Supplemental Reading

- Please finish the reading of the second document by Sept. 25.

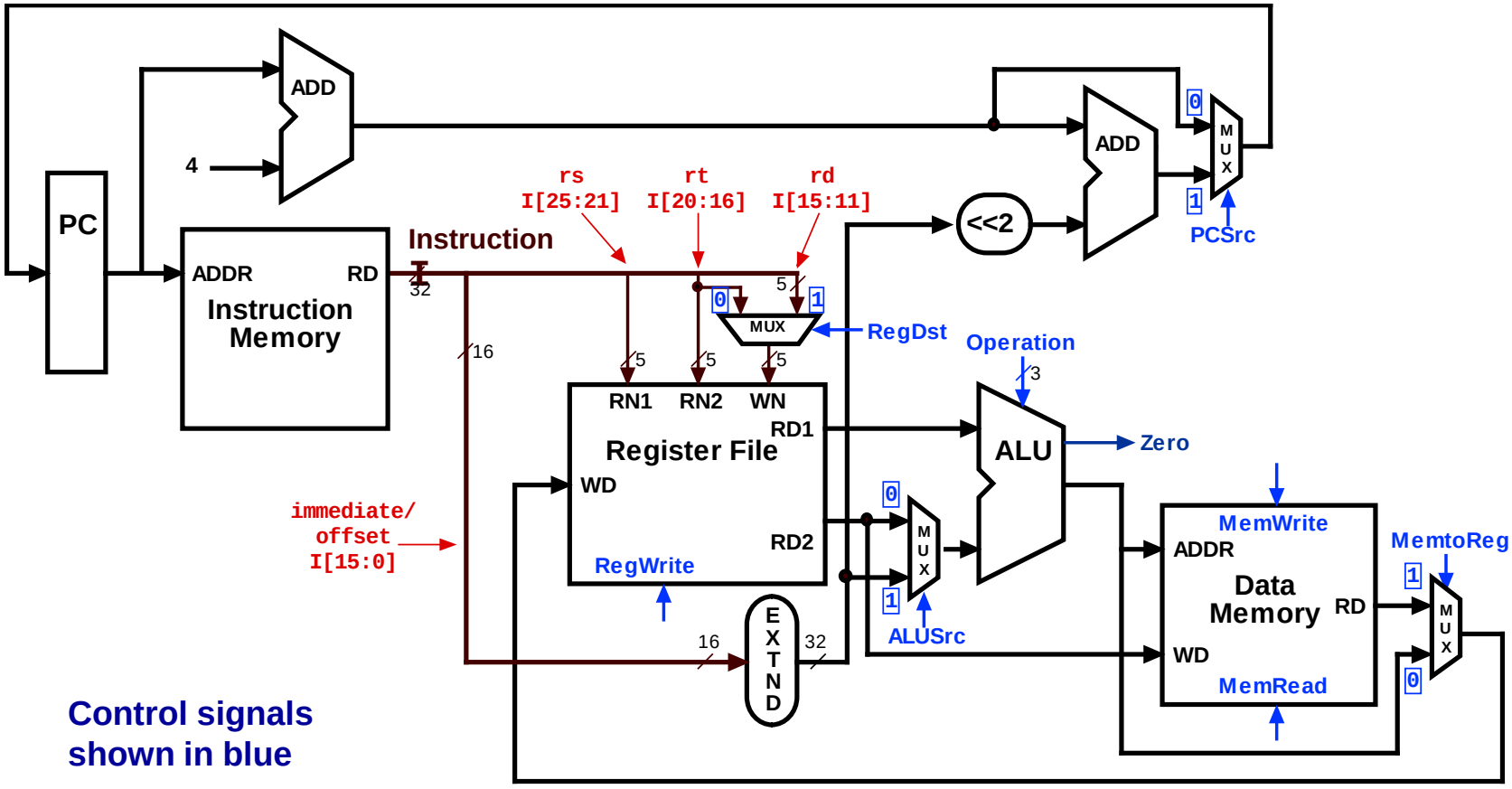
- The grader has emailed you all 8 reading documents in one zipped file.

CS572 Micro Architecture

Processor: Controller

These slides are adapted from notes by Dr. Dave Patterson (UCB)

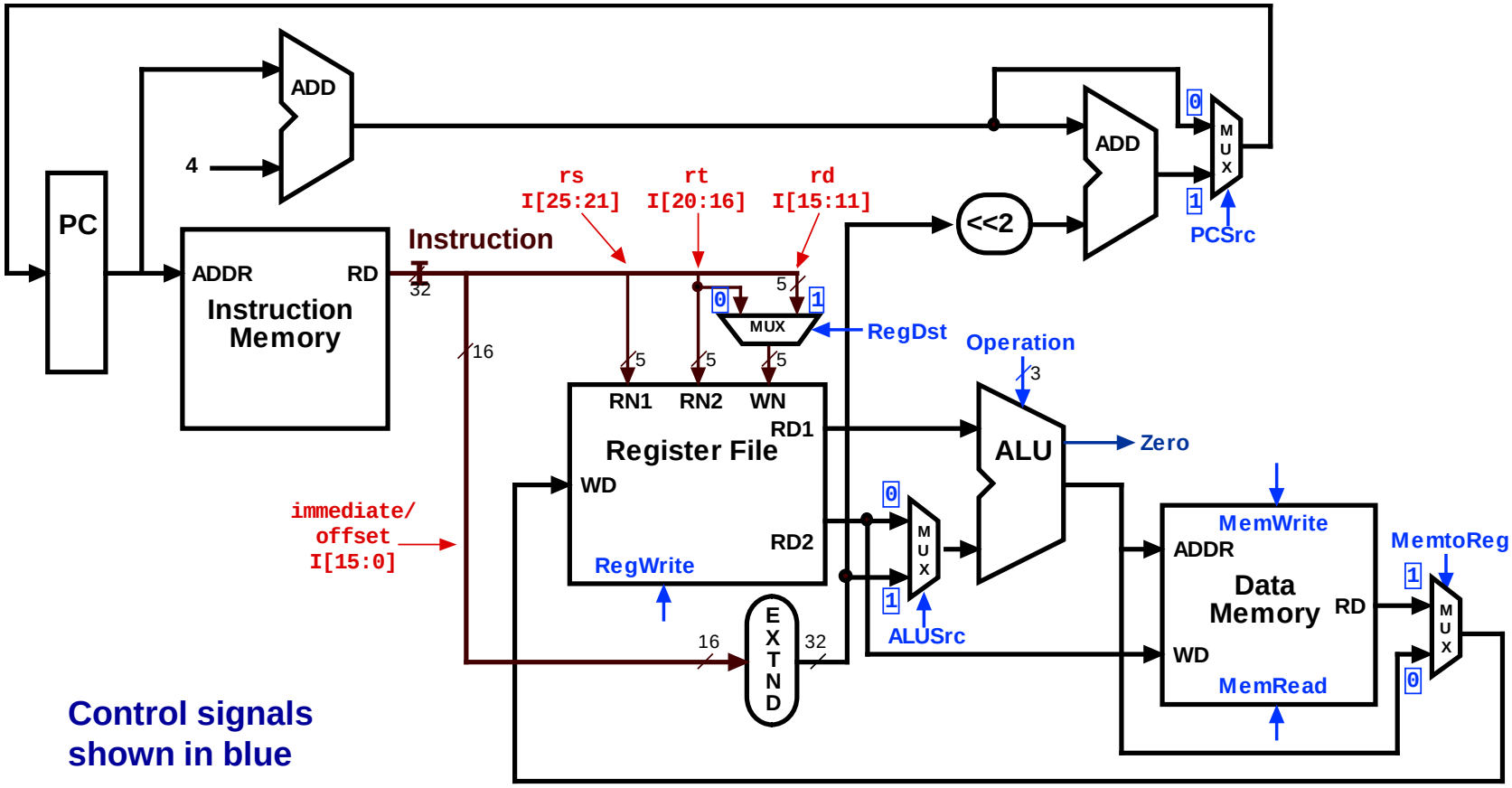
Complete Single-Cycle Datapath

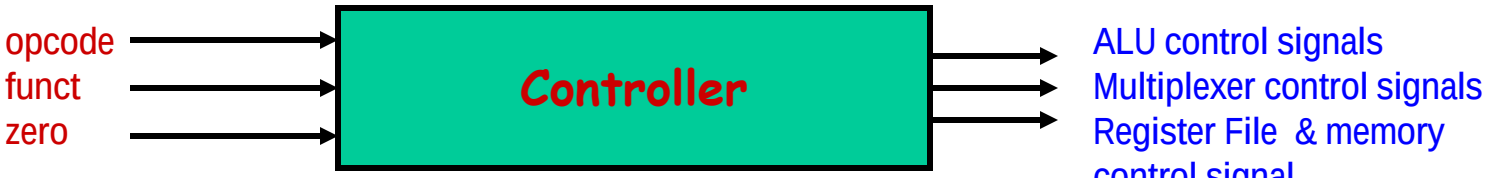


Control signals shown in blue

ALU control signals
Multiplexer control signals
Register File & memory control signal

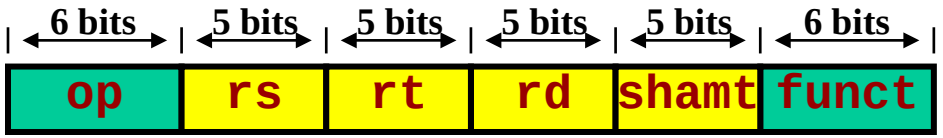
Complete Single-Cycle Datapath





Control Unit Design

- Desired function:
  - Given an instruction word….
  - Generate control signals needed to execute instruction
- Implemented as a *combinational* logic function:
  - Inputs



R-Format

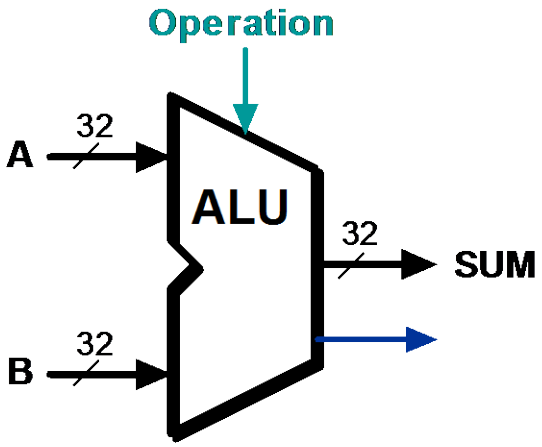
    - Instruction word - op and funct fields
    - ALU status output - Zero
  - Outputs - processor control points
    - ALU control signals
    - Multiplexer control signals
    - Register File & memory control signal

Determining Control Points

- For each instruction type, determine proper value for each control point (control signal)
  - 0
  - 1
  - X (don't care - either 1 or 0)
- Ultimately … use these values to build a truth table

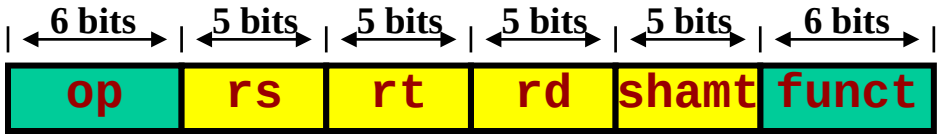
ALU Control Signals

- Functions:

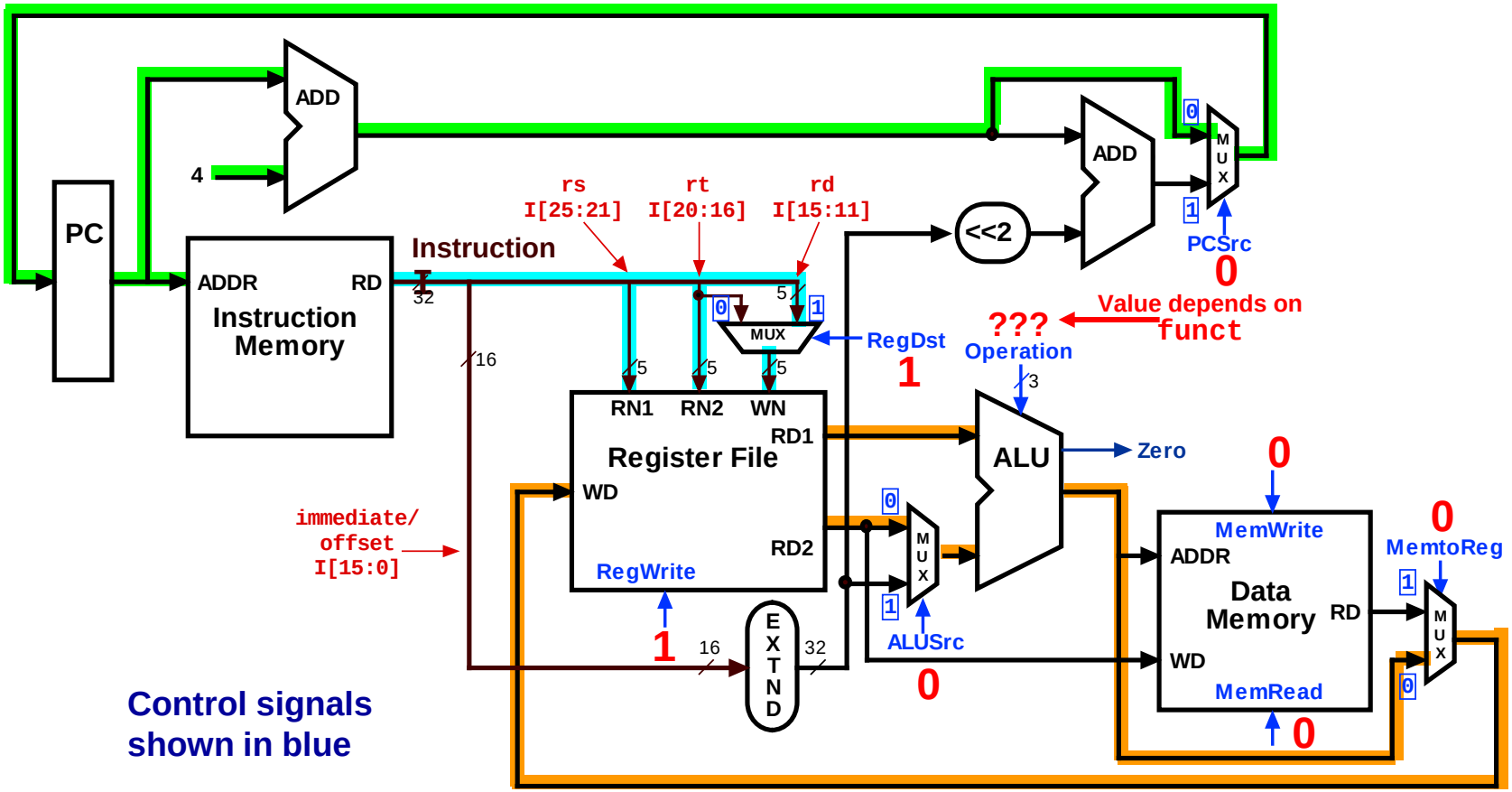


ALU control input	Function
000	AND
001	OR
010	add
110	subtract
111	set on less than

Control Signals: R-Type Instruction

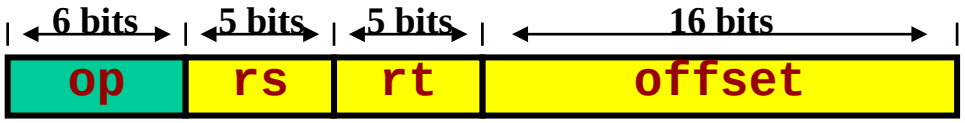


R-Format

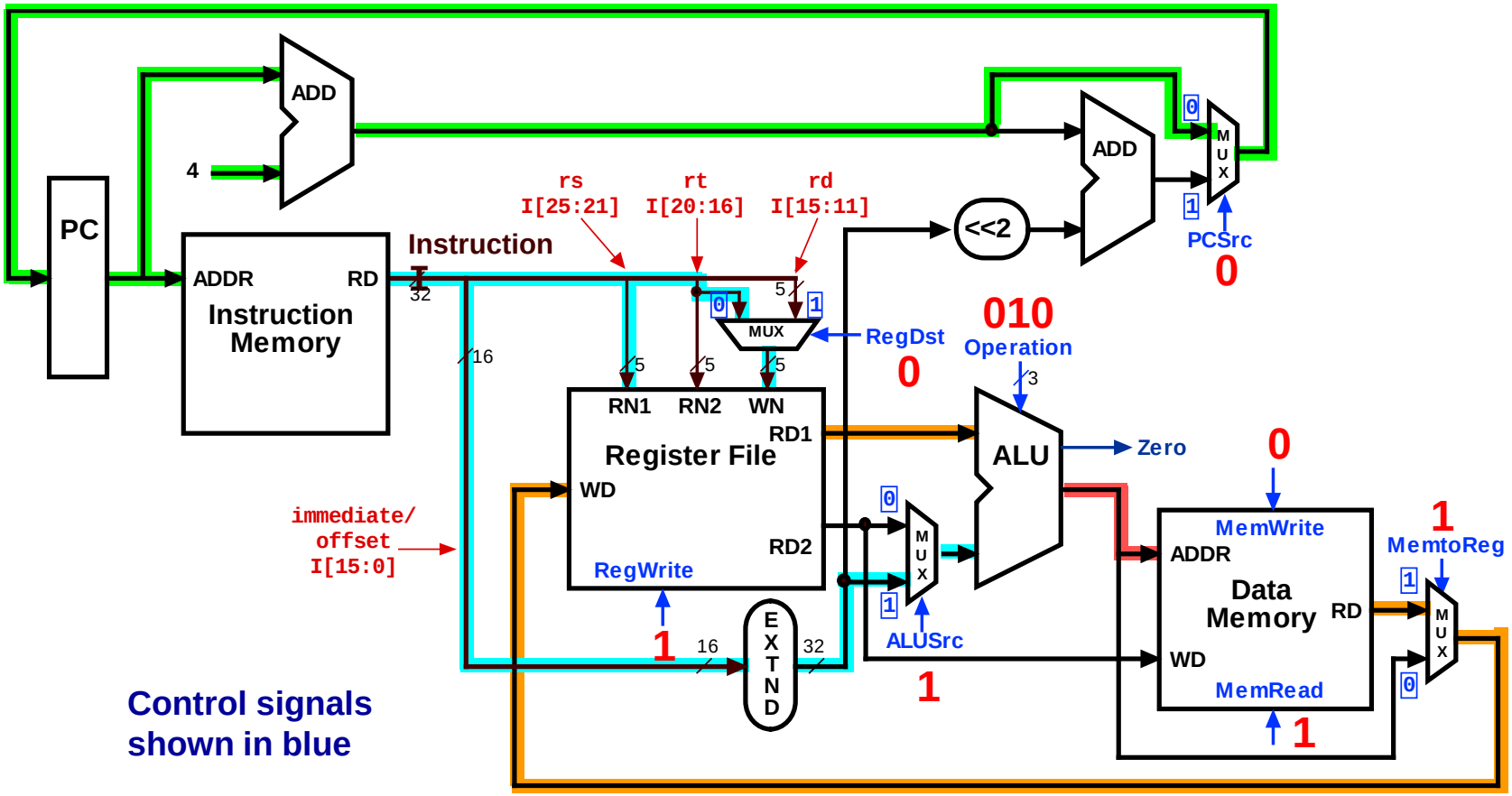


Control signals shown in blue

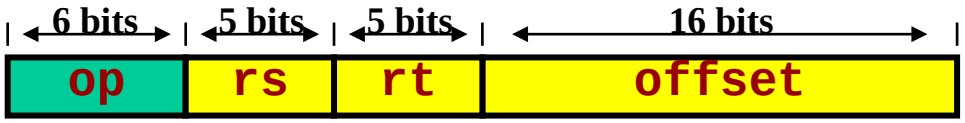
Control Signals: lw Instruction



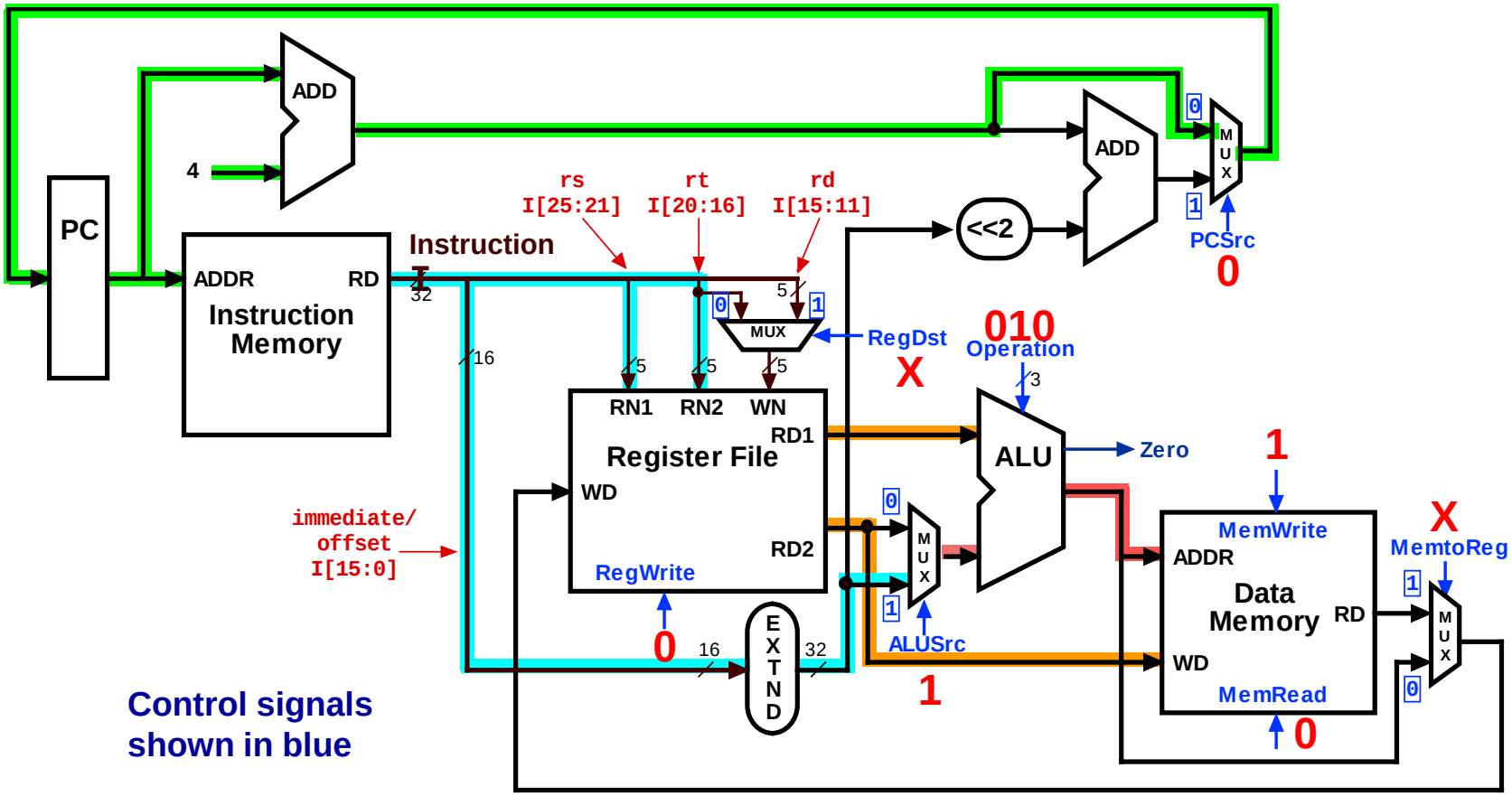
I-Format



Control Signals: sw Instruction

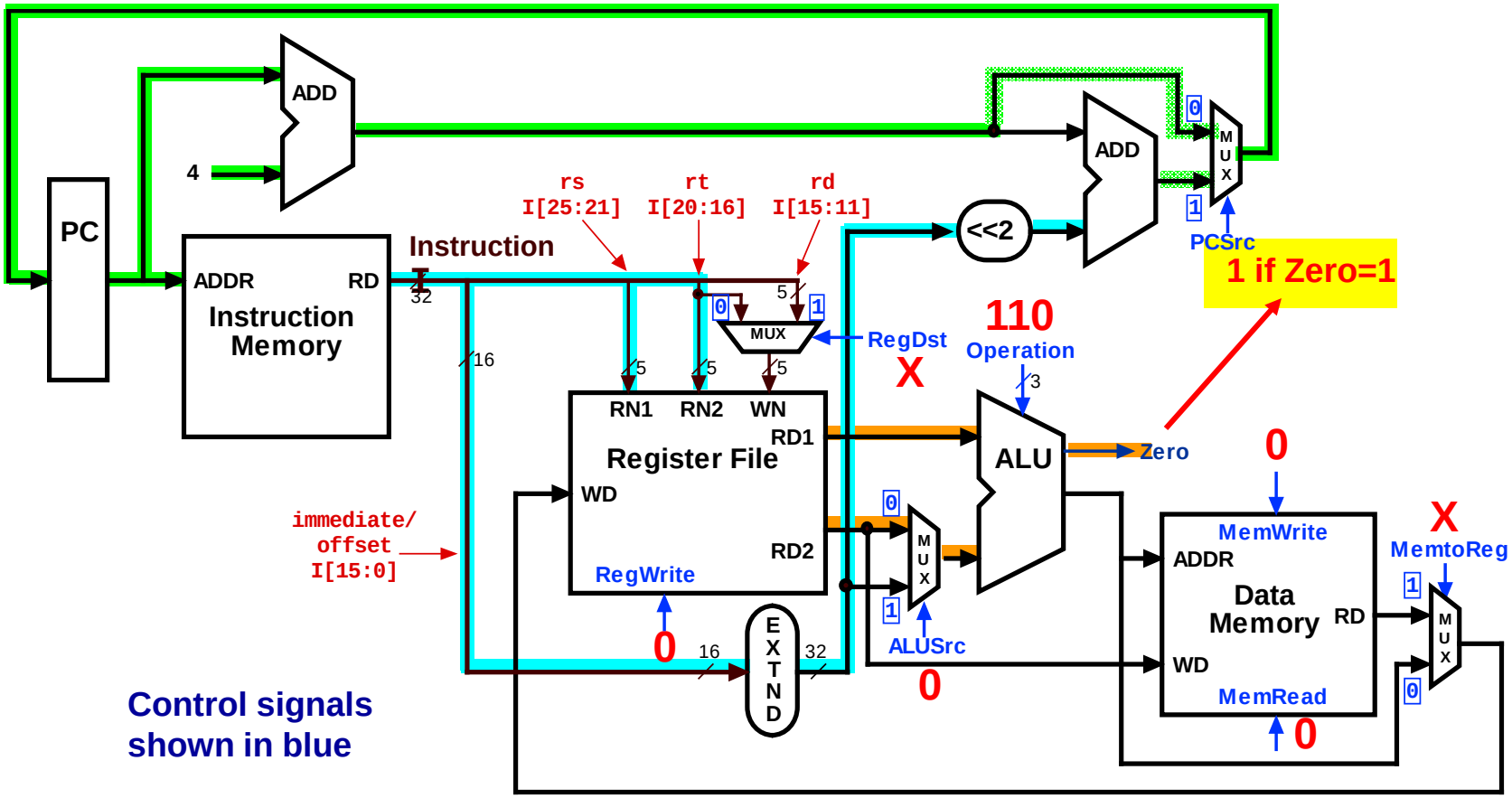


I-Format

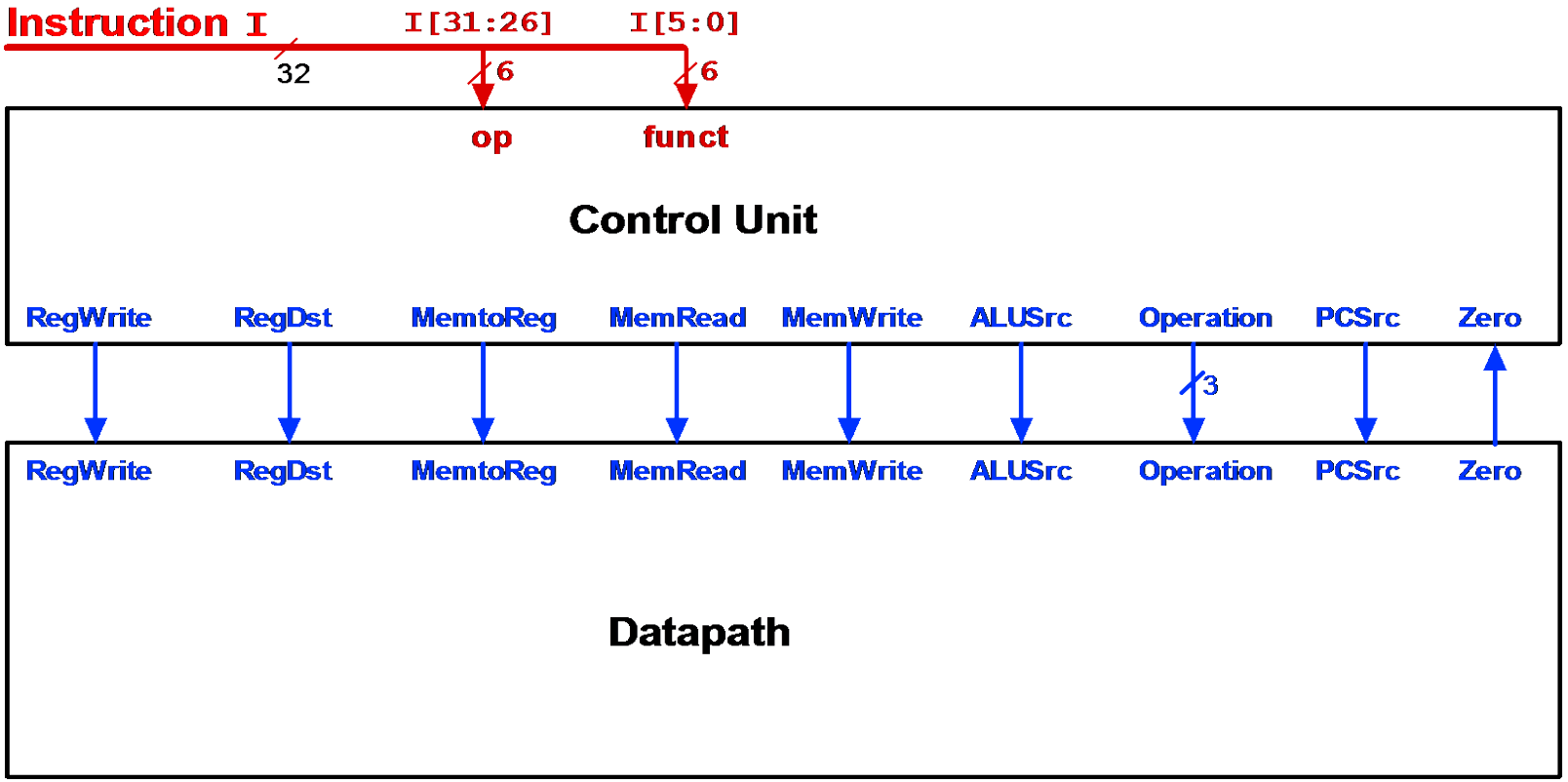


Control signals shown in blue

Control Signals: beq Instruction



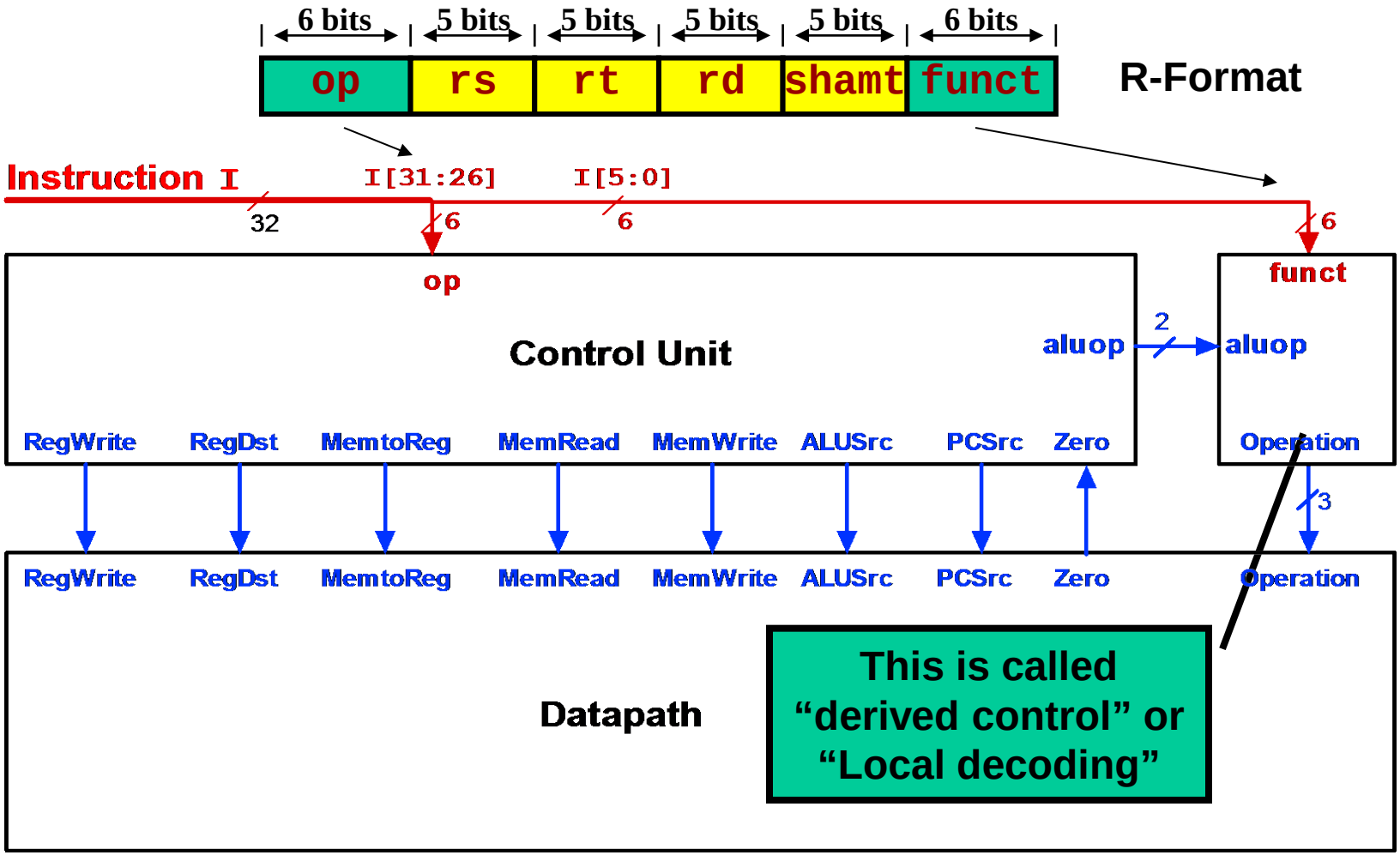
Control Unit Structure



Control Unit Structure (cont.)

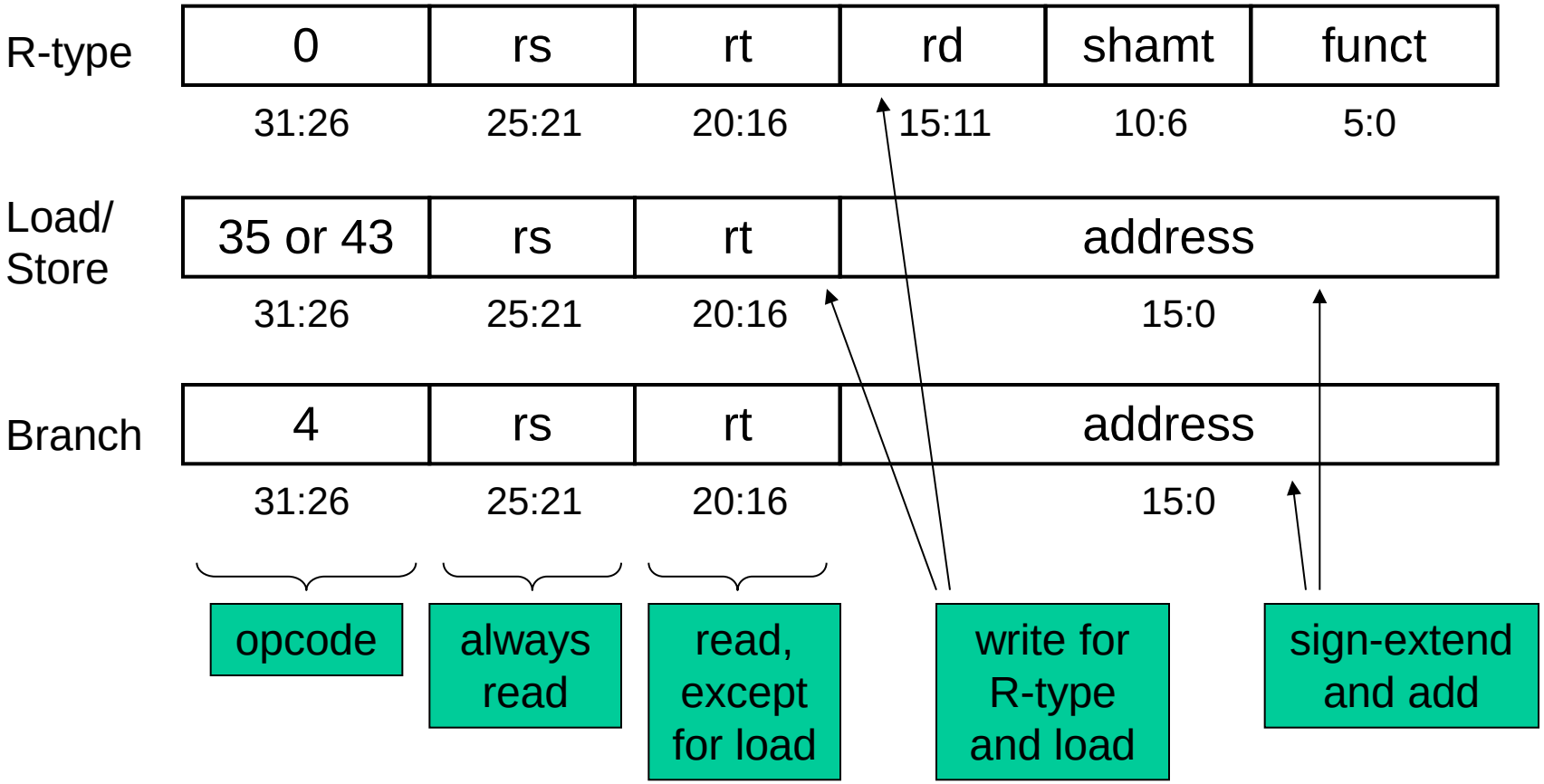
- Control unit as shown: one huge logic block
- Idea: decompose into smaller logic blocks
  - Smaller blocks can be faster
  - Smaller blocks are easier to work with
- Observation (rephrased):
  - The only control signal that depends on the `funct` field is the ALU Operation signal
  - Idea?: separate logic for ALU control

Modified Control Unit Structure

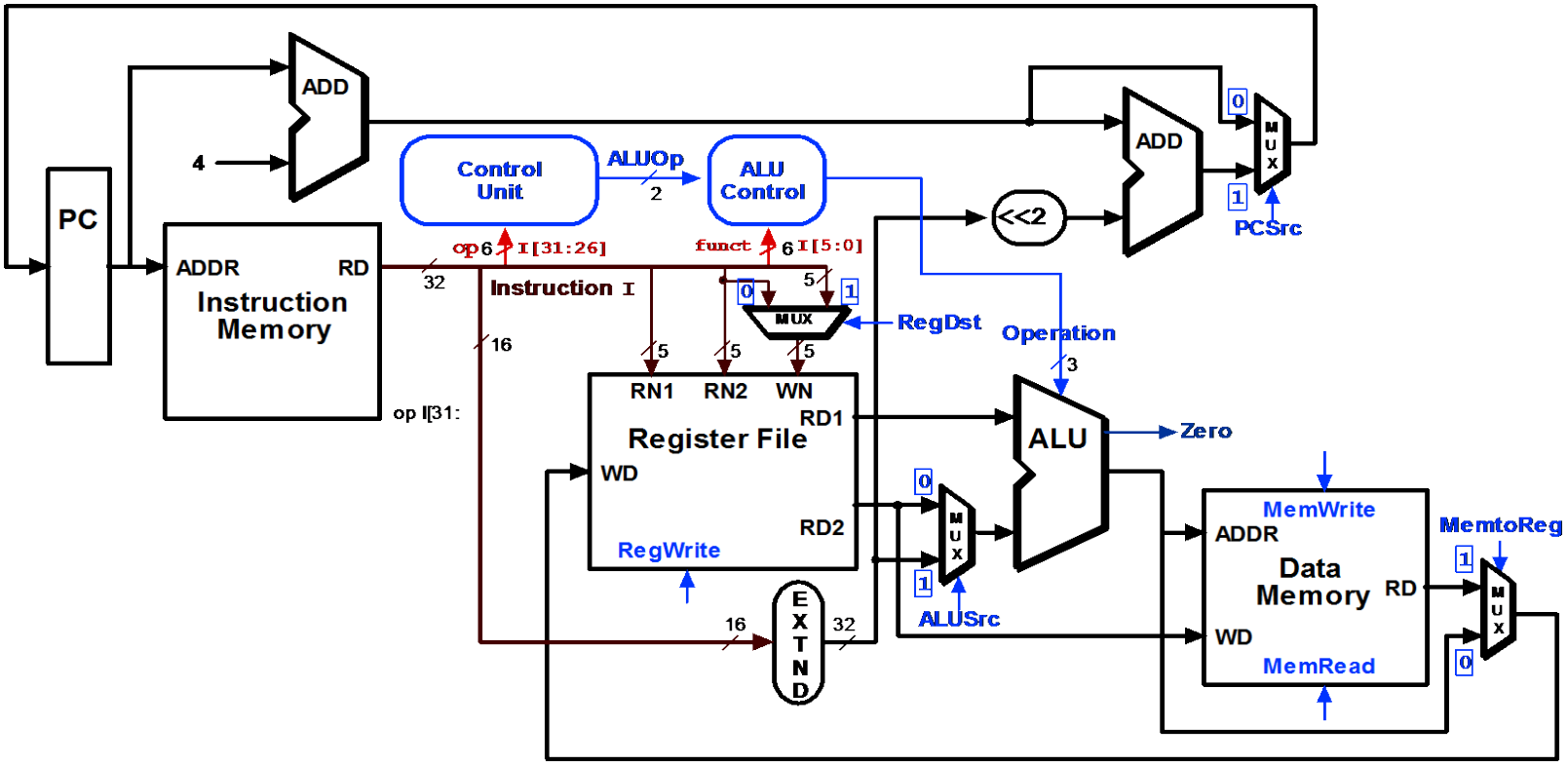


The Main Control Unit

- Control signals derived from instruction



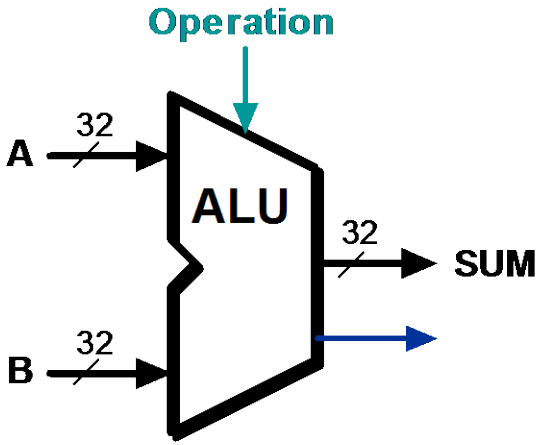
Datapath with Modified Control Unit



ALU Function

- Functions:

ALU control input	Function
000	AND
001	OR
010	add
110	subtract
111	set on less than



ALU Usage in Processor Design

- Usage depends on instruction type
  - Instruction type (specified by opcode)
  - funct field (r-type instructions only)
- Encode instruction type in ALUOp signal

XXXXXX means "don't care"

Instr. type	Operation	funct	Desired Action	ALU Ctl.	ALUOp
data transfer	lw	XXXXXX	add	010	00
data transfer	sw	XXXXXX	add	010	00
branch	beq	XXXXXX	subtract	110	01
r-type	add	100000	add	010	10
r-type	sub	100010	subtract	110	10
r-type	and	100100	and	000	10
r-type	or	100101	or	001	10
r-type	slt	101010	set on less than	111	10

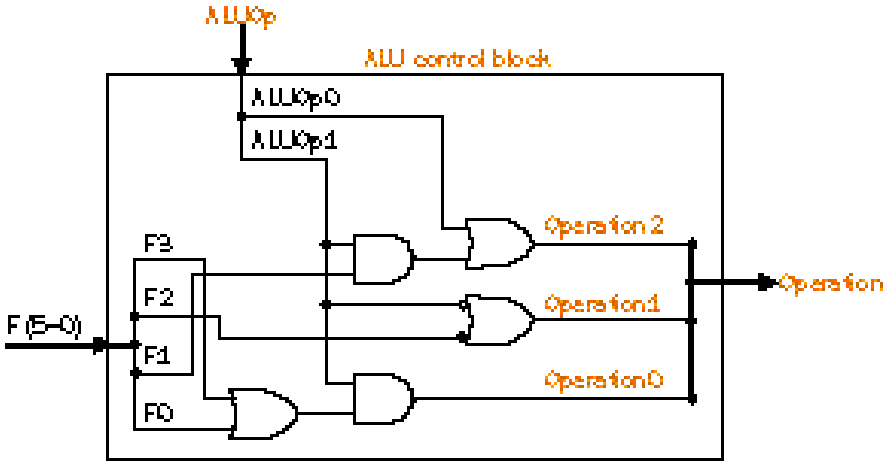
ALU Control: Truth Table

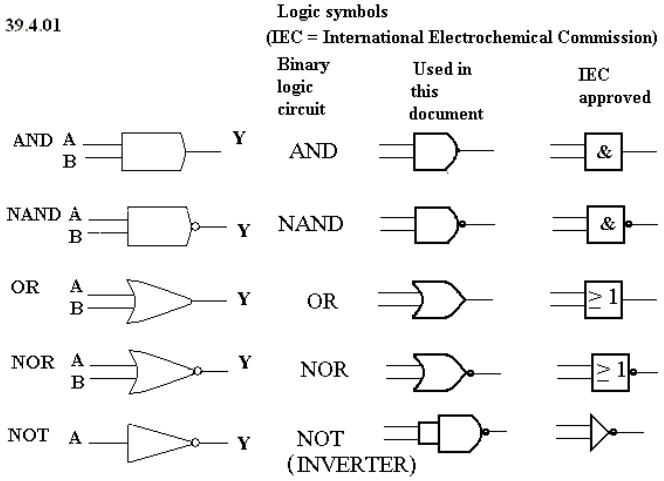
- Use don't care values to minimize length
  - Ignore F5, F4 (they are always "10")
  - Assume ALUOp never equals "11"

ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	Operation
0	0	X	X	X	X	X	X	010
X	1	X	X	X	X	X	X	110
1	X	X	X	0	0	0	0	010
1	X	X	X	0	0	1	0	110
1	X	X	X	0	1	0	0	000
1	X	X	X	0	1	0	1	001
1	X	X	X	1	0	1	0	111

Instr. type	Operation	funct	Desired Action	ALU Ctl.	ALUOp
data transfer	lw	XXXXXX	add	010	00
data transfer	sw	XXXXXX	add	010	00
branch	beq	XXXXXX	subtract	110	01
r-type	add	100000	add	010	10
r-type	sub	100010	subtract	110	10
r-type	and	100100	and	000	10
r-type	or	100101	or	001	10
r-type	slt	101010	set on less than	111	10

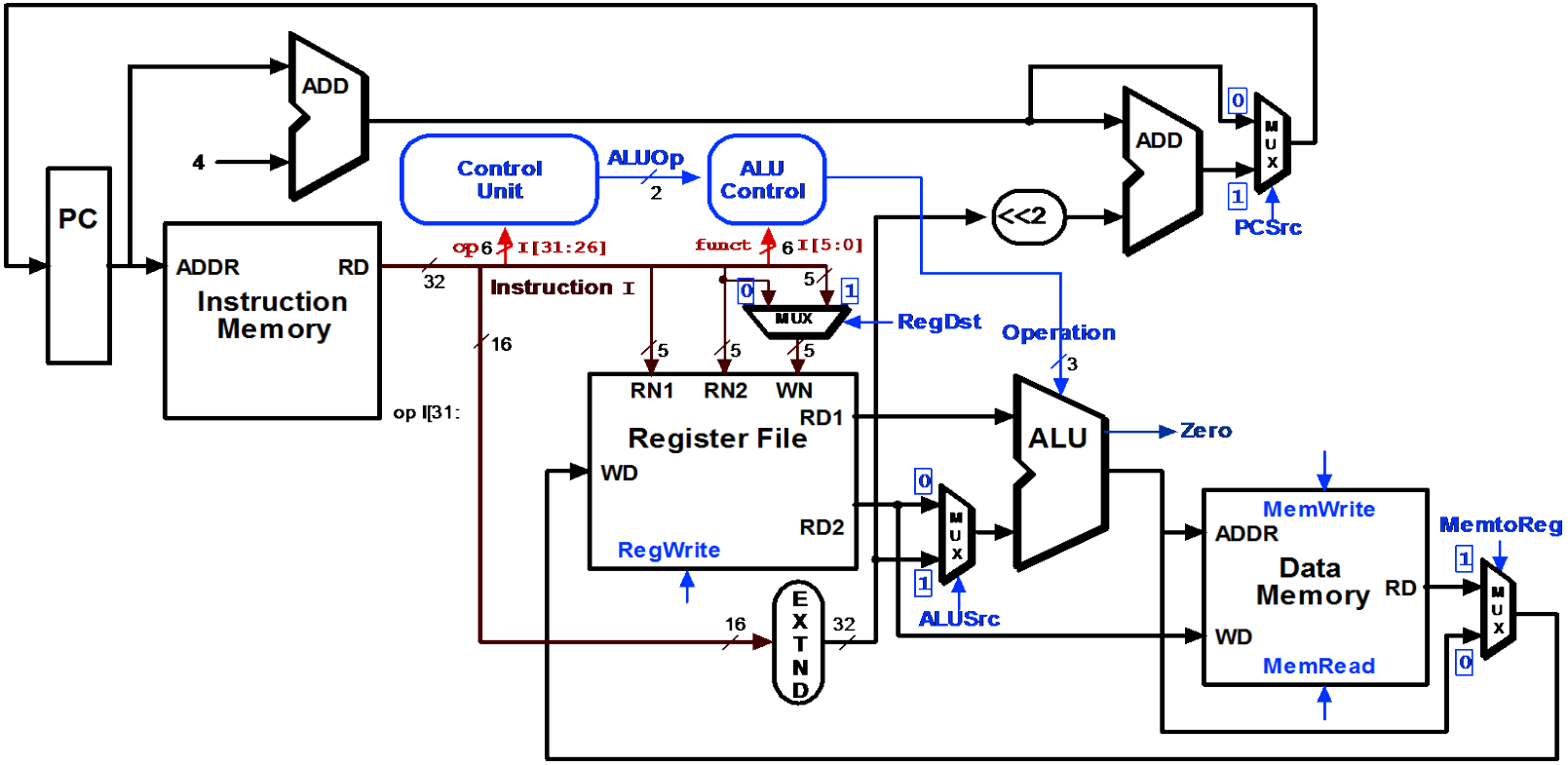
ALU Control - Implementation





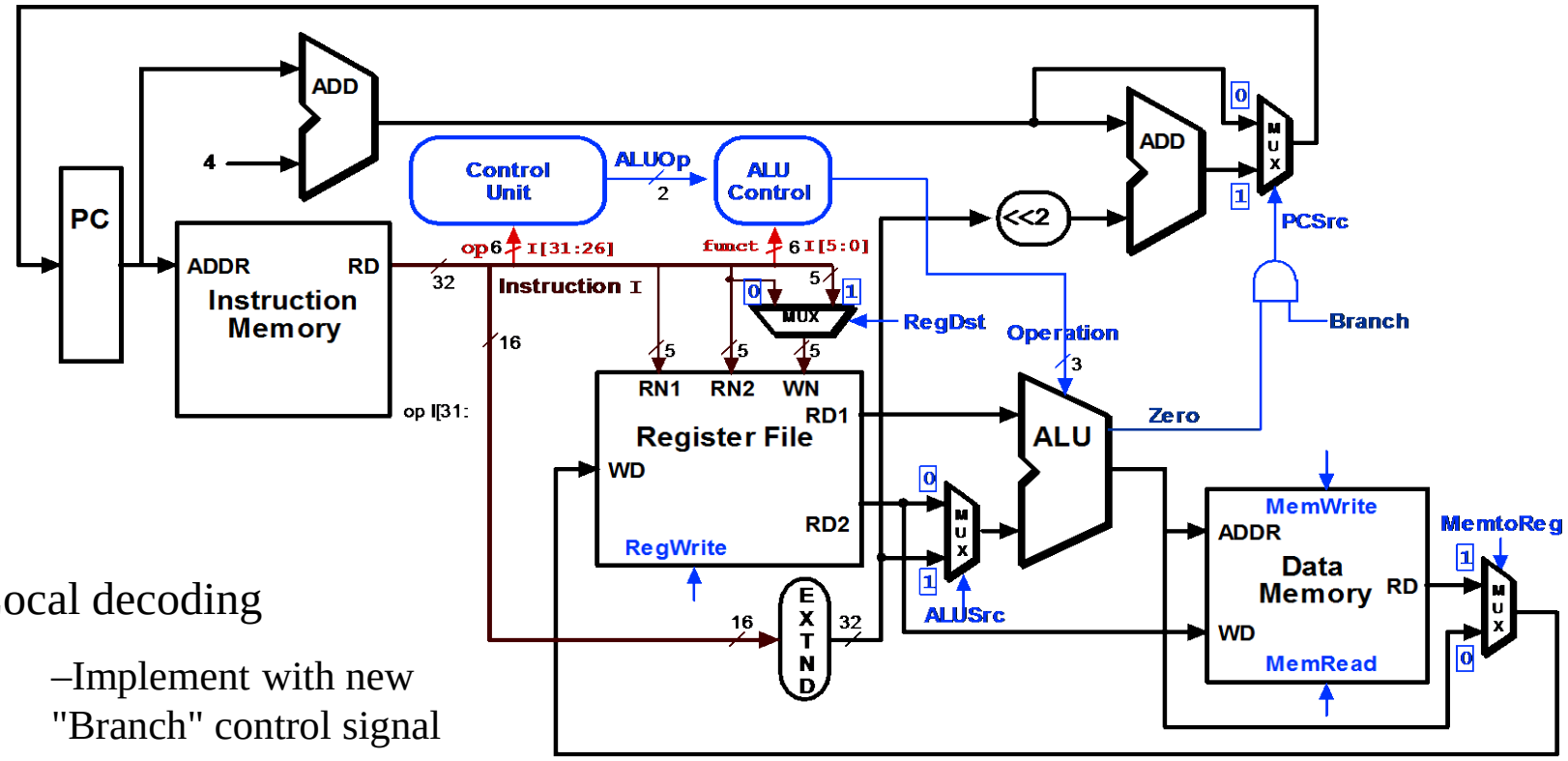
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	Operation
0	0	X	X	X	X	X	X	010
X	1	X	X	X	X	X	X	110
1	X	X	X	0	0	0	0	010
1	X	X	X	0	0	1	0	110
1	X	X	X	0	1	0	0	000
1	X	X	X	0	1	0	1	001
1	X	X	X	1	0	1	0	111

One More Modification - for Branch



- BEQ instruction depends on Zero output of ALU
- No other instruction uses Zero output

Processor Design: Branch Modification



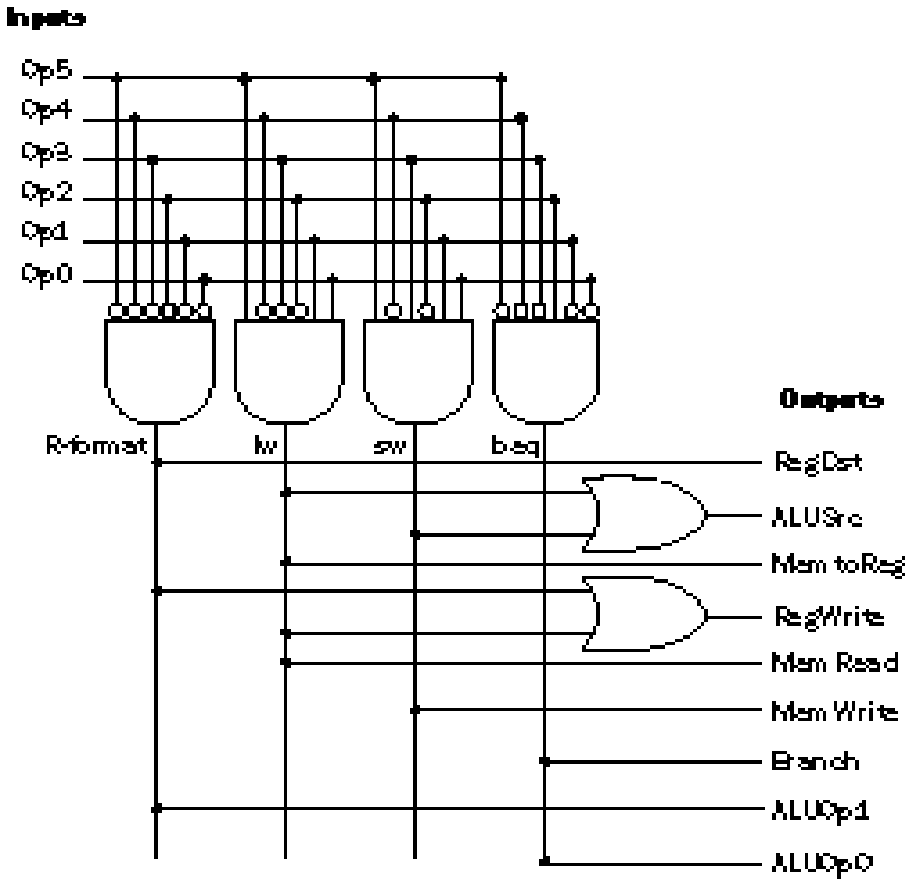
- Local decoding
  - –Implement with new "Branch" control signal
  - –Add AND gate to generate PCSrc

Control Unit Implementation

- Review: Opcodes for key instructions
- Control Unit Truth Table: Fill in the blanks
- Implementation: Decoder + 2 Gates

Input							Output								
OP	Op5	Op4	Op3	Op2	Op1	Op0	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
RT	0	0	0	0	0	0									
lw	1	0	0	0	1	1									
sw	1	0	1	0	1	1									
beq	0	0	0	1	0	0									

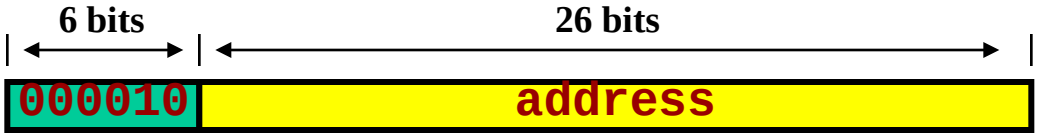
Control Unit Implementation



Source:
Tod Amon's COD2e Slides

Final Extension: Implementing `j` (jump)

- Instruction Format



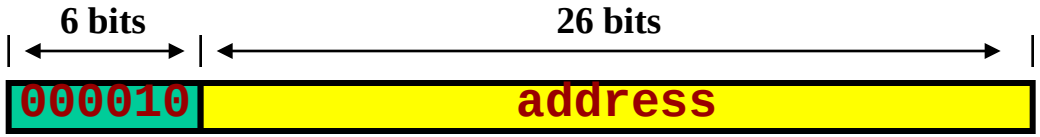
J-Format

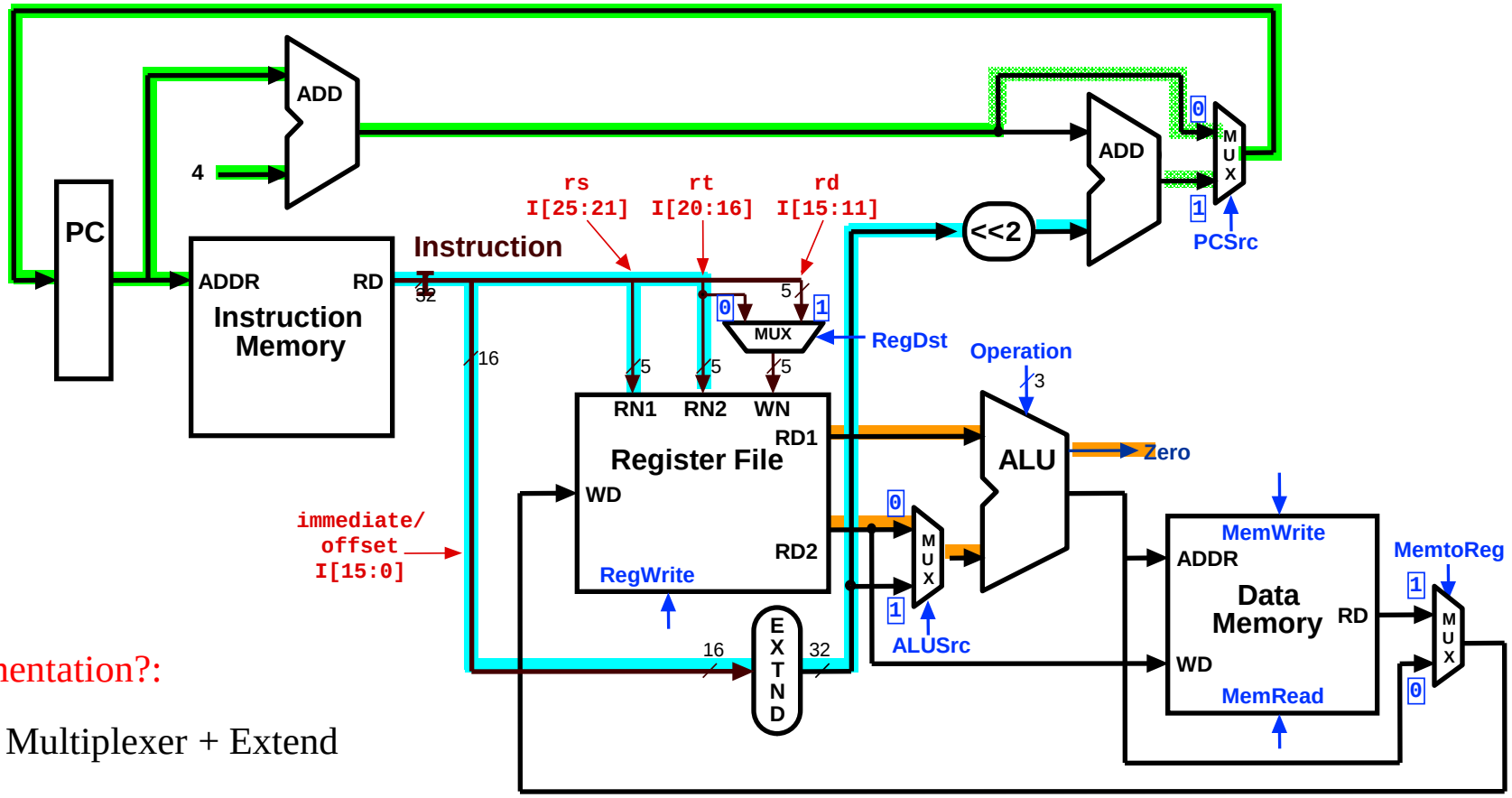
- Register Transfer:

  PC <- (PC + 4)[31:28] @ (I[25:0] << 2)

- Remember, it's unconditional

Final Extension: Implementing `j` (jump)





Implementation?:

1 Multiplexer + Extend

Final Extension: Implementing jump