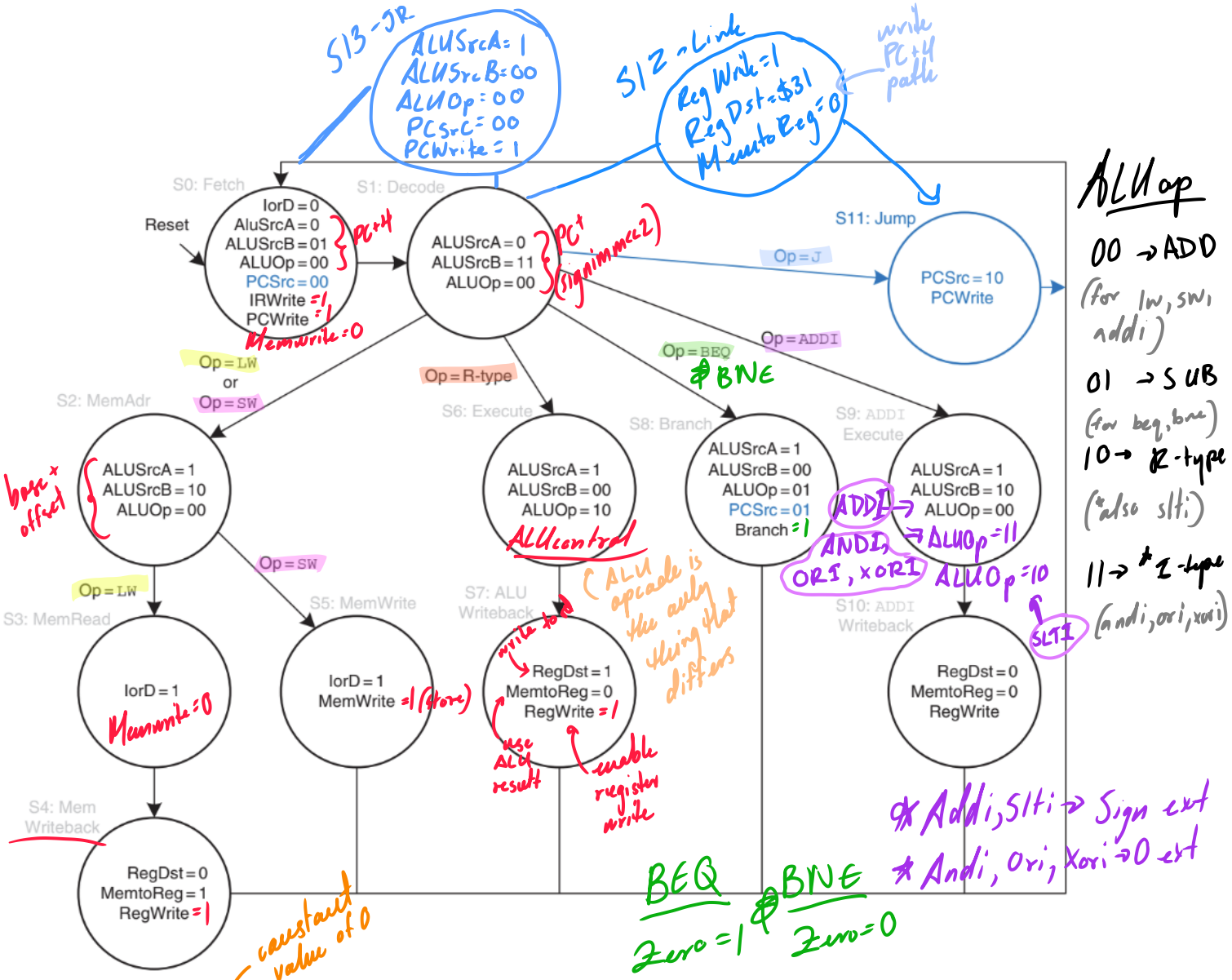

Figure 7.42 Main controller state for j

* nop = sll $0, $0, 0
opcode = 000000
↳ sll where shamt = 0
* uses sll ALUcontrol
For FSM, treat as "R-type"

S13 - JR
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 00
PCSrc = 00
PCWrite = 1

* A as ALU Inputs, ALUOp = add, Select ALU output, Write PC

Jr → S0 → S1 → S13 → S0

Jal → S0 → S1 → S12 → S11 → S0

Jump (j) → S0 → S1 → S11 → S0

R-types → S0 → S1 → S6 → S7 → S0

I-types → S0 → S1 → S9 → S10 → S0

BNE & BEQ → S0 → S1 → S8 → S0

Jal → Jump & link (in register 31) saves next PC, & jumps to address

Jr → Jump to address in register

- add R
- addi I
- sub R
- and R
- andi I
- or R
- ori I
- xor R
- xori I
- nor R
- sll R
- sra R
- srl R
- slt R
- slti I
- beq } Branches
- bne }
- j J
- jal J
- jr J
- lw
- sw
- nop

Noop (32'h0)

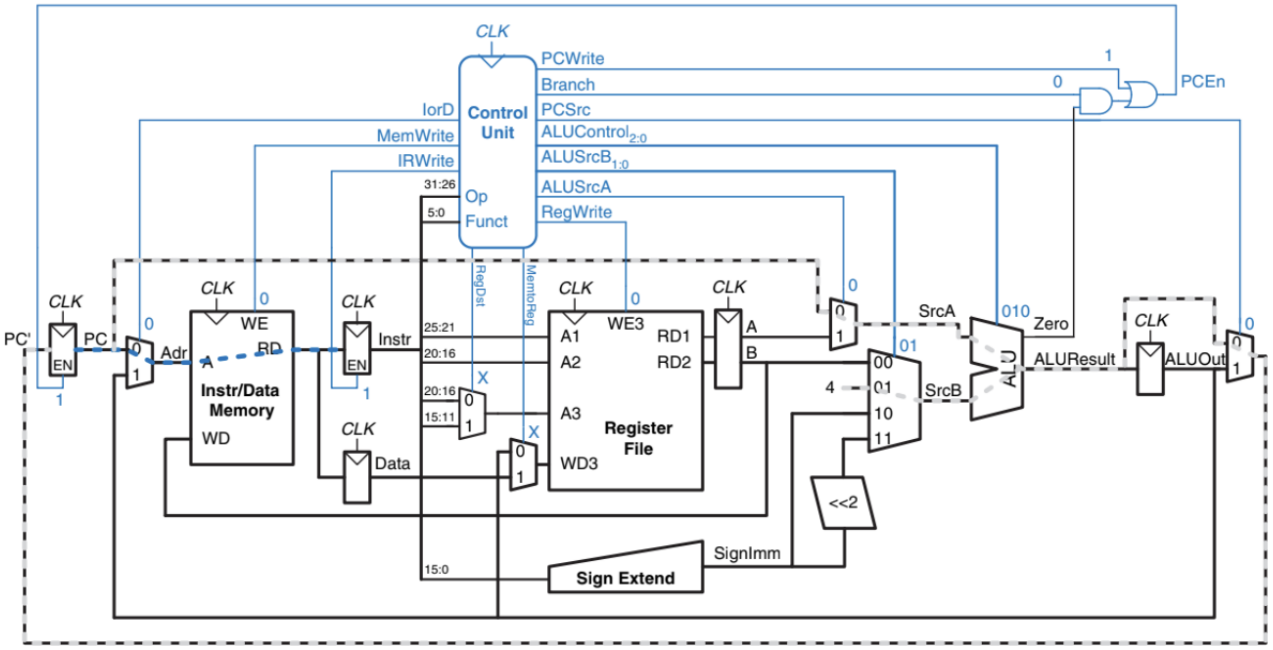

Figure 7.30 Data flow during the fetch step