CSE 31

Computer Organization

Lecture 25 – CPU Control

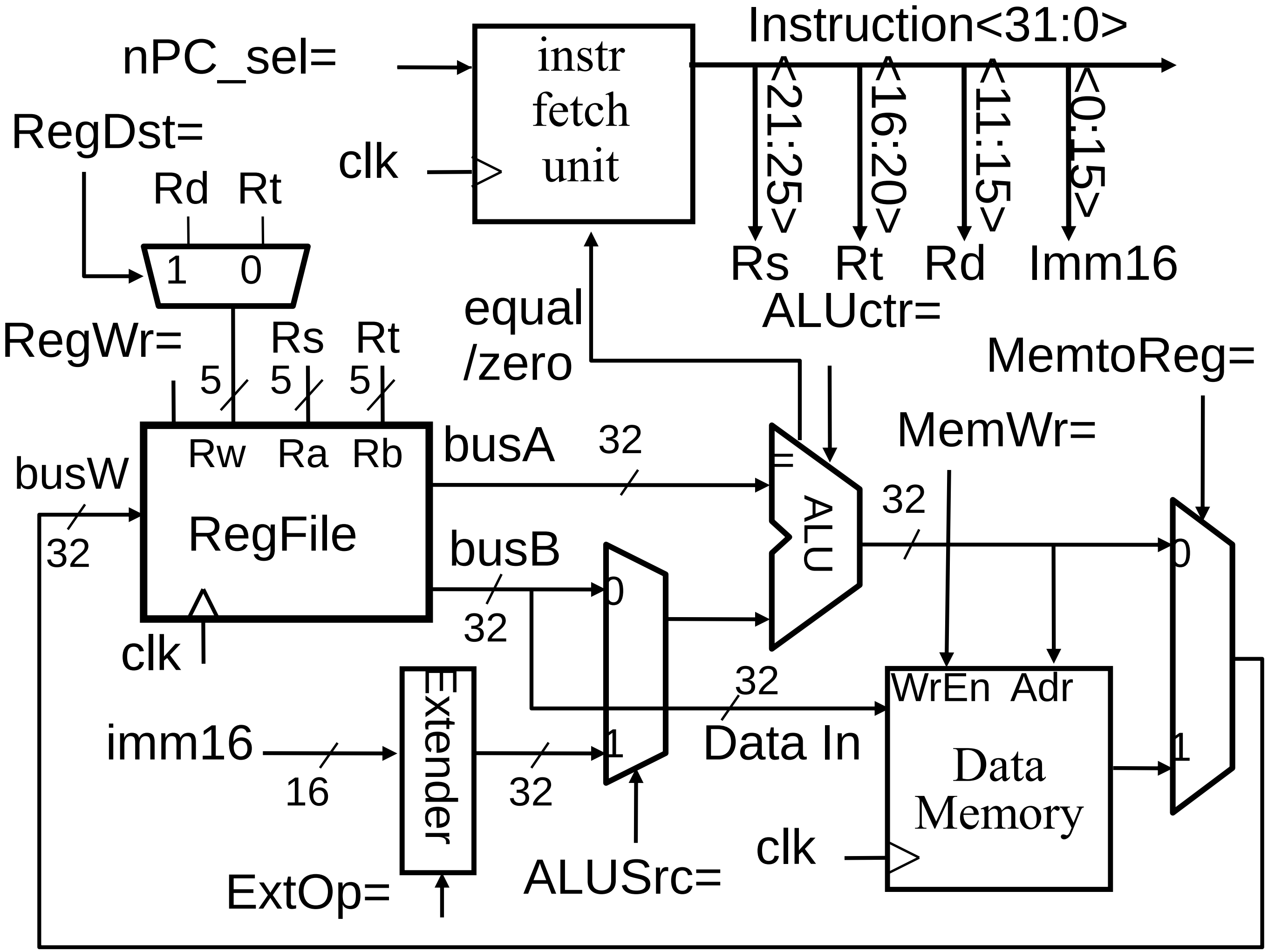
Announcement

- Lab #10
  - Due this week
- Project #2 demo during lab this week
- HW #8 in zyBooks (Through CatCourses)
  - Due Saturday (5/11) at 11:59pm
- Course evaluation online by 5/9

Announcement

- zyBooks assignment Re-dos
  - Re-submit at most 5 reading assignments **or** HW (**zyBooks only**)
  - Email to me (not your TAs)
    - Include your name, assignment numbers
    - (Monday) 5/13 at 11:59pm, no extension
  - Fill out online evaluation by 5/9, Thursday (70% of class)
- Final Exam
  - 5/11 (Saturday), 11:30 – 2:30pm
  - Cover all
  - Practice exam in CatCourses
  - Closed book
  - 2 sheet of note (8.5” x 11”)
  - MIPS reference sheet will be provided
  - Review: 5/10 (Friday) 2-4pm, COB2 140

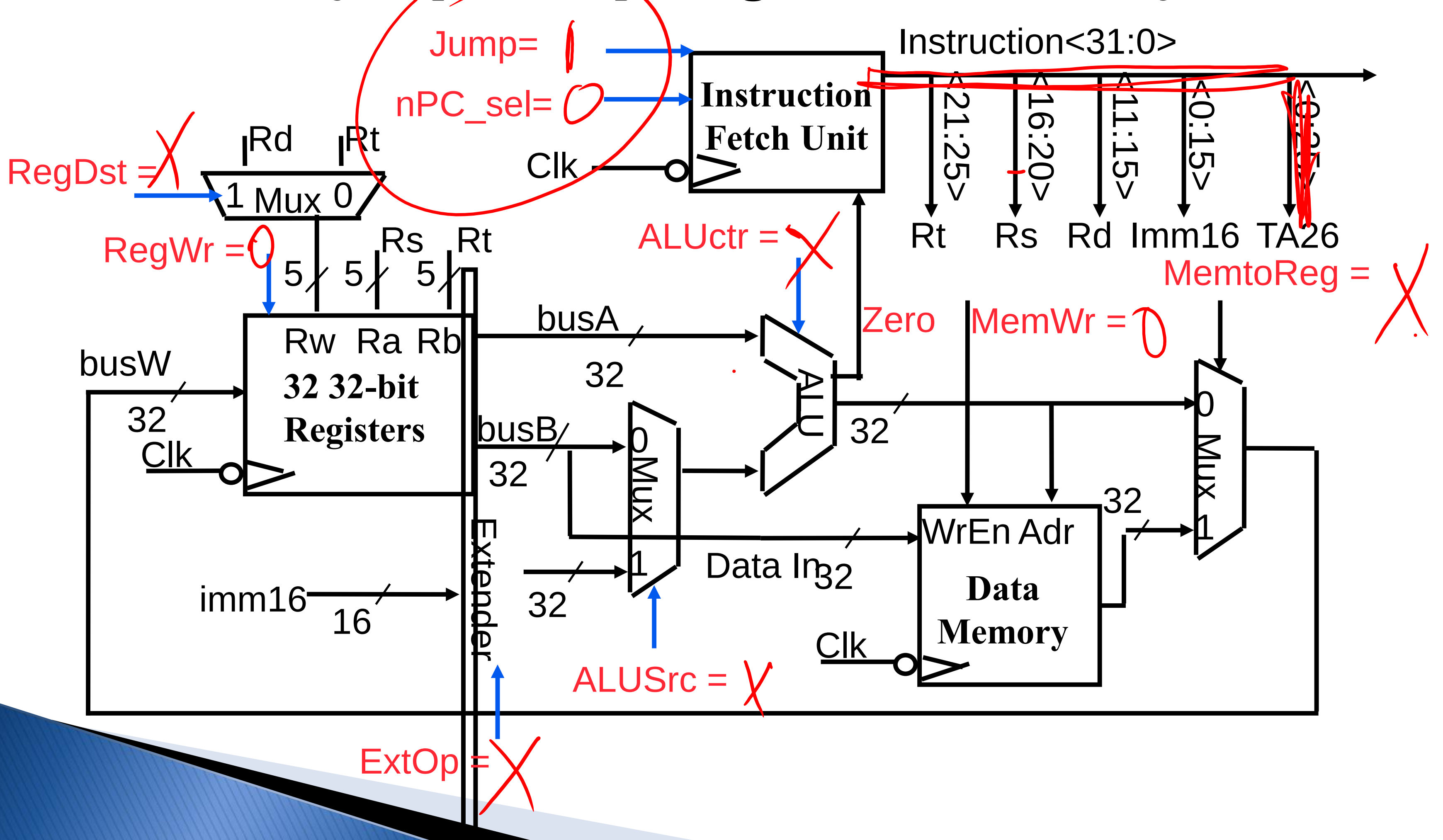
Single Cycle Datapath



The Single Cycle Datapath during Jump

	31 … 26	25 … 0	
J-type	op	target address	jump

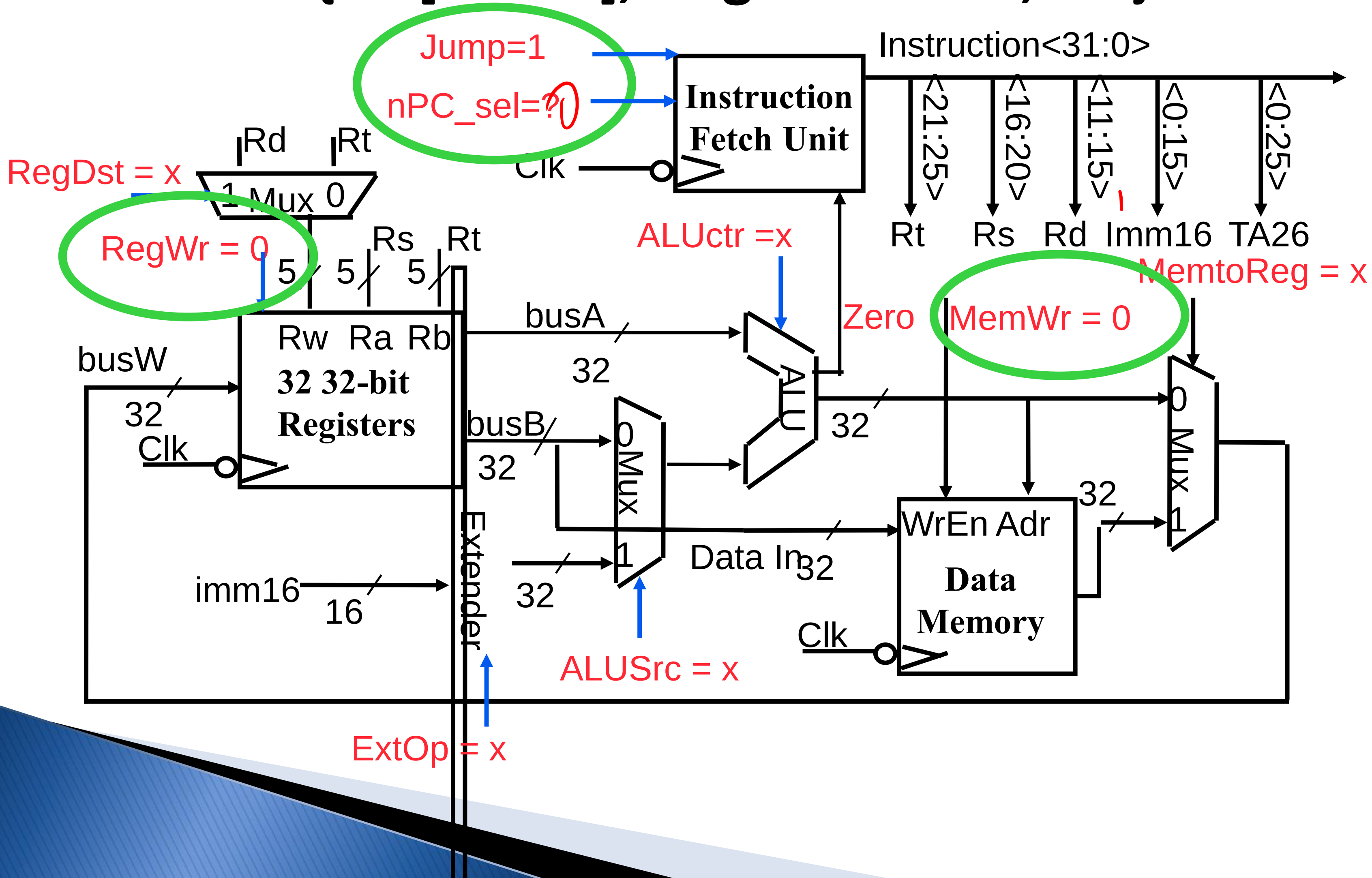
- **New PC = { PC[31..28], target address, 00 }**



The Single Cycle Datapath during Jump

	31 … 26	25 … 0	
J-type	op	target address	jump

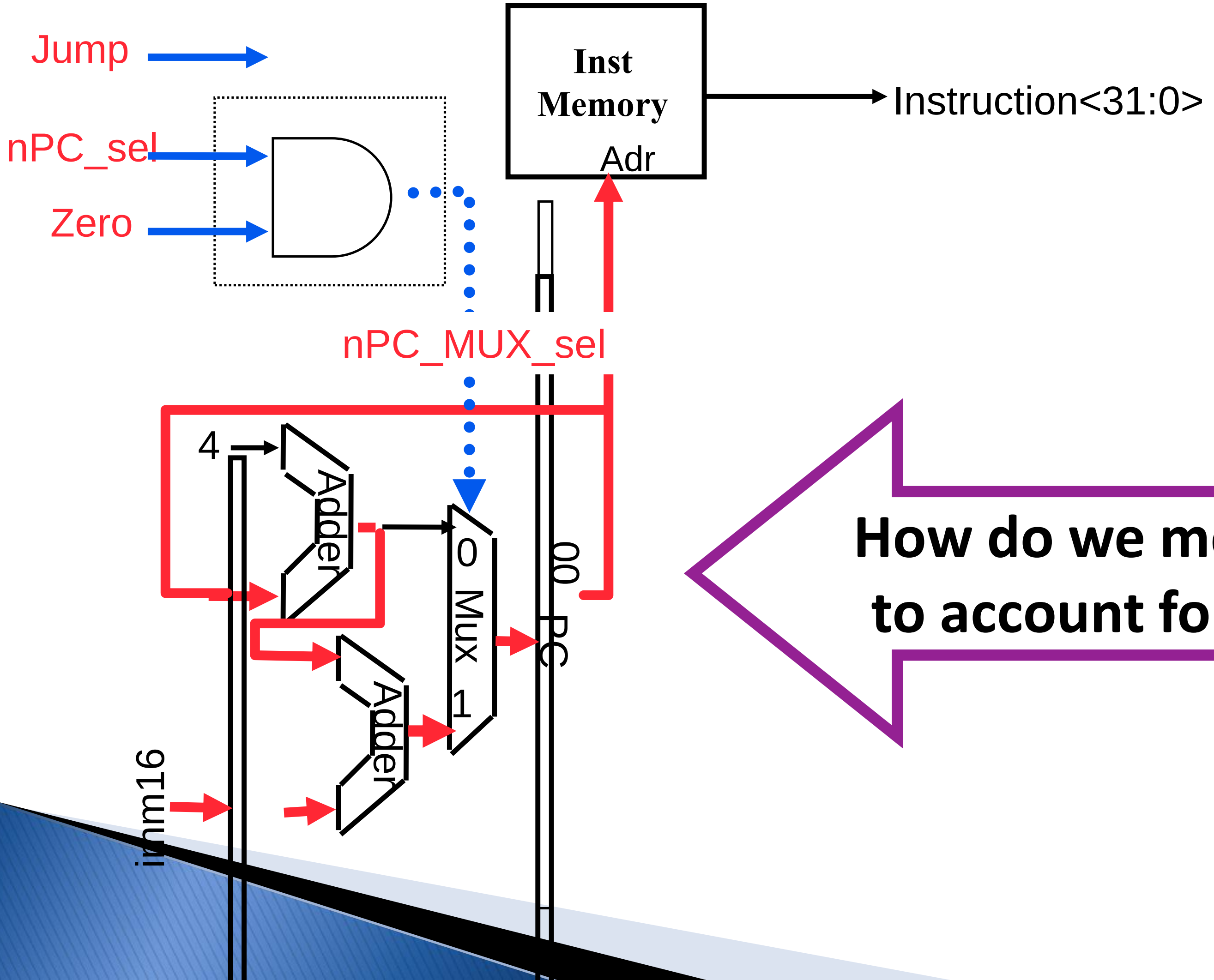
- **New PC = { PC[31..28], target address, 00 }**



Instruction Fetch Unit at the End of Jump

	31 26	25 0	
J-type	op	target address	jump

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

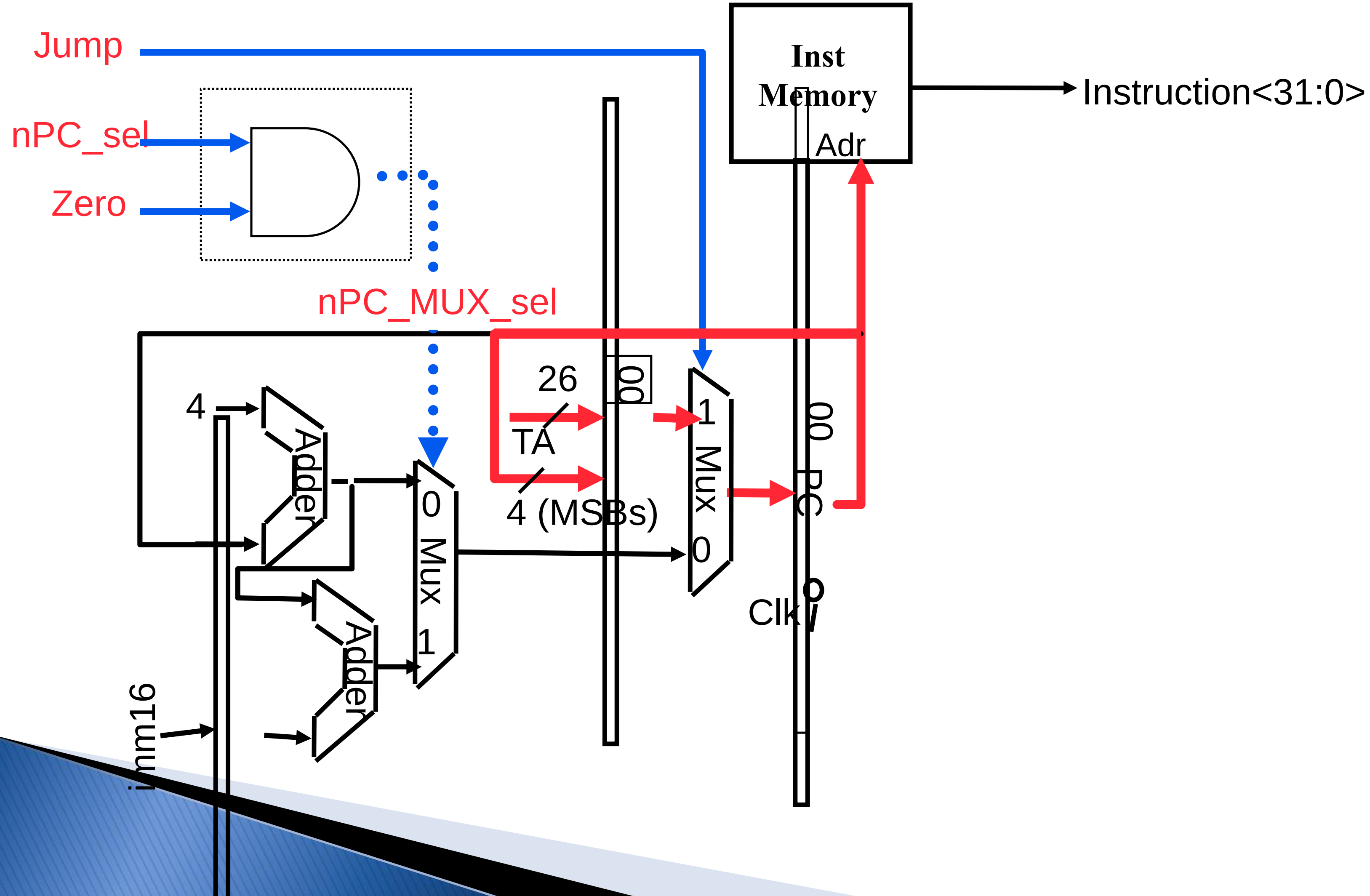


How do we modify this to account for jumps?

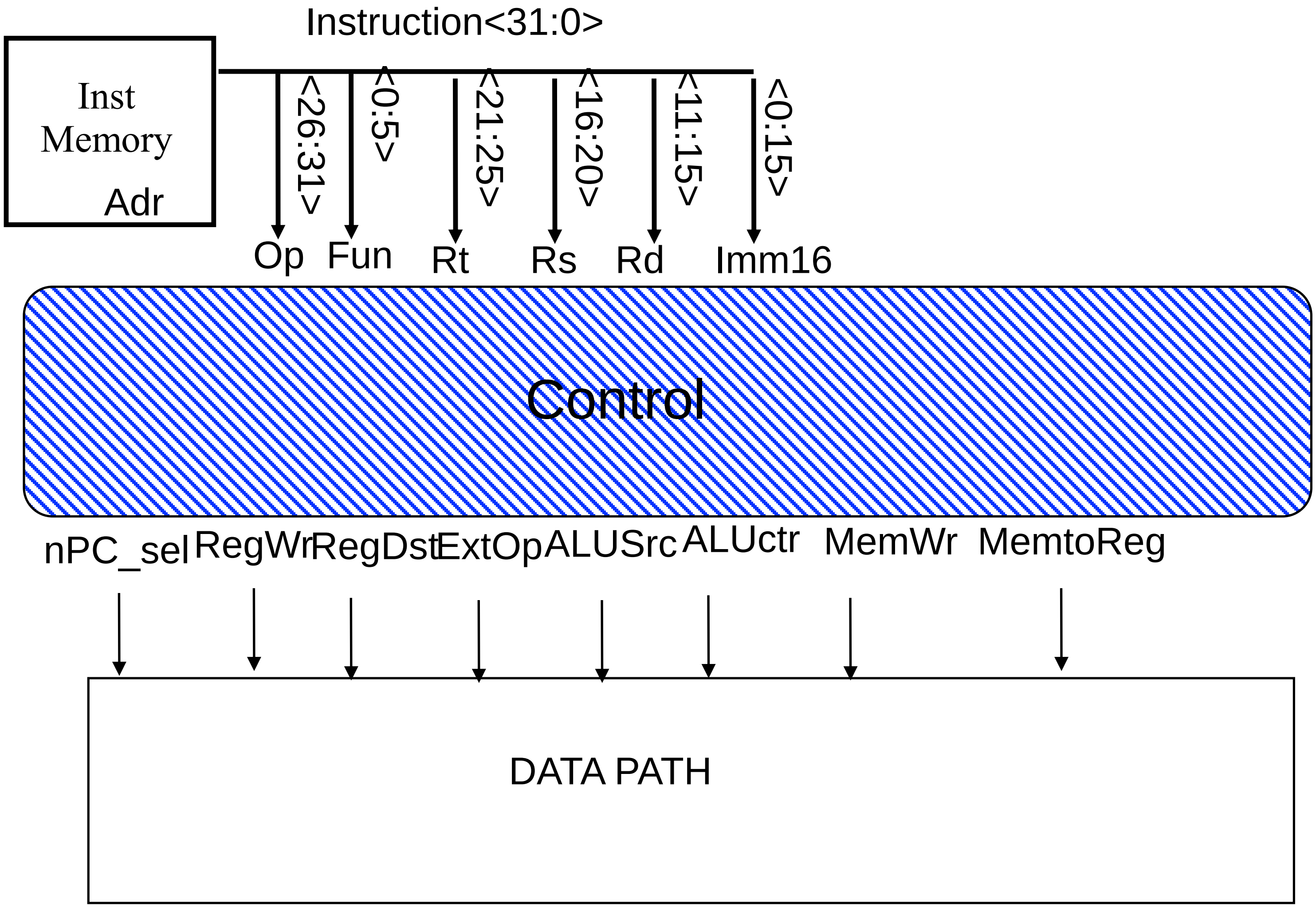
Instruction Fetch Unit at the End of Jump

	31 26	25 0	
J-type	op	target address	jump

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



Control Logic



Control Signals (1/2)

inst Register Transfer

add R[rd] ← R[rs] + R[rt]; PC ← PC + 4

ALUsrc = RegB, ALUctr = "ADD", RegDst = rd, RegWr, nPC_sel = "+4"

sub R[rd] ← R[rs] – R[rt]; PC ← PC + 4

ALUsrc = RegB, ALUctr = "SUB", RegDst = rd, RegWr, nPC_sel = "+4"

ori R[rt] ← R[rs] + zero_ext(Imm16); PC ← PC + 4

ALUsrc = Im, Extop = "Z",ALUctr = "OR", RegDst = rt,RegWr, nPC_sel ="+4"

lw R[rt] ← MEM[R[rs] + sign_ext(Imm16)]; PC ← PC + 4

ALUsrc = Im, Extop = "sn", ALUctr = "ADD", MemtoReg, RegDst = rt, RegWr, nPC_sel = "+4"

sw MEM[R[rs] + sign_ext(Imm16)] ← R[rs]; PC ← PC + 4

ALUsrc = Im, Extop = "sn", ALUctr = "ADD", MemWr, nPC_sel = "+4"

beq if (R[rs] == R[rt]) then PC ← PC + sign_ext(Imm16)] || 00 else PC ← PC + 4

nPC_sel = "br", ALUctr = "SUB"

Control Signals (2/2)

See Appendix A

func	10 0000	10 0010	We Don't Care :-)				
op	00 0000	00 0000	00 1101	10 0011	10 1011	00 0100	00 0010
	add	sub	ori	lw	sw	beq	jump
RegDst	1	1	0	0	x	x	x
ALUSrc	0	0	1	1	1	0	x
MemtoReg	0	0	0	1	x	x	x
RegWrite	1	1	1	1	0	0	0
MemWrite	0	0	0	0	1	0	0
nPCsel	0	0	0	0	0	1	?
Jump	0	0	0	0	0	0	1
ExtOp	x	x	0	1	1	x	x
ALUctr<2:0>	Add	Subtract	Or	Add	Add	Subtract	x

	31	26	21	16	11	6 0	
R-type	op	rs	rt	rd	shamt	funct	add, sub
I-type	op	rs	rt	immediate			ori, lw, sw, beq
J-type	op	target address					jump

Boolean Expressions for Controller

RegDst = add + sub
ALUSrc = ori + lw + sw
MemtoReg = lw
RegWrite = add + sub + ori + lw
MemWrite = sw
nPCsel = beq
Jump = jump
ExtOp = lw + sw
ALUctr[0] = sub + beq (assume ALUctr is 00 ADD, 01: SUB, 10: OR)
ALUctr[1] = or

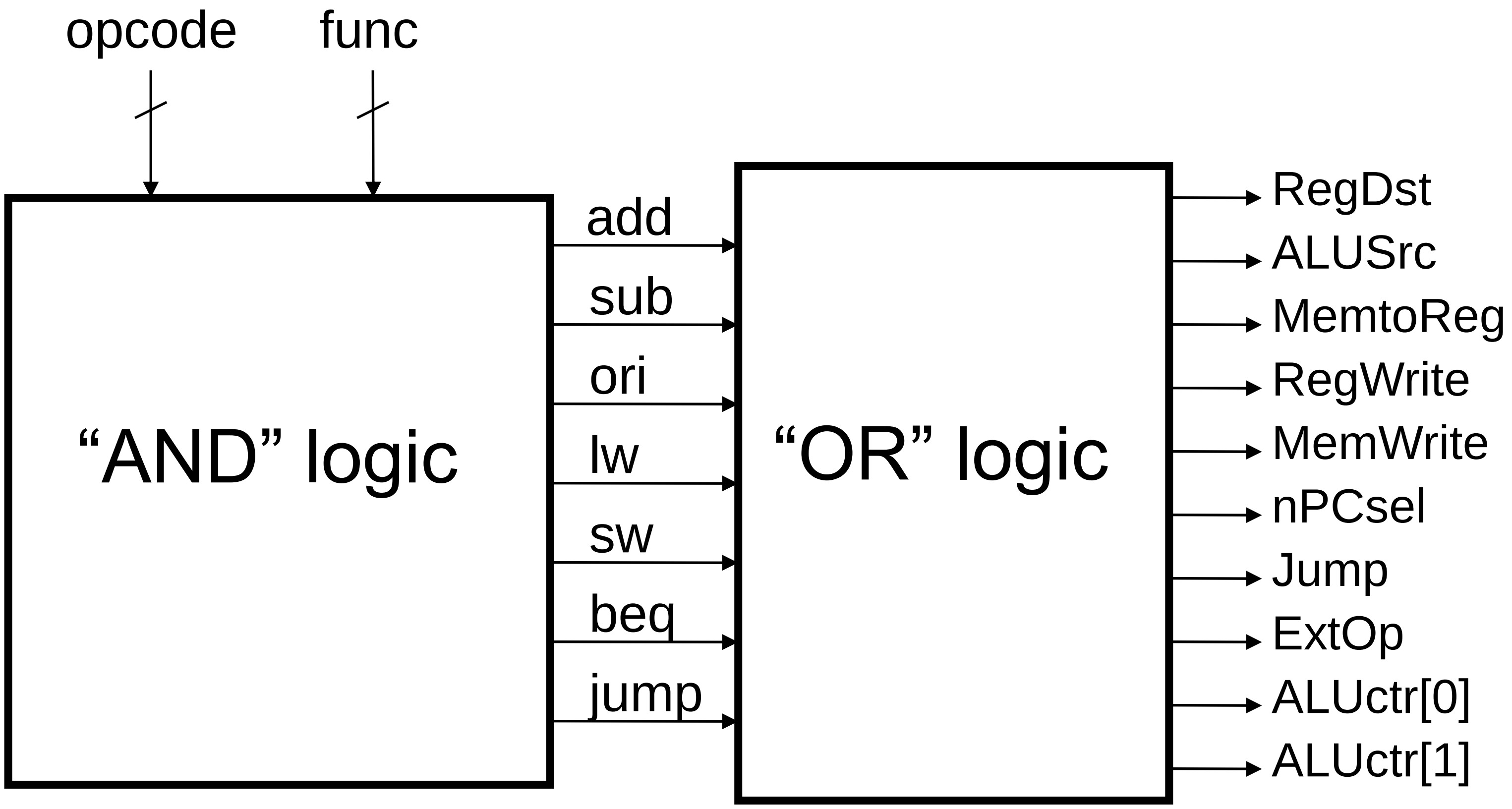
where,

rtype = $\sim op_5 \bullet \sim op_4 \bullet \sim op_3 \bullet \sim op_2 \bullet \sim op_1 \bullet \sim op_0$,
ori = $\sim op_5 \bullet \sim op_4 \bullet op_3 \bullet op_2 \bullet \sim op_1 \bullet op_0$
lw = $op_5 \bullet \sim op_4 \bullet \sim op_3 \bullet \sim op_2 \bullet op_1 \bullet op_0$
sw = $op_5 \bullet \sim op_4 \bullet op_3 \bullet \sim op_2 \bullet op_1 \bullet op_0$
beq = $\sim op_5 \bullet \sim op_4 \bullet \sim op_3 \bullet op_2 \bullet \sim op_1 \bullet \sim op_0$
jump = $\sim op_5 \bullet \sim op_4 \bullet \sim op_3 \bullet \sim op_2 \bullet op_1 \bullet \sim op_0$

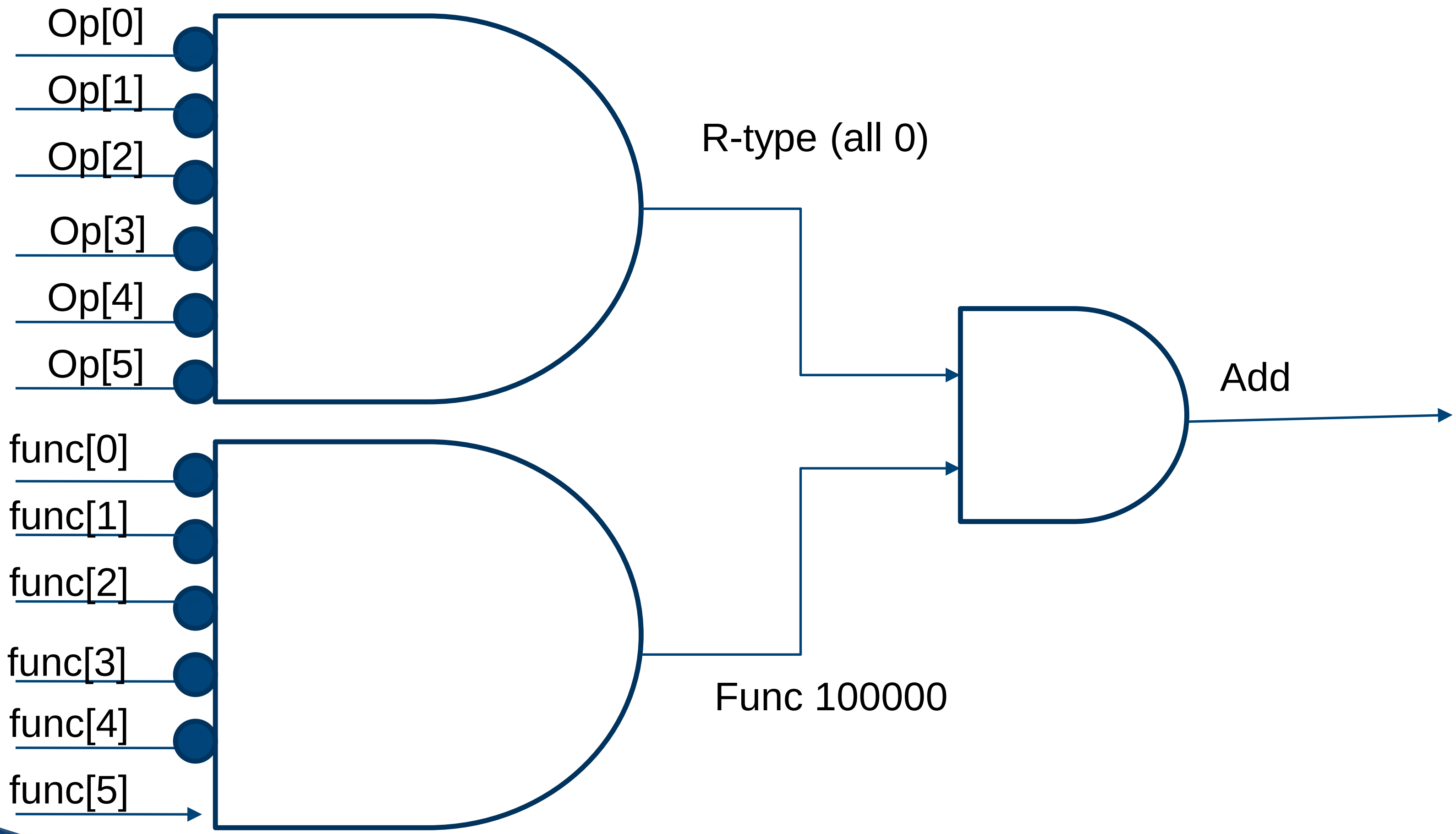
add = rtype $\bullet func_5 \bullet \sim func_4 \bullet \sim func_3 \bullet \sim func_2 \bullet \sim func_1 \bullet \sim func_0$
sub = rtype $\bullet func_5 \bullet \sim func_4 \bullet \sim func_3 \bullet \sim func_2 \bullet func_1 \bullet \sim func_0$

How do we implement this in gates?

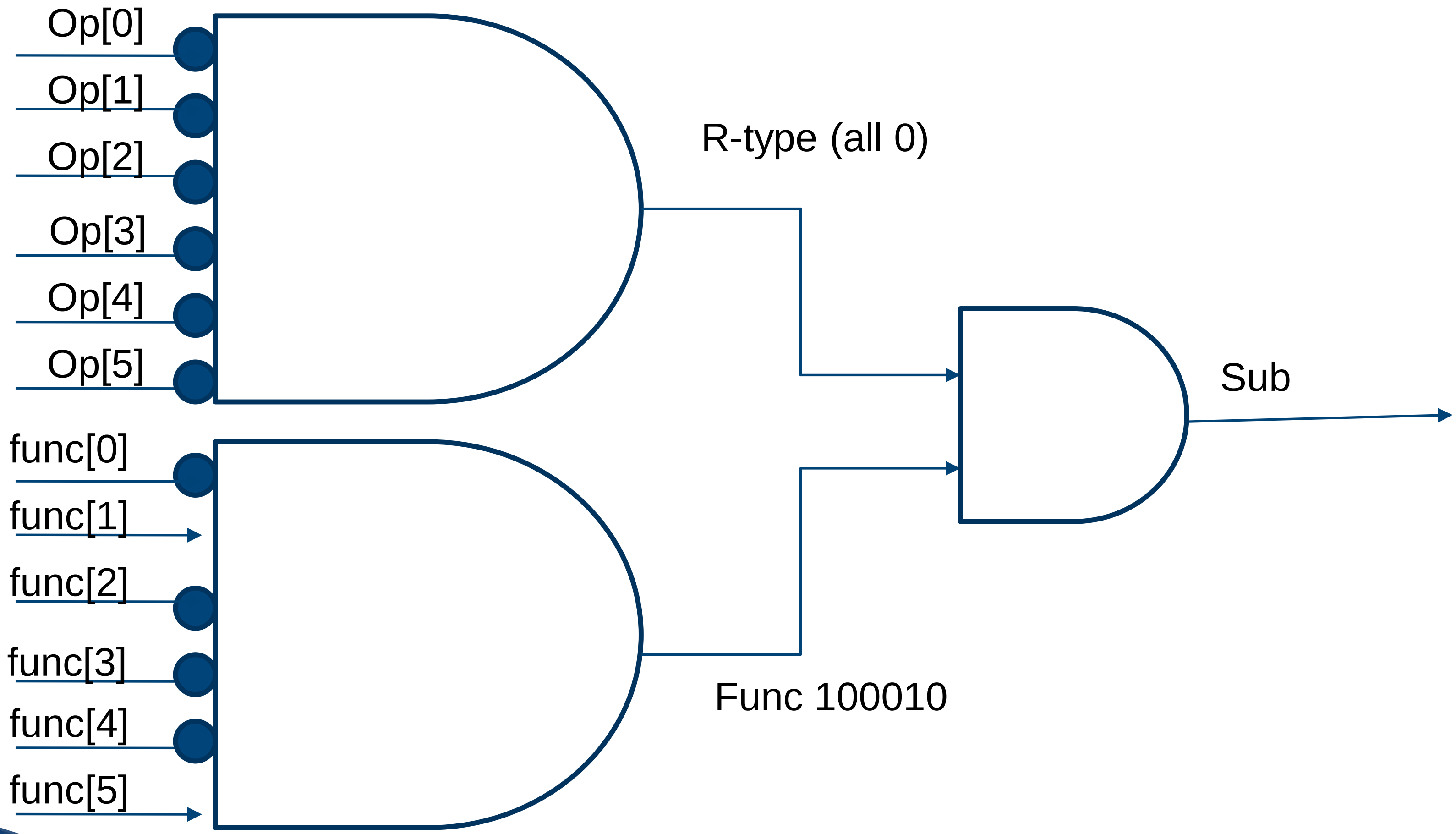
Controller Implementation



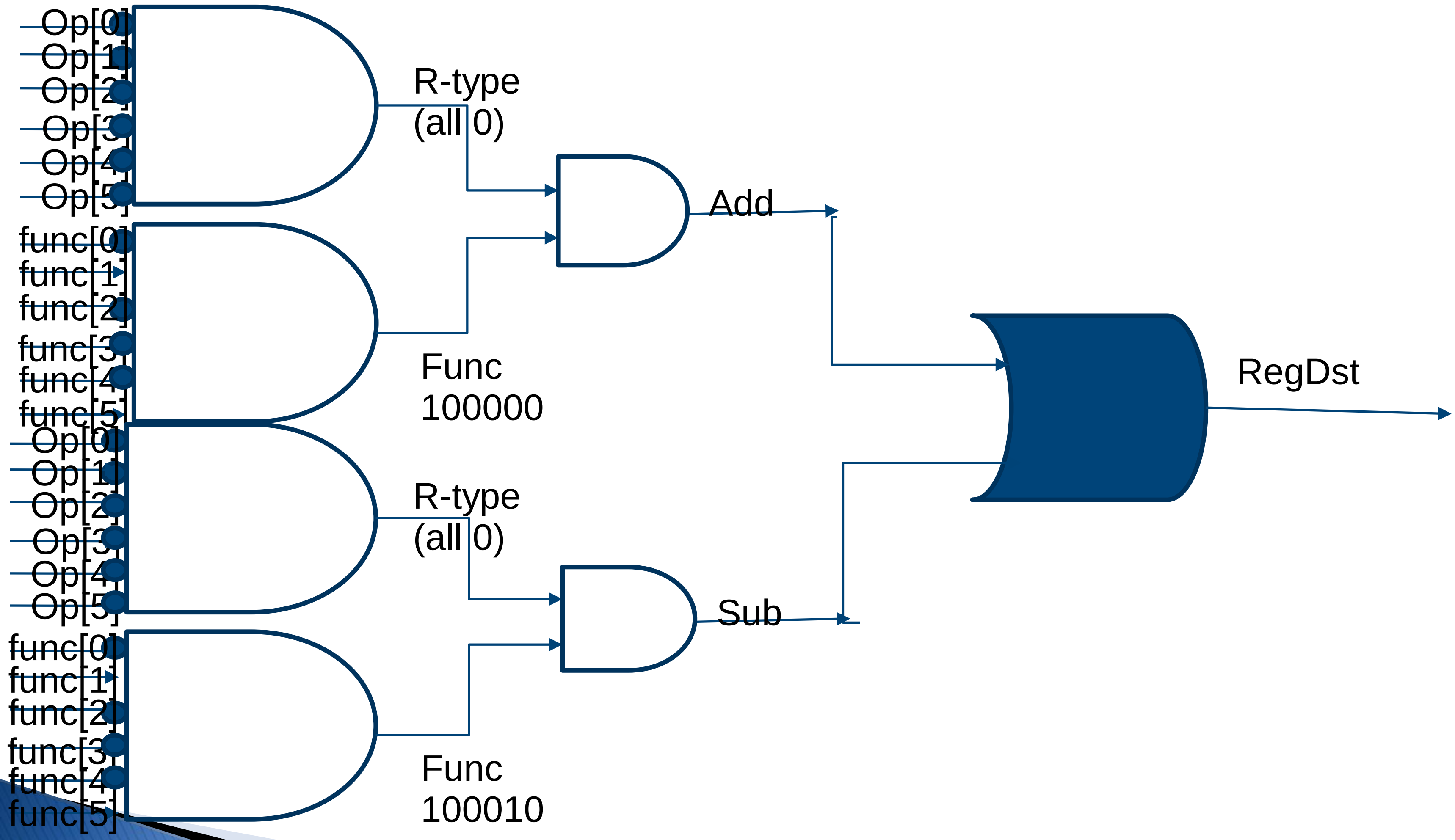
Add


Op[0]
Op[1]
Op[2]
Op[3]
Op[4]
Op[5]
func[0]
func[1]
func[2]
func[3]
func[4]
func[5]
R-type (all 0)
Func 100000
Add

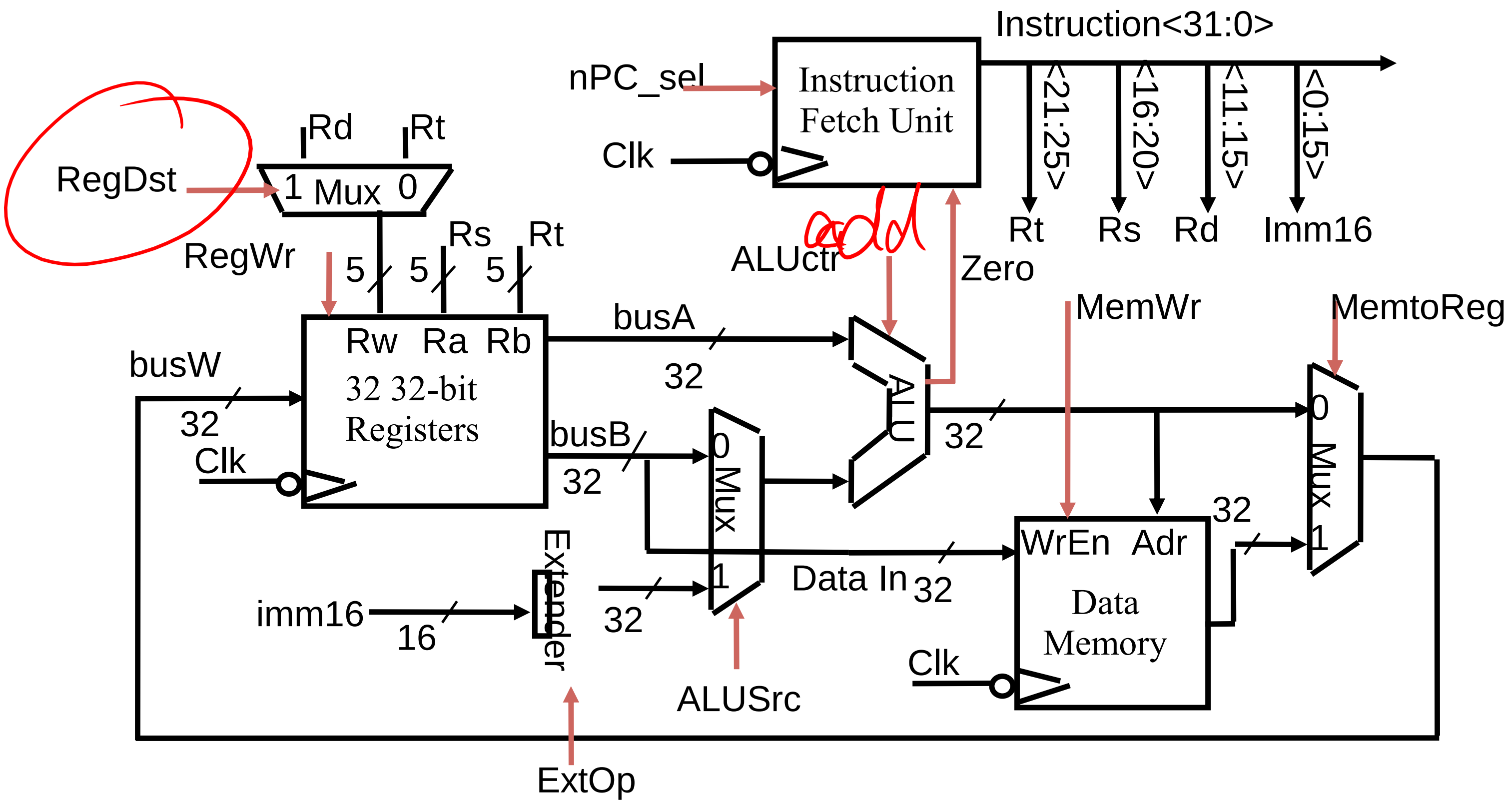
Sub


Op[0]
Op[1]
Op[2]
Op[3]
Op[4]
Op[5]
func[0]
func[1]
func[2]
func[3]
func[4]
func[5]
R-type (all 0)
Func 100010
Sub

RegDst


Op[0]
Op[1]
Op[2]
Op[3]
Op[4]
Op[5]
func[0]
func[1]
func[2]
func[3]
func[4]
func[5]
R-type
(all 0)
Func
100000
Add
Op[0]
Op[1]
Op[2]
Op[3]
Op[4]
Op[5]
func[0]
func[1]
func[2]
func[3]
func[4]
func[5]
R-type
(all 0)
Func
100010
Sub
RegDst

Quiz



1) MemToReg=‘x’ & ALUctr=‘sub’.
SUB or BEQ?

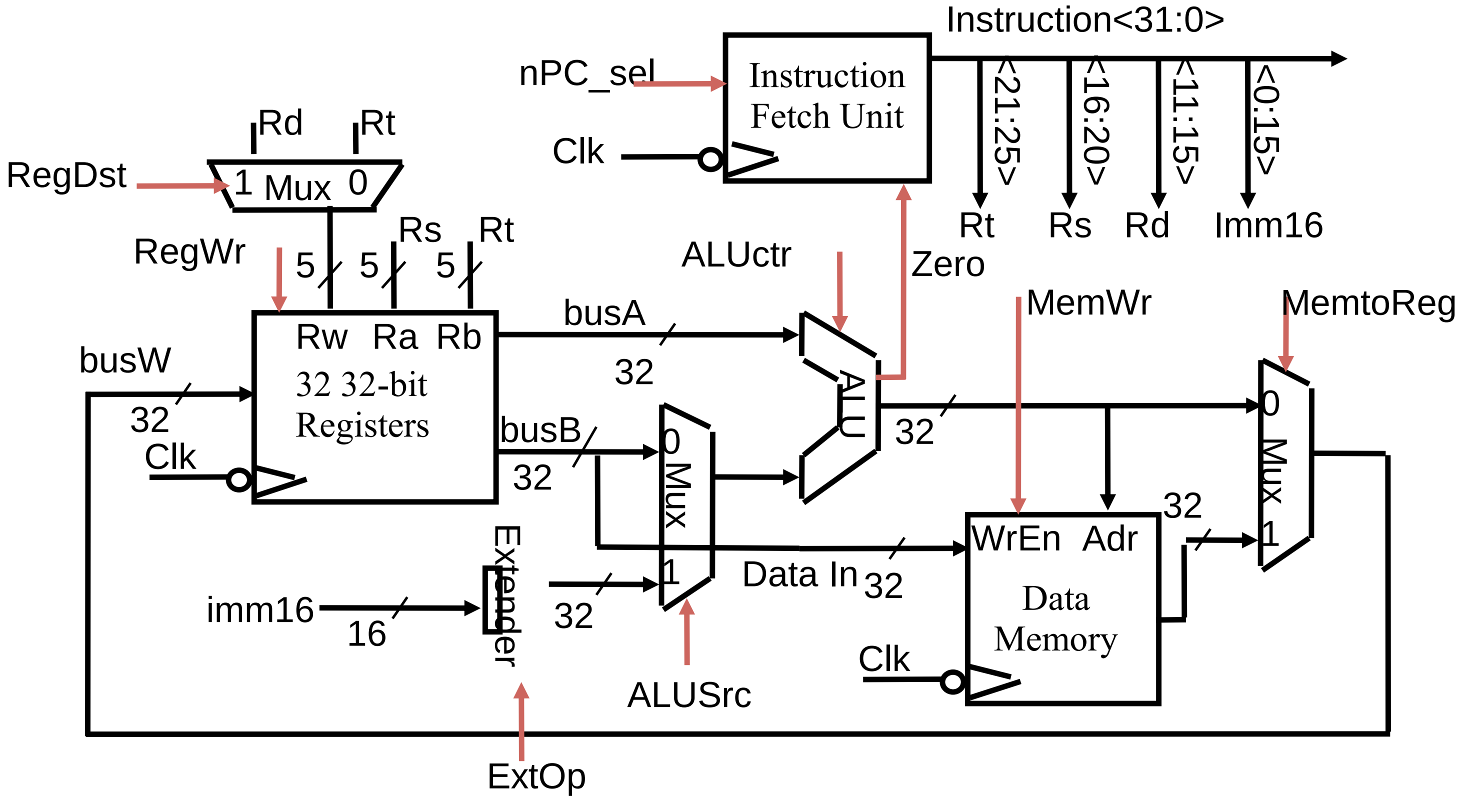
2) ALUctr=‘add’. Which 1 signal is different for all 3 of:
ADD, LW, & SW? RegDst or ExtOp?

1 0 X

X S S

	1	2
a)	S	R
b)	S	E
c)	B	R
d)	B	E

Quiz



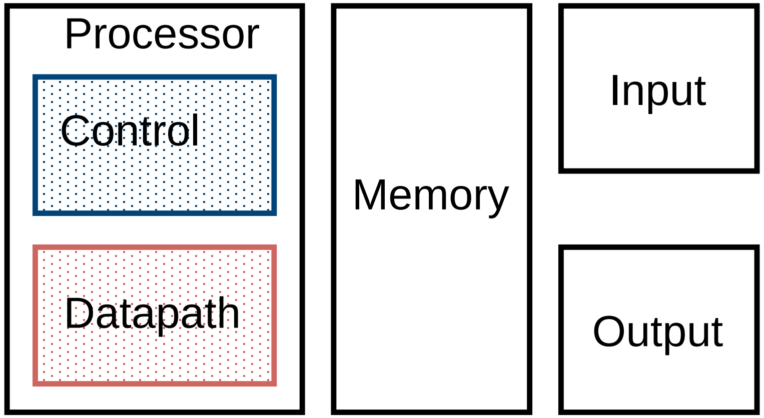
1) MemToReg='x' & ALUctr='sub'.
SUB or BEQ?

2) ALUctr='add'. Which 1 signal is different for all 3 of: ADD, LW, & SW? RegDst or ExtOp?

	12
a)	SR
b)	SE
c)	BR
d)	BE

Summary: Single-cycle Processor

- 5 steps to design a processor
  - **1. Analyze instruction set → datapath requirements**
  - **2. Select set of datapath components & establish clock methodology**
  - **3. Assemble datapath meeting the requirements**
  - **4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.**
  - **5. Assemble the control logic**
    - **Formulate Logic Equations**
    - **Design Circuits**


Processor
Control
Datapath
Memory
Input
Output