处理器体系结构

第四章 处理器的微架构A

--单周期MIPS处理器

(Micro-processor Architecture)

目录

*目标

题目：设计一个可实现以下指令的MIPS处理器，并对其进行性能优化

存取指令 lw、sw

算术逻辑指令 add、sub、and、or、slt

跳转指令 beq、j

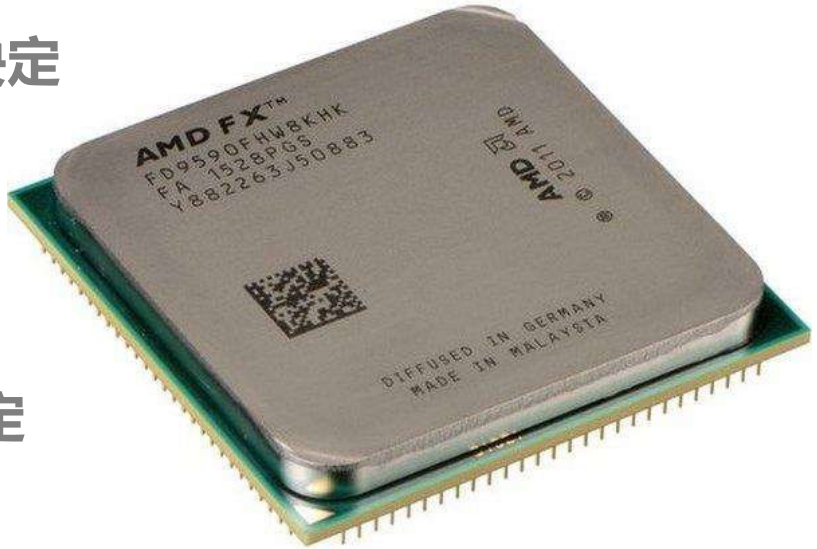
程序执行速度的影响因素： 由CPU硬件决定

- ***“主频”：周期长度**
- ***CPI：每条指令的周期数**
- ***程序总指令数**

由指令集和编译器决定

->单周期版本的MIPS处理器

->流水线版本的MIPS处理器



FX9590：8核心、4.7~5GHz、219W、需要水冷！

*指令的执行

指令的执行过程：

1. PC->指令计数器、取指令

2. 根据指令中的寄存器号、立即数，读取数据

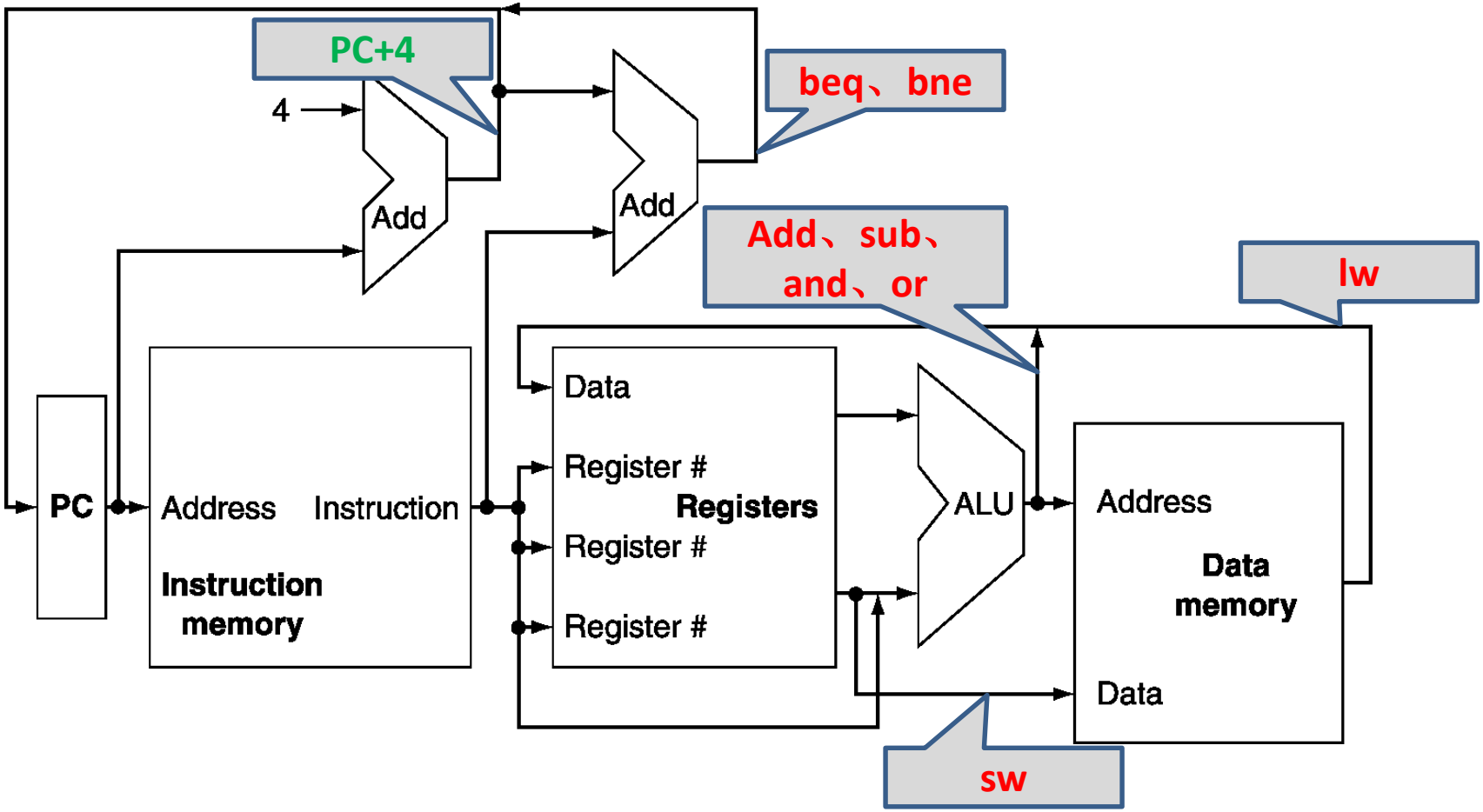
3. 根据指令类型执行不同的ALU操作

- **存储访问用ALU计算地址**
- **算术逻辑用ALU执行运算**
- **分支跳转用ALU进行比较**

4. 存储访问指令读取/写入数据；算术逻辑指令将结果写回寄存器；分支指令根据比较结果决定下一条指令的地址

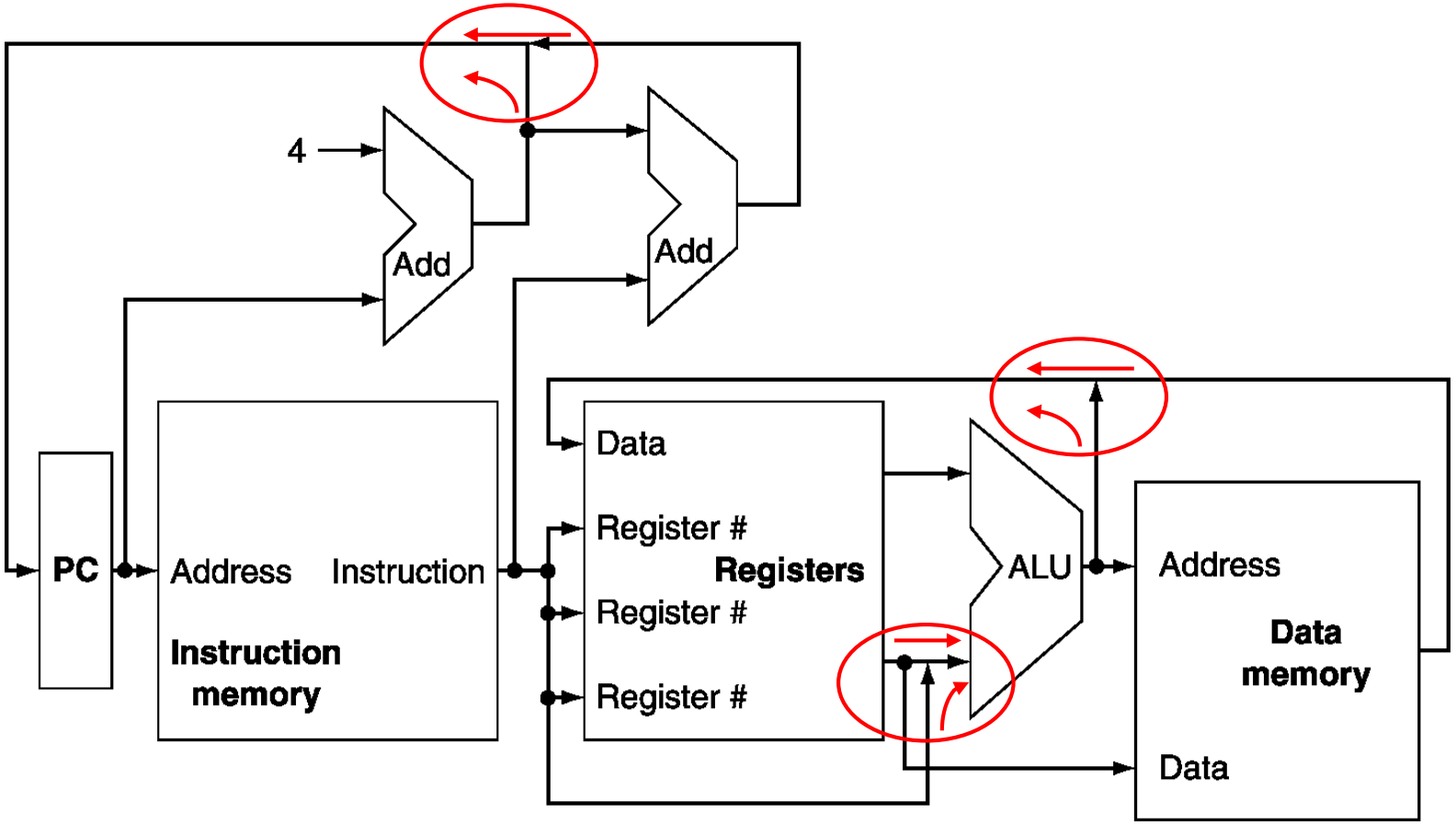
4.1.1 MIPS处理器基本结构

一个基本的处理器结构如下：



4.1.1 MIPS处理器基本结构

问题1：数据可能来自不同单元 **->需要使用数据选择器**

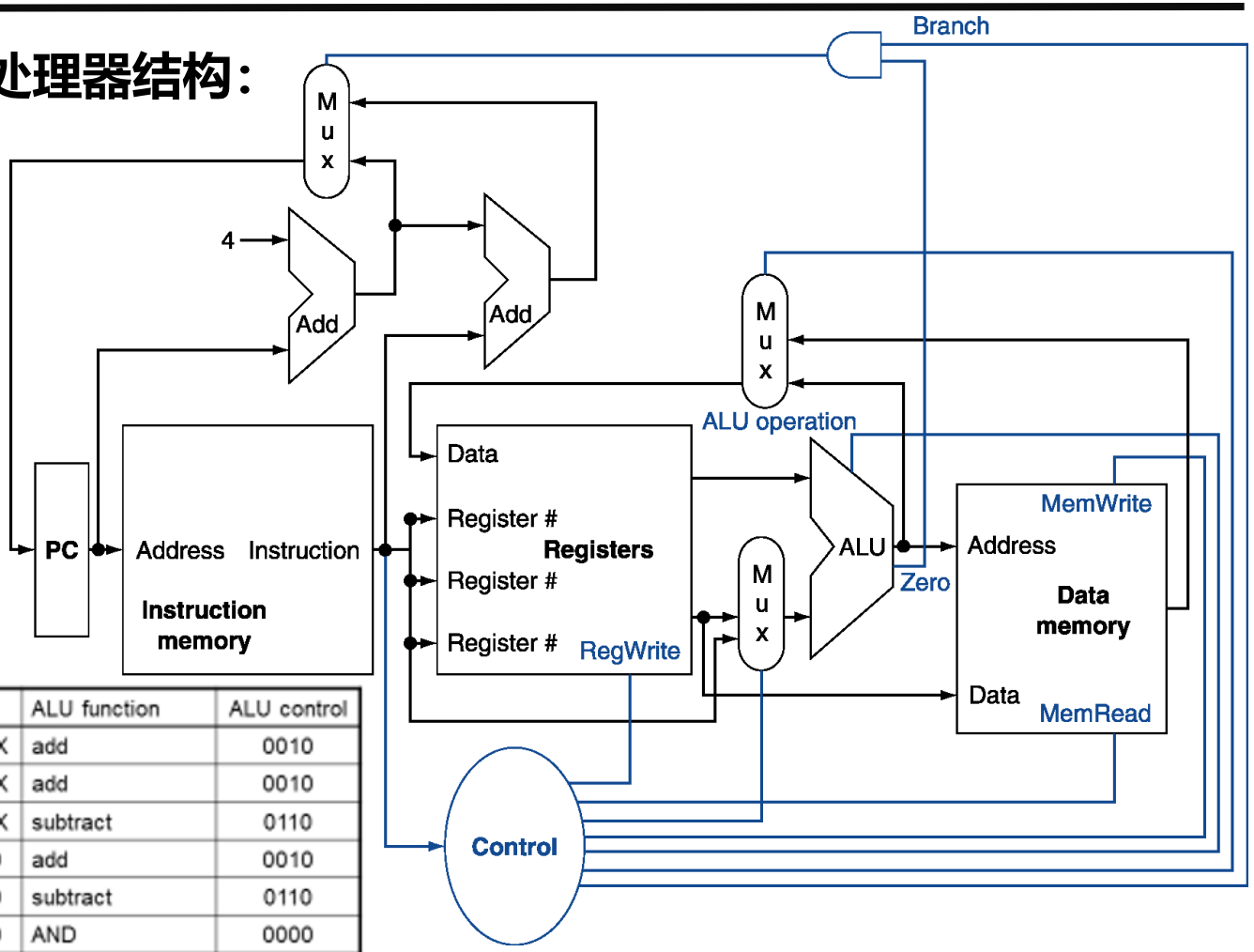


问题2：CPU如何根据不同的指令执行不同操作? **->需要控制**

4.1.1 MIPS处理器基本结构

加入选择器和控制器的处理器结构：

控制器根据指令中的操作码，设置数据选择器、以及各类控制信号



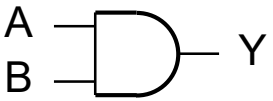
opcode	ALUOp	Operation	funct	ALU function	ALU control
lw	00	load word	XXXXXX	add	0010
sw	00	store word	XXXXXX	add	0010
beq	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		set-on-less-than	101010	set-on-less-than	0111

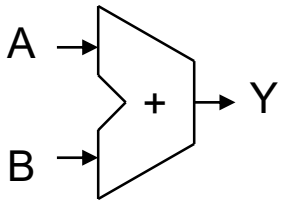
->组合逻辑、时序逻辑电路

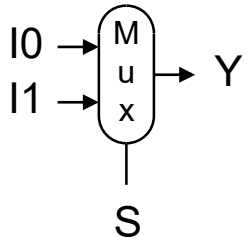
*组合与时序逻辑电路

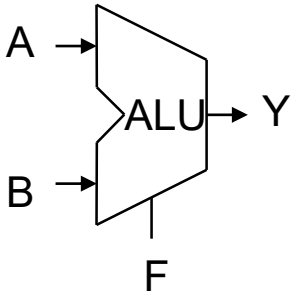
组合逻辑电路：对于一定的输入，总是产生相同输出的电路(输出与先前状态无关)

- 逻辑门
- 加法器
- 选择器
- 算术逻辑单元









如果没有时序逻辑电路...?

程序不能存储 =没有冯.诺依曼

=没有C和Java

=程序猿们没事做了，就变成酱紫->>>

状态单元：存储状态信息（寄存器、存储器）

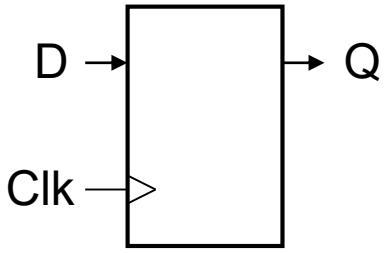
时序逻辑电路：包含状态单元的电路

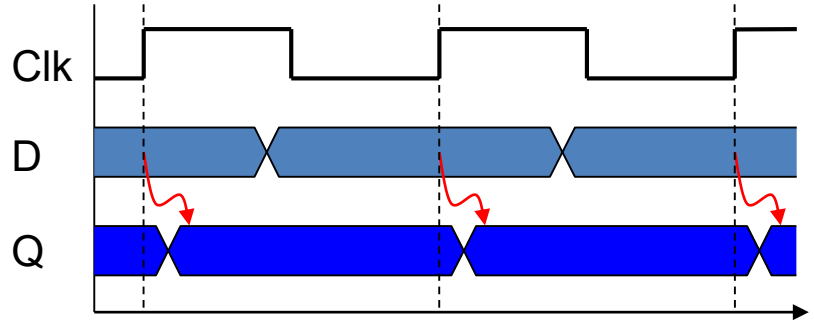
*时序逻辑电路

寄存器结构：

可由D触发器构成

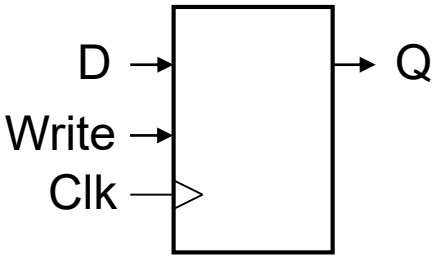
当遇到Clk上升沿时，保存D信号，否则保持不变

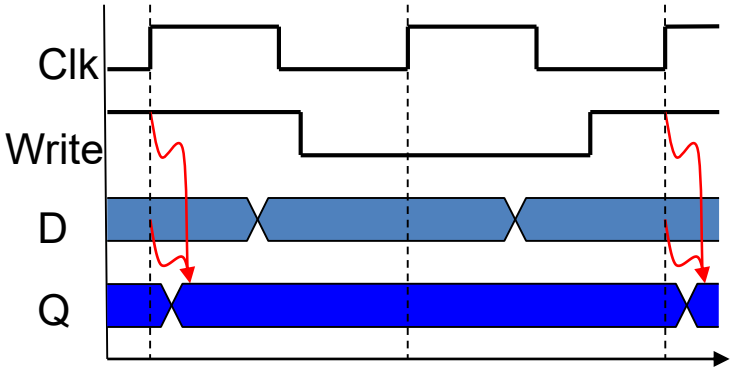




问题：数据会一直更新

带写入控制的寄存器：当写使能有效时，才会更新数据

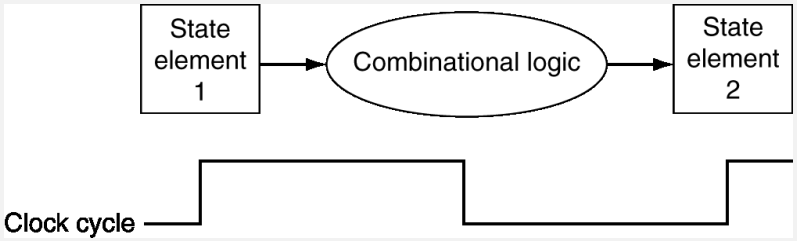


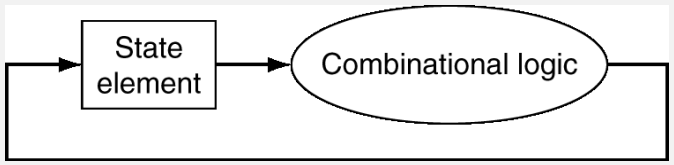


目录

4.1.2 时钟方法

两个时钟周期上升沿之间，由组合逻辑电路执行运算：





组合逻辑的输入来源于状态单元；输出写入到状态单元

ASIC设计广泛采用边沿触发：在时钟上升/下降沿更新数据，由控制信号决定是否写入，可以避免单个周期内的反馈

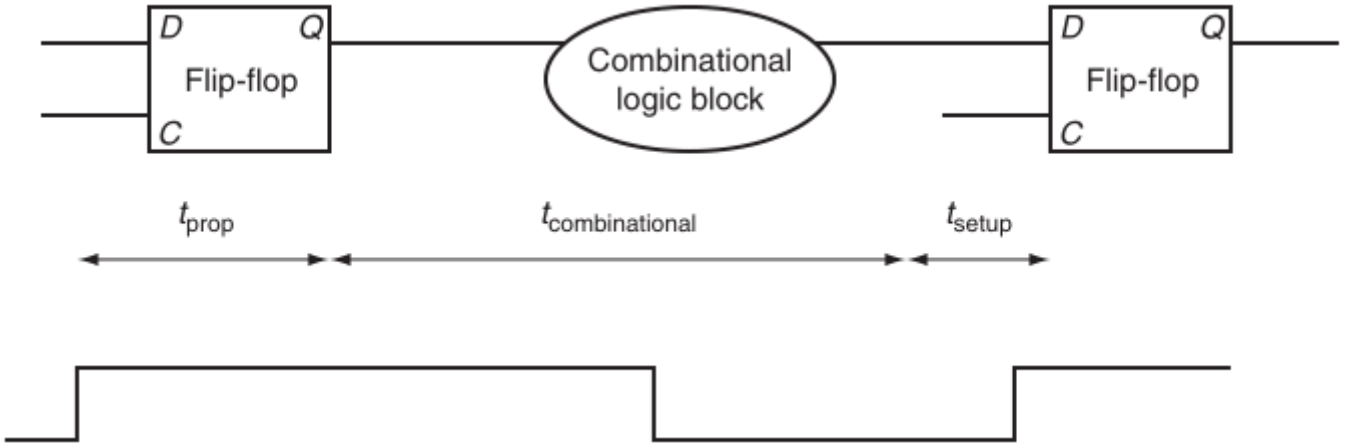
问题：对于状态单元，如果一个信号被同时读出和写入，则所读取信号为？

->需要时钟方法：相对时钟来讲，什么时候的数据是有效的？

->时钟周期应该设计为多少？组合逻辑电路的延迟应该满足什么条件？

4.1.2 时钟方法

Edge-triggered timing



$T > t_{prop} + t_{combinational,max} + t_{setup}$

$t_{combinational,min} > t_{hold} - t_{prop}$

$T > t_{prop} + t_{combinational,max} + t_{setup} + t_{skew}$

$t_{combinational,min} > t_{hold} + t_{skew} - t_{prop}$

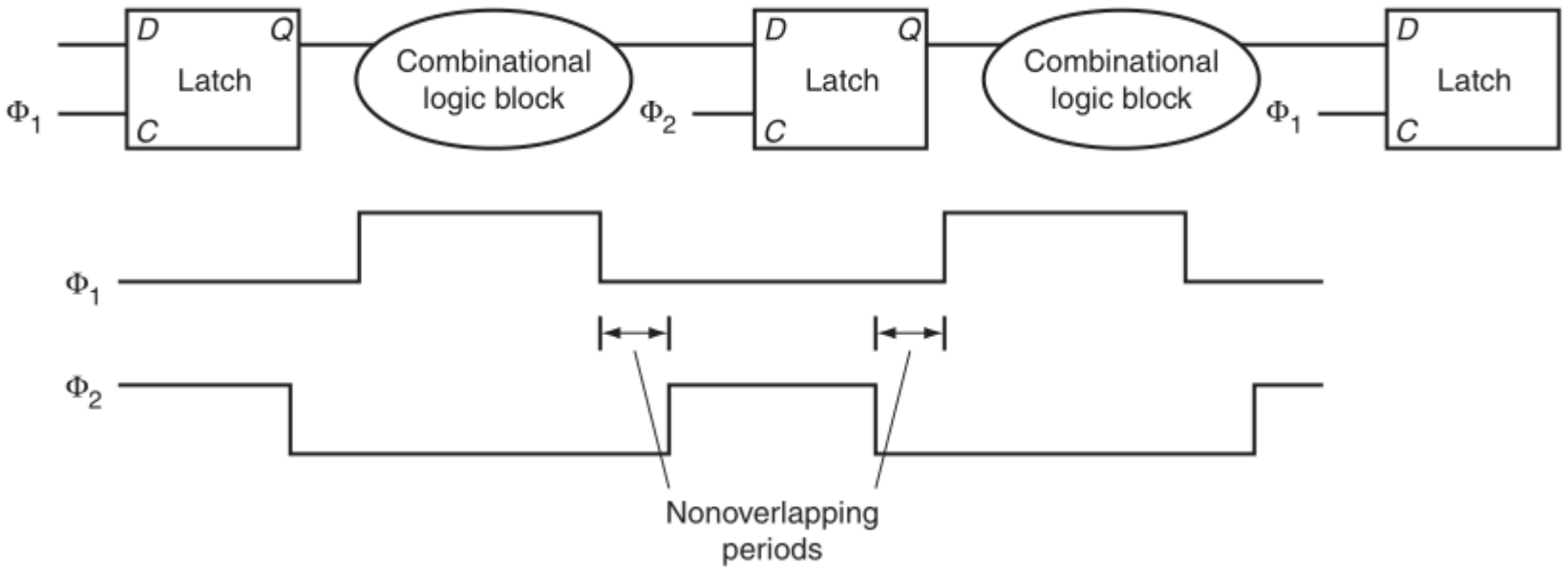
对组合逻辑电路具有双边约束:

$t_{combinational,max} < T - t_{prop} - t_{setup} - t_{skew}$

$t_{combinational,min} > t_{hold} + t_{skew} - t_{prop}$

4.1.2 时钟方法

Level-sensitive timing：同样可以避免单个周期内的反馈



一次只更新一个相位的数据，两相不交叠区域不更新数据

对组合逻辑电路只存在单边约束： $t_{combinational,max} < T_{\varphi} - t_{prop} - t_{setup} - t_{skew}$

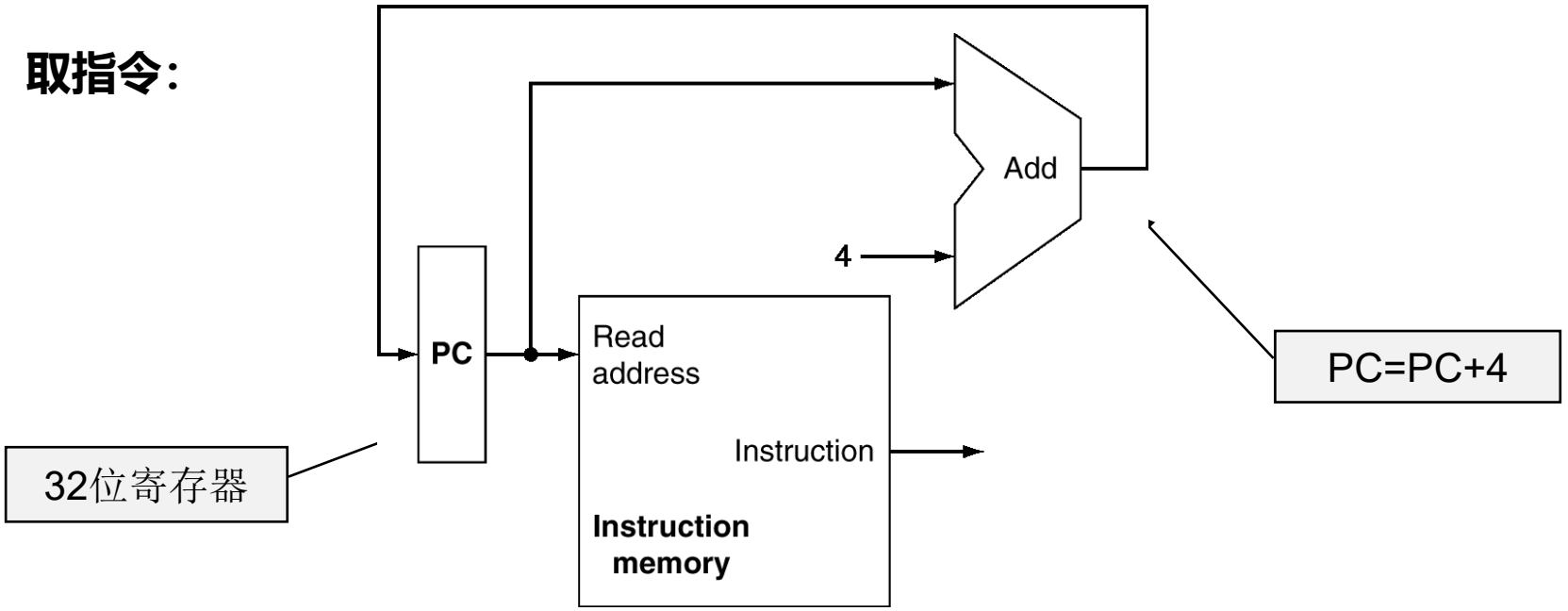
->当采用较快工艺时，单边约束自然满足，不会引入时序问题，便于功能设计

目录

4.1.3 建立基本数据通路

数据通路：处理器中，处理数据和地址的元素，比如：寄存器、ALU、选择器、内存等

取指令：



建立数据通路的方法：

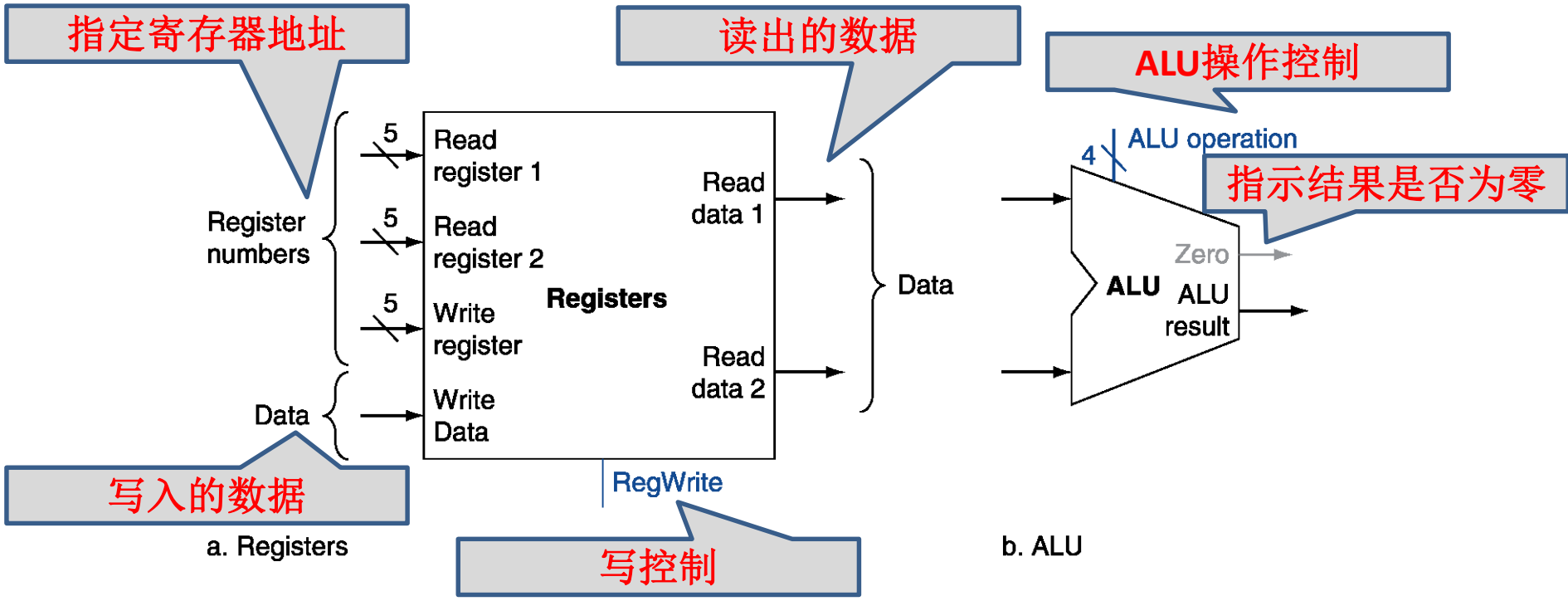
1. **实现单一指令的数据通路**
2. **使用数据选择器选择不同指令的数据 （搭积木）**

4.1.3 建立基本数据通路--R型指令

读取两个寄存器操作数

执行算术逻辑运算

将结果写回寄存器



4.1.3 建立基本数据通路--存取数指令

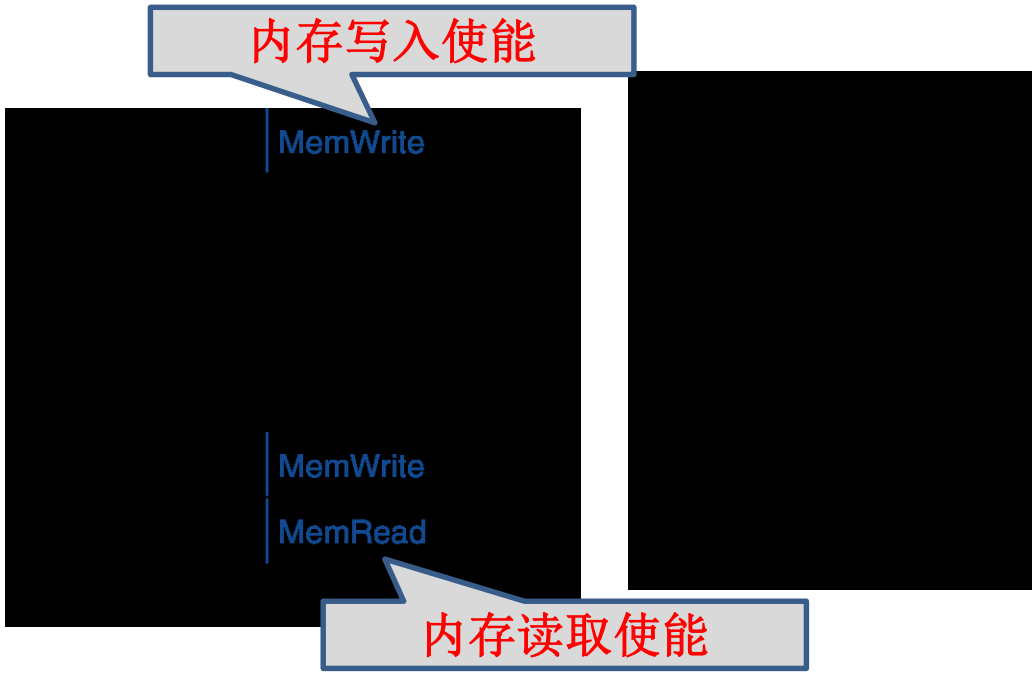
读取/写入寄存器

Load：读取内存数据，写入寄存器

Store：将寄存器数写入内存

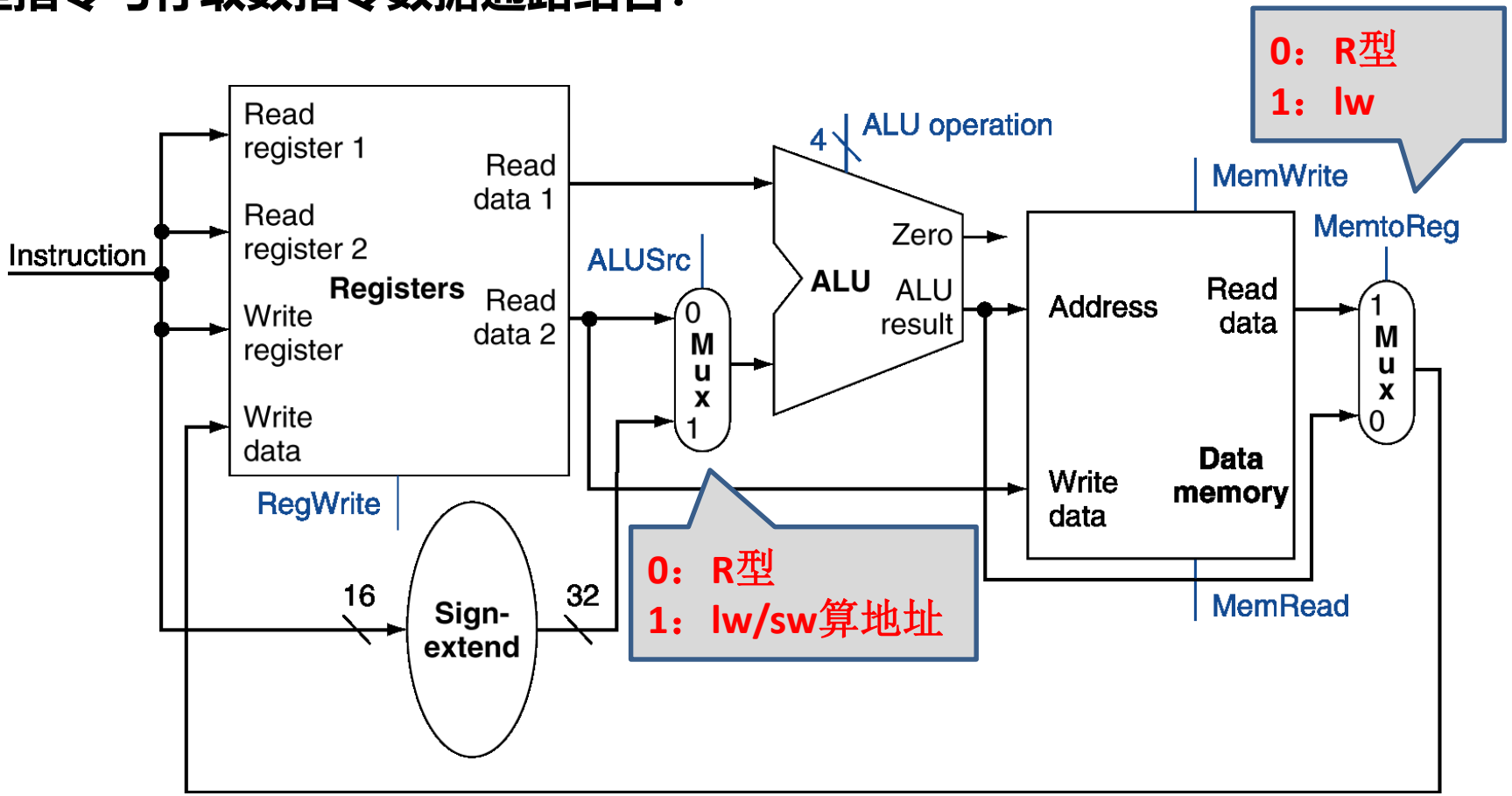
使用ALU计算地址：16位数符号位扩展

lw/sw $t0, AddrConstant4($s1)



4.1.3 建立基本数据通路--存取数数据通路

将R型指令与存取数指令数据通路结合：



4.1.3 建立基本数据通路--分支指令

PC+4：取指阶段完成

读取寄存器中的操作数

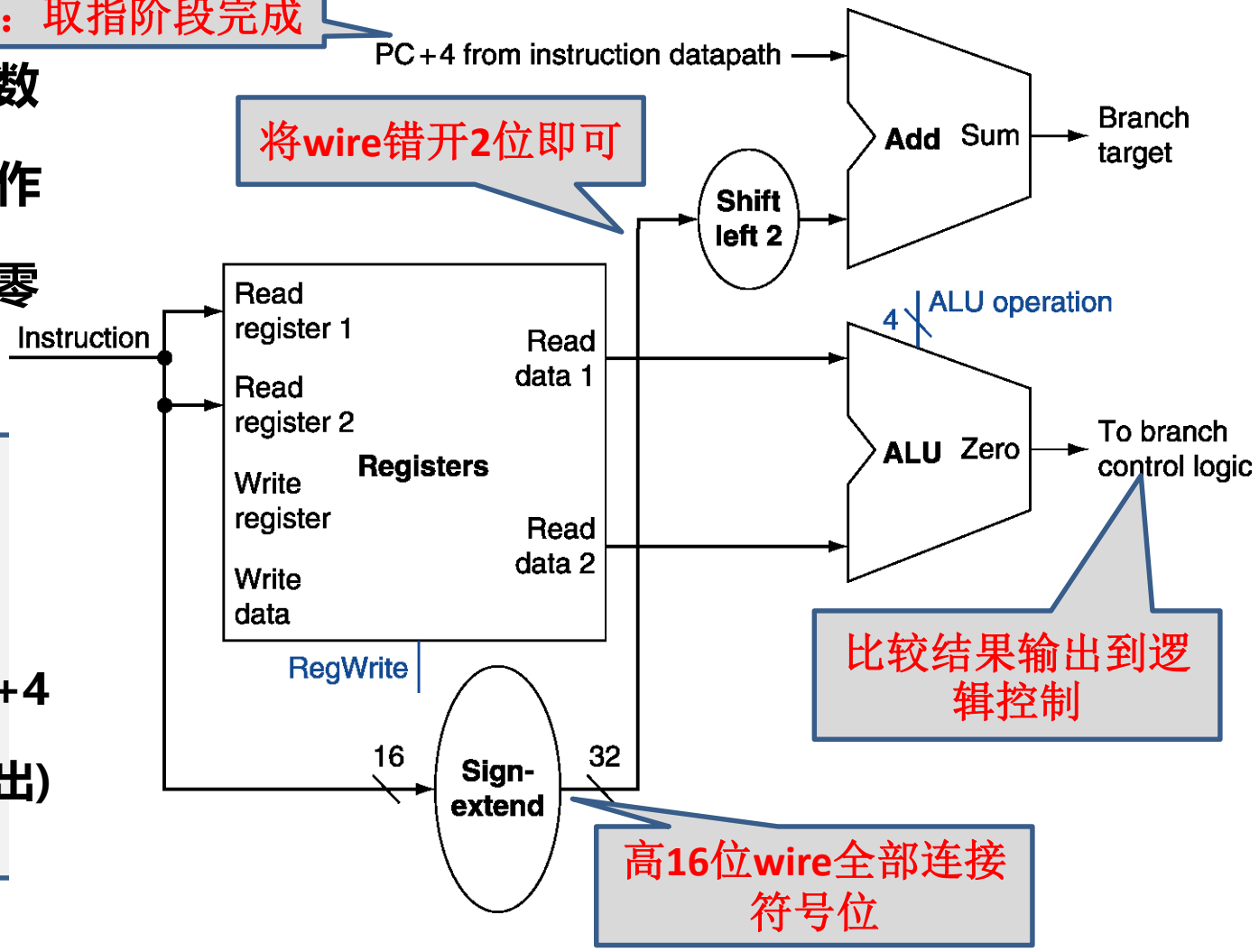
比较操作数：用ALU作减法，看结果是否为零

目标地址：

符号位扩展的偏量

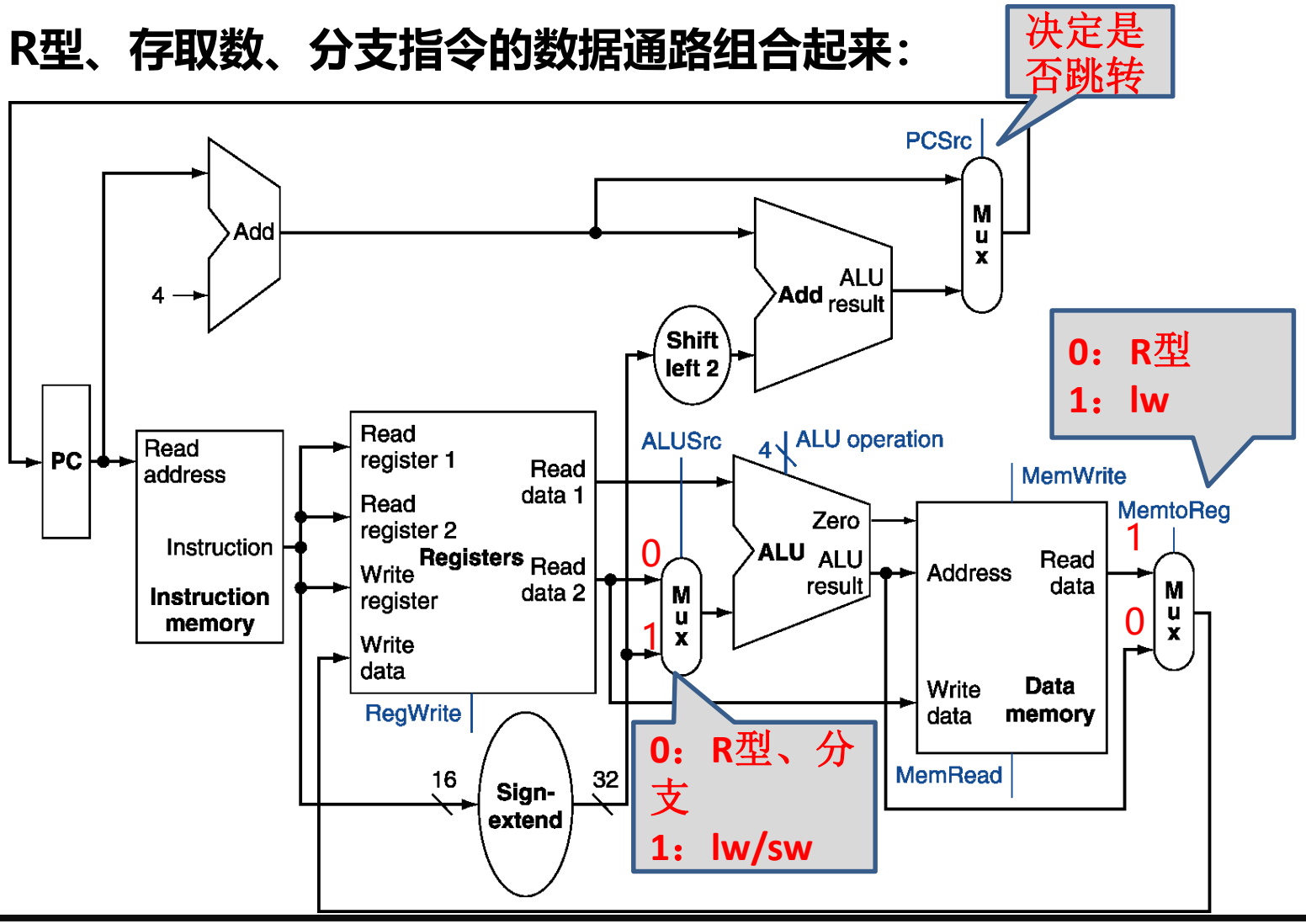
左移两位，再加PC+4

(已经在取指阶段算出)



4.1.3 建立基本数据通路--完成基本数据通路

将取指、R型、存取数、分支指令的数据通路组合起来：



目录

4.1.4 MIPS处理器的控制--ALU控制

所有指令都需要使用ALU：

存取指令：F=add

分支指令：F=subtract

R型指令：F取决于函数码

问题：ALU如何根据不同指令执行不同操作?　　->需要ALU控制

ALU 控制	操作
0000	AND
0001	OR
0010	add
0110	subtract
0111	set-on-less-than
1100	NOR

4.1.4 MIPS处理器的控制--ALU控制

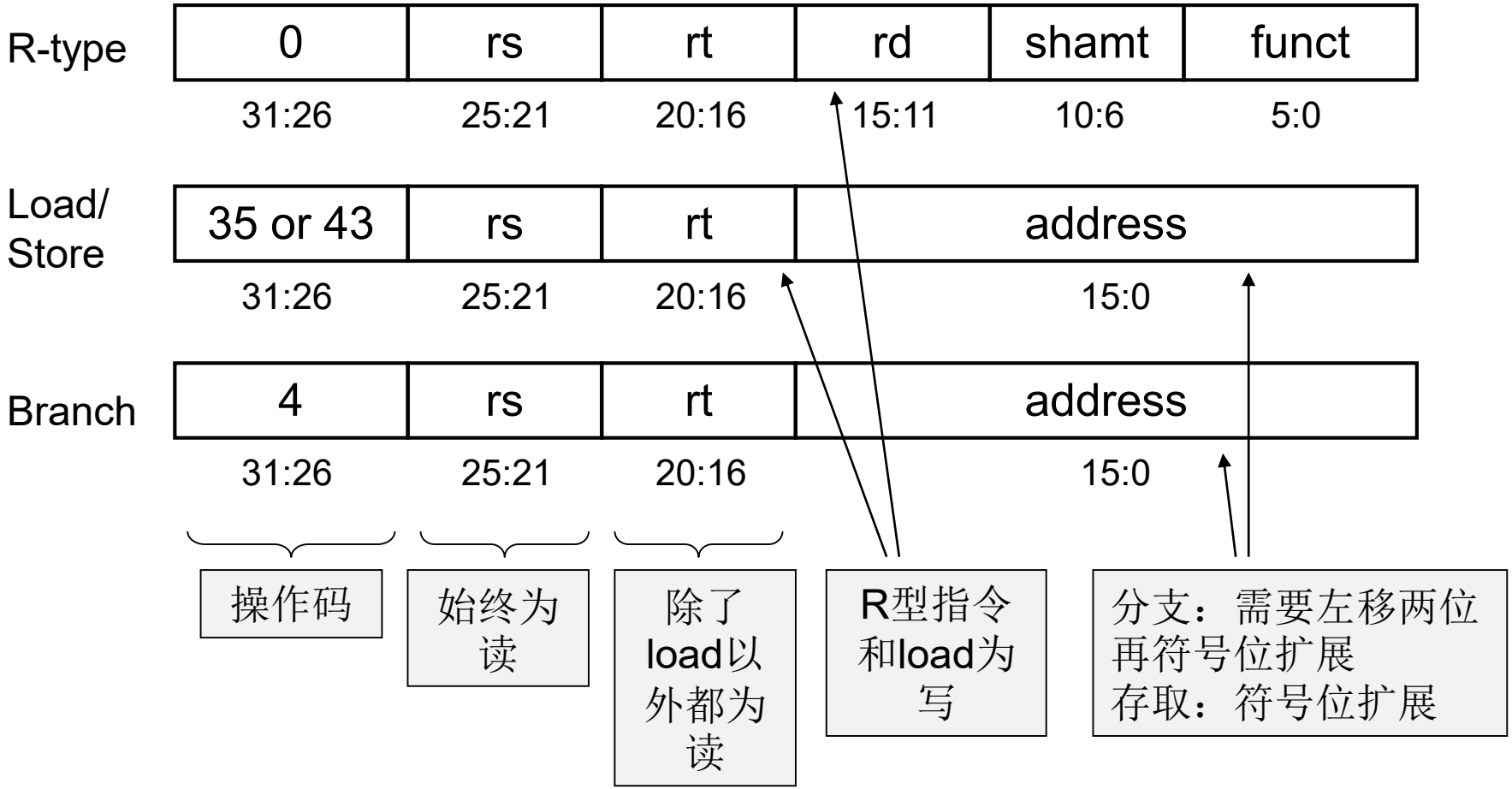
思考：主控单元的延迟会产生什么影响？　　->对时钟周期非常重要！

***多级译码：ALUop作为ALU控制单元的输入，进而生成ALU control；可减小主控单元的规模**

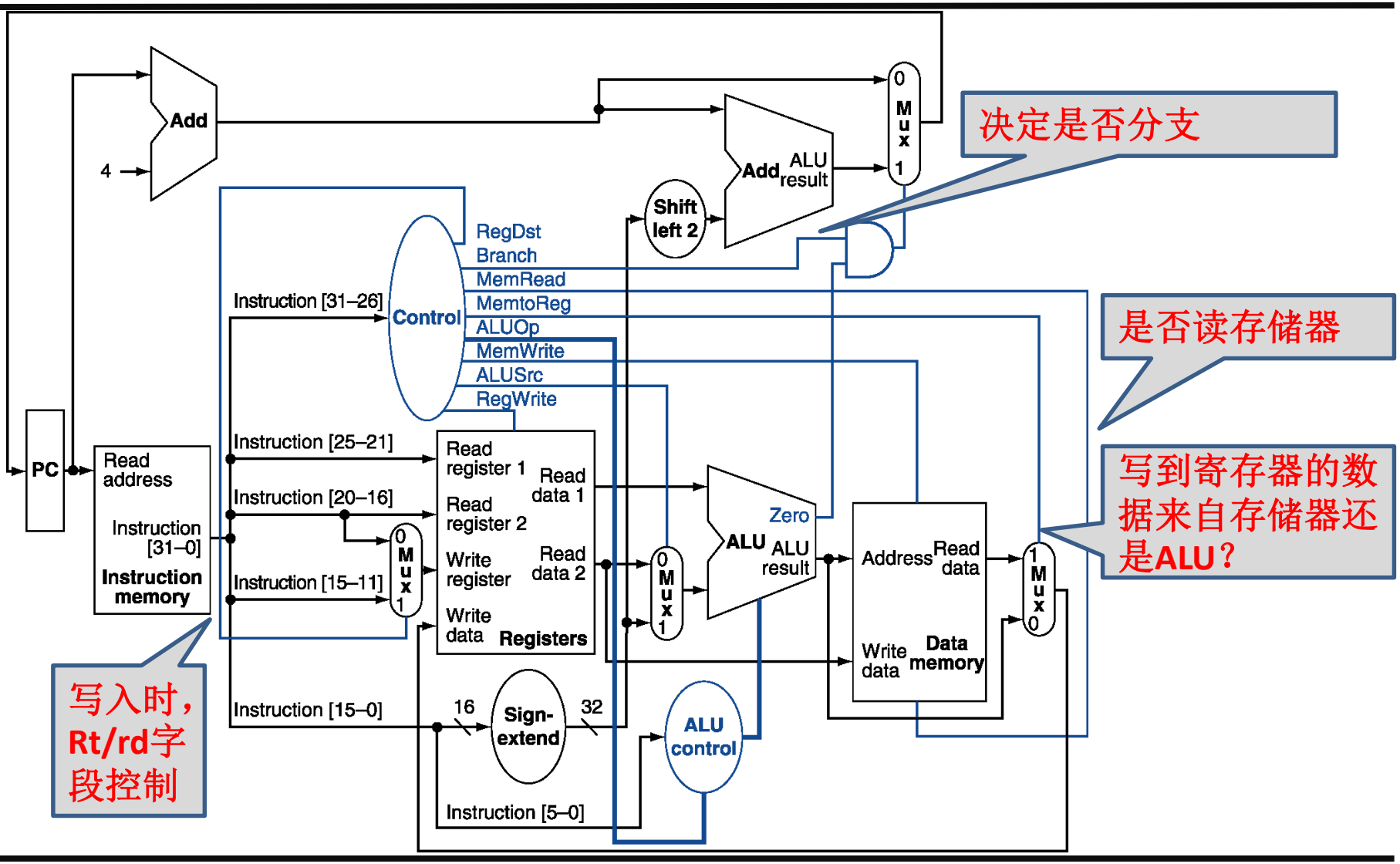
opcode	ALUOp	Operation	funct	ALU function	ALU control
lw	00	load word	XXXXXX	add	0010
sw	00	store word	XXXXXX	add	0010
beq	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		set-on-less-than	101010	set-on-less-than	0111

4.1.4 MIPS处理器的控制--主控单元

由指令得到的控制信号

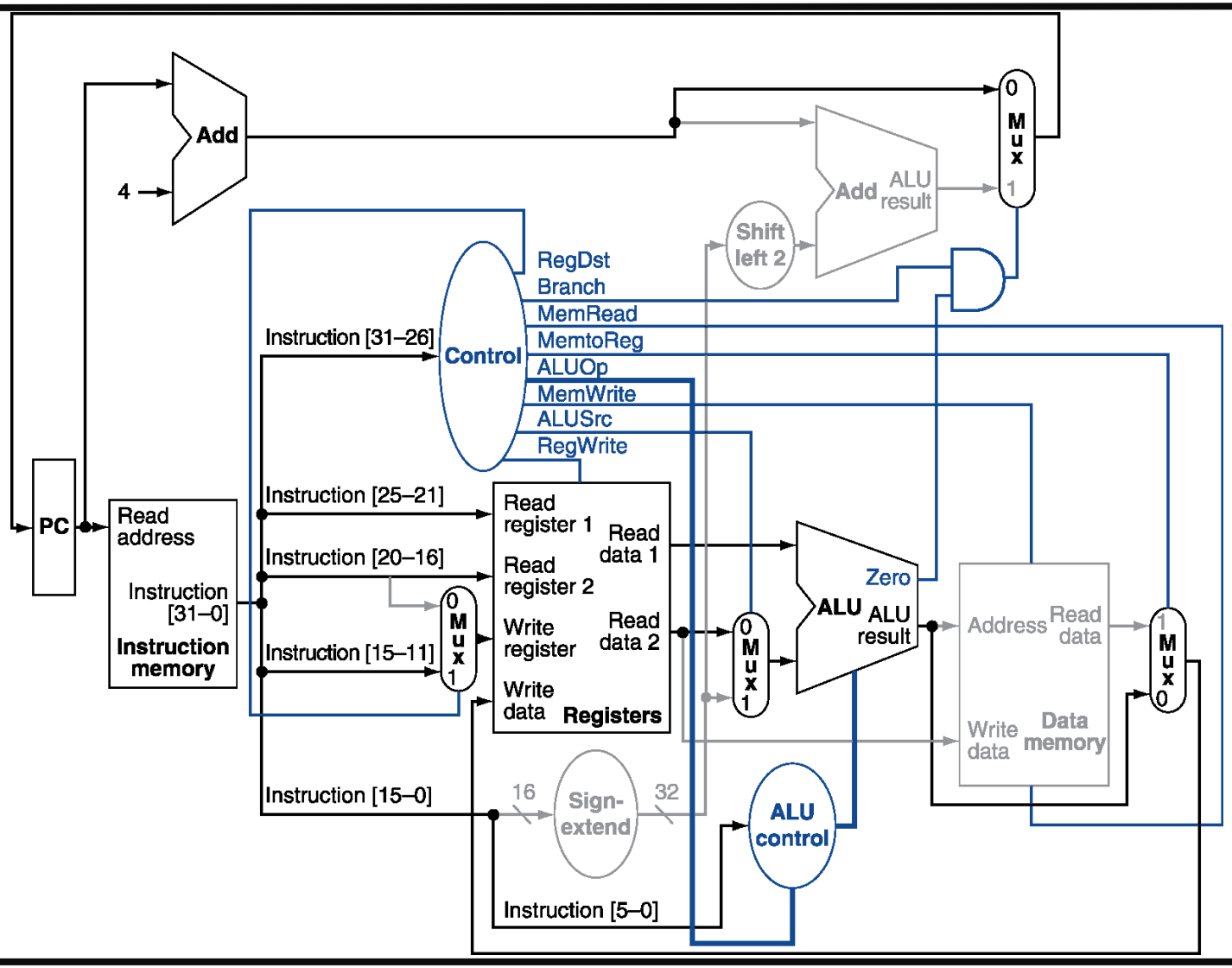


4.1.4 MIPS处理器的控制--带控制(主控、ALU控制)的数据通路



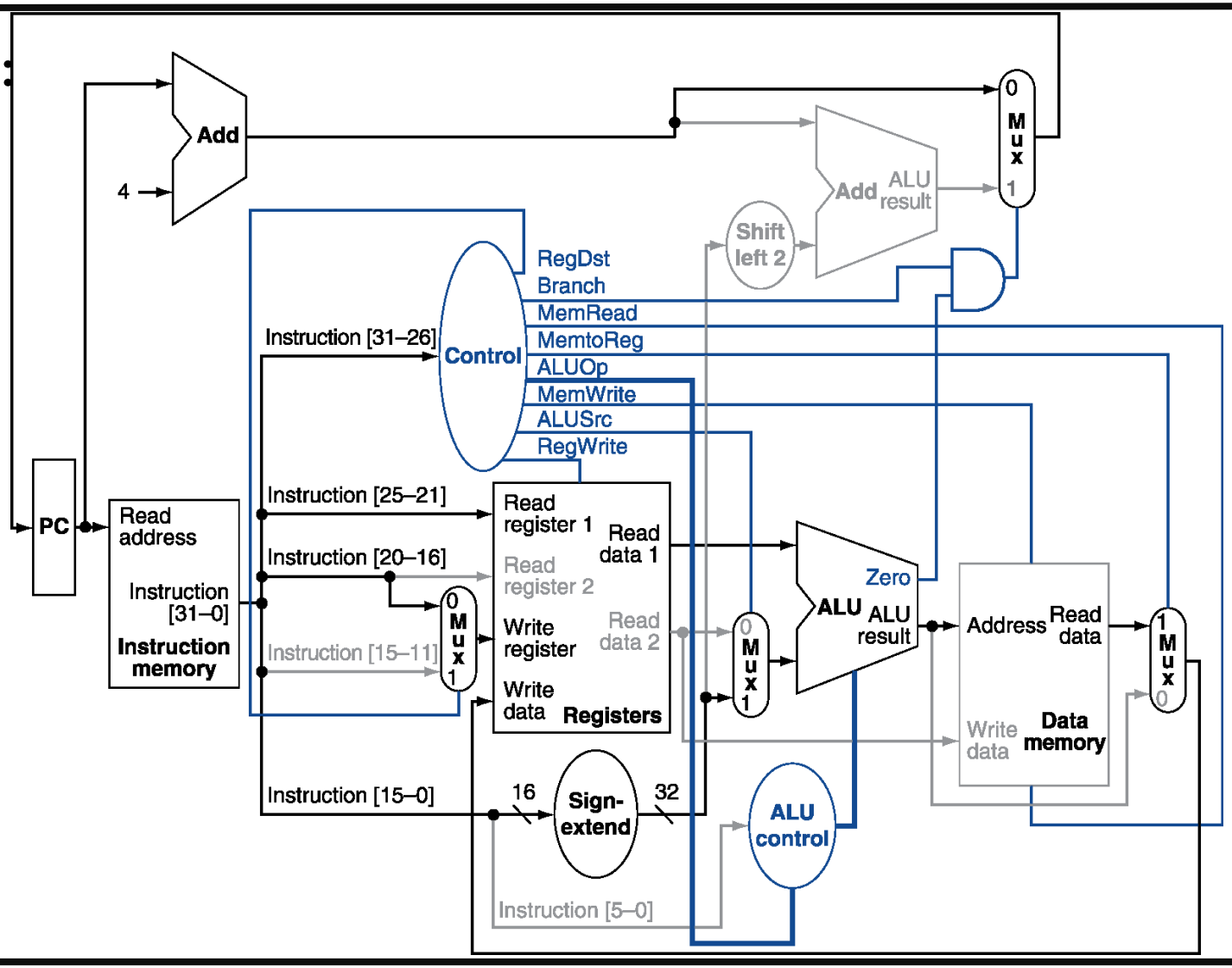
4.1.4 MIPS处理器的控制--带控制的数据通路

R型指令：



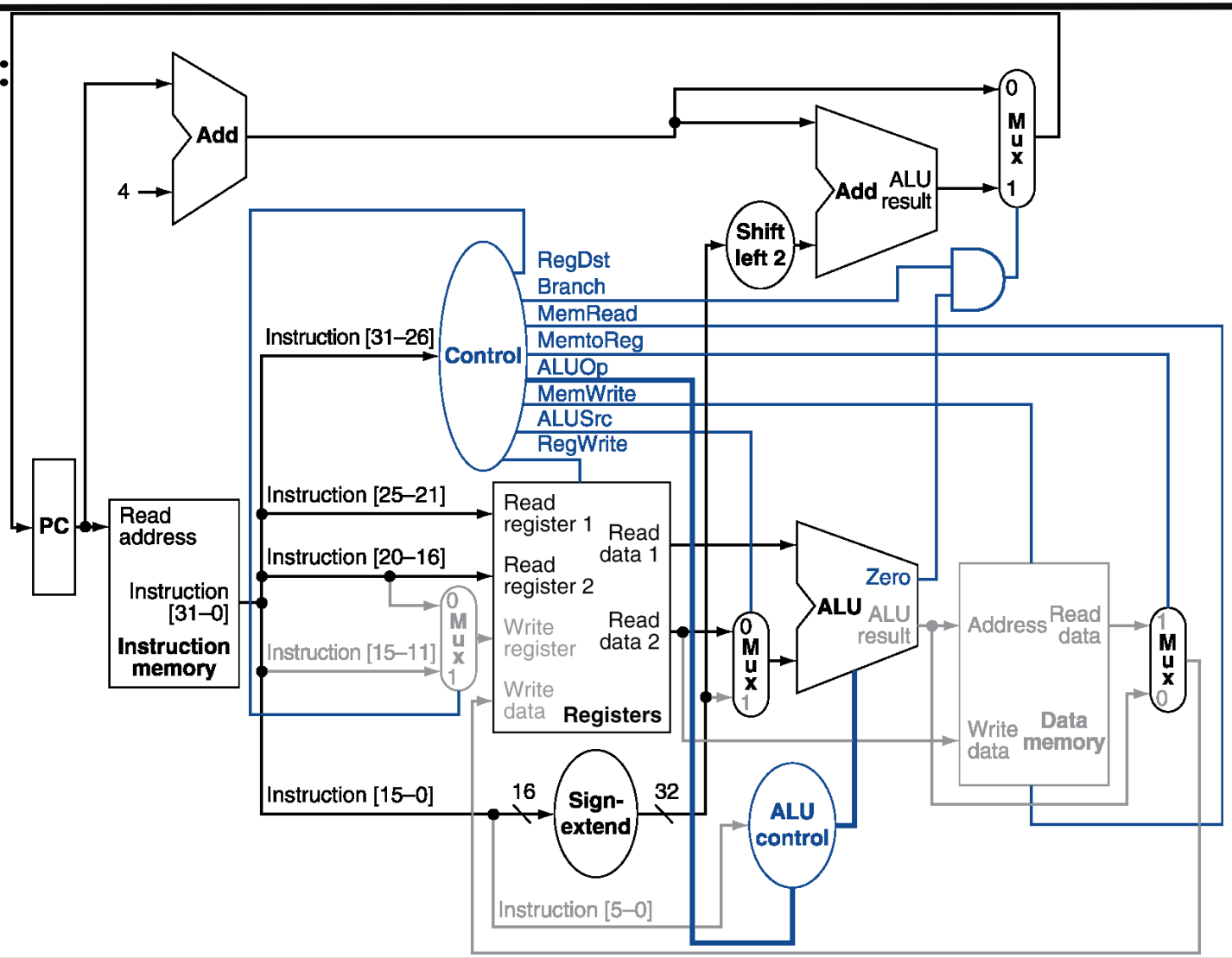
4.1.4 MIPS处理器的控制--带控制的数据通路

取数指令：

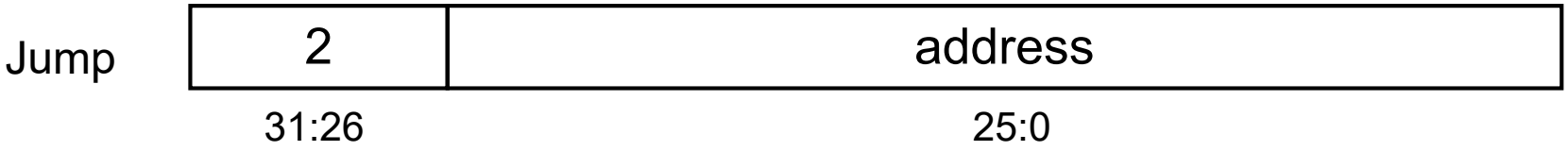


4.1.4 MIPS处理器的控制--带控制的数据通路

beq指令：



*跳转指令的实现

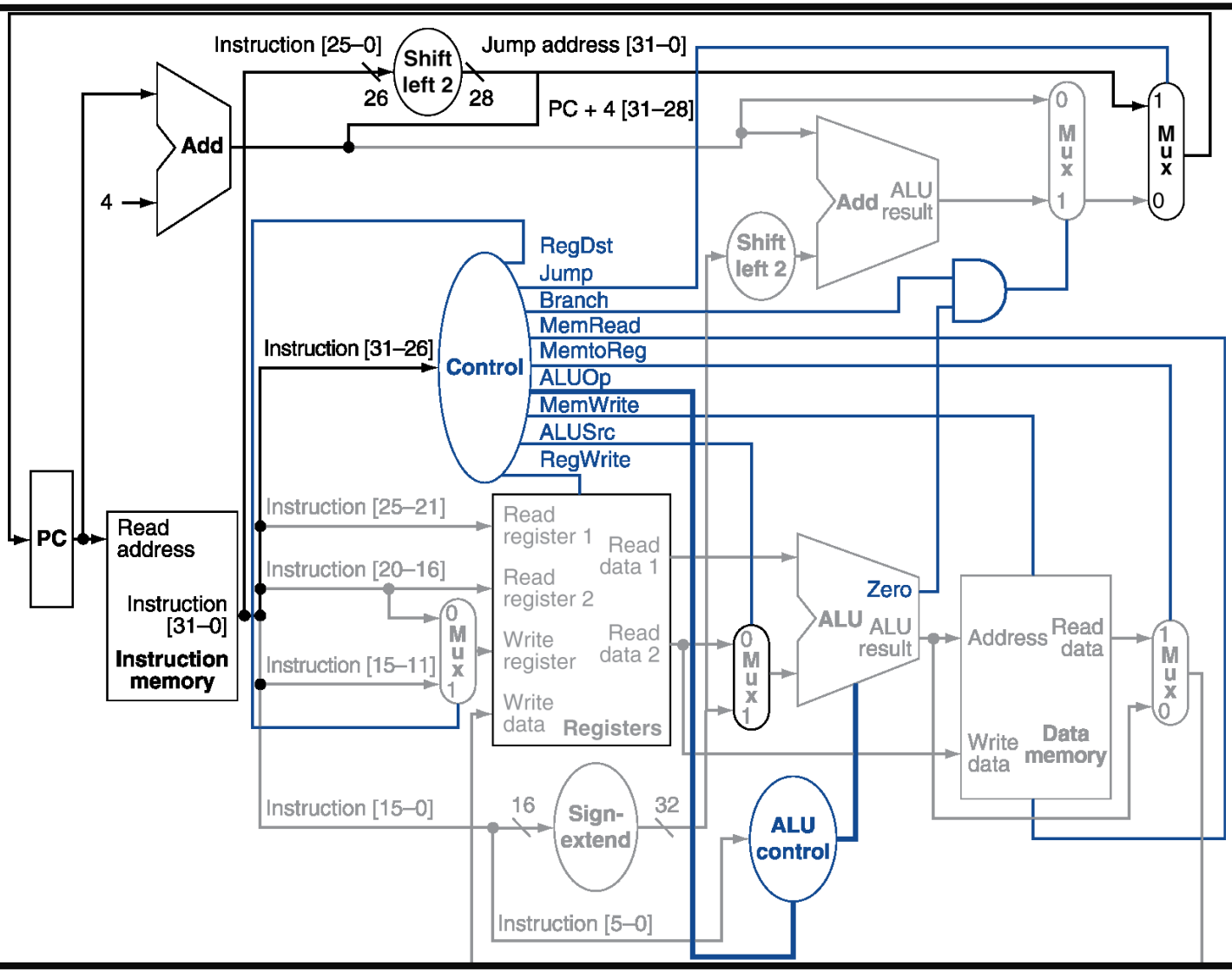


Jump使用字地址

低26位立即数地址替换PC的中间26位

需要从操作码解码出额外的控制信号

*加入跳转的数据通路


Instruction [25–0]
Shift left 2
Jump address [31–0]
26
28
PC + 4 [31–28]
Add
4
0
M
u
x
1
1
M
u
x
0
Add
ALU result
Shift left 2
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
Zero
ALU
ALU result
Address
Read data
Write data
Data memory
Instruction [15–0]
16
Sign-extend
32
ALU control
Instruction [5–0]

*性能的考虑

最长延迟决定了时钟周期

关键路径：取数指令（指令存储器->寄存器->ALU->数据存储器->寄存器）

问题：对不同的指令给予不同的周期是不现实的！

->需要流水线化