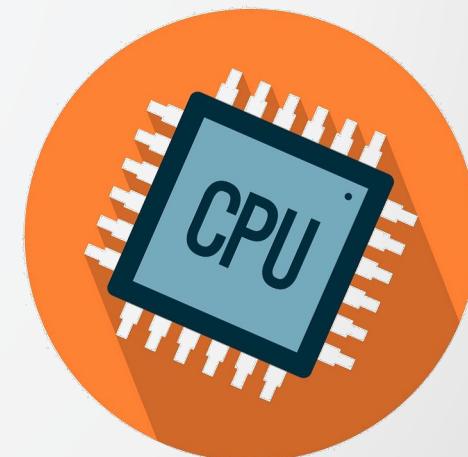




计算机组成原理



实验、MIPS CPU设计



||| CPU设计实验

1

实验目标与任务

- 学生理解单周期**MIPS CPU**基本原理
 - 能在**Logisim**平台中设计实现单周期**MIPS CPU**
 - 8条核心指令或更多，能运行冒泡排序测试程序
- 学生理解**MIPS多周期处理器**的基本原理，
 - 能在**Logisim**平台中设计实现**MIPS 多周期CPU**
 - 硬布线控制器，微程序控制器两种方案
 - 8条核心指令

2

核心指令集

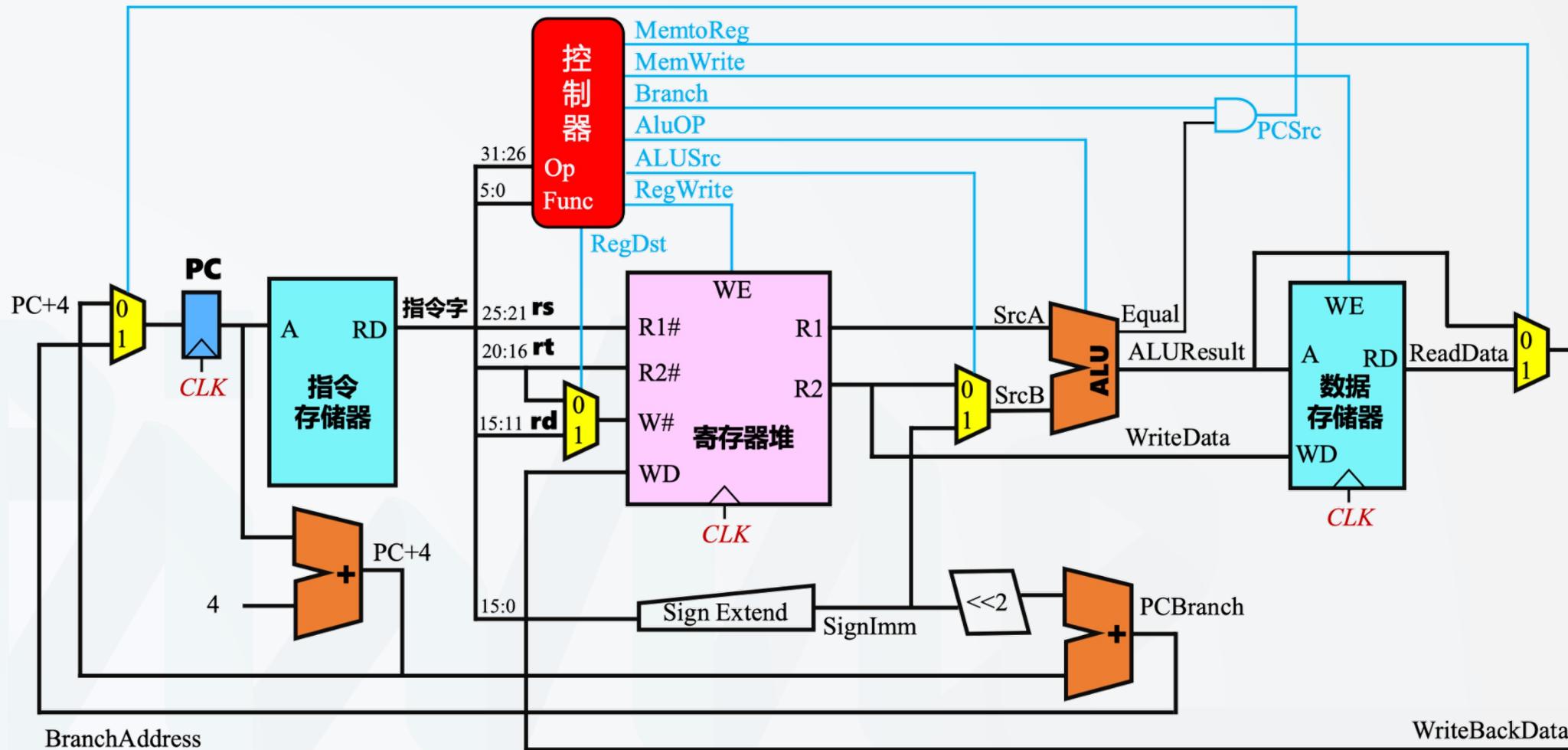
可实现内存区域冒泡排序

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

CPU设计实验

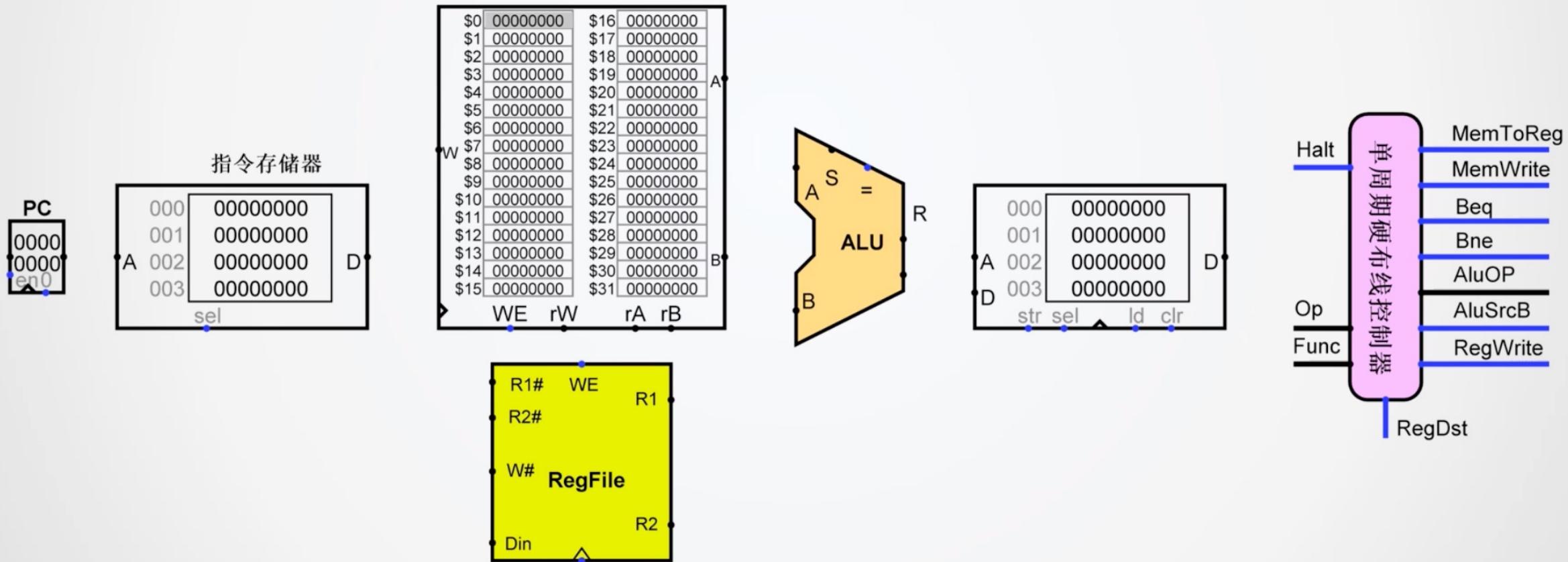
3

构建单周期MIPS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



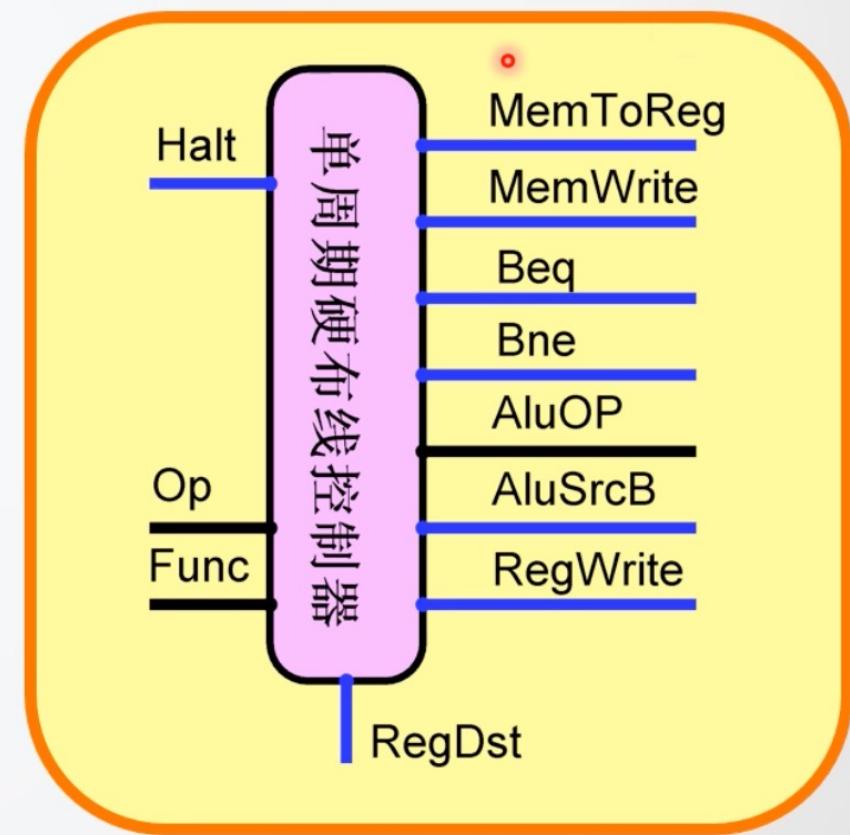
步骤2：设计单周期MIPS控制器

■ 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

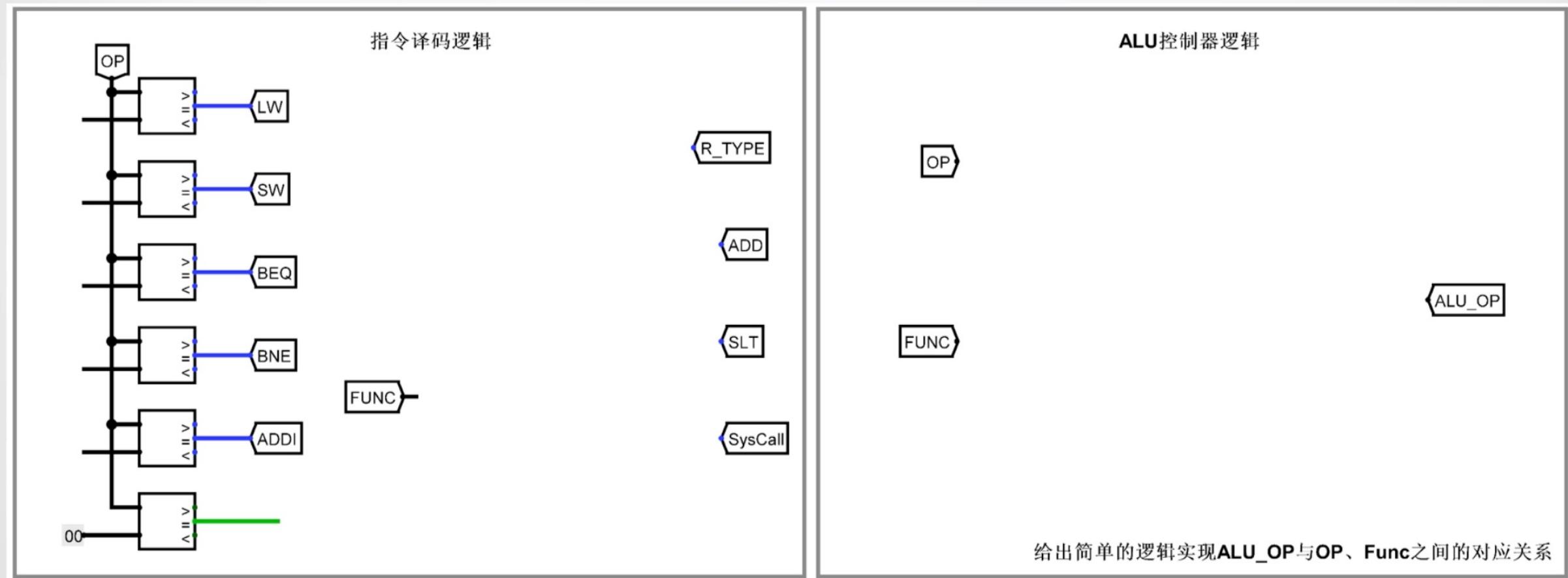


||控制信号功能说明（8条核心指令集）

#	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

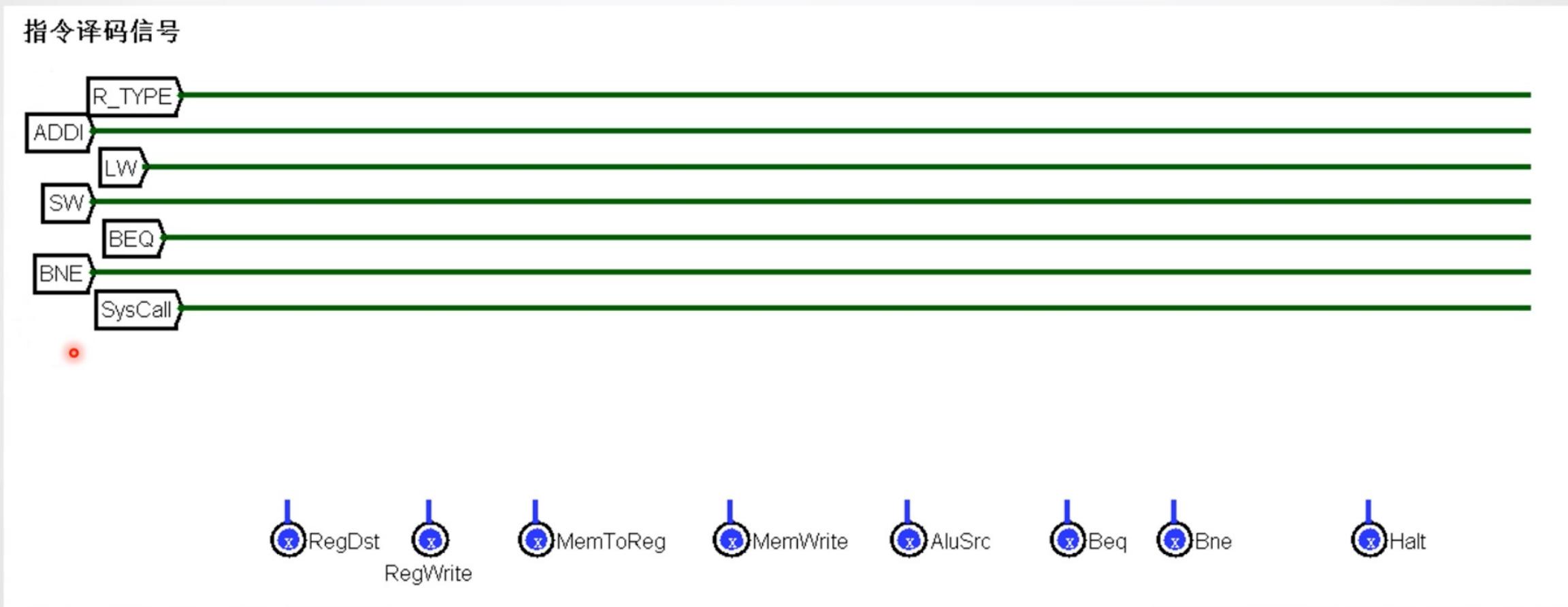
完善硬布线控制器内部逻辑

- 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

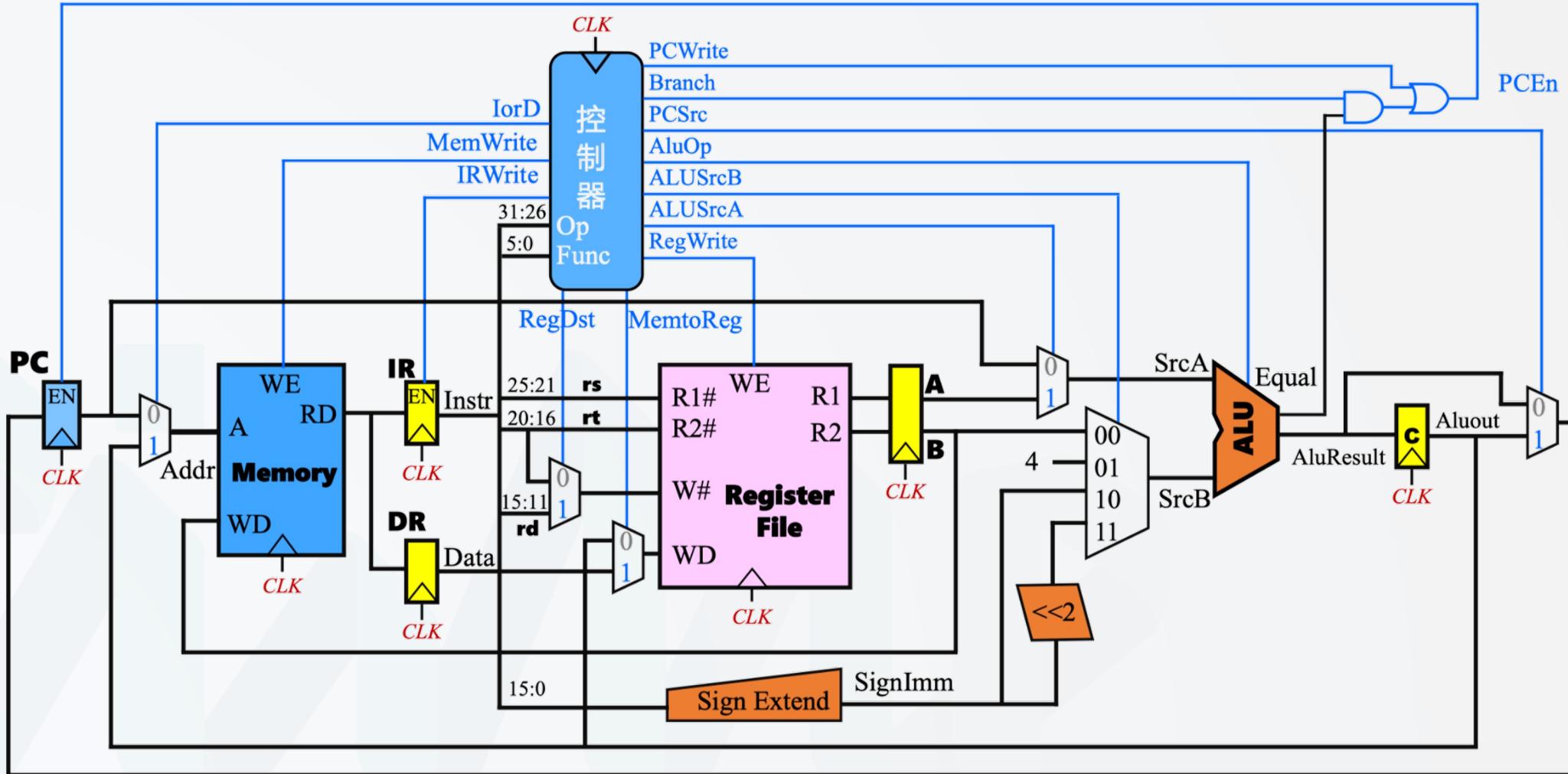
- 在指令存储器中载入排序程序sort.hex
- 时钟自动仿真，Windows：Ctrl+k Mac: command+k 运行程序
- 程序停机后，查看数据存储器中排序情况，有符号降序排列

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

III CPU设计实验

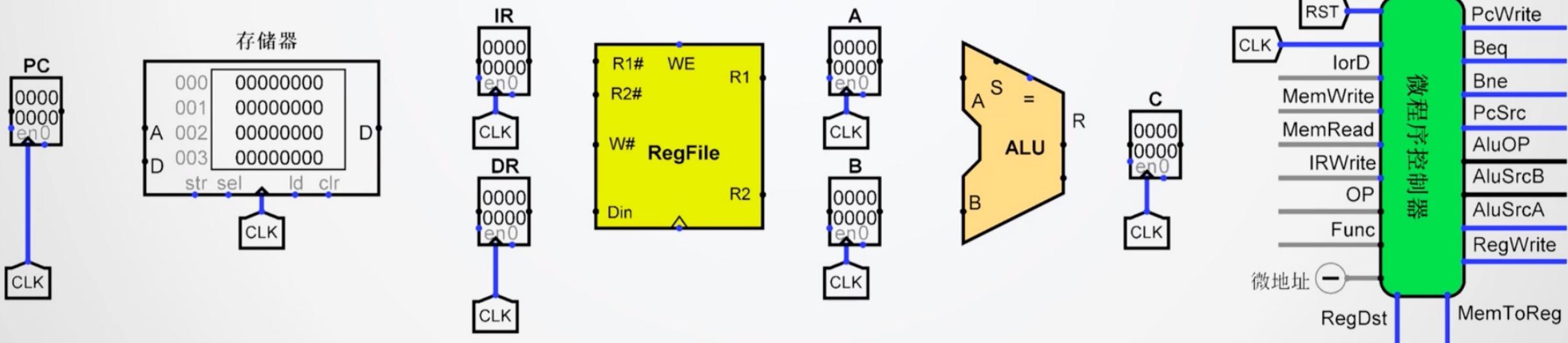
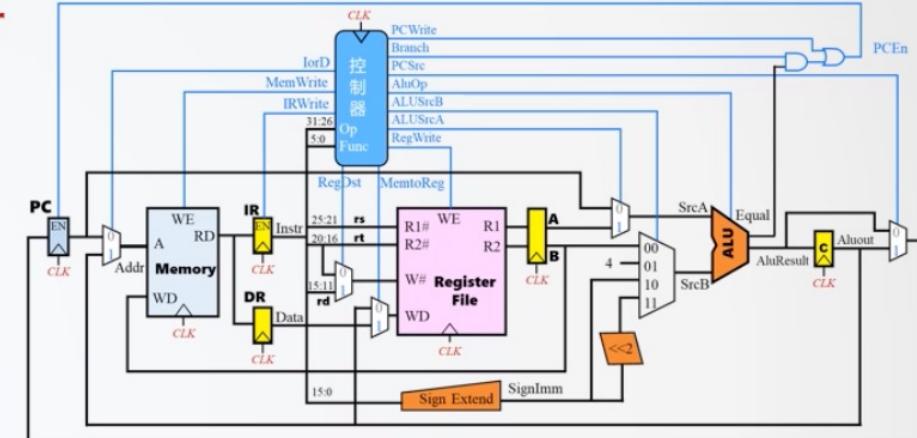
6

多周期MIPS 顶层视图



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

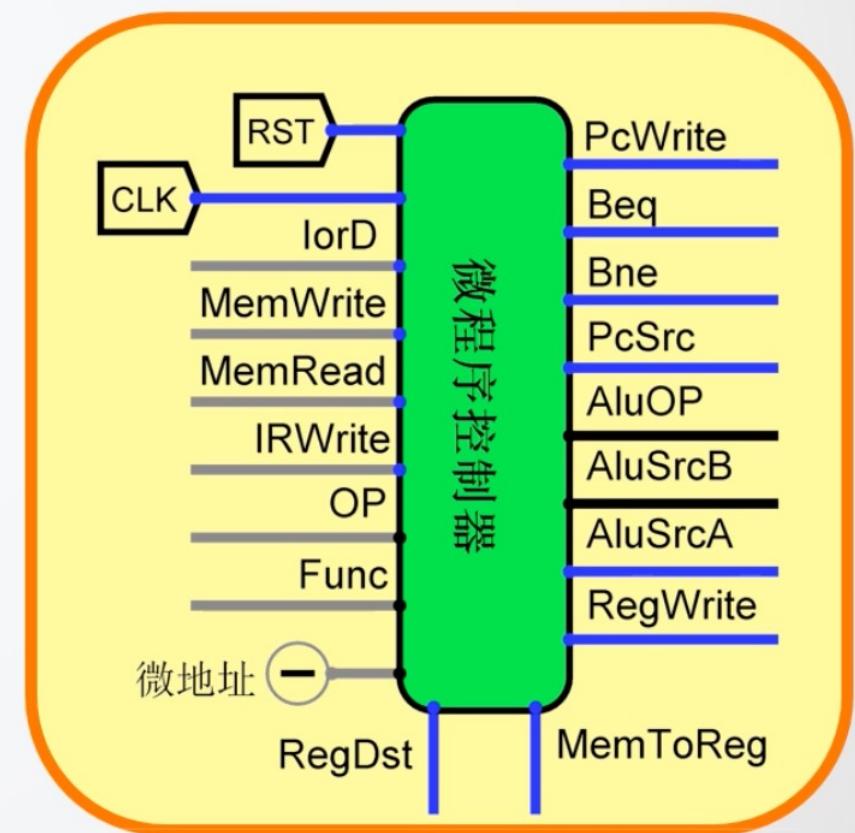
输出信号

- 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

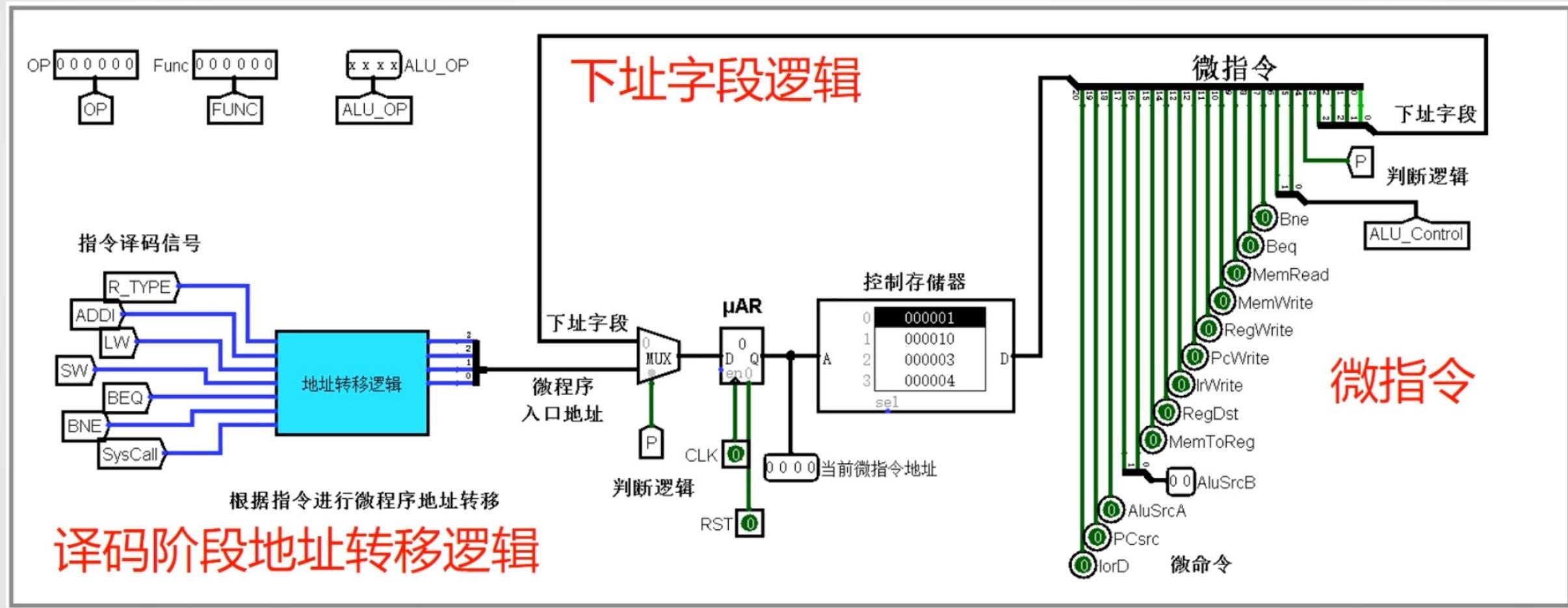


控制信号功能说明 (8条核心指令集)

#	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

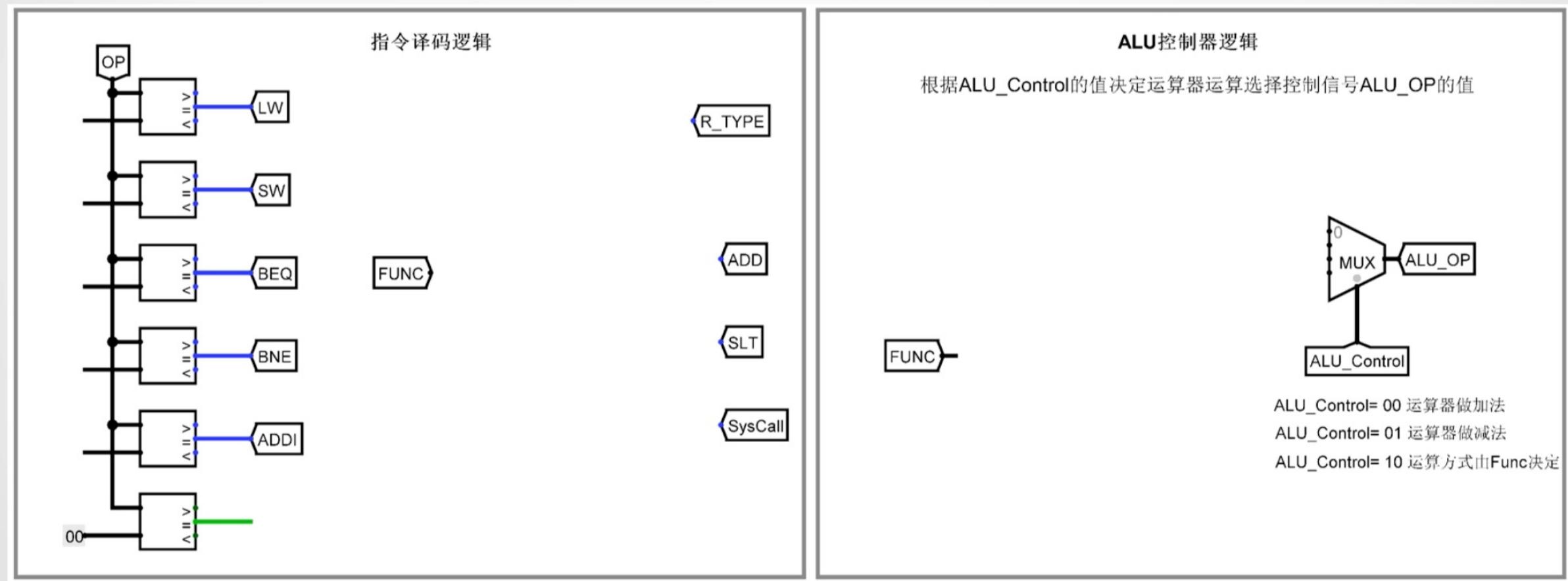
微程序控制器内部架构

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)

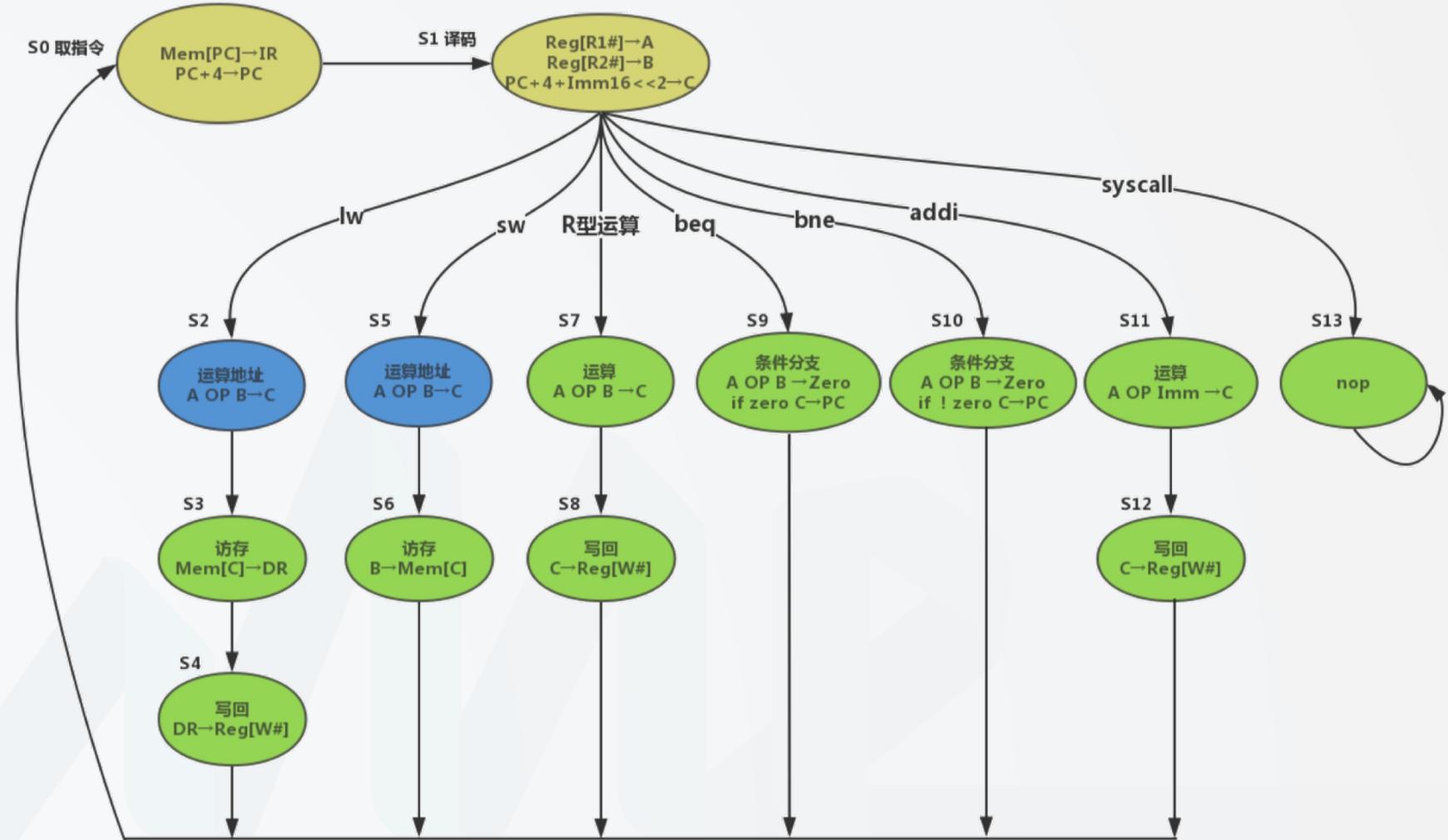
□ 左侧是译码信号，右侧是指令对应的微程序入口地址，地址转移逻辑会自动生成

机器指令译码信号							微程序入口地址				
R_Type	ADDI	LW	SW	BEQ	BNE	SYSCALL	入口地址 10进制	S3	S2	S1	S0
1							7	0	1	1	1
	1						11	1	0	1	1
		1					9	1	0	0	1
			1								
				1							
					1						
						1					

微程序地址入口表 地址逻辑自动生成

7

构建指令周期状态转换图



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

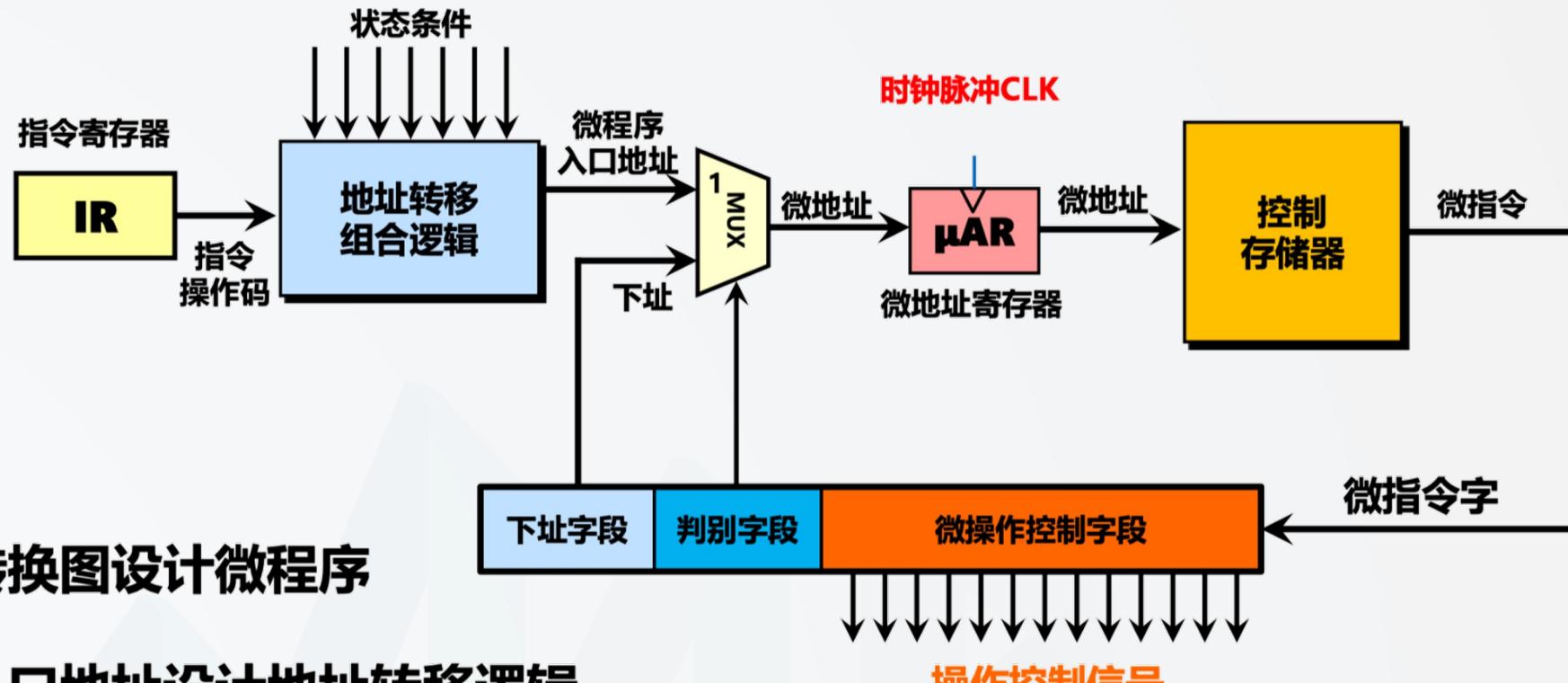
微指令功能	状态	微指令地址	lOrD	PcSrc	AluSrcA	AluSrcB	MemToReg	RegDst	BEQ	BNE	AluControl	P	下址字段	微指令	十六进制
取指令	0	0000	0	0	0	01	0	0	0	0	00	0	0001	00001001100100000001	13201
译码	1	0001	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000000000000000	0
LW1	2	0010	00	0	0	00	0	0	0	0	00	0	0000	00000000000000000000000000000000	0
LW2	3	0011	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000000000000000	0
LW3	4	0100	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000000000000000	0
SW1	5	0101	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000000000000000	0
SW2	6	0110	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000000000000000	0
	7	0111	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000000000000000	0
	8	1000	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000000000000000	0
	9	1001	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000000000000000	0
	10	1010	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000000000000000	0
	11	1011	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000000000000000	0
	12	1100	0	0	0	00	0	0	0	0	00	0	0000	00000000000000000000000000000000	0
	13	1101	1	0	0	00	0	0	0	0	11	0	0000	10000000000000001100000	100060

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

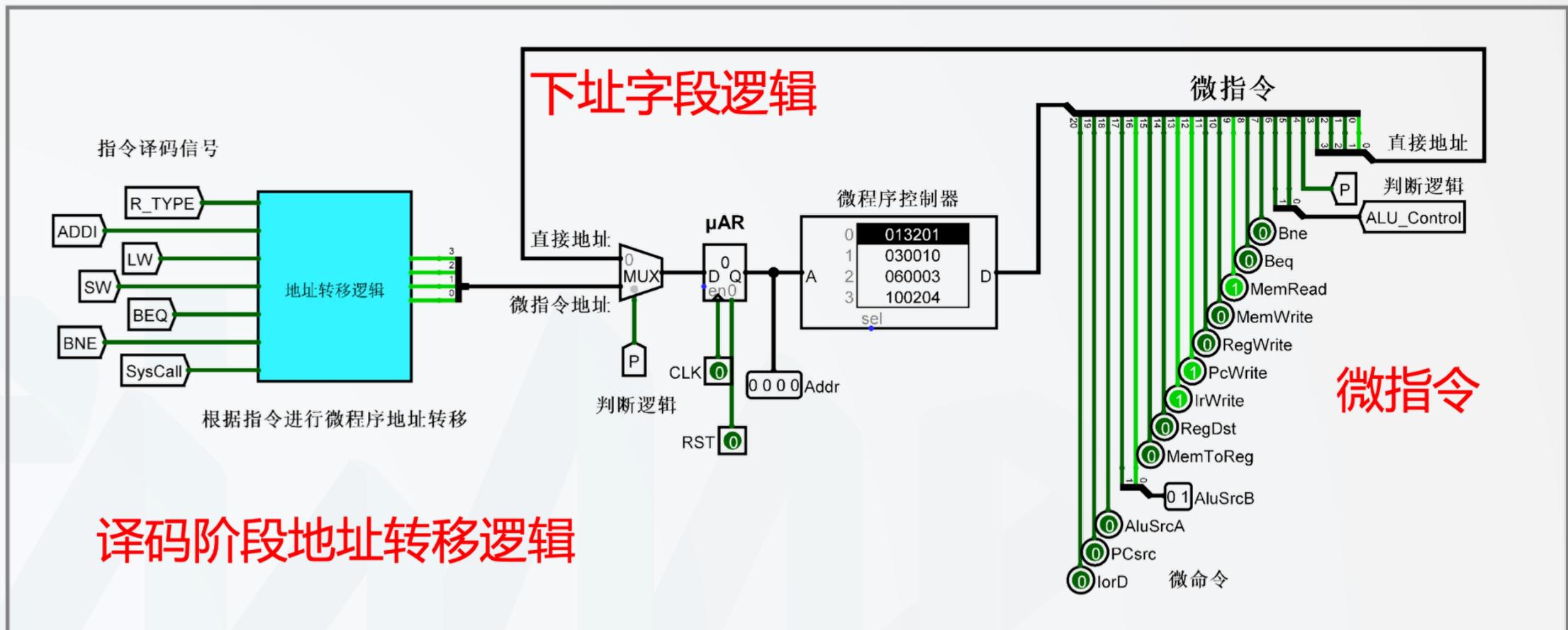
8 构造微程序控制器



- 根据状态转换图设计微程序
- 按微程序入口地址设计地址转移逻辑
- 构造微程序控制器

|| CPU设计实验

9 微程序控制器Logisim实现



II CPU设计实验

A

硬布线控制器Logisim实现

