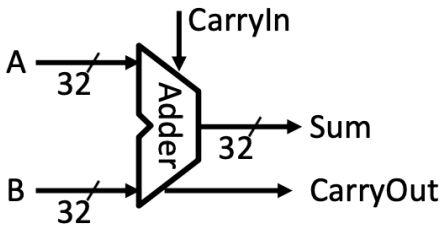
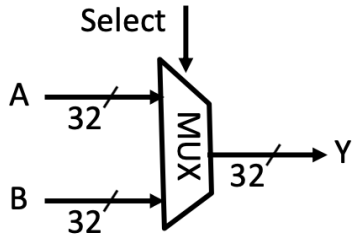
Discussion 7
Datapath

Yutong Wang
<wangyt32023@>
3/30/2025

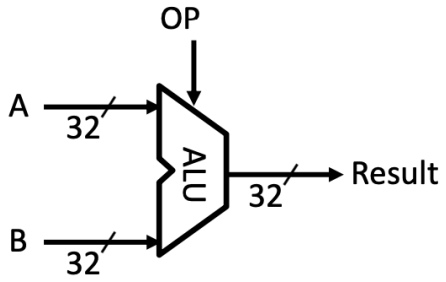
Datapath Components



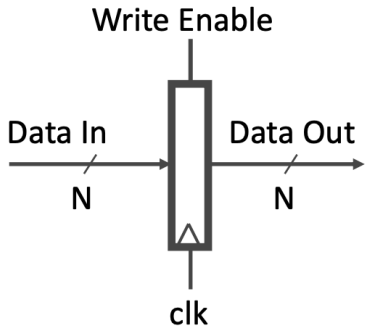
Adder



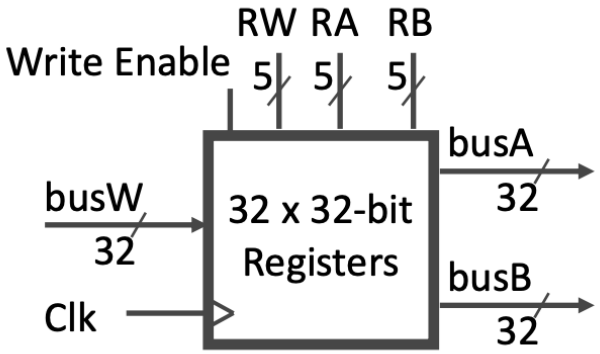
Multiplexer



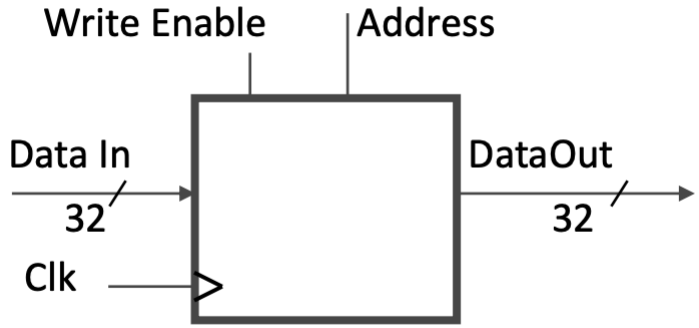
ALU



register



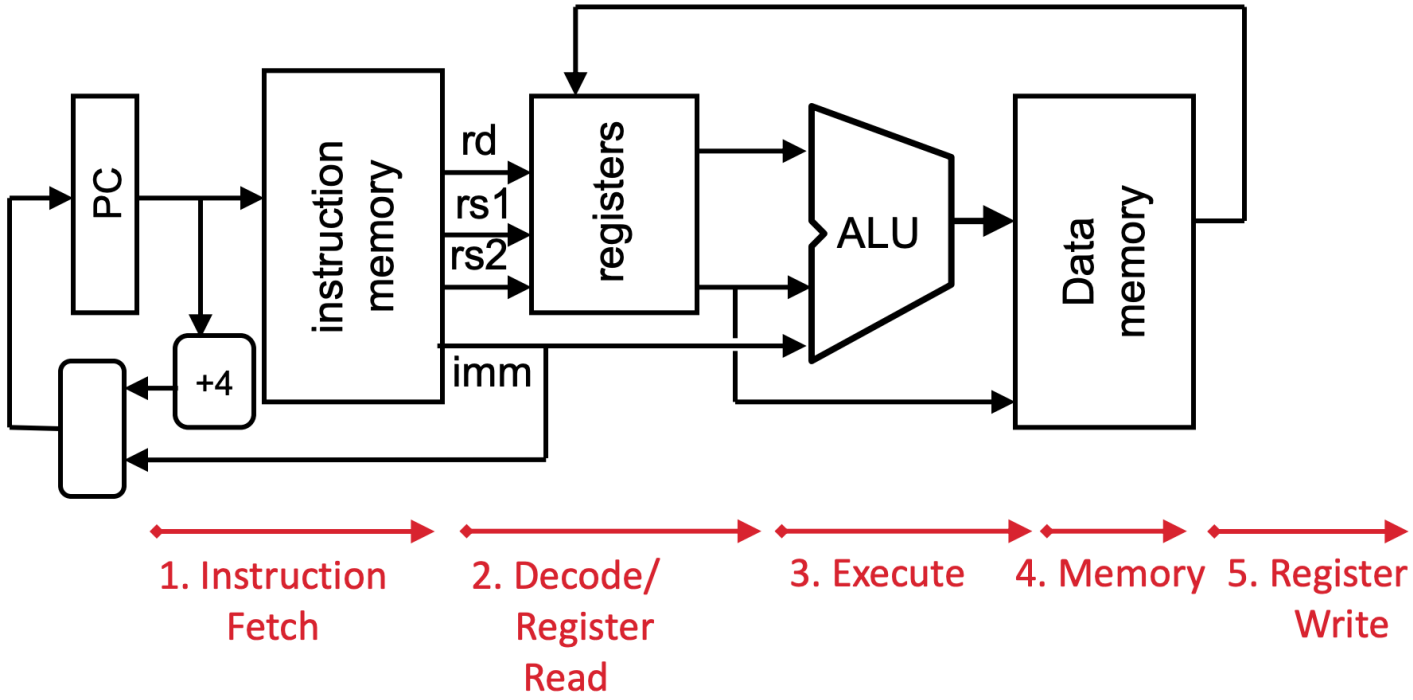
register file



memory

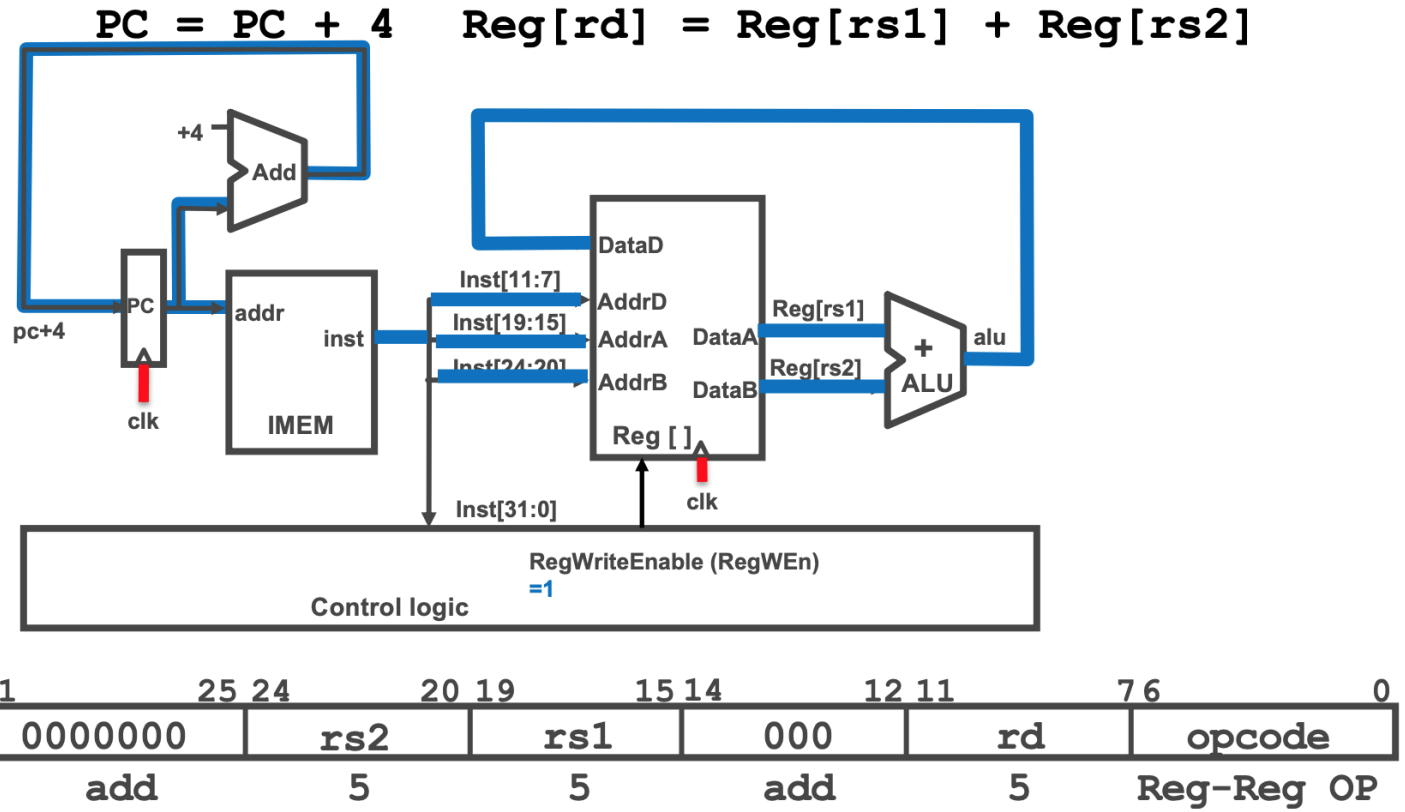
5 Stages of Datapath

- **IF**: **I**nstruction **F**etch
- **ID**: **I**nstruction **D**ecode
- **EX**: **Ex**ecute
- **MEM**: **Mem**ory
- **WB**: **W**rite **B**ack

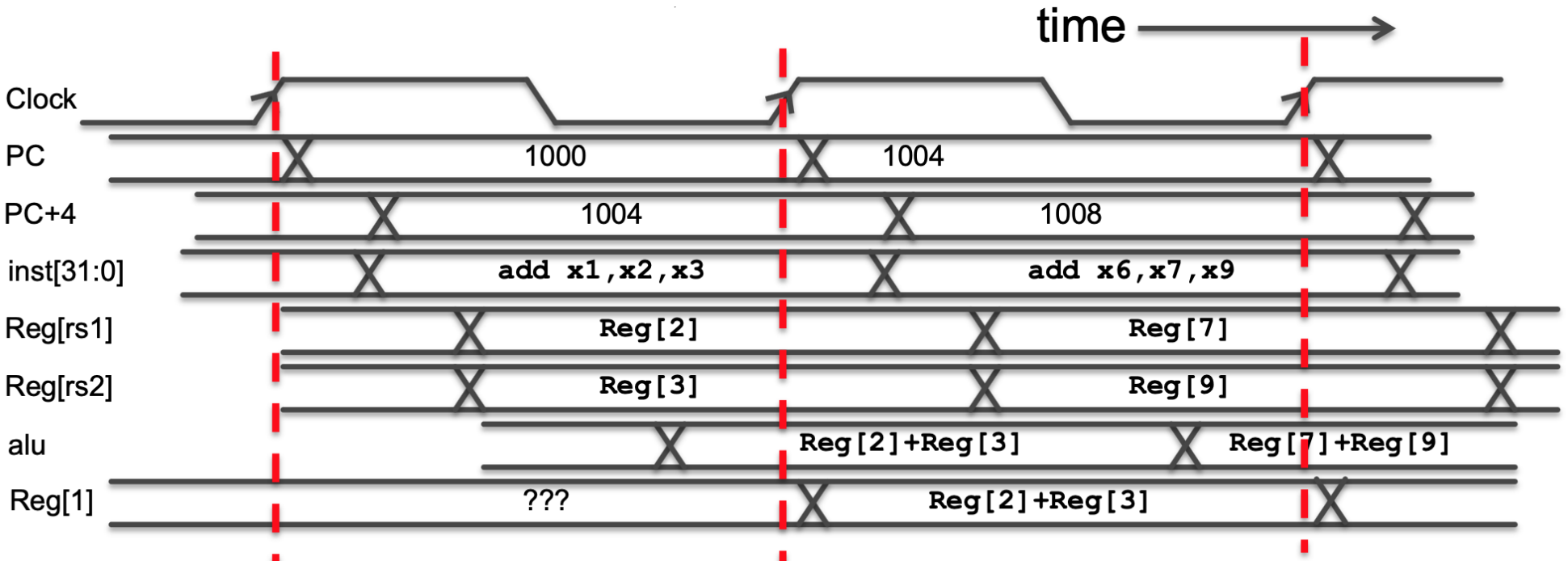


R-format: add

- **IF**
- **ID**
- **EX**: alu = R[rs1] + R[rs2]
- ~~**MEM**~~
- **WB**: R[rd] = alu
- PC = PC + 4



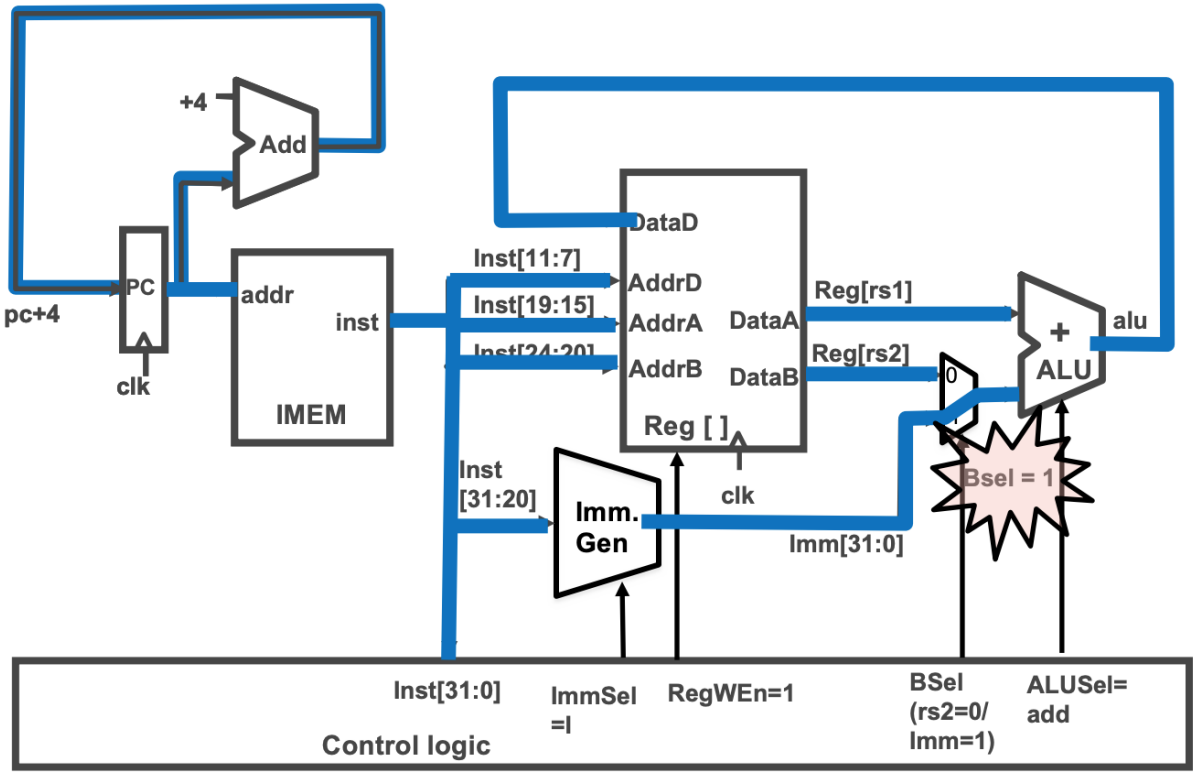
Time Fiagram



Attention! **WB** actually finishes after the next rising edge of Clock.

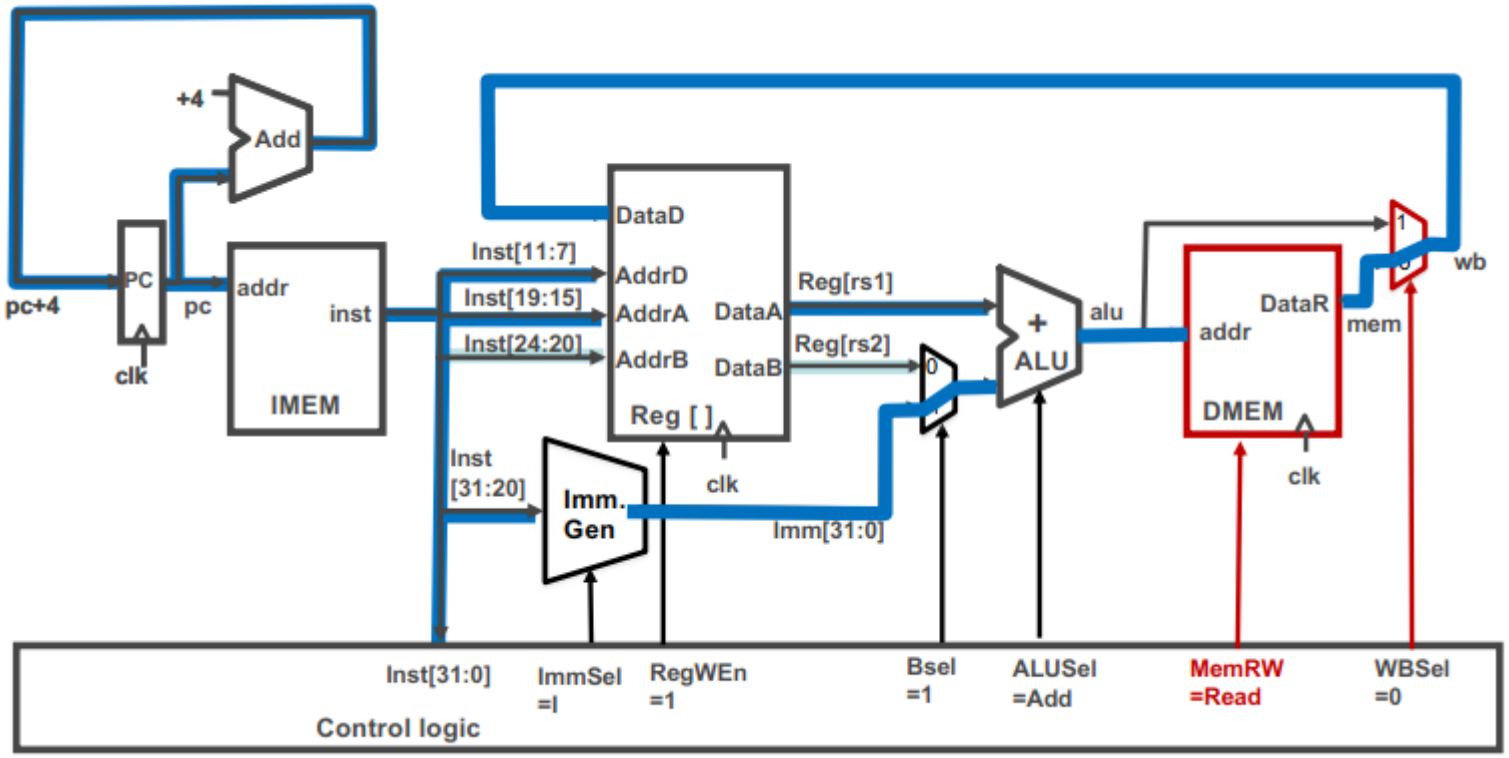
I-format: addi

- **IF**
- **ID**
- **EX**: alu = R[rs1] + imm
- ~~**MEM**~~
- **WB**: R[rd] = alu
- PC = PC + 4



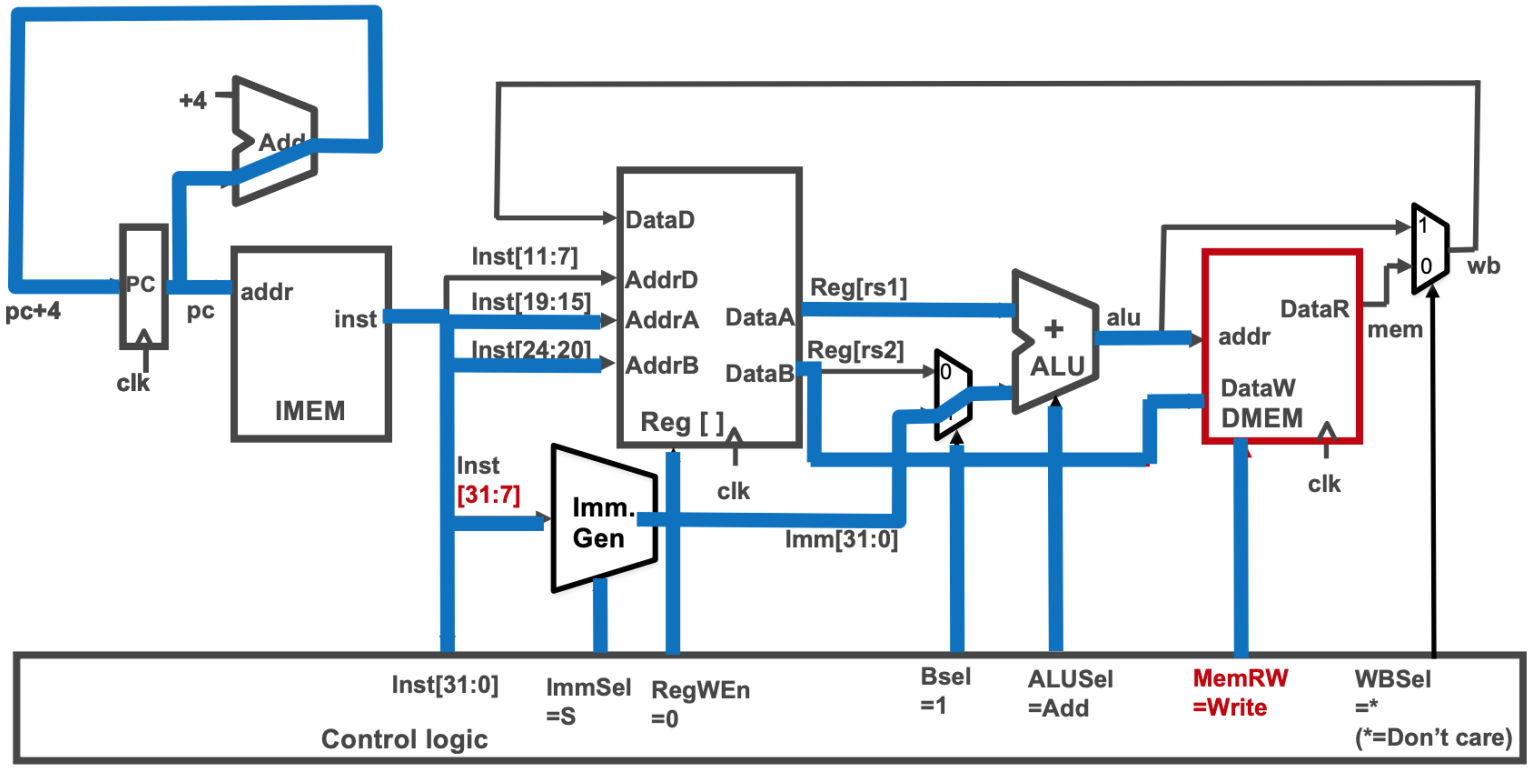
I-format: lw

- **IF**
- **ID**
- **EX**: alu = R[rs1] + imm
- **MEM**: mem = M[alu]
- **WB**: R[rd] = mem
- PC = PC + 4



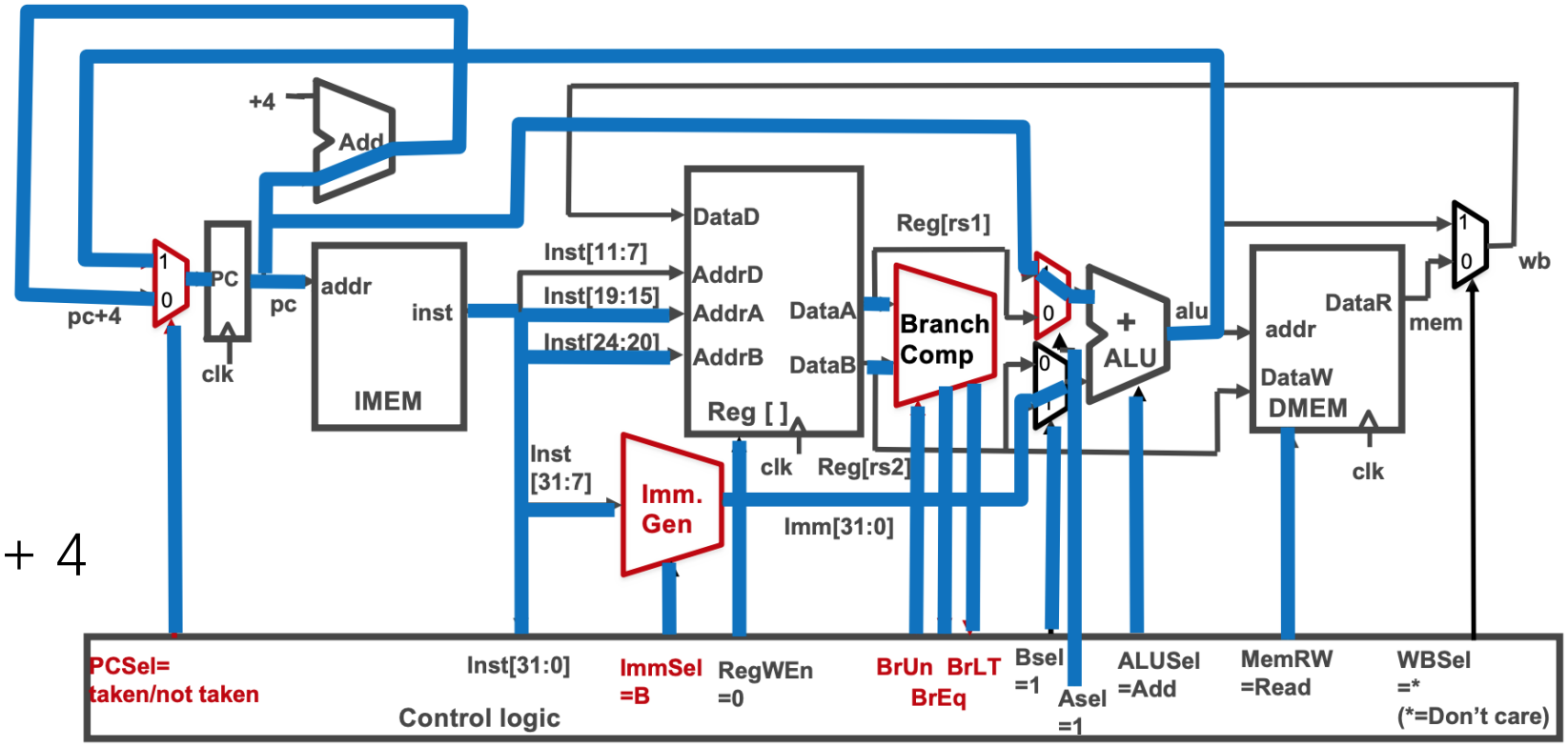
S-format: sw

- **IF**
- **ID**
- **EX**: alu = R[rs1] + imm
- **MEM**: M[alu] = R[rs2]
- ~~**WB**~~
- PC = PC + 4



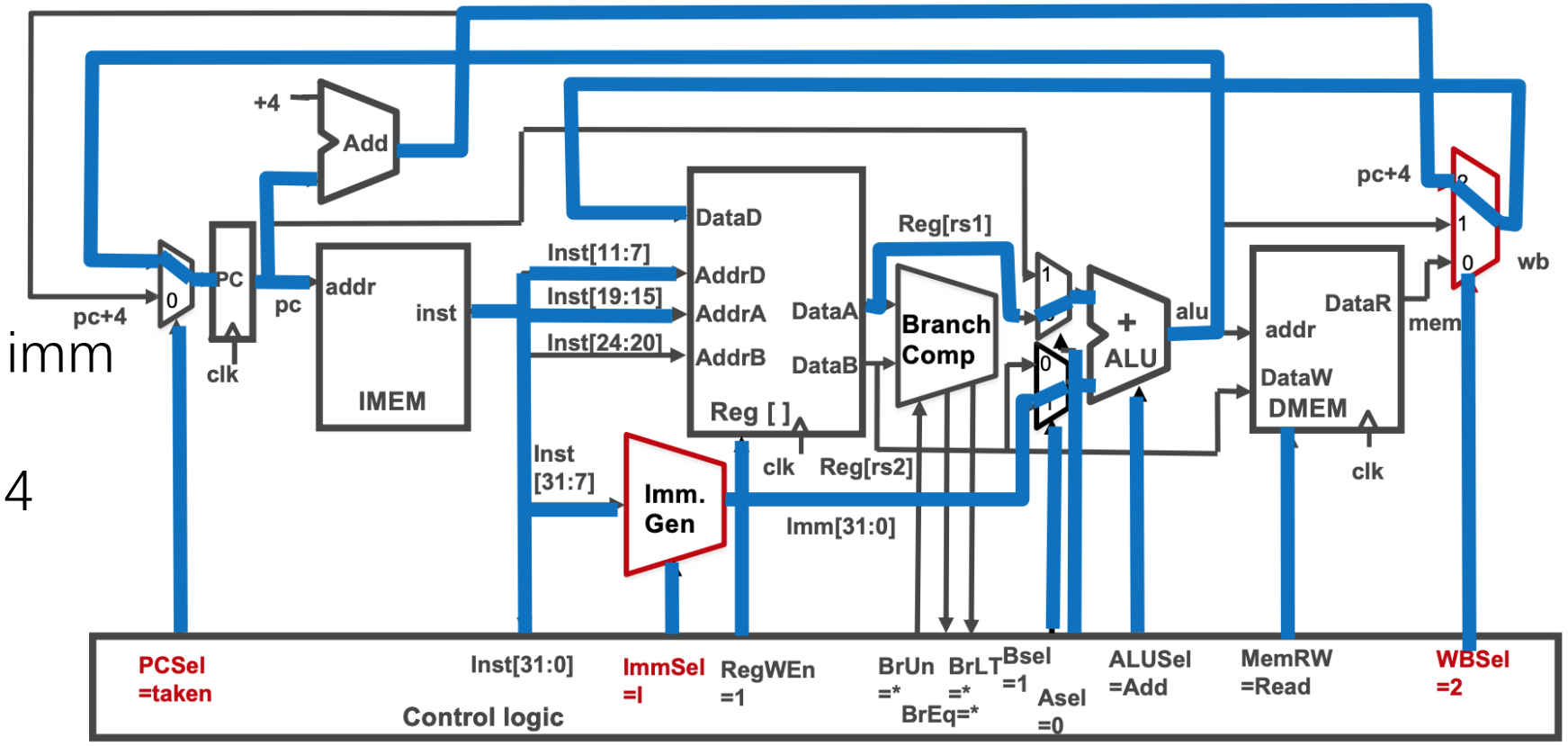
B-format

- **IF**
- **ID**
- **EX**: alu = PC + imm
- ~~**MEM**~~
- ~~**WB**~~
- PC = PCSel ? alu : PC + 4



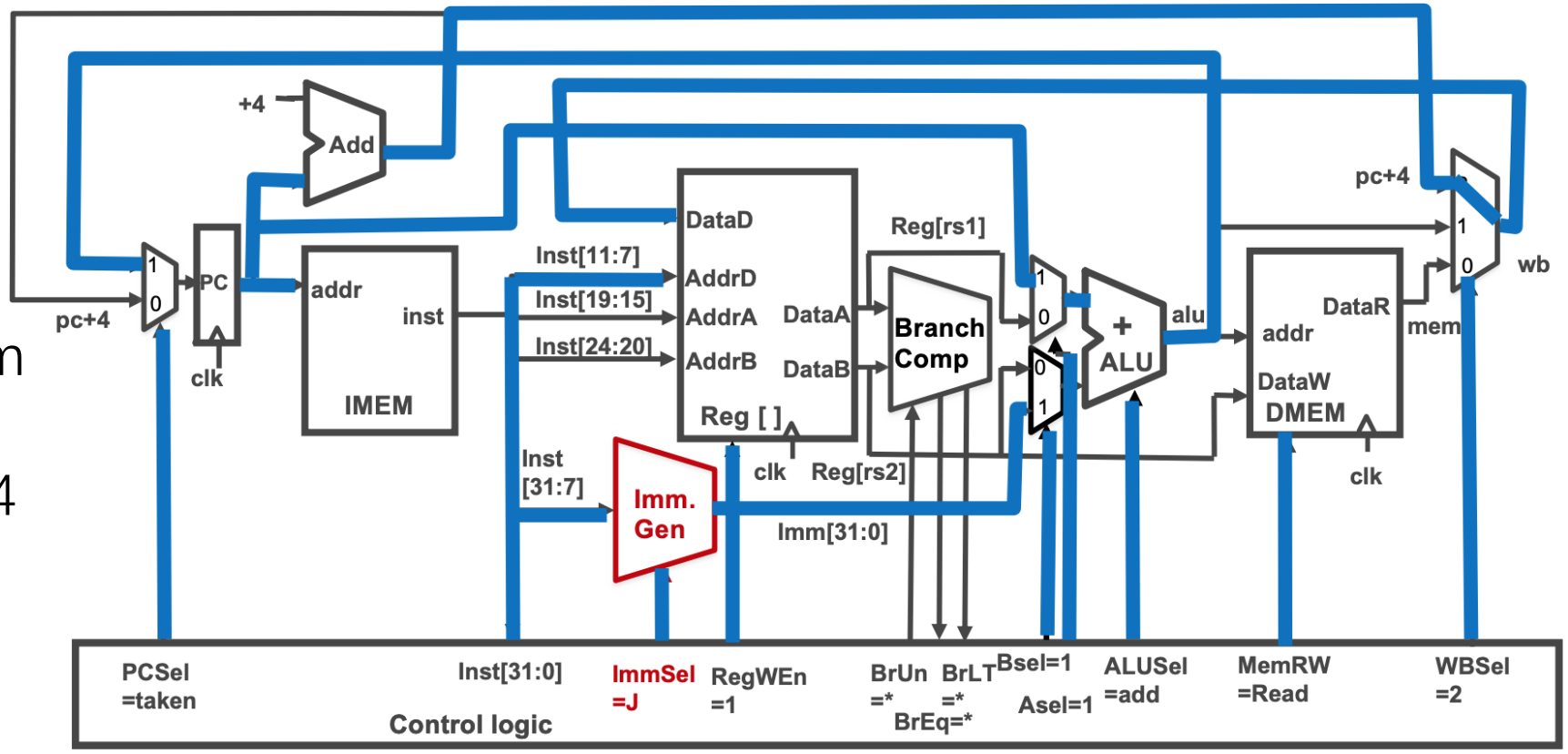
jalr

- **IF**
- **ID**
- **EX**: alu = R[rs1] + imm
- ~~**MEM**~~
- **WB**: R[rd] = PC + 4
- PC = alu



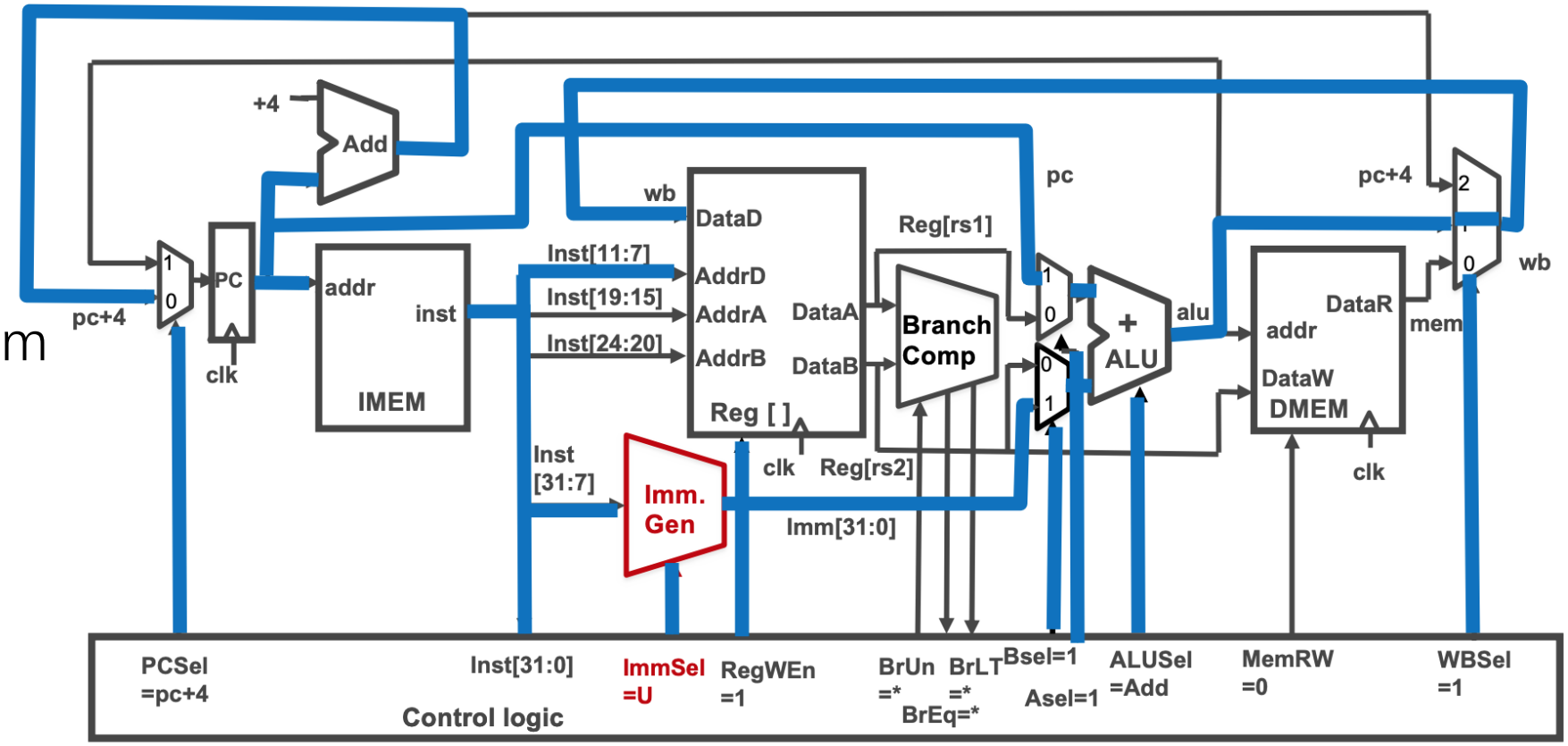
jal

- **IF**
- **ID**
- **EX**: alu = PC + imm
- ~~**MEM**~~
- **WB**: R[rd] = PC + 4
- PC = alu



U-format: auipc

- **IF**
- **ID**
- **EX**: alu = PC + imm
- ~~**MEM**~~
- **WB**: R[rd] = alu
- PC = PC + 4



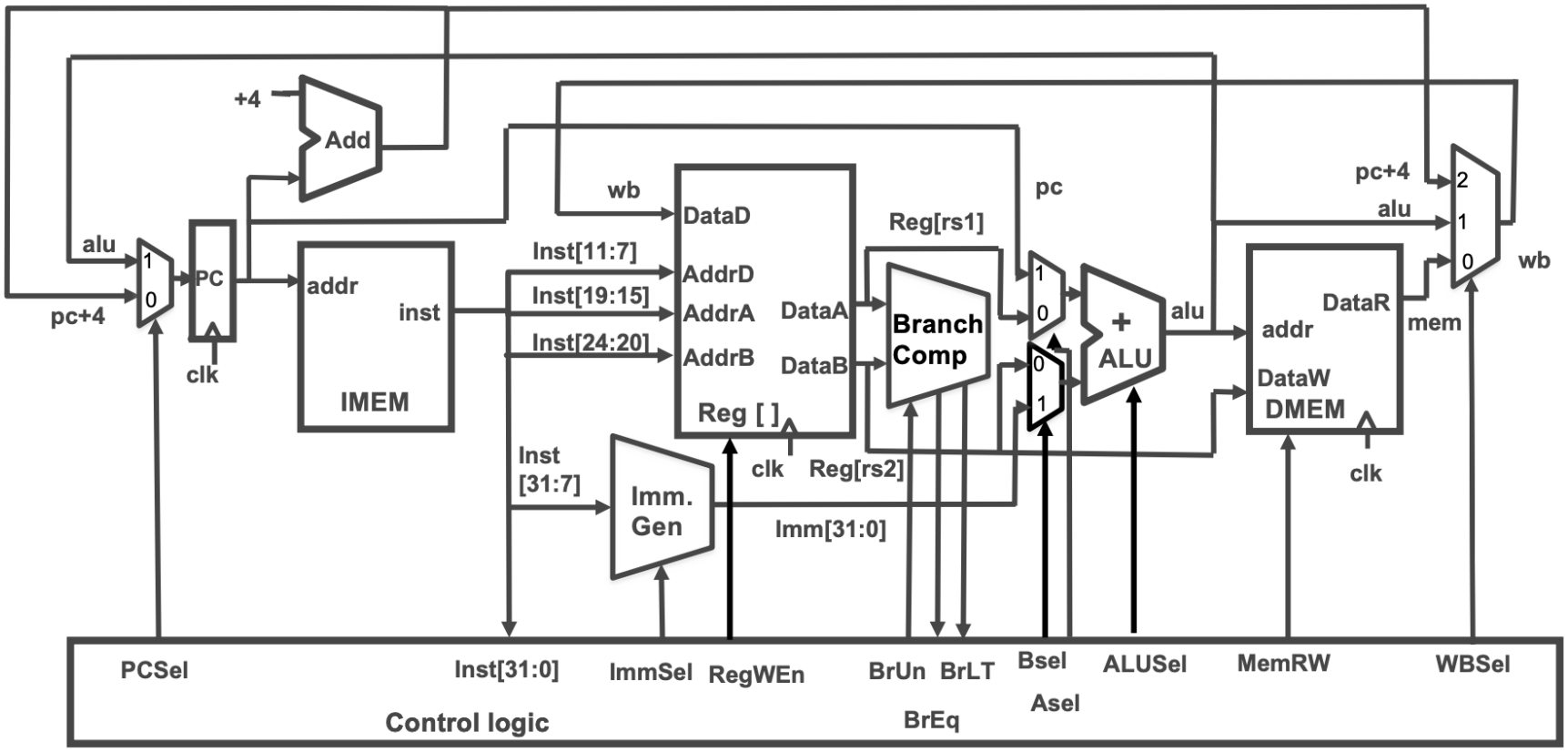
Example from CA 2022 final

For instruction `jalr ra`, select the correct value for the control logic.

1. PCSel:
A. alu
B. pc + 4

2. Asel:
A. 0
B. 1

3. WBsel:
A. pc + 4
B. alu
C. mem



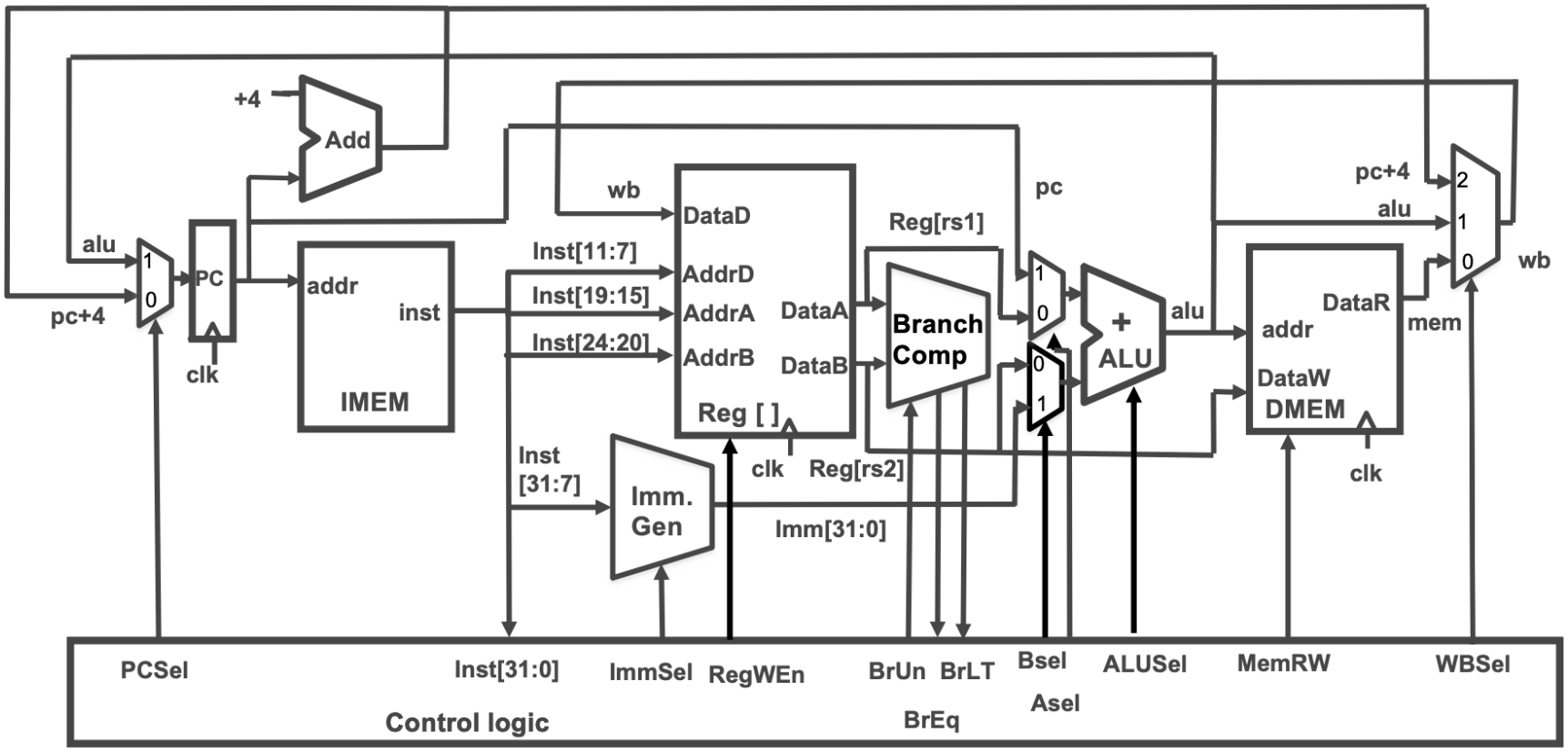
Example from CA 2022 final

For instruction `jalr ra`, select the correct value for the control logic.

1. PCSel:
A. alu
B. pc + 4

2. Asel:
A. 0
B. 1

3. WBsel:
A. pc + 4
B. alu
C. mem



Example from CA 2022 final

(True or False)

1. ( ) For instruction add, the value is written back to Reg as soon as it is computed by ALU.

2. ( ) Except Write Enable, all the input and output buses of register file are 32-bit.

3. ( ) For register file and memory, CLK is a factor ONLY during write operation.

4. ( ) Register file holds all the registers needed for instruction execution.

Example from CA 2022 final

(True or False)

1. (**F**) For instruction add, the value is written back to Reg as soon as it is computed by ALU.

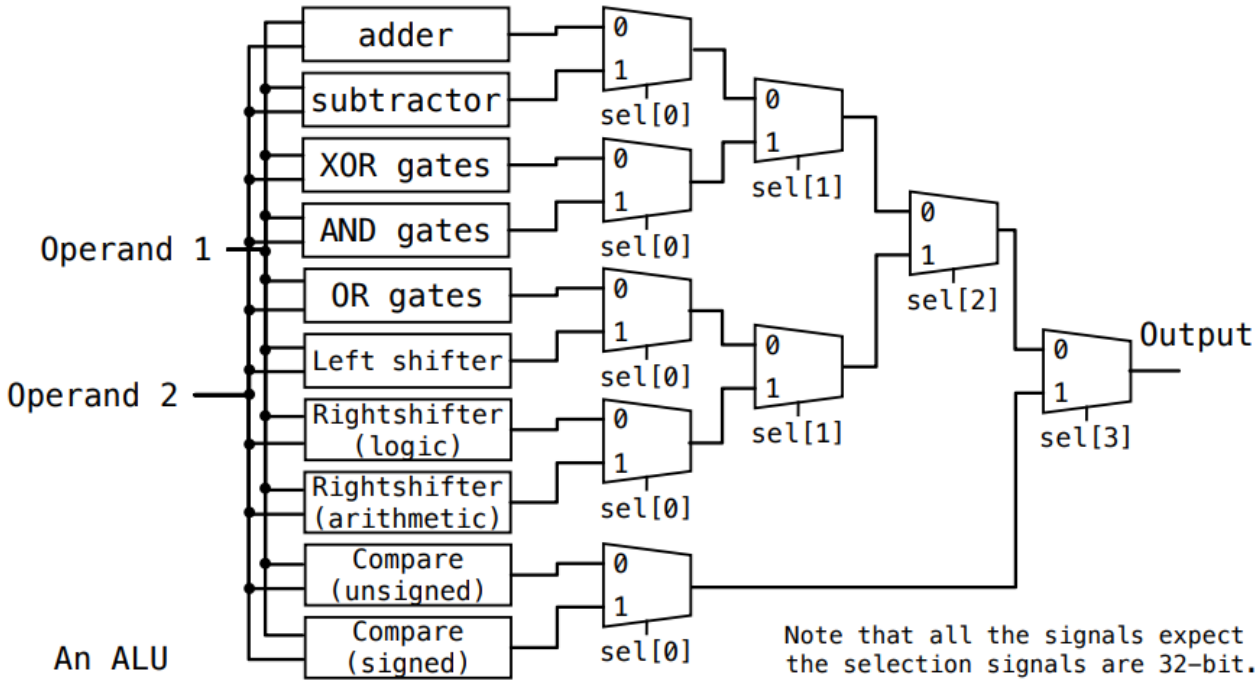
2. (**F**) Except Write Enable, all the input and output buses of register file are 32-bit.

3. (**T**) For register file and memory, CLK is a factor ONLY during write operation.

4. (**F**) Register file holds all the registers needed for instruction execution.

Midterm I 2024

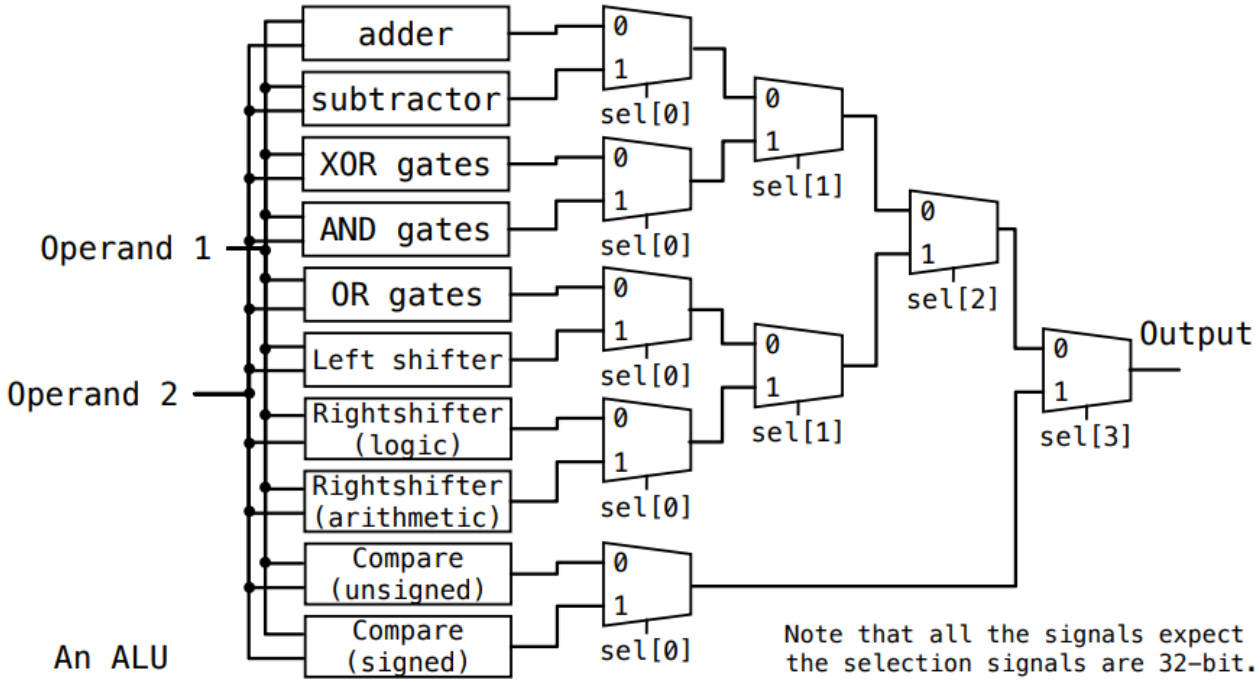
Datapath Below is a possible implementation of an ALU in a CPU that supports RV32I arithmetic and logic instructions. Assume that the rectangles are logic blocks that implement the corresponding functions described by the text. Please indicate the selection signals (**in binary**) of the multiplexer array so that the output of the corresponding logic block is selected when certain instructions are executed. Tips: An "X" can be used to represent that I do not care what this bit is. For example, "X100" means that it can either be "0100" or "1100".




```
addi x2, x2, -1        sel[3:0]=_

sub x2, x2, x0         sel[3:0]=_

sra x2, x2, x1         sel[3:0]=_

sltu x2, x2, x0        sel[3:0]=_
```

Midterm I 2024

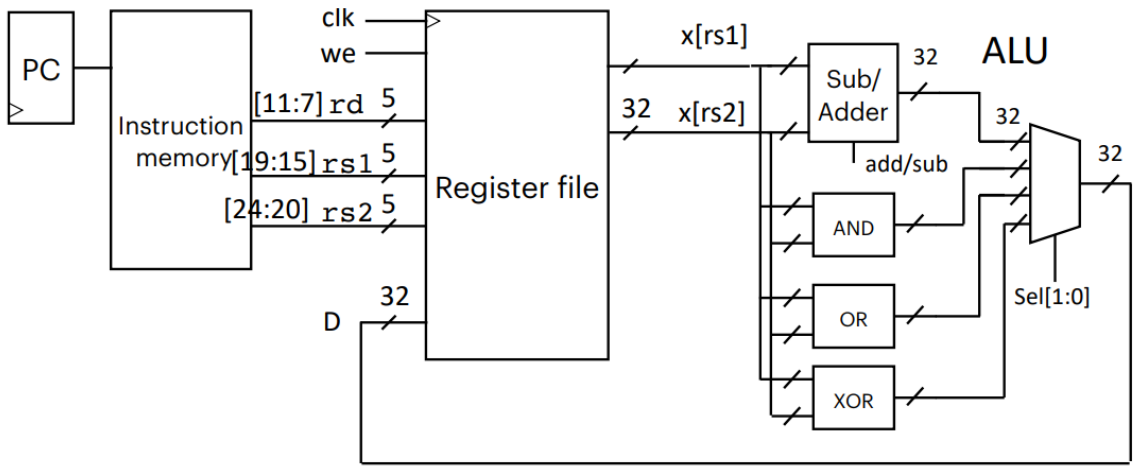
Datapath Below is a possible implementation of an ALU in a CPU that supports RV32I arithmetic and logic instructions. Assume that the rectangles are logic blocks that implement the corresponding functions described by the text. Please indicate the selection signals (**in binary**) of the multiplexer array so that the output of the corresponding logic block is selected when certain instructions are executed. Tips: An "X" can be used to represent that I do not care what this bit is. For example, "X100" means that it can either be "0100" or "1100".




```
addi x2, x2, -1        sel[3:0]=__0000__

sub x2, x2, x0         sel[3:0]=__0001__

sra x2, x2, x1         sel[3:0]=__0111__

sltu x2, x2, x0        sel[3:0]=__1xx0__
```

It is fine if "x" is replaced with 0 or 1 for the `sltu` instruction.

Midterm I 2023



We will estimate the maximum clock frequency step by step. By "delay", we refer to **propagation delays**, unless stated otherwise. The delay of each element is shown in the table below.

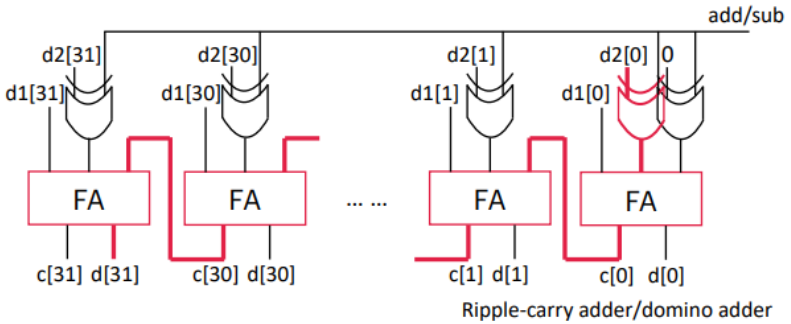
Circuit	2-input AND	2-input OR	2-input XOR	DFF clk-to-Q
Delay (ps)	10	15	50	20
Circuit	2-to-1 multiplexer	5-32 decoder	1-bit full adder	3-input AND gate
Delay (ps)	15	100	30	15

Midterm I 2023

(a) **Delay of the ALU [6 points]**

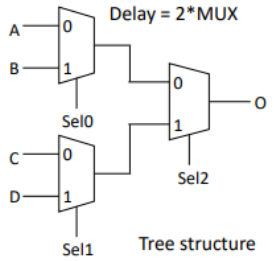
The ALU consists of functional units such as the adder/subtracter (circuit shown below) and different types of logic gates, and a 4-to-1 multiplexer built from the 2-to-1 multiplexer using tree structure. Calculate the delay of the adder/subtracter and then the maximum delay of the ALU.

Solution:



Ripple-carry adder/domino adder

add/sub delay $= XOR_{delay} + 32 * (\text{1-bit bull adder})_{delay} = 50 + 960 = 1010$ ps.



Tree structure

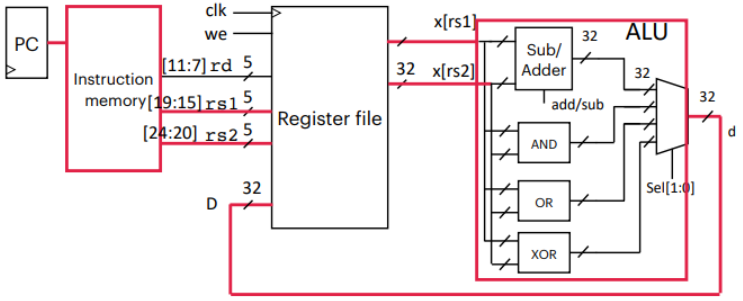
4-to-1 MUX delay = 2*2-to-1 MUX delay = 30 ps.

ALU max delay = 4-to-2 MUX delay + add/sub delay = 1040 ps.

Midterm I 2023

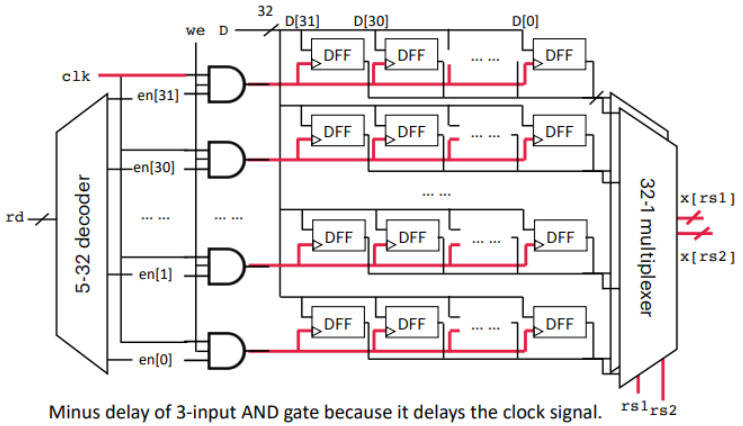
(b) **Delay of the datapath [9 points]**

The detailed circuit of the register file is shown below. Assume all the DFFs has a setup time of 50 ps. Indicate the elements of which the delay should be included to calculate the minimum clock cycle and the corresponding delay numbers. After that, calculate the maximum frequency of this datapath. Again, assume the 32-to-1 multiplexer is built from 2-to-1 multiplexer by the tree structure. Instruction memory delay is 80 ps.



Solution:

Critical path = clock-to-q (PC) + Imem delay + Reg. file delay + ALU delay + setup time (Reg. file).



Minus delay of 3-input AND gate because it delays the clock signal.

Reg. file delay in total = 32-1 MUX delay − 3-input AND delay
= 5 × 2-1 MUX delay − 3-input AND delay = 60 ps.

Minus the delay of the 3-input AND gate because it delays the clock signal, resulting in extra time for DFF setup in the register file.

Critical path = 20 + 80 + 60 + 1040 + 50 = 1250 ps.

Max frequency = 1/(Critical path delay) = 800 MHz.

(c) Use the "funct3" field of the R-type instructions to generate the multiplexer select signal for the ALU. Assume the multiplexer selects add/sub, AND, OR and XOR results when the select signal is 00, 01, 10, 11, respectively.

Midterm I 2023

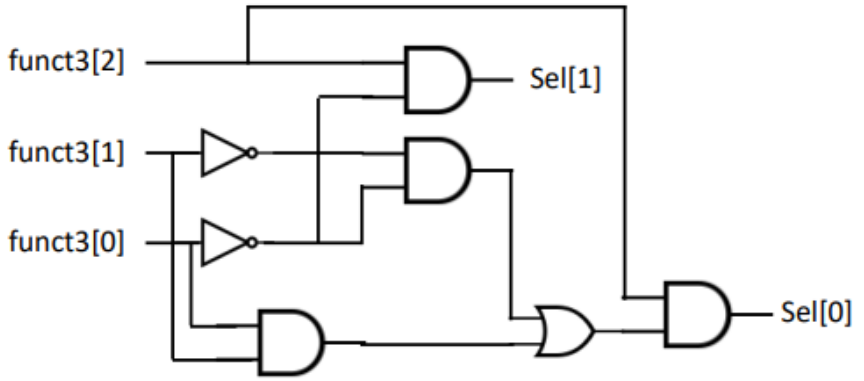
(c) Use the "funct3" field of the R-type instructions to generate the multiplexer select signal for the ALU. Assume the multiplexer selects add/sub, AND, OR and XOR results when the select signal is 00, 01, 10, 11, respectively.

Solution:

	funct3[2]	funct3[2]	funct3[2]	Sel[1]	Sel[0]
add/sub	0	0	0	0	0
AND	1	1	1	0	1
OR	1	1	0	1	0
XOR	1	0	0	1	1

$$Sel[1] = funct3[2] \cdot \overline{funct3[0]}$$

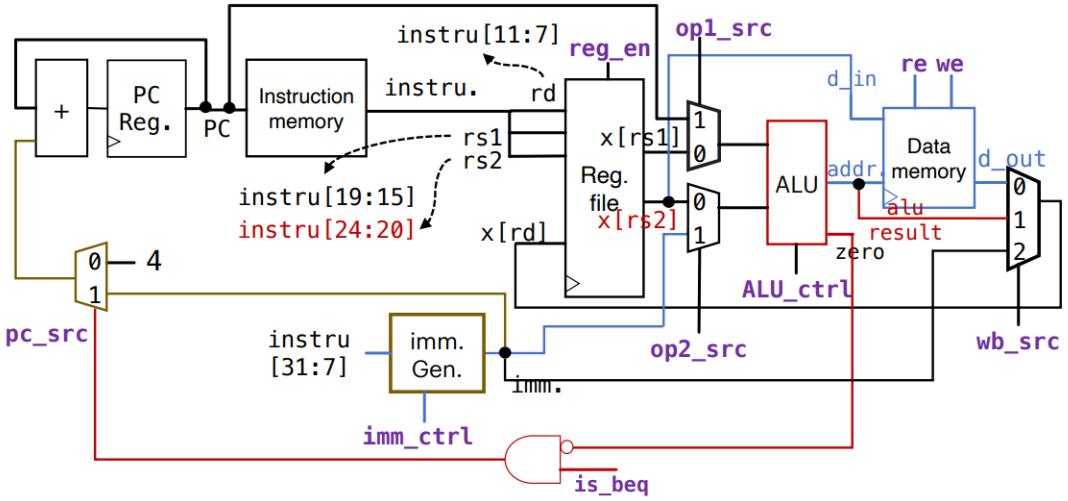
$$Sel[0] = funct3[2] \cdot (funct3[1] \cdot funct3[0] + \overline{funct3[1]} \cdot \overline{funct3[0]})$$



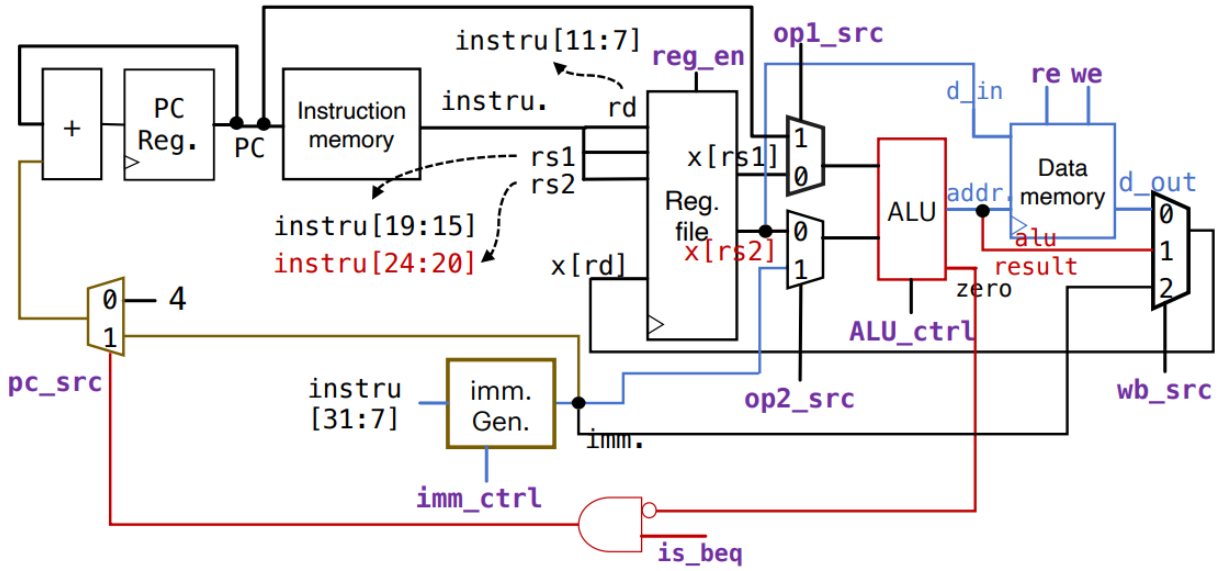
Midterm II 2024

6. **Datapath.** We would like to add supports for more RV32I instructions by modifying the single-cycle CPU datapath we covered in the lectures. It originally supports R- and I-type arithmetic and logic operations, **`lw`**, **`sw`** and **`beq`** instructions. The circuit diagram after modification is shown below. The modified datapath also supports U-type instructions. We first add a multiplexer to select the input of the ALU between **`PC`** and **`x[rs1]`** using **`op1_src`** signal. Besides, an extra option is added to the rightmost write-back multiplexer using a select signal of "2". This option selects the shifted immediate (upper 20 bits indicated by the **`lui`** or **`auipc`** instruction, lower 12 bits are 0) as the value written to the register file. Assume that the shifted immediate is provided by "imm. Gen." block in the circuit diagram.

   "**`reg_en`**" denotes register file write-enable signal; "**`re`**" stands for data memory read-enable signal; "**`we`**" represents data memory write-enable signal. We use logic '1' to enable and logic '0' to disable.



Midterm II 2024



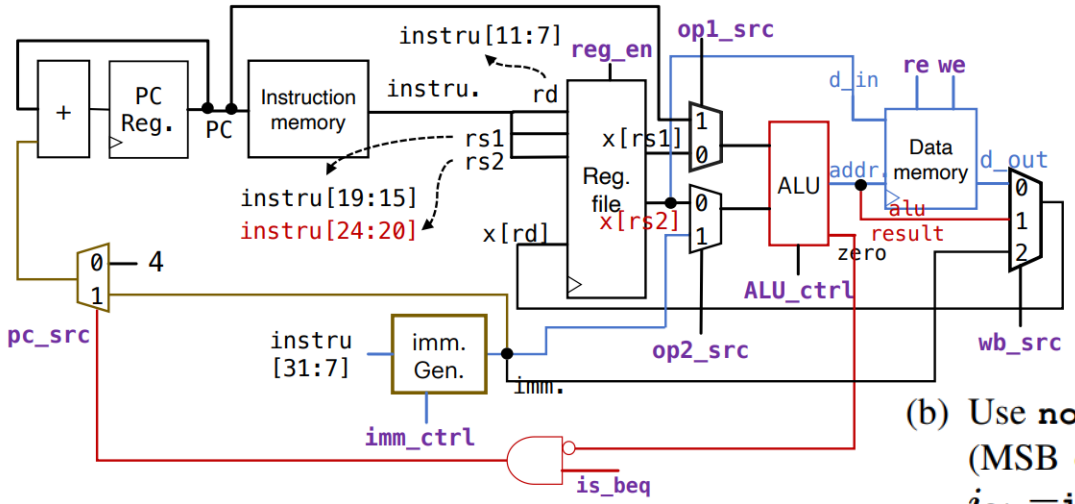
(a) Please indicate the values of the control signals in the table below when executing **lui**, **auipc** and **add** instruction. Use 'x' to indicate either '0' or '1' works.

Solution:

	reg_en	op1_src	op2_src	re	we	wb_src
lui	1	x	x	0	0	2
auipc	1	1	1	0	0	1
add	1	0	0	0	0	1

For **re**, it is also acceptable to fill in 'x', but '1' receives half the mark, since we have no reason to read explicitly; for **lui** instruction, either '0' or '1' receives half the mark for control signals **op1_src** and **op2_src**.

Midterm II 2024



(b) Use **not**, **and** and **or** logics **only** to design logics that produce **op1_src**, **wb_src[1]** (MSB of **wb_src**) signals. We use i_k to represent the kth bit in the instruction, i.e., i_{31} =**instruction[31]**, i_{30} =**instruction[30]**, ..., i_0 =**instruction[0]**. Note that we still ensure that R- and I-type arithmetic and logic operations, **lw**, **sw** and **beq** work properly. **Write down the logic expressions to complete your logic design.**

Solution:

op1_src = $\bar{i_6}\bar{i_5}i_4\bar{i_3}i_2i_1i_0$.

Only for **auipc**, **op1_src** is 1. Only **opcode** (0010111) is used to identify **auipc**.

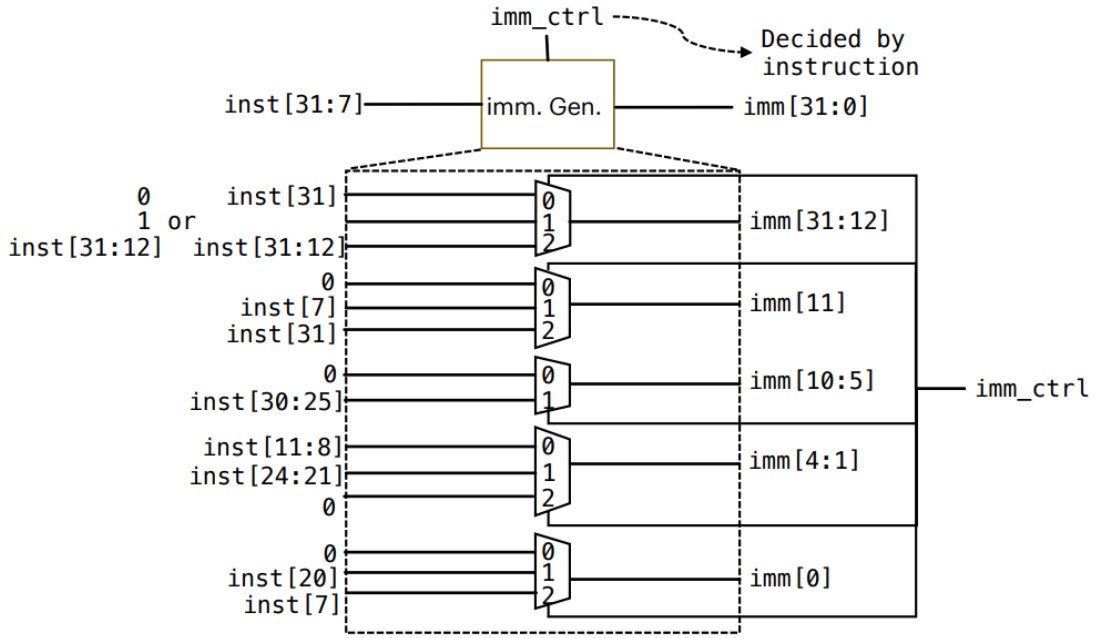
wb_src[1] = $\bar{i_6}i_5i_4\bar{i_3}i_2i_1i_0$.

Only for **lui**, **wb_src[1]** is 1. Only **opcode** (0110111) is used to identify **lui**. Many of you include the logic to identify `beq`. However, as per the datapath, `PC+imm` is calculated by the leftmost adder instead of the ALU, while the ALU is used to compare `x[rs1]` and `x[rs2]`. So it is not appropriate to include the logic to identify `beq`.

Midterm II 2024

(c) Next, we modify the "imm. Gen." block to support the production of the shifted immediate for **`lui`** and **`auipc`** instructions. Please connect the input signals properly. Note that it should still support the production of the immediate for I-type arithmetic and logic operations, **`lw`**, **`sw`** and **`beq`** instructions. Disregard the selection signals for the multiplexers. Hint: You can use **`inst[m:n]`** to indicate the **`m`**th to **`n`**th bits of the instruction. The signal connections for generating **`imm[4:1]`** is given as a reference. Modify it if you found errors in the connections. Add inputs if the input ports for the multiplexer is insufficient; or leave it blank if the input ports are more than enough.

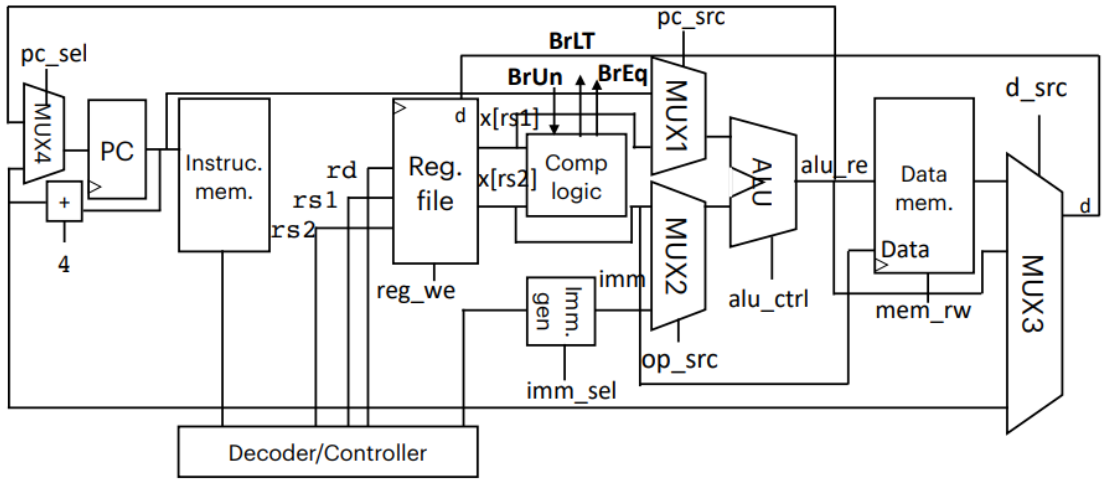
Solution:



To be more precise, the '1' for `imm[31:12]` should be `FFFFF`, i.e., 20 '1's.

Midterm II 2023

3. **Single-cycle CPU datapath [12 points]**
Below is the single-cycle CPU datapath we have learnt in the lectures.



(1) Which of the following type(s) of instructions experience(s) the longest propagation delay?_____ [2 points]

A. Conditional branch.
B. Load.
C. I-type arithmetic and logic.
D. Store.

(2) For an immediate (I-) type arithmetic and logic instruction, what is the total propagation delay? _____ (The subscript indicates the component/type of the delay.) [2 points]

A. $t_{\text{clock-to-q}} + t_{\text{MUX}} + t_{\text{PC}} + t_{\text{instruction memory}} + t_{\text{decoder/controller}} + \max\{t_{\text{Reg. file}}, t_{\text{immediate generation}}\} + t_{\text{MUX}} + t_{\text{data memory}} + t_{\text{MUX}} + t_{\text{setup time}}$.

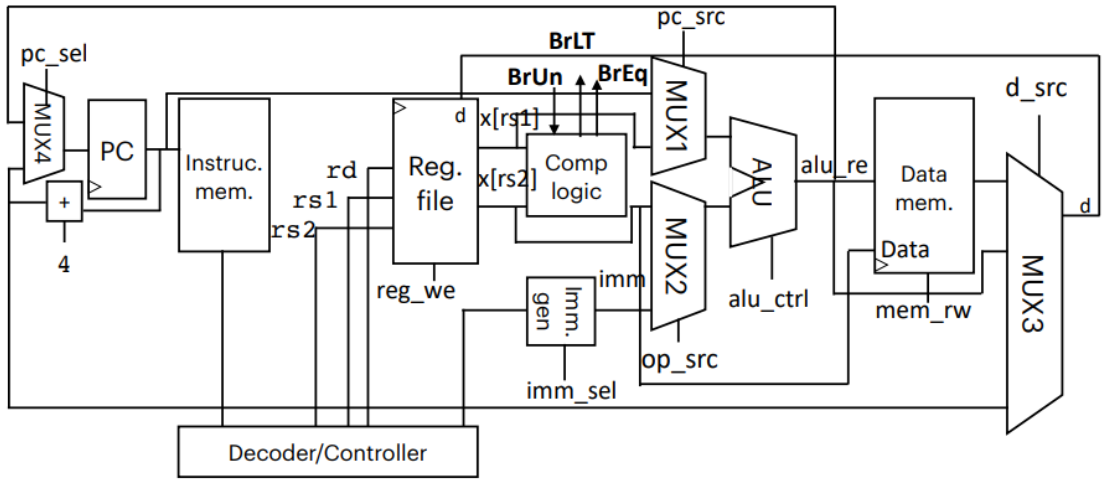
B. $t_{\text{clock-to-q}} + t_{\text{instruction memory}} + t_{\text{decoder/controller}} + \max\{t_{\text{Reg. file}}, t_{\text{immediate generation}}\} + t_{\text{MUX}} + t_{\text{ALU}} + t_{\text{MUX}} + t_{\text{setup time}}$.

C. $t_{\text{clock-to-q}} + t_{\text{MUX}} + t_{\text{adder}} + t_{\text{PC}} + t_{\text{setup time}}$.

D. None of the above.

Midterm II 2023

3. **Single-cycle CPU datapath [12 points]**
Below is the single-cycle CPU datapath we have learnt in the lectures.



(1) Which of the following type(s) of instructions experience(s) the longest propagation delay?_____ [2 points]

A. Conditional branch.
B. Load.
C. I-type arithmetic and logic.
D. Store.

(3) For an R-type instruction, select from below which piece of data is selected for multiplexers MUX1, MUX2, MUX3 and MUX4? What about U-type `auipc` instruction? [8 points]

R-type instructions:	U-type `auipc`:
MUX1_A__	MUX1__B__
MUX2_A__	MUX2__D__
MUX3_F_	MUX3__F__
MUX4__C__	MUX4__C__

A. Data from the register file.
B. Data from the PC register (value of `PC`).
C. Data from the PC register (value of `PC+4`).
D. Data of the immediate.
E. Data from the data memory.
F. Data from the ALU calculation.