

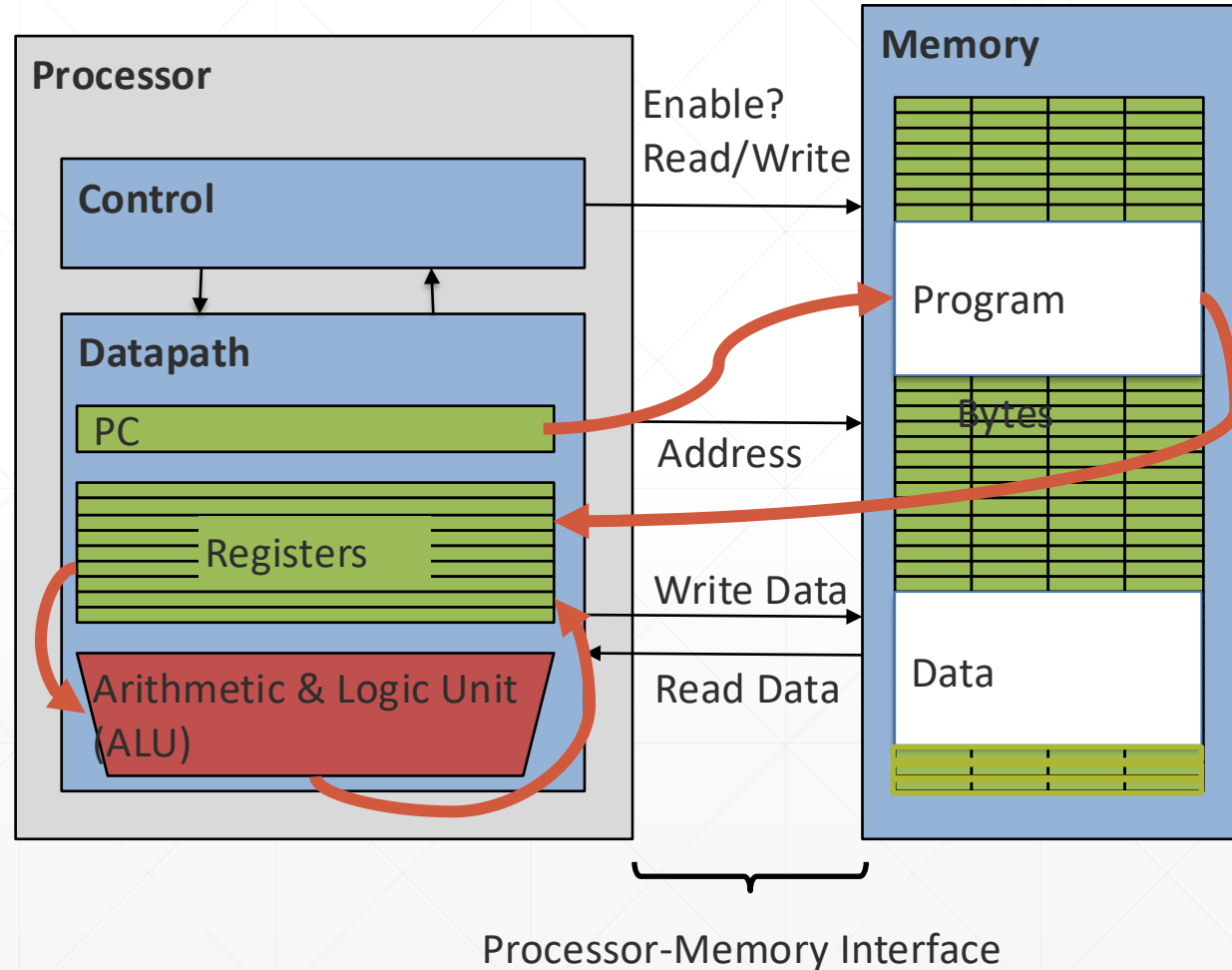
CS110A

Datapath and

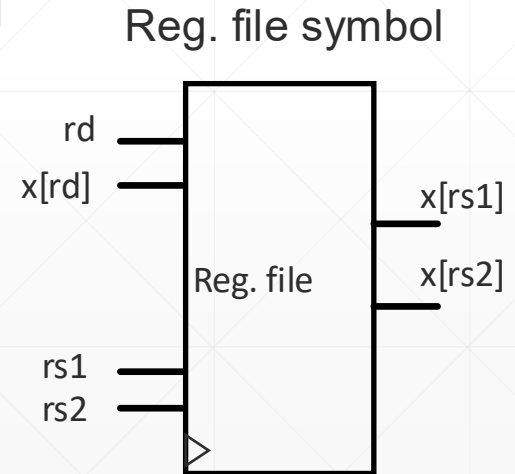
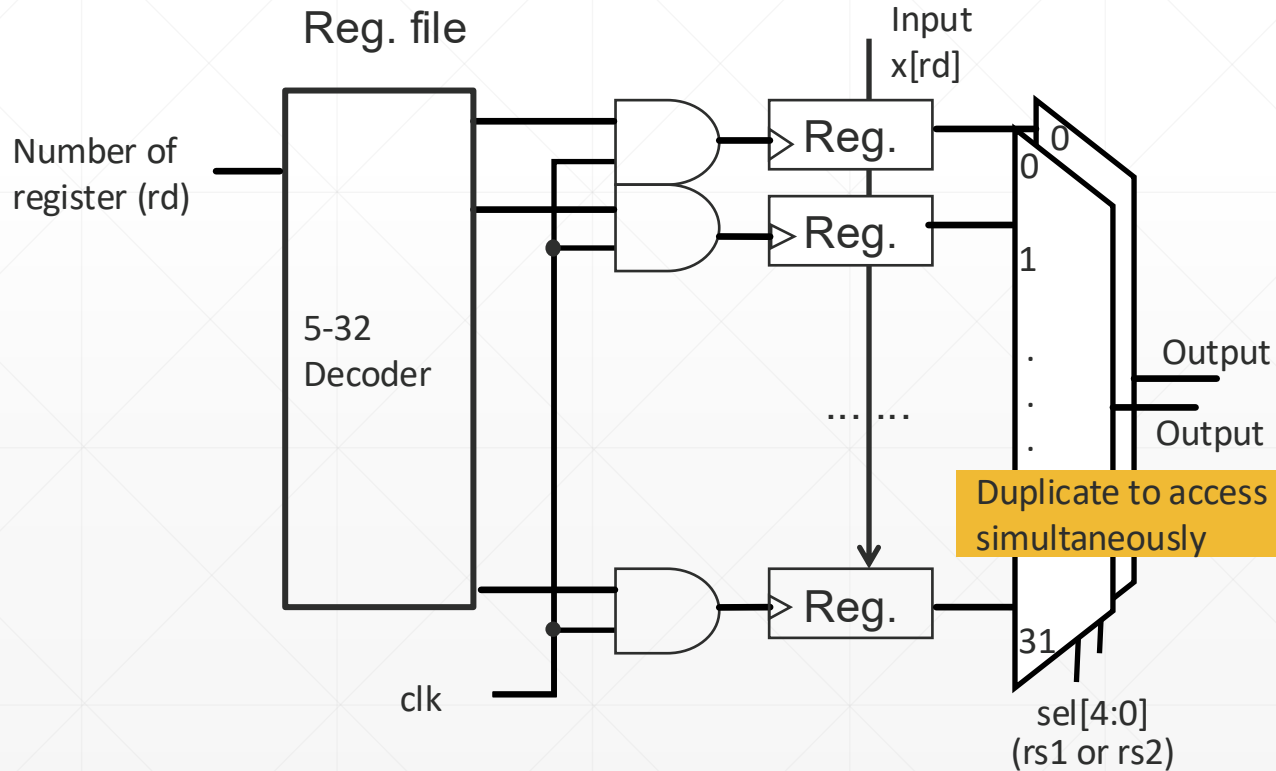
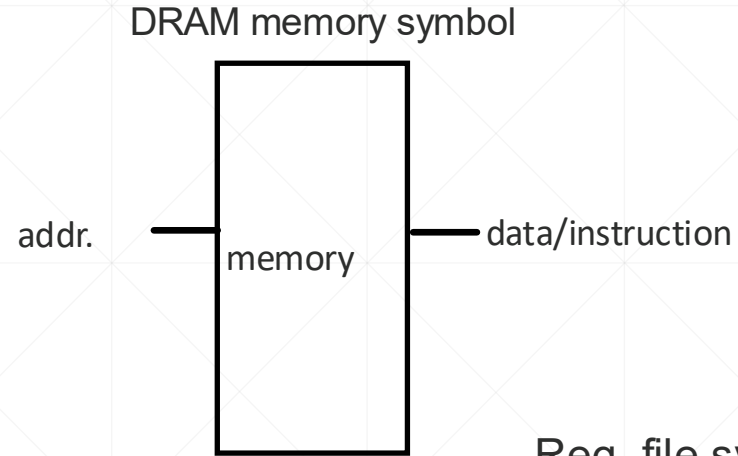
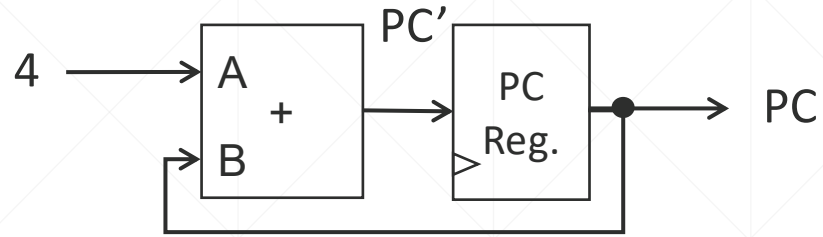
Control Logic (II)

Chundong Wang

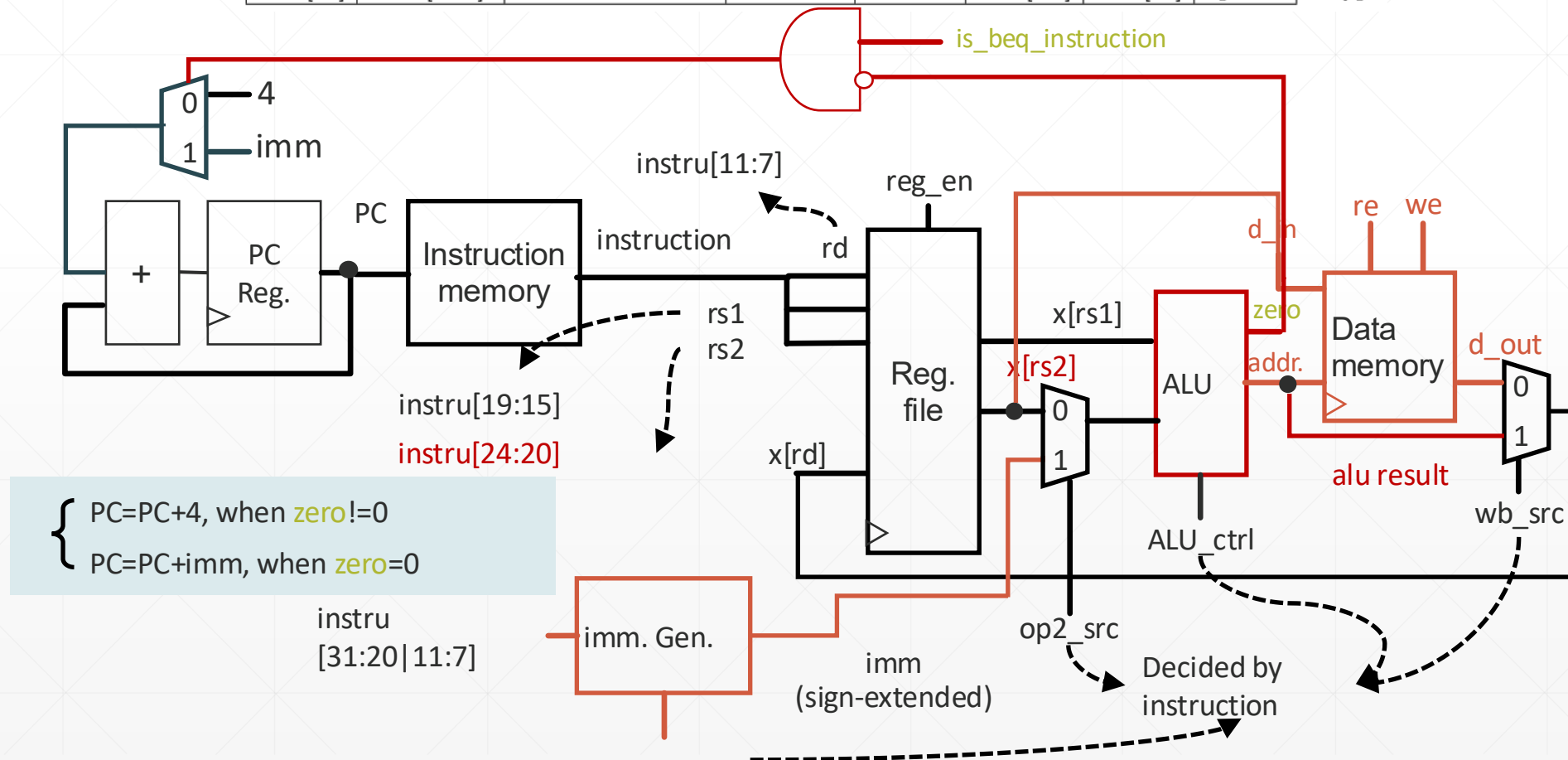
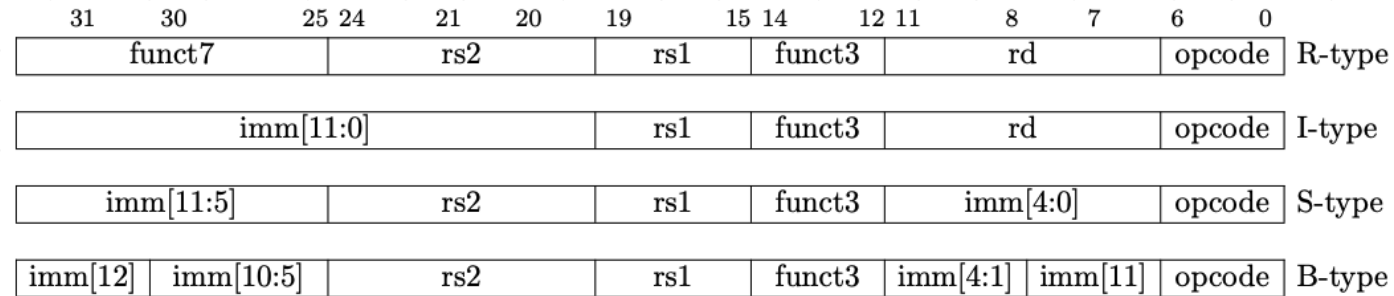
Datapath and Control Logic



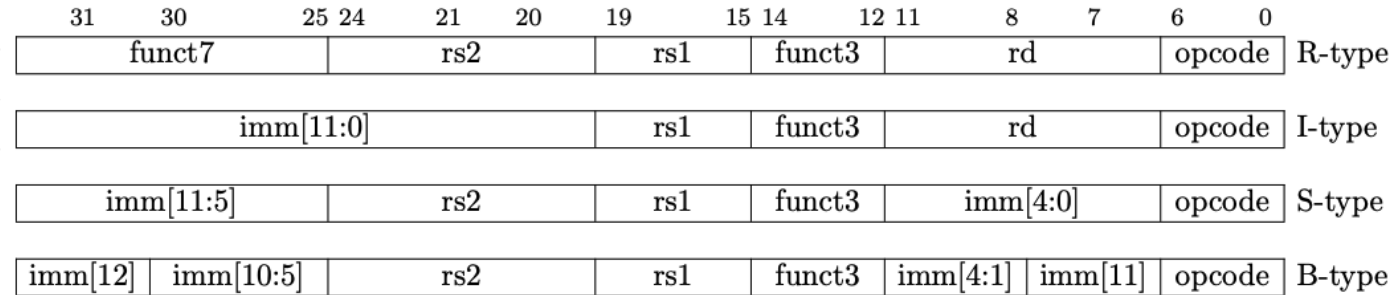
Datapath



Datapath for B-type

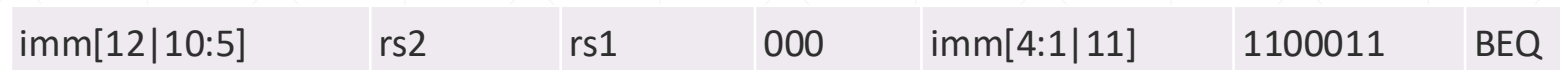


Datapath for B-type



`is_beq_instruction`

- Recall beq

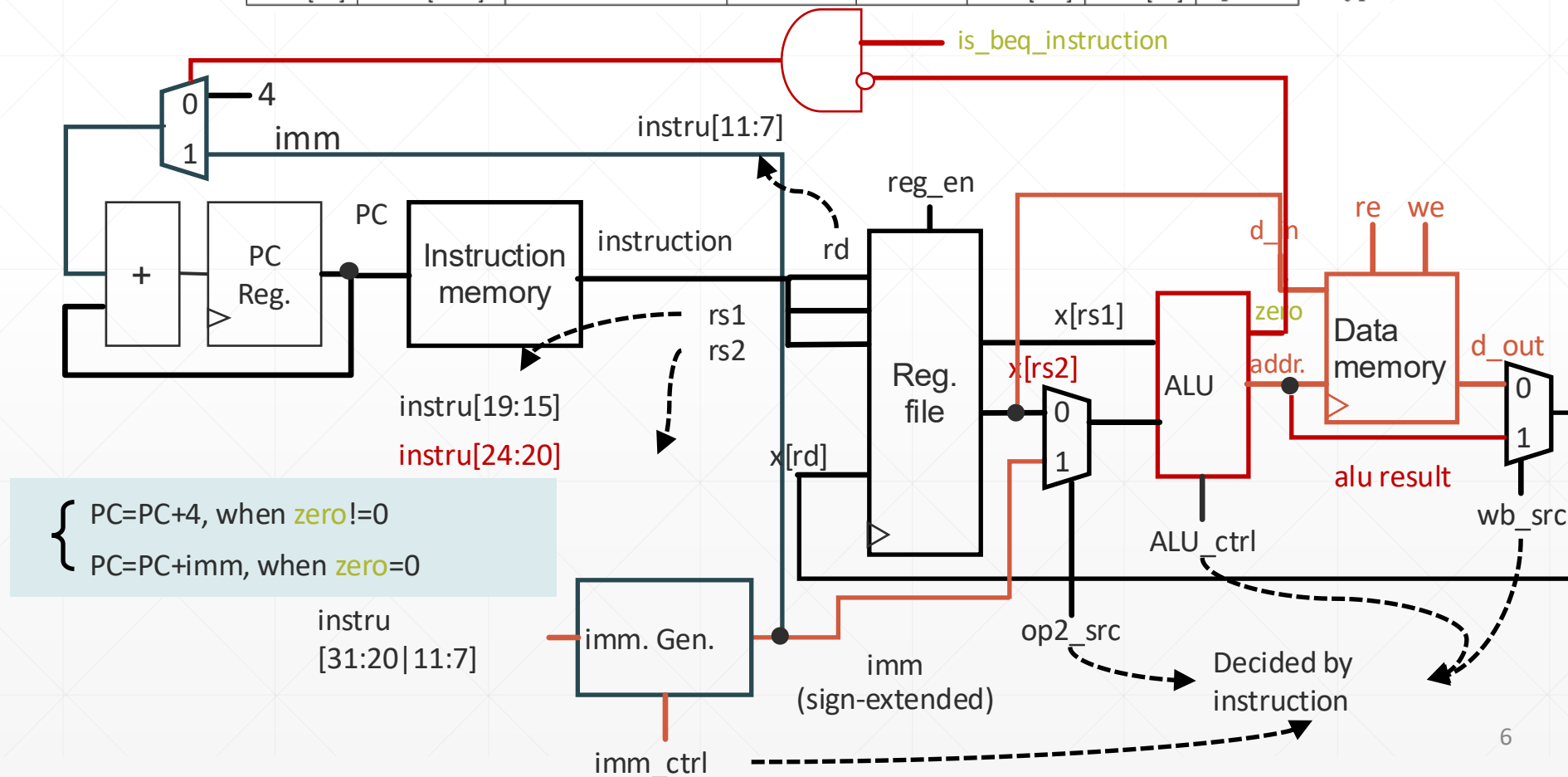
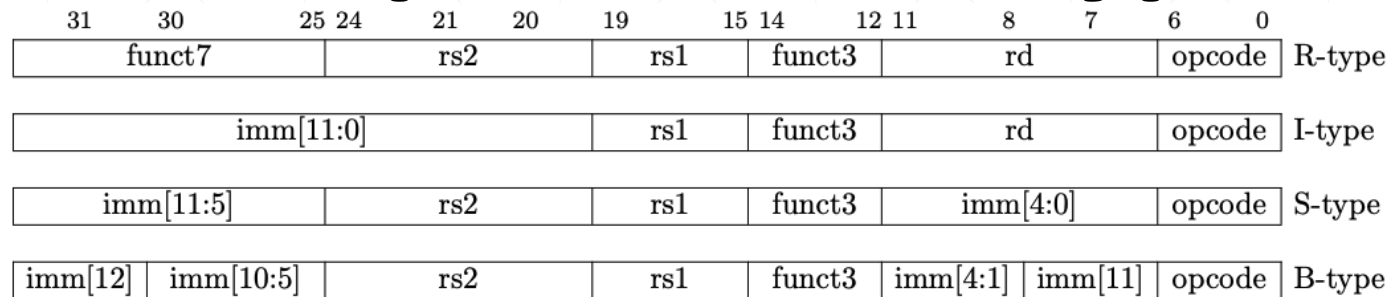


Truth table

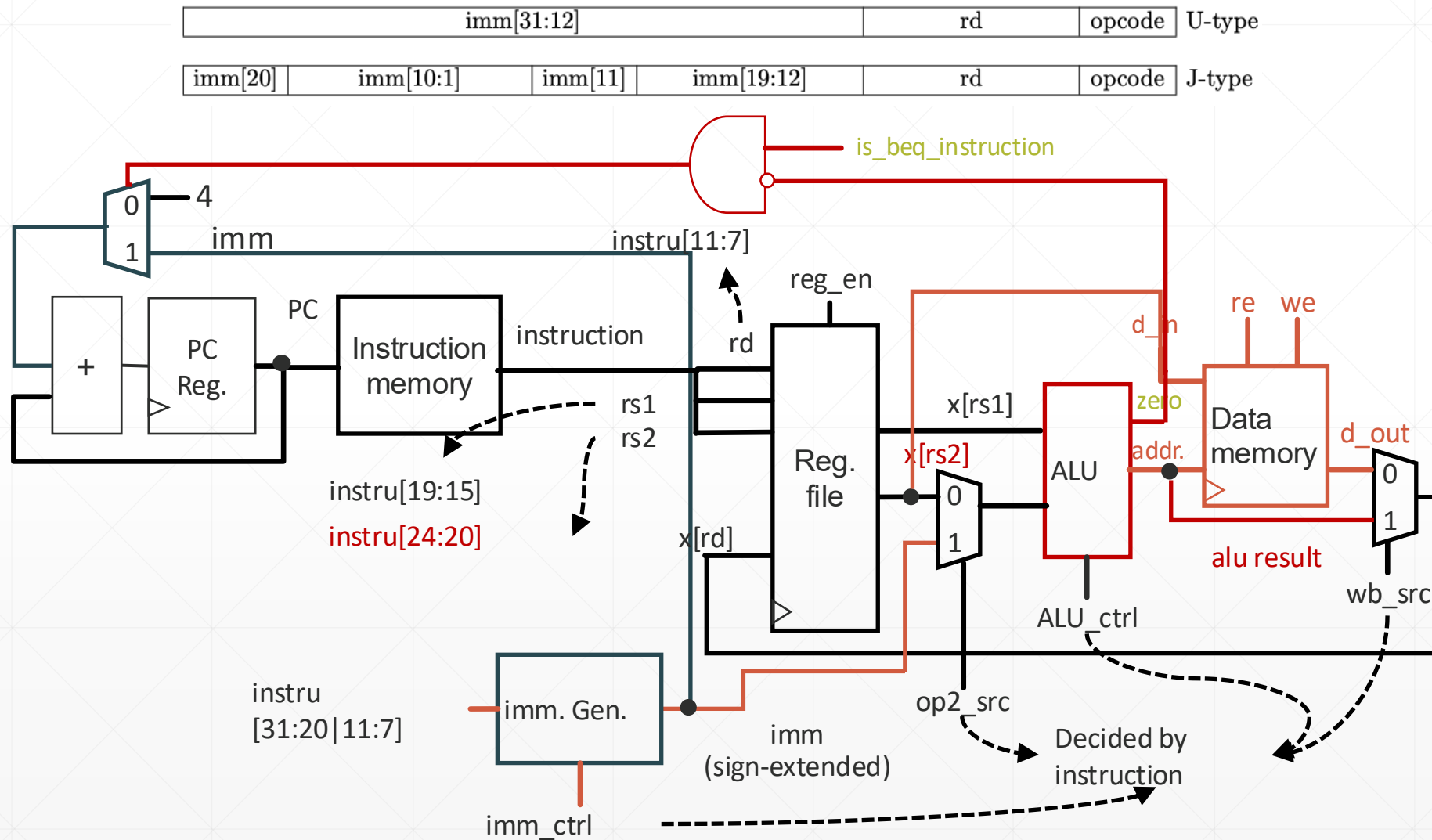
Instru[14:12 6:0]	<code>is_beq_instruction</code>
000 1100011	1
All the other cases	0

`is_beq_instruction = (¬i[14])(¬i[13])(¬i[12])i[6]i[5](¬i[4])(¬i[3])(¬i[2])i[1]i[0]`

Datapath for B-type

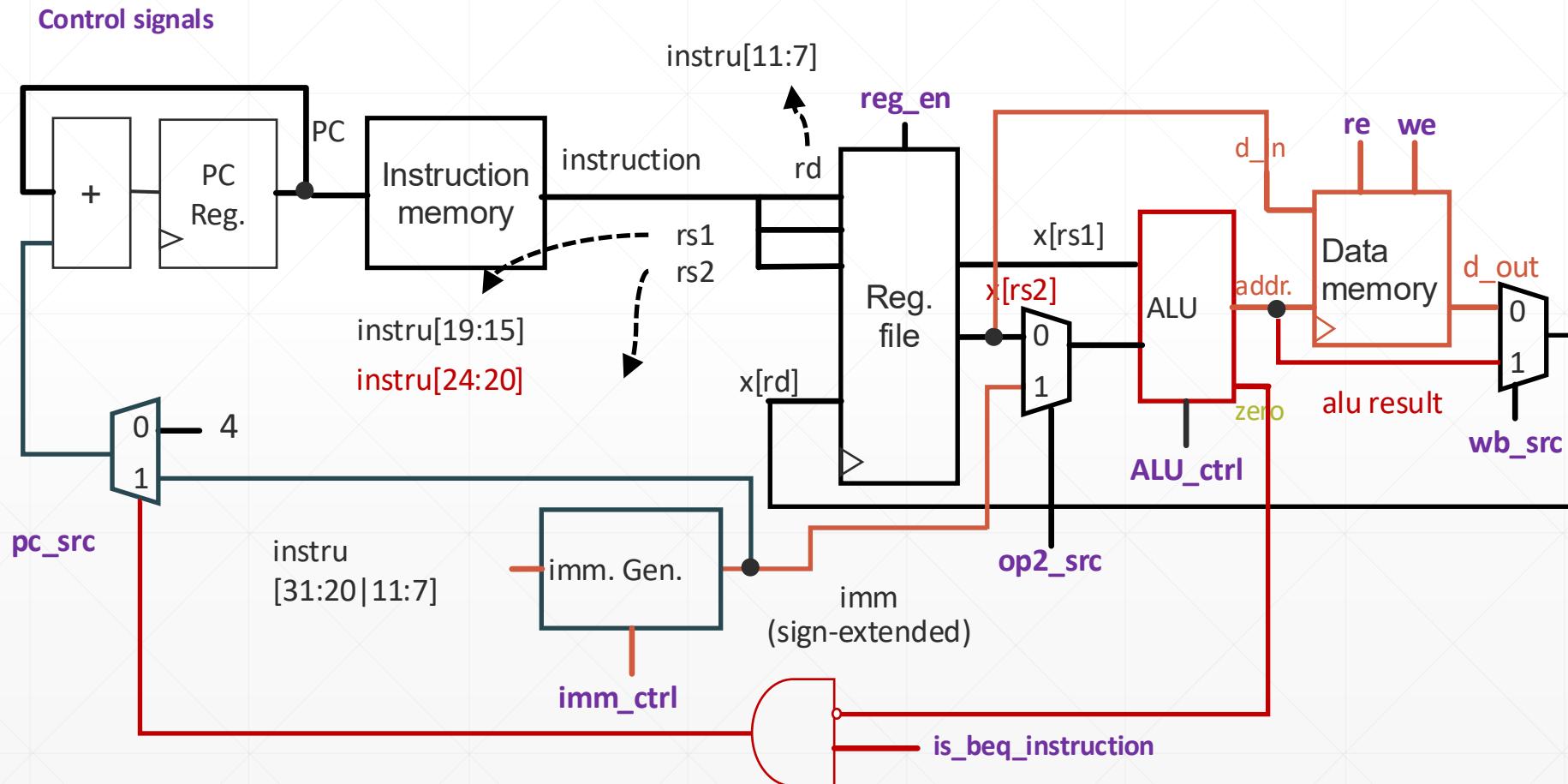


Datapath for the other

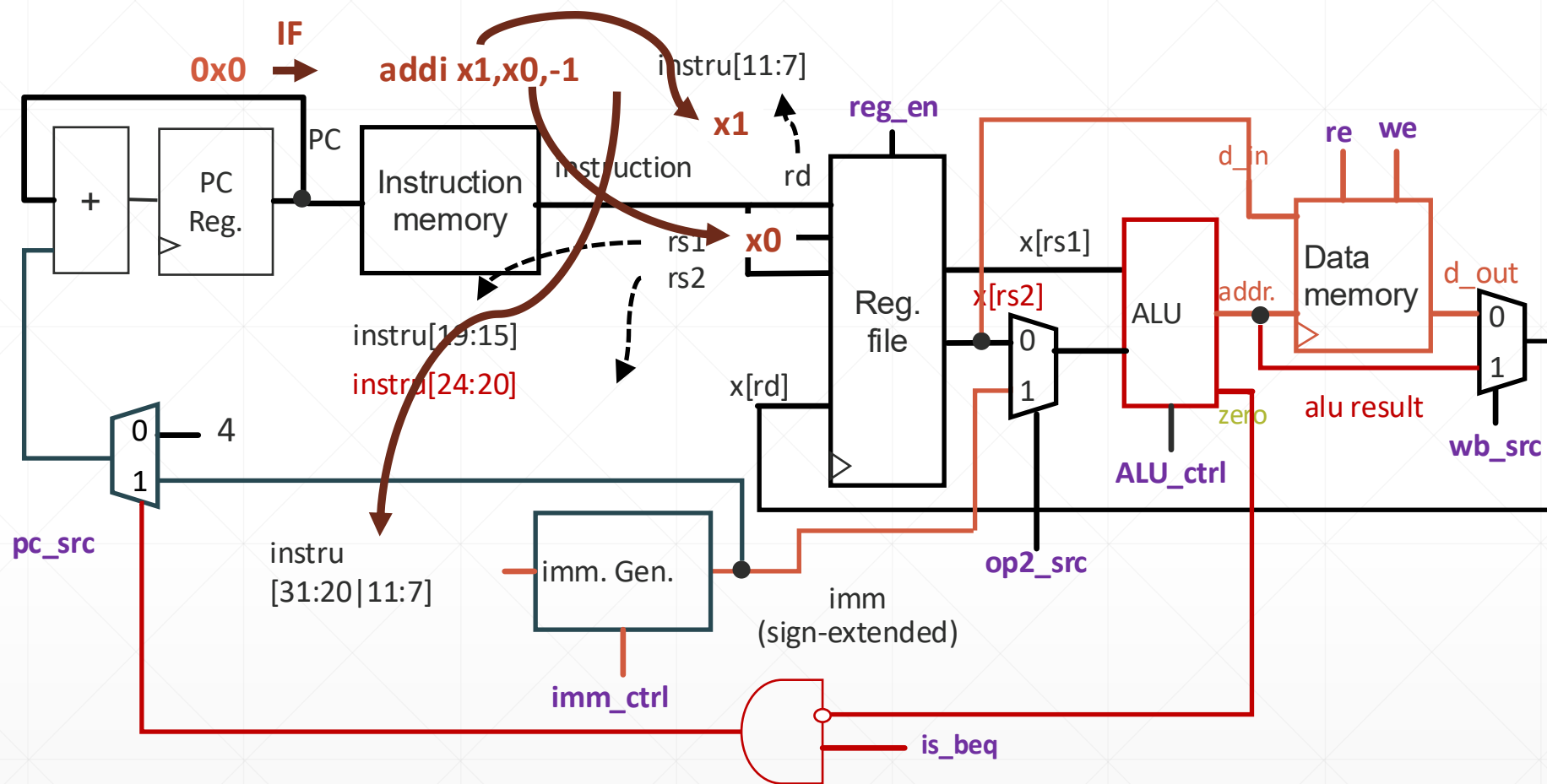


Control signals

- This is a datapath that supports R-type & I-type arithmetic and logic operations, lw, sw and beq



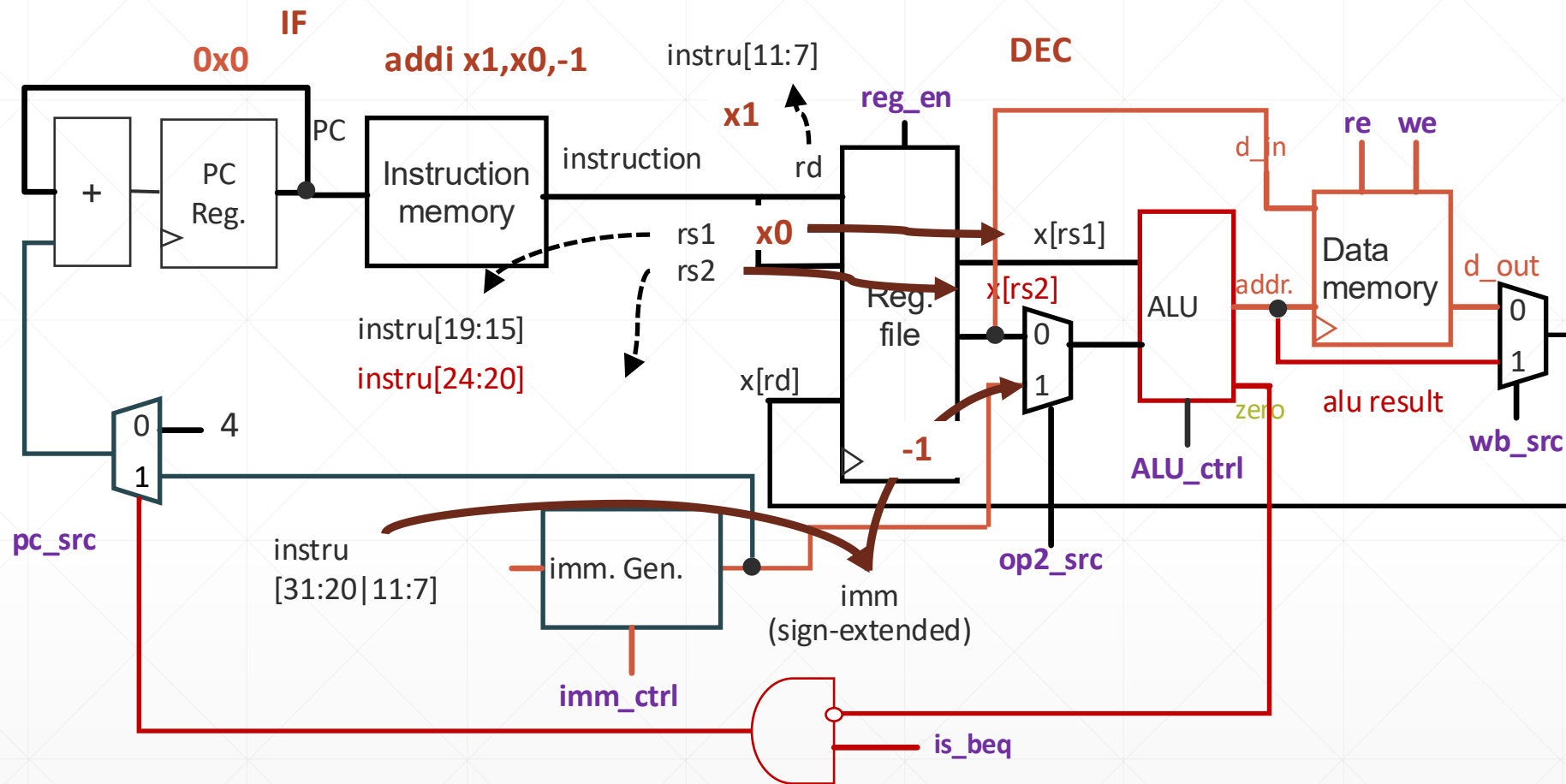
Datapath working example



	reg_en	re	we	alu_ctrl	imm_ctrl	wb_src	op2_src	is_beq
addi	1	0	0	add	I-type	1	1	0

0x0: addi x1, x0, -1
 0x4: ori x2, x0, 128
 0x8: add x3, x1, x2
 0xc: sw x3, 0(x3)
 0x10: lw x5, 0(x3)
 0x14: beq x3, x5, -12

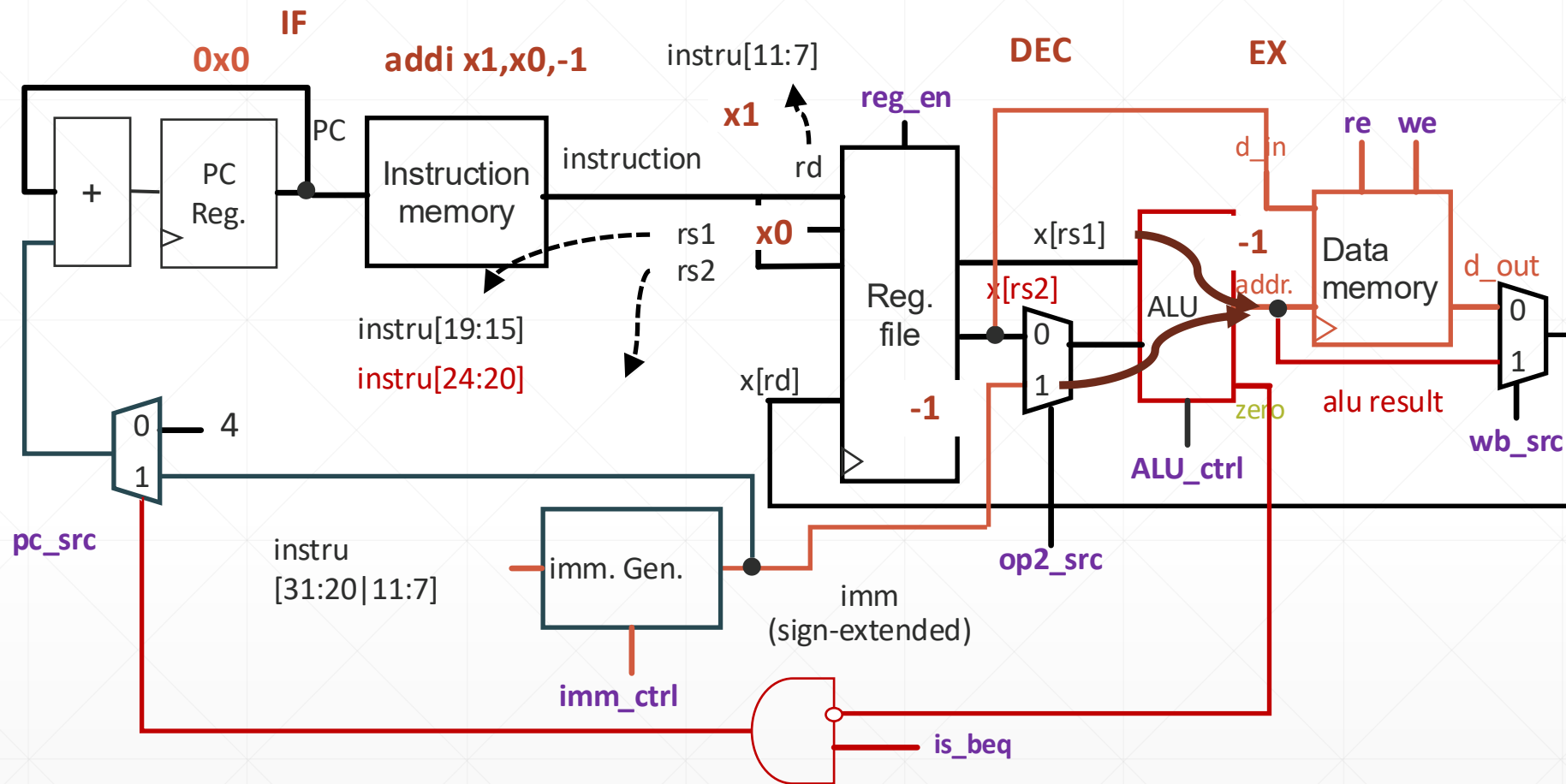
Datapath working example



	reg_en	re	we	alu_ctrl	imm_ctrl	wb_src	op2_src	is_beq
addi	1	0	0	add	I-type	1	1	0

0x0: `addi x1, x0, -1`
 0x4: `ori x2, x0, 128`
 0x8: `add x3, x1, x2`
 0xc: `sw x3, 0(x3)`
 0x10: `lw x5, 0(x3)`
 0x14: `beq x3, x5, -12`

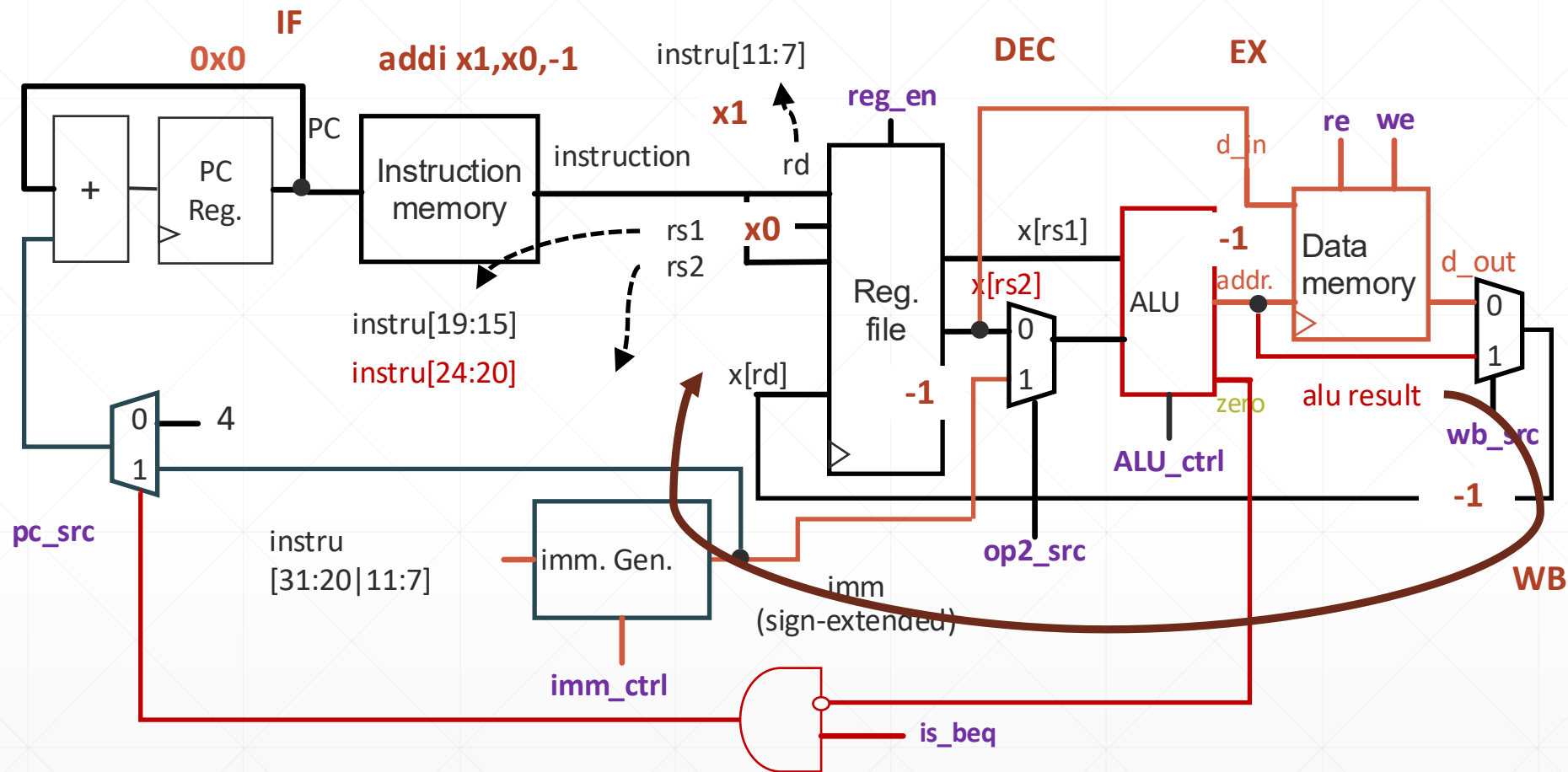
Datapath working example



	reg_en	re	we	alu_ctrl	imm_ctrl	wb_src	op2_src	is_beq
addi	1	0	0	add	I-type	1	1	0

0x0: `addi x1, x0, -1`
 0x4: `ori x2, x0, 128`
 0x8: `add x3, x1, x2`
 0xc: `sw x3, 0(x3)`
 0x10: `lw x5, 0(x3)`
 0x14: `beq x3, x5, -12`

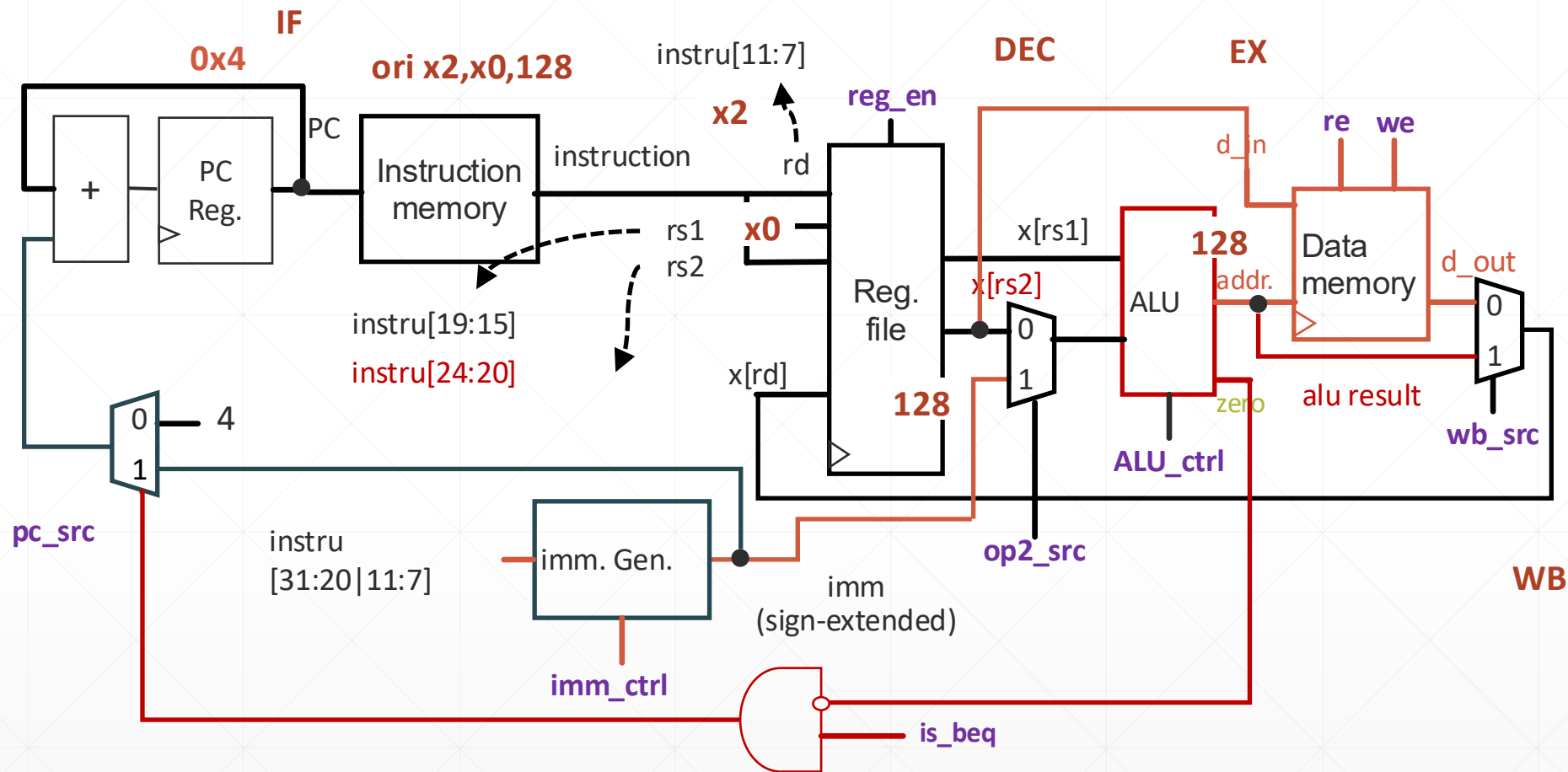
Datapath working example



	reg_en	re	we	alu_ctrl	imm_ctrl	wb_src	op2_src	is_beq
addi	1	0	0	add	I-type	1	1	0

0x0: `addi x1, x0, -1`
 0x4: `ori x2, x0, 128`
 0x8: `add x3, x1, x2`
 0xc: `sw x3, 0(x3)`
 0x10: `lw x5, 0(x3)`
 0x14: `beq x3, x5, -12`

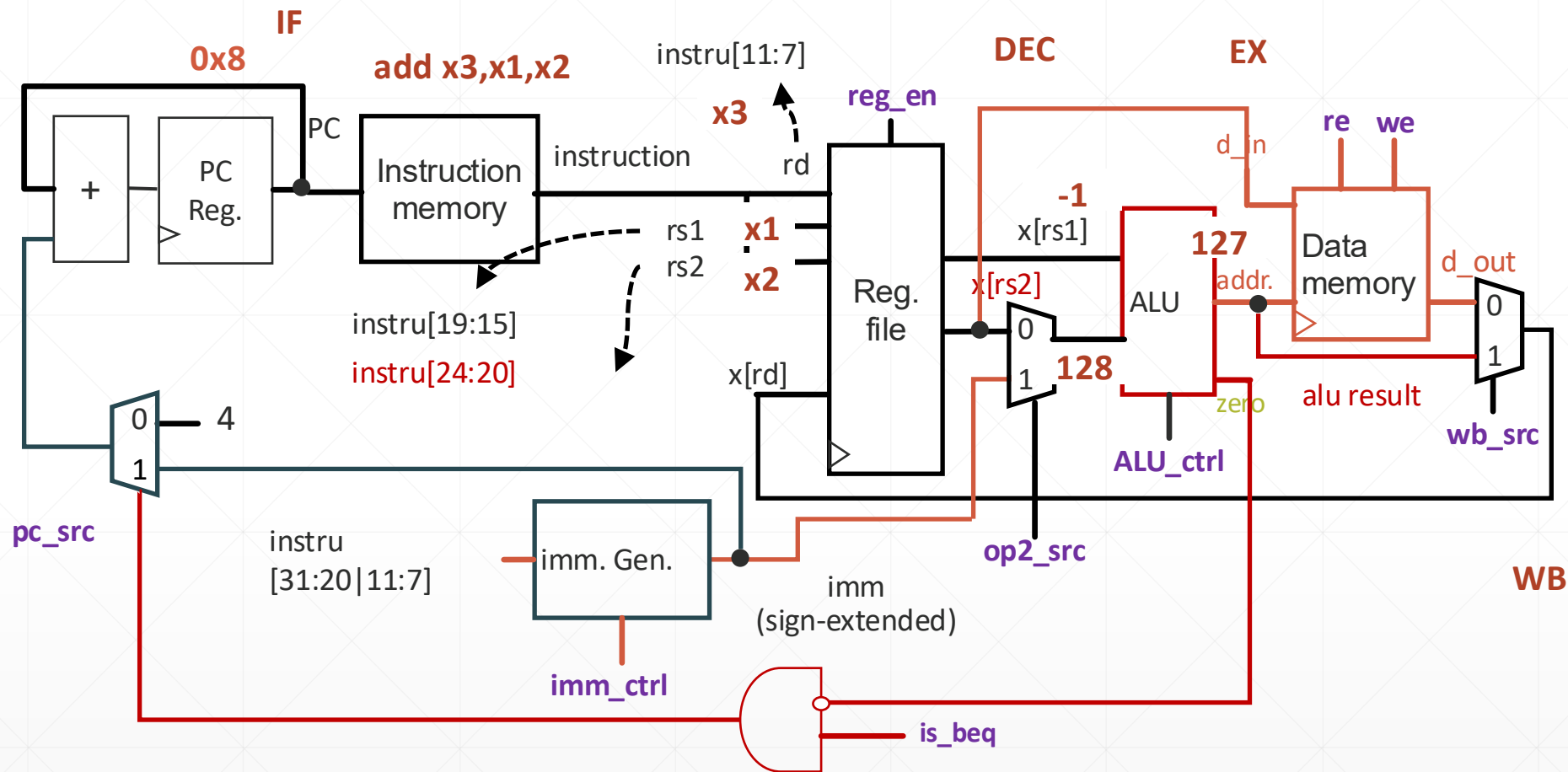
Datapath working example



	reg_en	re	we	alu_ctrl	imm_ctrl	wb_src	op2_src	is_beq
ori	1	0	0	or	I-type	1	1	0

0x0: `addi x1, x0, -1`
 0x4: `ori x2, x0, 128`
 0x8: `add x3, x1, x2`
 0xc: `sw x3, 0(x3)`
 0x10: `lw x5, 0(x3)`
 0x14: `beq x3, x5, -12`

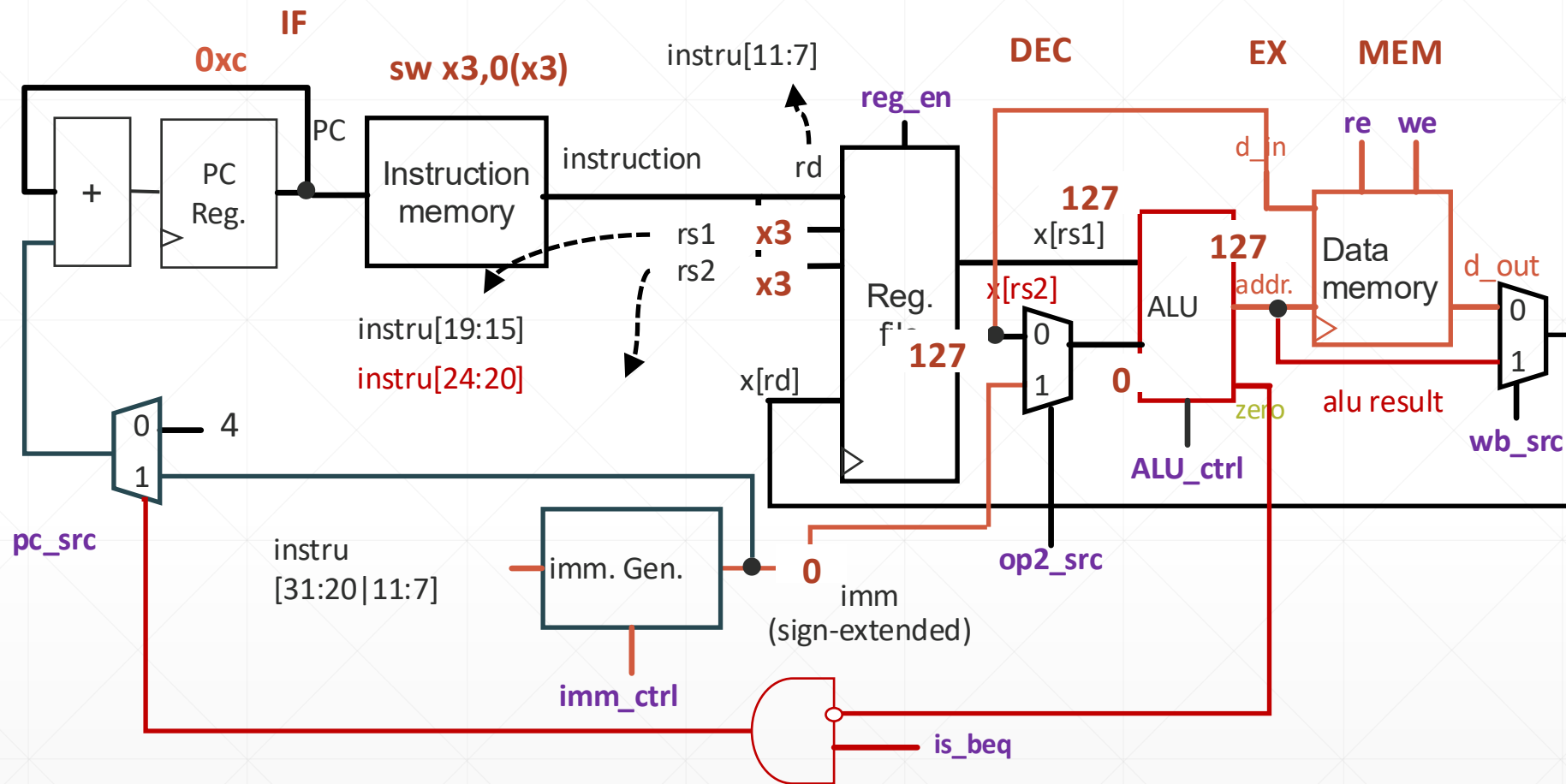
Datapath working example



	reg_en	re	we	alu_ctrl	imm_ctrl	wb_src	op2_src	is_beq
add	1	0	0	add	X	1	0	0

0x0: addi x1, x0, -1
 0x4: ori x2, x0, 128
0x8: add x3, x1, x2
 0xc: sw x3, 0(x3)
 0x10: lw x5, 0(x3)
 0x14: beq x3, x5, -12

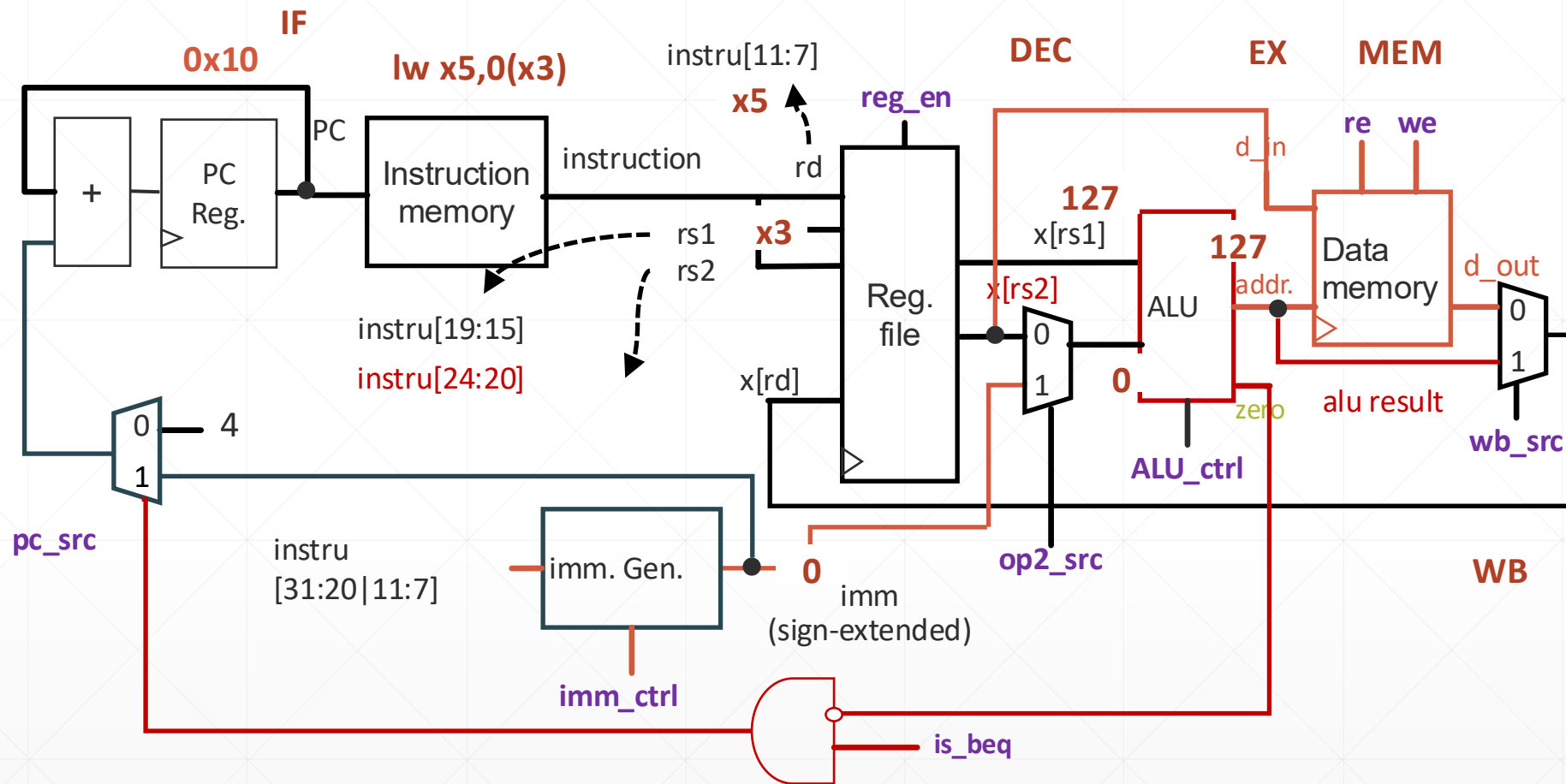
Datapath working example



	reg_en	re	we	alu_ctrl	imm_ctrl	wb_src	op2_src	is_beq
sw	0	0	1	add	S-type	X	1	0

0x0: addi x1, x0, -1
 0x4: ori x2, x0, 128
 0x8: add x3, x1, x2
0xc: sw x3, 0(x3)
 0x10: lw x5, 0(x3)
 0x14: beq x3, x5, -12

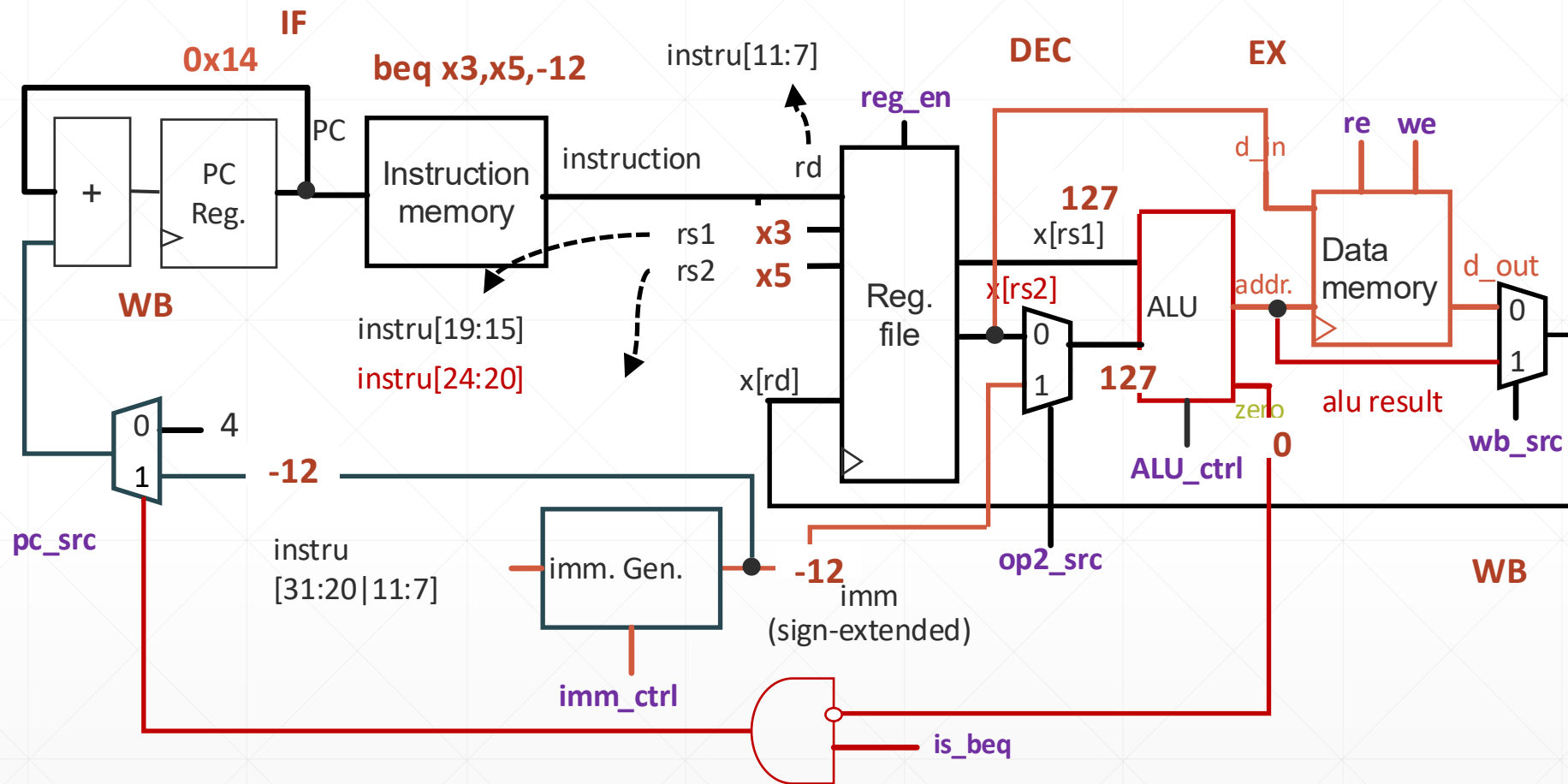
Datapath working example



	reg_en	re	we	alu_ctrl	imm_ctrl	wb_src	op2_src	is_beq
lw	1	1	0	add	I-type	0	1	0

0x0: `addi x1, x0, -1`
 0x4: `ori x2, x0, 128`
 0x8: `add x3, x1, x2`
 0xc: `sw x3, 0(x3)`
 0x10: `lw x5, 0(x3)`
 0x14: `beq x3, x5, -12`

Datapath working example



	reg_en	re	we	alu_ctrl	imm_ctrl	wb_src	op2_src	is_beq
beq	0	0	0	X	B-type	X	0	1

0x0: `addi x1, x0, -1`

0x4: `ori x2, x0, 128`

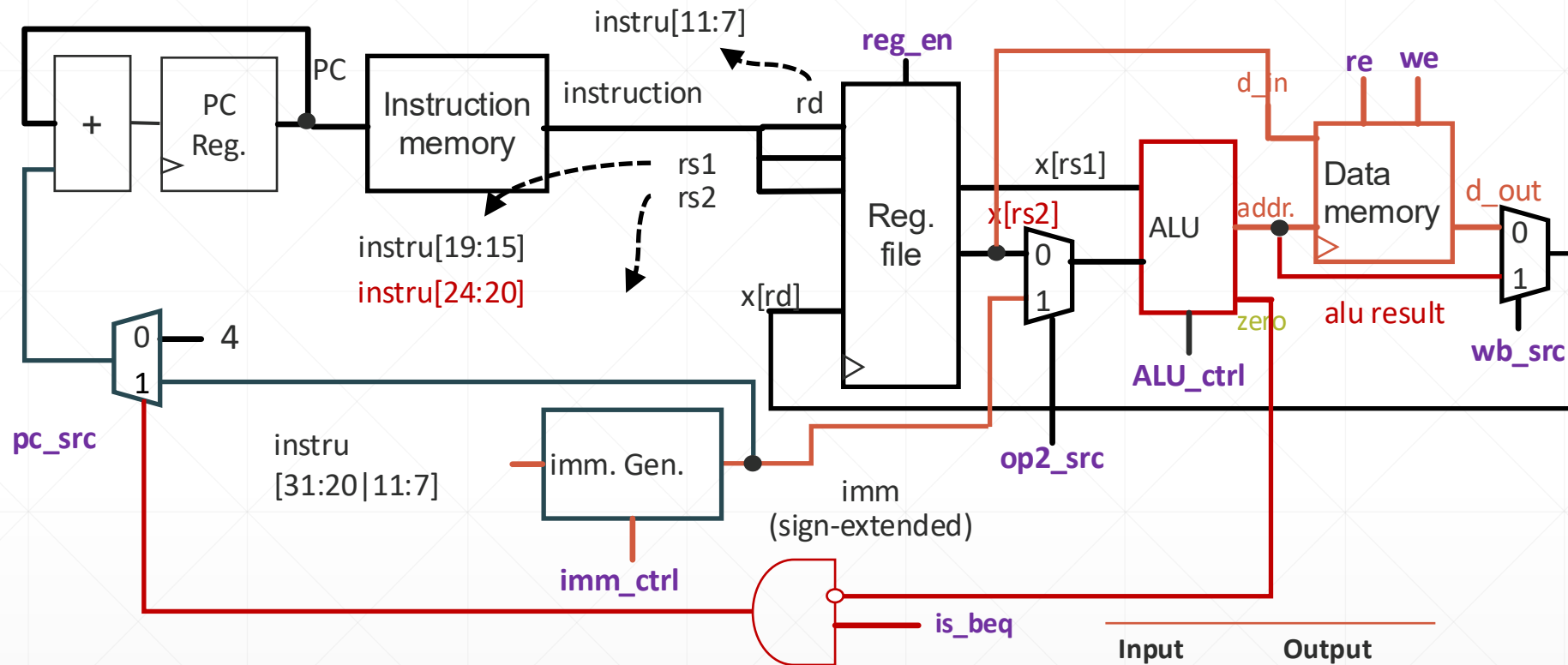
0x8: `add x3, x1, x2`

0xc: `sw x3, 0(x3)`

0x10: `lw x5, 0(x3)`

0x14: `beq x3, x5, -12`

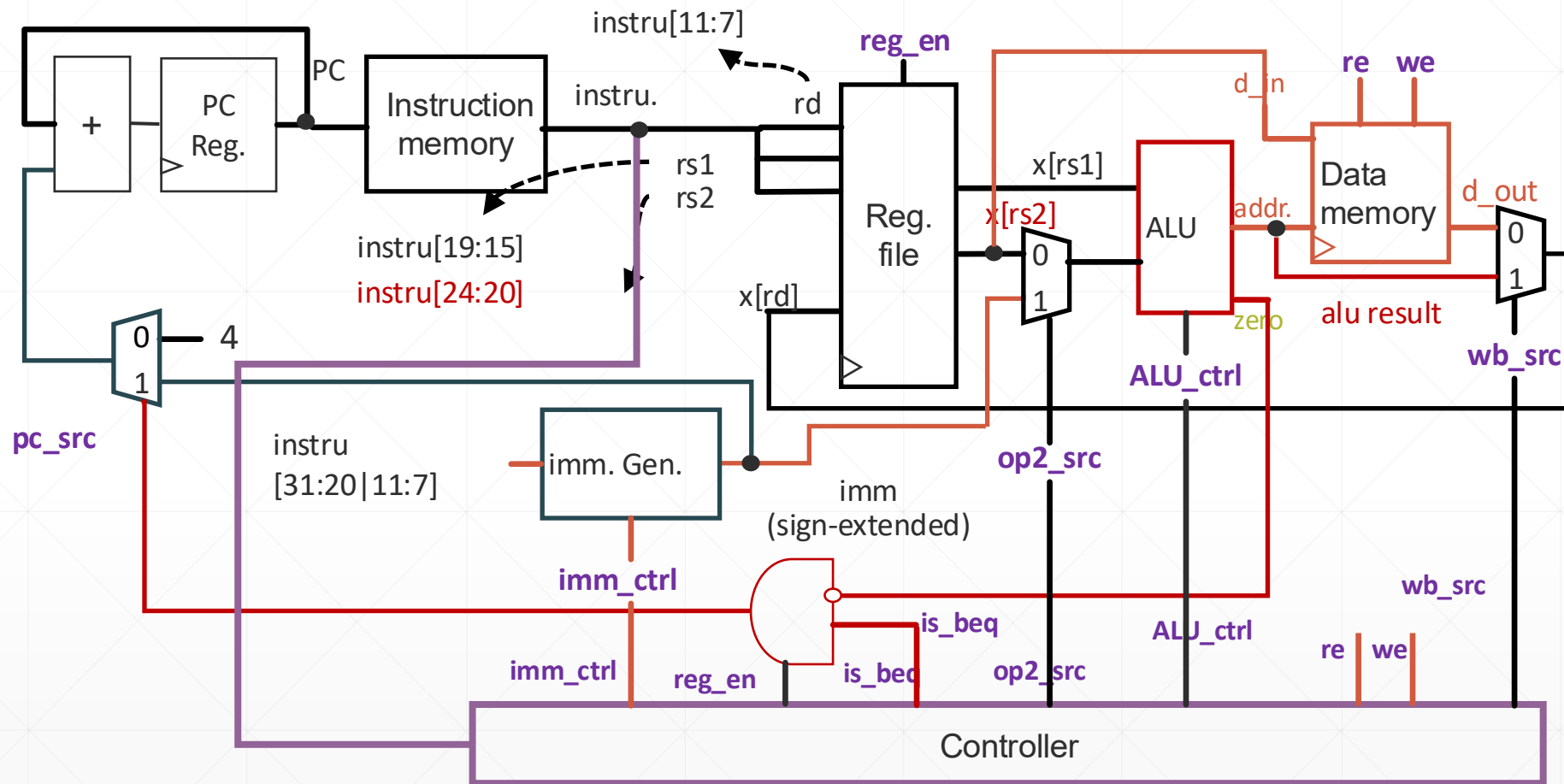
Controller



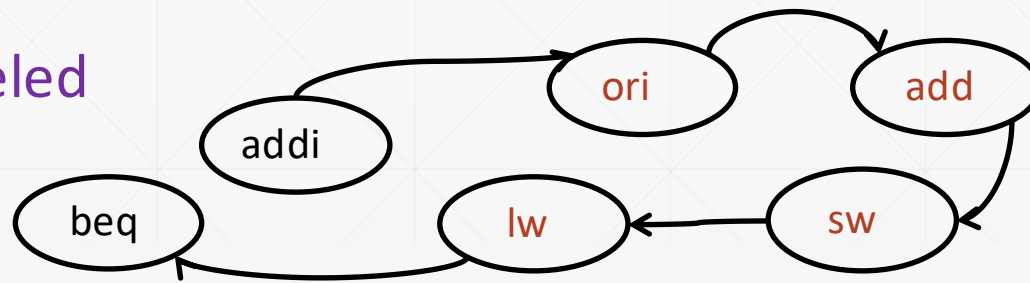
- Generate control signals guiding the datapath to execute instructions
- The inputs are instructions, the outputs are the control signals
- Once the type of instruction is determined, the control signal is determined

Input	Output
Instru.	reg_en
	re
	we
	alu_ctrl
	imm_ctrl
	wb_src
	op2_src
	is_beq

Controller

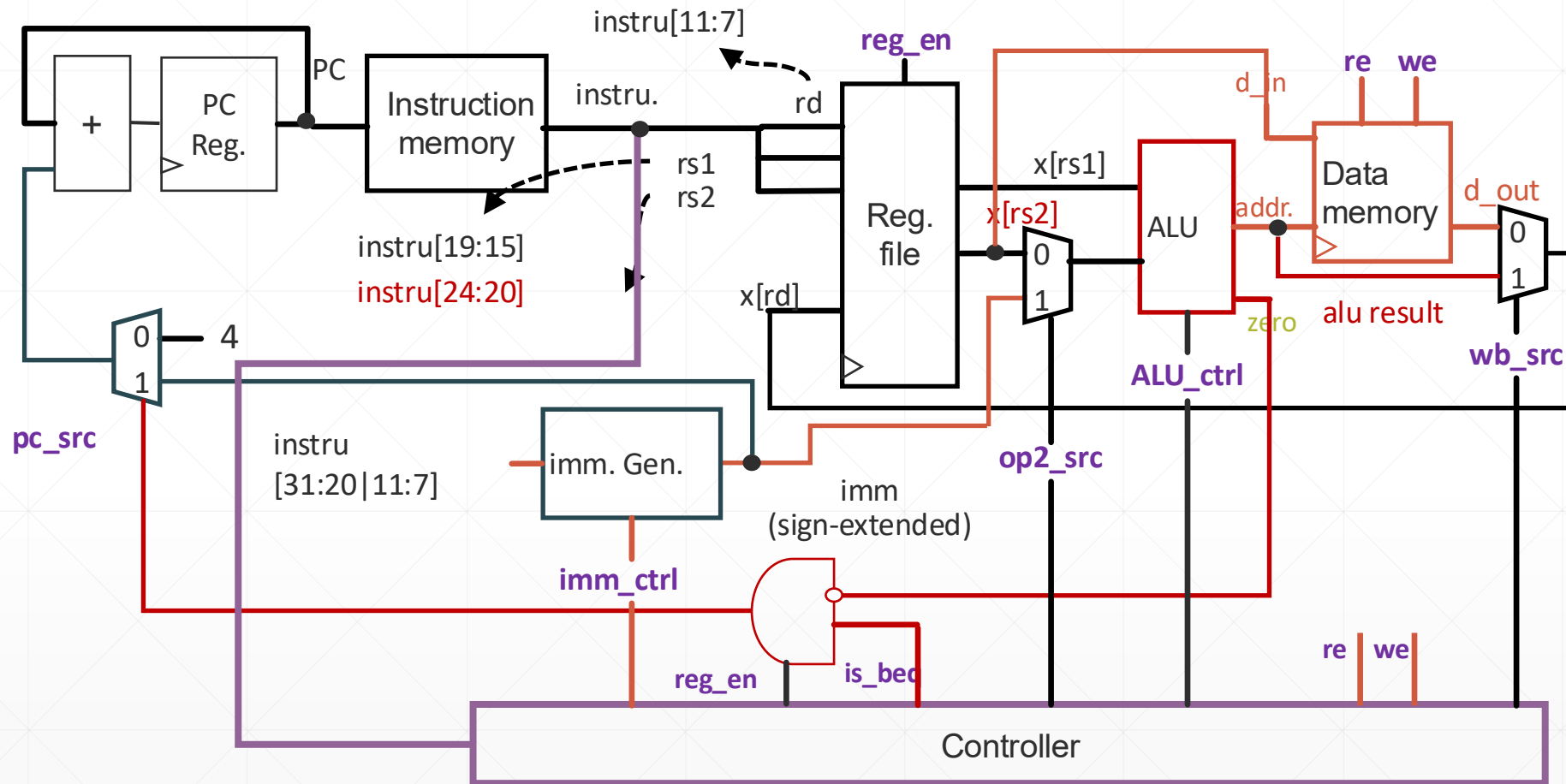


- Can be modeled by an FSM

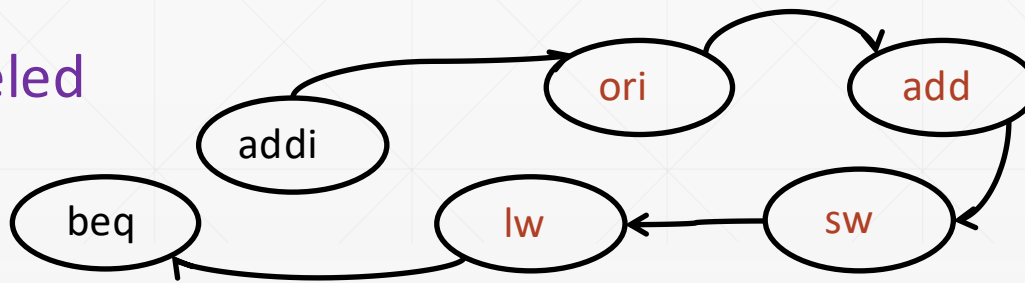


0x0: addi x1, x0, -1
 0x4: ori x2, x0, 128
 0x8: add x3, x1, x2
 0xc: sw x3, 0(x3)
 0x10: lw x5, 0(x3)
 0x14: beq x3, x5, -12

Controller

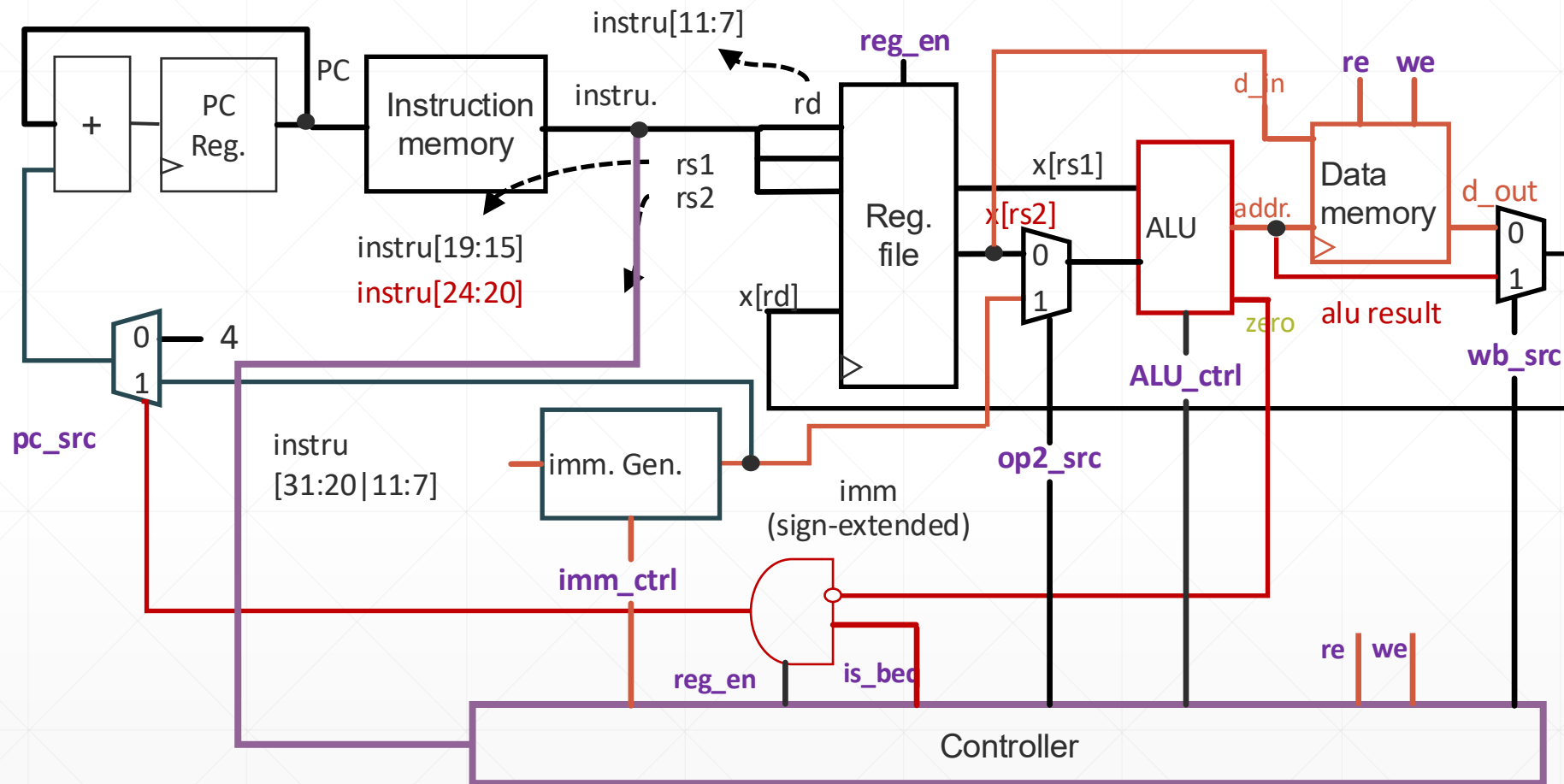


- Can be modeled by an FSM

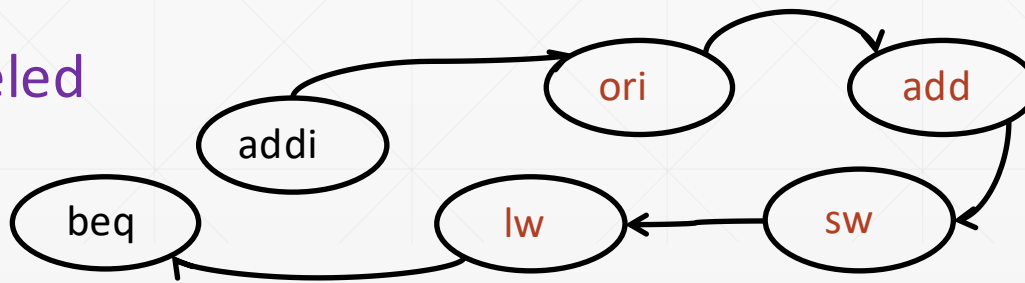


- Can be implemented by a Mealy machine

Controller

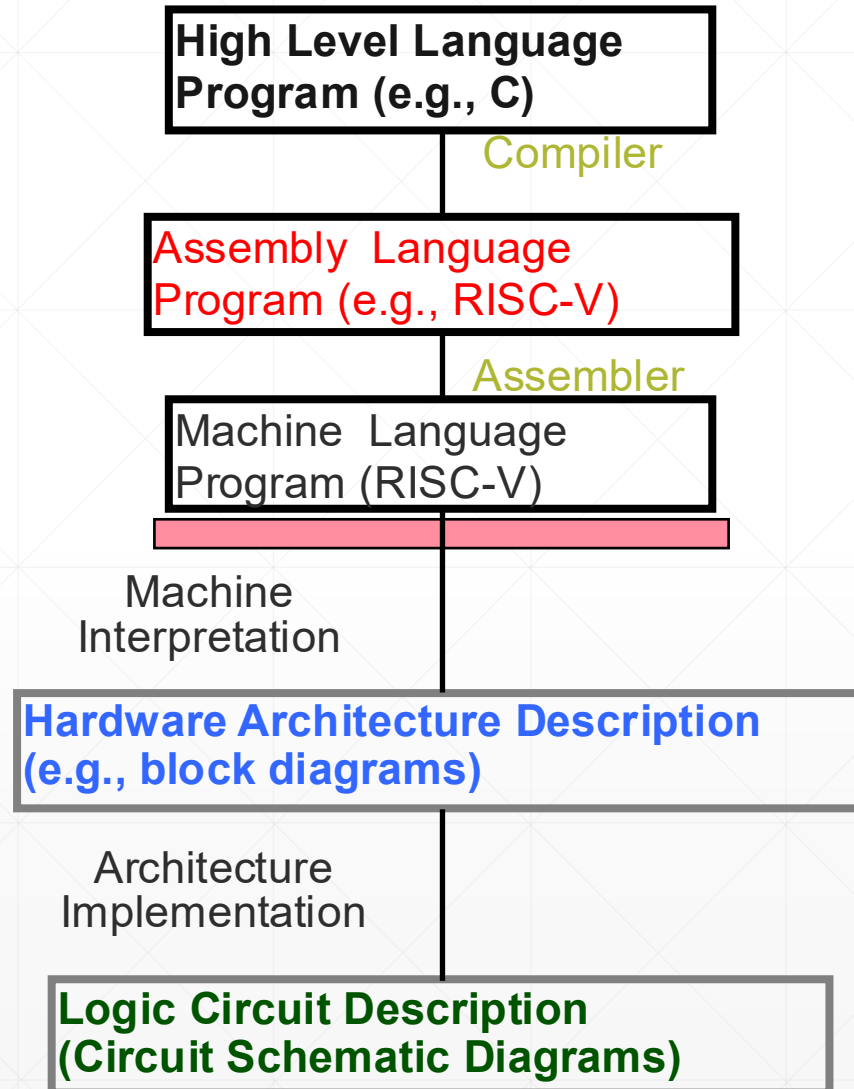


- Can be modeled by an FSM



- Can also be implemented by read-only memory (ROM), optional

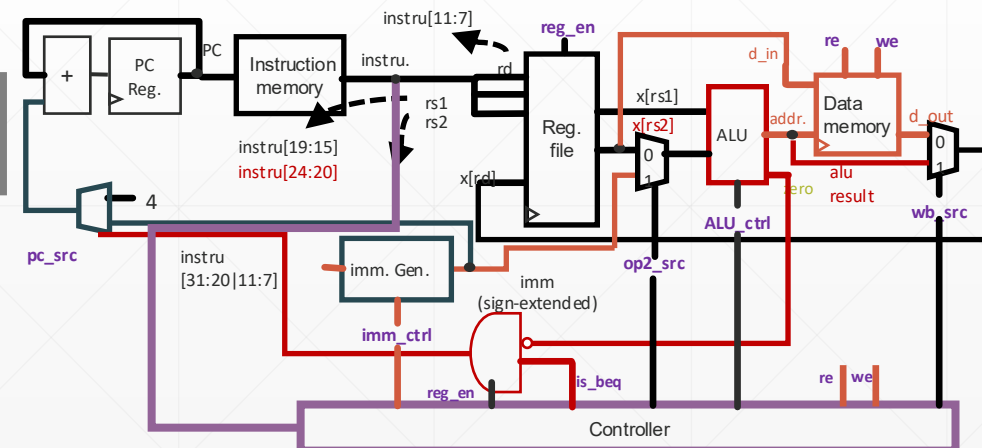
Full stack explained



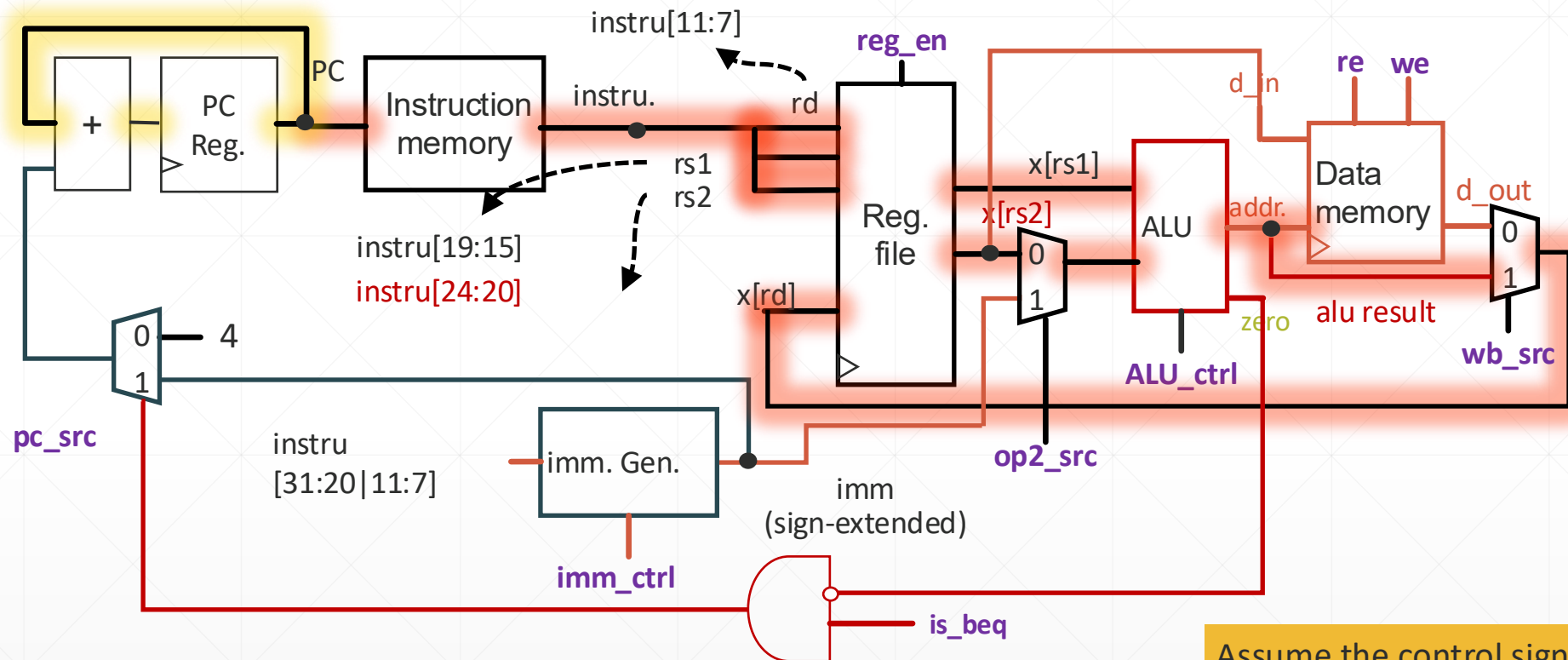
```
temp = v[k];  
v[k] = v[k+1];  
v[k+1] = temp;
```

```
lw    t0, 0(s2)  
lw    t1, 4(s2)  
sw    t1, 0(s2)  
sw    t0, 4(s2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000  
1010 1111 0101 1000 0000 1001 1100 0110  
1100 0110 1010 1111 0101 1000 0000 1001  
0101 1000 0000 1001 1100 0110 1010 1111
```



Datapath timing analysis



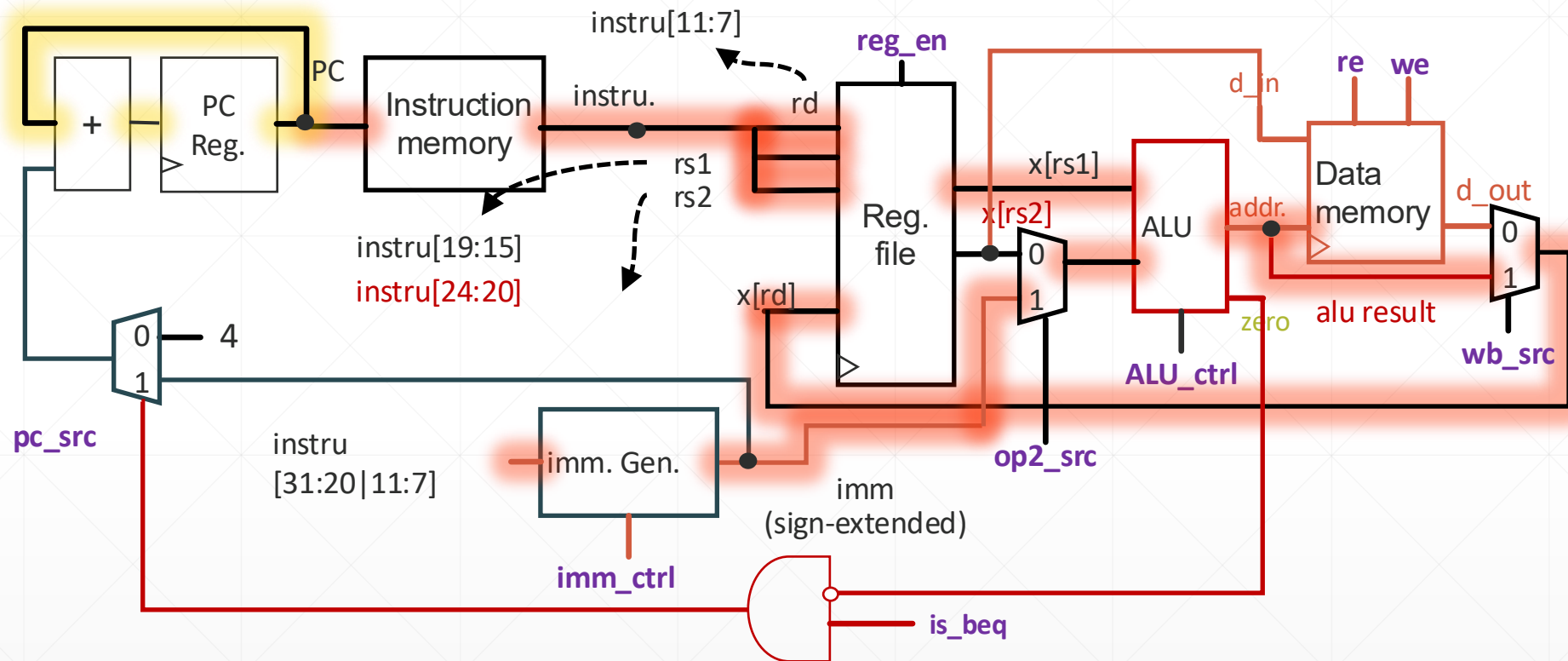
Assume the control signal is fast

$$t_{clk-to-q} + t_{add} + t_{setup}$$

$$\frac{t_{clk-to-q} + t_{Imem}}{t_{IF}} + \frac{t_{reg} + t_{mux}}{t_{DEC}} + \frac{t_{alu}}{t_{EX}} + \frac{t_{mux} + t_{setup}}{t_{WB}}$$

R-type datapath

Datapath timing analysis



$$t_{clk-to-q} + t_{add} + t_{setup}$$

$$t_{clk-to-q} + t_{Imem} + t_{reg} + t_{alu} + t_{mux} + t_{setup}$$

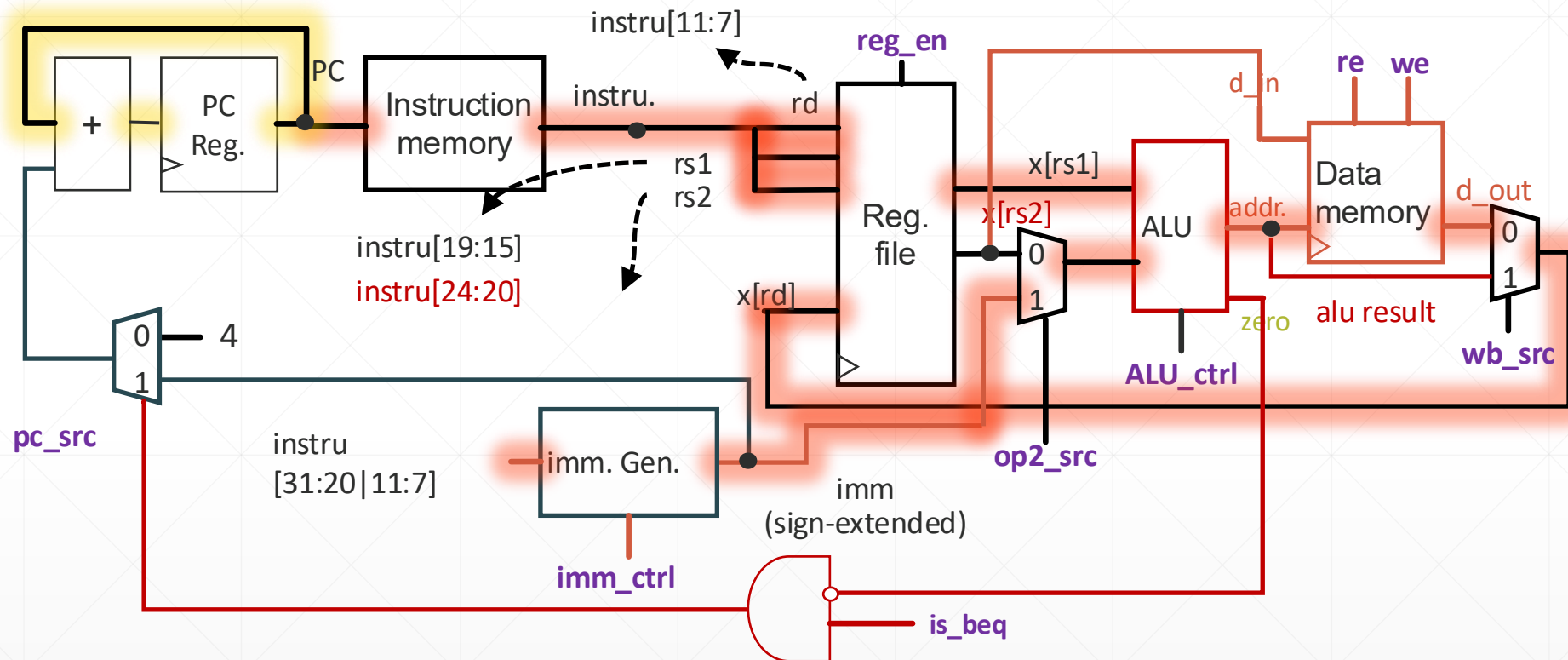
$$t_{clk-to-q} + t_{Imem} + t_{imm} + t_{mux} + t_{alu} + t_{mux} + t_{setup}$$

$$t_{IF} + t_{DEC} + t_{EX} + t_{WB}$$

} max

I-type arithmetic & logic datapath

Datapath timing analysis



$$t_{clk-to-q} + t_{add} + t_{setup}$$

$$t_{IF} + t_{DEC} + t_{EX} + t_{MEM} + t_{WB}$$

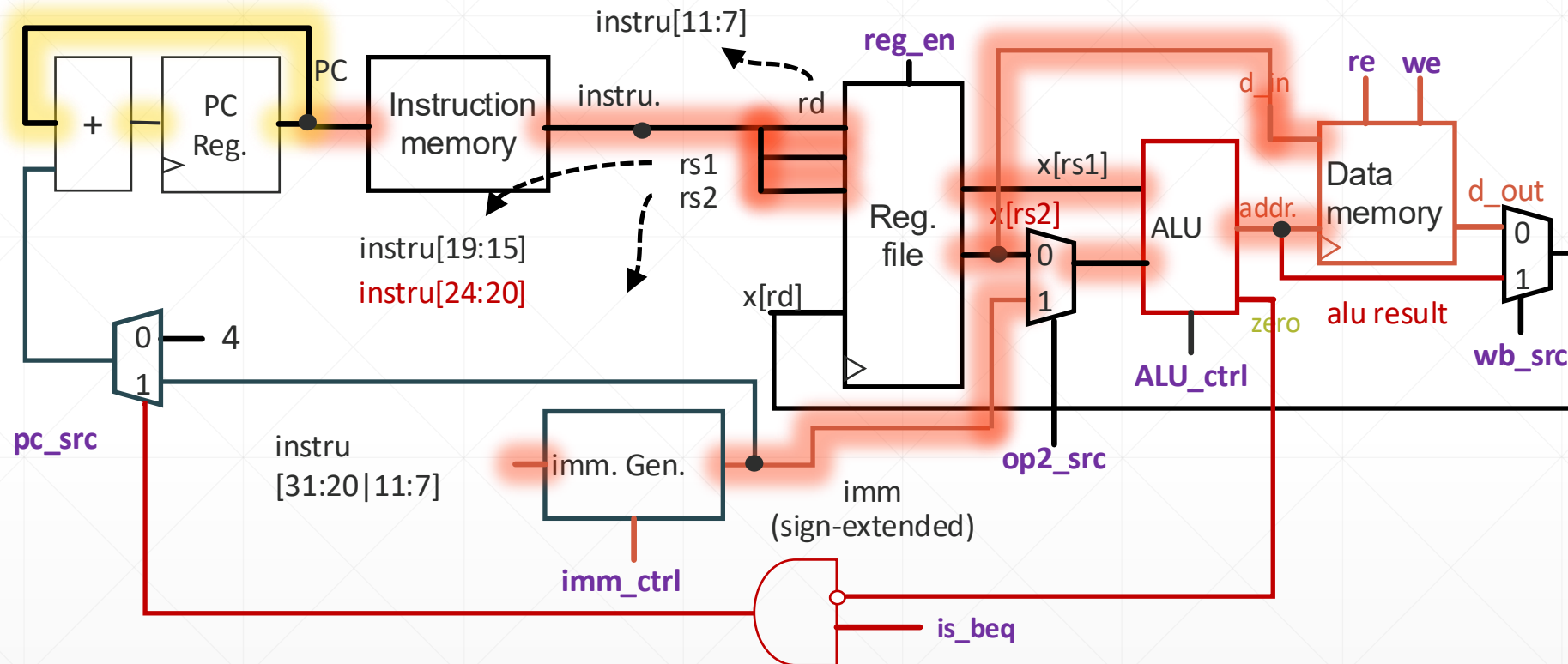
$$t_{clk-to-q} + t_{Imem} + t_{reg} + t_{alu} + t_{Dmem} + t_{mux} + t_{setup}$$

$$t_{clk-to-q} + t_{Imem} + t_{imm} + t_{mux} + t_{alu} + t_{Dmem} + t_{mux} + t_{setup}$$

} max

I-type load datapath

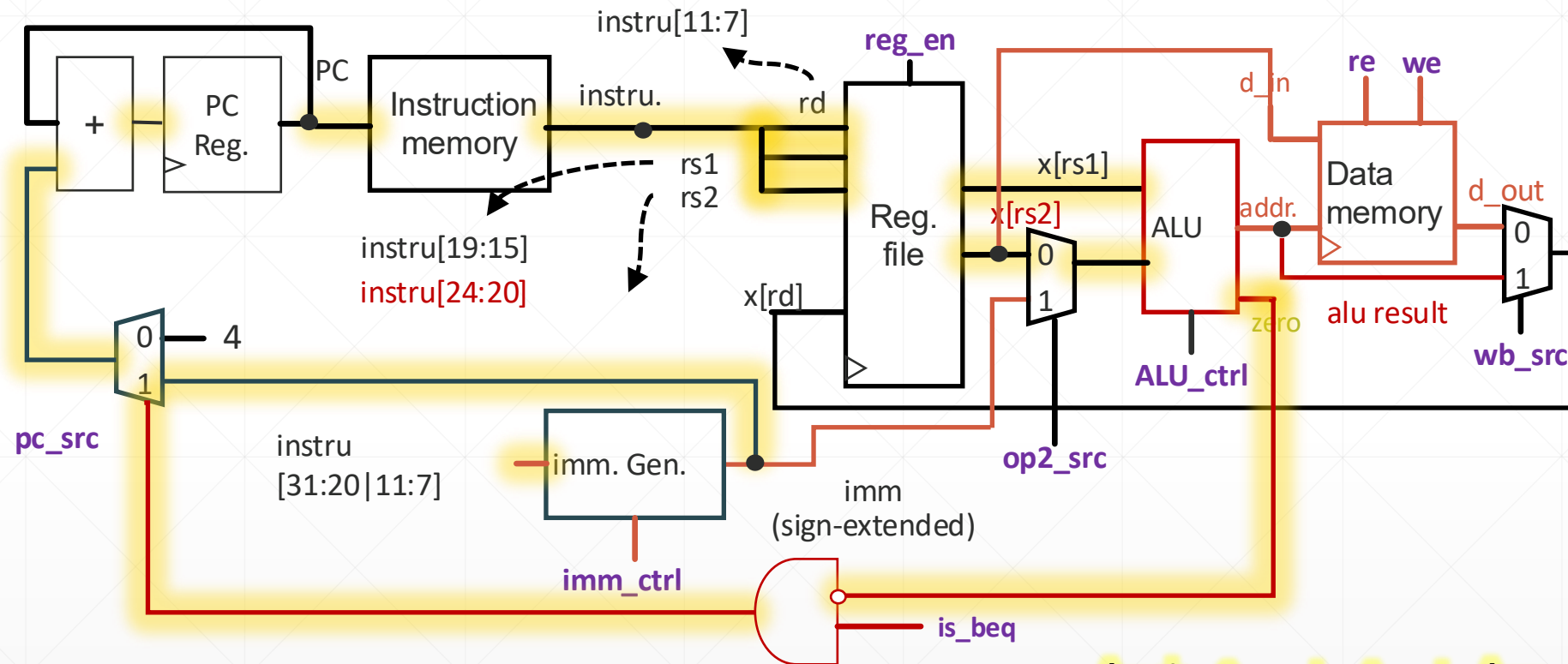
Datapath timing analysis



$$\begin{aligned}
 & t_{clk-to-q} + t_{add} + t_{setup} \\
 & t_{clk-to-q} + t_{Imem} + t_{reg} + t_{Dmem} \\
 & t_{clk-to-q} + t_{Imem} + t_{reg} + t_{alu} + t_{Dmem} \\
 & t_{clk-to-q} + t_{Imem} + t_{imm} + t_{mux} + t_{alu} + t_{Dmem}
 \end{aligned}
 \left. \vphantom{\begin{aligned} & t_{clk-to-q} + t_{add} + t_{setup} \\ & t_{clk-to-q} + t_{Imem} + t_{reg} + t_{Dmem} \\ & t_{clk-to-q} + t_{Imem} + t_{reg} + t_{alu} + t_{Dmem} \\ & t_{clk-to-q} + t_{Imem} + t_{imm} + t_{mux} + t_{alu} + t_{Dmem} \end{aligned}} \right\} \max \quad t_{IF} + t_{DEC} + t_{EX} + t_{MEM}$$

S-type datapath

Datapath timing analysis



$$t_{clk-to-q} + t_{Imem} + t_{reg} + t_{mux} + t_{alu} + t_{and} + t_{mux} + t_{add} + t_{setup}$$

$$t_{clk-to-q} + t_{Imem} + t_{imm} + t_{mux} + t_{add} + t_{setup}$$

$$t_{IF} + t_{DEC} + t_{EX} + t_{WB}$$

} max

B-type datapath

Summary

- We have built a single-cycle CPU
- It supports R-type, I-type arithmetic & logic and load (lw), S-type sw and beq
- Datapath and controller are built separately
- Different instruction activates different parts or steps/stages (IF/DEC/EXE/MEM/WB) of the datapath, thus has different delays. The longest delay (critical path) is used to estimate maximum frequency
- Nearly no CPU uses single-cycle design today.