计算机与网络技术

第4讲　简易CPU设计

课程回顾

- **计算机指令系统（MIPS）**
  - 指令系统
  - 寻址方式

- **计算机逻辑电路模块**
  - 计算机微体系结构
  - 运算器（ALU）
  - 多路开关、译码器
  - 寄存器、指令指针（程序计数器）与取指单元
  - 存储器、I/O接口（微处理器外部）

寻址方式

寻址方式（Addressing Mode）

- 指令设计、编写：指明操作数来源的方式
- 指令执行：确定本条指令所需要的操作数的地址，确定下一条要执行指令地址

- 常见操作数寻址方式：
  - 立即数寻址（立即寻址）
  - 寄存器寻址
  - 直接寻址
  - 寄存器间接寻址、（存储器）间接寻址
  - 变址寻址
  - 隐含寻址

- 指令的寻址方式：
  - 顺序寻址
  - 跳跃寻址

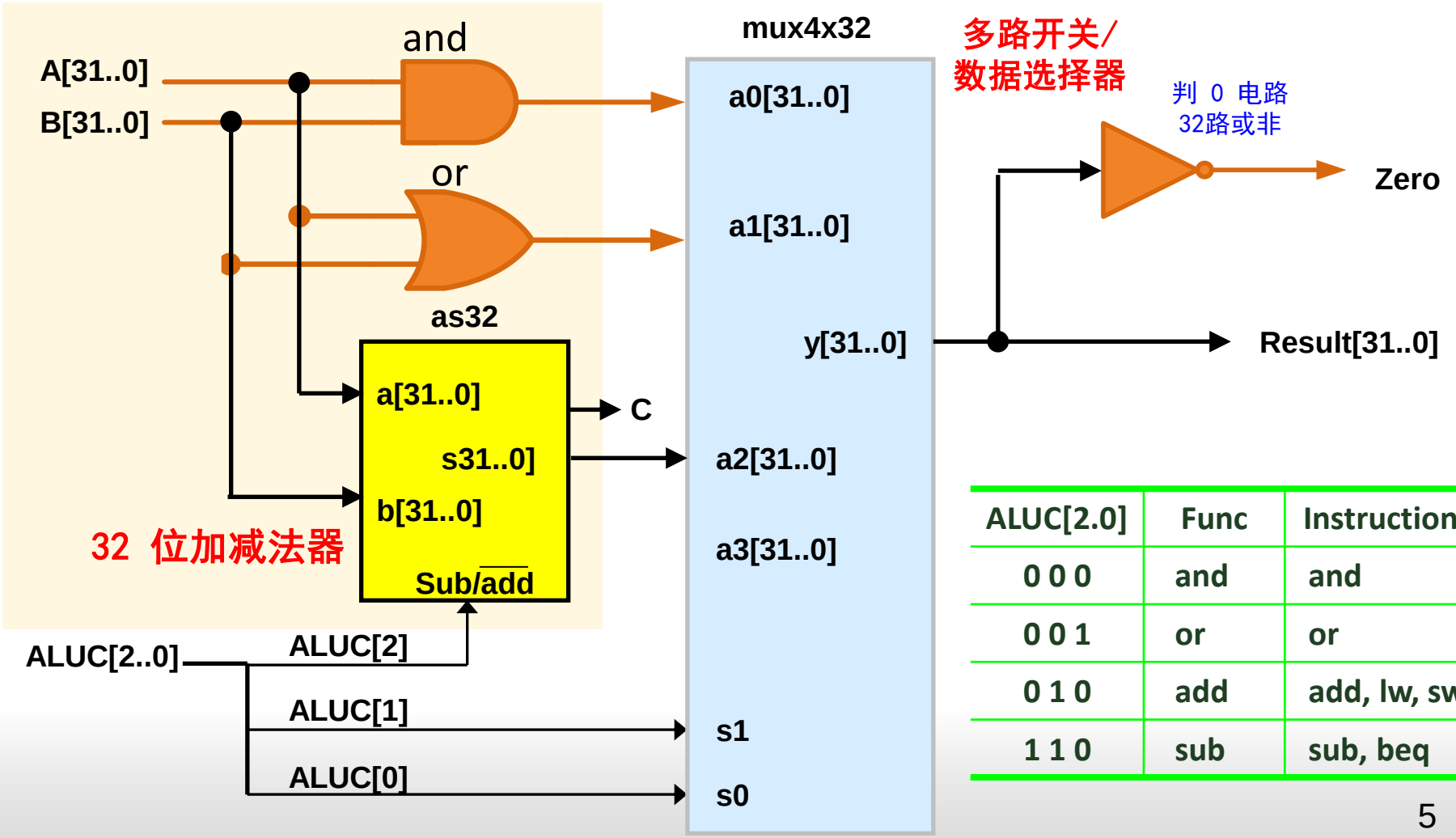
计算机逻辑电路模块

计算机常用逻辑电路模块

- 运算器（ALU）
  - 加法器、减法器、乘法器、除法器
  - 逻辑运算器、比较器（关系运算）
- 多路开关/数据选择器
- 译码器（指令译码、地址译码）
- 寄存器与寄存器组
- 指令指针（程序计数器）与取指单元
- 存储器（微处理器外部）
- I/O接口（微处理器内&外部）

计算机逻辑电路模块

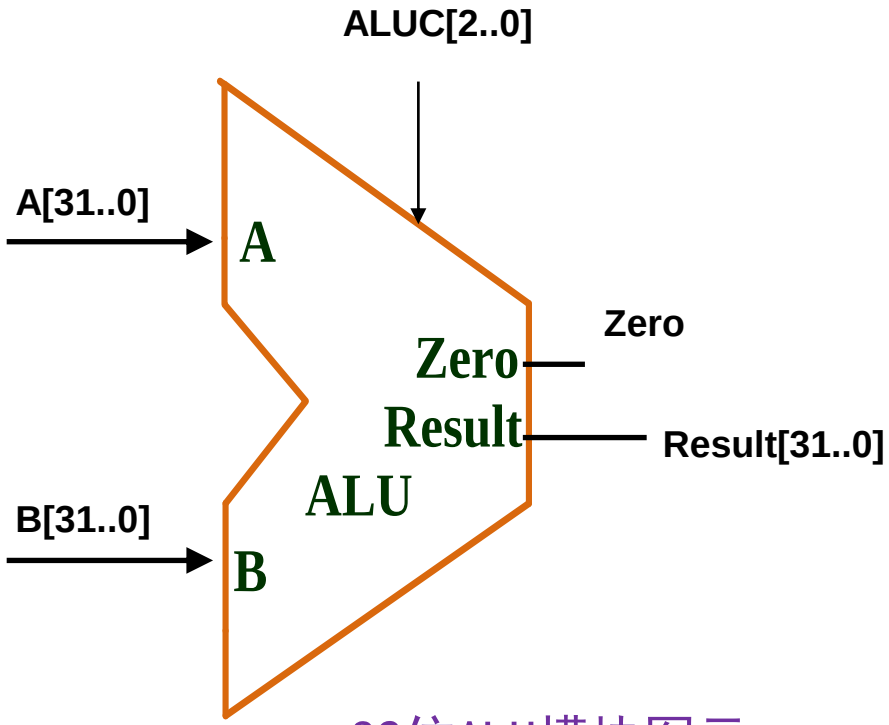
运算器（ALU）：32位ALU结构



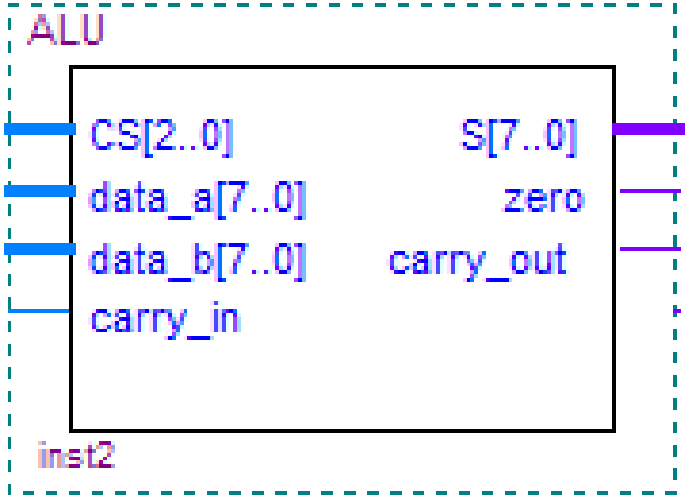
ALUC[2.0]	Func	Instructions
0 0 0	and	and
0 0 1	or	or
0 1 0	add	add, lw, sw
1 1 0	sub	sub, beq

计算机逻辑电路模块

运算器（ALU）：ALU图符



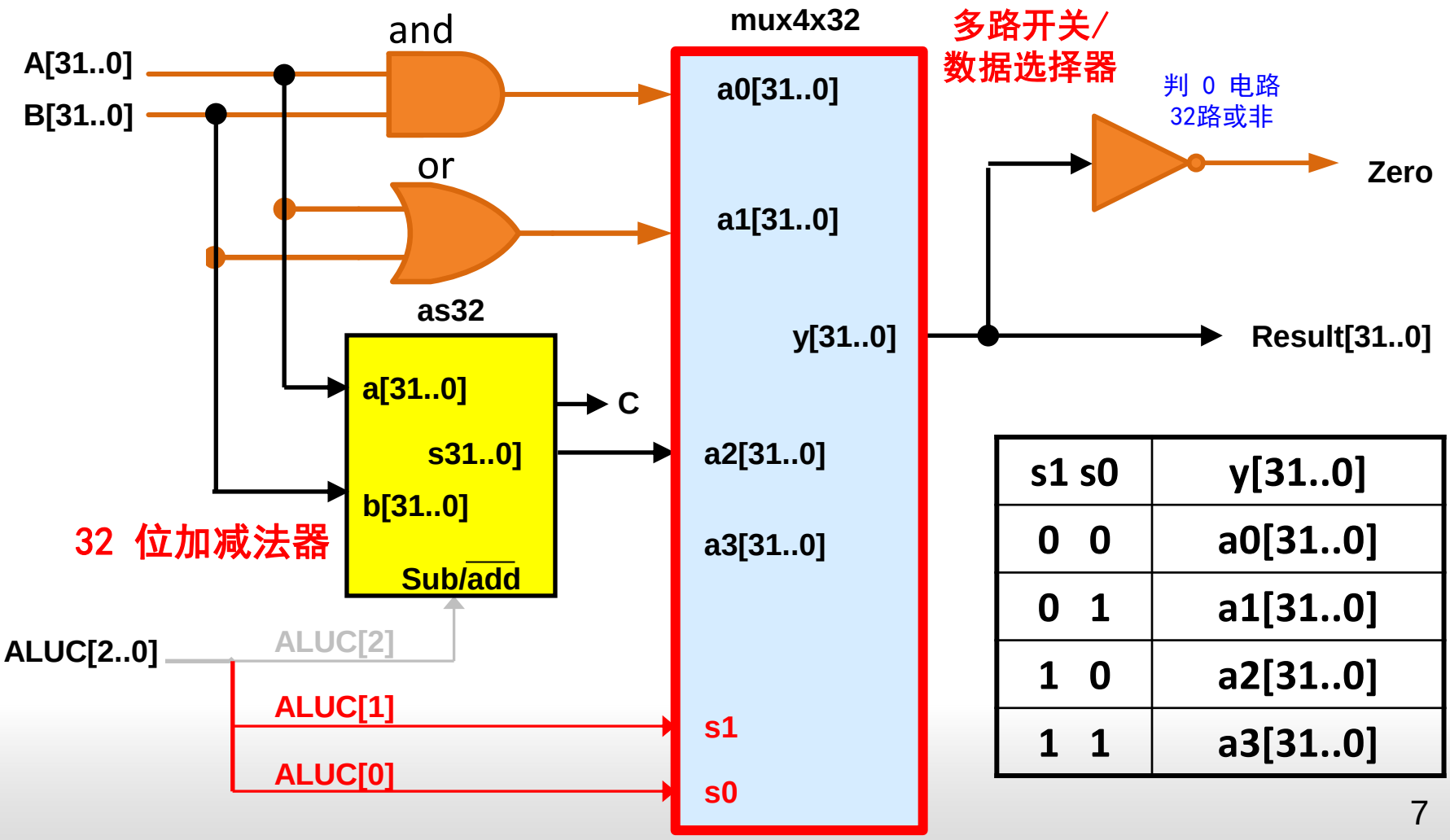
32位ALU模块图示



8位ALU模块封装图

计算机逻辑电路模块

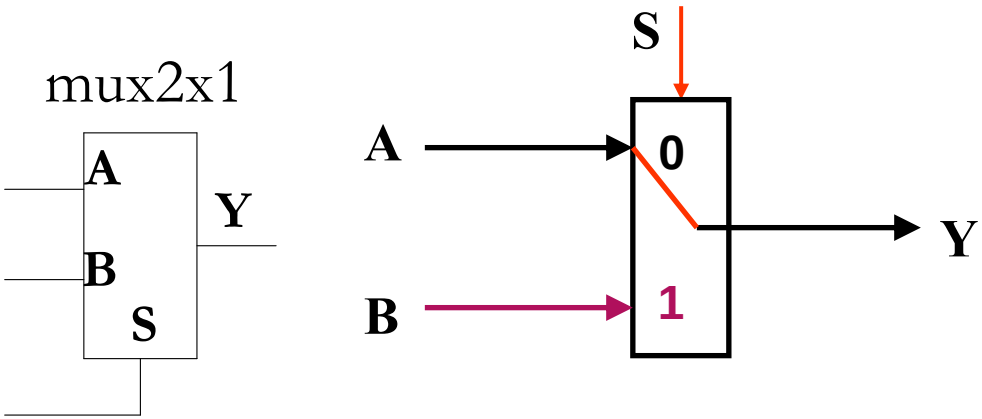
ALU中的多路开关/数据选择器

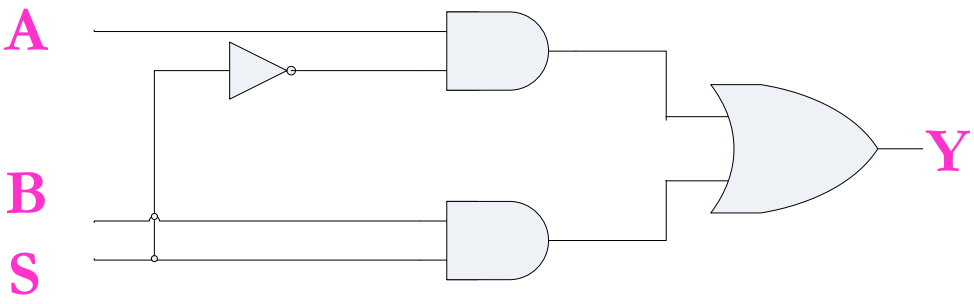


s1 s0	y[31..0]
0 0	a0[31..0]
0 1	a1[31..0]
1 0	a2[31..0]
1 1	a3[31..0]

计算机逻辑电路模块

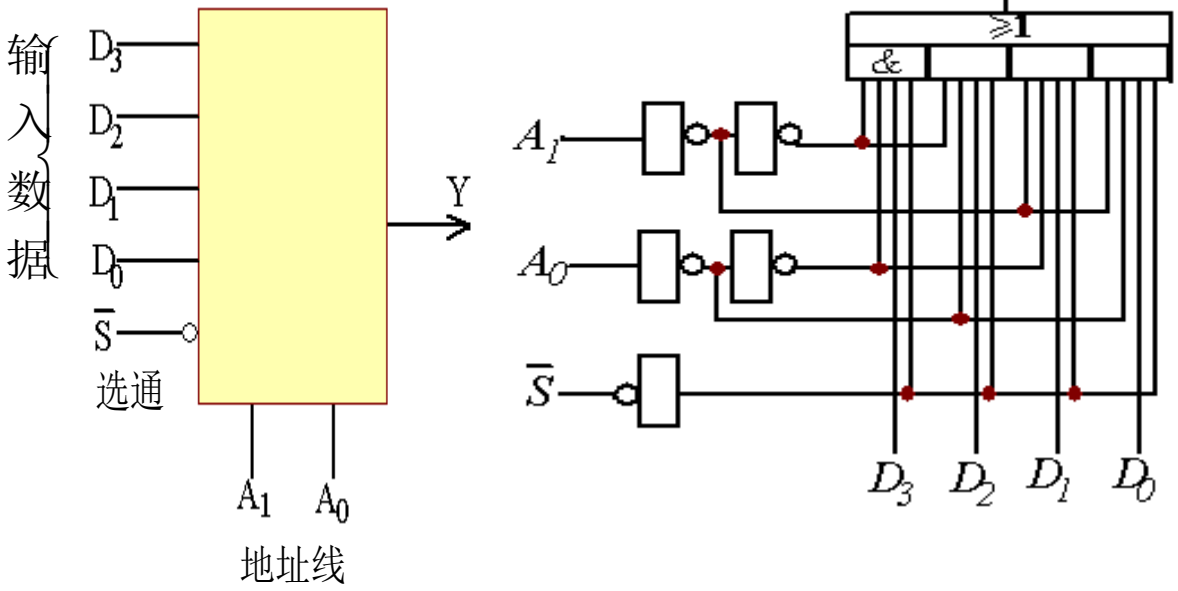
多路开关/数据选择器（Multiplexer）：2选1





S	A	B	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

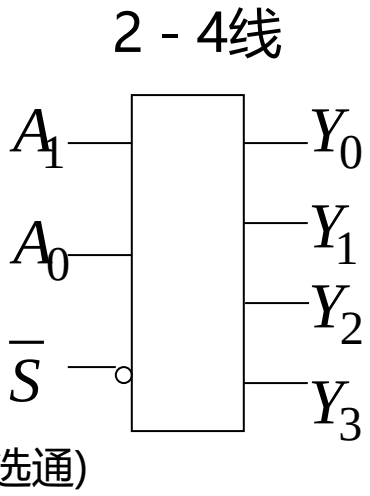
多路开关/数据选择器（Multiplexer）：4选1

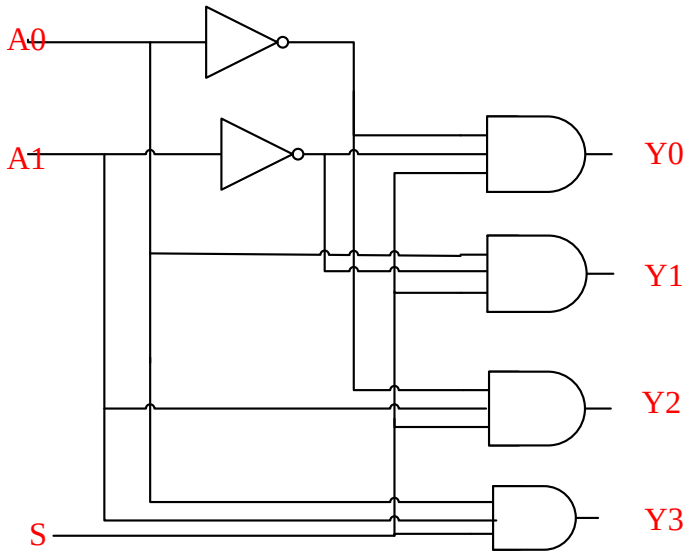


$\overline{S}$	A_1	A_2	Y
1	X	X	0
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3

计算机逻辑电路模块

译码器：2-4译码器（n >> 2^n）



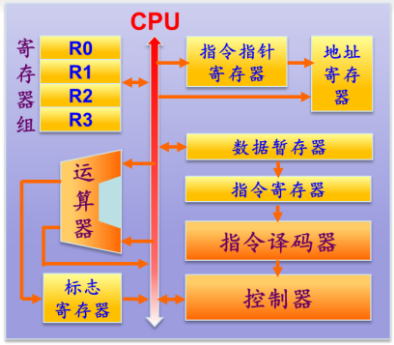


输出Y：通常作为下一级模块的选择、使能等信号

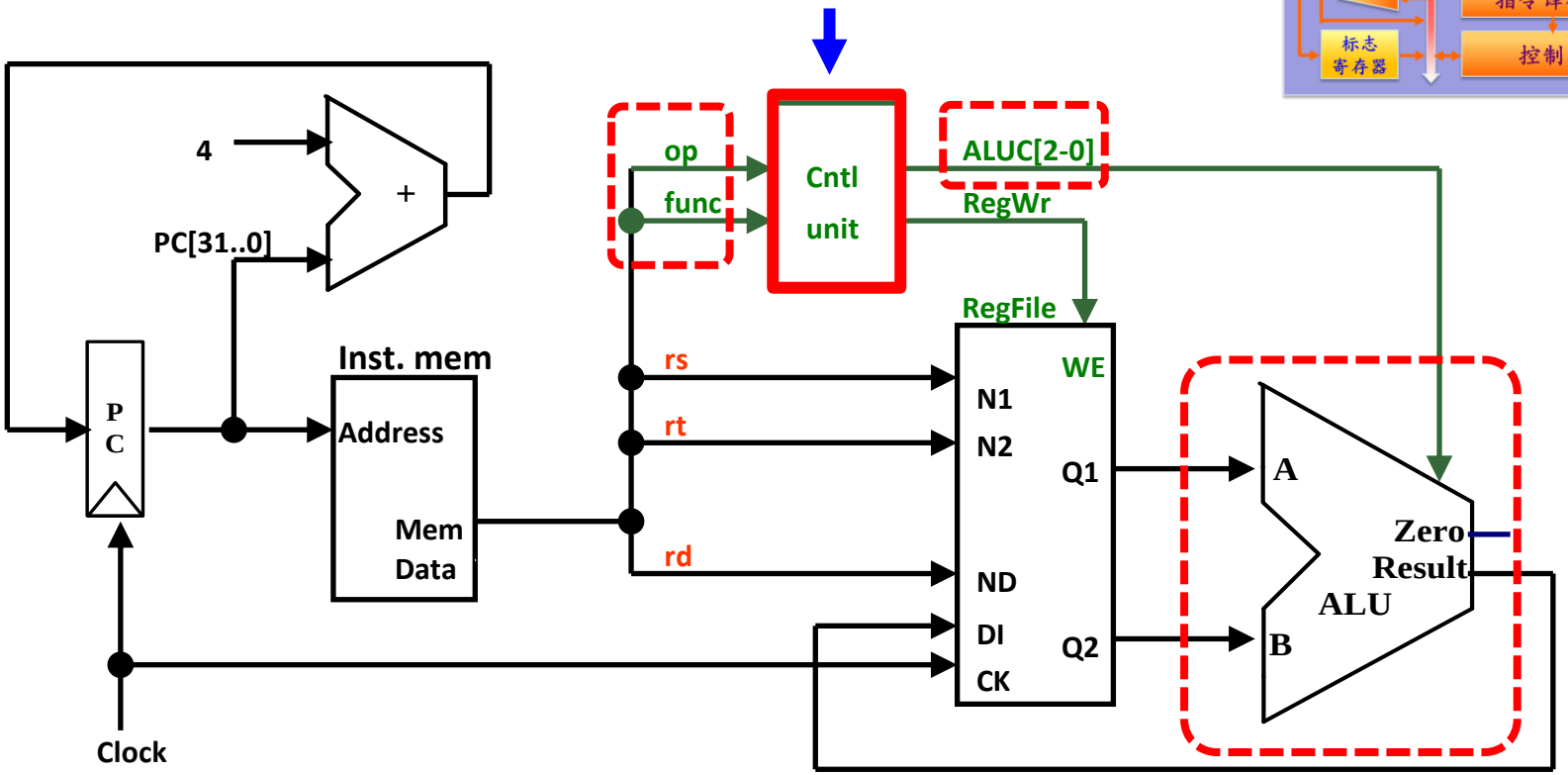
S	A_1	A_2	Y_i
1	X	X	0000
0	0	0	0001
0	0	1	0010
0	1	0	0100
0	1	1	1000

计算机逻辑电路模块

译码器：用于指令译码

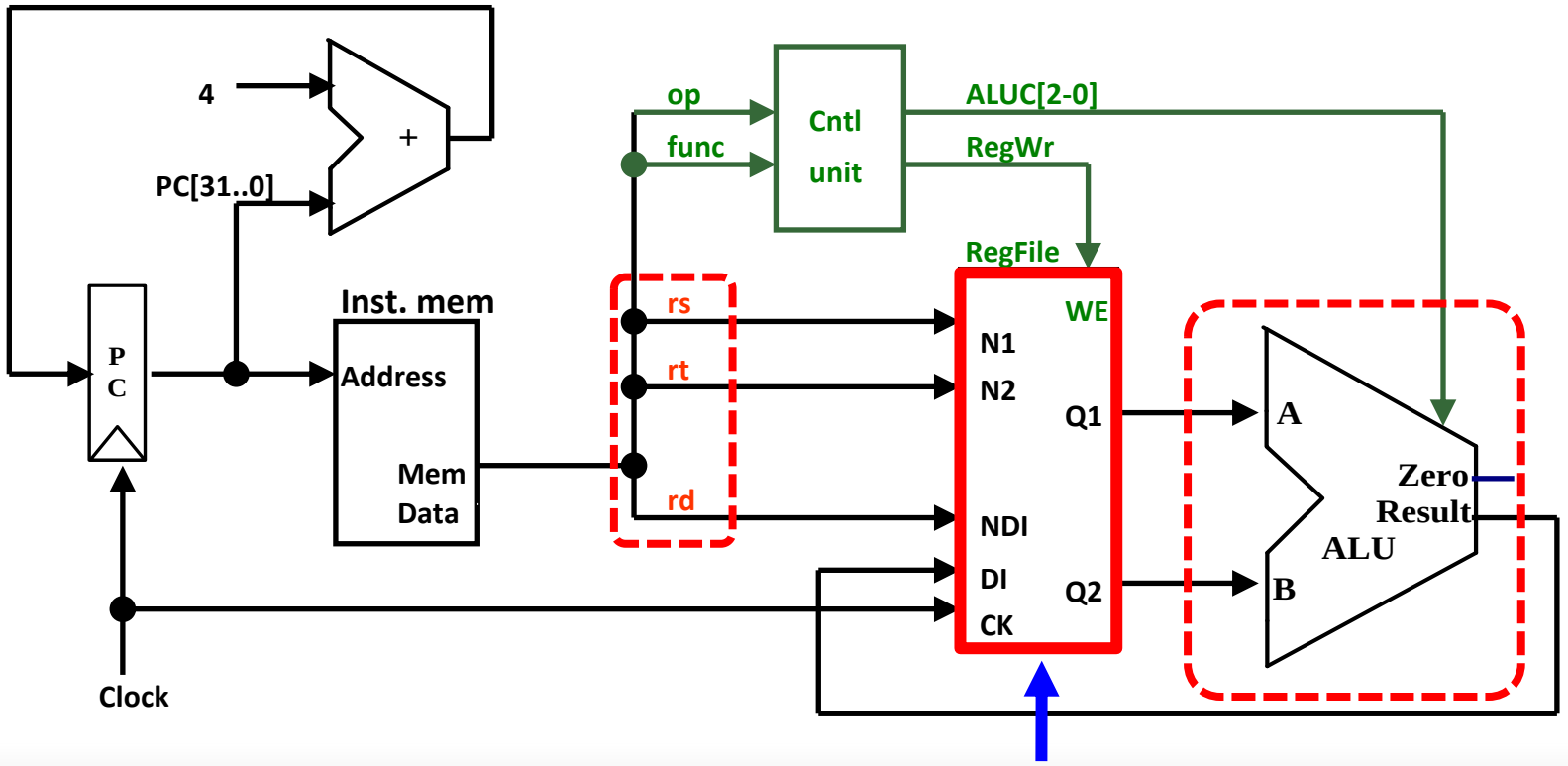


控制器：内含指令译码器



计算机逻辑电路模块

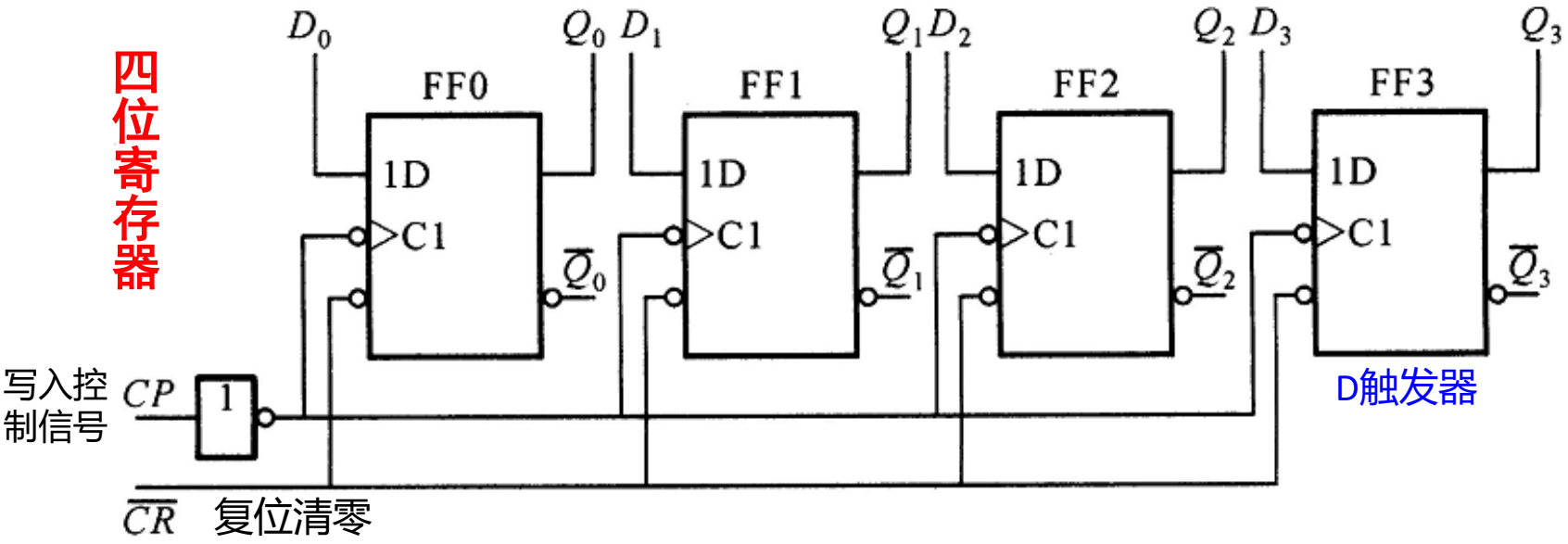
译码器：用于地址译码



寄存器组：内含地址译码器

计算机逻辑电路模块

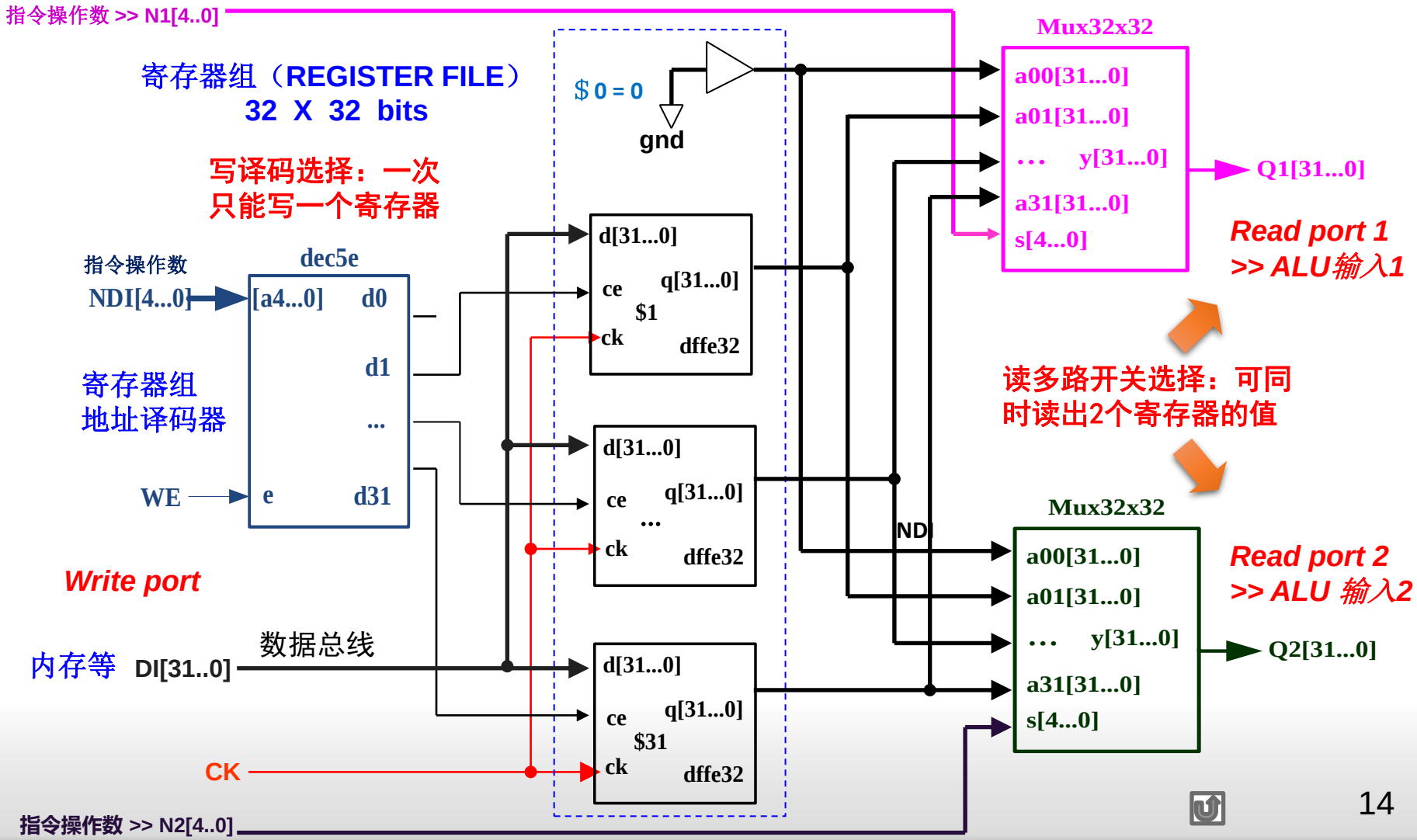
寄存器（Register）：4位寄存器



寄存器： CPU的重要组成部分，用来暂存指令、数据和地址

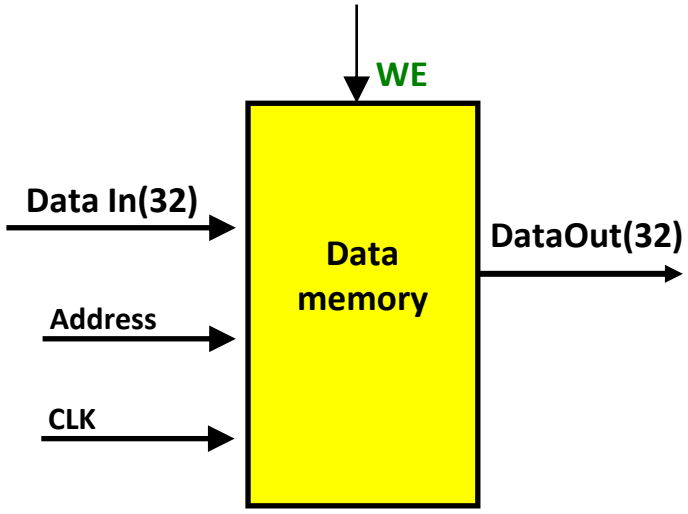
计算机逻辑电路模块

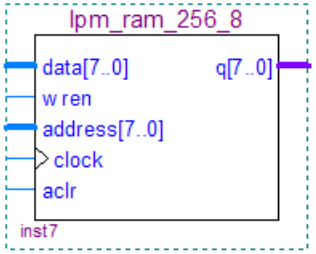
寄存器（Register）：寄存器组

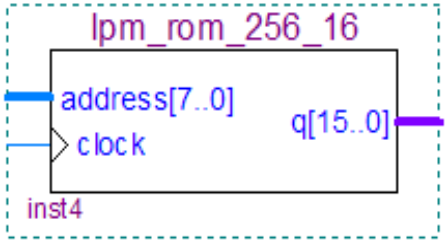


计算机逻辑电路模块

存储器（Memory）：~= 寄存器组 + 地址译码器



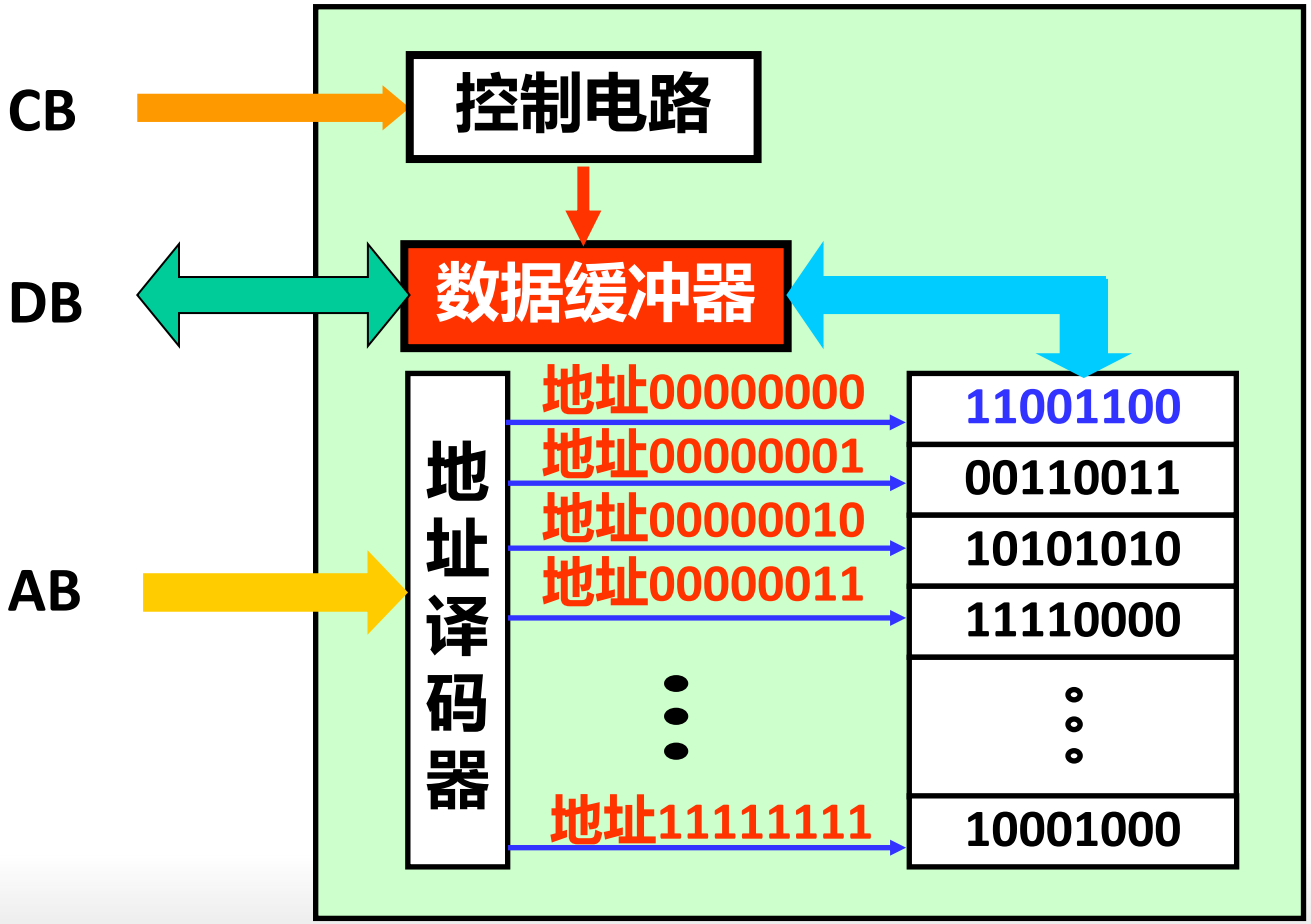




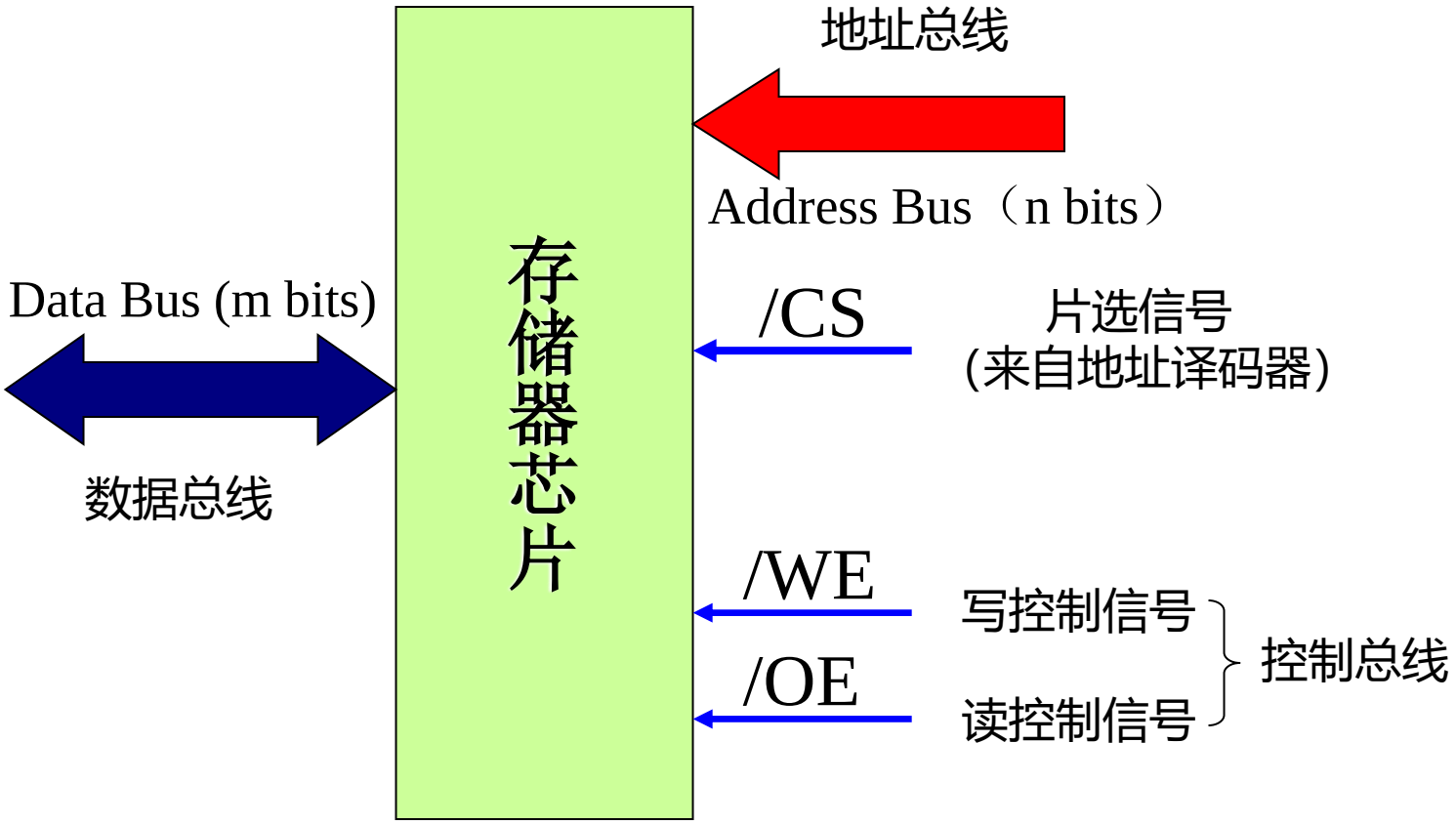
- Memory (idealized)
  - One input bus: Data In
  - One output bus: Data Out
- Memory word is selected by:
  - Write Enable = 1: address selects the memory data to be written via the Data In bus
  - Address selects the data to put on Data Out
- Clock input (CLK)
  - The CLK input is a factor ONLY during write operation
  - During read operation, behaves as a combinational logic block:
    - Address valid => Data Out valid after “access time.”

计算机逻辑电路模块

存储器芯片结构：



存储器总线连接


地址总线
Address Bus（n bits）
Data Bus (m bits)
存储器芯片
/CS
片选信号
(来自地址译码器)
数据总线
/WE
写控制信号
/OE
读控制信号
控制总线

提纲

- CPU设计基本步骤
- 系统指令数据通路设计
- 系统指令控制逻辑设计
- 系统指令数据通路与控制逻辑集成
- CPU时序分析
- CPU性能指标

CPU设计基本步骤

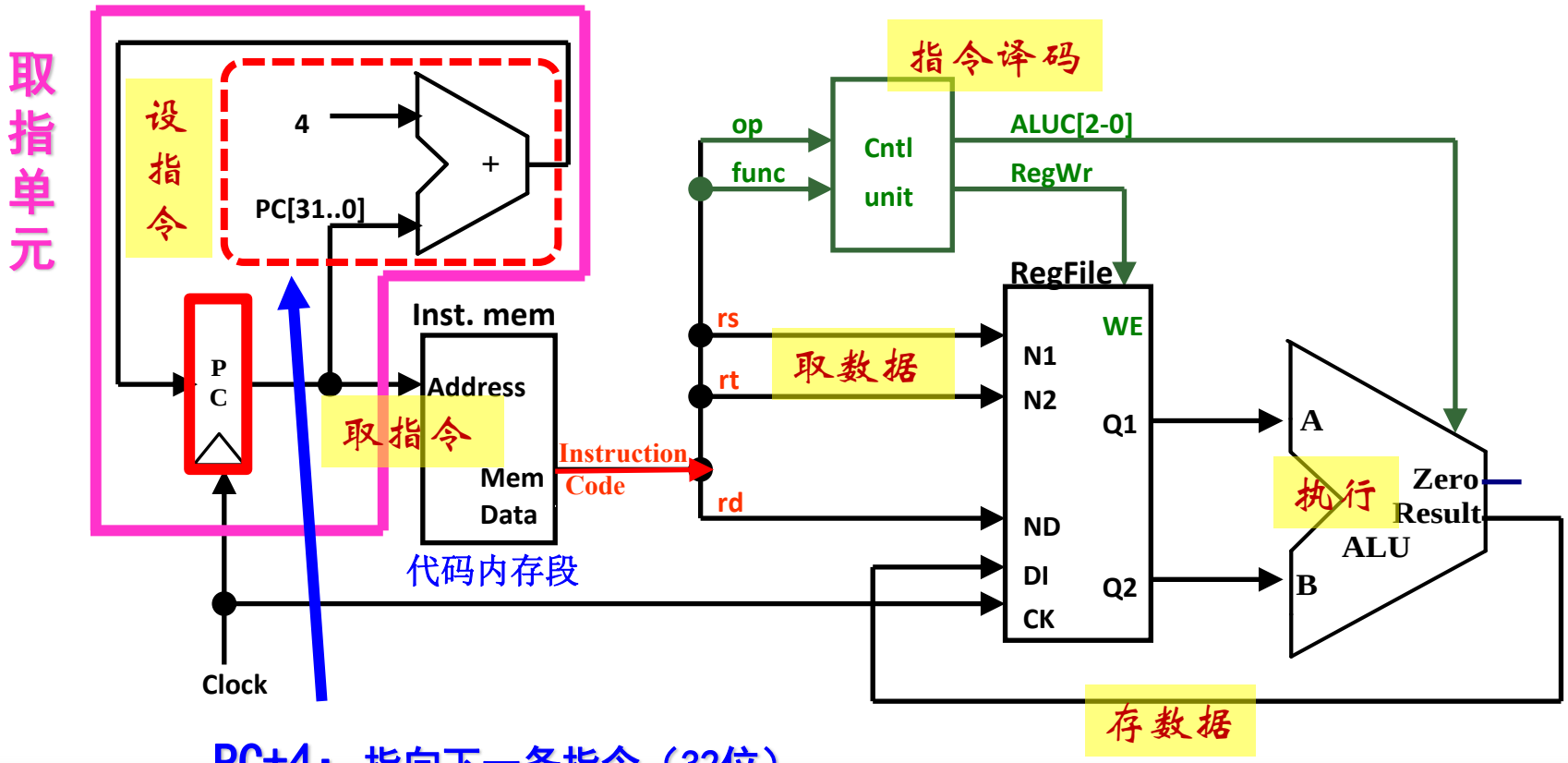
1. 根据功能和性能需求，设计指令系统
2. 分析系统指令特点，明确数据通路（Datapath） 需求；设计合理的数据通路，满足数据在存储单元、寄存器、运算器等逻辑电路模块之间的传输需求
3. 分析指令执行控制过程，设计控制逻辑
4. 实现指令集中全部指令数据通路和控制逻辑的集成

数据通路（Datapath）：指令执行过程中，数据所经过的路径，包括路径中的指令执行部件

指令控制部件（Control）：对执行部件发出控制信号，包括对指令进行译码，生成指令对应的控制信号，控制数据通路的动作

CPU数据通路实现

指令数据通路：取指令-取数据-指令译码-执行-存数据-设指令



PC+4：指向下一条指令（32位）

提纲

- CPU设计基本步骤
- **系统指令数据通路设计**
- 系统指令控制逻辑设计
- 系统指令数据通路与控制逻辑集成
- CPU时序分析
- CPU性能指标

系统指令数据通路设计

32位MIPS系统：指令格式

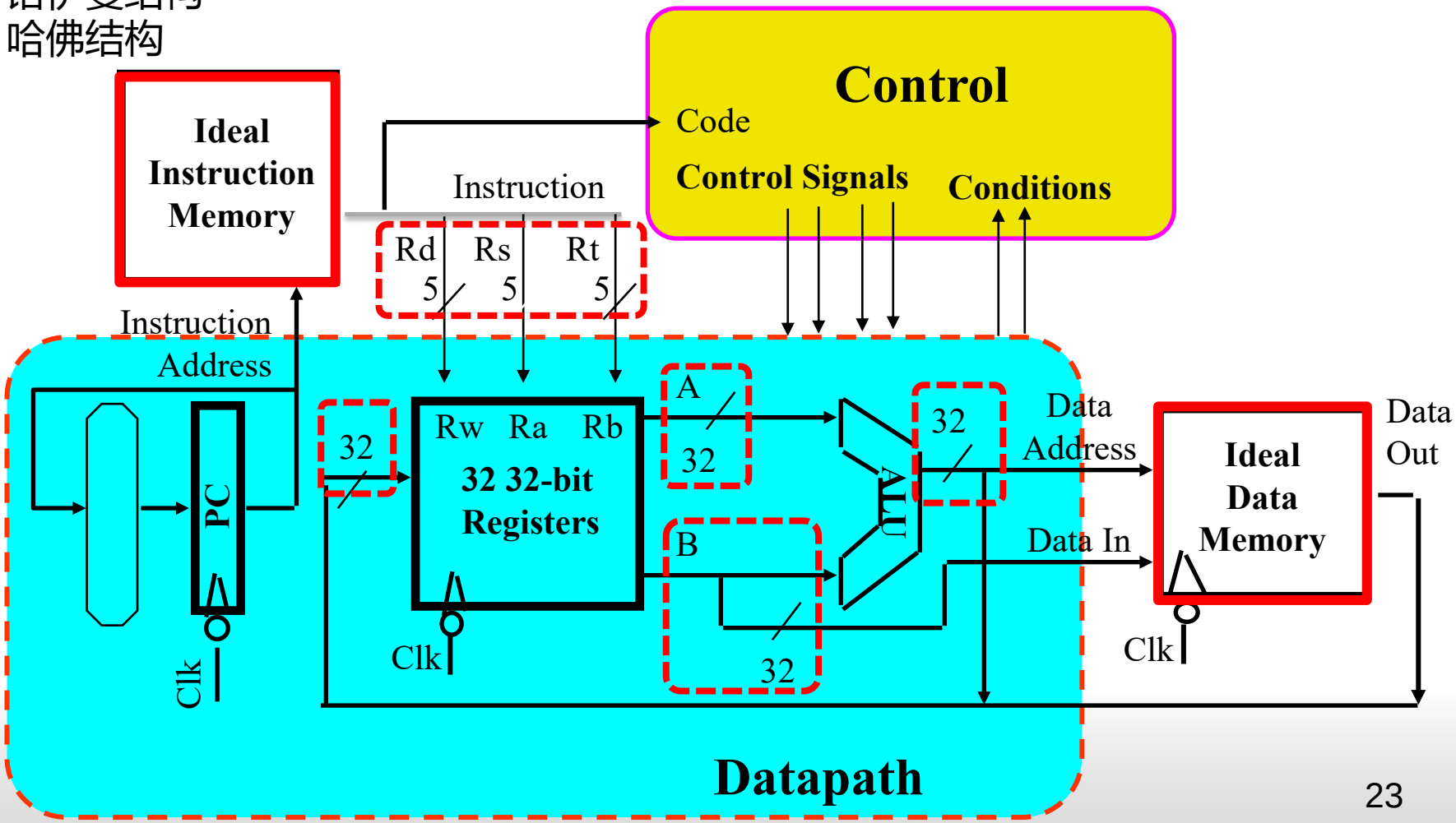
	31 26	25 21	20 16	15 11	10 6	5 0
R-type	Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Rd (5 bits)	Shift amt (5 bits)	Func (6 bits)
I-type	Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Immediate (16 bits)		
J-type	Opcode (6 bits)	Target address (Offset address to PC) (26 bits)				

- Opcode: partially specifies what instruction it is; Equal to 0 for all R-Format instructions
- Funct: combined with opcode, this number exactly specifies the instruction
- Rs (Source Register): generally used to specify register containing first operand
- Rt (Target Register): generally used to specify register containing second operand
- Rd (Destination Register): generally used to specify register which will receive result of Opcode
- Shamt: This field contains the amount a shift instruction will shift by.
- Immediate: address offset or immediate value
- Target address (Offset address to PC): target address of the jump instruction

系统指令数据通路设计

系统指令的数据通路总体设计：基于指令特点

程序存储
- 诺伊曼结构
- 哈佛结构

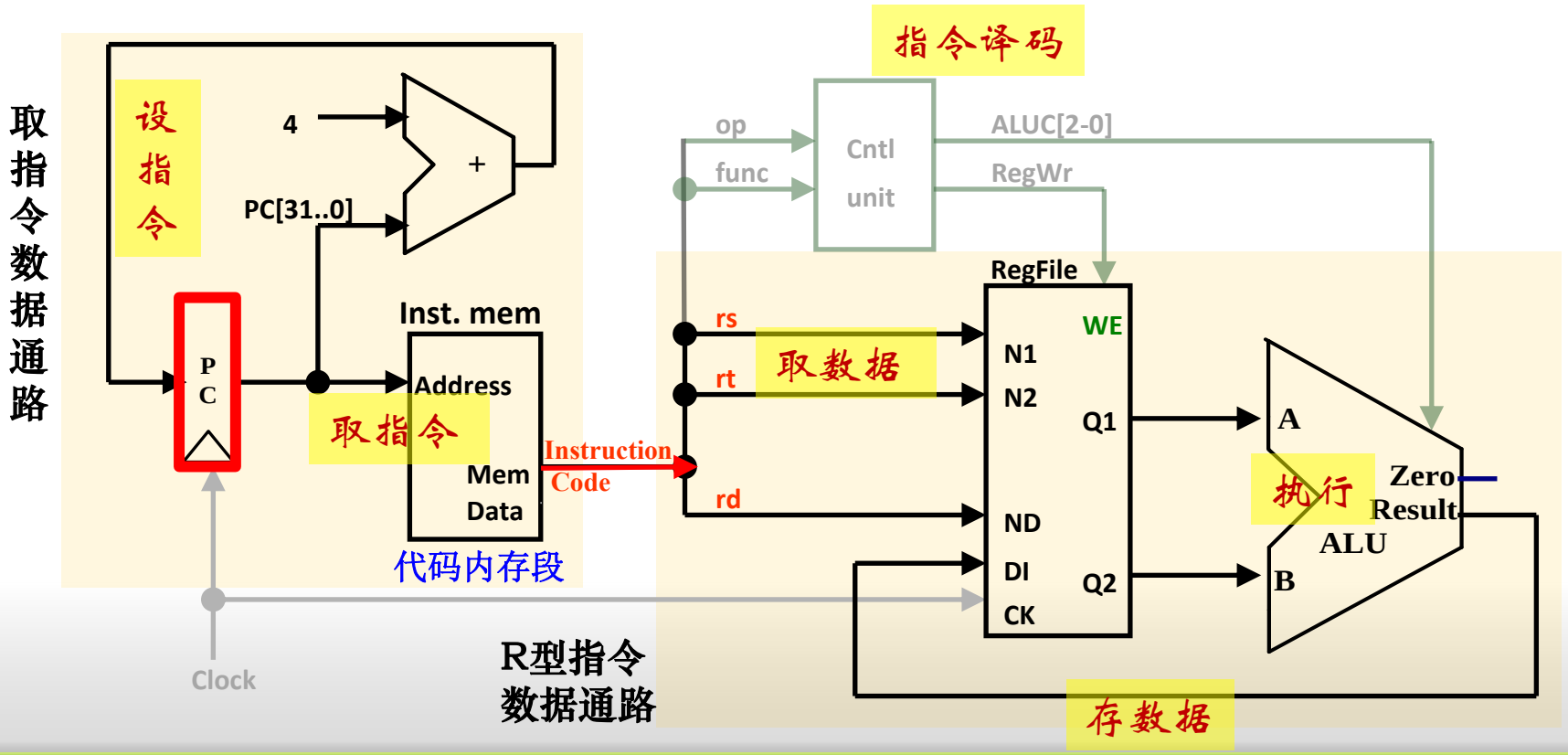


系统指令数据通路设计

单条指令的数据通路设计：32位MIPS系统R指令

R-format add/sub/and/or/slt rs, rt, rd # rd = rs *op* rt					
Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Rd (5 bits)	Shift amt (5 bits)	Func (6 bits)

三操作数的R指令：ALU输入来源于两个源寄存器，ALU计算结果返回目标寄存器

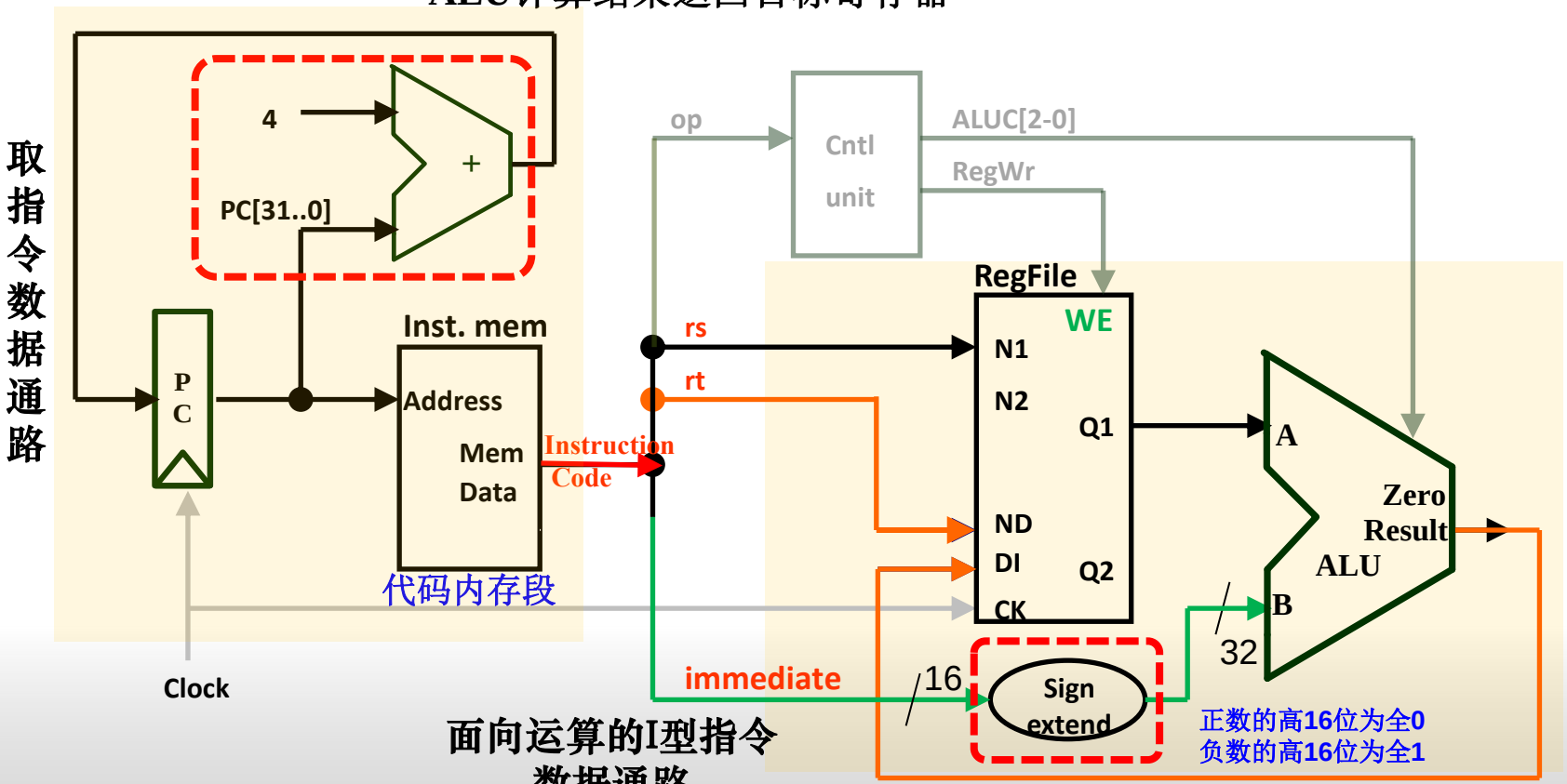


系统指令数据通路设计

单条指令的数据通路设计：32位MIPS系统I指令

I-format ori rs, rt, imm16			# rt = rs *op* SignExtend[imm16]
Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Immediate (16 bits)

面向运算的I指令：ALU输入一个来源于源寄存器，另一个来源于立即数；ALU计算结果返回目标寄存器



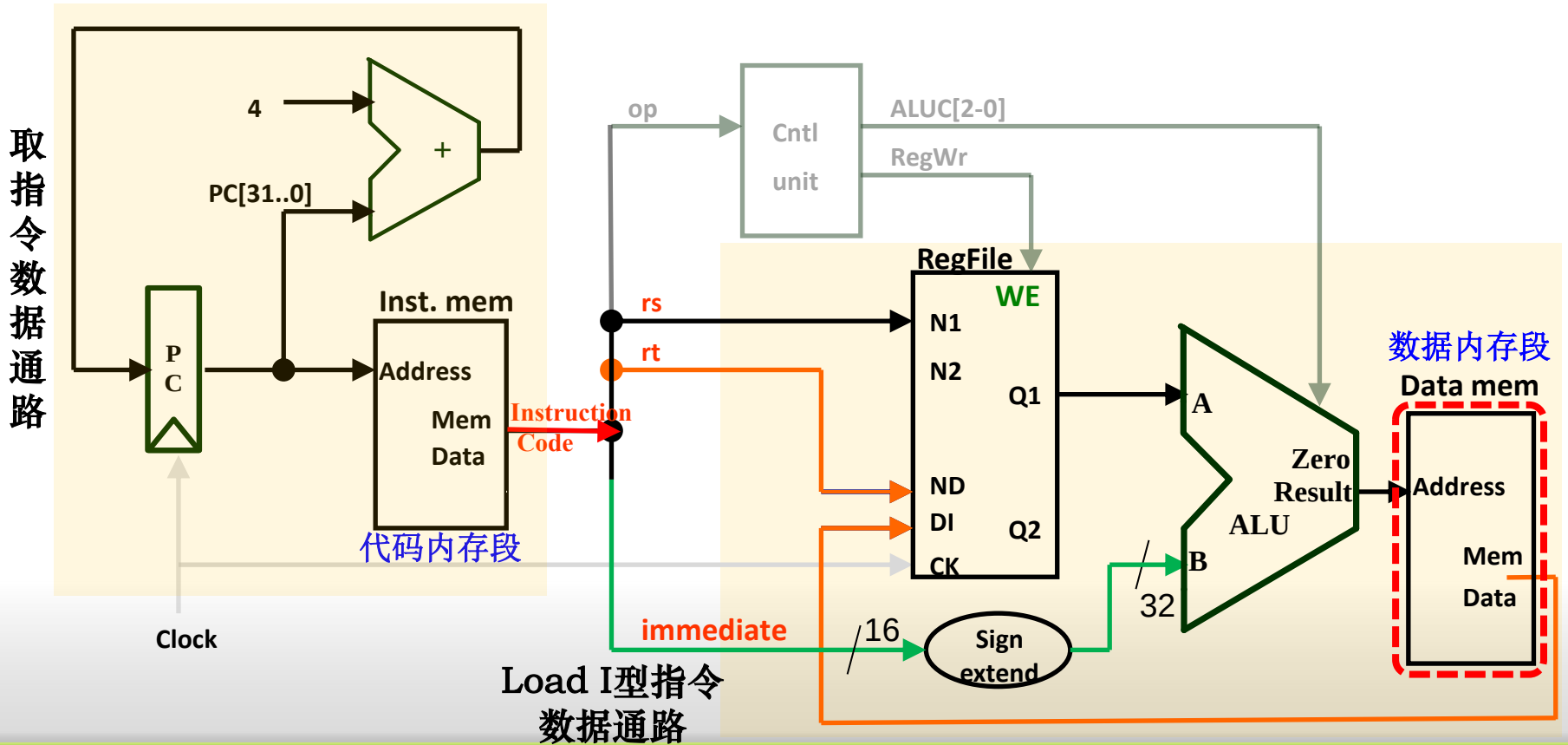
面向运算的I型指令数据通路

系统指令数据通路设计

单条指令的数据通路设计：32位MIPS系统I指令

I-format lw rt, #imm16(rs)			# rt = Memory[rs + imm16]
Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Immediate (16 bits)

面向访存的I指令（lw）： ALU输入一个来源于源寄存器，另一个来源于立即数（用于计算地址）；
ALU的输出是访存地址；存储单元输出到目标寄存器

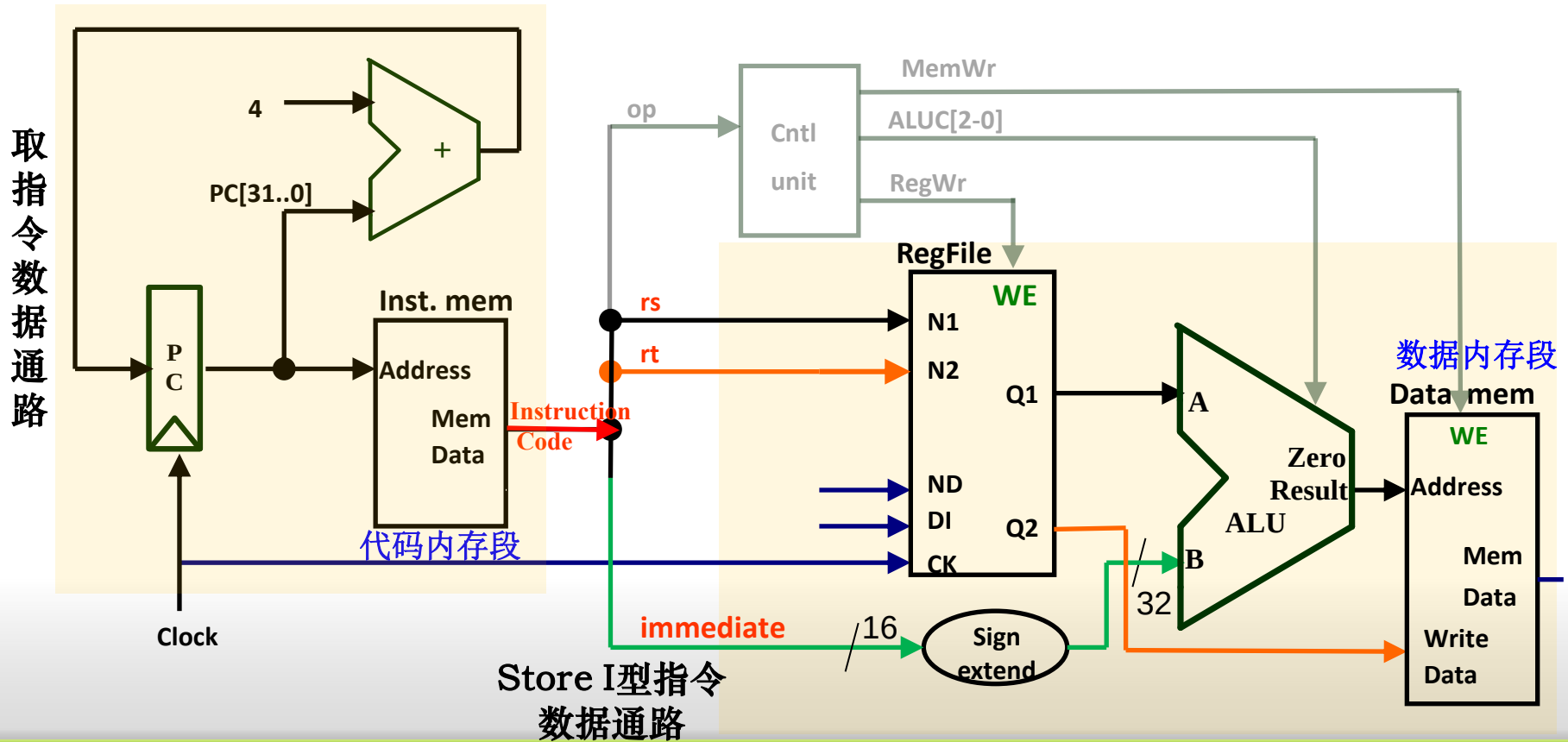


系统指令数据通路设计

单条指令的数据通路设计：32位MIPS系统I指令

I-format　sw　rt, #imm16(rs)　　# Memory[rs + imm16] = rt			
Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Immediate (16 bits)

面向访存的I指令（写）： ALU输入一个来源于源寄存器，另一个来源于立即数（用于计算地址）；
ALU的输出是访存地址；目标寄存器输入到存储单元



系统指令数据通路设计

单条指令的数据通路设计：32位MIPS系统I指令

I-format beq rs, rt, #imm16			# if rs = rt, goto PC+4+#imm16*4
Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Immediate (16 bits)

– mem[PC] Fetch the instruction from memory

– Equal = (R[rs] == R[rt]) Calculate the branch condition

R[rs] - R[rt] , zero flag

– Branch : calculate the next instruction's address :

– if (Equal) then if (zero)

PC <= PC + 4 + { SignExt(imm16)*4}

– else

PC <= PC + 4

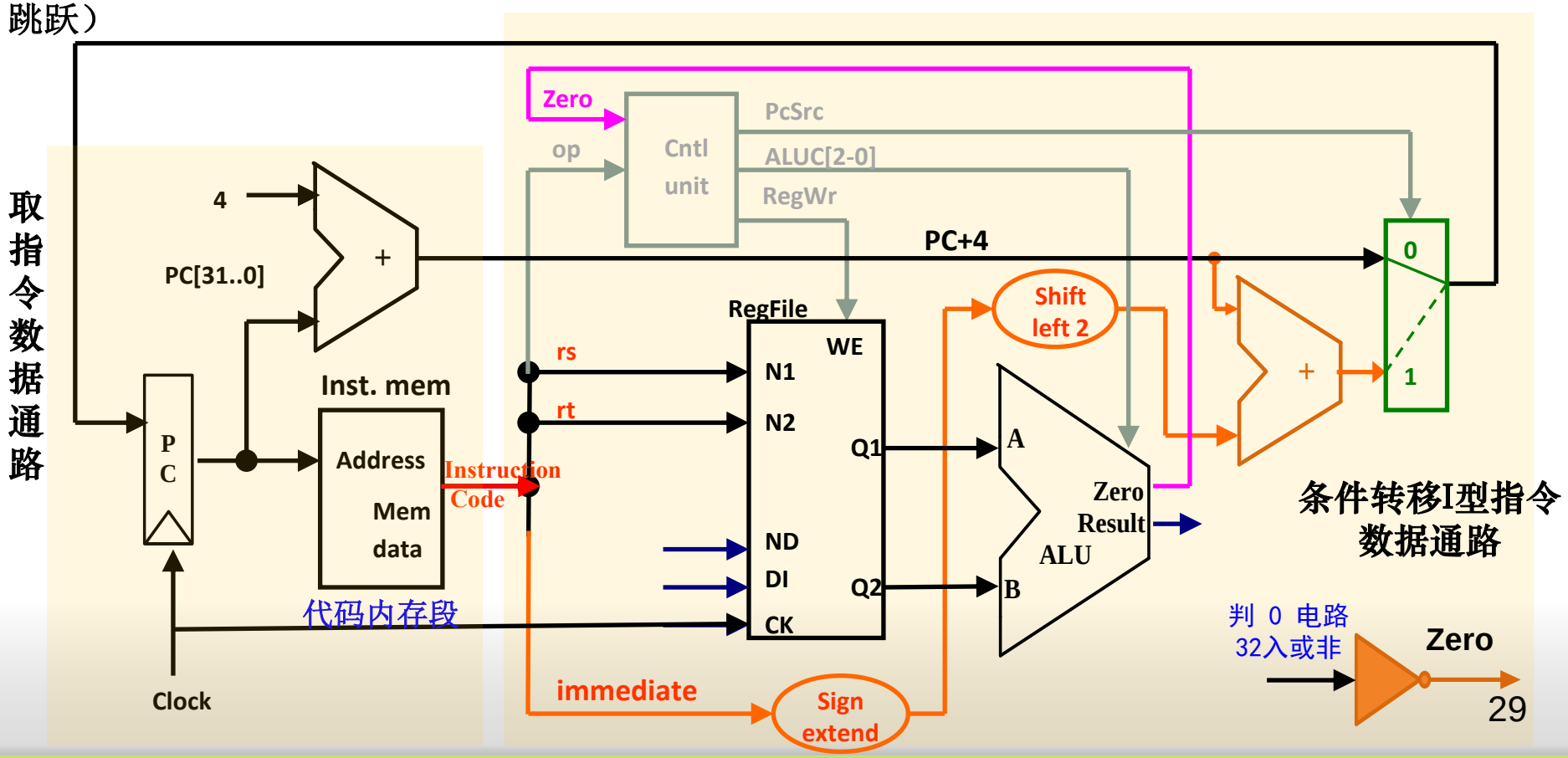
左移，最低两位00
地址对齐 align

系统指令数据通路设计

单条指令的数据通路设计：32位MIPS系统I指令

I-format beq rs, rt, #imm16			# if rs = rt, goto PC+4+#imm16*4
Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Immediate (16 bits)

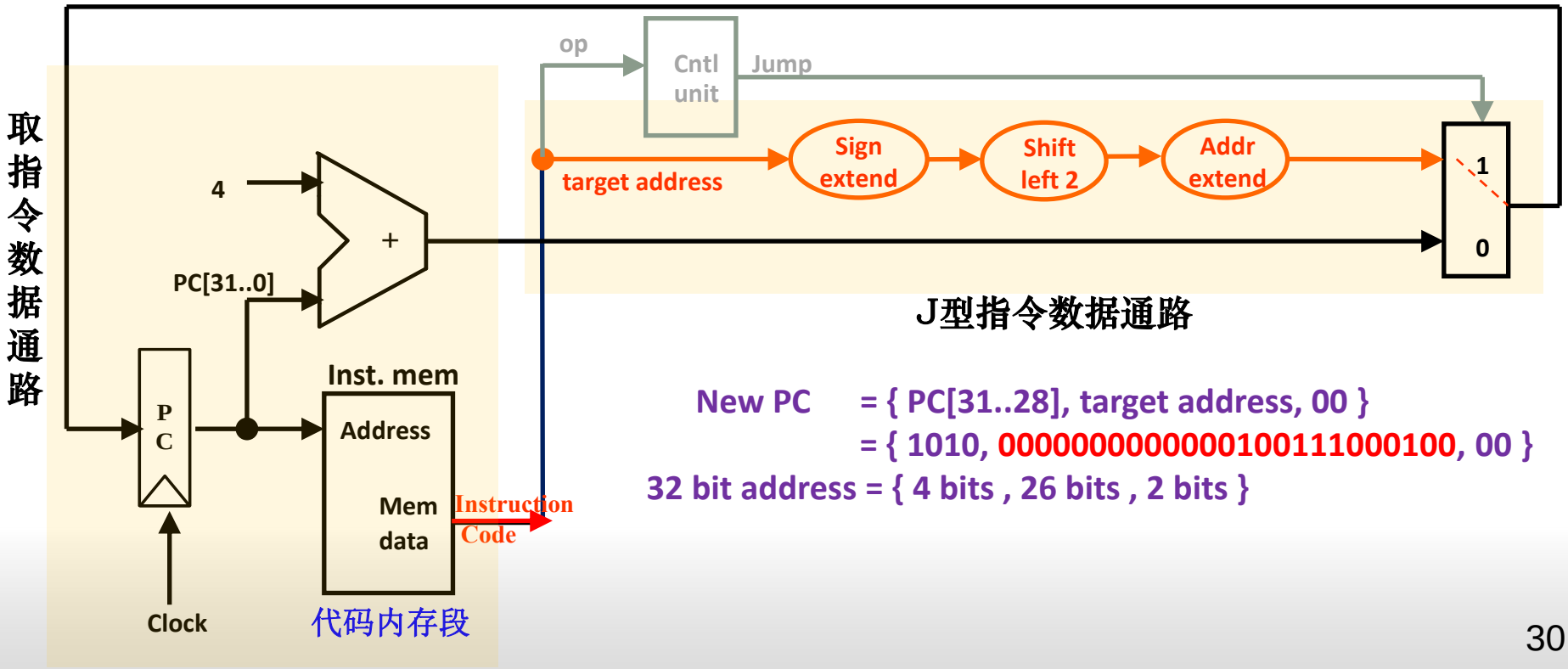
条件转移I指令：ALU的输入来源于两个源寄存器；ALU的判零输出到控制逻辑电路（选择顺序或跳跃）



系统指令数据通路设计

单条指令的数据通路设计：32位MIPS系统J指令

J-format j target address # PC = target address*4	
Opcode (6 bits)	Immediate (26 bits)



J型指令数据通路

New PC = { PC[31..28], target address, 00 }

= { 1010, 00000000000000100111000100, 00 }

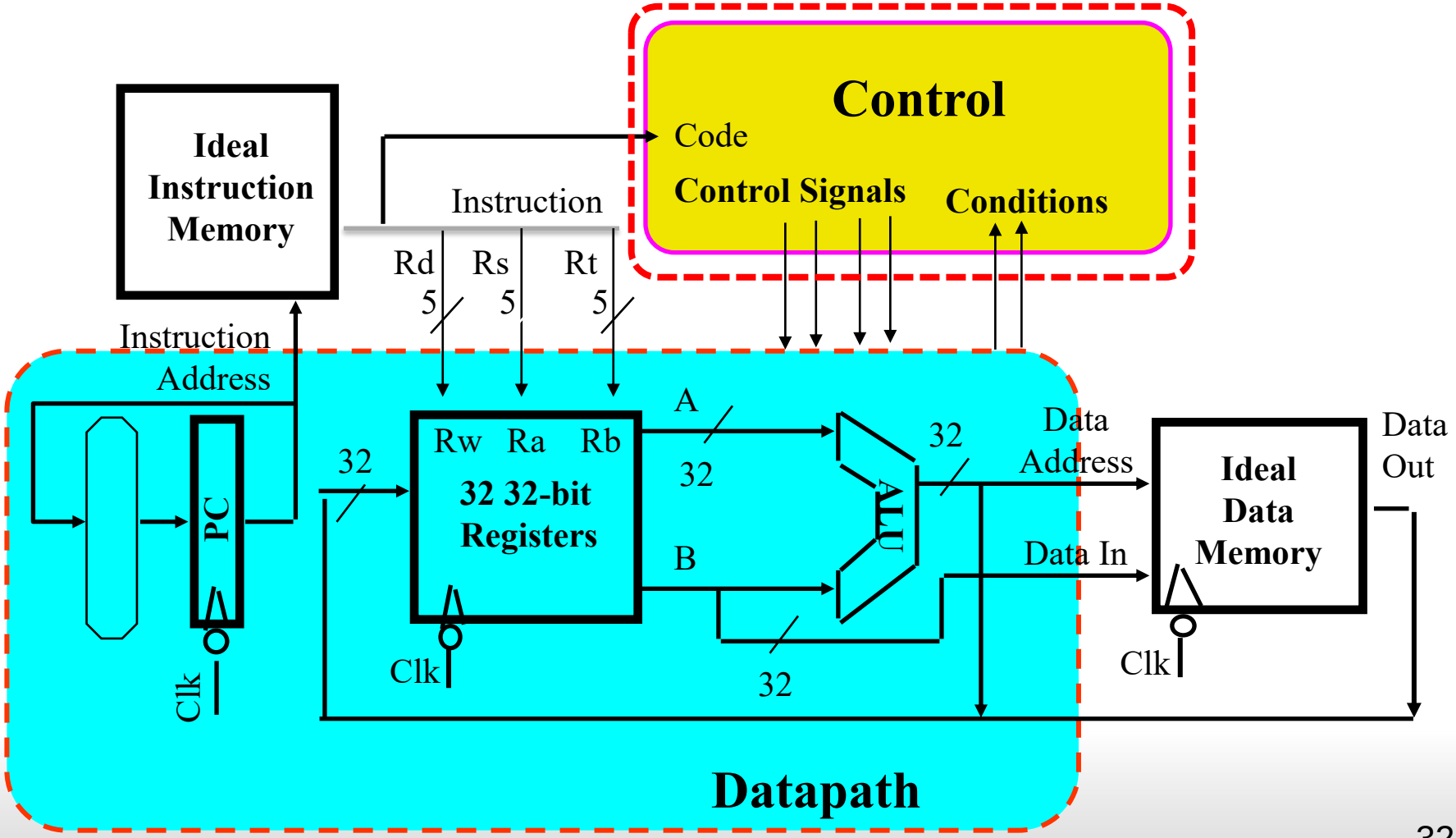
32 bit address = { 4 bits , 26 bits , 2 bits }

提纲

- CPU设计基本步骤
- 系统指令数据通路设计
- **系统指令控制逻辑设计**
- 系统指令数据通路与控制逻辑集成
- CPU时序分析
- CPU性能指标

系统指令控制逻辑设计

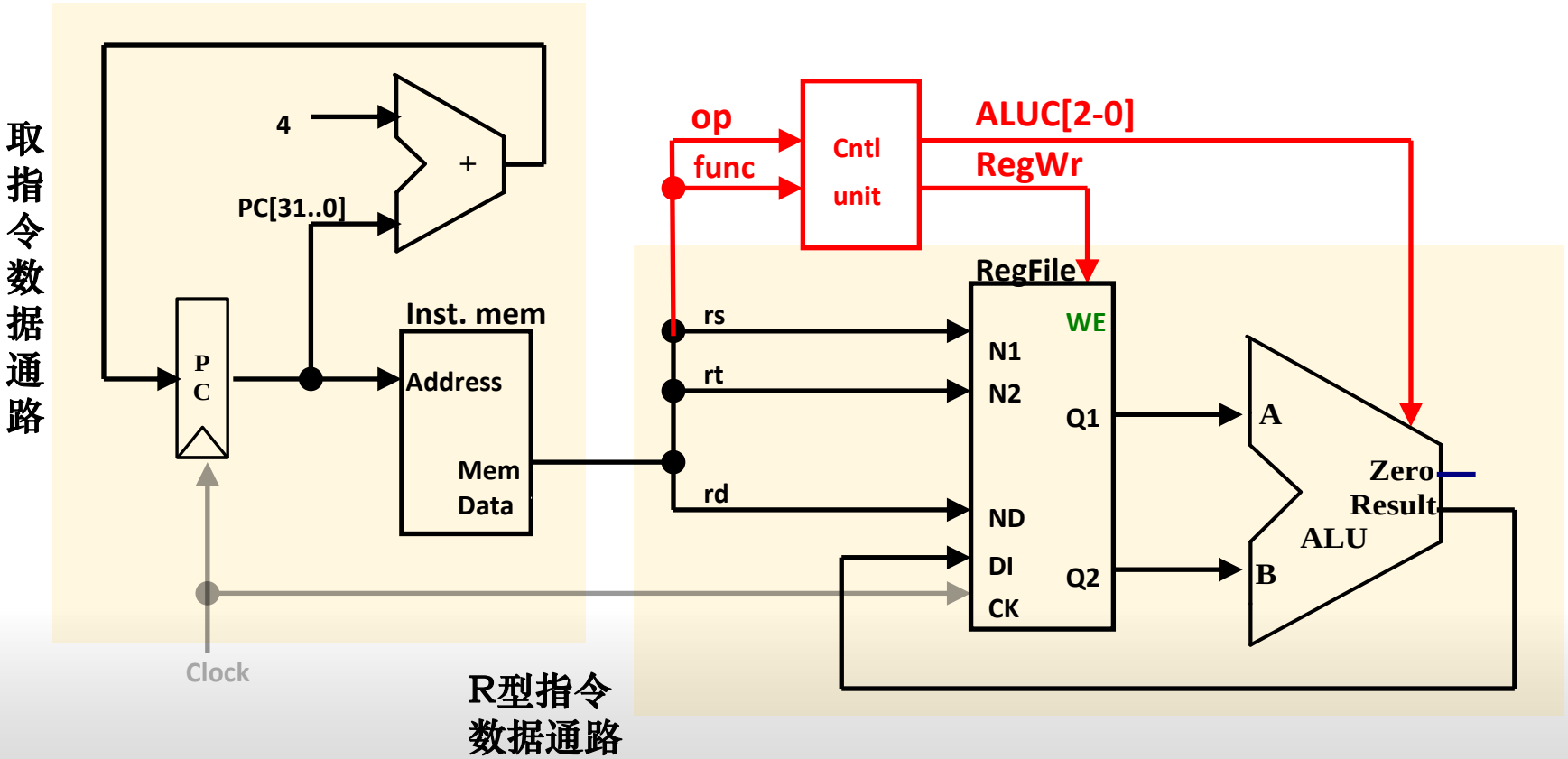
系统指令的控制逻辑设计



系统指令控制逻辑设计

单条指令的控制逻辑设计：32位MIPS系统R指令

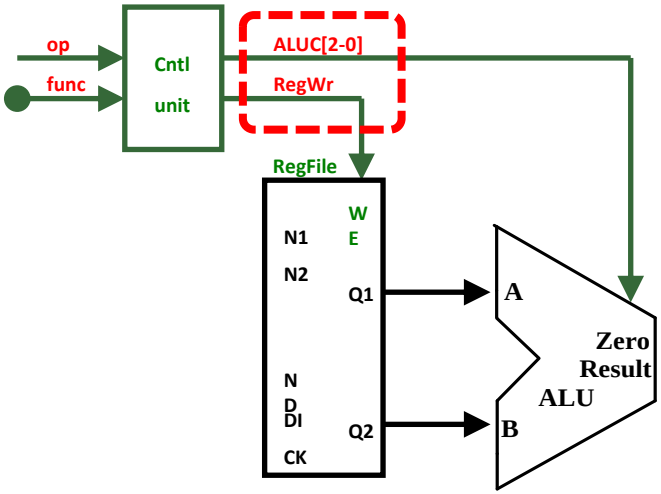
R-format add/sub/and/or/slt rs, rt, rd # rd = rs *op* rt					
Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Rd (5 bits)	Shift amt (5 bits)	Func (6 bits)



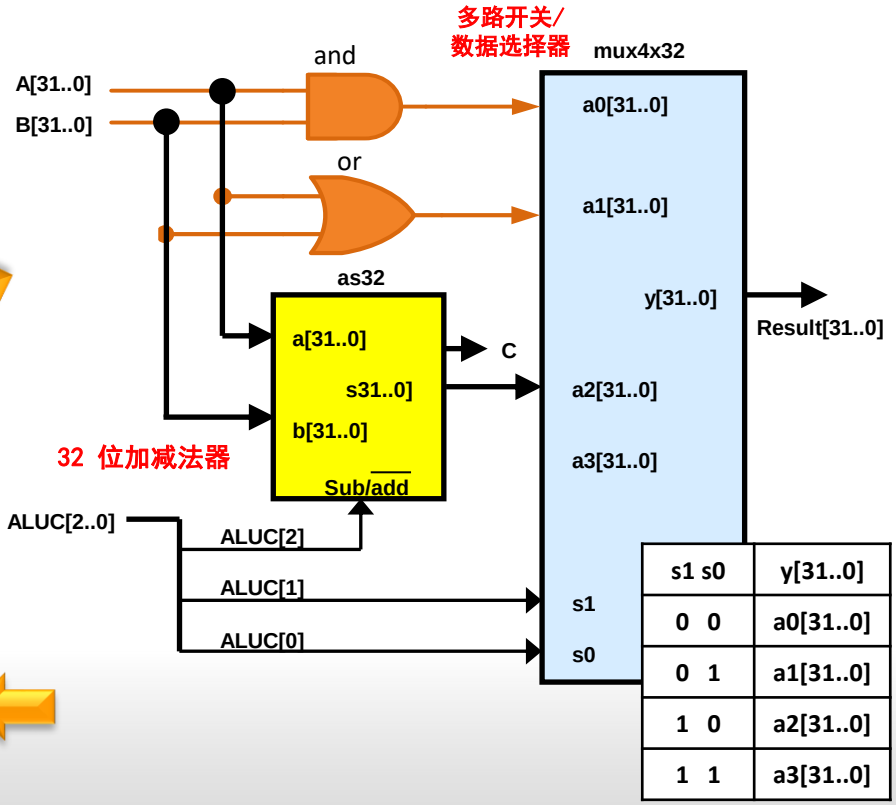
系统指令控制逻辑设计

单条指令的控制逻辑设计：32位MIPS系统R指令

R-format add/sub/and/or/slt rs, rt, rd # rd = rs *op* rt					
Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Rd (5 bits)	Shift amt (5 bits)	Func (6 bits)



op/func	ALUC[2-0]
and	0 00
or	0 01
add	0 10
sub	1 10



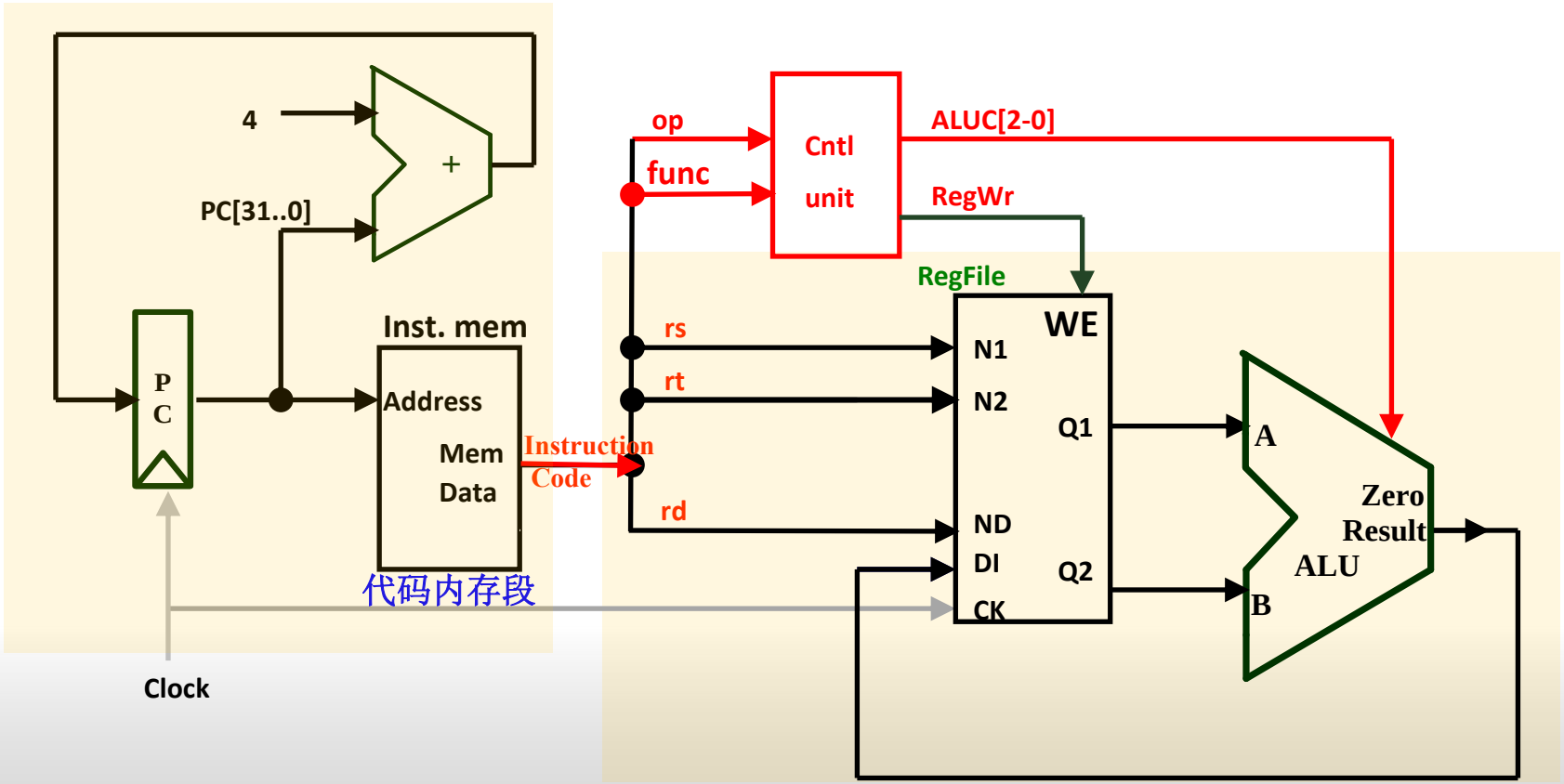
s1 s0	y[31..0]
0 0	a0[31..0]
0 1	a1[31..0]
1 0	a2[31..0]
1 1	a3[31..0]

系统指令控制逻辑设计

数据通路的综合

32位MIPS系统R指令

R-format add/sub/and/or/slt rs, rt, rd # rd = rs *op* rt					
Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Rd (5 bits)	Shift amt (5 bits)	Func (6 bits)

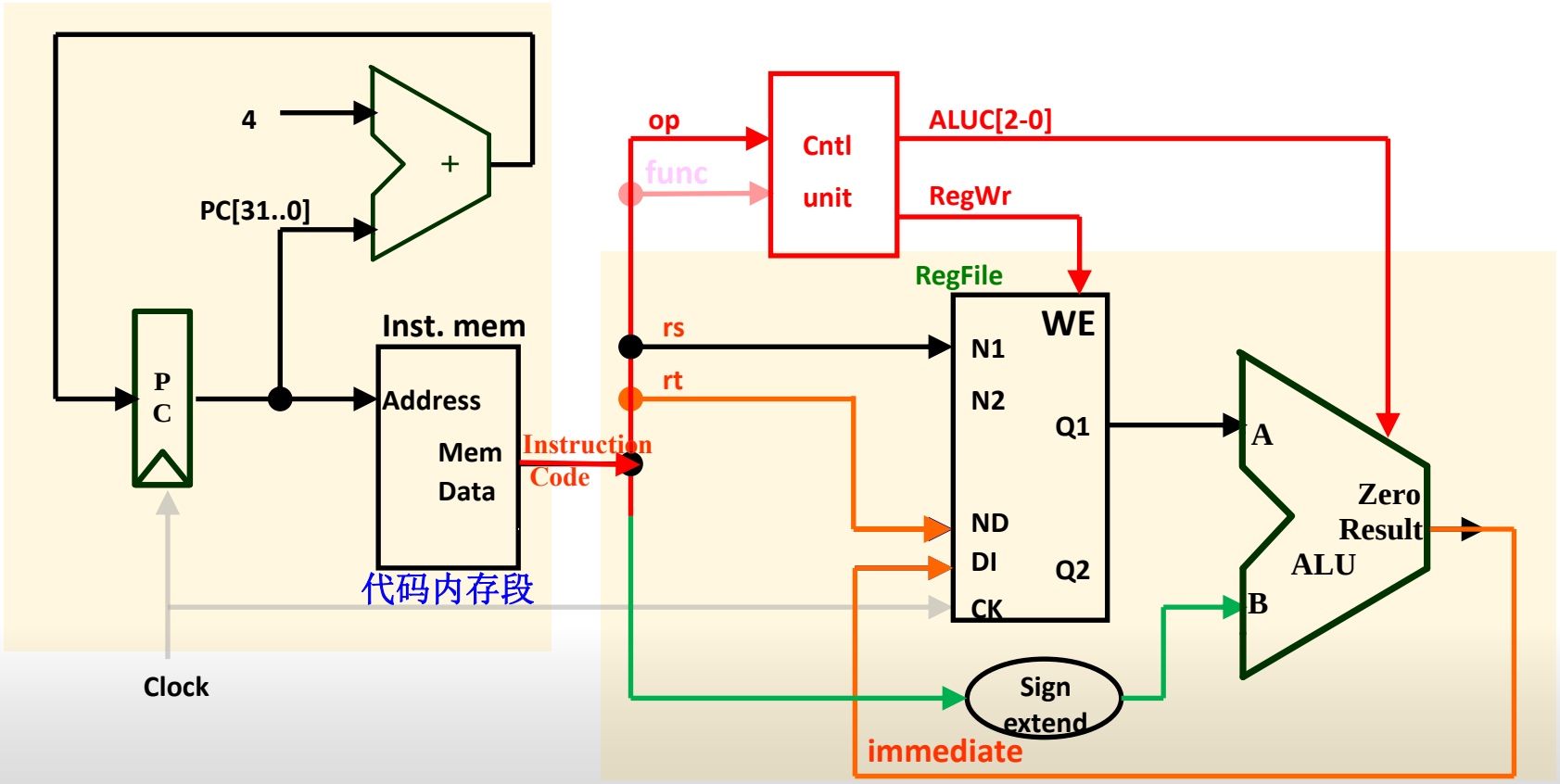


系统指令控制逻辑设计

数据通路的综合

32位MIPS系统I指令

I-format ori rs, rt, imm16			# rt = rs *op* SignExtend[imm16]
Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Immediate (16 bits)



系统指令控制逻辑设计

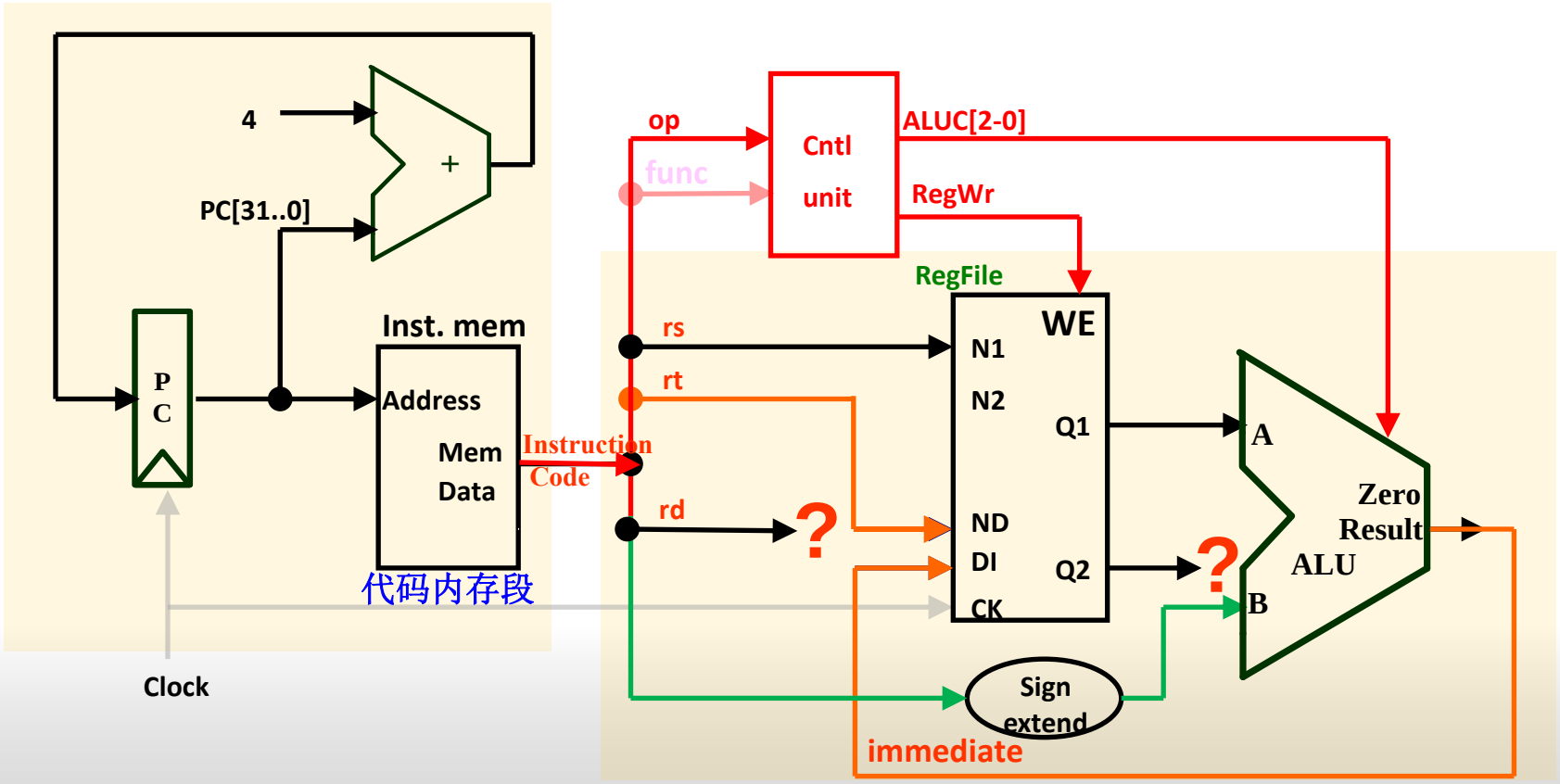
数据通路的综合

不同类型的指令在数据通路上有差别，例如：

R型指令： 3个寄存器操作数（写入寄存器编号为Rd）； ALU的两个输入都是寄存器

I型指令： 2个寄存器操作数（写入寄存器编号为Rt)； ALU中的一个输入为立即数

数据通路综合： 在同一数据通路上运行不同的指令



系统指令控制逻辑设计

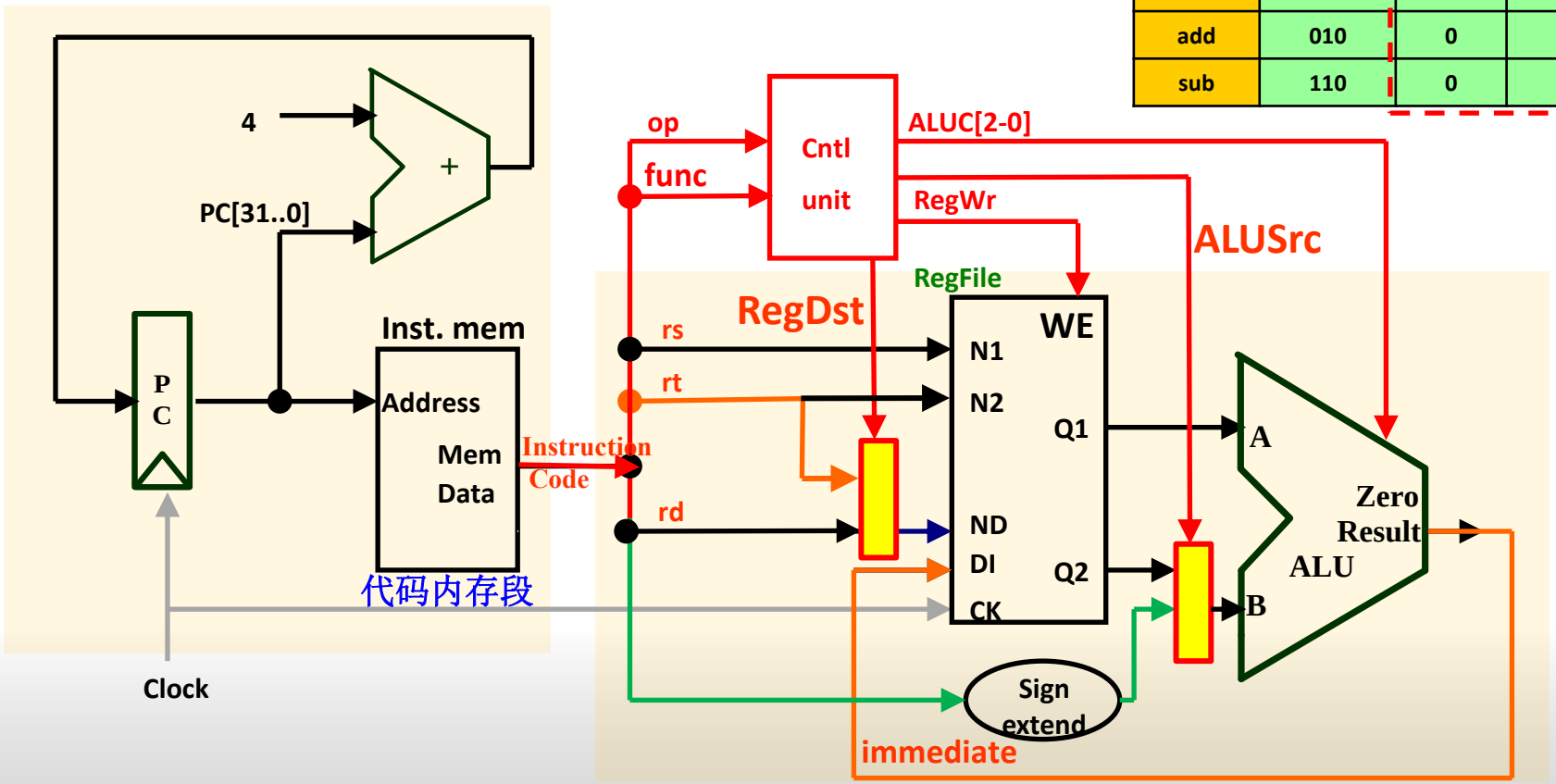
数据通路的综合和控点

对于有多个输入来源的，增加MUX，引入控制信号

RegDst 0: select rd; 1: select rt

ALUsrc 0: select register Q2; 1: select immediate

op/func	ALUC[2-0]	RegDst	ALUSrc
and	000	0	0
or, ori	001	0, 1	0, 1
add	010	0	0
sub	110	0	0

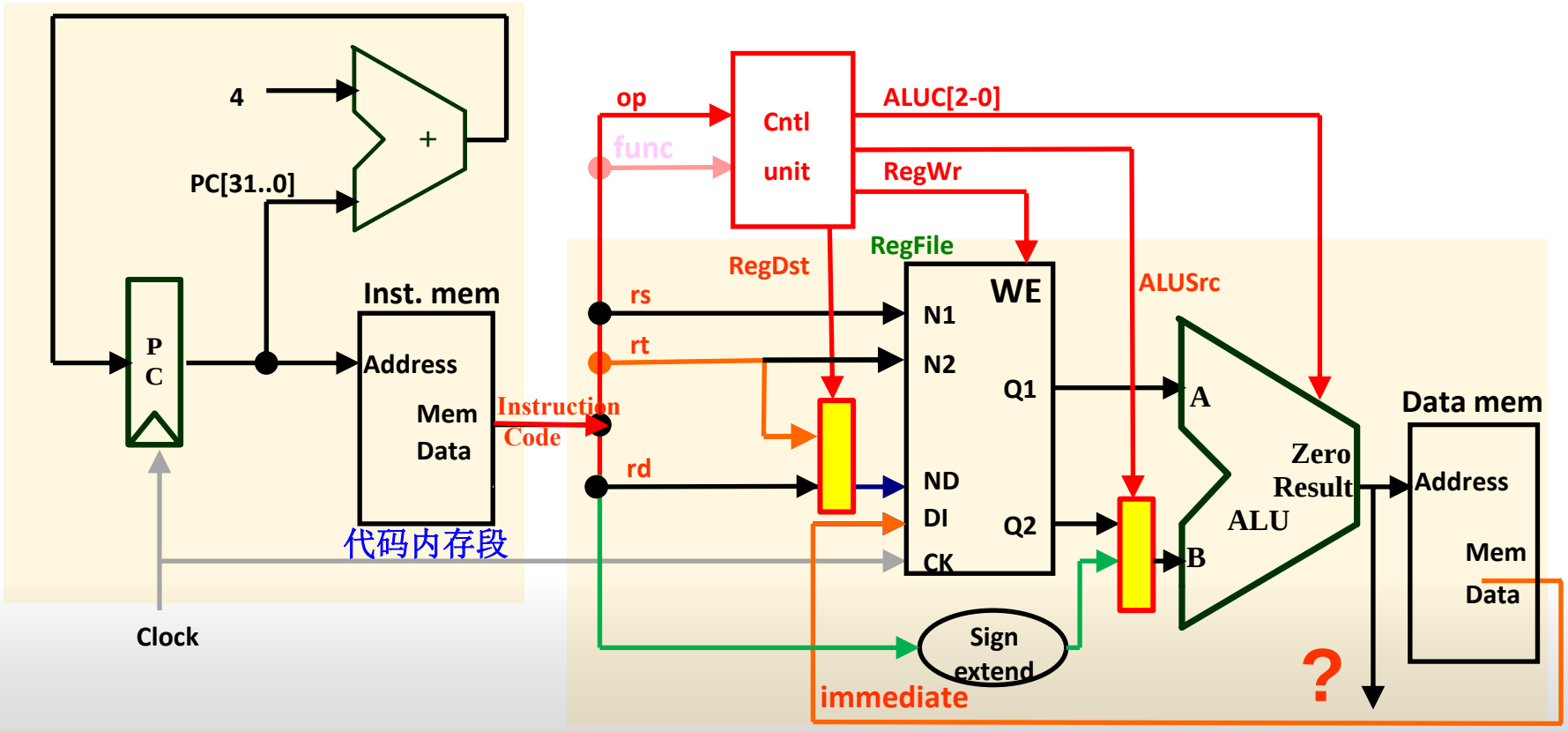


系统指令控制逻辑设计

数据通路的综合和控点

访存型I指令（lw, 读内存）：写回的数据来源于存储器

I-format lw rt, #imm16(rs)			# rt = Memory[rs + imm16]
Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Immediate (16 bits)



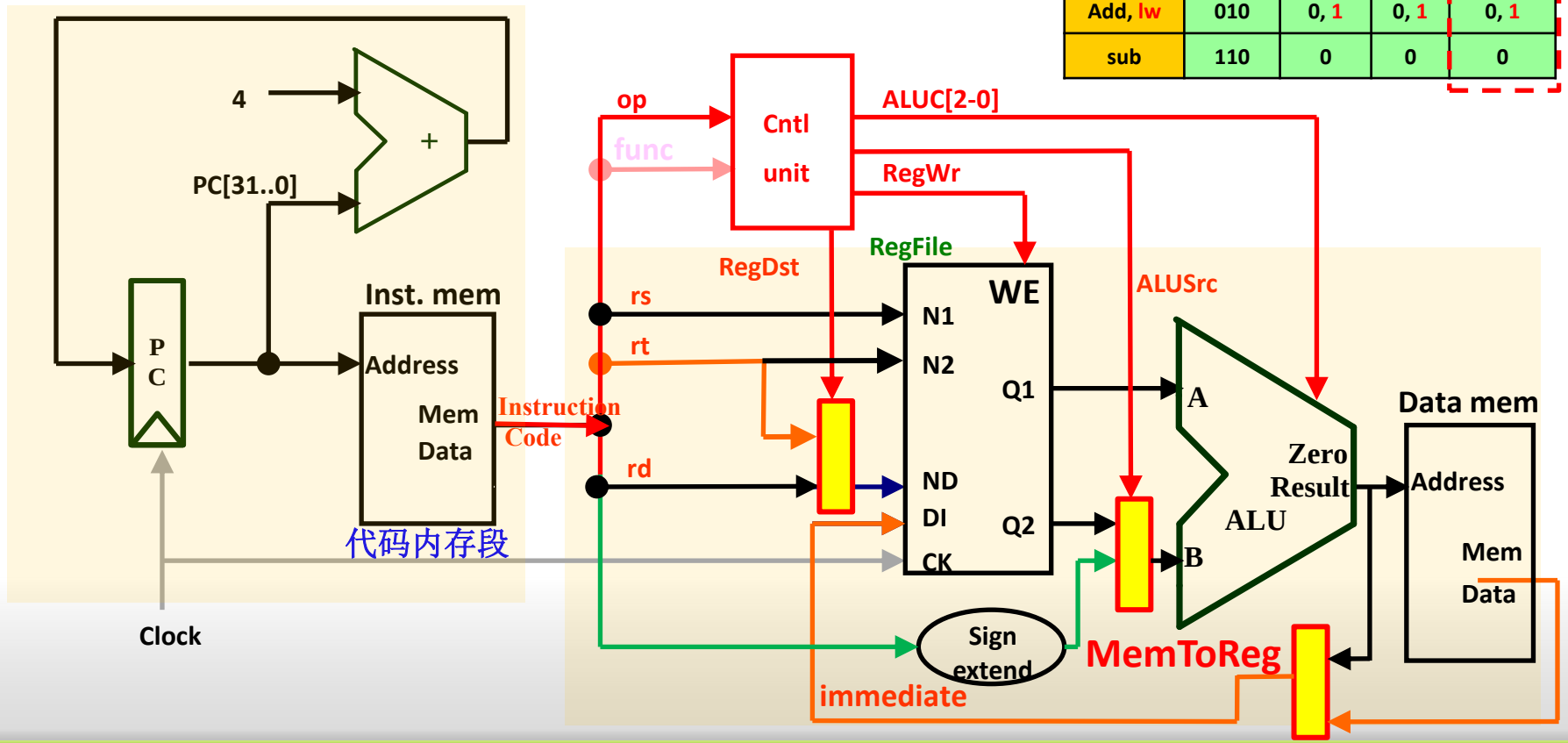
系统指令控制逻辑设计

数据通路的综合和控点

访存型I指令（lw, 读内存）：写回的数据来源于存储器

MemToReg 0: ALU结果写回； 1: 内存写回

op/func	ALUC	Reg Dst	ALU Src	Mem ToReg
and	000	0	0	0
or, ori	001	0, 1	0, 1	0, 0
Add, lw	010	0, 1	0, 1	0, 1
sub	110	0	0	0

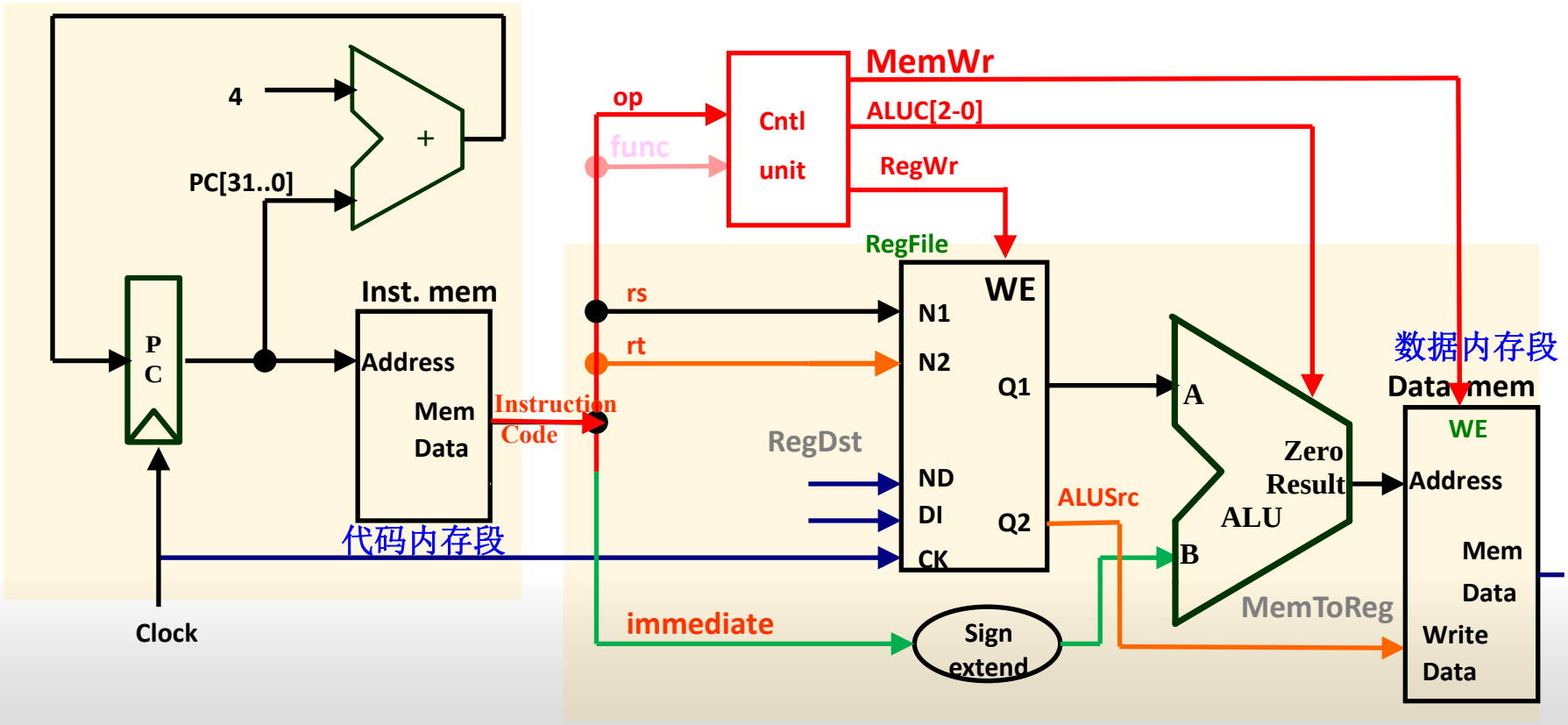


系统指令控制逻辑设计

数据通路的综合和控点

访存型I指令（ sw, 写内存）：寄存器数据写入内存

I-format sw rt, #imm16(rs)			# Memory[rs + imm16] = rt
Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Immediate (16 bits)



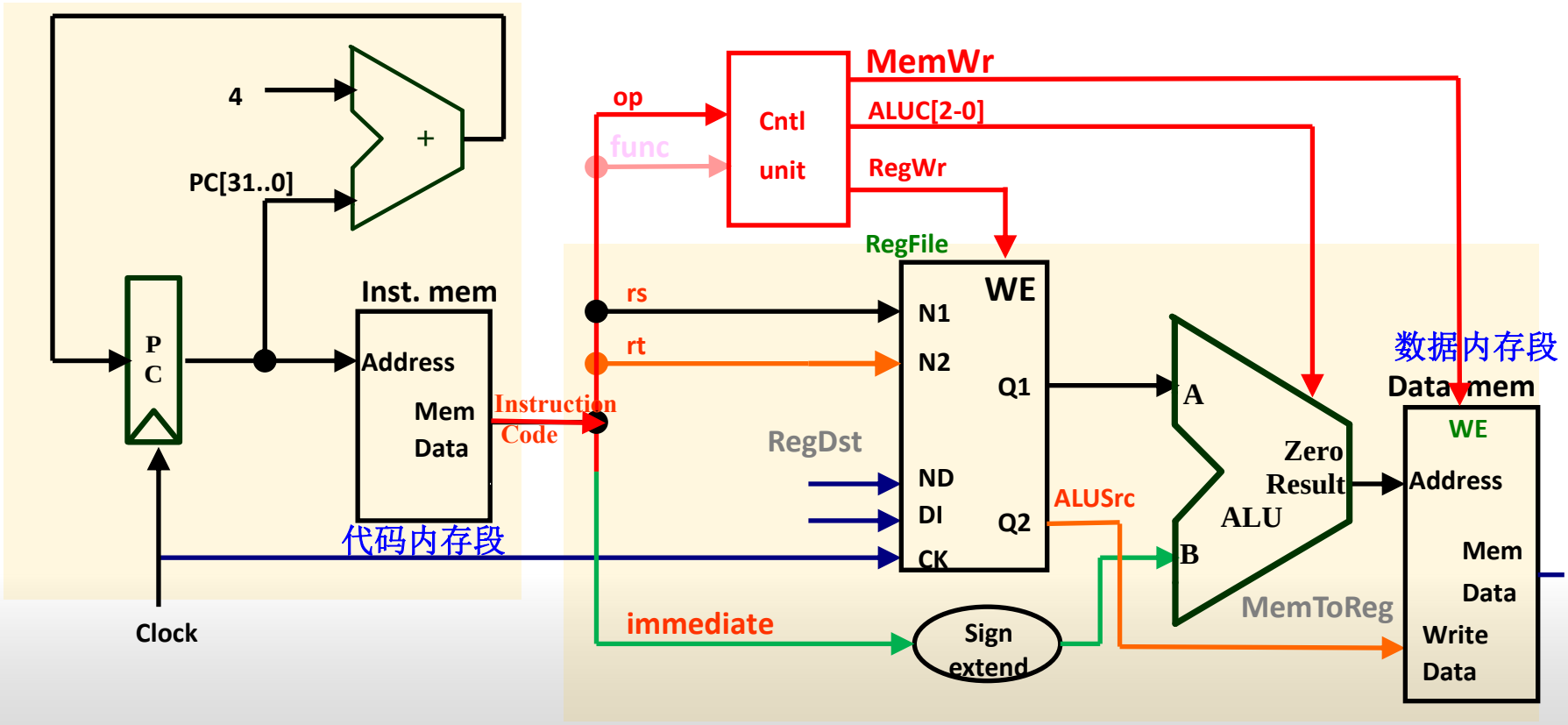
系统指令控制逻辑设计

数据通路的综合和控点

访存型I指令（sw, 写内存）：

寄存器数据写入内存（MemWr)

op/func	ALUC	Reg Dst	ALU Src	Mem ToReg	Mem Wr
and	000	0	0	0	0
or, ori	001	0, 1	0, 1	0, 0	0,0
Add, lw, sw	010	0, 1, X	0, 1, 1	0, 1, X	0,0,1
Sub	110	0	0	0	0

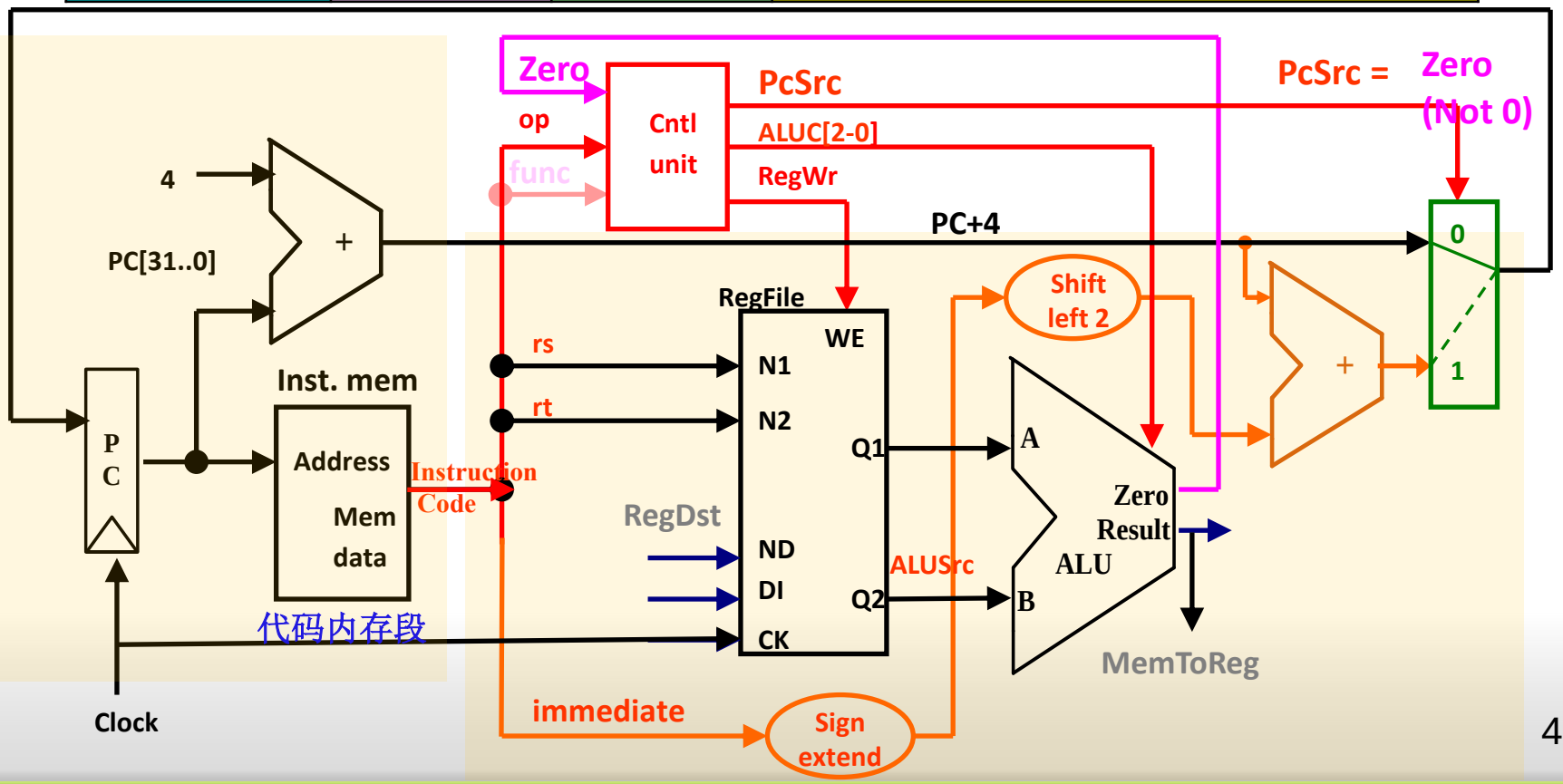


系统指令控制逻辑设计

数据通路的综合和控点

条件转移I指令

I-format beq rs, rt, #imm16			# if rs = rt, goto PC+4+#imm16*4
Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Immediate (16 bits)

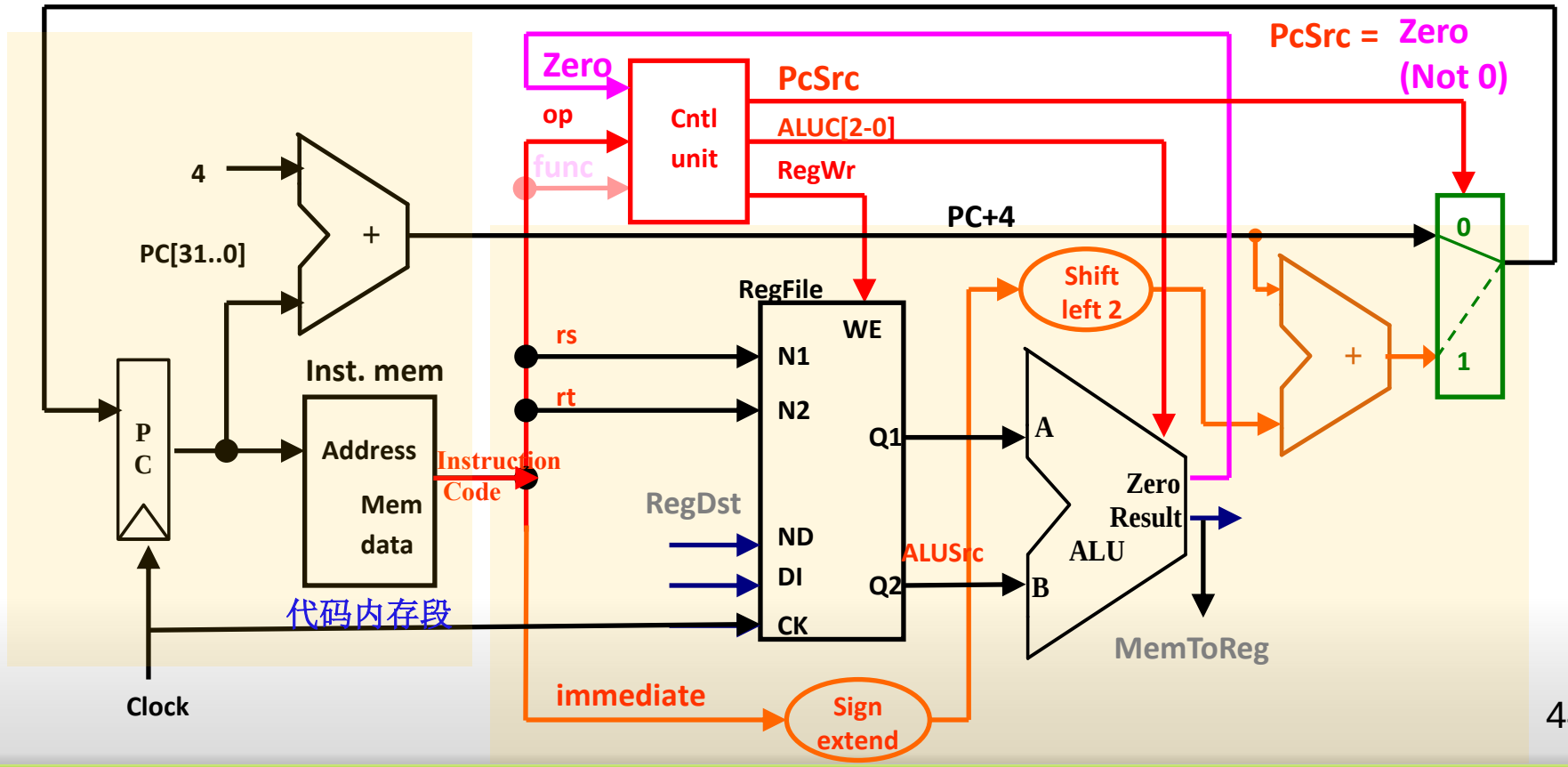


系统指令控制逻辑设计

数据通路的综合和控点

条件转移I指令

op/func	ALUC	Reg Dst	ALU Src	Mem ToReg	MemWr	PcSrc
and	000	0	0	0	0	0
or, ori	001	0, 1	0, 1	0, 0	0,0	0,0
Add, lw, sw	010	0, 1, X	0, 1, 1	0, 1, X	0,0,1	0,0,0
Sub, beq	110	0, X	0, 0	0, X	0, 0	0, Zero

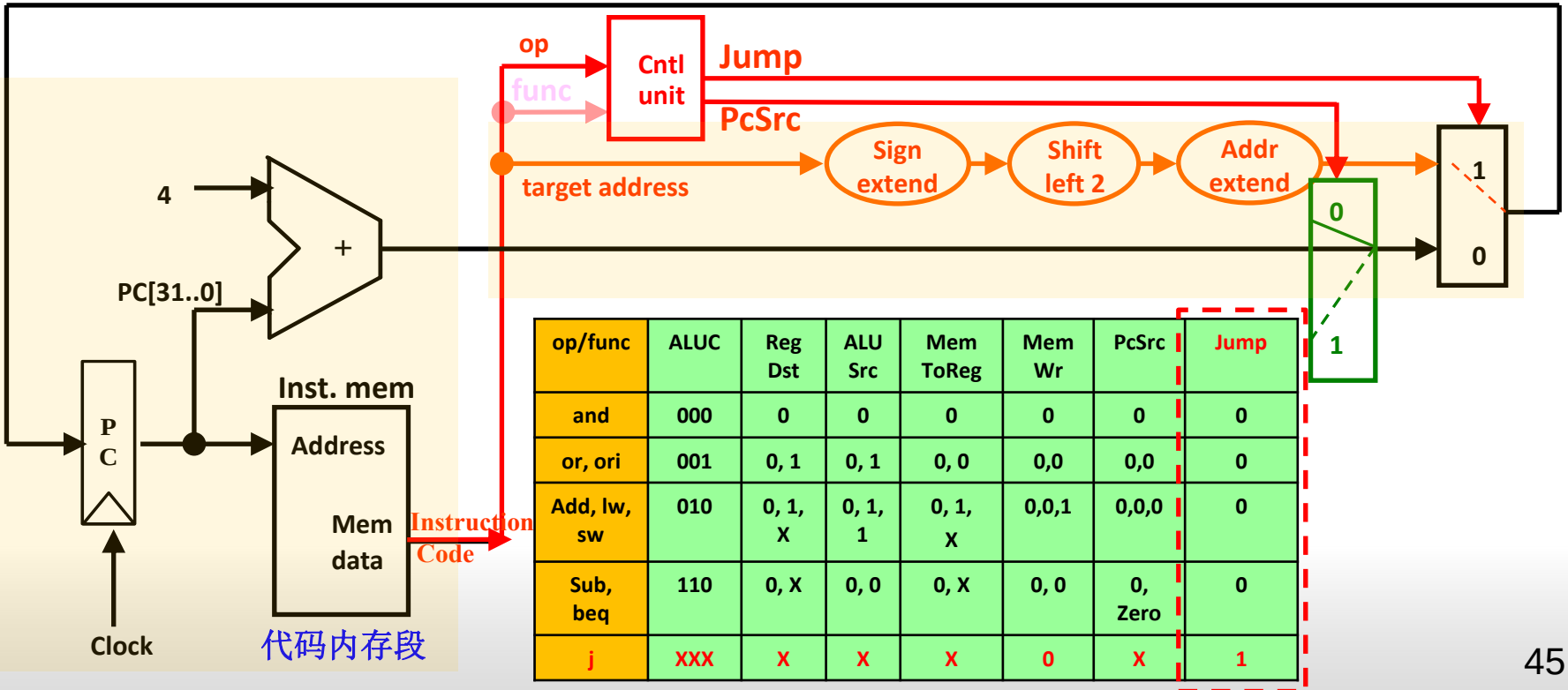


系统指令控制逻辑设计

数据通路的综合和控点

J型指令

J-format j target address # PC = target address*4	
Opcode (6 bits)	Immediate (26 bits)



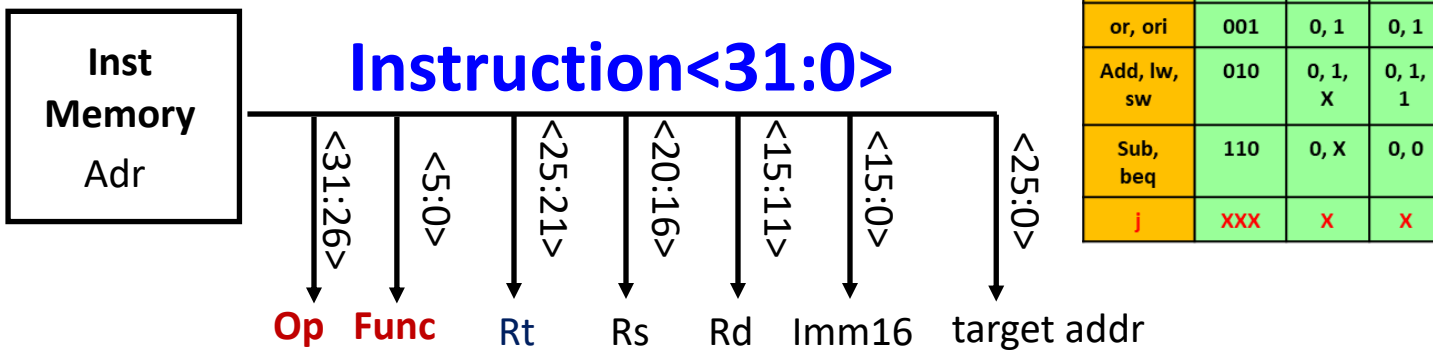
op/func	ALUC	Reg Dst	ALU Src	Mem ToReg	Mem Wr	PcSrc	Jump
and	000	0	0	0	0	0	0
or, ori	001	0, 1	0, 1	0, 0	0,0	0,0	0
Add, lw, sw	010	0, 1, X	0, 1, 1	0, 1, X	0,0,1	0,0,0	0
Sub, beq	110	0, X	0, 0	0, X	0, 0	0, Zero	0
j	XXX	X	X	X	0	X	1

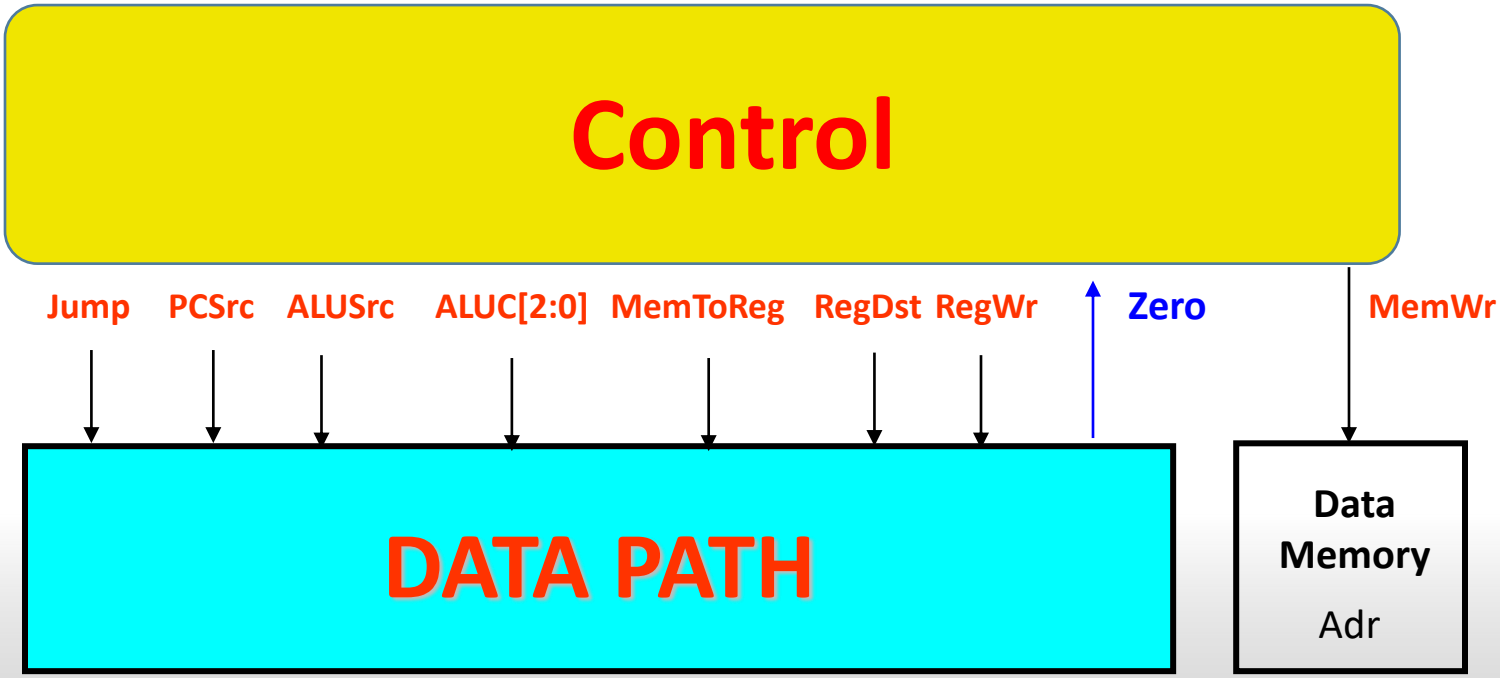
系统指令控制逻辑设计

系统指令的控制逻辑总体实现方案

系统指令集成
(全部指令数据通路+控制逻辑)

op/func	ALUC	Reg Dst	ALU Src	Mem ToReg	Mem Wr	PcSrc	Jump
and	000	0	0	0	0	0	0
or, ori	001	0, 1	0, 1	0, 0	0,0	0,0	0
Add, lw, sw	010	0, 1, X	0, 1, 1	0, 1, X	0,0,1	0,0,0	0
Sub, beq	110	0, X	0, 0	0, X	0, 0	0, Zero	0
j	XXX	X	X	X	0	X	1





系统指令控制逻辑设计

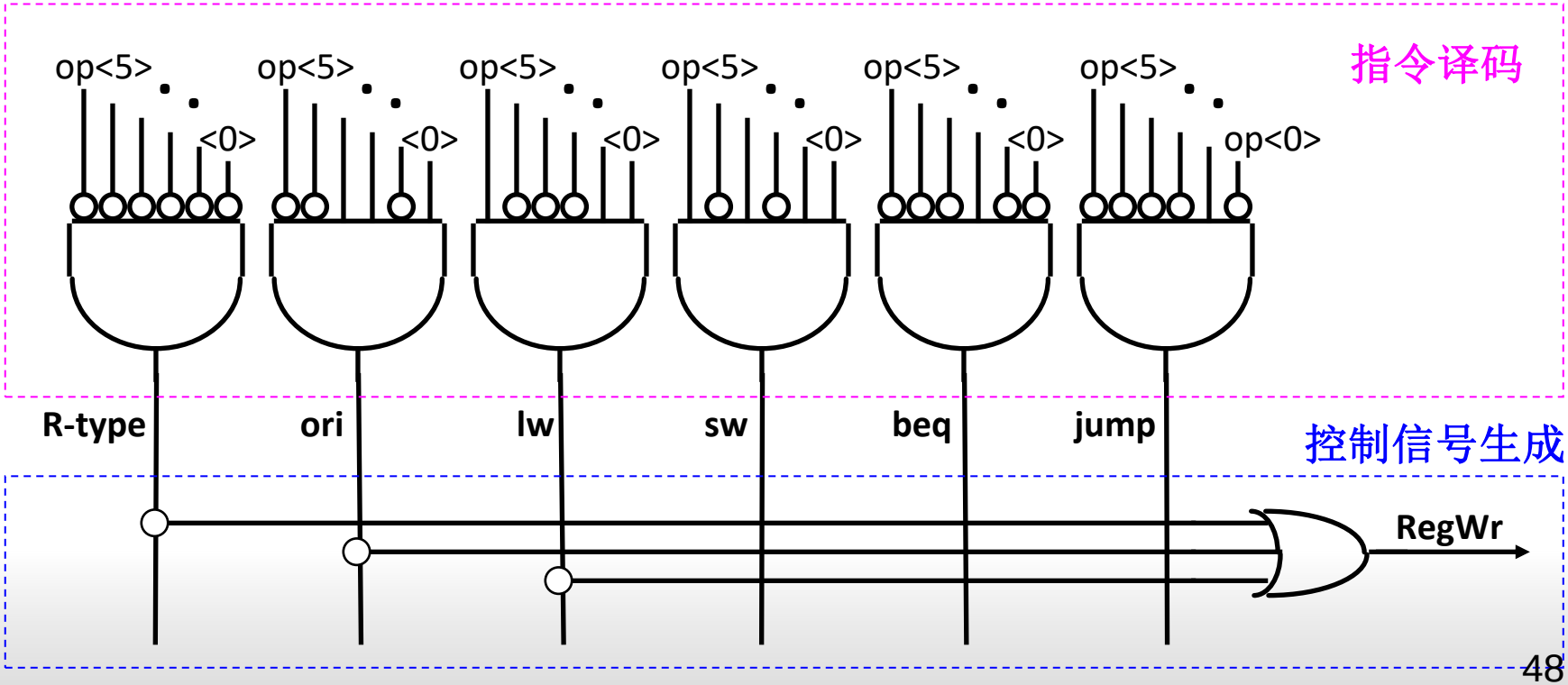
系统指令的控制逻辑集合

Control Signals	Instructions								
	and	or	add	sub	ori	lw	sw	beq	j
Jump	0	0	0	0	0	0	0	0	**1**
PcSrc	0	0	0	0	0	0	0	**Zero**	X
ALUSrc	0	0	0	0	**1**	**1**	**1**	0	X
ALUC[2]	0	0	0	1	0	0	0	1	X
ALUC[1]	0	0	1	1	0	1	1	1	X
ALUC[0]	0	1	0	0	1	0	0	0	X
MemToReg	0	0	0	0	0	**1**	X	X	X
RegDst	0	0	0	0	**1**	**1**	X	X	X
RegWr	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0
MemWr	0	0	0	0	0	0	**1**	0	0

系统指令控制逻辑设计

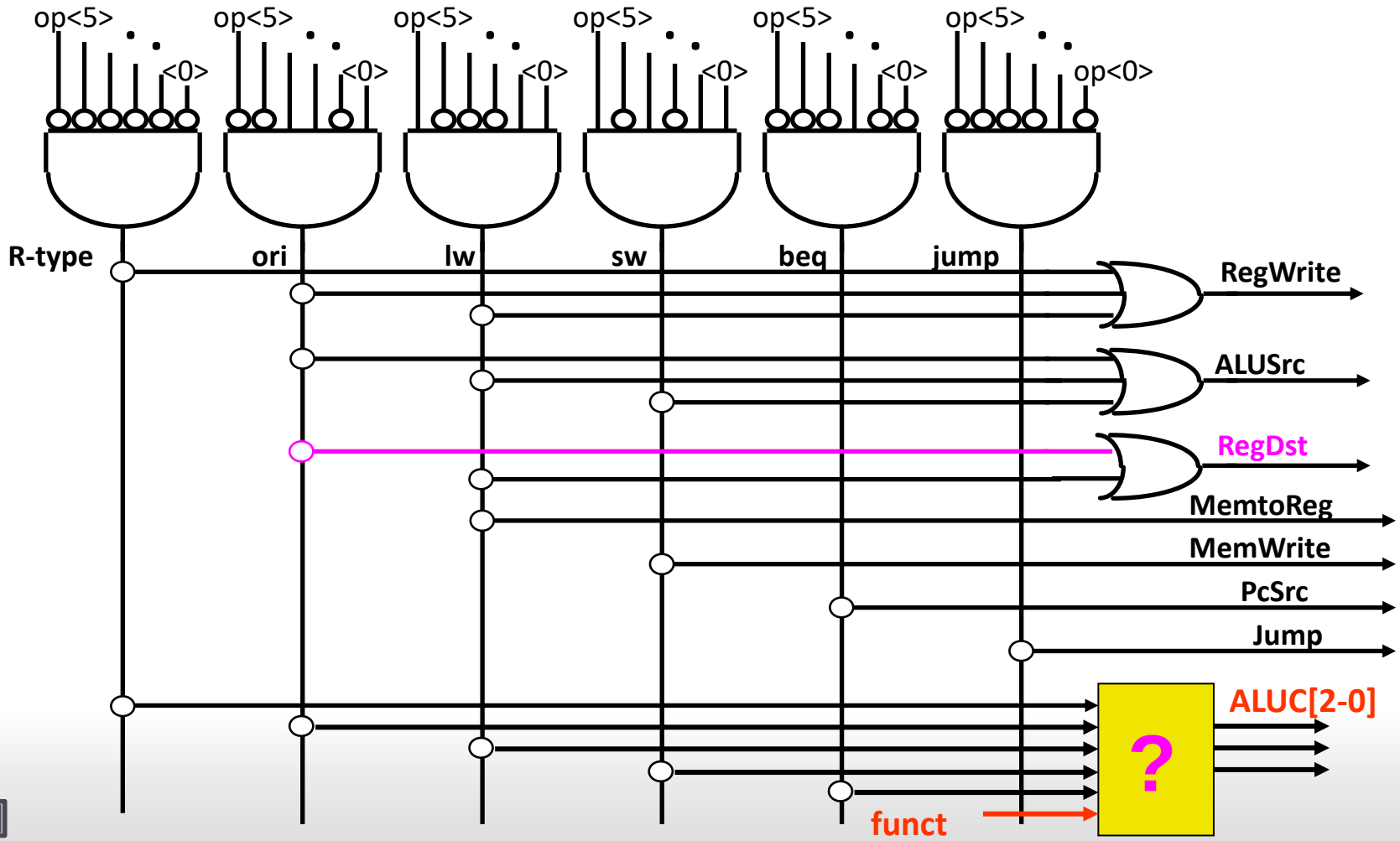
系统指令的控制逻辑实现：RegWr控制信号

value	00 0000	00 1101	10 0011	10 1011	00 0100	00 0010
op	R-type	ori	lw	sw	beq	j
RegWr	1	1	1	0	0	0



系统指令控制逻辑设计

系统指令的控制逻辑实现：控制信号集成



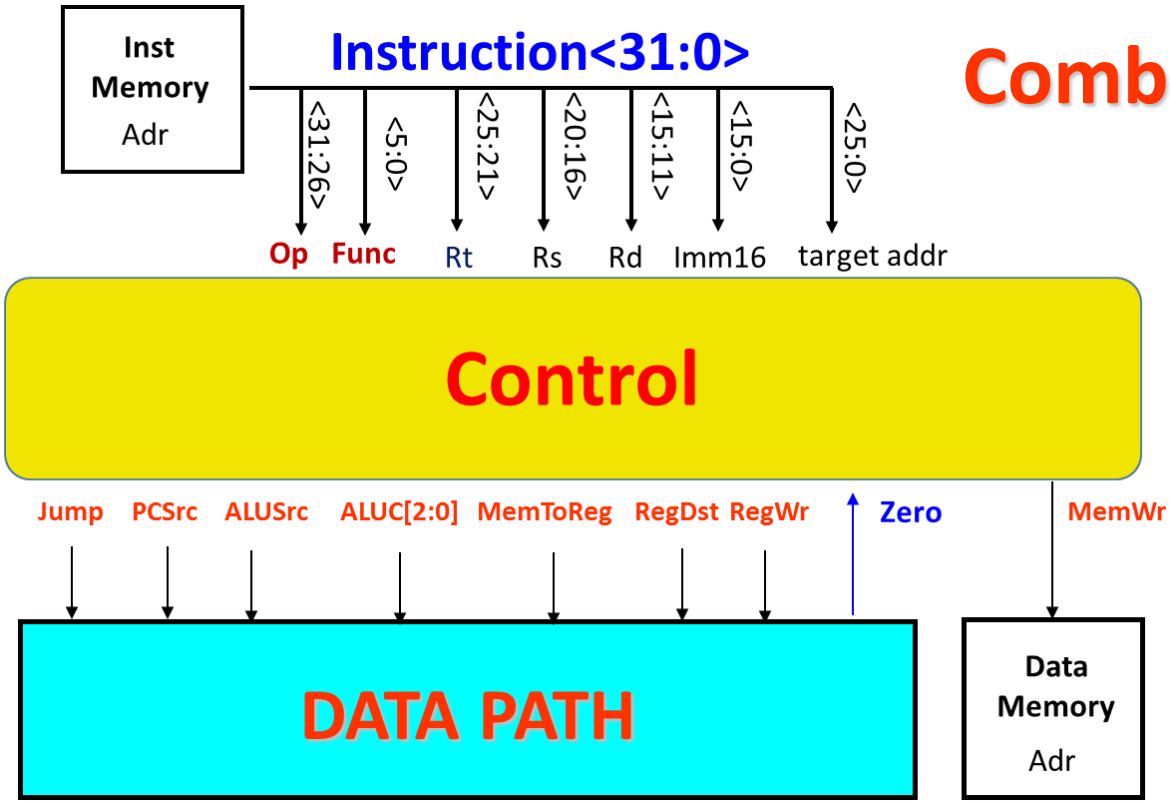
提纲

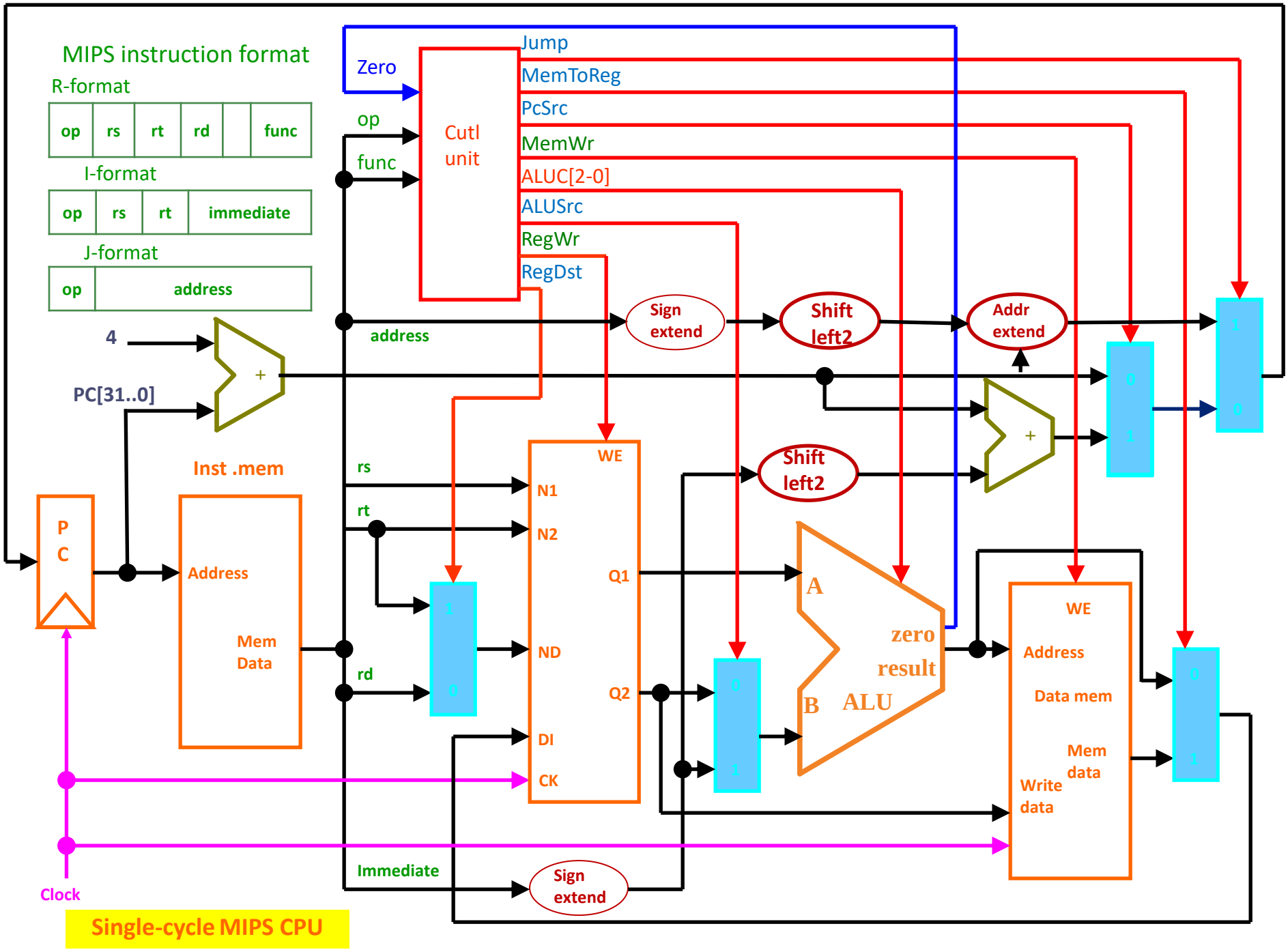
- CPU设计基本步骤
- 系统指令数据通路设计
- 系统指令控制逻辑设计
- **系统指令数据通路与控制逻辑集成**
- CPU时序分析
- CPU性能指标

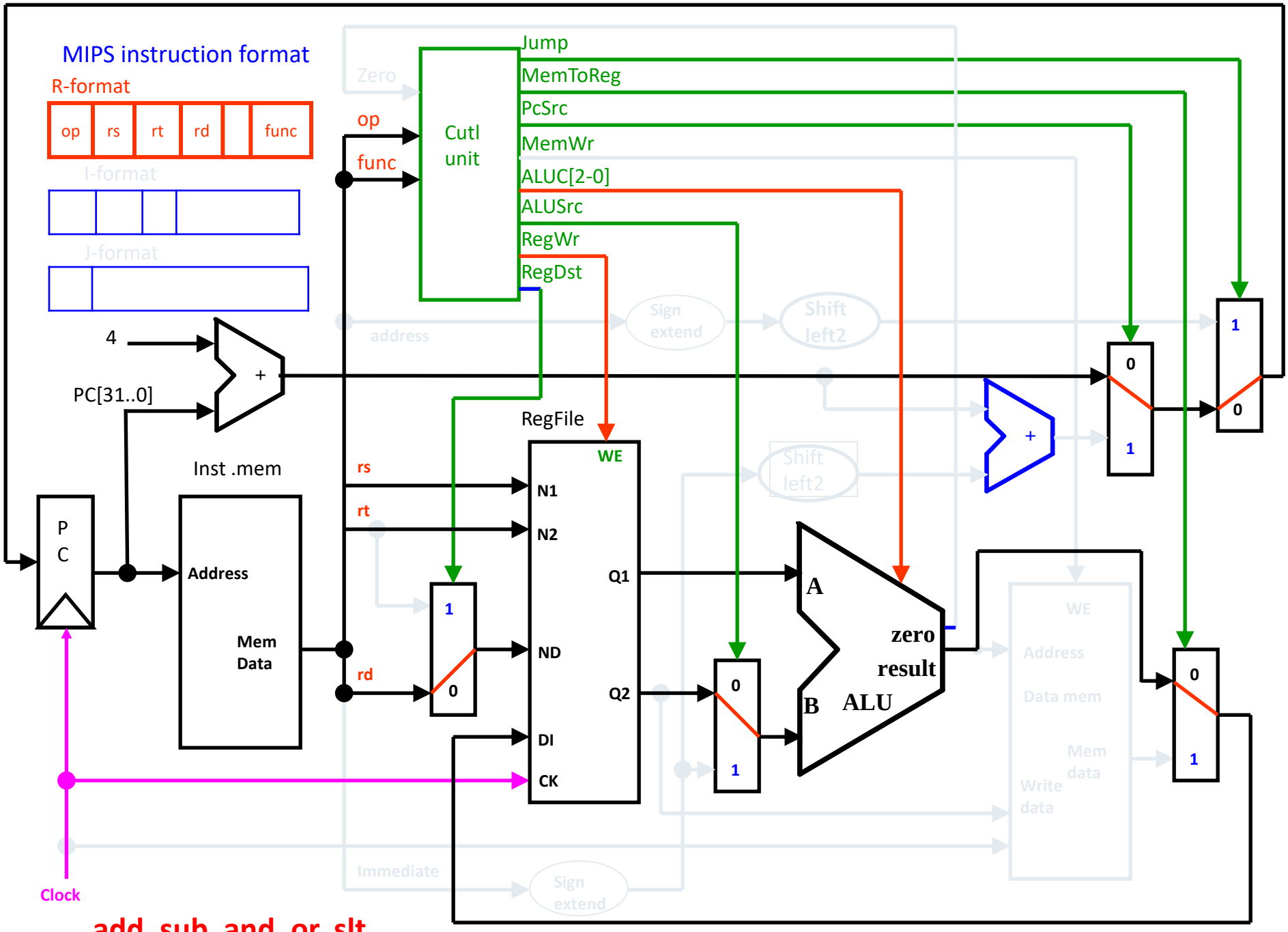
指令系统数据通路与控制逻辑集成

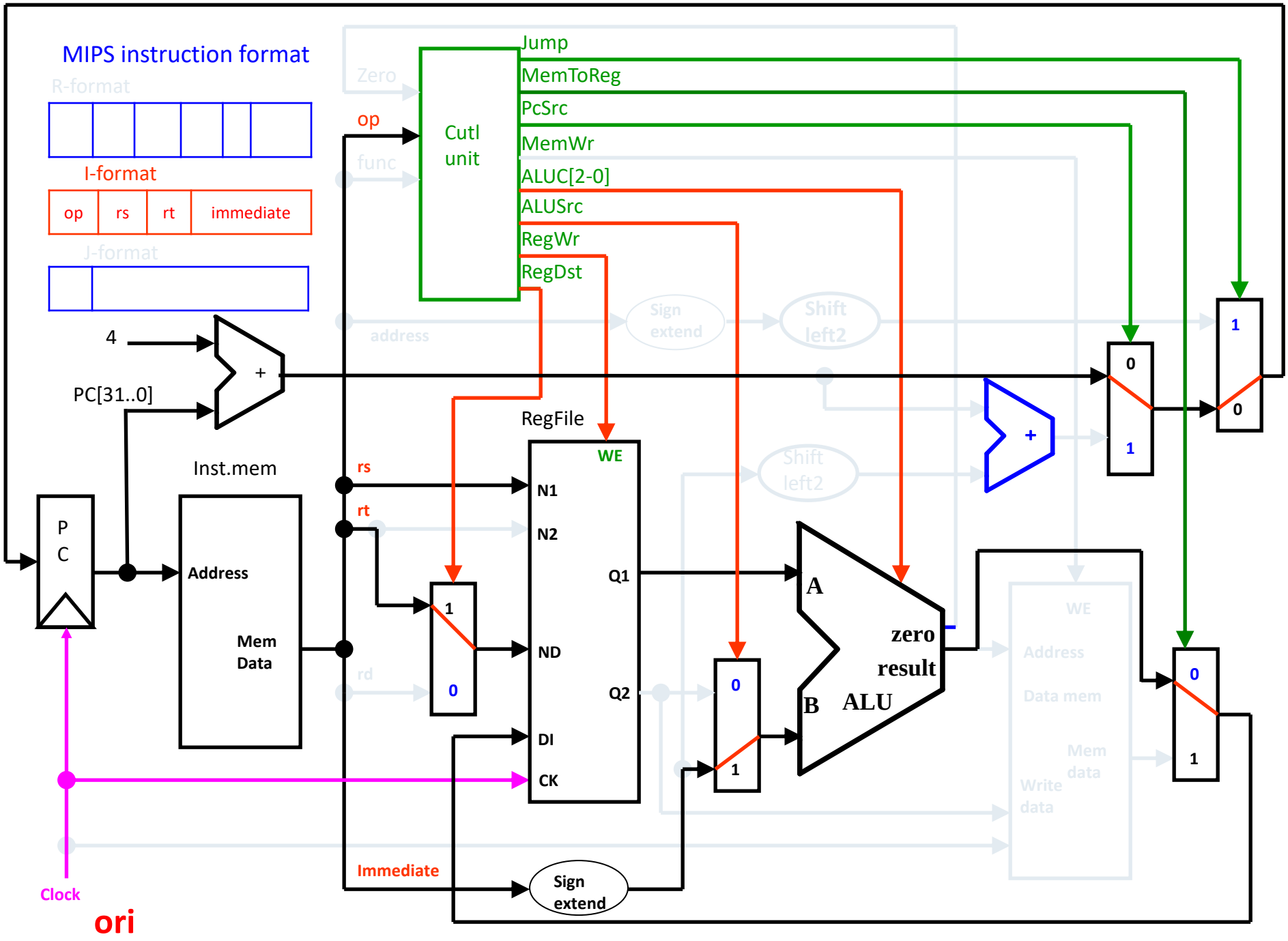
集成（数据通路 + 控制器）

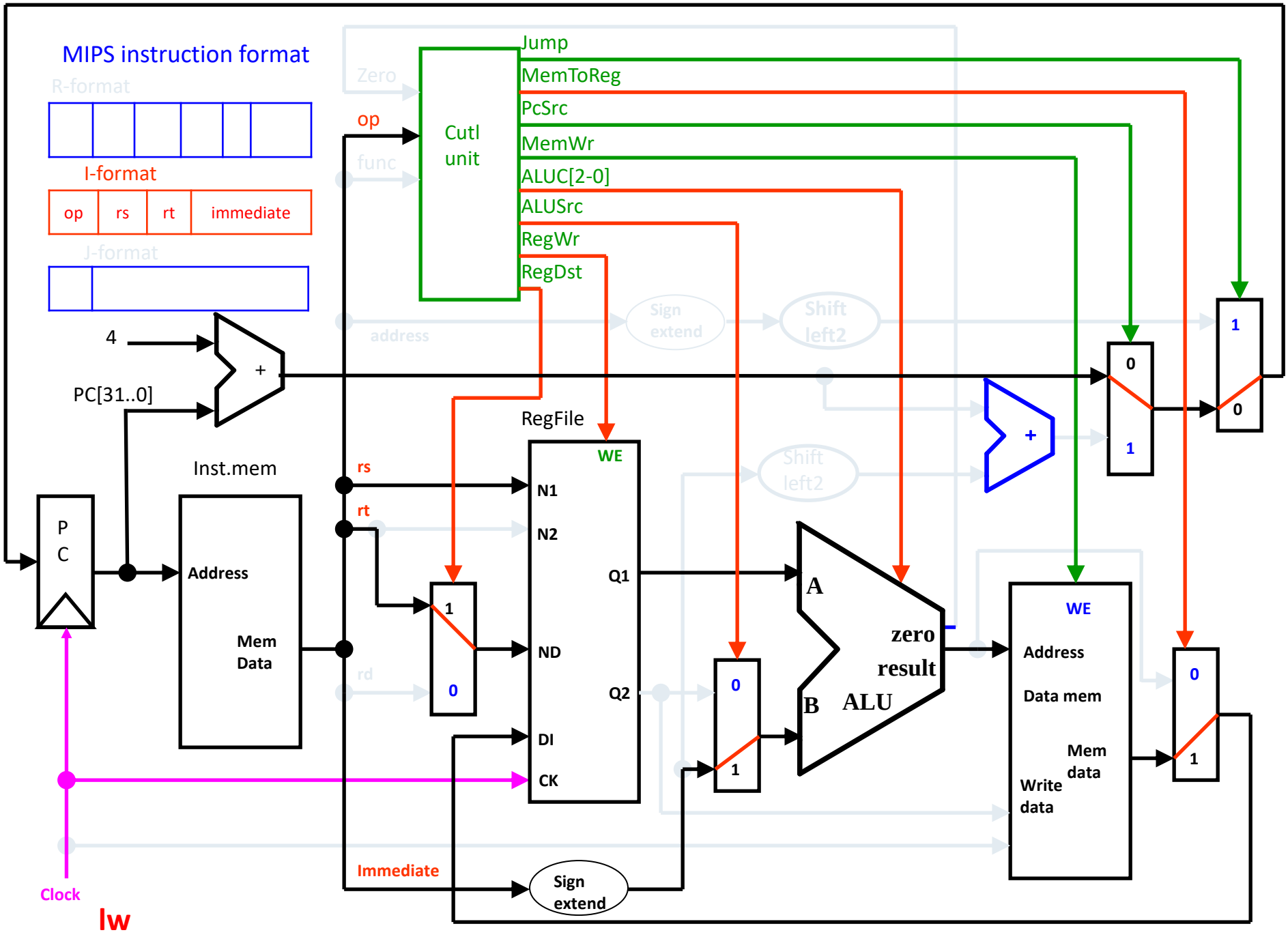
Combine All Together

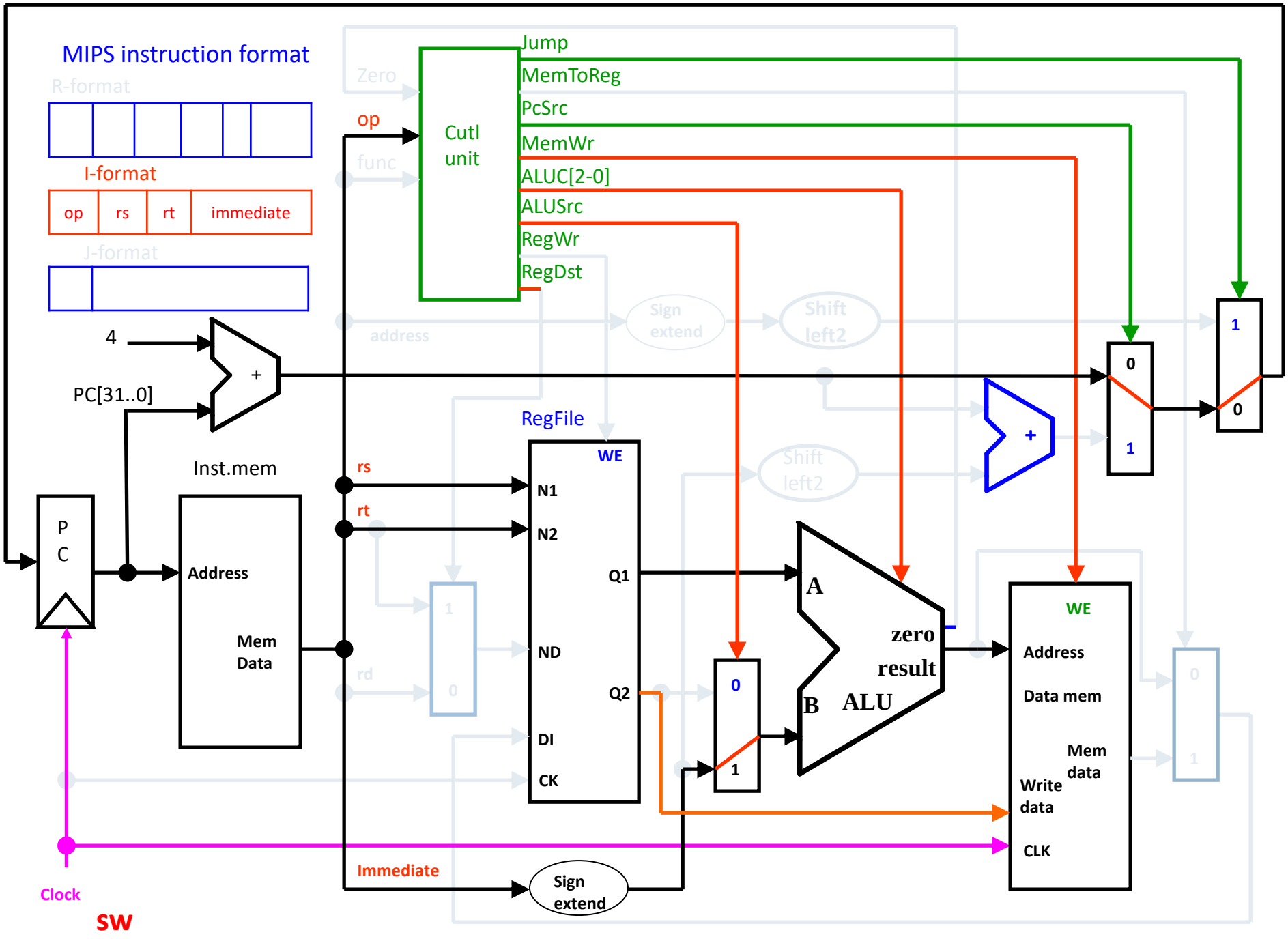


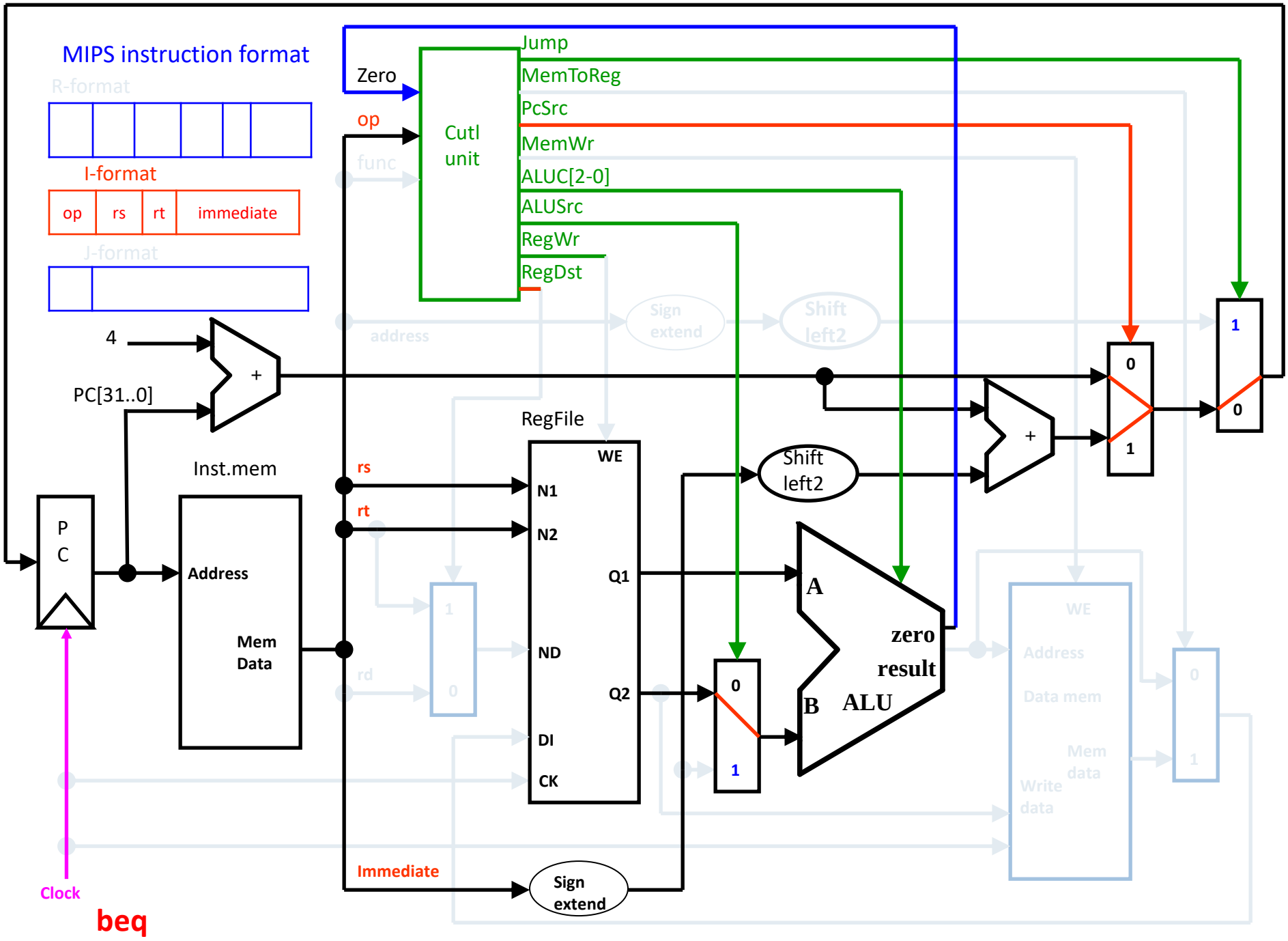

MIPS instruction format
R-format
op
rs
rt
rd
func
I-format
op
rs
rt
immediate
J-format
op
address
Zero
op
func
Cutl unit
Jump
MemToReg
PcSrc
MemWr
ALUC[2-0]
ALUSrc
RegWr
RegDst
Sign extend
Shift left2
Addr extend
address
4
PC[31..0]
+
Inst .mem
P C
Address
Mem Data
rs
rt
rd
WE
N1
N2
ND
DI
CK
Q1
Q2
Shift left2
A
B
ALU
zero
result
WE
Address
Data mem
Mem data
Write data
Immediate
Sign extend
Clock
Single-cycle MIPS CPU

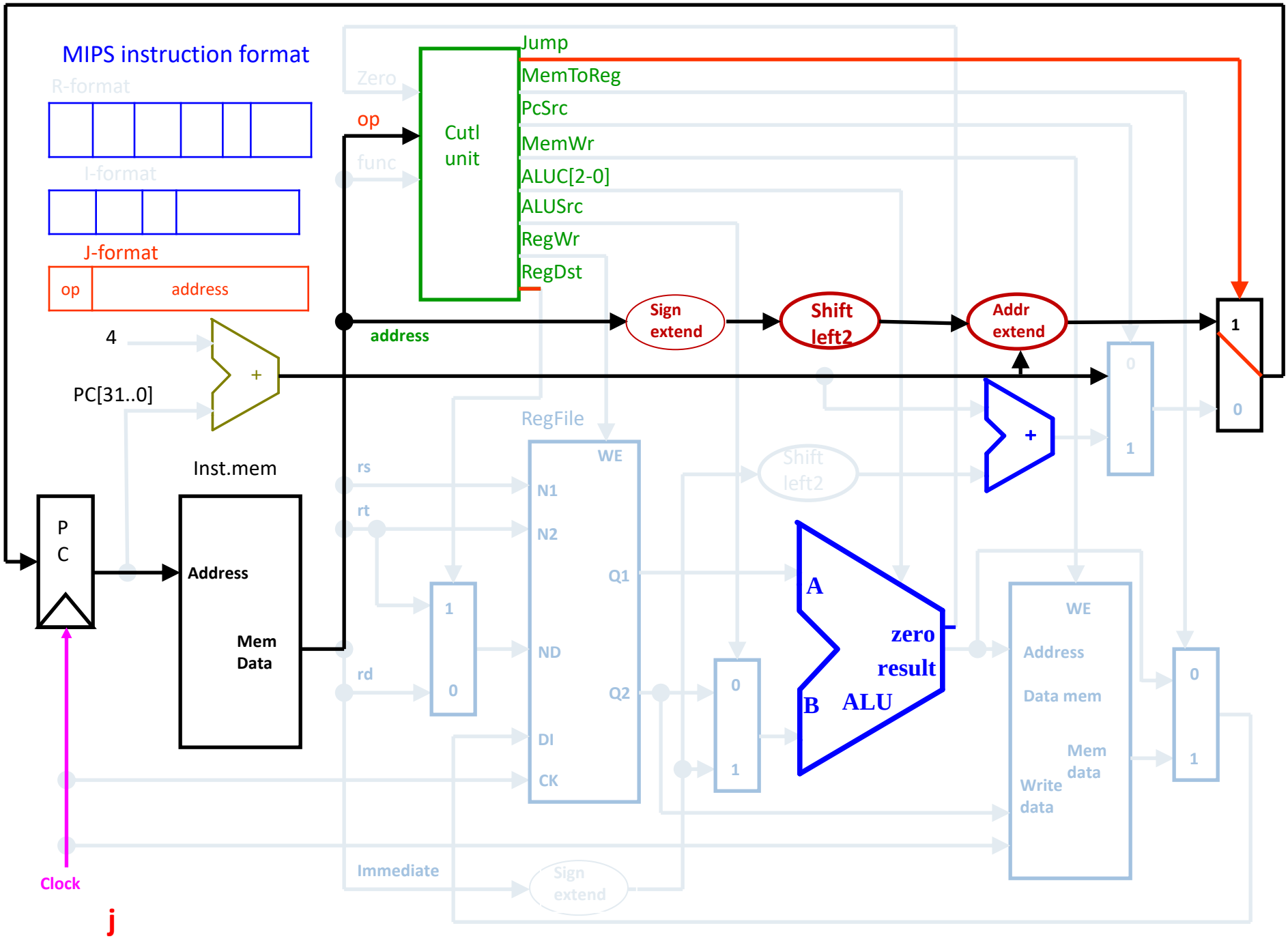

MIPS instruction format
R-format
op
rs
rt
rd
func
I-format
J-format
Zero
op
func
Cutl unit
Jump
MemToReg
PcSrc
MemWr
ALUC[2-0]
ALUSrc
RegWr
RegDst
address
Sign extend
Shift left2
4
PC[31..0]
+
Inst .mem
RegFile
WE
rs
rt
rd
N1
N2
ND
DI
CK
Q1
Q2
P C
Address
Mem Data
A
B
ALU
zero
result
Shift left2
WE
Address
Data mem
Mem data
Write data
Immediate
Sign extend
Clock
add, sub, and, or, slt


MIPS instruction format
R-format
I-format
op
rs
rt
immediate
J-format
Zero
op
func
Cutl unit
Jump
MemToReg
PcSrc
MemWr
ALUC[2-0]
ALUSrc
RegWr
RegDst
address
Sign extend
Shift left2
4
+
PC[31..0]
Inst.mem
RegFile
WE
rs
rt
rd
N1
N2
ND
DI
CK
Q1
Q2
P C
Address
Mem Data
1
0
A
B
ALU
zero
result
Immediate
Sign extend
Clock
ori


MIPS instruction format
R-format
I-format
op
rs
rt
immediate
J-format
Zero
op
func
Cutl unit
Jump
MemToReg
PcSrc
MemWr
ALUC[2-0]
ALUSrc
RegWr
RegDst
address
Sign extend
Shift left2
4
+
PC[31..0]
Inst.mem
P C
Address
Mem Data
RegFile
WE
rs
rt
rd
N1
N2
ND
DI
CK
Q1
Q2
1
0
A
B
ALU
zero
result
Address
Data mem
Write data
Mem data
Immediate
Sign extend
Clock
lw


MIPS instruction format
R-format
I-format
op
rs
rt
immediate
J-format
Zero
op
func
Cutl unit
Jump
MemToReg
PcSrc
MemWr
ALUC[2-0]
ALUSrc
RegWr
RegDst
address
Sign extend
Shift left2
4
PC[31..0]
+
0
1
RegFile
WE
rs
rt
N1
N2
Q1
ND
Q2
DI
CK
Inst.mem
P C
Address
Mem Data
rd
A
B
ALU
zero
result
WE
Address
Data mem
Mem data
Write data
CLK
Immediate
Sign extend
Clock
SW


MIPS instruction format
R-format
I-format
op
rs
rt
immediate
J-format
Zero
op
func
Cutl unit
Jump
MemToReg
PcSrc
MemWr
ALUC[2-0]
ALUSrc
RegWr
RegDst
4
PC[31..0]
+
address
Sign extend
Shift left2
1
0
Inst.mem
RegFile
WE
rs
rt
N1
N2
Q1
ND
Q2
DI
CK
Address
Mem Data
P C
A
B
ALU
zero
result
rd
Immediate
Sign extend
Clock
beq
Data mem
Write data
Mem data


MIPS instruction format
R-format
I-format
J-format
op
address
Zero
op
func
Cutl unit
Jump
MemToReg
PcSrc
MemWr
ALUC[2-0]
ALUSrc
RegWr
RegDst
4
PC[31..0]
+
address
Sign extend
Shift left2
Addr extend
1
0
RegFile
WE
rs
rt
rd
N1
N2
ND
DI
CK
Q1
Q2
Inst.mem
P C
Address
Mem Data
A
B
ALU
zero
result
Address
Data mem
Mem data
Write data
Immediate
Sign extend
Clock
j

课后作业：

复习：

《计算机组成原理》第8章《中央处理器》

预习：

《计算机组成原理》第1.4章节《计算机的性能指标》、第10章《运算方法与运算器》

思考题：

《计算机组成原理》 P225页习题 8.1、8.3题

谢 谢

请不要将课件上传到公共网络平台上~~