Single-Cycle MIPS32 Processor Design



Single-Cycle MIPS Processor Design

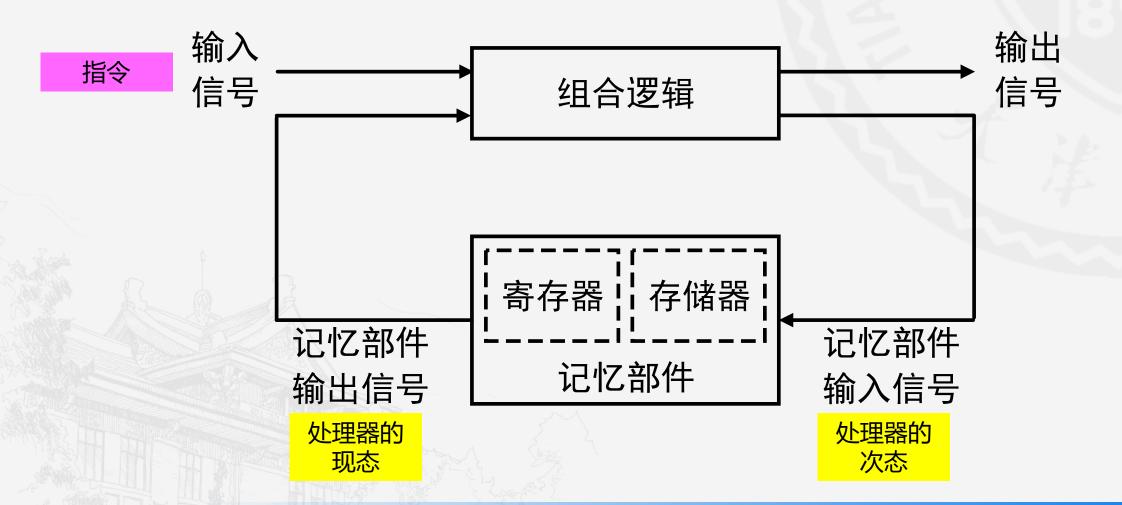
设计要求

- 单周期MIPS32处理器
 - 每条指令均在一个时钟周期完成
 - 32个32位寄存器,哈佛结构,小端模式,支持23条指令
- 数据通路 + 控制通路
 - 数据通路: 完成对指令中操作数的运算、存储等处理工作
 - 控制通路: 从数据通路中接收指令, 并对其进行翻译以告知数据通路如何处理
- 处理器设计相当于在各个记忆部件之间添加组合逻辑电路,在控制单元的控制下根据 当前电路的状态计算出电路的新状态



Single-Cycle MIPS Processor Design

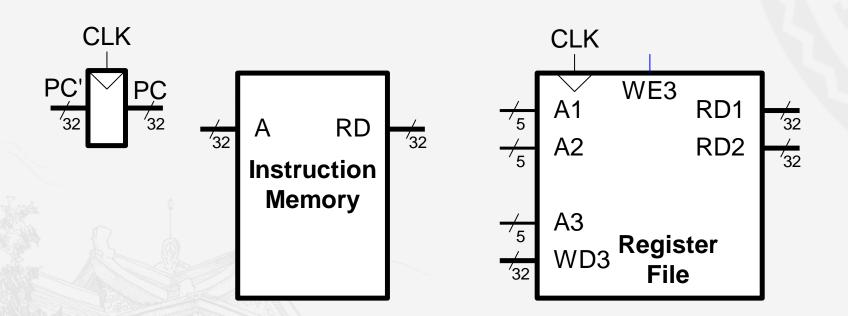
MiniMIPS32处理器概念模型

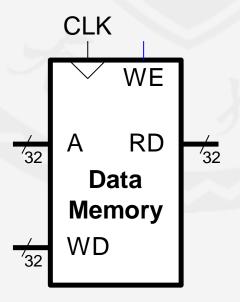




Single-Cycle MIPS Processor Design

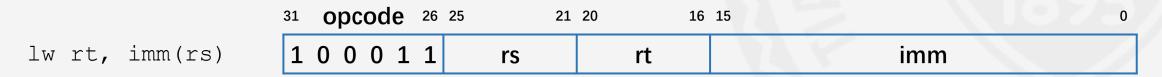
记忆部件





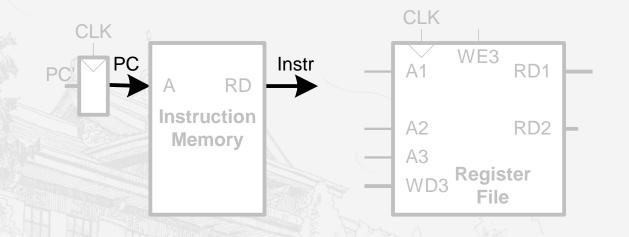
Single-Cycle MIPS Processor Design

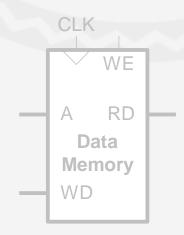
lw指令数据通路(I类型指令)



imm需要符号扩展为32位有符号数

Step 1 取指令



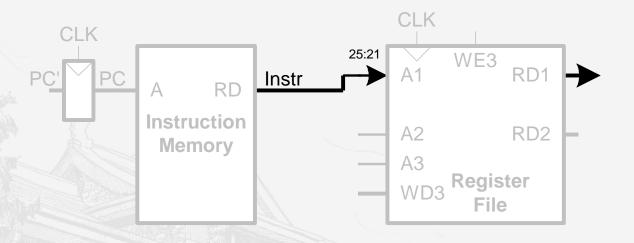


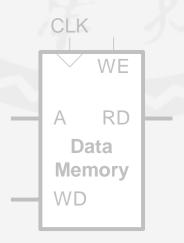
Single-Cycle MIPS Processor Design

 31 opcode 26 25
 21 20
 16 15
 0

 1w rt, imm(rs)
 1 0 0 0 1 1
 rs
 rt
 imm

Step 2-1 译码阶段 — — 从寄存器取操作数



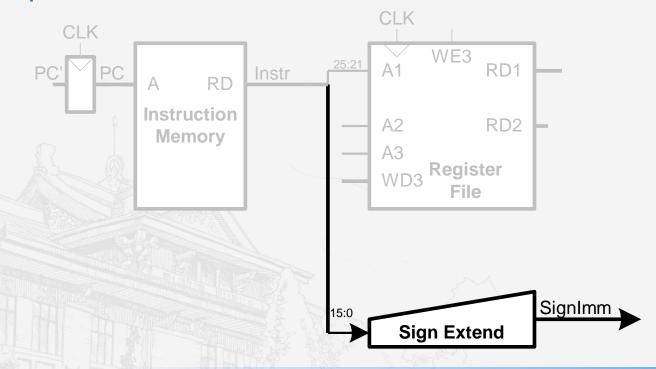


Single-Cycle MIPS Processor Design

 31 opcode 26 25
 21 20
 16 15

 1w rt, imm(rs)
 1 0 0 0 1 1
 rs
 rt
 imm

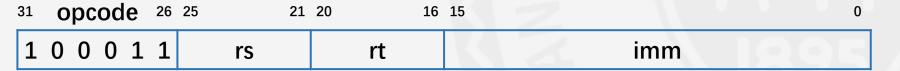
Step 2-2 译码阶段 — —另一个操作数进行符号扩展

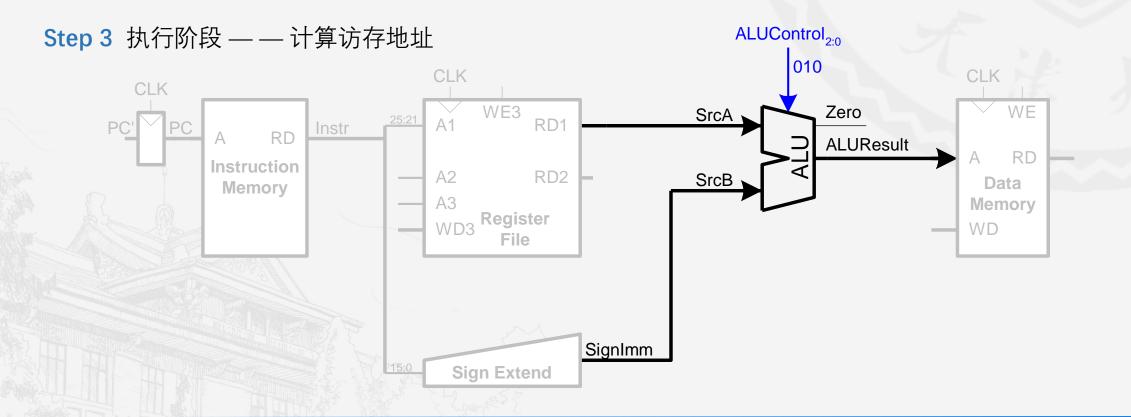




Single-Cycle MIPS Processor Design

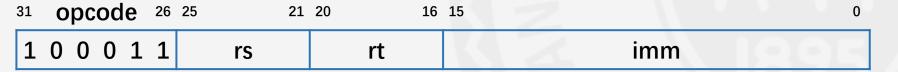
lw rt, imm(rs)





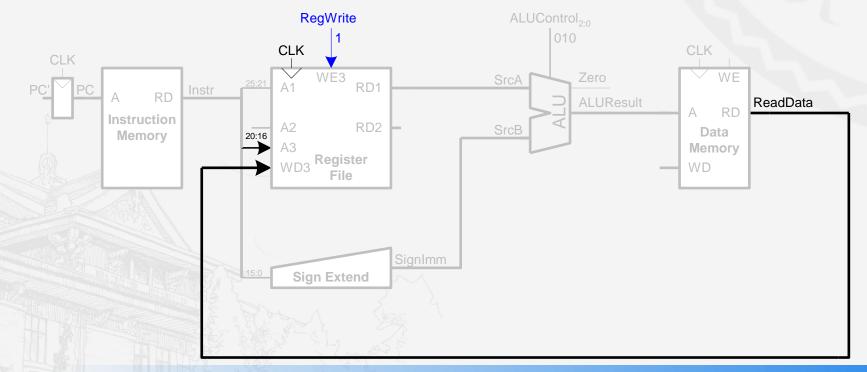
Single-Cycle MIPS Processor Design

lw rt, imm(rs)



imm需要符号扩展为32位有符号数

Step 4 访存阶段/写回阶段 — —从数据存储器取回数据,写入寄存器文件



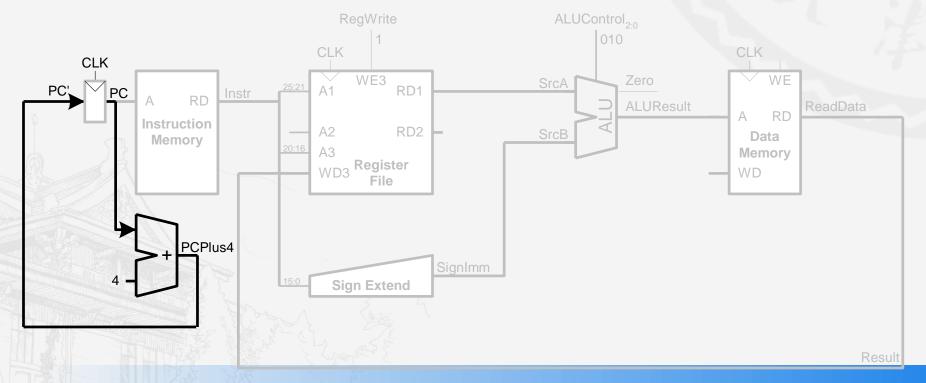
Single-Cycle MIPS Processor Design

lw rt, imm(rs)

31	opcode ²	6 25	21	20 16	15 0
1	0 0 0 1 3	L	rs	rt	imm

imm需要符号扩展为32位有符号数

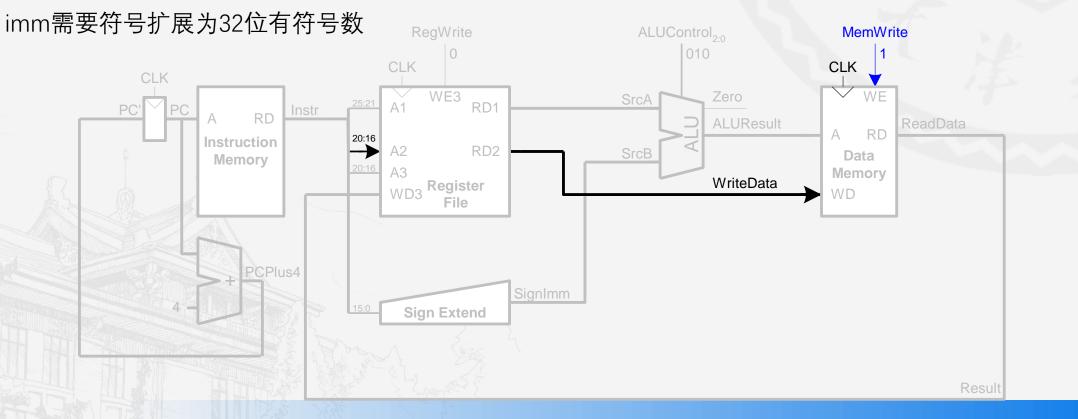
Step 5 PC — — 计算下一条指令的地址



Single-Cycle MIPS Processor Design

sw指令数据通路 (I类型指令)







Single-Cycle MIPS Processor Design

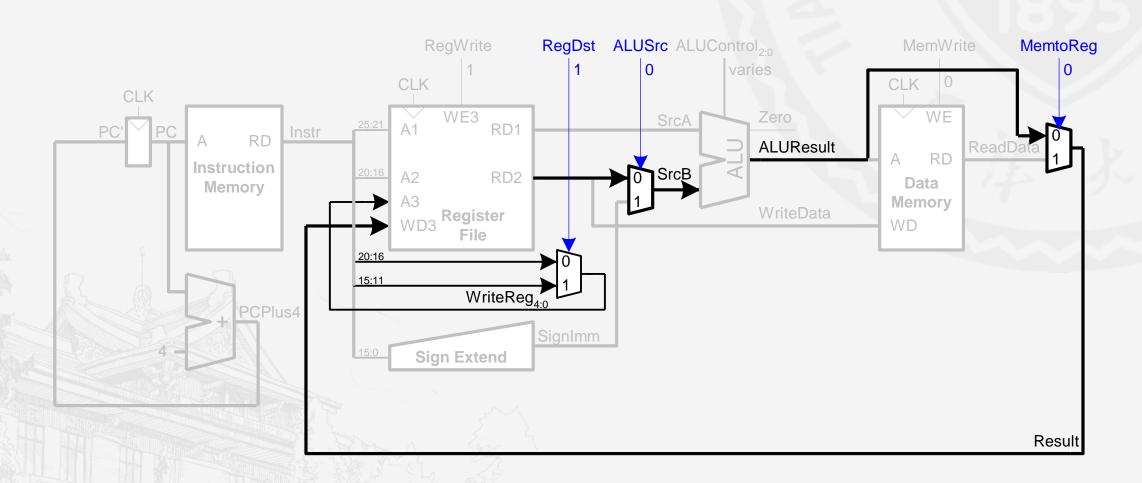
R类型指令回顾

				31	op	coc	de	26	25		21	20		16	15		11	10			6	5		fund	t	0
add	rd,	rs,	rt	0	0 0	0	0	0		rs			rt			rd		0	0	0 () C	1	. 0	0 0	0	0
																			5	Ą.						
				31	ор	coc	de	26	25		21	20		16	15		11	10			6	5	È	fund	;t	0
sub	rd,	rs,	rt	0	0 0	0	0	0		rs			rt			rd		0	0	0 () C	1	. 0	0 0	1	0
				21	op	α	46	26	25		21	20		16	15		11	10			6	5		fund	t:	0
				31	υþ	COC	1 C														·	ŭ				
sll	rd,	rt,	shamt					_	0 0				rt			rd			sha	am		_		0 0		
sll	rd,	rt,	shamt					_					rt							am		_				
sll	rd,	rt,	shamt	0		0	0	0	0 0				rt		15				sha		t	0	0		0	0
		rt,		31	0 0	oc	0 de	26	0 0		0 0		rt rt				11	10	sha		t	5	0	0 0	0 ct	0



Single-Cycle MIPS Processor Design

R类型指令数据通路



Single-Cycle MIPS Processor Design

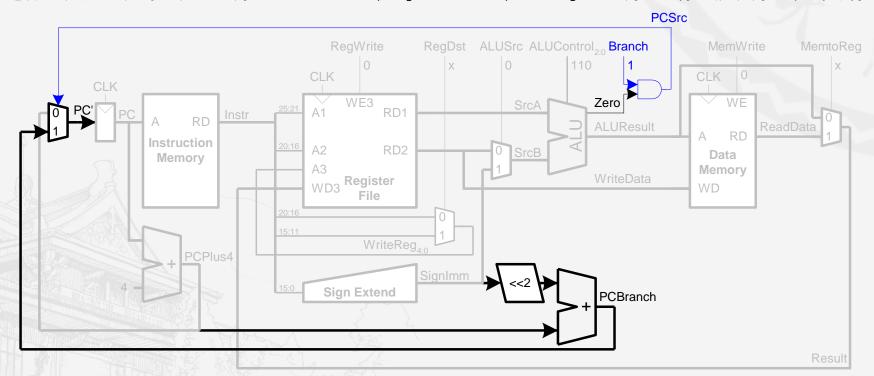
beq指令数据通路(I类型指令)

31 opcode 26 25 21 20 16 15 0

beq rs, rt, label

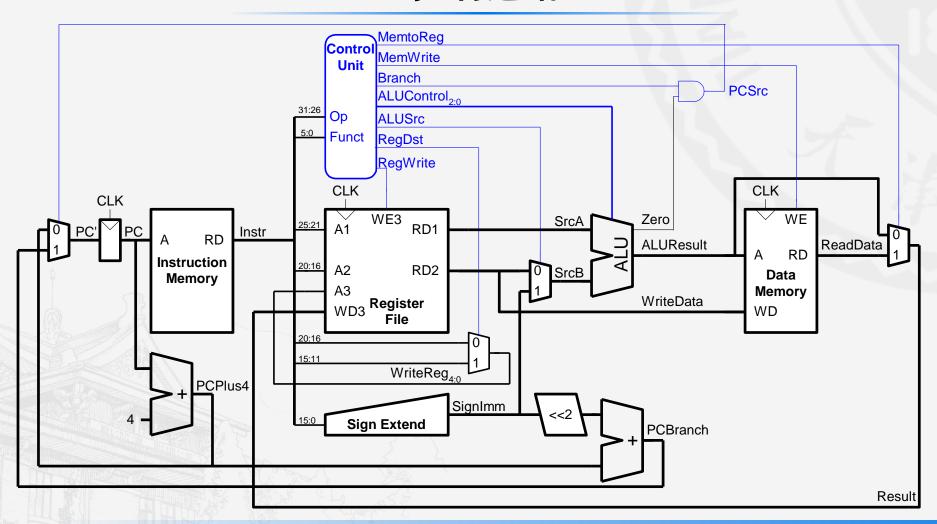
0 0 0 1 0 0 rs rt imm

当rs==rt时,进行跳转,跳转的目标地址为 PC + 4 + (SignImm <<2), SignImm为imm符号扩展为32位的有符号数



Single-Cycle MIPS Processor Design

控制通路



Single-Cycle MIPS Processor Design

I类型指令

addi rt, rs, imm

31	opcode ²⁶	25 21	20 16	15 0
0	0 1 0 0 0	rs	rt	imm

imm需要符号扩展为32位有符号数

ori rt, rs, imm

opcode 26	25 21	20 16	15 0
0 0 1 1 0 1	rs	rt	imm

imm需要符号扩展为32位有符号数

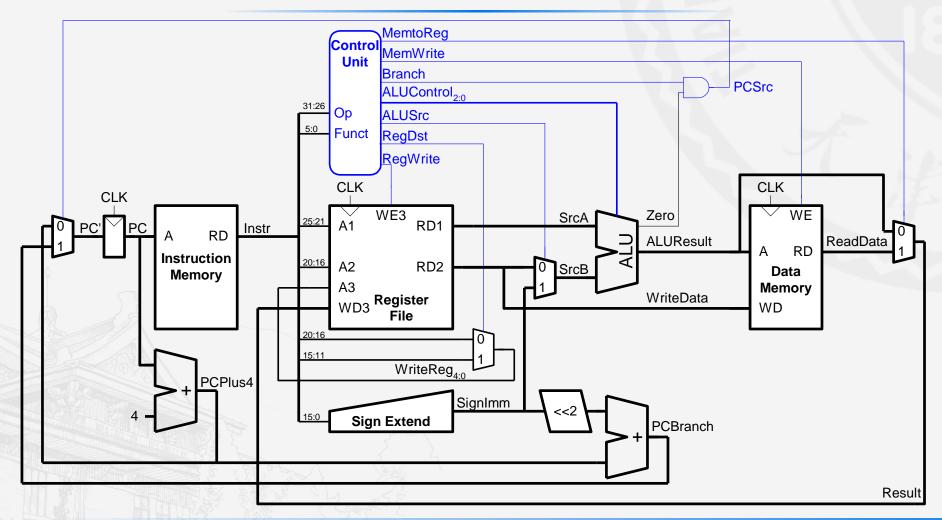
 31 opcode
 26 25
 21 20
 16 15
 0

 1 0 0 0 1 1
 rs
 rt
 imm

lw rt, imm(rs)

Single-Cycle MIPS Processor Design

addi/ori指令的实现:数据通路没有任何变化



Single-Cycle MIPS Processor Design

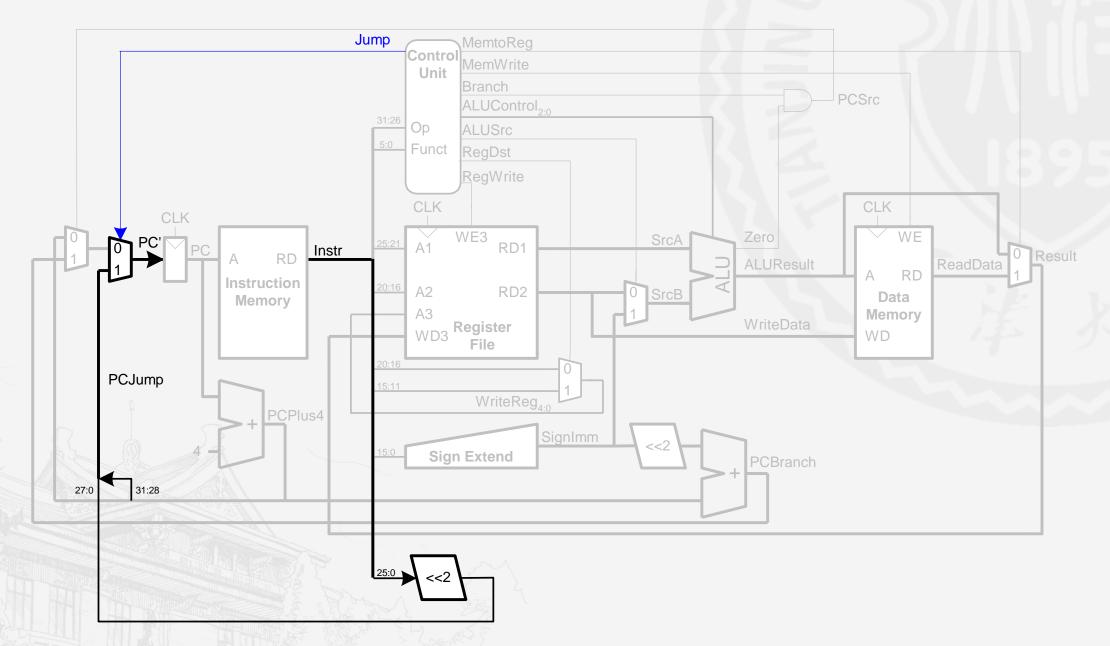
label

J类型指令

31 opcode 26 25 0 0 0 0 1 0 addr

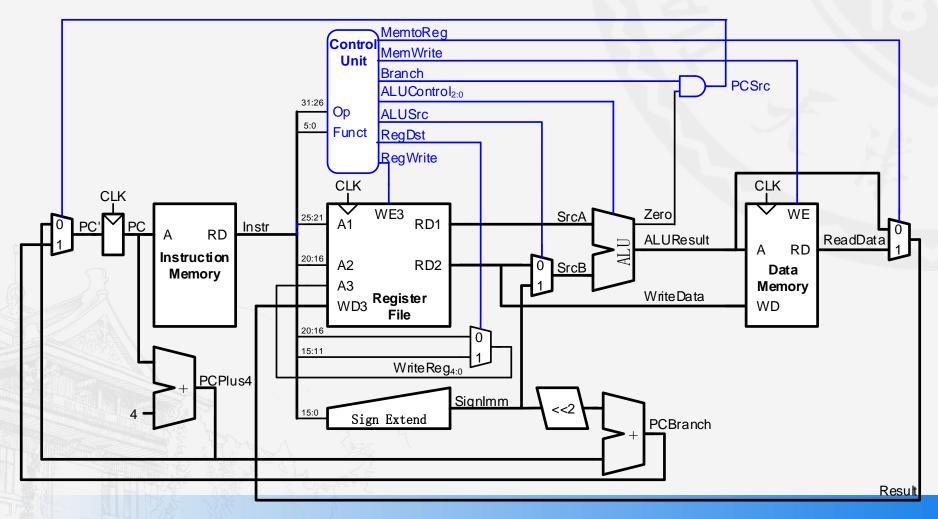
跳转的目标地址为 { (PC+4)[31:28], addr, 2'b0 }

跳转的目标地址为 { (PC+4)[31:28], addr, 2'b0 }



控制器逻辑

Instruction	Op _{5:0}	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUOp _{1:0}	Jump
R-type	000000	1	1	0	0	0	0	由funct决定	0
lw	100011	1	0	1	0	0	1	加法	0
SW	101011	0	Х	1	0	1	Х	加法	0
beq	000100	0	Х	0	1	0	Х	减法	0
j	000010	0	Х	Х	Х	0	Х	XX	1



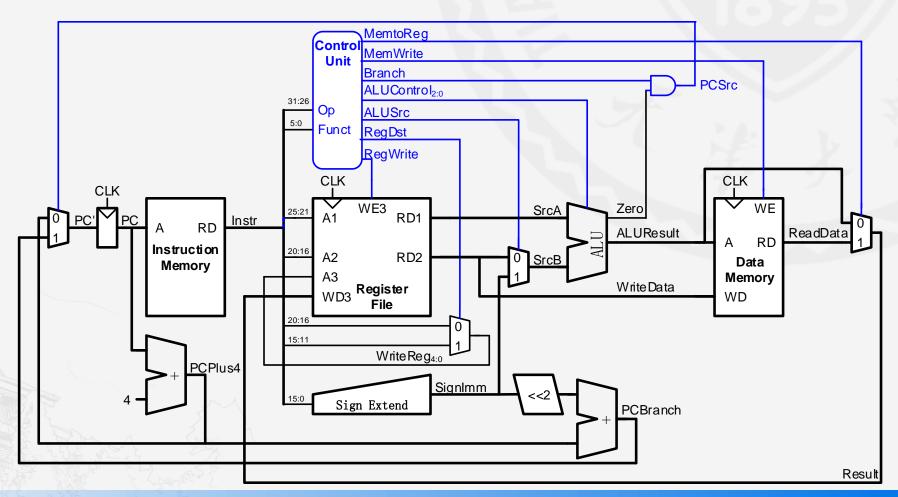
问题: sll指令如何实现?

sll rd, rt, shamt

 31
 opcode
 26
 25
 21
 20
 16
 15
 11
 10
 6
 5
 funct
 0

 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0

 $[rd] = [rt] \ll shamt$





Single-Cycle MIPS Processor Design

实验要求

■ 支持 10 条指令: lw SW lui ori addiu addu slt beq bne ■ 每条指令均在一个时钟周期完成

采用哈佛结构

