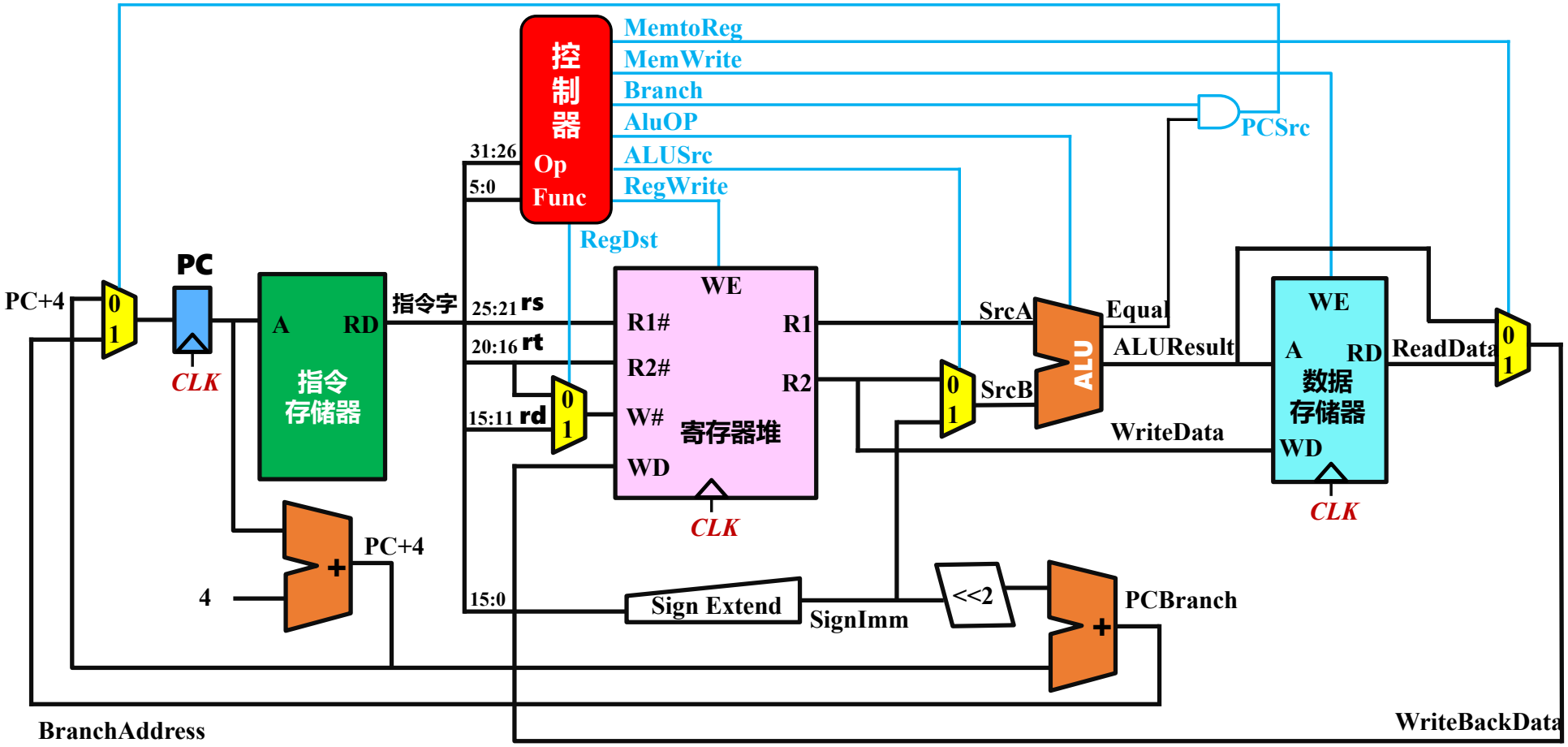
- 掌握硬布线控制器设计的基本原理
- 能利用相关原理在**Logisim**平台中设计实现**MIPS**单周期**CPU**

- 主要任务
  - 绘制MIPS CPU数据通路
  - 实现单周期硬布线控制器
  - 基于8条MIPS核心指令集完成冒泡排序测试程序设计
  - 测试联调与验证

核心指令集 （可实现内存区域冒泡排序）

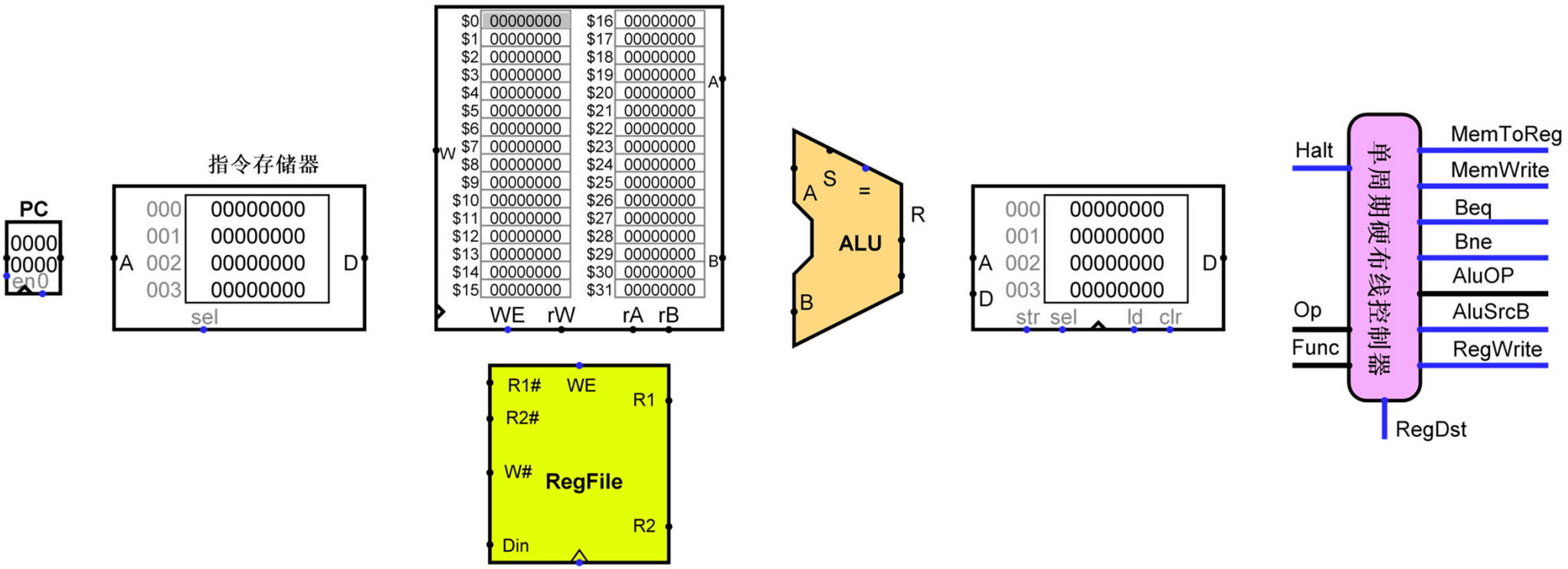
#	MIPS指令	RTL功能描述
1	add $rd,$rs,$rt	R[$rd]←R[$rs]+R[$rt]　溢出时产生异常，且不修改R[$rd]
2	slt $rd,$rs,$rt	R[$rd]←R[$rs]<R[$rt]　小于置1，有符号比较
3	addi $rt,$rs,imm	R[$rt]←R[$rs]+$SignExt_{16b}$(imm)　　溢出产生异常
4	lw $rt,imm($rs)	R[$rt]←$Mem_{4B}$(R[$rs]+$SignExt_{16b}$(imm))
5	sw $rt,imm($rs)	Mem_{4B}(R[$rs]+$SignExt_{16b}$(imm))←R[$rt]
6	beq $rs,$rt,imm	if(R[$rs] = R[$rt])　PC ← PC + $SignExt_{18b}$({imm, 00})
7	bne $rs,$rt,imm	if(R[$rs] != R[$rt]) PC ← PC + $SignExt_{18b}$({imm, 00})
8	syscall	系统调用，这里用于停机

单周期MIPS参考数据通路



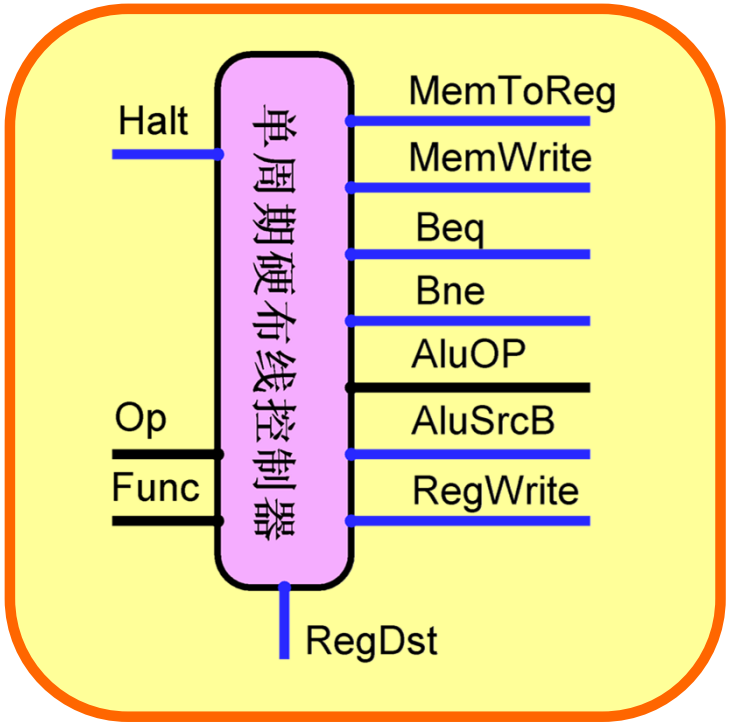
步骤1：构建MIPS主机通路

- 在MIPS单周期CPU子电路中，利用如下组件构建MIPS 单周期CPU数据通路
  - PC、IMEM、RegFile、ALU、DMEM、Controller



步骤2：设计单周期MIPS控制器

- **输入信号**
  - **指令字Opcode，Func字段（12位）**
- **输出信号**
  - 多路选择器选择信号
  - 内存访问控制信号
  - 寄存器写使能信号
  - 运算器控制信号、指令译码信号
- 纯组合逻辑电路、**无时序逻辑**

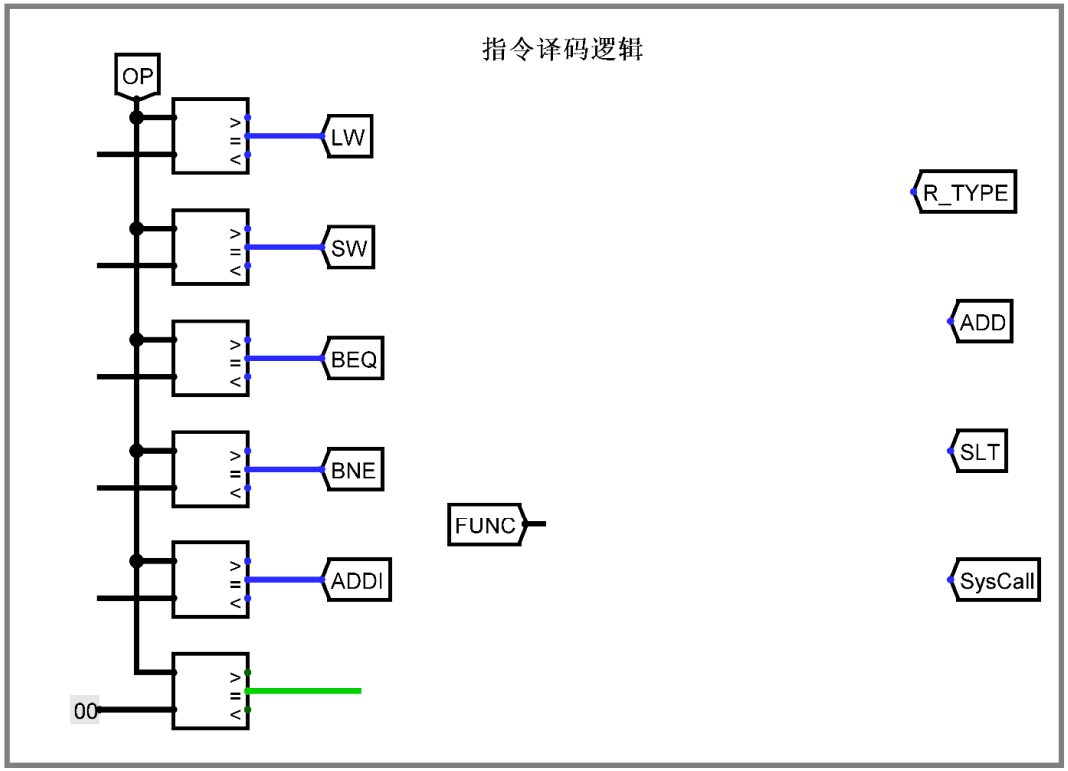


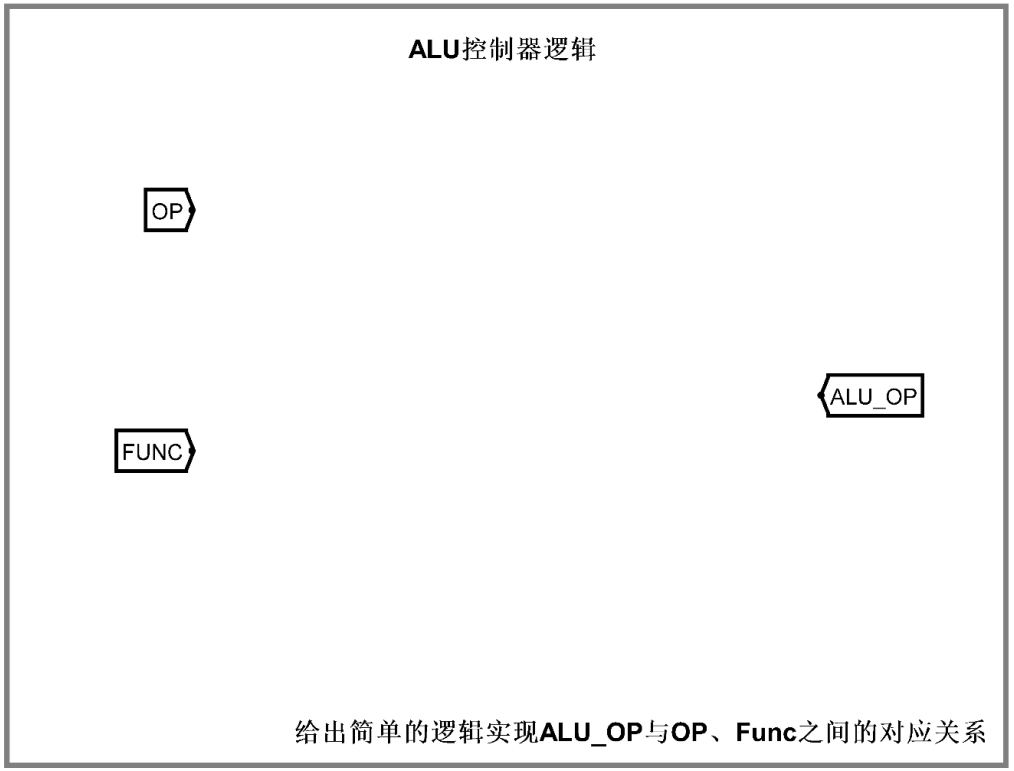
控制信号功能说明 （8条核心指令集）

#	控制信号	信号说明	产生条件
1	MemToReg	写入寄存器的数据来自存储器	lw指令
2	MemWrite	写内存控制信号	sw指令　未单独设置MemRead信号
3	Beq	Beq指令译码信号	Beq指令
4	Bne	Bne指令译码信号	Bne指令
5	AluOP	运算器操作控制符	加法，比较两种运算
6	AluSrcB	运算器第二输入选择	Lw指令，sw指令，addi
7	RegWrite	寄存器写使能控制信号	寄存器写回信号
8	RegDst	写入寄存器选择控制信号	R型指令
9	Halt	停机信号，取反后控制PC使能端	syscall指令

完善硬布线控制器内部逻辑

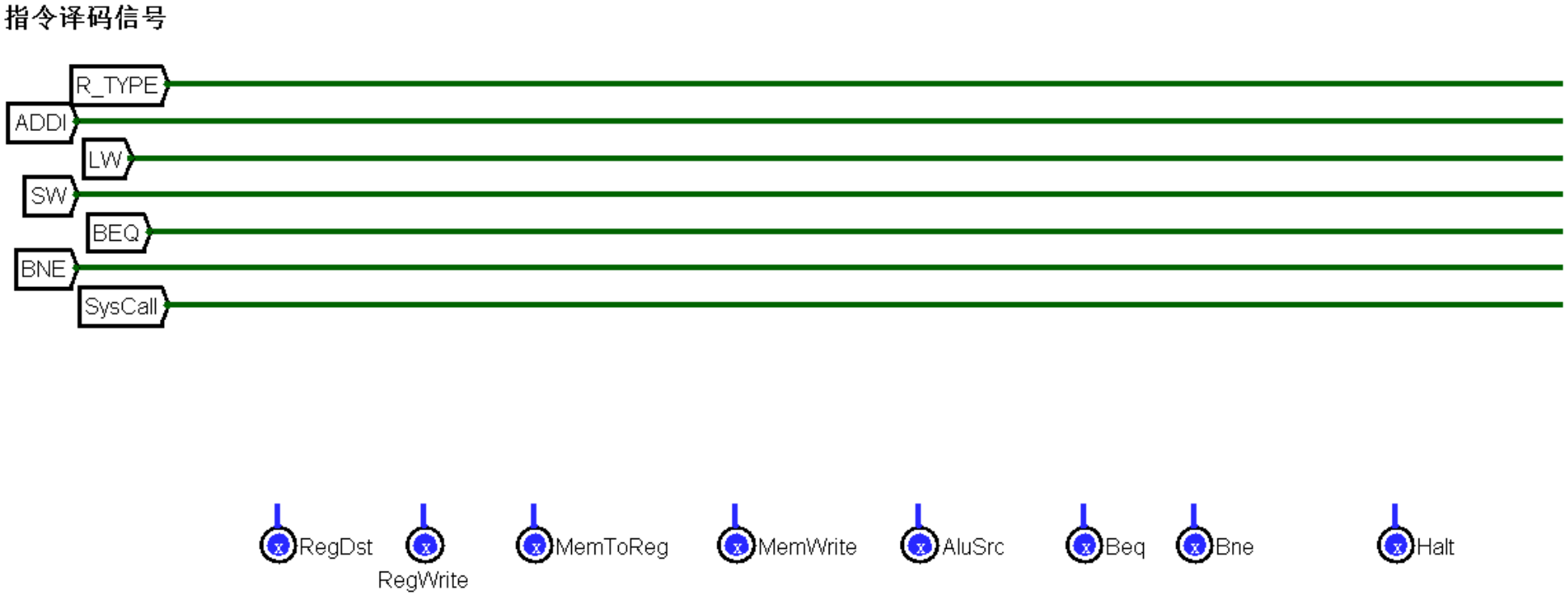
- 打开 CPU.circ　打开单周期硬布线控制器电路
- 实现 指令译码、ALU控制 逻辑





完善控制信号逻辑

- 增加简单的组合逻辑
- 根据给出的指令译码信号，实现所有控制信号逻辑

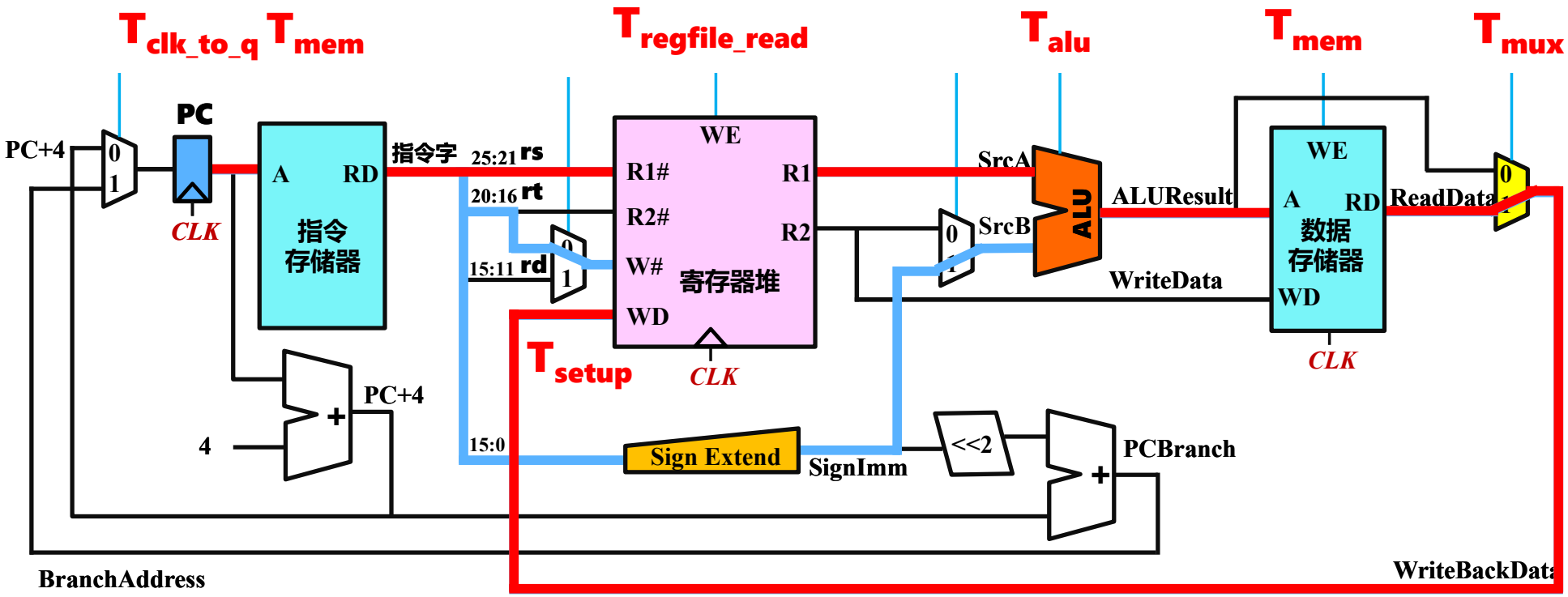


步骤3：CPU测试

- 阅读、理解基于8条MIPS指令编写的汇编排序程序sort.asm
- 对sort.asm进行编译，生成机器指令程序sort.hex， 并载入在指令存储器中
- 时钟自动仿真，Windows： Ctrl+k Mac: command+k 运行程序
- 程序停机后，查看数据存储器中排序情况，有符号降序排列
- 验收的差异化：要求改变排序数字的大小，升序排列

000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
010	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
020	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
030	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
040	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
050	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
060	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
070	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
080	00000006	00000005	00000004	00000003	00000002	00000001	00000000	ffffffff

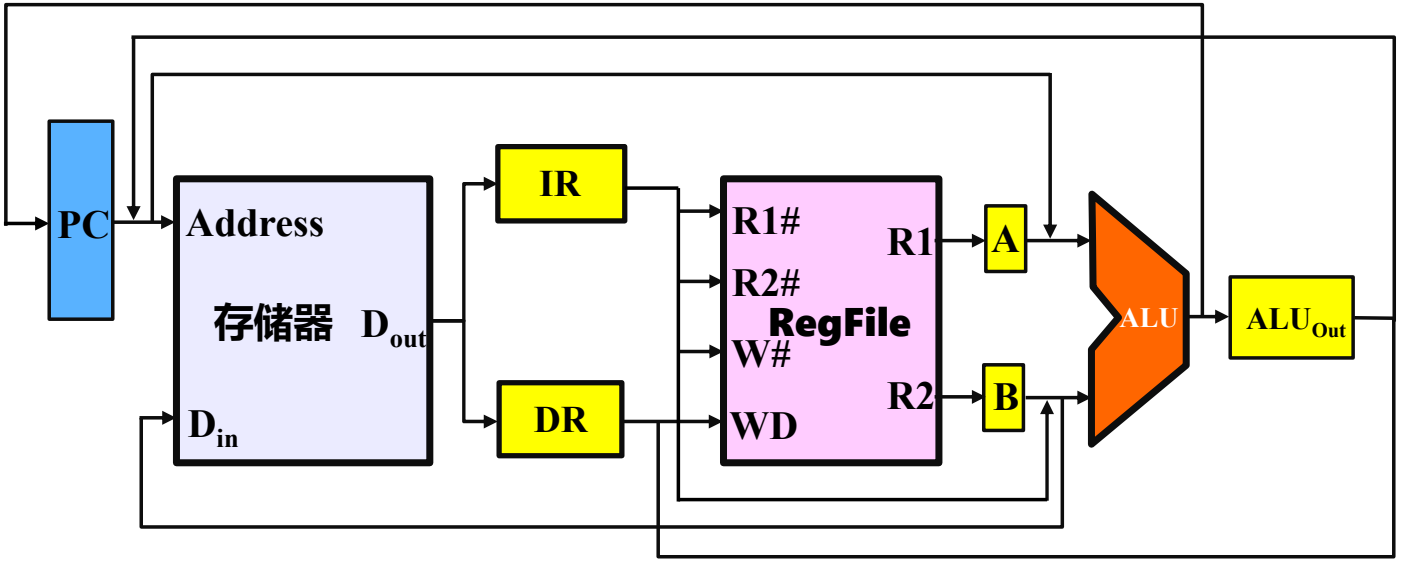
单周期MIPS关键路径---LW指令



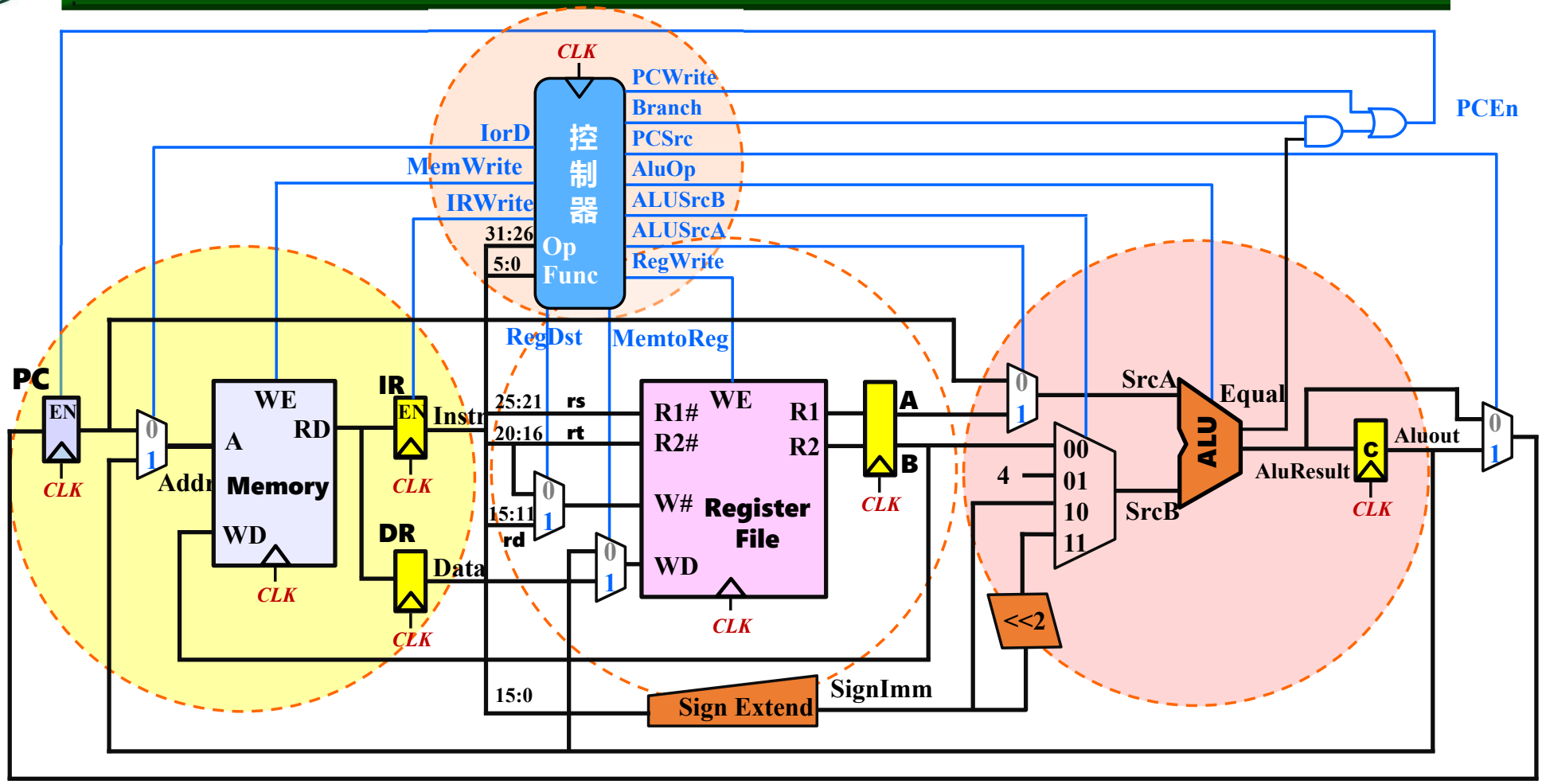
性能取决于最慢的指令，时钟周期过长

多周期MIPS数据通路特点

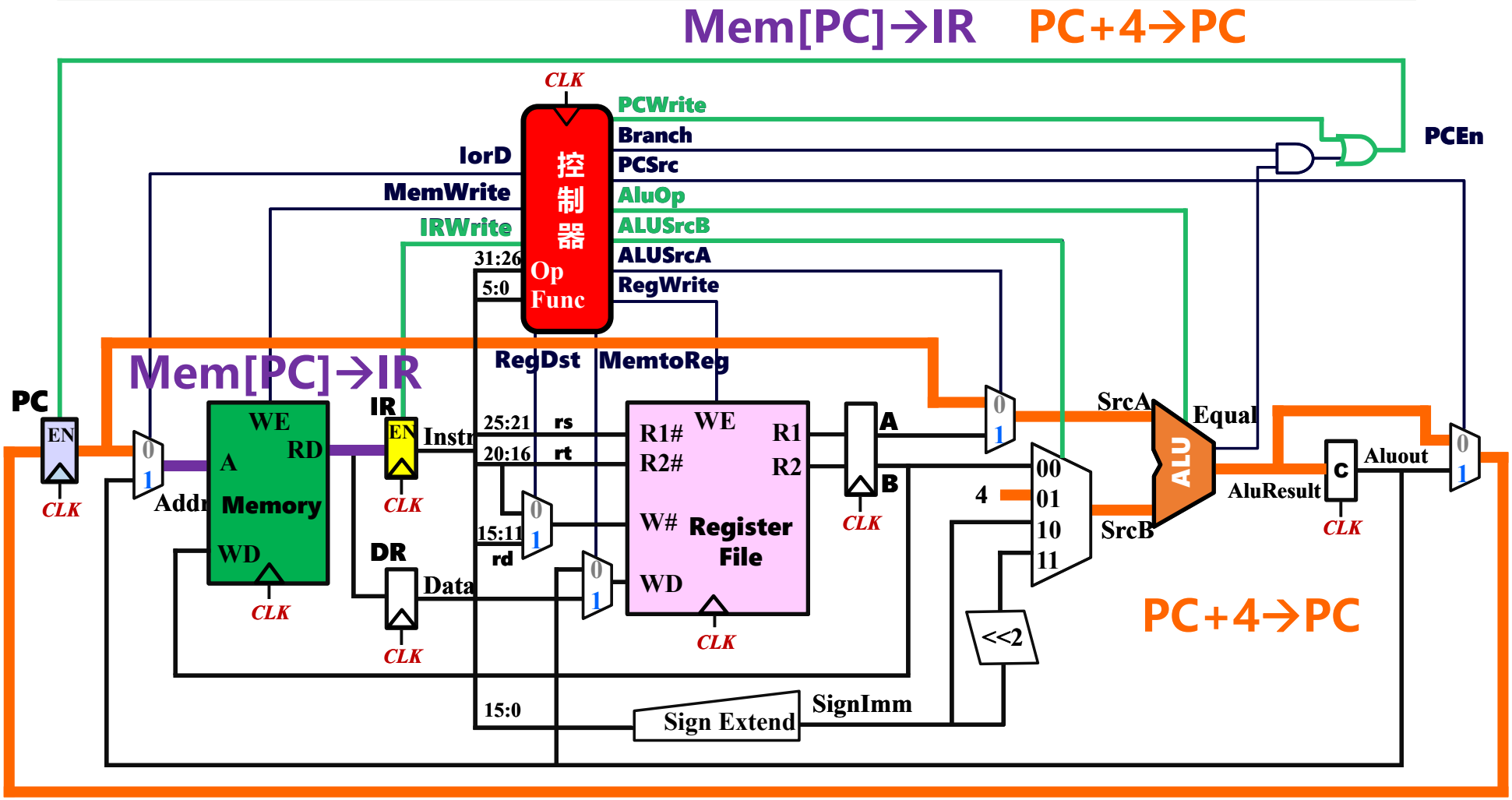
- 不再区分指令/数据存储器，分时使用功能部件
- 时钟周期变小、传输通路变短
- 功能部件输出端增加寄存器锁存数据



多周期MIPS CPU数据通路

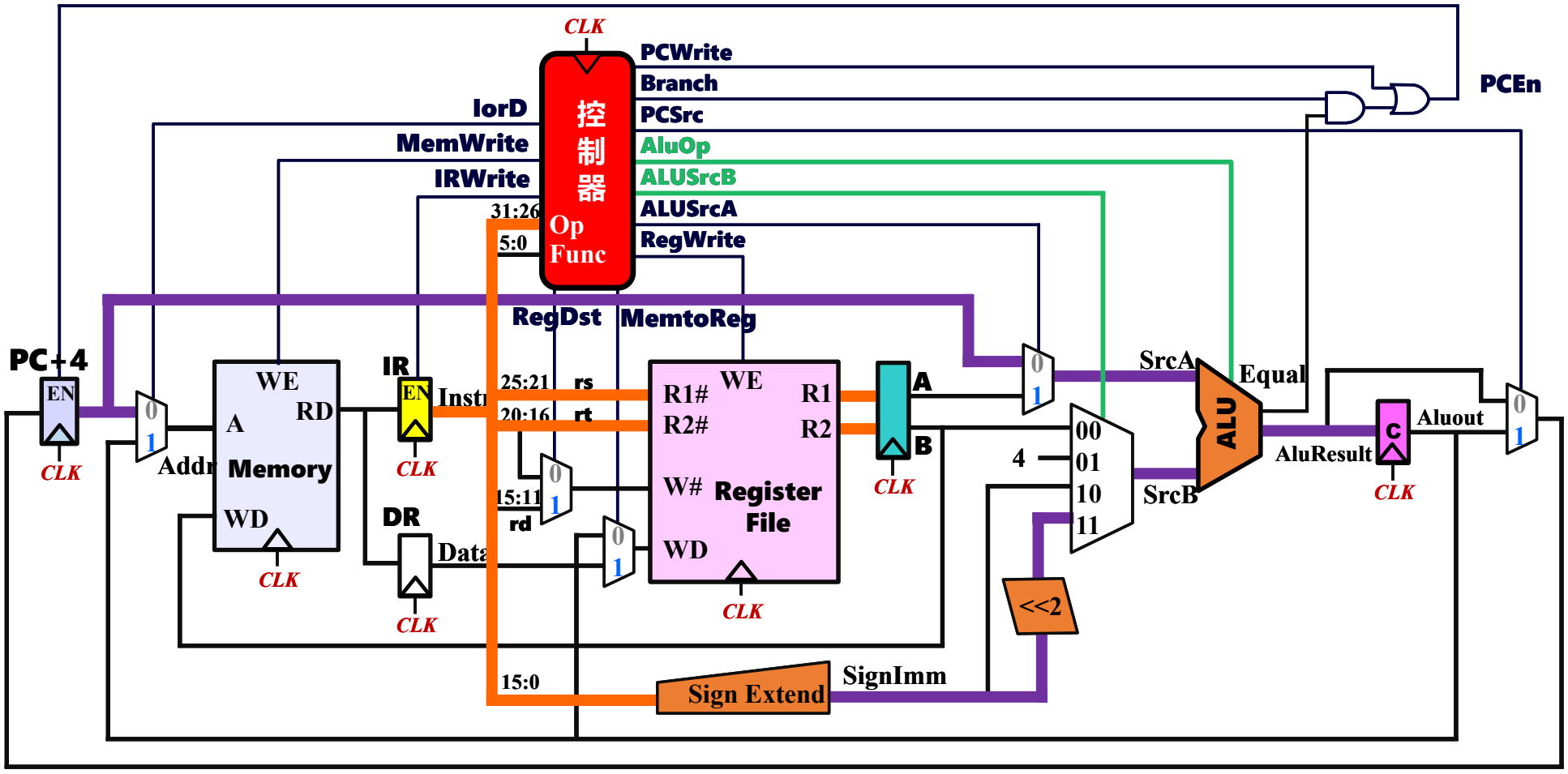


多周期MIPS取指令阶段T1

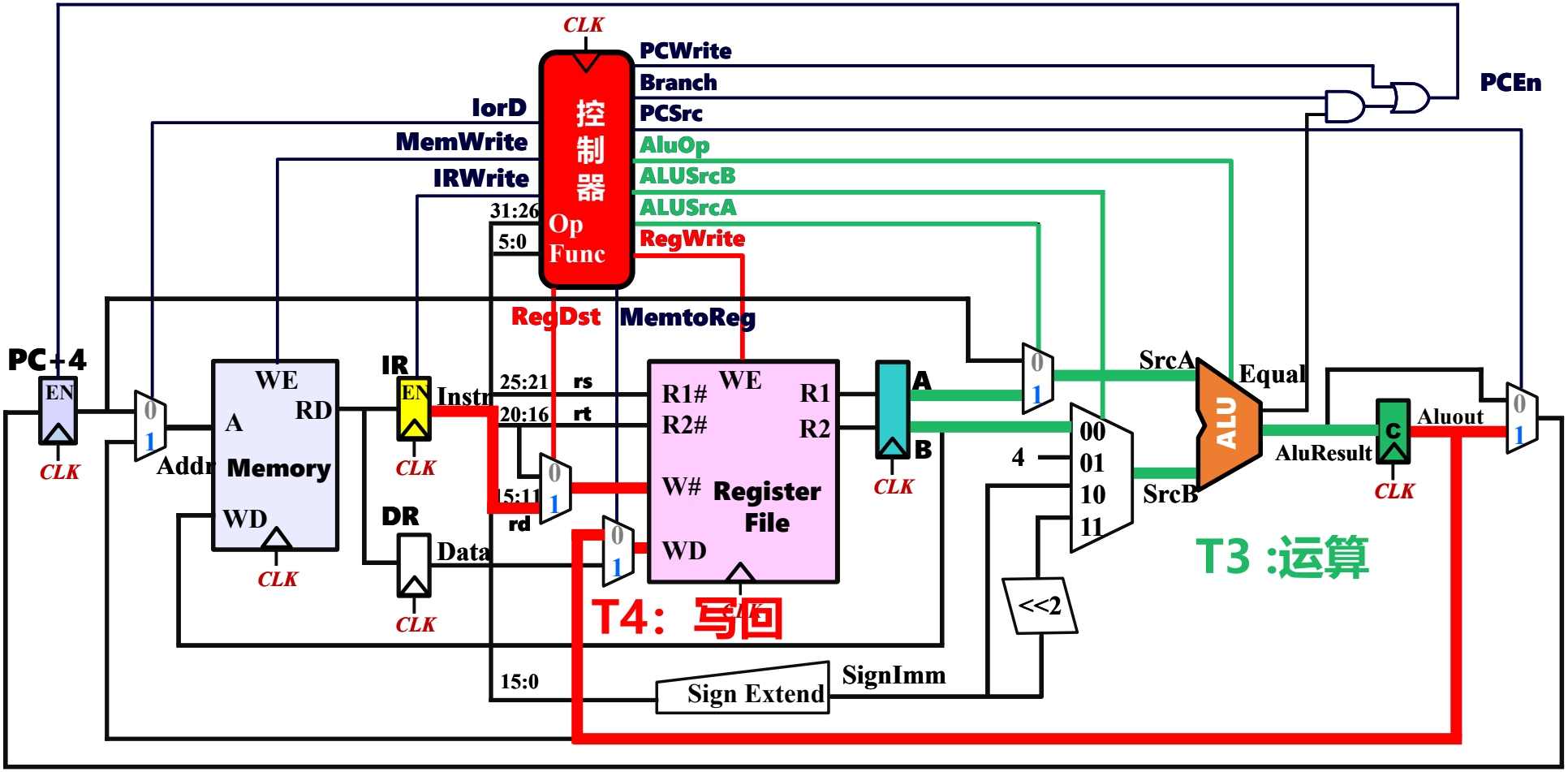


多周期MIPS取指令阶段T2

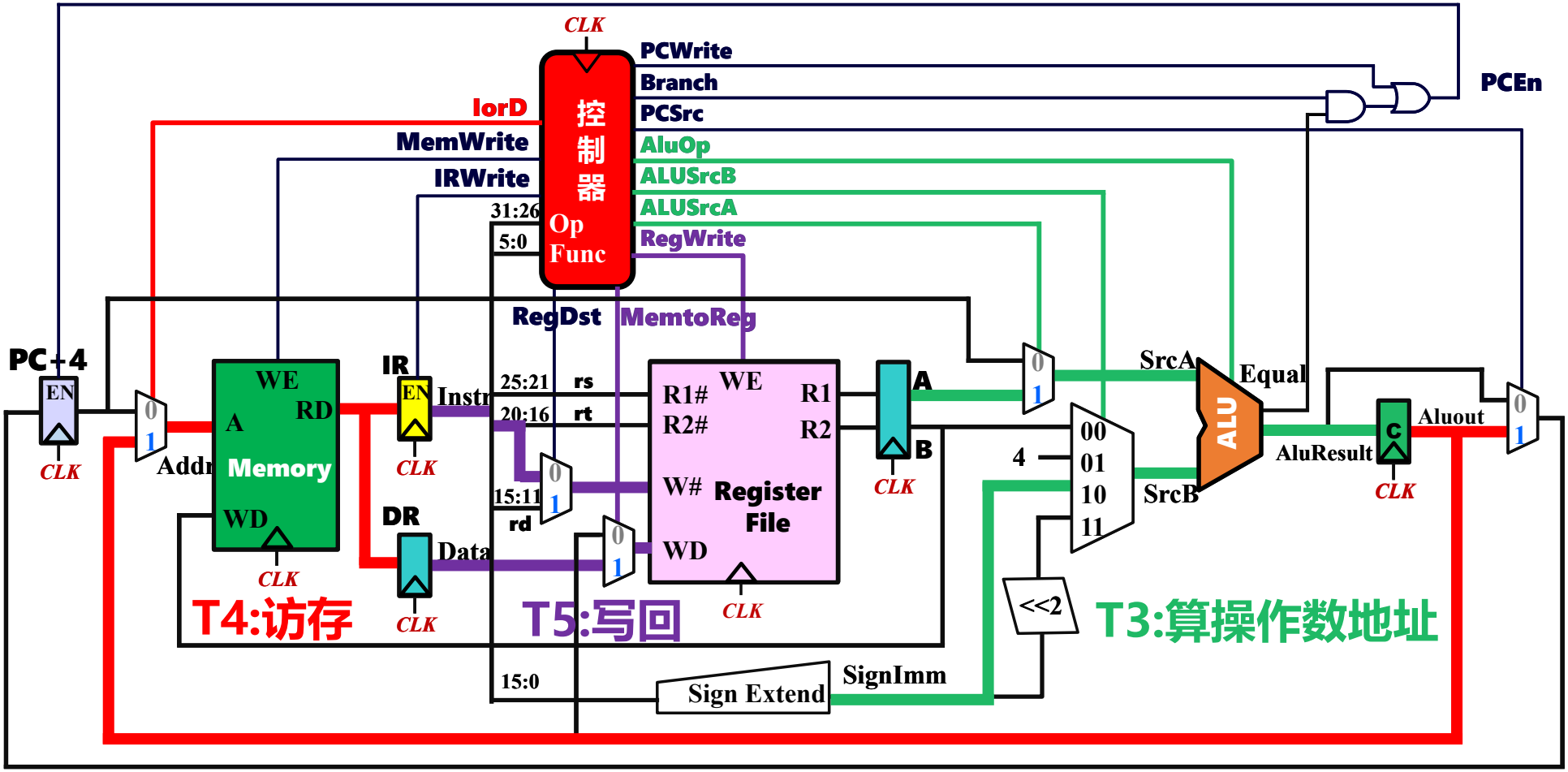
译码、Reg→A、B、PC+4+Imm16<<2→C



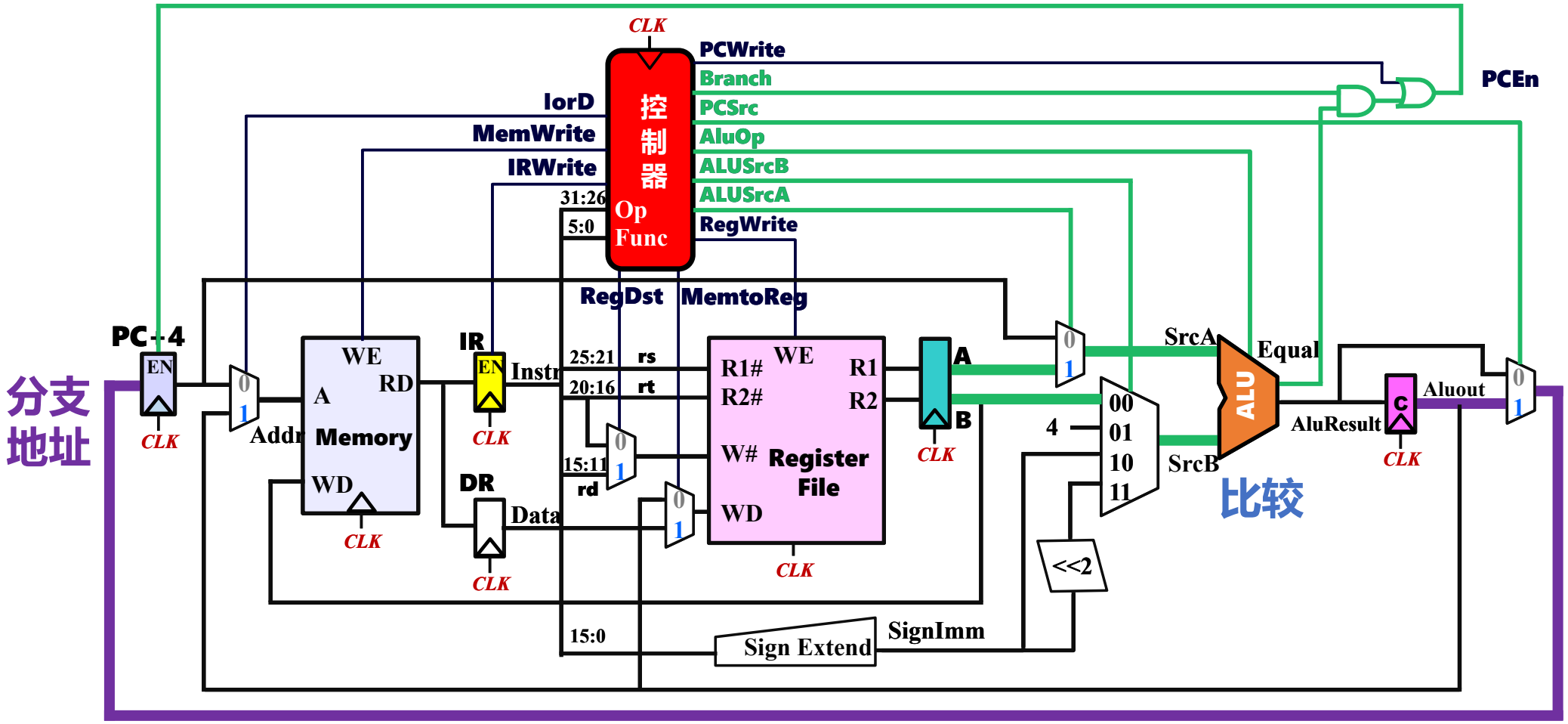
R型指令执行状态周期T3~T4



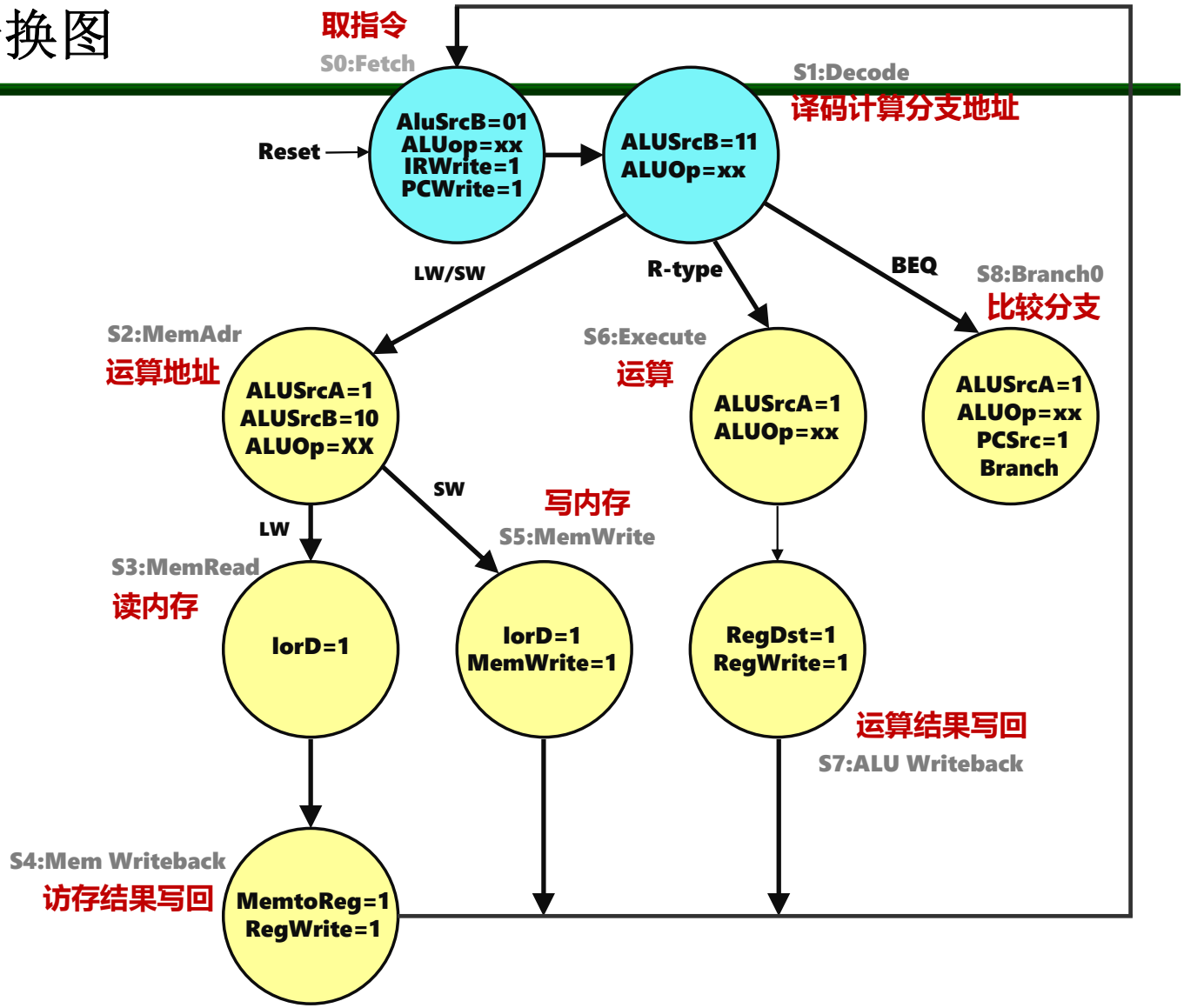
LW指令执行状态周期T3~T5



Beq指令执行状态周期T3



多周期状态转换图


取指令
S0:Fetch
AluSrcB=01
ALUop=xx
IRWrite=1
PCWrite=1
Reset
S1:Decode
译码计算分支地址
ALUSrcB=11
ALUOp=xx
LW/SW
R-type
BEQ
S8:Branch0
比较分支
ALUSrcA=1
ALUOp=xx
PCSrc=1
Branch
S2:MemAdr
运算地址
ALUSrcA=1
ALUSrcB=10
ALUOp=XX
S6:Execute
运算
ALUSrcA=1
ALUOp=xx
SW
写内存
S5:MemWrite
LW
S3:MemRead
读内存
IorD=1
IorD=1
MemWrite=1
RegDst=1
RegWrite=1
运算结果写回
S7:ALU Writeback
S4:Mem Writeback
访存结果写回
MemtoReg=1
RegWrite=1

MIPS多周期CPU微程序控制器设计实验目的

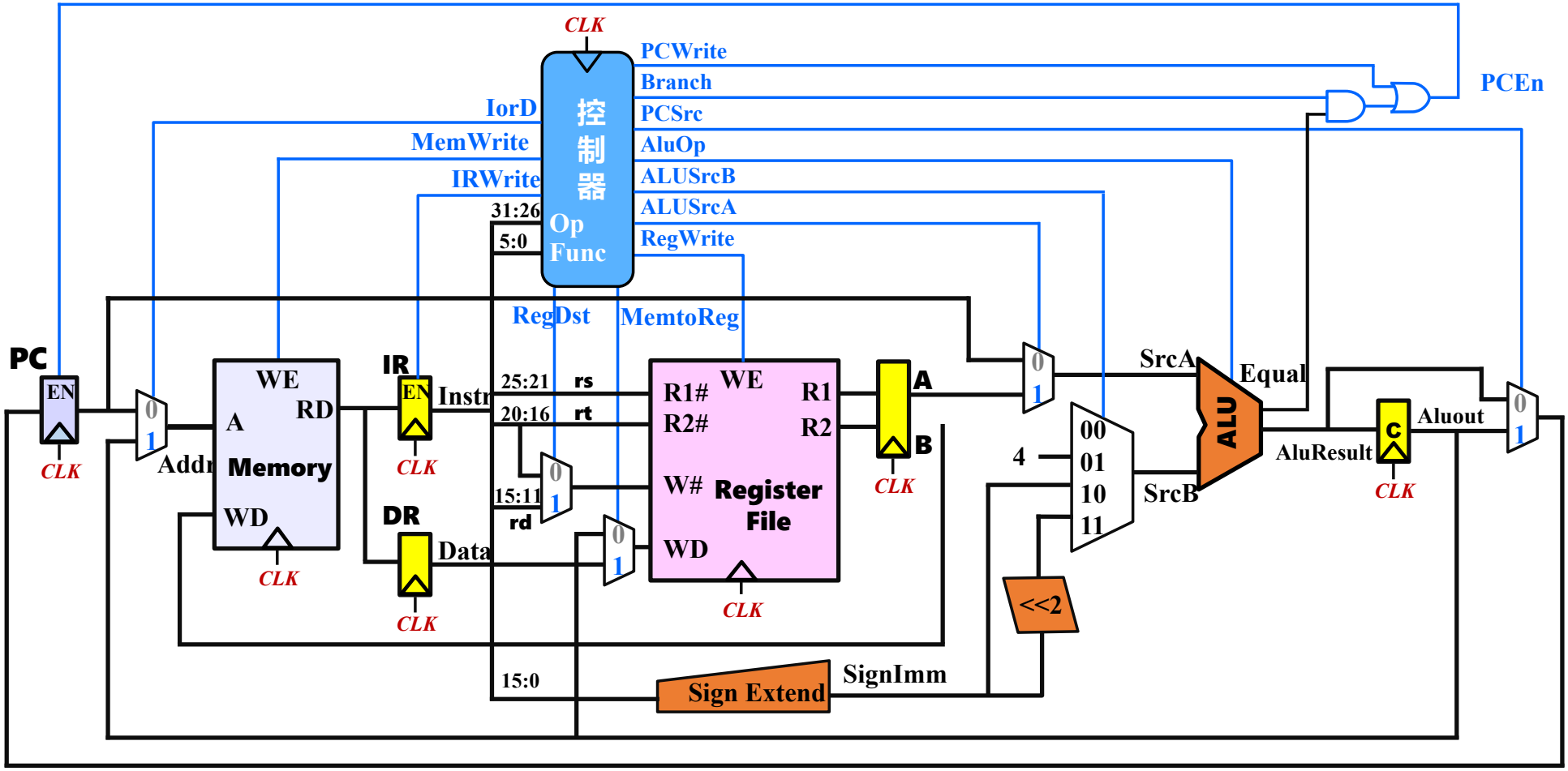
- 掌握多周期MIPS CPU设计原理
- 掌握微程序控制器设计的基本原理
- 利用微程序控制器的设计实现多周期MIPS处理器

- 主要任务
  - 绘制多周期MIPS CPU数据通路
  - 实现微程序控制器
  - 测试联调

核心指令集8条 （可实现内存区域冒泡排序）

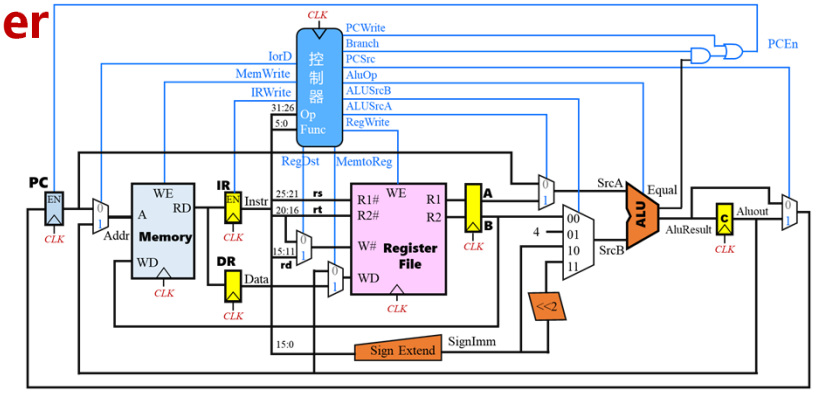
#	MIPS指令	RTL功能描述
1	add $rd,$rs,$rt	R[$rd]←R[$rs]+R[$rt]　溢出时产生异常，且不修改R[$rd]
2	slt $rd,$rs,$rt	R[$rd]←R[$rs]<R[$rt]　小于置1，有符号比较
3	addi $rt,$rs,imm	R[$rt]←R[$rs]+$SignExt_{16b}$(imm)　　溢出产生异常
4	lw $rt,imm($rs)	R[$rt]←$Mem_{4B}$(R[$rs]+$SignExt_{16b}$(imm))
5	sw $rt,imm($rs)	Mem_{4B}(R[$rs]+$SignExt_{16b}$(imm))←R[$rt]
6	beq $rs,$rt,imm	if(R[$rs] = R[$rt]) PC ← PC + $SignExt_{18b}$({imm, 00})
7	bne $rs,$rt,imm	if(R[$rs] != R[$rt]) PC ← PC + $SignExt_{18b}$({imm, 00})
8	syscall	系统调用，这里用于停机

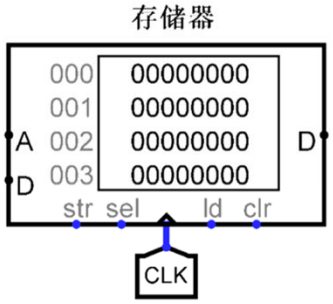
多周期MIPS CPU数据通路参考

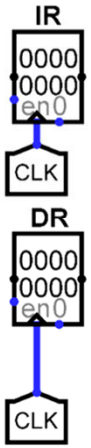


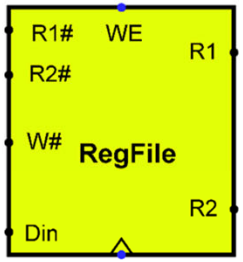
步骤1：构建多周期MIPS CPU数据通路

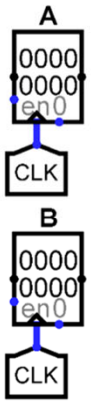
- 在MIPS多周期CPU（微程序）子电路中，利用如下组件构建CPU数据通路
  - PC、MEM、IR、DR、RegFile、ALU、Controller

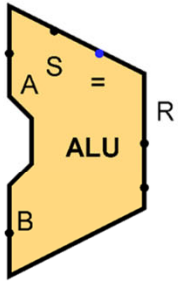


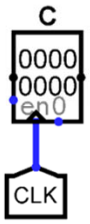


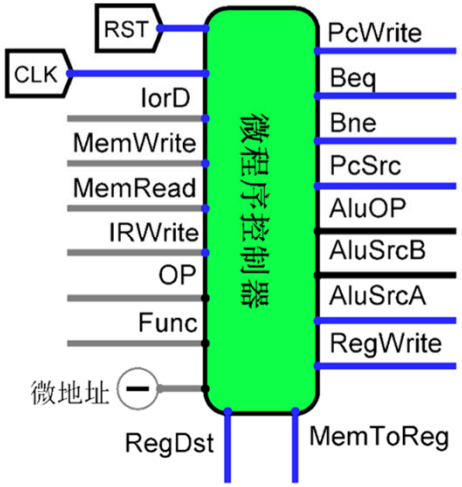






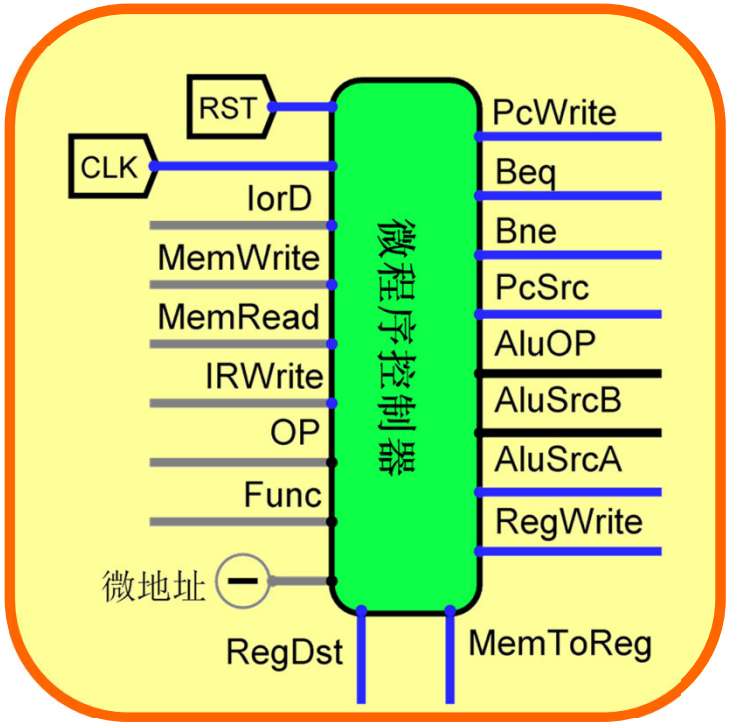






步骤2：设计微程序控制器

- 输入信号
  - 指令字Opcode，Func字段（12位）
  - 时钟信号、复位信号
- 输出信号
  - 多路选择器选择信号
  - 内存访问控制信号
  - 寄存器写使能信号
  - 运算器控制信号、指令译码信号

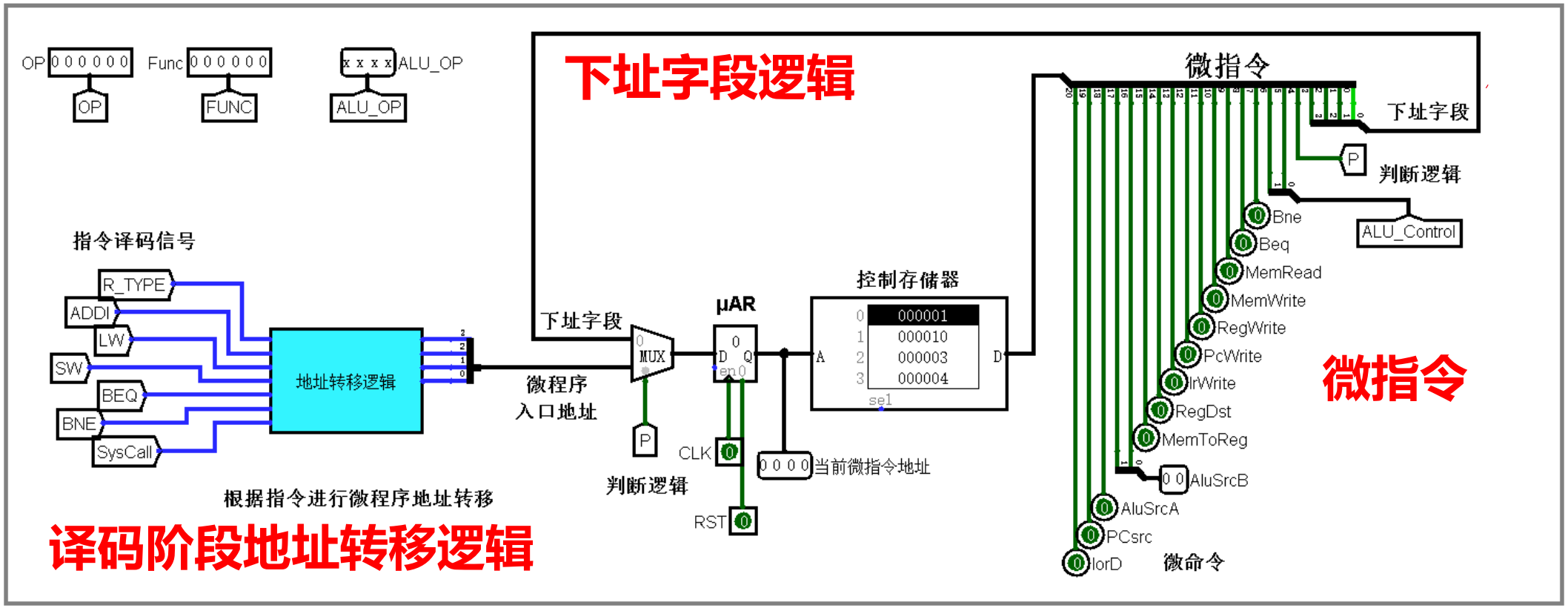


控制信号功能说明 （8条核心指令集）

#	控制信号	信号说明	产生条件
1	PCWrite	PC写使能控制	取指令周期，分支指令执行
2	IorD	指令还是数据	0表示指令，1表示数据
3	IRwrite	指令寄存器写使能	高电平有效
4	MemWrite	写内存控制信号	sw指令
5	MemRead	读内存控制信号	lw指令 取指令
6	Beq	Beq指令译码信号	Beq指令
7	Bne	Bne指令译码信号	Bne指令
8	PcSrc	PC输入来源	顺序寻址还是跳跃寻址
9	AluOP	运算器操作控制符 4位	ALU_Control控制，00加，01减，10由Funct定
10	AluSrcA	运算器第一输入选择	
11	AluSrcB	运算器第二输入选择	Lw指令，sw指令，addi
12	RegWrite	寄存器写使能控制信号	寄存器写回信号
13	RegDst	写入寄存器选择控制信号	R型指令
14	MemToReg	写入寄存器的数据来自存储器	lw指令

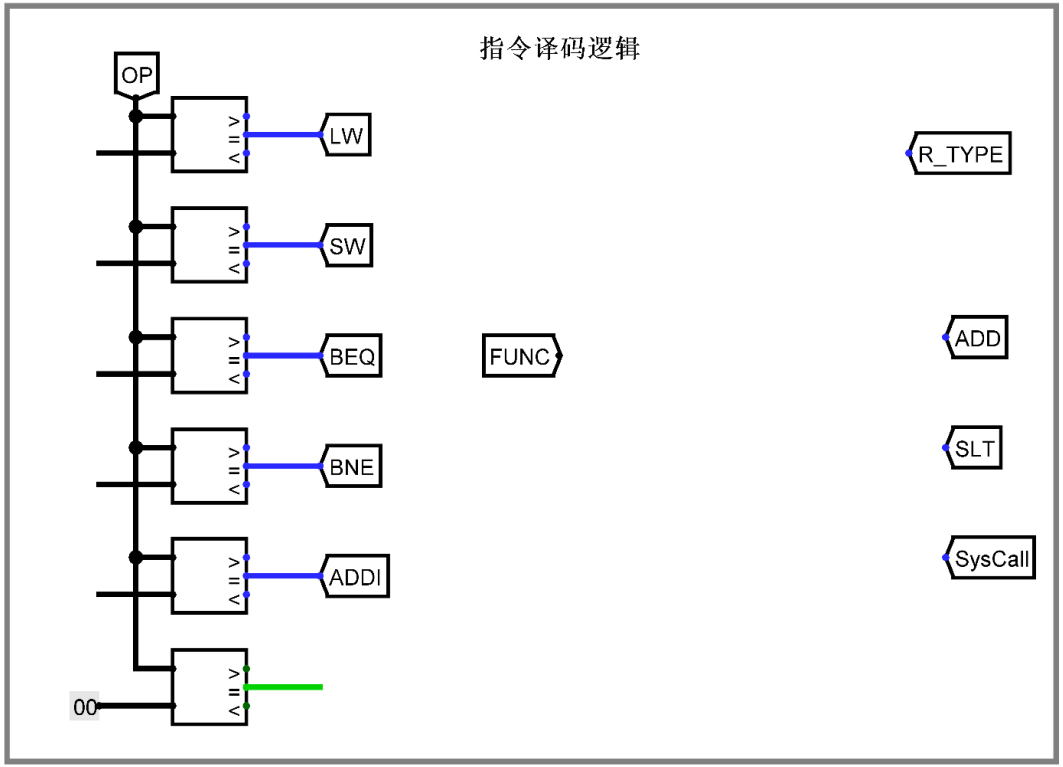
微程序控制器内部架构

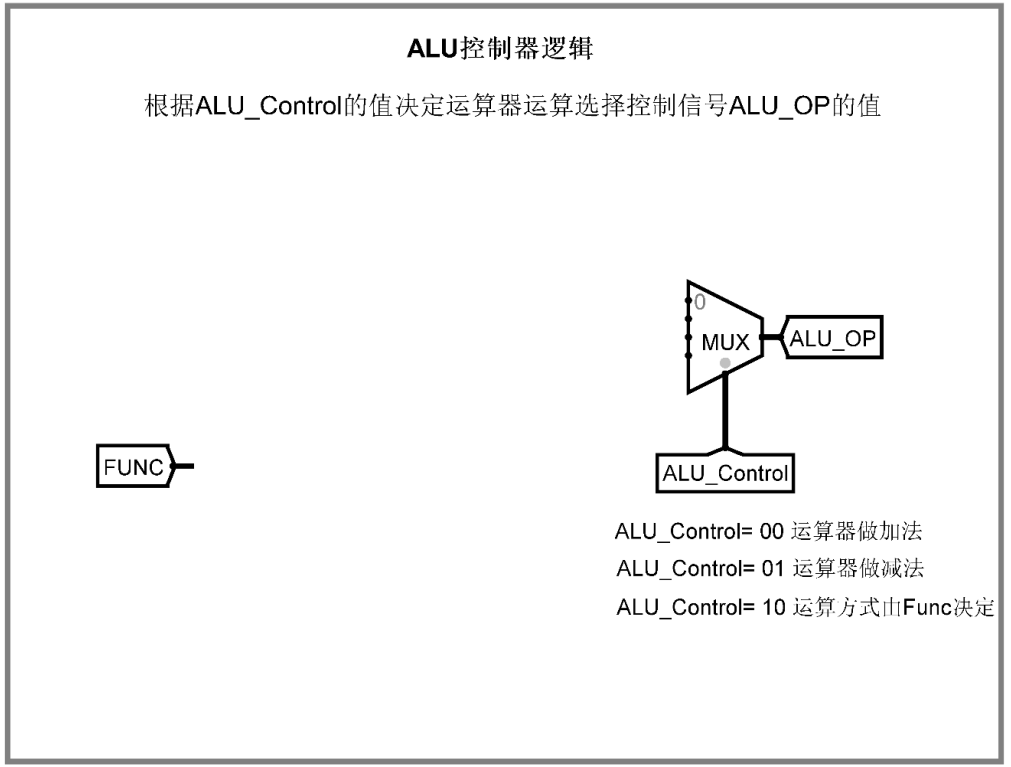
- **载入微程序，设计地址转移逻辑**



1.完善控制器内部逻辑

- 打开 CPU.circ　打开多周期微程序控制器电路
- 首先完成如下电路逻辑：　指令译码、ALU控制





2.实现微程序地址转移逻辑

- 填写微程序地址入口表 （Excel表 微程序地址转移逻辑自动生成.xlsx ）
  - 左侧是译码信号，右侧是对应类型指令对应的入口地址，地址转移逻辑会自动生成
- 自动生成 微程序地址转移逻辑 子电路
  - Logisim的分析电路功能 Project → Analyze Circuit

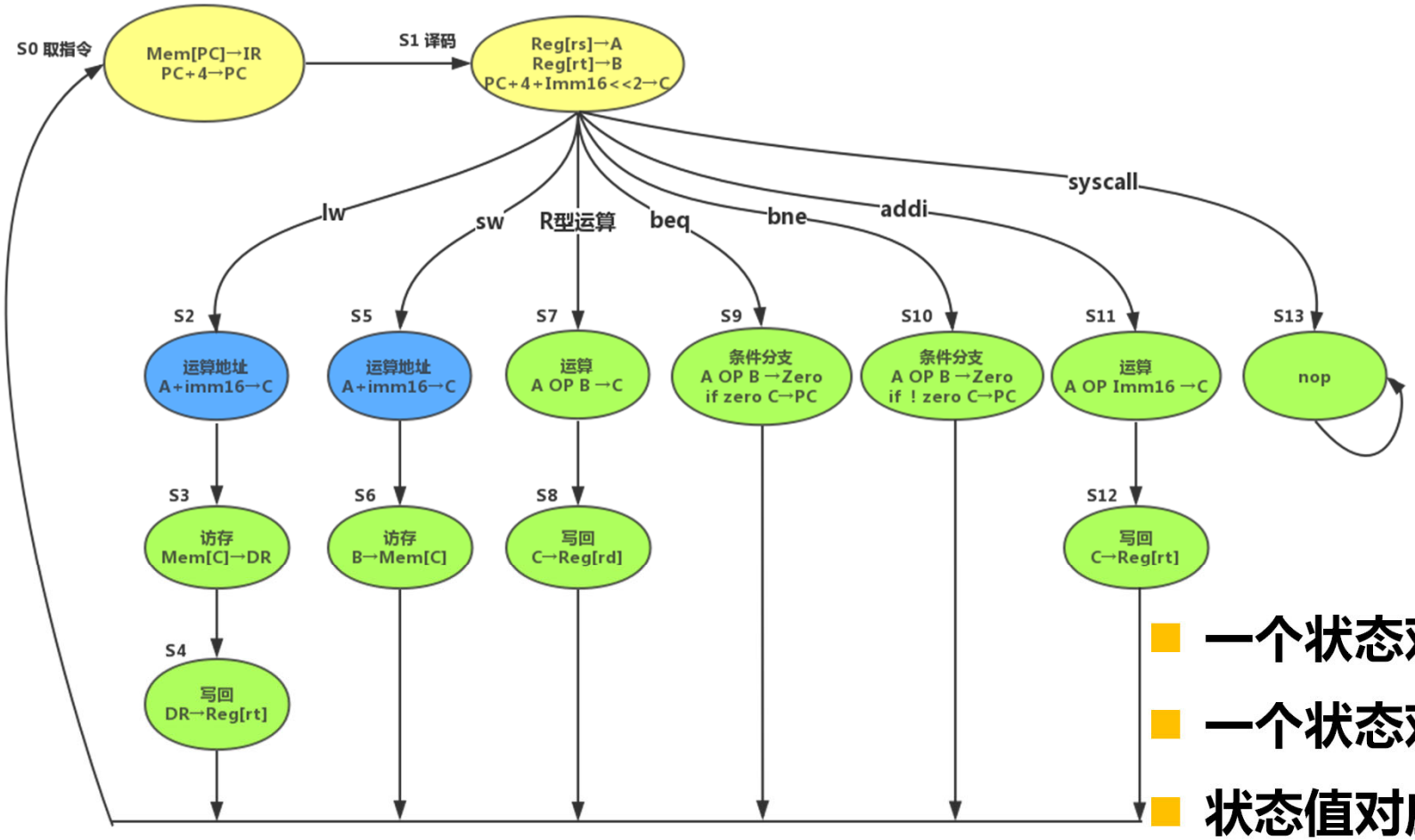
R_Type	ADDI	LW	SW	BEQ	BNE	SYSCALL	最小项表达式	S3	S2	S1	S0
R_Type&							R_Type		R_Type+	R_Type+	R_Type+
	ADDI&						ADDI	ADDI+		ADDI+	ADDI+
		LW&					LW	LW+			LW+
			SW&				SW				
				BEQ&			BEQ				
					BNE&		BNE				
						SYSCALL&	SYSCALL				
								ADDI+LW	R Type	R_Type+ADDI	R_Type+ADDI+LW

微程序地址入口表 地址逻辑自动生成

微程序入口地址示例

	A	B	C	D	E	F	G	I	J	K	L	M
1	机器指令译码信号							微程序入口地址				
2	R_Type	ADDI	LW	SW	BEQ	BNE	SYSCALL	入口地址10进制	S3	S2	S1	S0
3	1						0	7	0	1	1	1
4		1						11	1	0	1	1
5			1					2	0	0	1	0
6				1				5	0	1	0	1
7					1			9	1	0	0	1
8						1		10	1	0	1	0
9							1	13	1	1	0	1

构建指令状态变换图



- 一个状态对应一个时钟周期
- 一个状态对应一条微指令
- 状态值对应微指令地址

3.根据状态图构建微程序

- 状态值 → 微指令地址
- 不同状态 → 微控制信号、P字段设置、下址字段 → 微指令 → 微程序

微指令、微程序自动生成　Excel表格

微指令功能	状态	微指令地址	IorD	PcSrc	AluSrcA	AluSrcB	MemToReg	RegDst	BEQ	BNE	AluControl	P	下址字段	微指令	十六进制
取指令	0	0000	0	0	0	01	0	0	0	0	00	0	0001	00001001100100000001	13201
译码	1	0001	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000	0
LW1	2	0010	00	0	0	00	0	0	0	0	00	0	0000	00000000000000000000	0
LW2	3	0011	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000	0
LW3	4	0100	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000	0
SW1	5	0101	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000	0
SW2	6	0110	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000	0
	7	0111	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000	0
	8	1000	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000	0
	9	1001	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000	0
	10	1010	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000	0
	11	1011	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000	0
	12	1100	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000	0
	13	1101	1	0	0	00	0	0	0	0	11	0	0000	10000000000001100000	100060

微程序指令示例

微指令功能	状态	微指令地址	IorD	PcSrc	AluSrcA	AluSrcB	MemToReg	RegDst	IrWrite	PcWrite	RegWrite	MemWrite	MemRead	BEQ	BNE	AluControl	P	下址字段	微指令	十六进制
取指令	0	0000	0	0	0	01	0	0	1	1	0	0	1	0	0	00	0	0001	00001001100100000001	13201
译码	1	0001	0	0	0	11	0	0	0	0	0	0	0	0	0	00	1	0000	00011000000000010000	30010
LW1	2	0010	00	0	1	10	0	0	0	0	0	0	0	0	0	00	0	0011	00011000000000000011	60003
LW2	3	0011	1	0	0	00	0	0	0	0	0	0	1	0	0	00	0	0100	10000000001000000100	100204
LW3	4	0100	0	0	0	00	1	0	0	0	1	0	0	0	0	00	0	0000	00000100010000000000	8800
SW1	5	0101	0	0	1	10	0	0	0	0	0	0	0	0	0	00	0	0110	00110000000000000110	60006
SW2	6	0110	1	0	0	00	0	0	0	0	0	1	0	0	0	00	0	0000	10000000001000000000	100400
R_TYPE1	7	0111	0	0	1	00	0	0	0	0	0	0	0	0	0	10	0	1000	00100000000001001000	40048
R_TYPE2	8	1000	0	0	0	00	0	1	0	0	1	0	0	0	0	00	0	0000	00000010010000000000	4800
BEQ	9	1001	0	1	1	00	0	0	0	0	0	0	0	1	0	00	0	0000	01100000000010000000	C0100
BNE	10	1010	0	1	1	00	0	0	0	0	0	0	0	0	1	00	0	0000	01100000000001000000	C0080
ADDI1	11	1011	0	0	1	10	0	0	0	0	0	0	0	0	0	00	0	1100	00110000000000001100	6000C
ADDI2	12	1100	0	0	0	00	0	0	0	0	1	0	0	0	0	00	0	0000	00000000100000000000	800
SysCall	13	1101	1	0	0	00	0	0	0	0	0	0	0	0	0	11	0	1101	10000000000001101101	10006D

第1步： 在第1列安排微程序，通常取指令部分放置在0号单元，同一指令的微程序中的微指

第2步： 填写D到S列的微指令控制信号，注意其中aluSrcB，AluControl为2位，下地址字段

第3步： 完成第2步后，最后一列微指令16进制会自动更新

第3步： 将最后一列的16进制编码复制粘贴到Logisim中的控制存储器中

微指令十六进制编码直接复制粘贴到控存中

步骤3：CPU测试

- 在存储器中载入排序程序sort.hex
- 时钟自动仿真，Windows： Ctrl+k Mac: command+k 运行程序
- 程序停机后，查看数据存储器中排序情况，有符号降序排列

000	2010ffff	20110000	ae300200	22100001	22310004	ae300200	22100001	22310004
010	22310004	ae300200	22100001	22310004	ae300200	22100001	22310004	ae300200
020	2231fffc	1611fff8	22100004	2011001c	1611fff5	2002000a	0000000c	00000000
030	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
040	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
050	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
060	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
070	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
080	00000006	00000005	00000004	00000003	00000002	00000001	00000000	ffffffff