第五章　单周期处理器

1. 处理器设计的 主要步骤
2. 数据通路的建立
3.运算指令的 控制信号
4.访存指令的 控制信号
5.分支指令的 控制信号
6.控制信号的集成

不同维度的指令分类

	R型指令	I型指令	J型指令
运算指令	addu rd,rs,rt subu rd,rs,rt	ori rt,rs,*imm16*	
访存指令		lw rt,*imm16*(rs) sw rt,*imm16*(rs)	
分支指令		beq rs,rt,*imm16*	

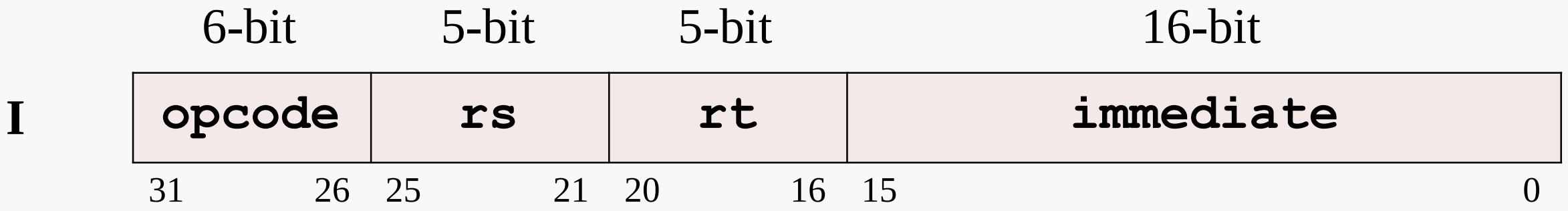
lw指令的操作步骤

lw　rt,　imm16(rs)

① MEM[PC] 从指令存储器中取回指令

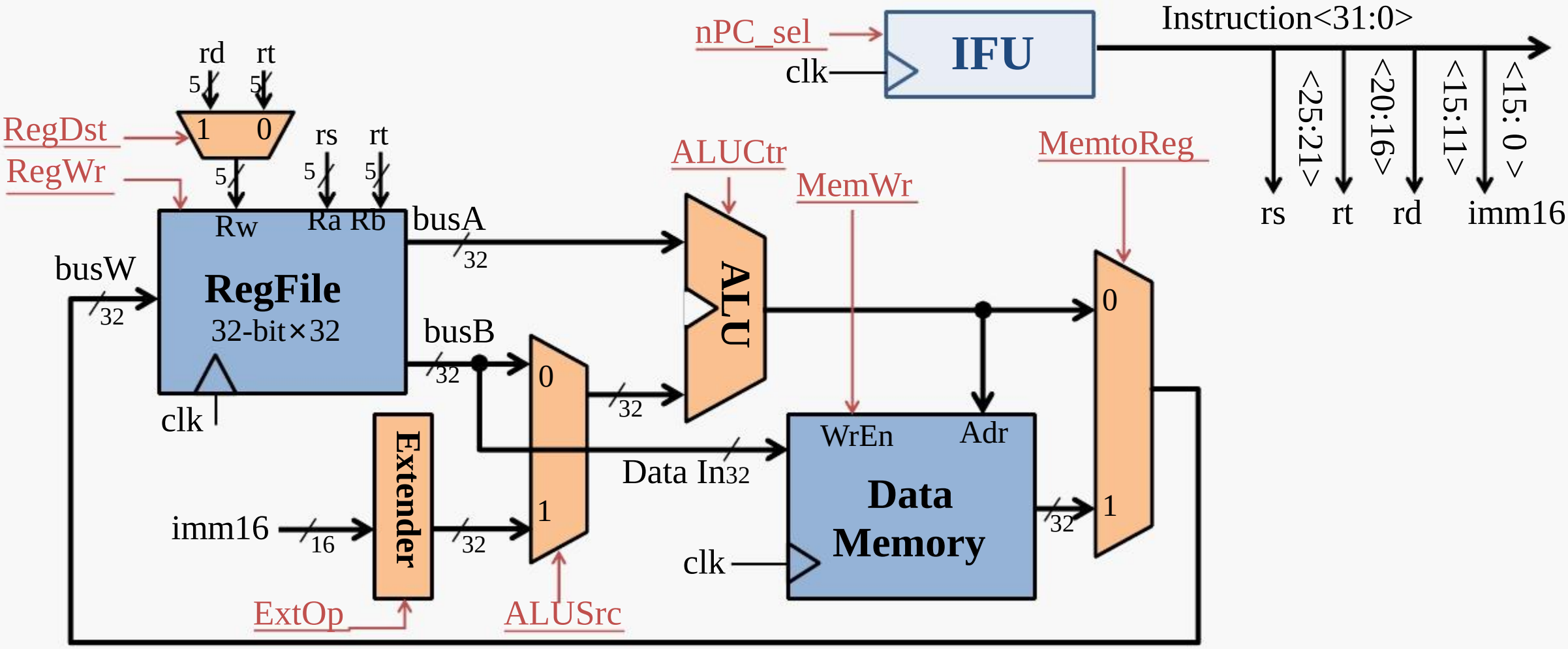
② R[rt]=DataMemory{R[rs]+SignExt[imm16]} 指令指定的操作

③ PC=PC + 4 计算下一条指令的地址



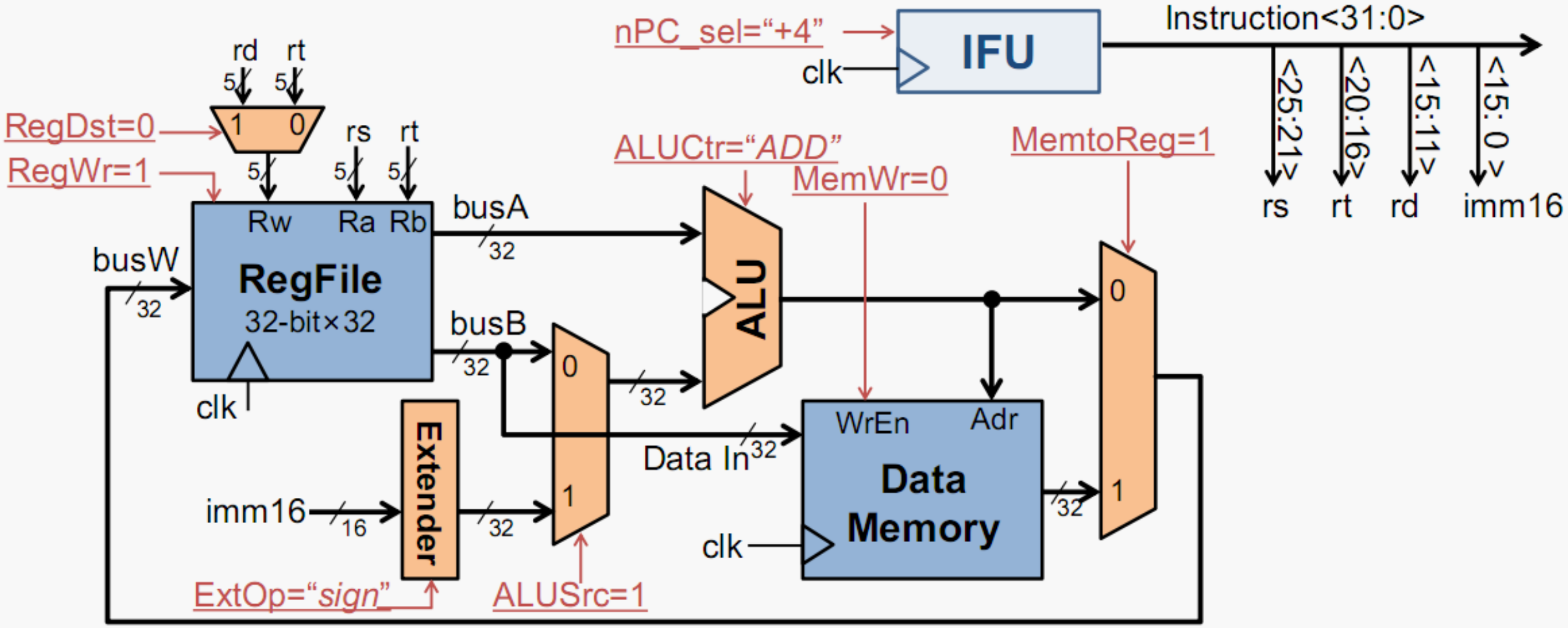
lw指令的操作步骤（2）

R[rt]=Data Memory{R[rs]+SignExt[imm16]}



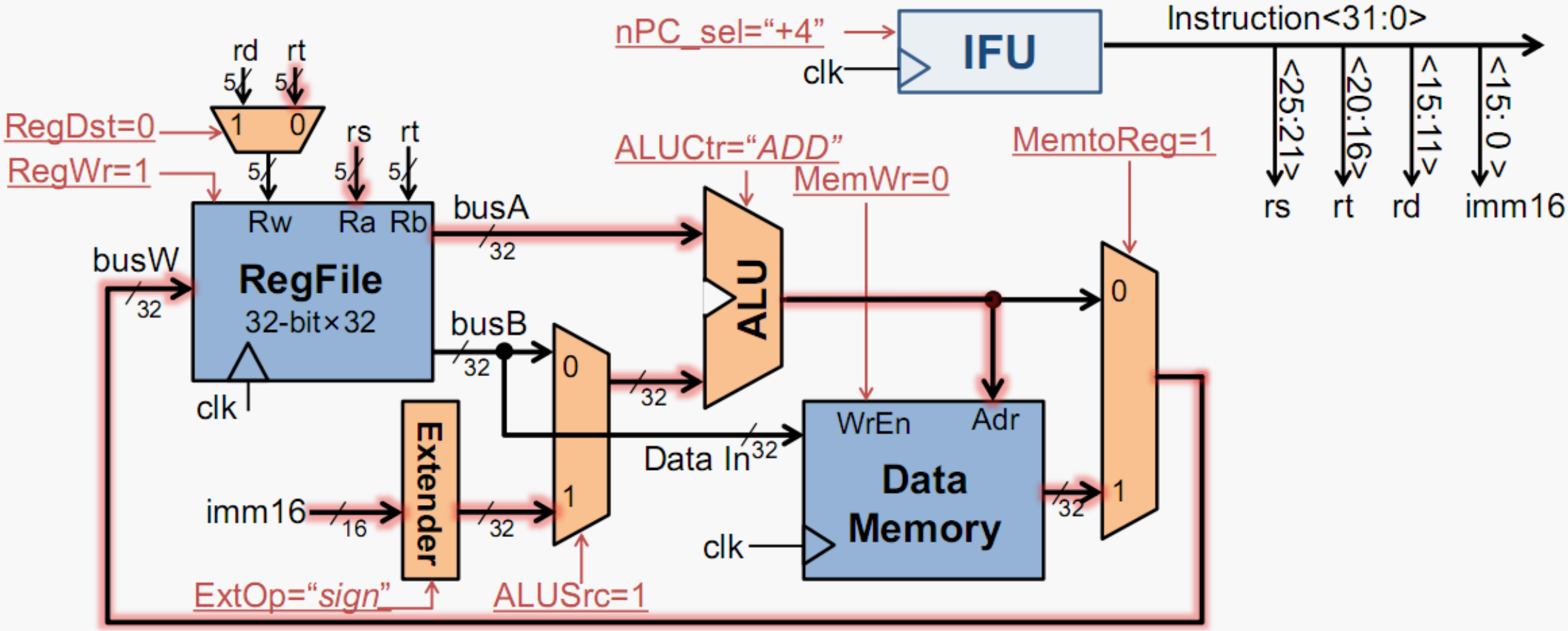
lw指令的操作步骤（2）

R[rt]=Data Memory{R[rs]+SignExt[imm16]}



lw指令的操作步骤（2）

R[rt]=Data Memory{R[rs]+SignExt[imm16]}



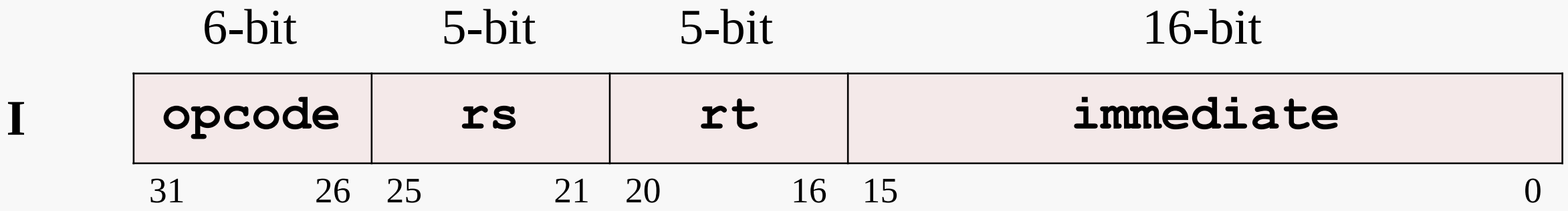
sw指令的操作步骤

```
sw  rt,  imm16(rs)
```

① `MEM[PC]` 从指令存储器中取回指令

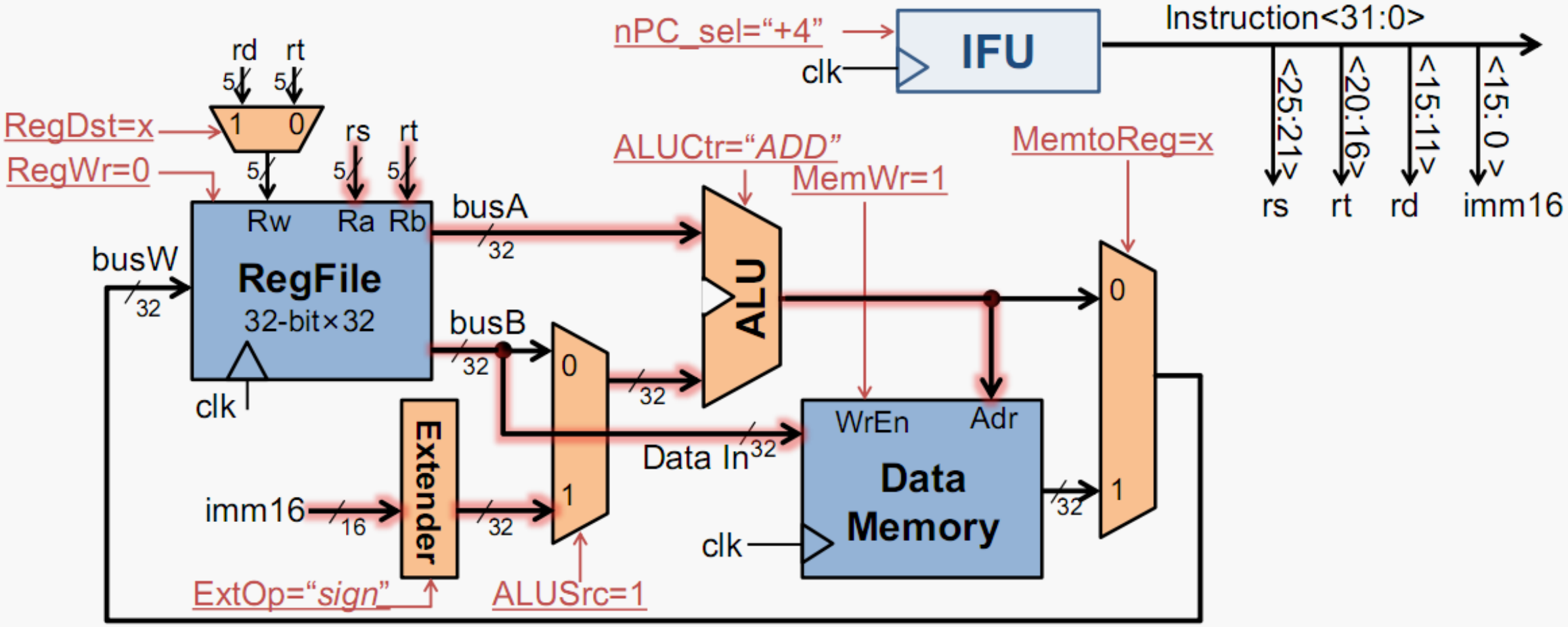
② `DataMemory{R[rs]+SignExt[imm16]}=R[rt]` 指令指定的操作

③ `PC=PC + 4` 计算下一条指令的地址



sw指令的操作步骤（2）

DataMemory{R[rs]+SignExt[imm16]}=R[rt]



第五章　单周期处理器

1. 处理器设计的 主要步骤
2. 数据通路的建立
3.运算指令的 控制信号
4.访存指令的 控制信号
5.分支指令的 控制信号
6.控制信号的集成

不同维度的指令分类

	R型指令	I型指令	J型指令
运算指令	addu rd,rs,rt subu rd,rs,rt	ori rt,rs,*imm16*	
访存指令		lw rt,*imm16*(rs) sw rt,*imm16*(rs)	
分支指令		beq rs,rt,*imm16*	

条件分支指令的示例

```
if(i==j)
     f=g+h;
else
     f=g-h;
```

C语言代码

```
 beq $s3,$s4,True       #  branch  i==j
 sub $s0,$s1,$s2        #  f=g-h(false)
 j    Next              #  goto Next

True: add $s0,$s1,$s2 #  f=g+h (true)
Next: …
```

MIPS汇编语言代码

beq指令的操作步骤

```
beq  rs, rt,  imm16
```

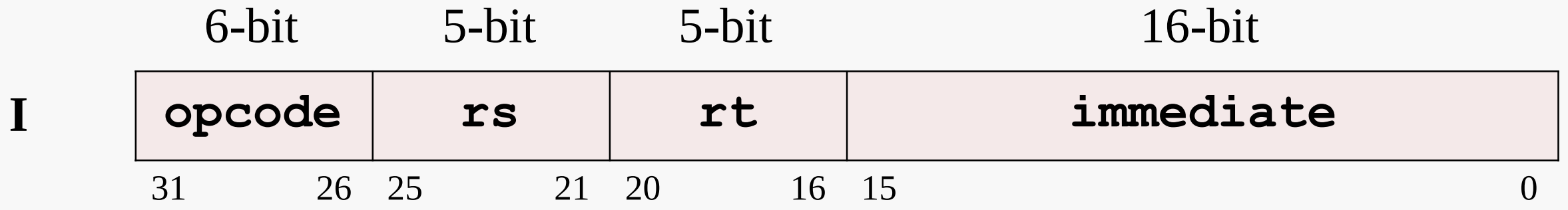
① `MEM[PC]` 从指令存储器中取回指令

② `if(R[rs]-R[rt]==0)` 判断转移条件是否成立

③ `then  PC = PC + 4 + SignExt[imm16]*4 ;`

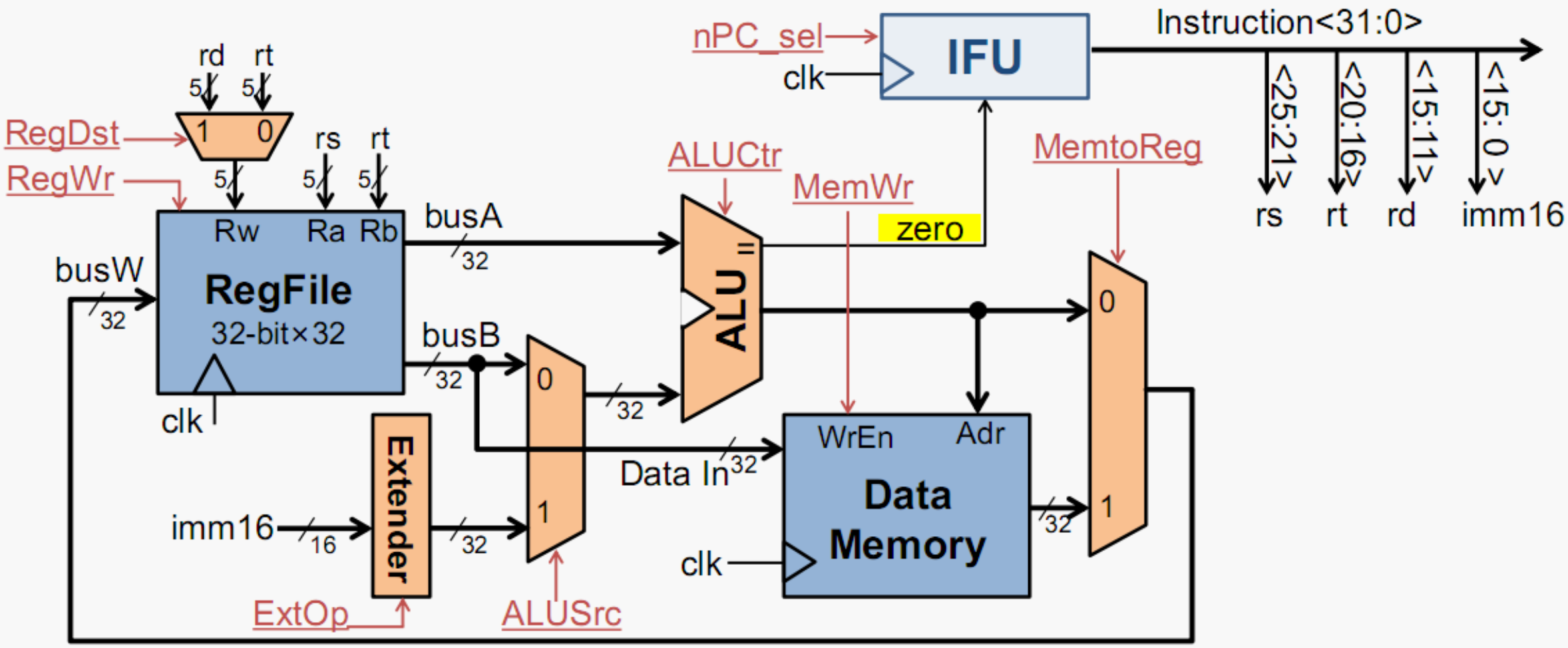
`else  PC = PC + 4 ;`

计算下一条指令的地址



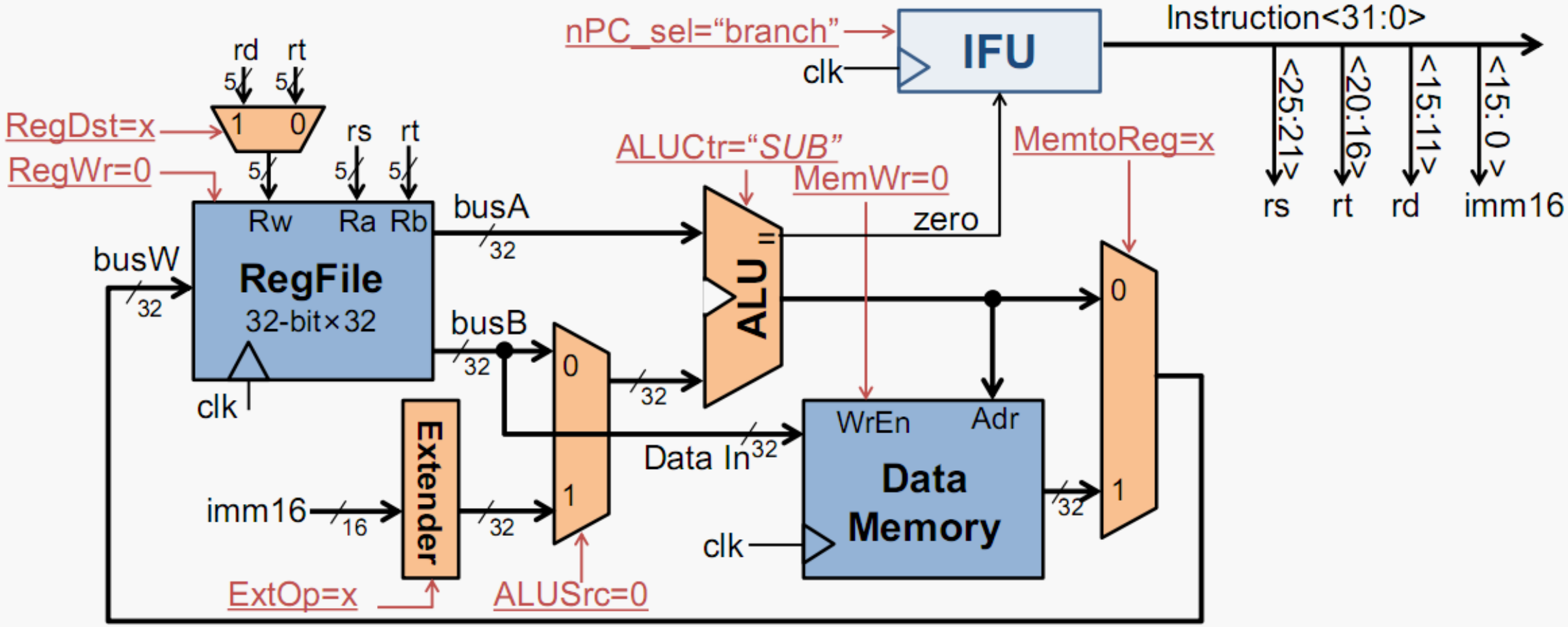
beq指令的操作步骤（2）

```
if (R[rs]-R[rt]==0) then zero=1; else zero=0;
```



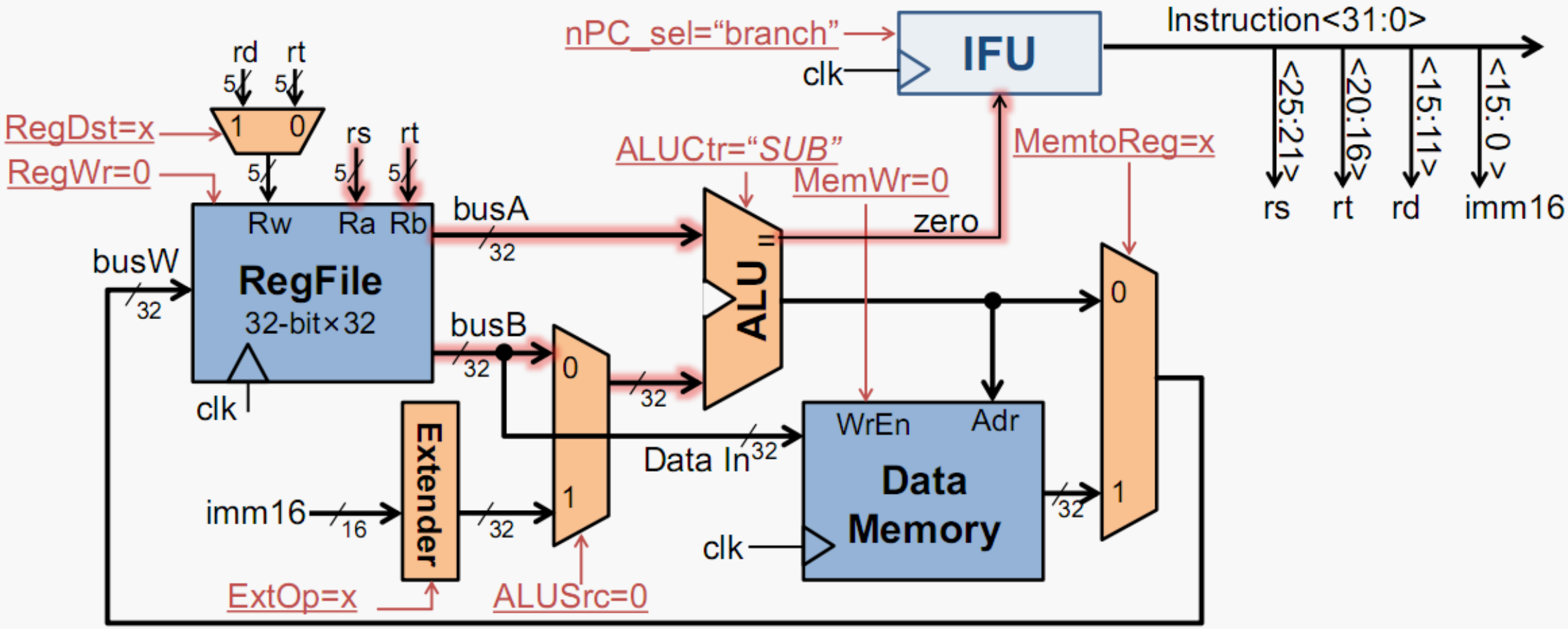
beq指令的操作步骤（2）

```
if (R[rs]-R[rt]==0) then zero=1; else zero=0;
```



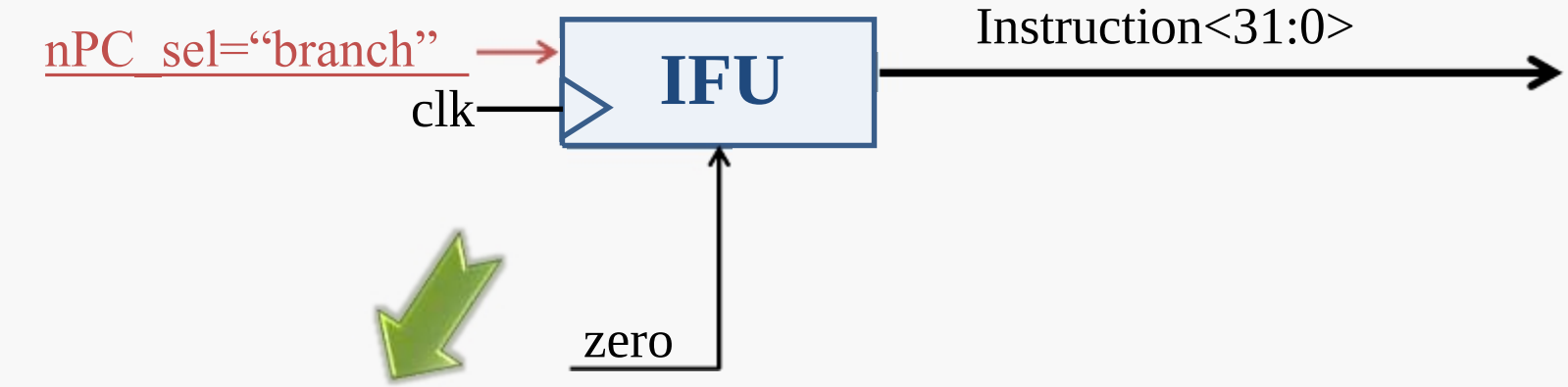
beq指令的操作步骤（2）

```
if (R[rs]-R[rt]==0) then zero=1; else zero=0;
```

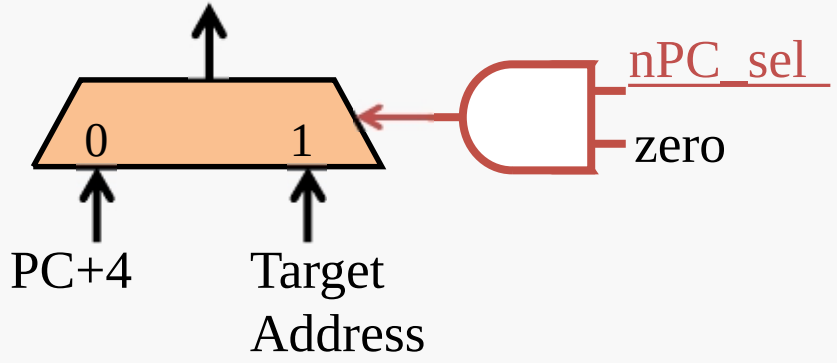


beq指令的操作步骤（3）

```
if(zero==0) then PC=PC+4 + SignExt[imm16]*4 ;
      else PC=PC+4;
```

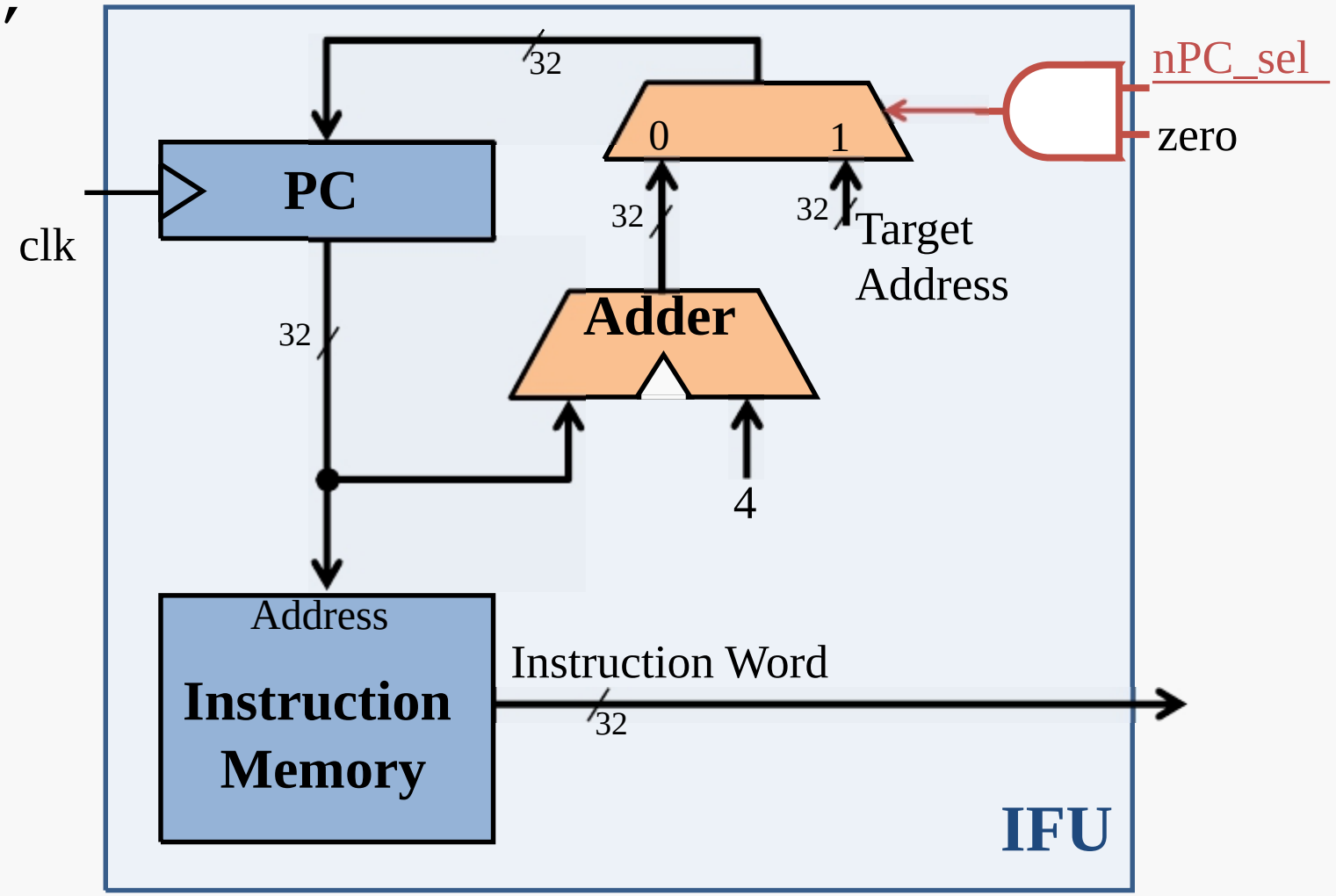


nPC_sel	zero	MUX
0 ("+4")	x	0 (PC+4)
1 ("branch")	0 (≠)	0 (PC+4)
1 ("branch")	1 (=)	1 (Target Address)



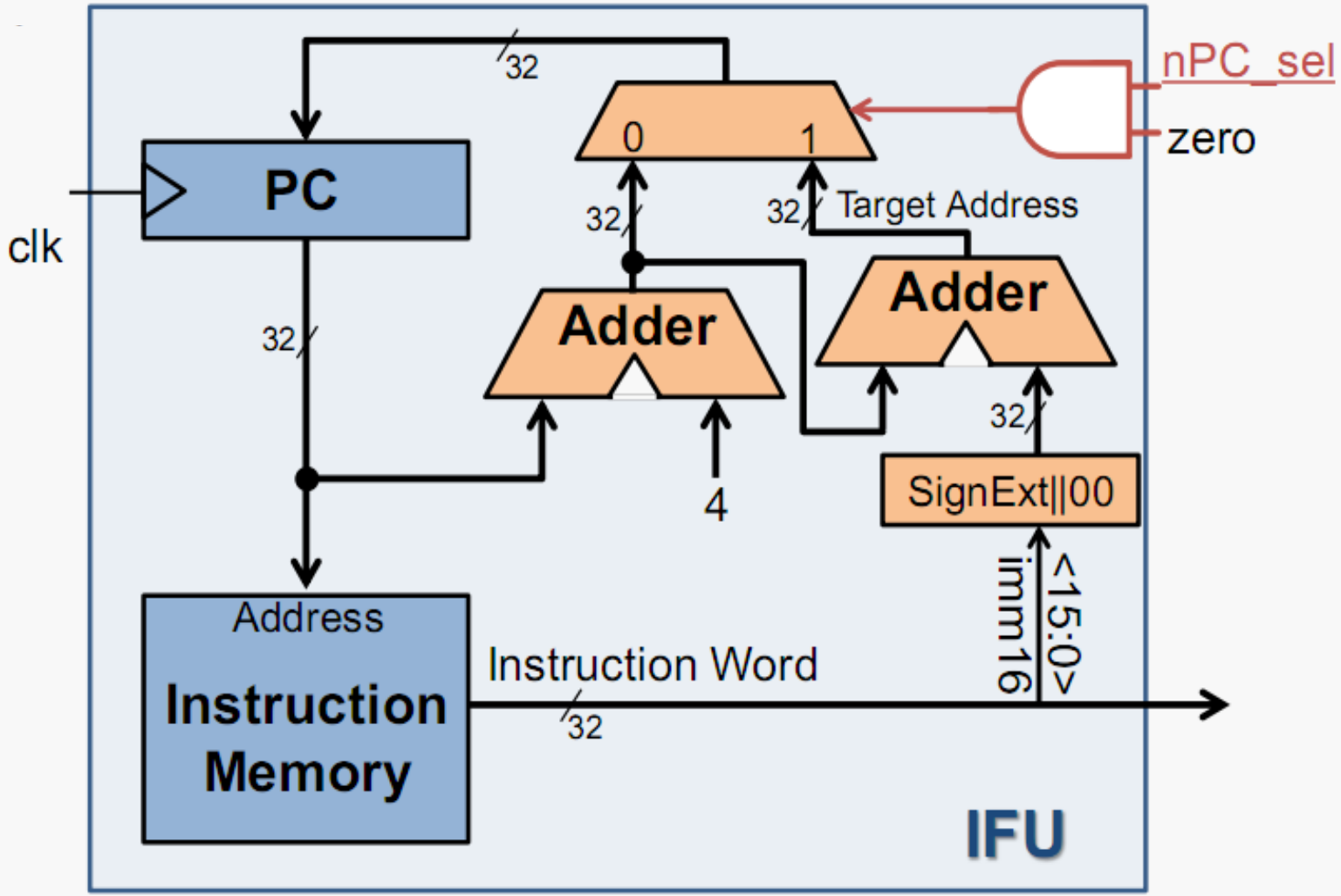
beq指令的操作步骤（3）

```
if(zero==0) then PC=PC+4 + SignExt[imm16]*4 ;
      else PC=PC+4;
```



beq指令的操作步骤（3）

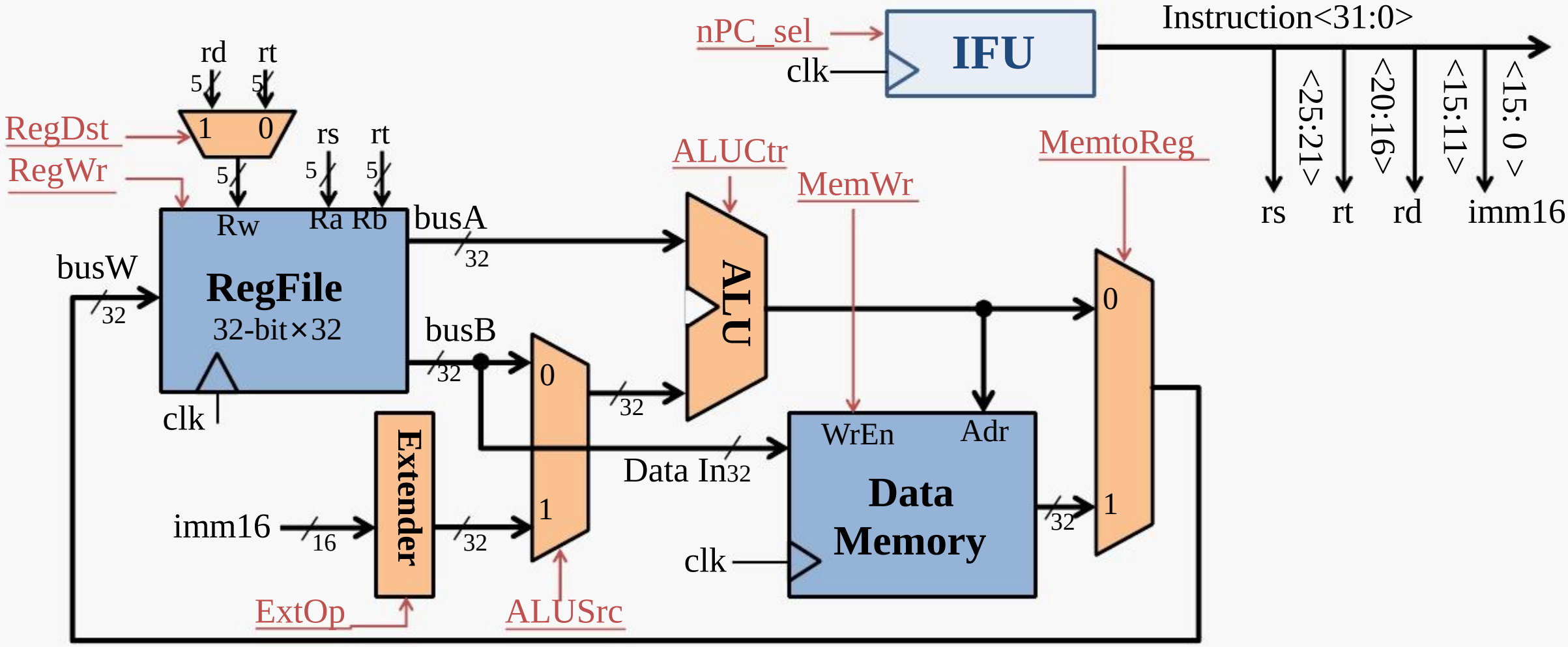
```
if(zero==0) then PC=PC+4 + SignExt[imm16]*4 ;
      else PC=PC+4;
```



第五章　单周期处理器

1. 处理器设计的 主要步骤
2. 数据通路的建立
3.运算指令的 控制信号
4.访存指令的 控制信号
5.分支指令的 控制信号
6.控制信号的集成

现有指令所需的控制信号



处理器的设计步骤

① 分析指令系统，得出对数据通路的需求 ✓

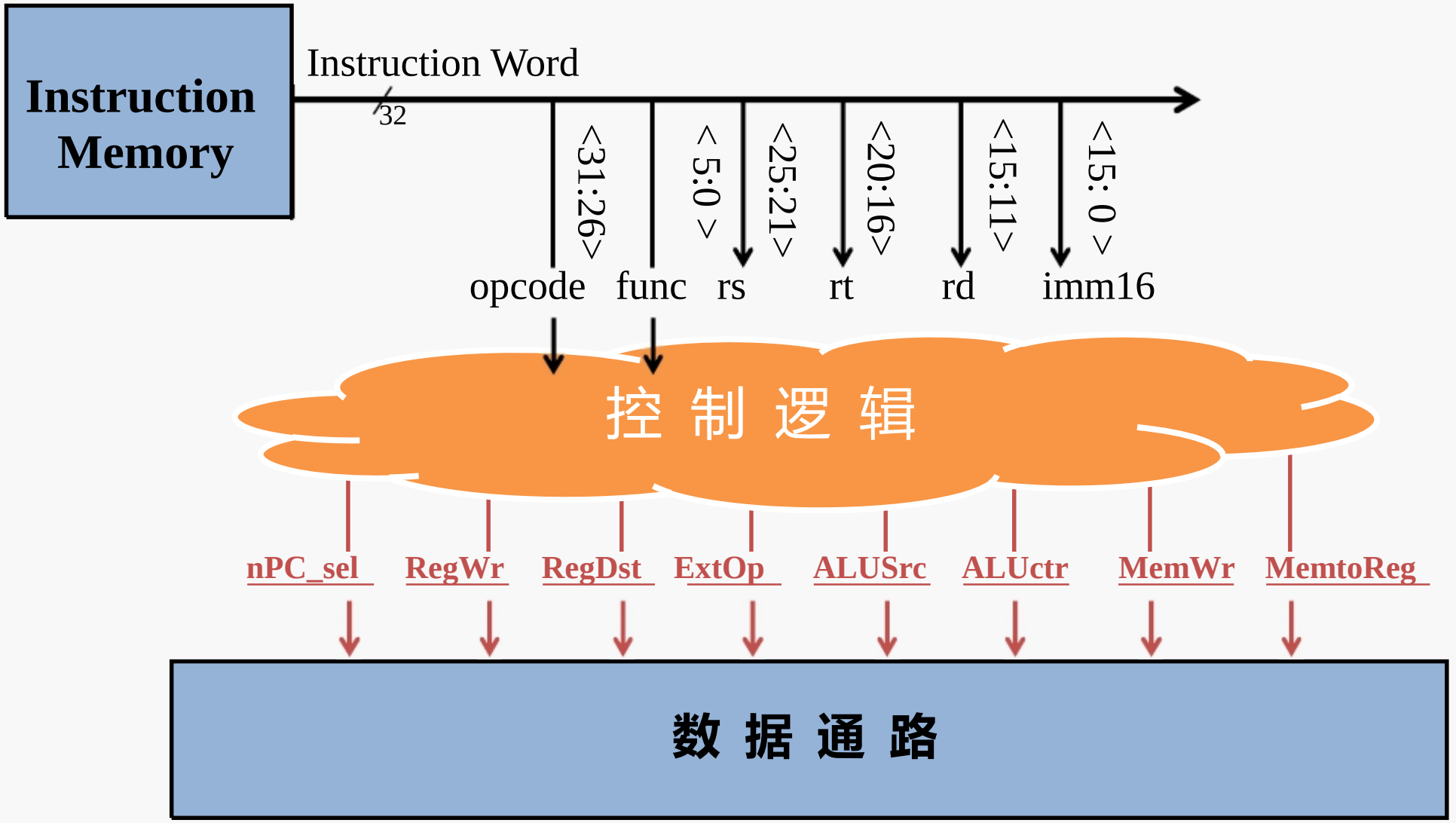
② 为数据通路选择合适的组件 ✓

③ 连接组件建立数据通路 ✓

④ 分析每条指令的实现，以确定控制信号 ✓

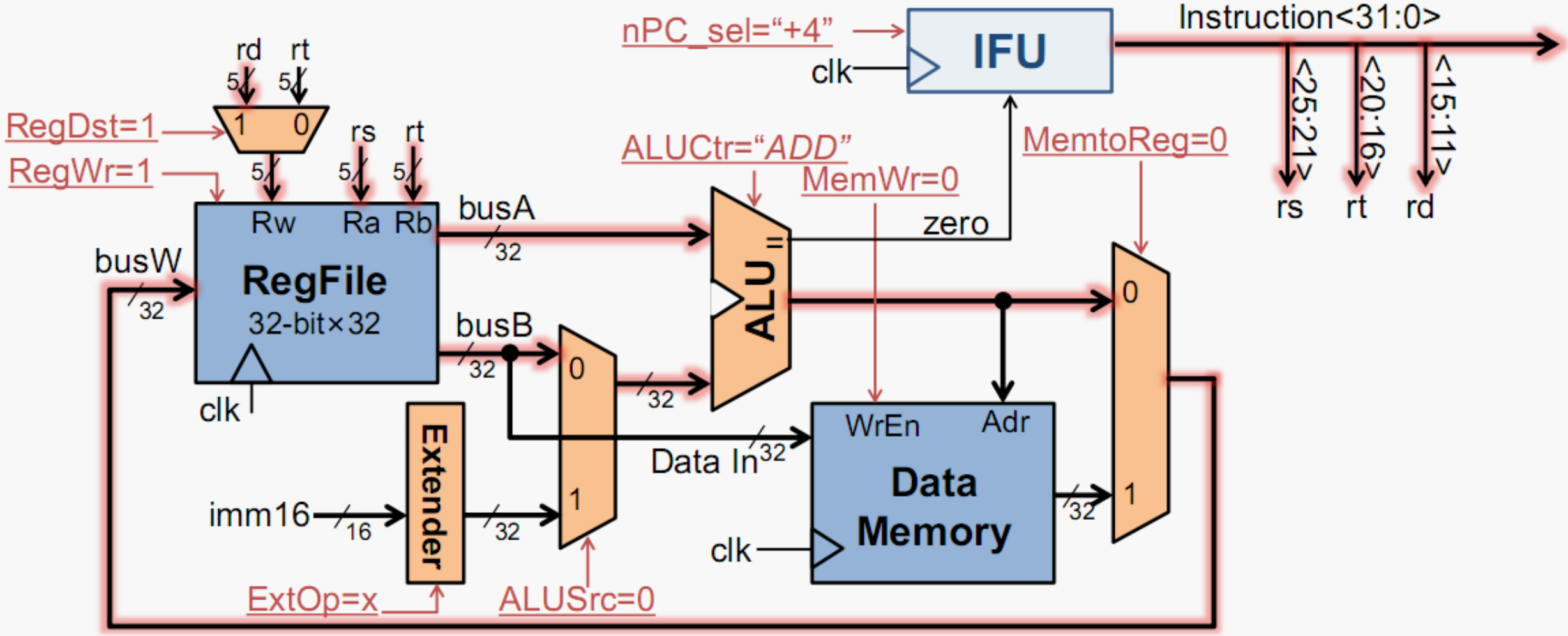
⑤ 集成控制信号，形成完整的控制逻辑

控制逻辑与数据通路


Instruction Memory
Instruction Word
32
<31:26>
<5:0>
<25:21>
<20:16>
<15:11>
<15:0>
opcode
func
rs
rt
rd
imm16
控 制 逻 辑
nPC_sel
RegWr
RegDst
ExtOp
ALUSrc
ALUctr
MemWr
MemtoReg
数 据 通 路

控制信号的汇总（以add指令为例）

```
add: R[rd] ⟵ R[rs] + R[rt]; PC ⟵ PC + 4
```



控制信号的逻辑表达式

func	100000	100010	/			
opcode(op)	000000	000000	001101	100011	101011	000100
	add	sub	ori	lw	sw	beq
RegDst	1	1	0	0	x	x
ALUSrc	0	0	1	1	1	0
MemtoReg	0	0	0	1	x	x
RegWr	1	1	1	1	0	0
MemWr	0	0	0	0	1	0
nPC_sel	0	0	0	0	0	1
ExtOp	x	x	0	1	1	x
ALUctr<1:0>	00(ADD)	01(SUB)	10(OR)	00(ADD)	00(ADD)	01(SUB)

控制信号的逻辑表达式

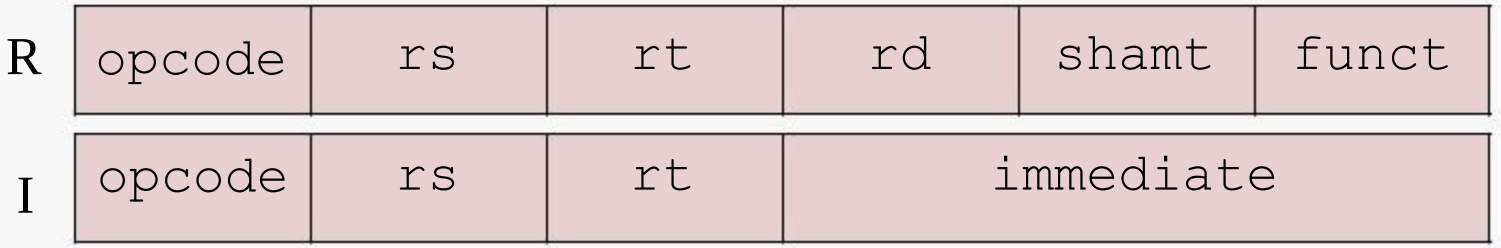
func	100000	100010	/			
opcode(op)	000000	000000	001101	100011	101011	000100
	add	sub	ori	lw	sw	beq
RegDst	1	1	0	0	x	x

RegDst = add + sub

add =rtype· func5·~func4·~func3·~func2·~func1·~func0

sub =rtype· func5·~func4·~func3·~func2· func1·~func0

rtype =~op5·~op4·~op3·~op2·~op1·~op0

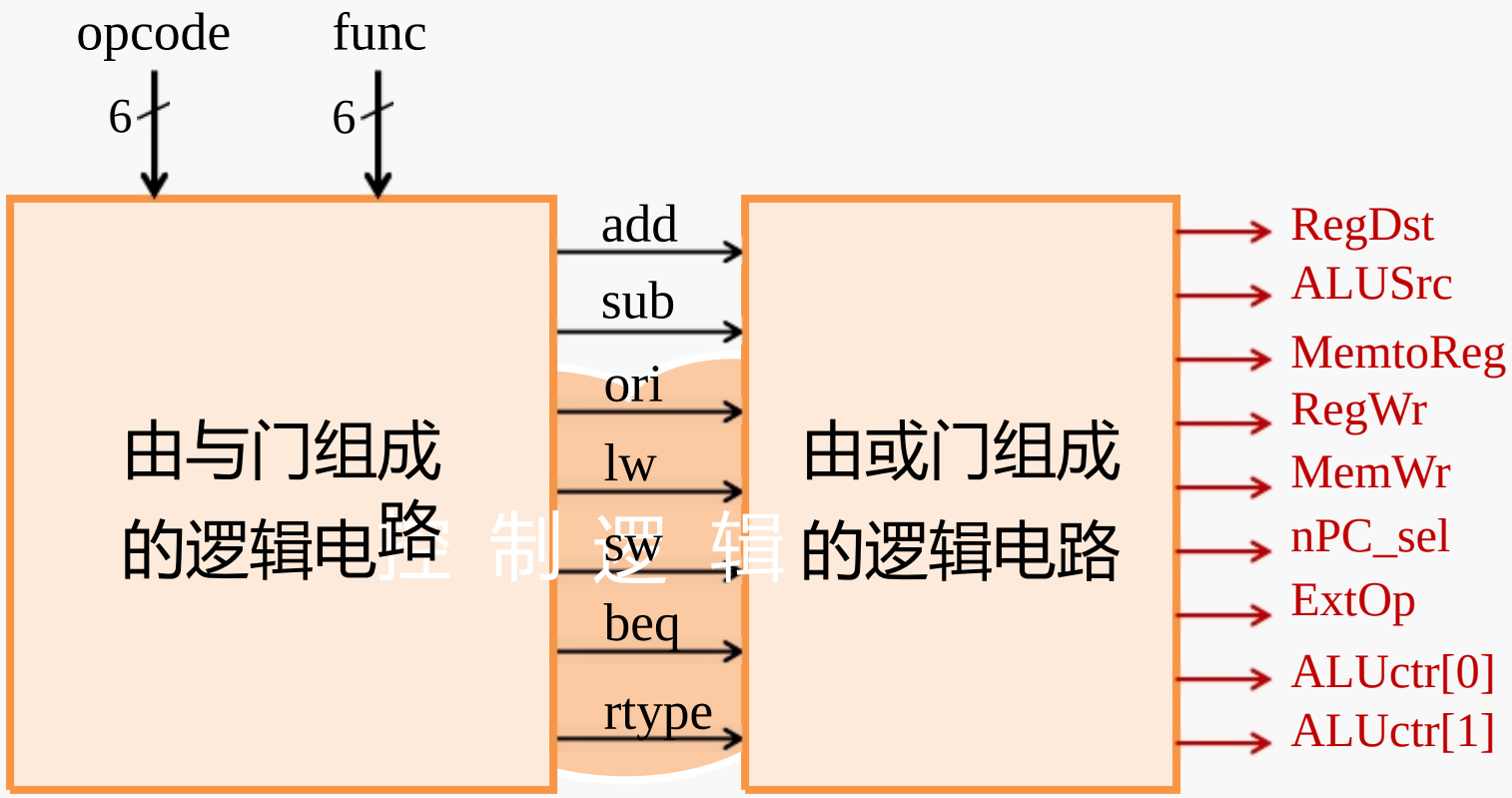


控制器的逻辑表达式

```
RegDst    = add + sub
ALUSrc    = ori + lw + sw
MemtoReg  = lw
RegWr     = add + sub + ori + lw
MemWr     = sw
nPC_sel   = beq
ExtOp     = lw + sw
ALUctr[0] = sub + beq
ALUctr[1] = or
```

```
add   = rtype·func5·~func4·~func3·~func2·~func1·~func0
sub   = rtype·func5·~func4·~func3·~func2· func1·~func0
rtype = ~op5·~op4·~op3·~op2·~op1·~op0,
ori   = ~op5·~op4· op3· op2·~op1· op0
lw    =  op5·~op4·~op3·~op2· op1· op0
sw    =  op5·~op4· op3·~op2· op1· op0
beq   = ~op5·~op4·~op3· op2·~op1·~op0
```

控制器的实现示意图


opcode
6
func
6
由与门组成的逻辑电路
add
sub
ori
lw
sw
beq
rtype
由或门组成的逻辑电路
RegDst
ALUSrc
MemtoReg
RegWr
MemWr
nPC_sel
ExtOp
ALUctr[0]
ALUctr[1]

处理器的设计步骤

① 分析指令，得出对数据通路的需求 ✓

② 为数据通路选择合适的组件 ✓

③ 连接组件建立数据通路 ✓

④ 分析每条指令的实现，以确定控制信号 ✓

⑤ 集成控制信号，形成完整的控制逻辑 ✓