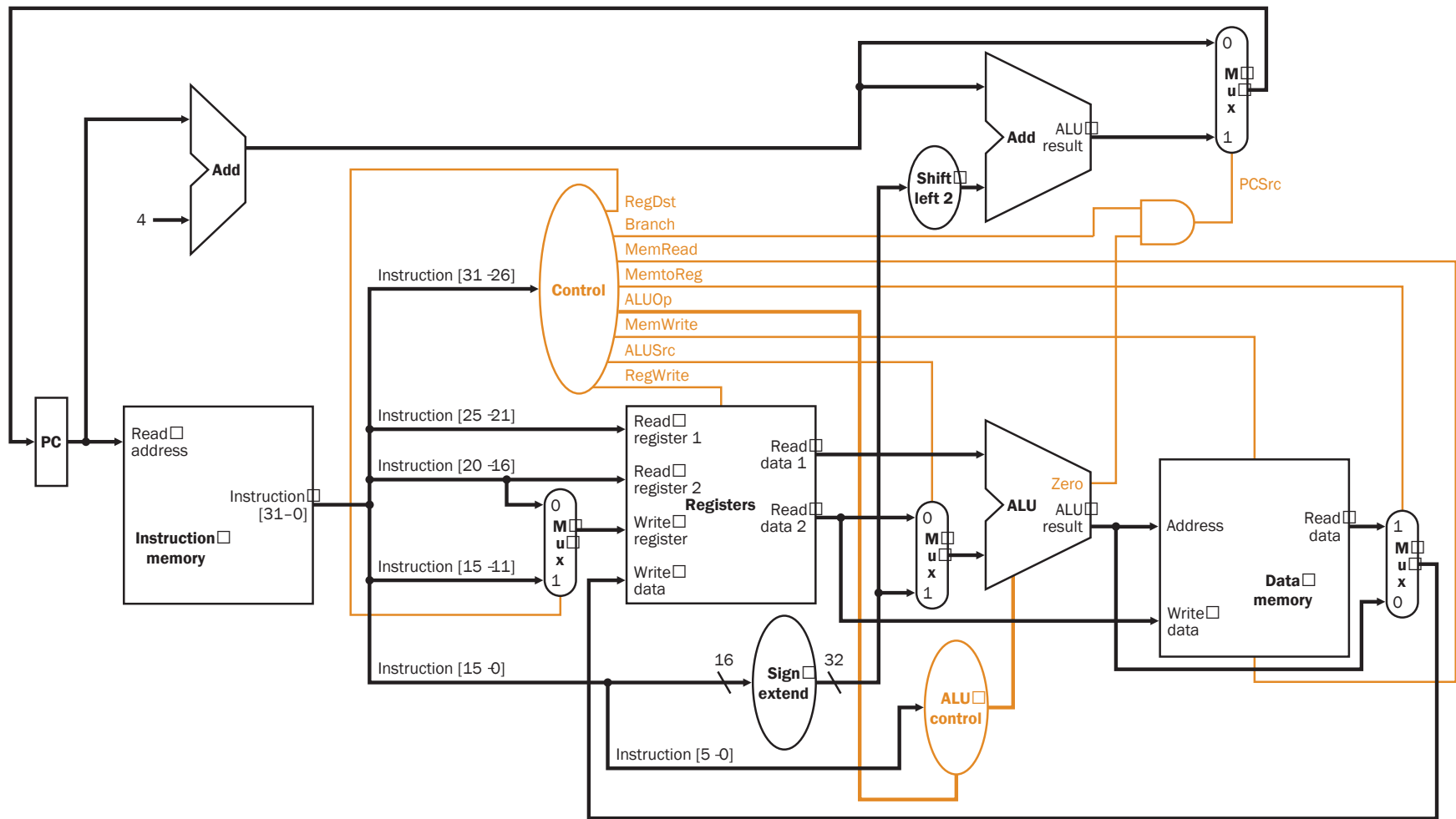


3. **Give Up the Funk (30 points):** Consider the single-cycle processor implementation. Your task will be to augment this datapath with a new instruction: the *funkyb* instruction. This instruction will be an R-type instruction, and will have the following effect:

```
if (R[rs]<R[rt])
    PC=PC+4+R[rs]           // Note that these two statements
    R[rd]=R[rt]             // will be concurrent.
else
    PC=PC+4+R[rt]           // Note that these two statements
    R[rd]=R[rs]             // will be concurrent.
```

Implement *funkyb* on the **single cycle datapath**. Use the R-type instruction format – so this instruction will have the same opcode as all other R-types. Use a unique function field to modify the ALU controller to implement this instruction, not the main controller.

**Implement your solution on the following two pages.** All other instructions must still work correctly after your modifications. You should not add any new ALUs, register file ports, or ports to memory.



### Main Controller

Input or Output	Signal Name	R-format	lw	sw	Beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

### ALU Controller

Opcode	ALUOp	instruction	function	ALU Action	ALUCtrl
Lw	00	load word	XXXXXX	add	010
Sw	00	store word	XXXXXX	add	010
Beq	01	branch equal	XXXXXX	subtract	110
R-type	10	add	100000	add	010
R-type	10	subtract	100010	subtract	110
R-type	10	AND	100100	AND	000
R-type	10	OR	100101	OR	001
R-type	10	SLT	101010	SLT	111