COMPUTER ORGANIZATION AND ARCHITECTURE (IT 2202)

Processor Design

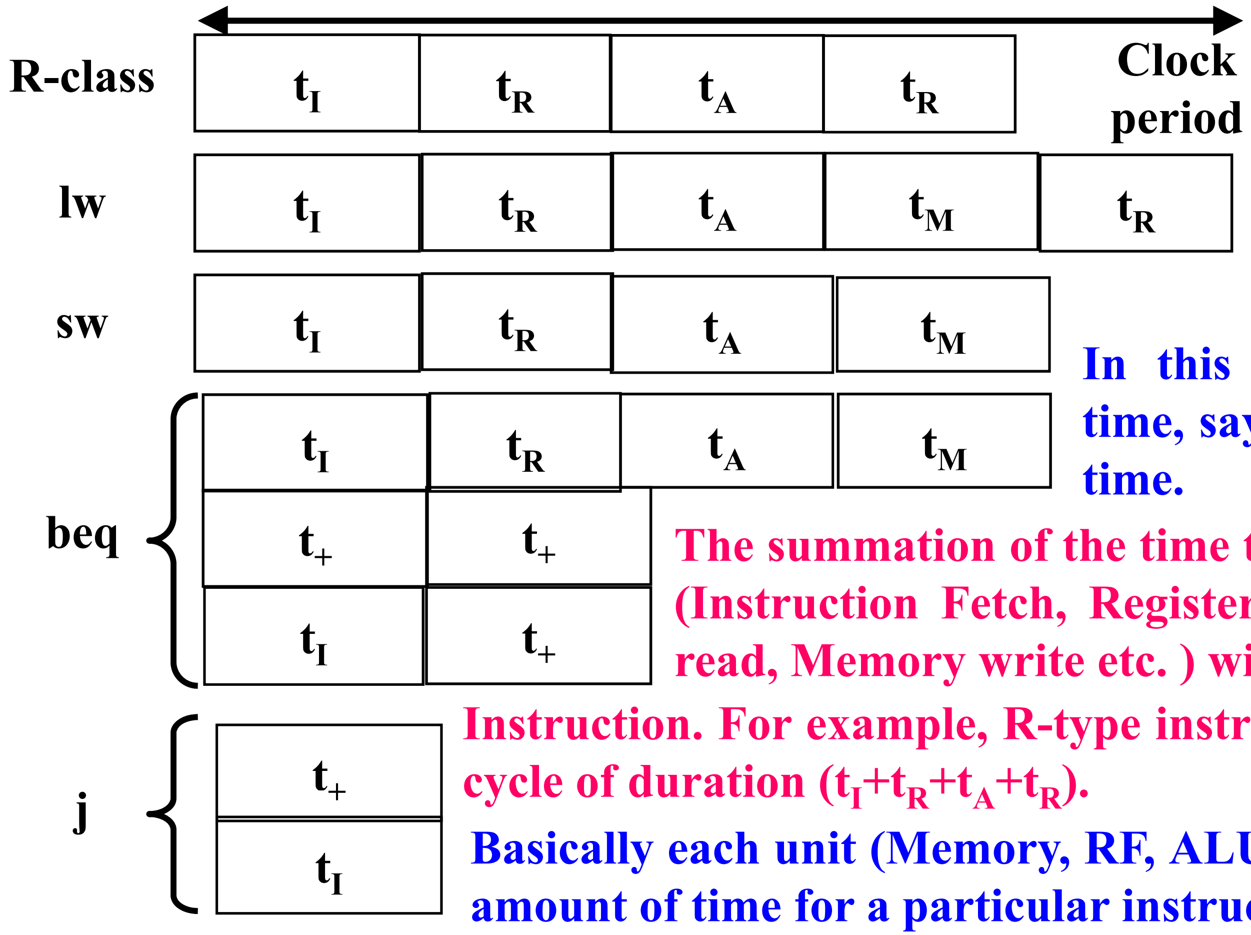
Analyzing performance

The typical delay amount for the different components has been assumed as follows…

	Component wise Delay
• Register	0
• Adder	t_+
• ALU	t_A
• Multiplexer	0
• Register file	t_R
• Program memory	t_I
• Data memory	t_M
• Bit manipulation components	0

Delay in the critical path determines the clock time for the system

Clock Period in Single Cycle Design



In this design, all the activities are performed in continuous time i.e., each instruction is executed in single cycle.

In this cycle, each activity takes a time, say, Instruction fetch will take t_I time.

The summation of the time taken by the different activists (Instruction Fetch, Register File Read, Write, Memory read, Memory write etc.) will form the execution time of Instruction. For example, R-type instruction will be executed in a cycle of duration (t_I+t_R+t_A+t_R).

Basically each unit (Memory, RF, ALU, DM etc) consumes a certain amount of time for a particular instruction. Summation of the time taken by each unit for a particular instruction is the execution time of that Instruction. Instruction that is taking maximum time among the instruction determines the cycle time of the processor. In this design, lw determines the clock time of the processor because the execution time for lw is maximum ((t_I+t_R+t_A+t_M+t_R).

Clock cycle time for single cycle design is the execution time of lw as lw takes maximum time [(t_I+t_R+t_A+t_M+t_R)] to execute.

Problems with Single Cycle Design

- Slowest instruction pulls down the clock frequency
- Resource utilization is poor
- There are some instructions which are impossible to be implemented in this manner

Solution

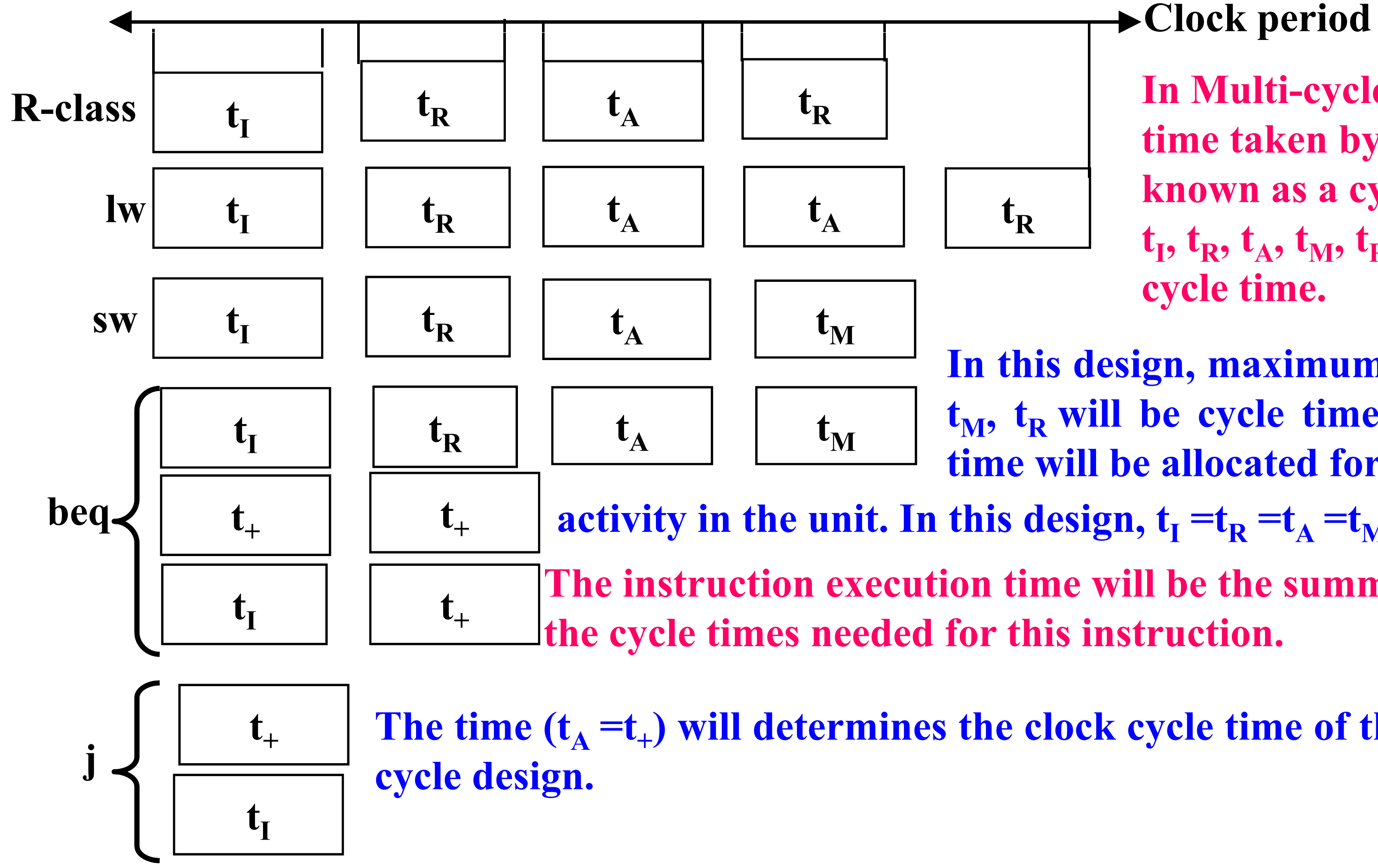
- Processor Design with Multi-cycle Approach

Multi-Cycle Design

- **In single cycle design, each instruction is executed in a single cycle. Basically, all the steps like Instruction memory, Register File, ALU, data memory will operates serially in a single step.**

- **In Multi-cycle design, the Instruction operations are divided into steps using step registers. All the steps require equal time step.**

Clock Period in Multi-Cycle Design



In Multi-cycle design, the time taken by a unit is known as a cycle. Each of t_I, t_R, t_A, t_M, t_R is known as cycle time.

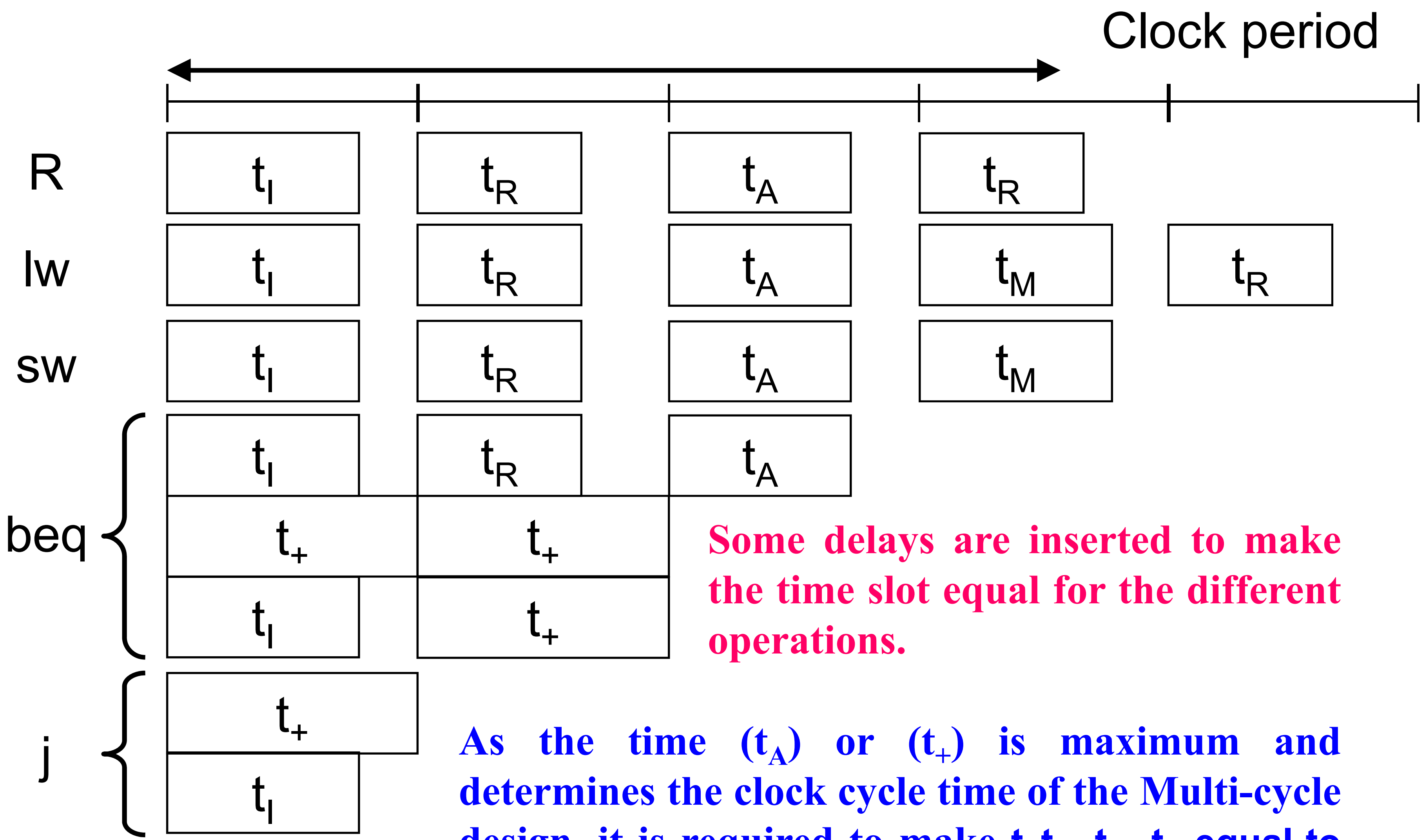
In this design, maximum of t_I, t_R, t_A, t_M, t_R will be cycle time. The equal time will be allocated for each activity in the unit. In this design, $t_I = t_R = t_A = t_M = t_R$.

The instruction execution time will be the summation of all the cycle times needed for this instruction.

The time ($t_A = t_+$) will determines the clock cycle time of the Multi-cycle design.

Actually, the activity time in the different components is different. But in the muti-cycle design, the time slots for all the activities are same. By using some techniques, same amount of time is provided for each operation though some of the operation are completed in less time.

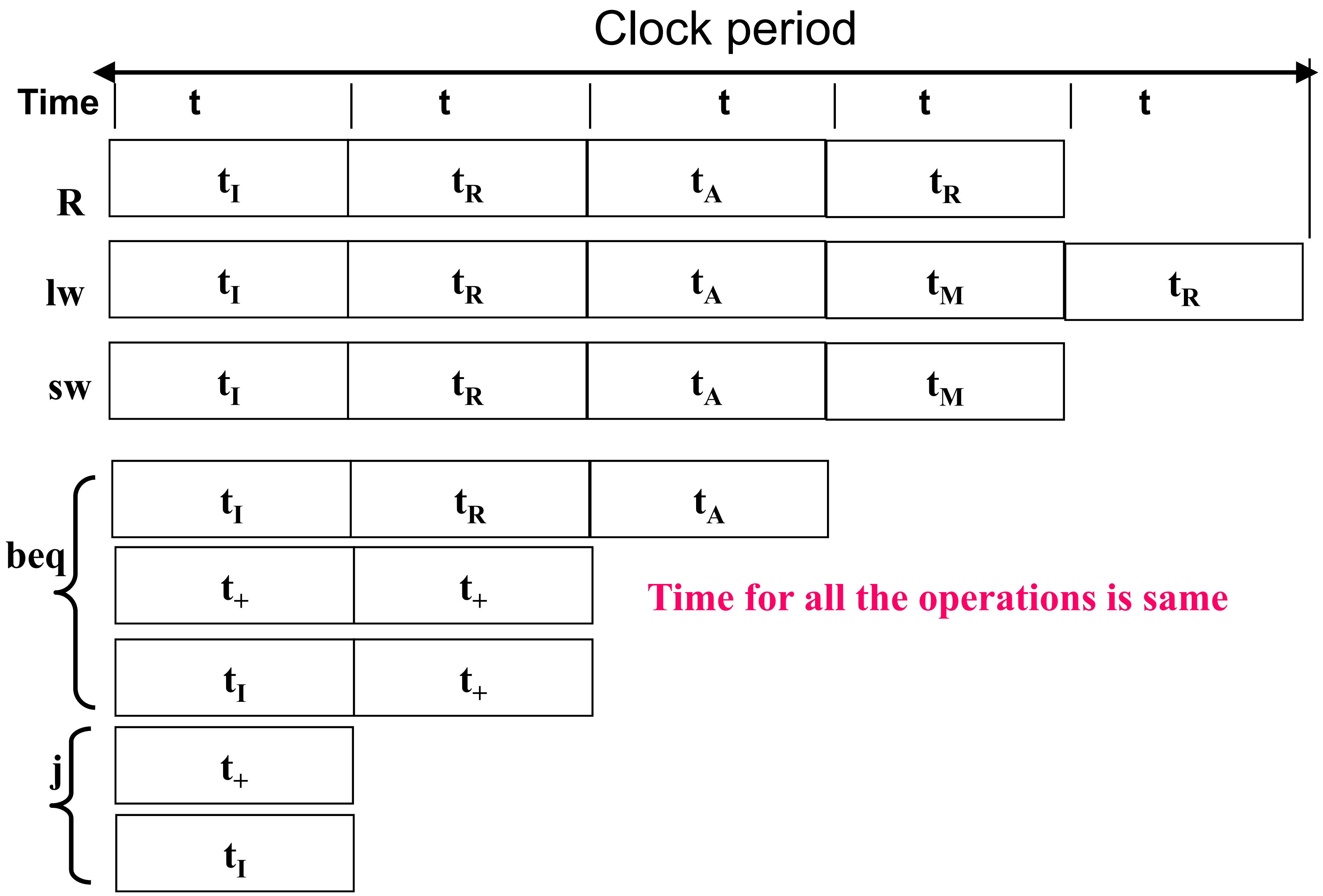
Unbalanced Delays



Some delays are inserted to make the time slot equal for the different operations.

As the time (t_A) or (t_+) is maximum and determines the clock cycle time of the Multi-cycle design, it is required to make t_I,t_R, t_M, t_R equal to t_A.

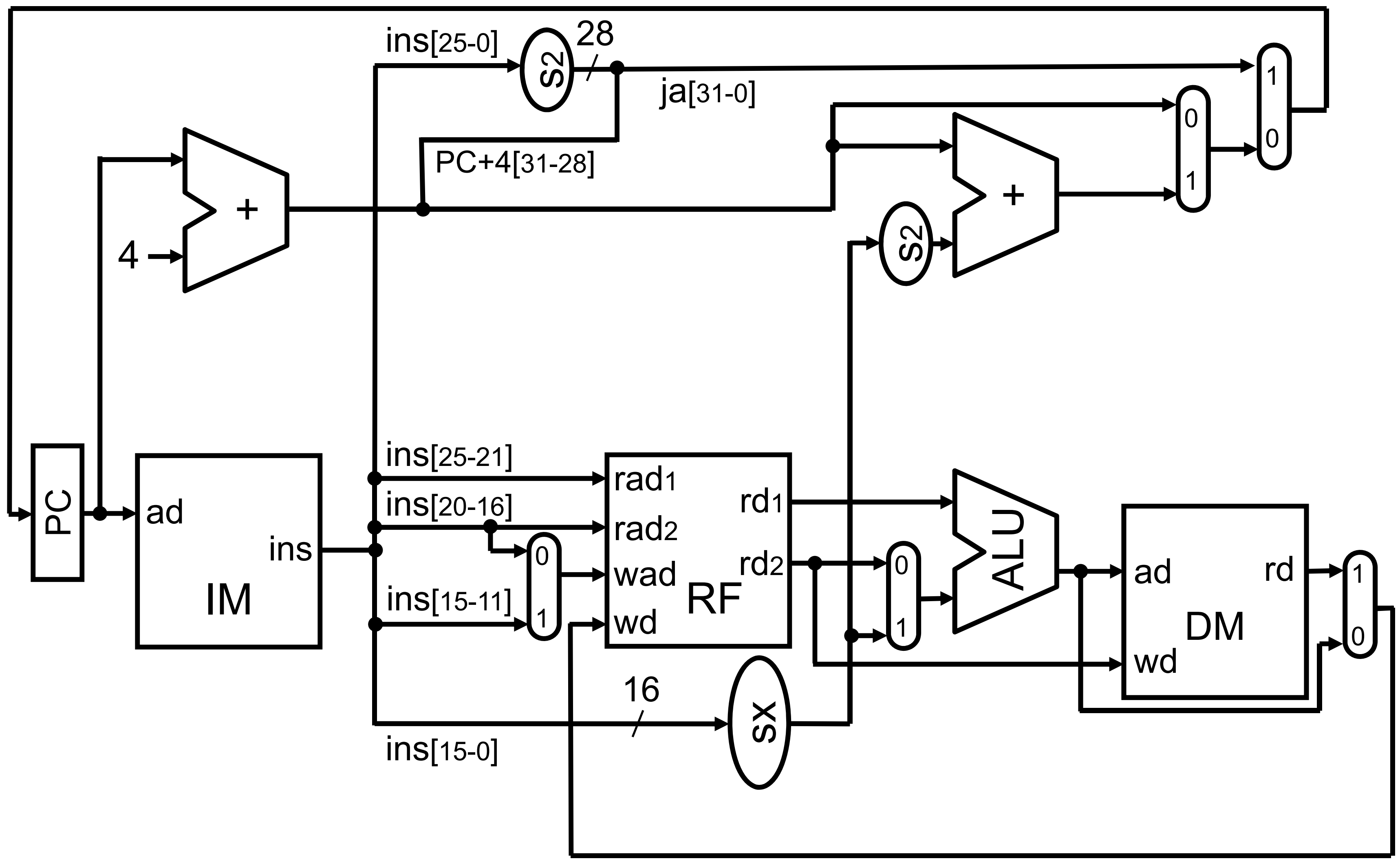
Clock Period in Multi-Cycle Design



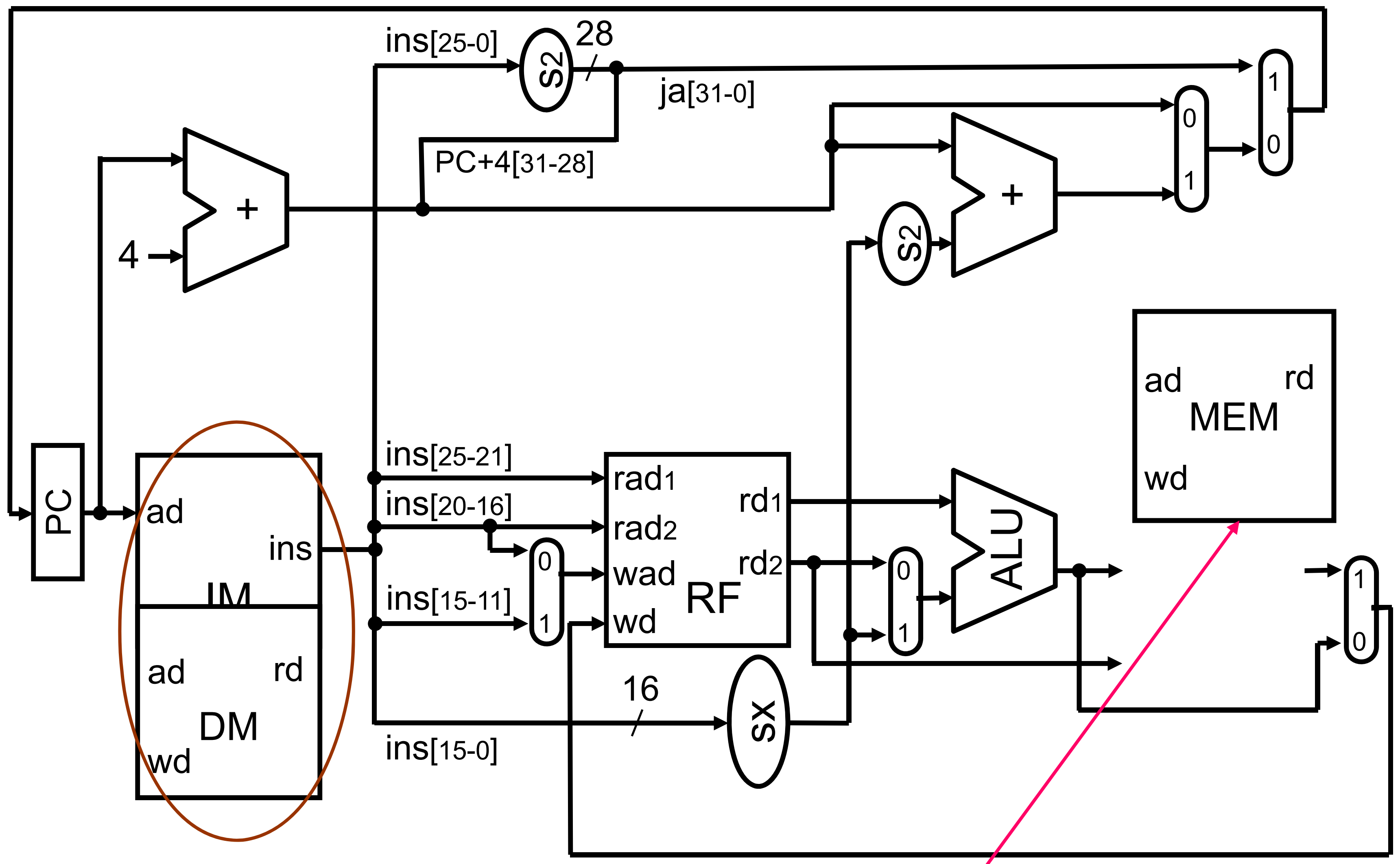
Improving Resource Utilization

- Eliminating two adders used in single cycle design
- Sharing/reusing a resource (say ALU) in different clock cycles
- Storing results in registers
- more multiplexing may be required
- Resources in this design: RF, ALU, MEM.

Single Cycle Datapath

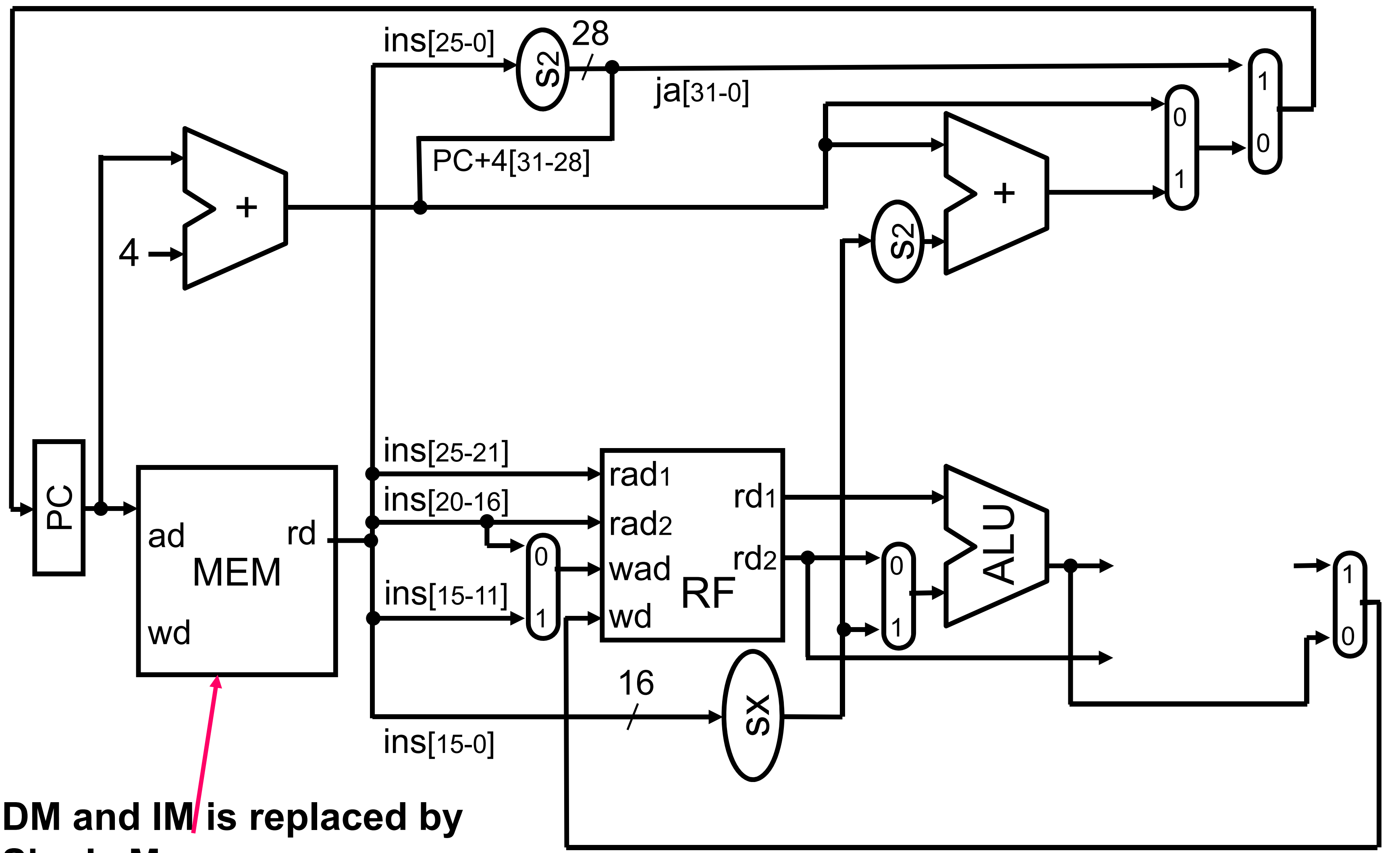


Merge IM and DM



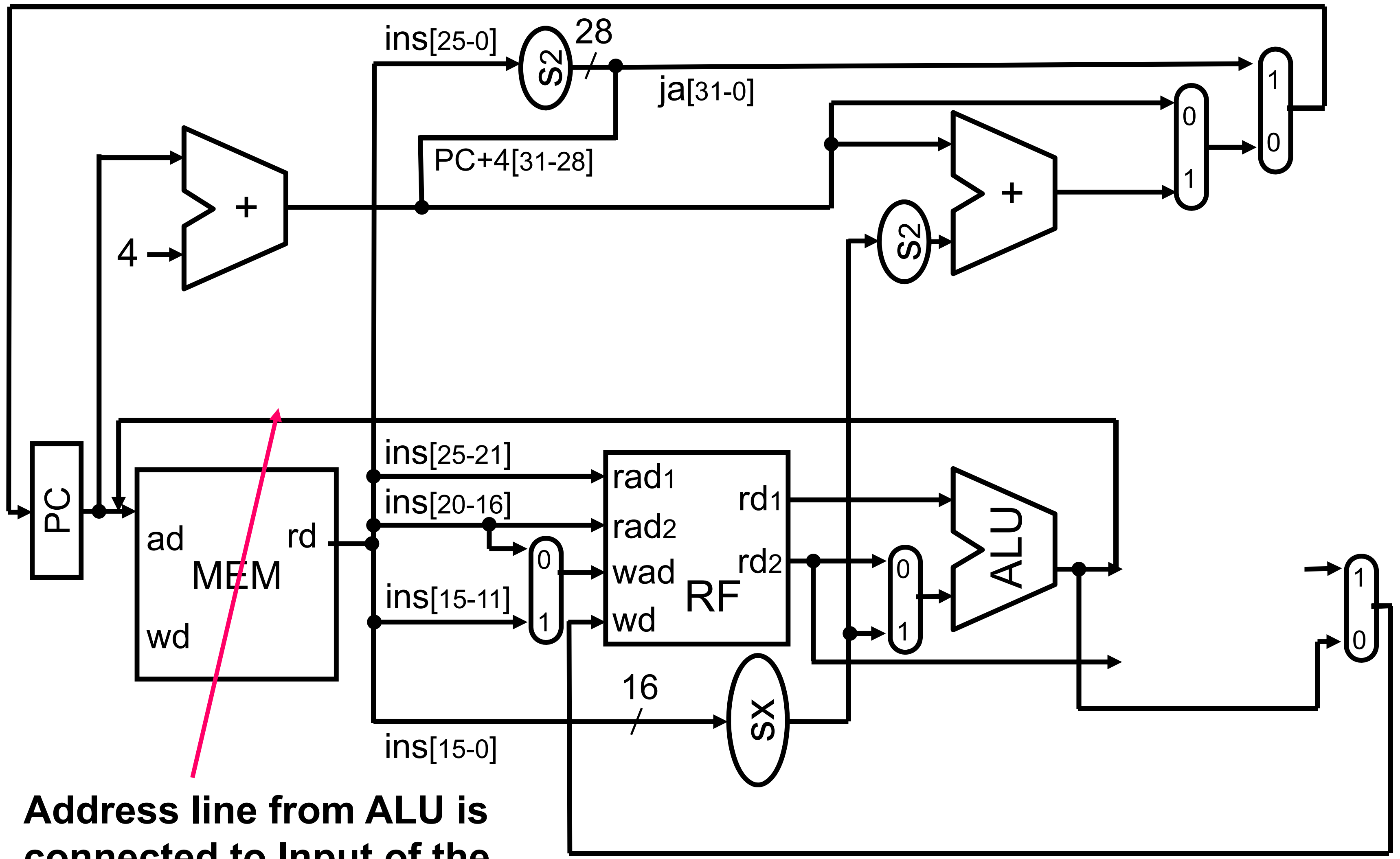
First, we will merge IM and DM using Single Memory.

Merge IM and DM



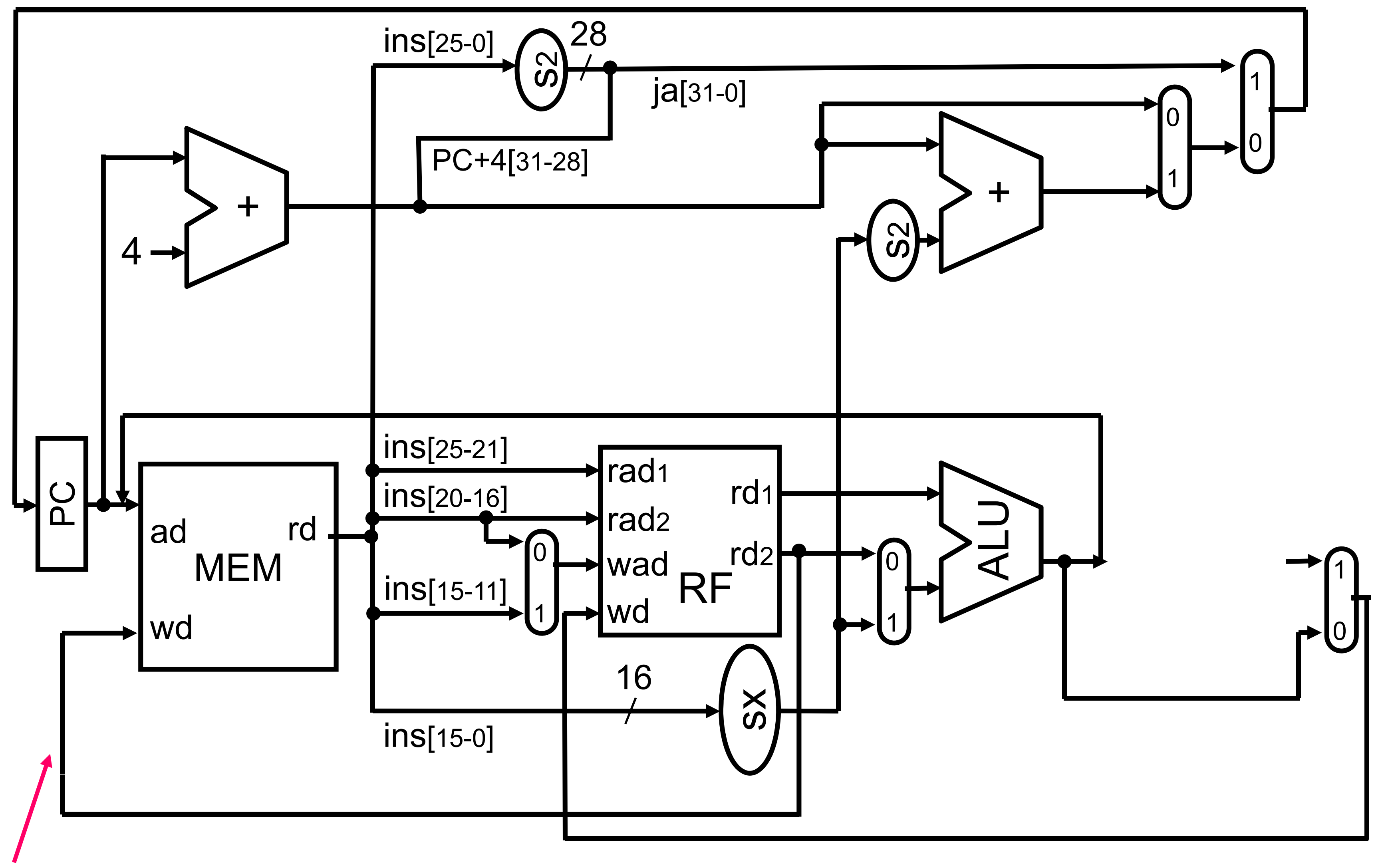
DM and IM is replaced by Single Memory.

Merge IM and DM



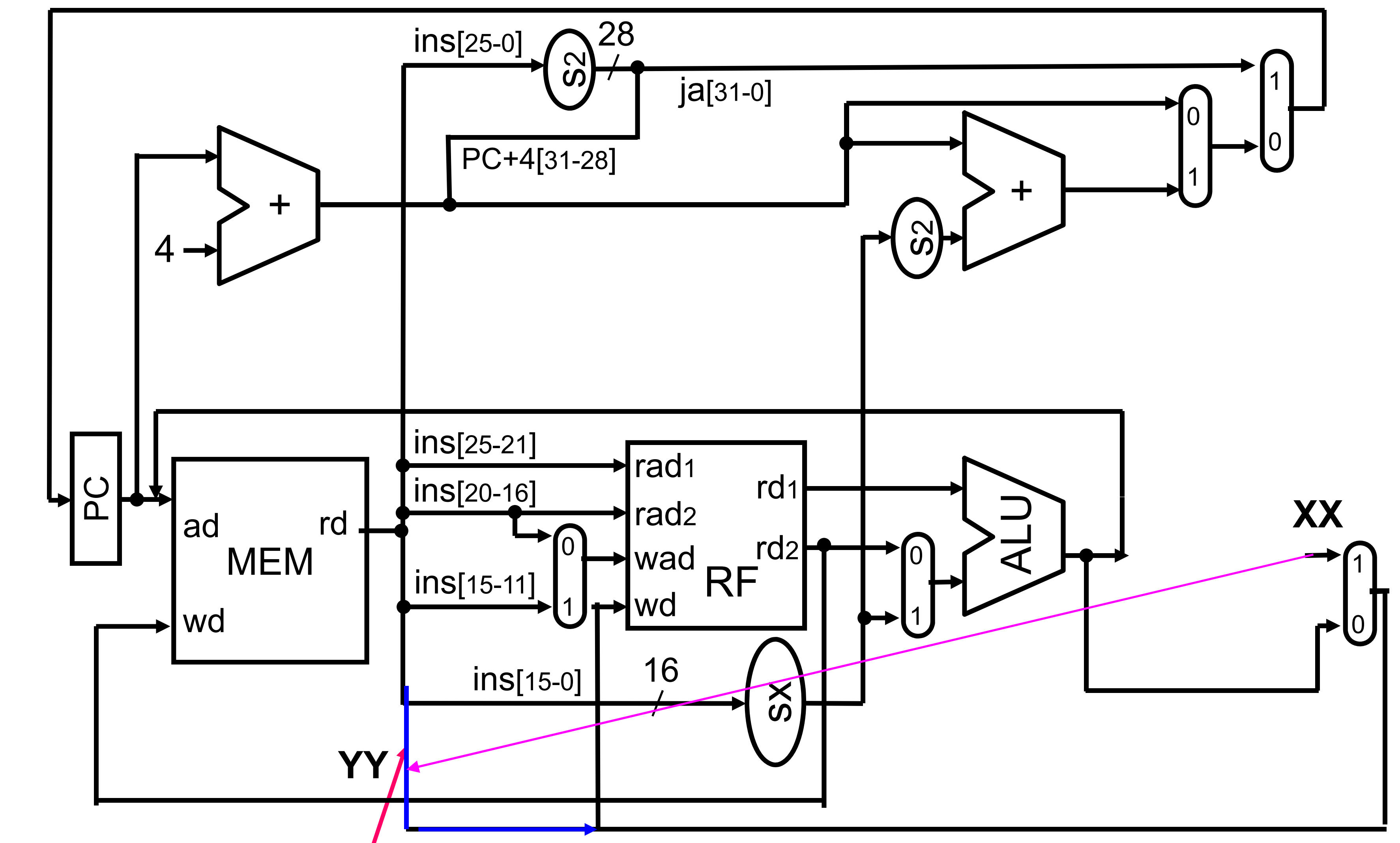
Address line from ALU is connected to Input of the Memory

Merge IM and DM



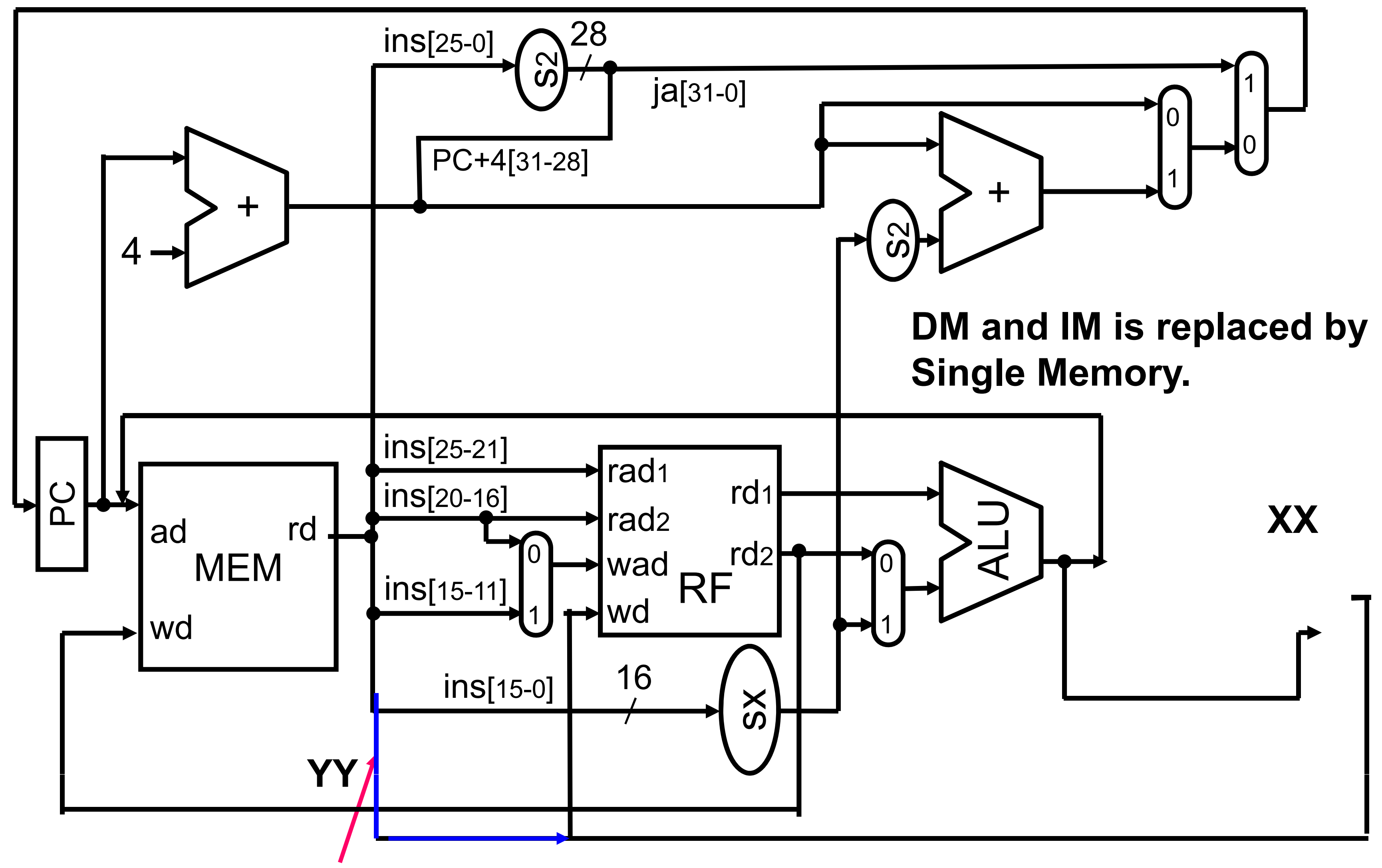
Data line carrying Data from RF to the Memory location through WD port of MEM

Merge IM and DM



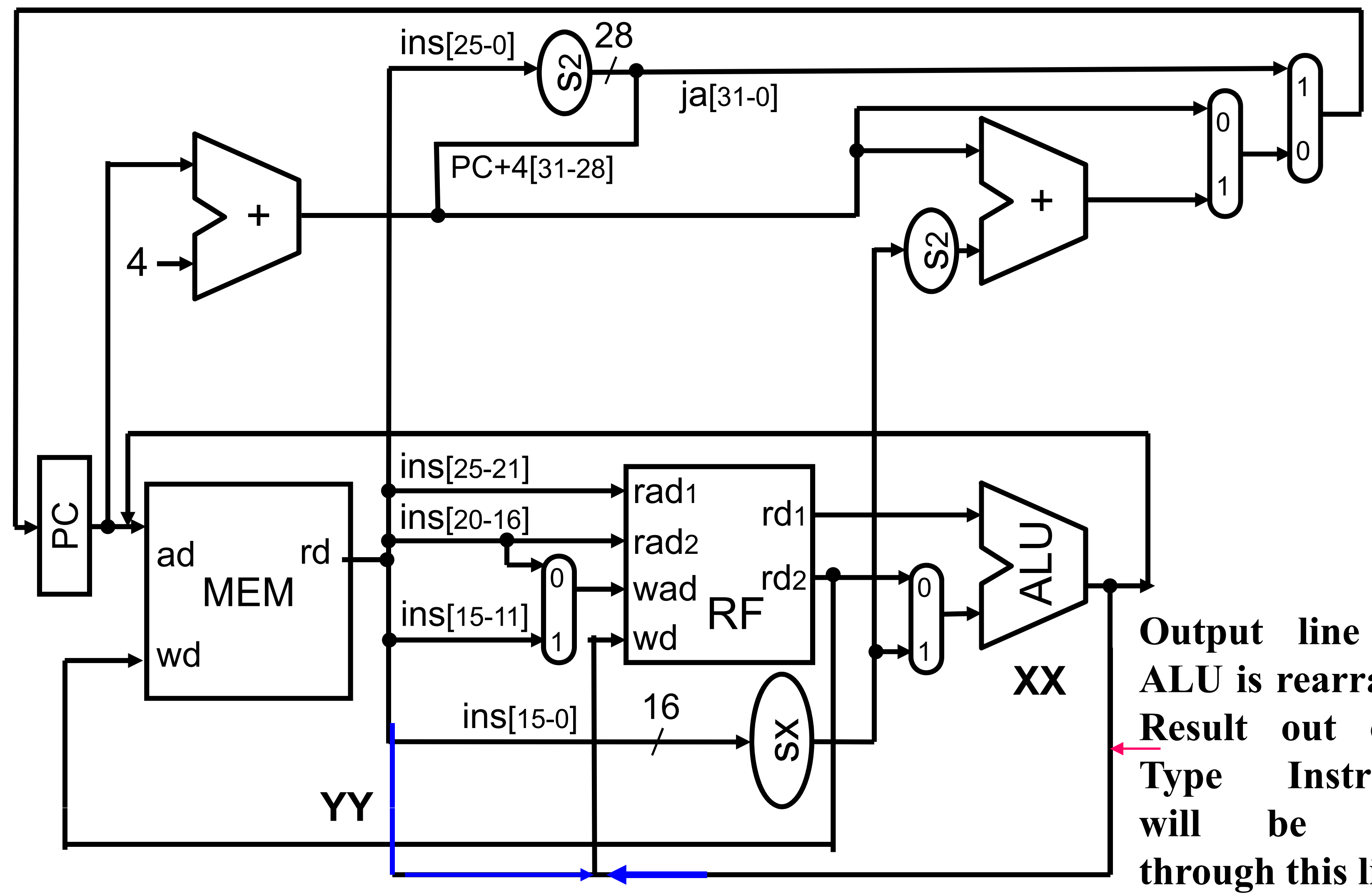
Data Memory line is rearranged

Merge IM and DM



Data Memory line XX is replaced by YY. XX line and connected MUX are removed

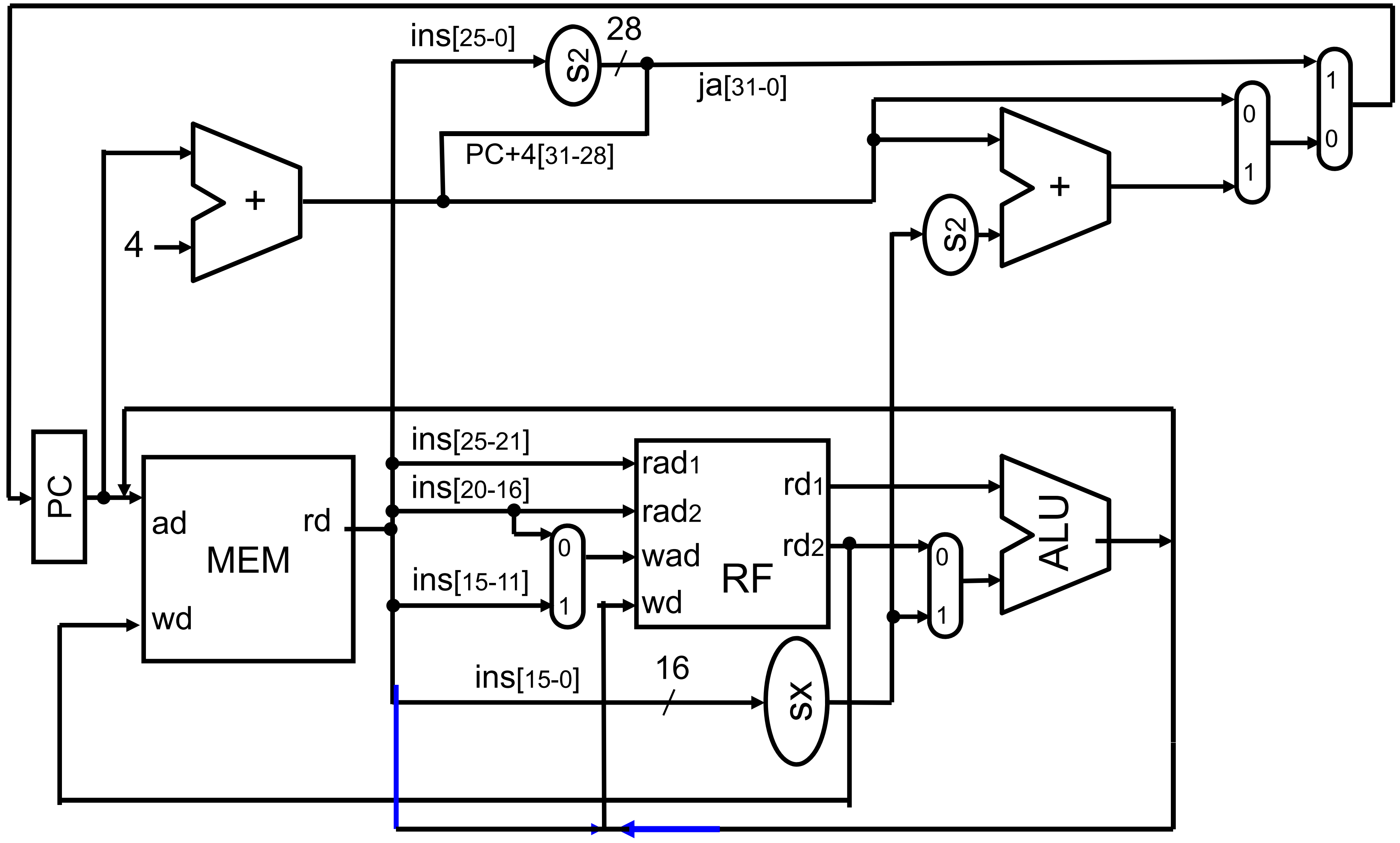
Merge IM and DM



Output line from ALU is rearranged. Result out of R-Type Instruction will be stored through this line.

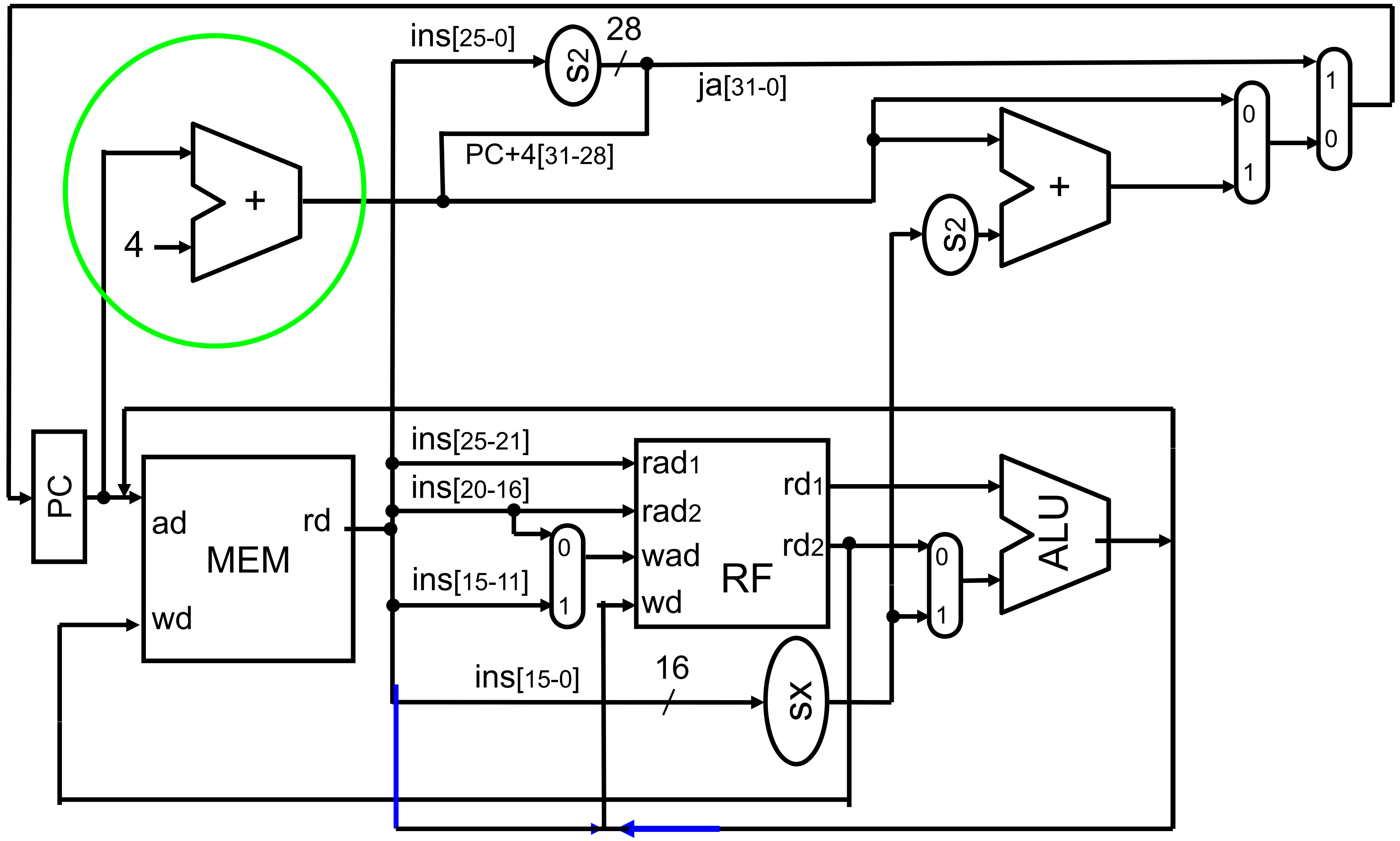
Rearrange the Diagram

Hardware reduction has been done using Single Memory by combing the actions of IM and DM.



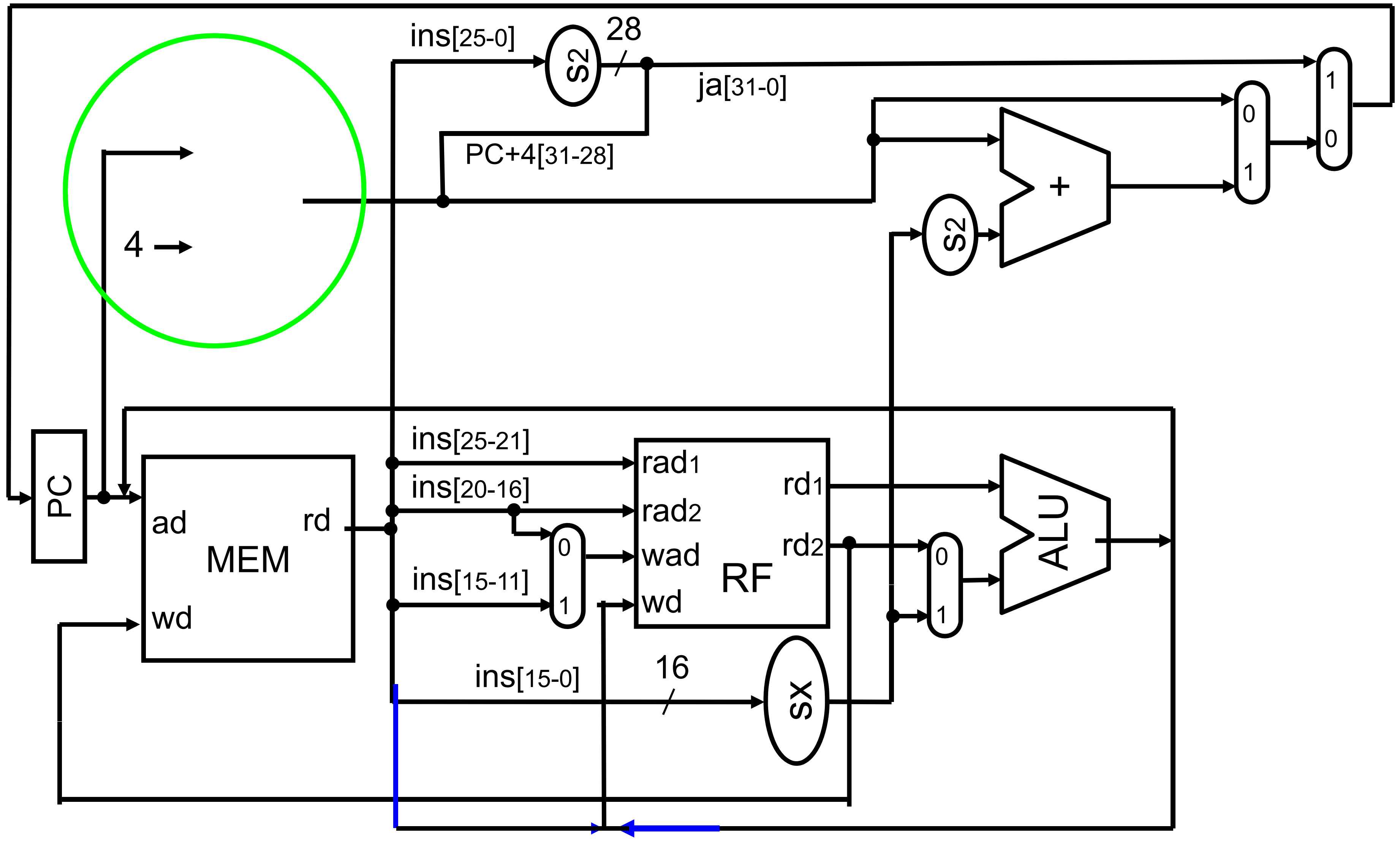
Elimination of First Adder

Now we will remove two adders and use single ALU



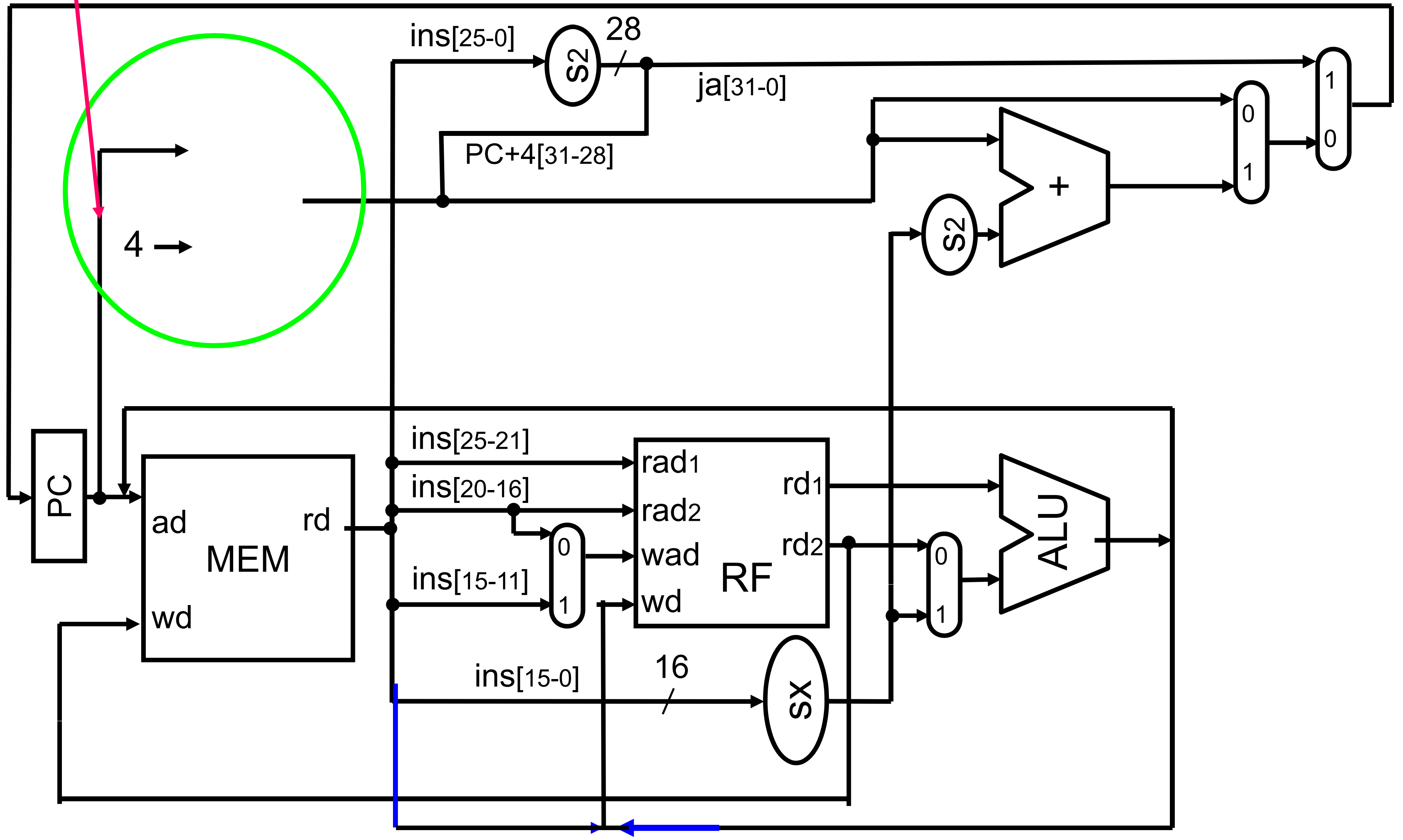
Elimination of First Adder

Adder is removed.



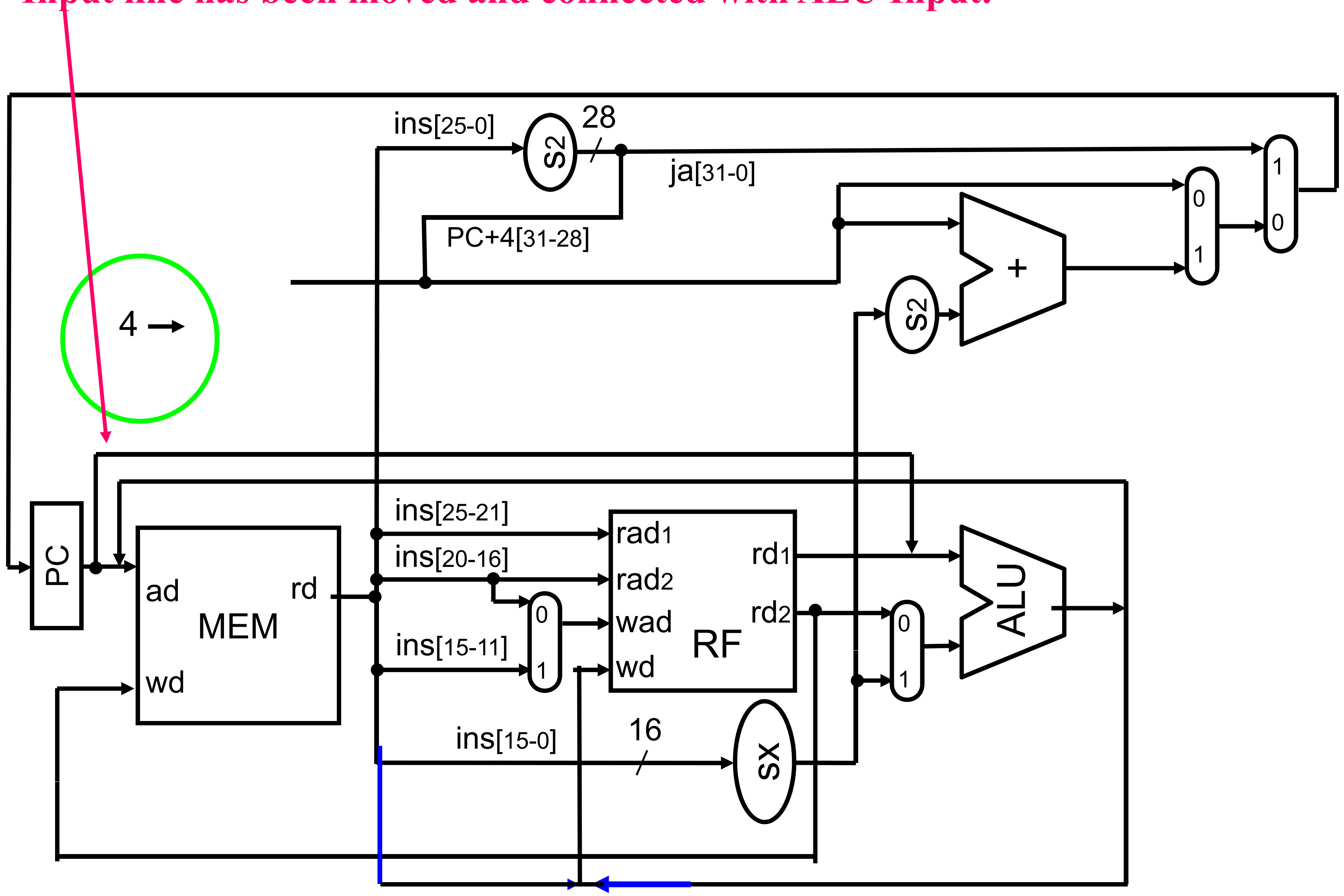
Elimination of First Adder

Input will be moved and rearranged.

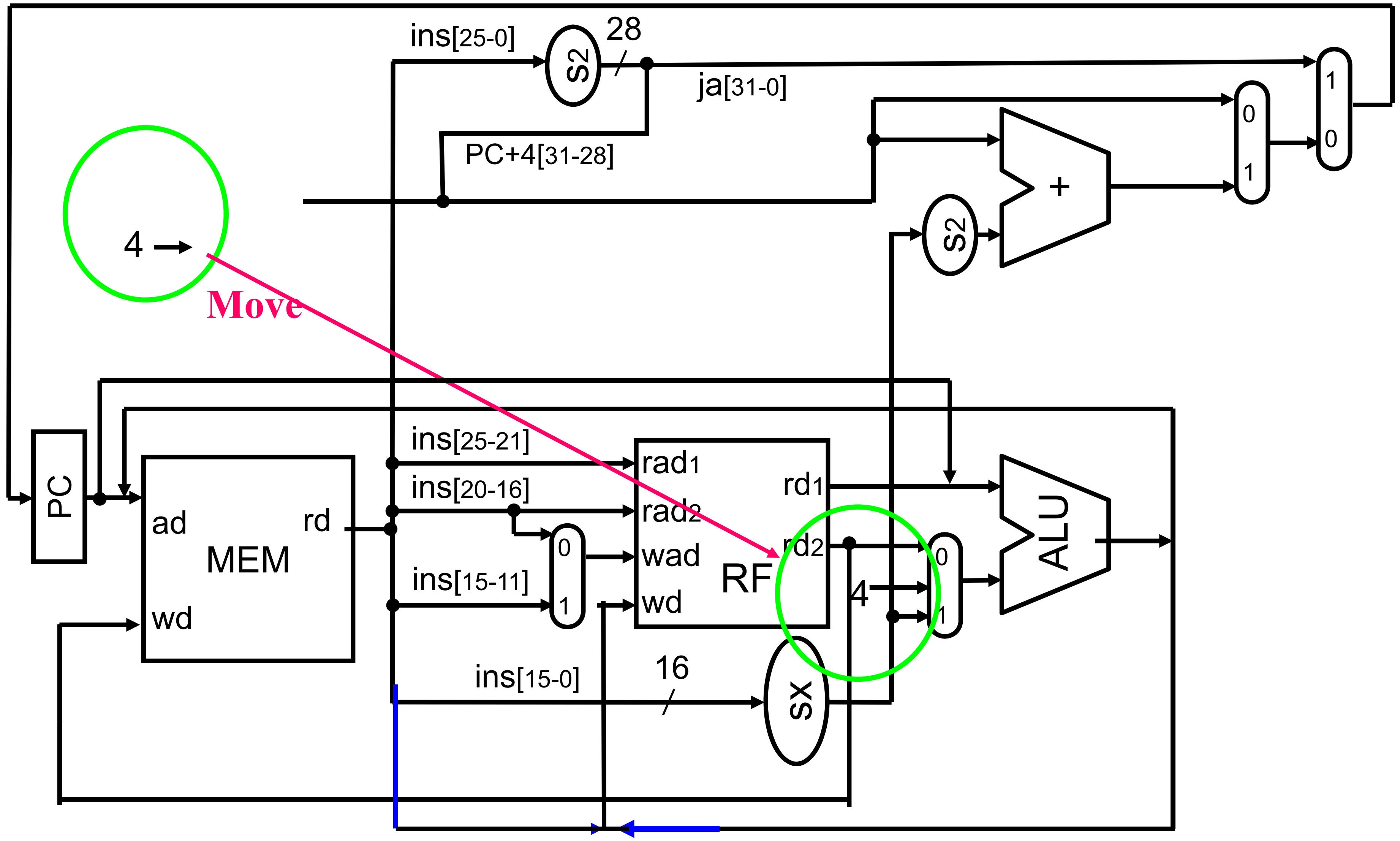


Elimination of First Adder

Input line has been moved and connected with ALU Input.

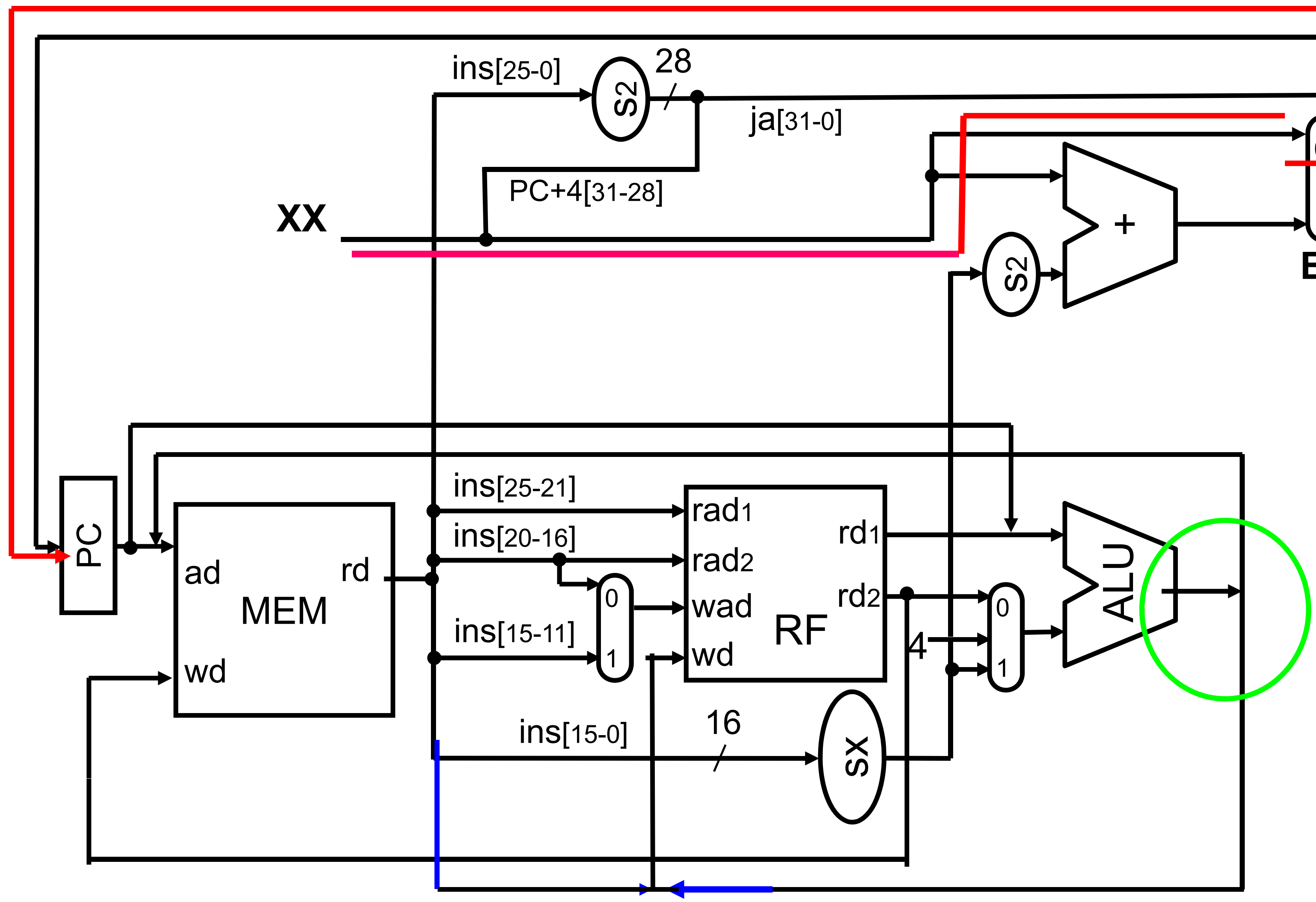


Elimination of First Adder



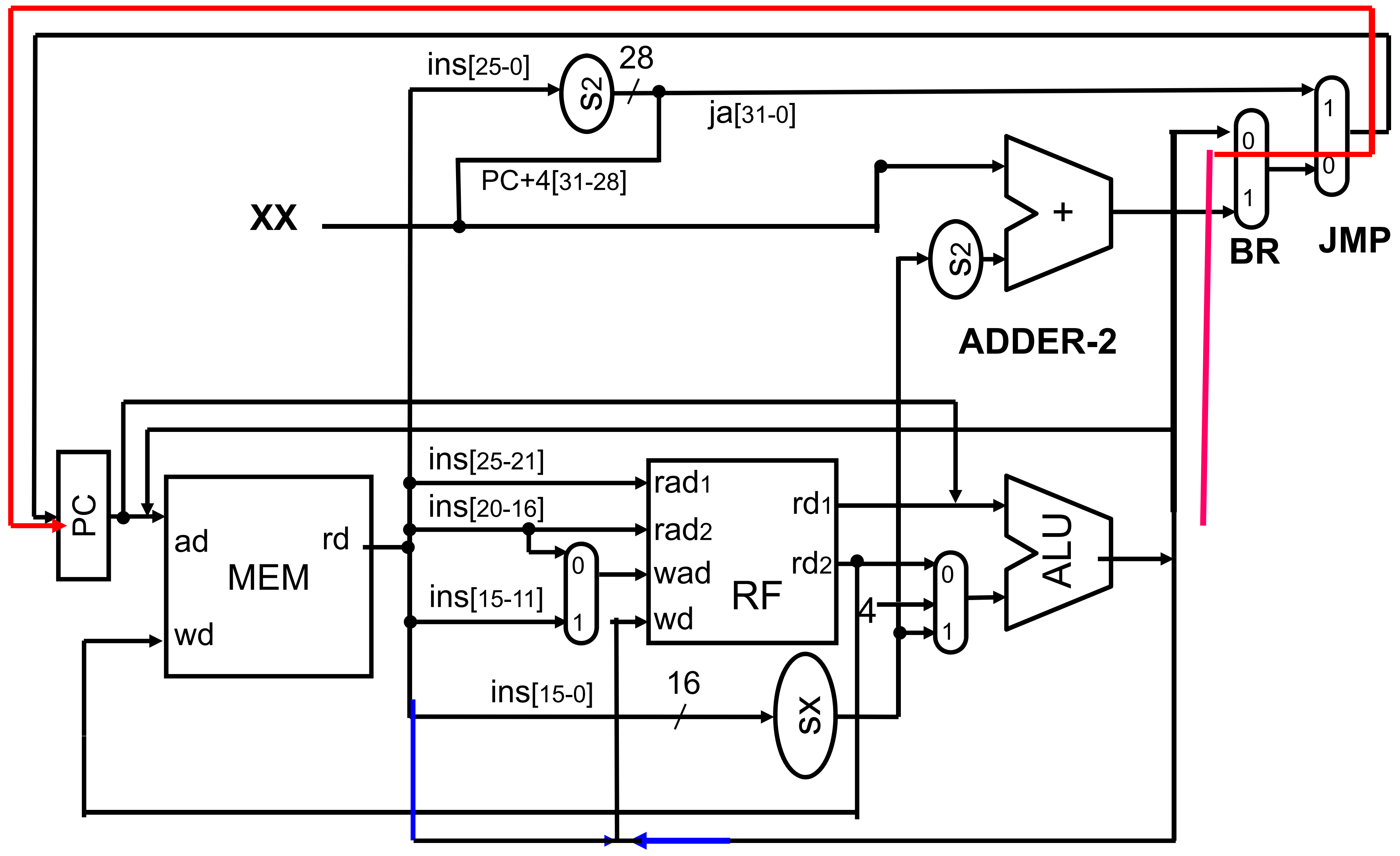
Rearrange after Elimination of First Adder

Output (XX) of first adder was coming to PC input through BR and JMP MUX. Now it will come from the output of ALU.



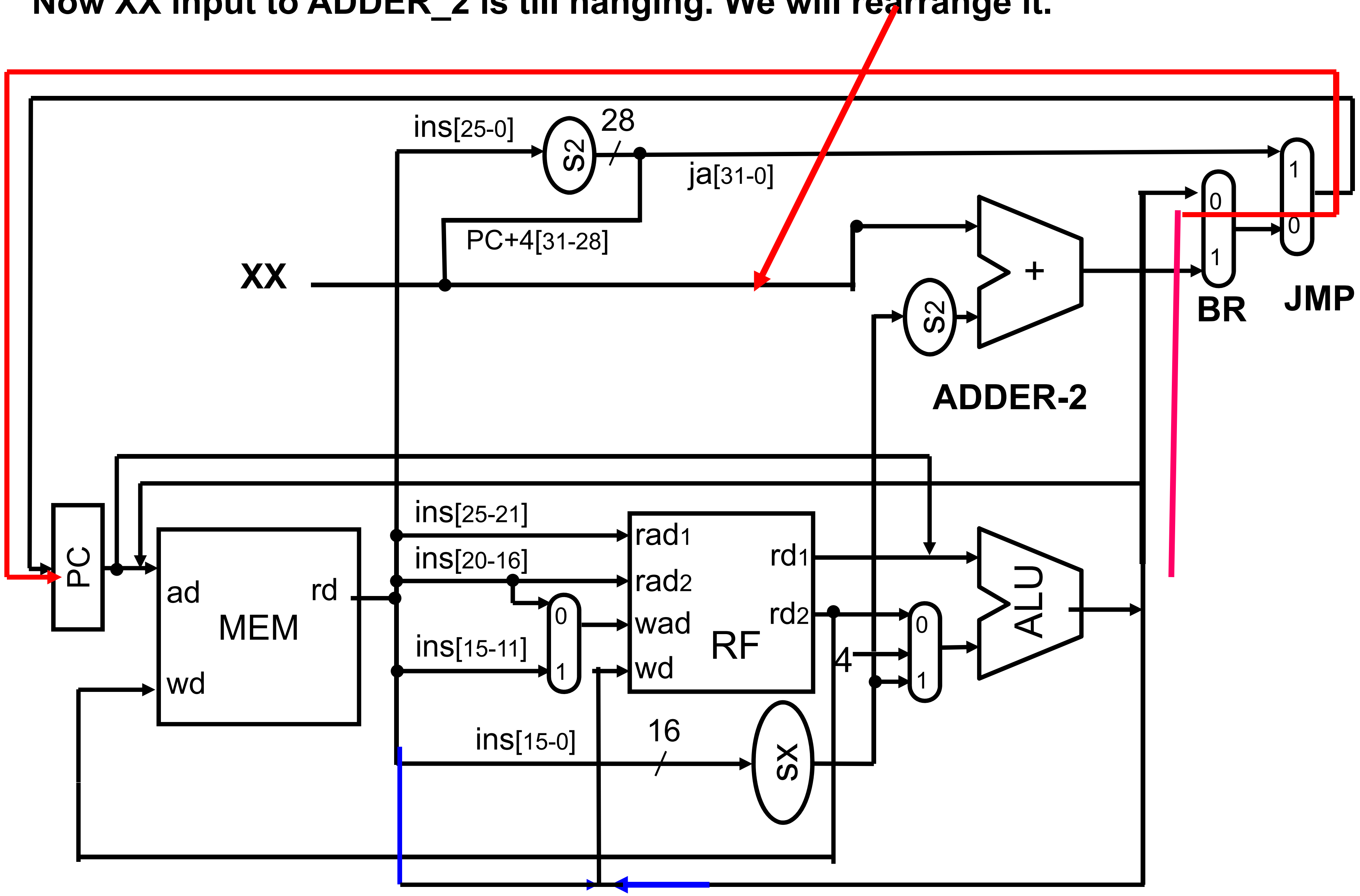
Rearrange after Elimination of First Adder

Now (PC+4) output is coming to PC input through BR and JMP MUX from the output of ALU.



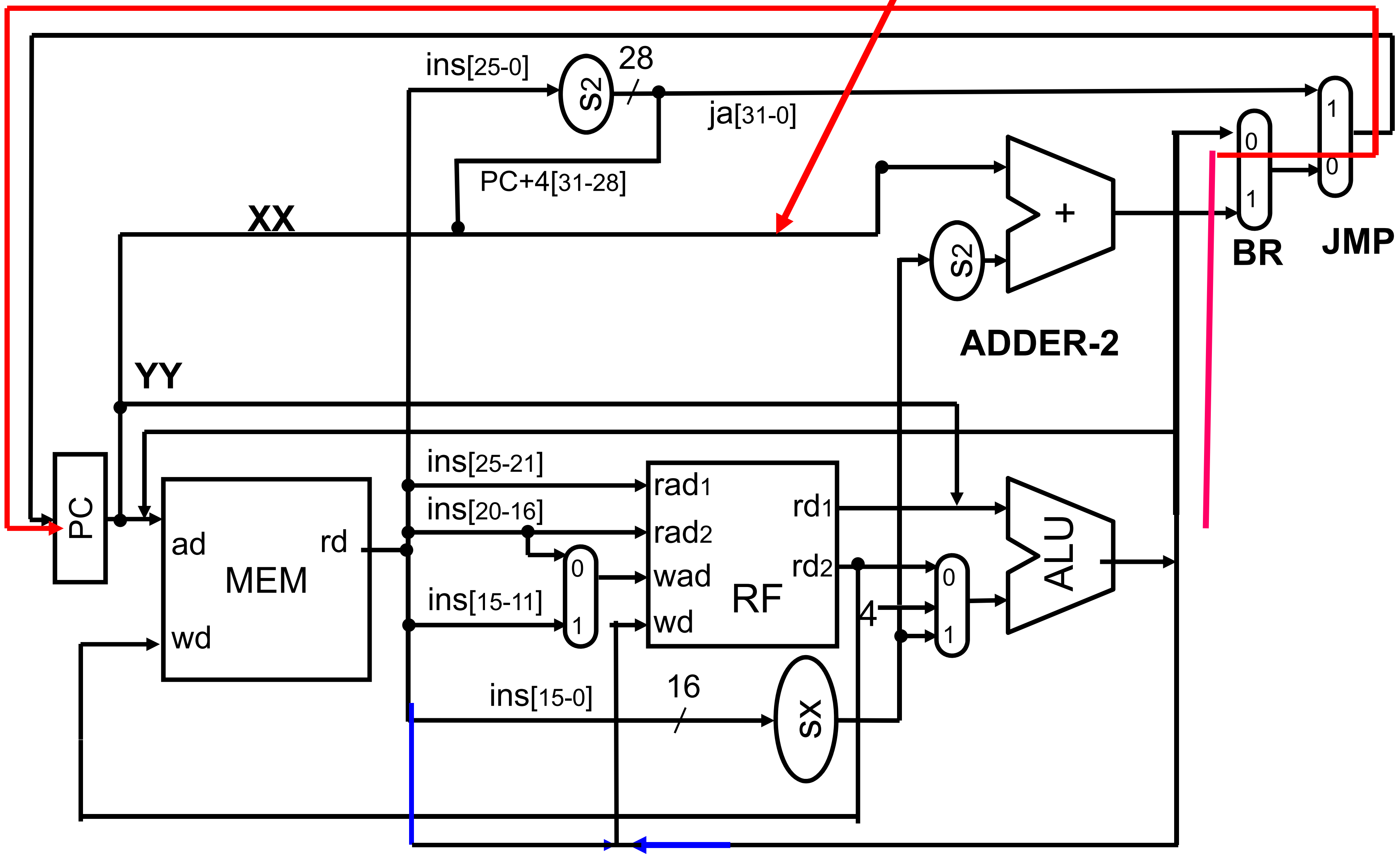
Rearrange after Elimination of First Adder

Now XX input to ADDER_2 is till hanging. We will rearrange it.



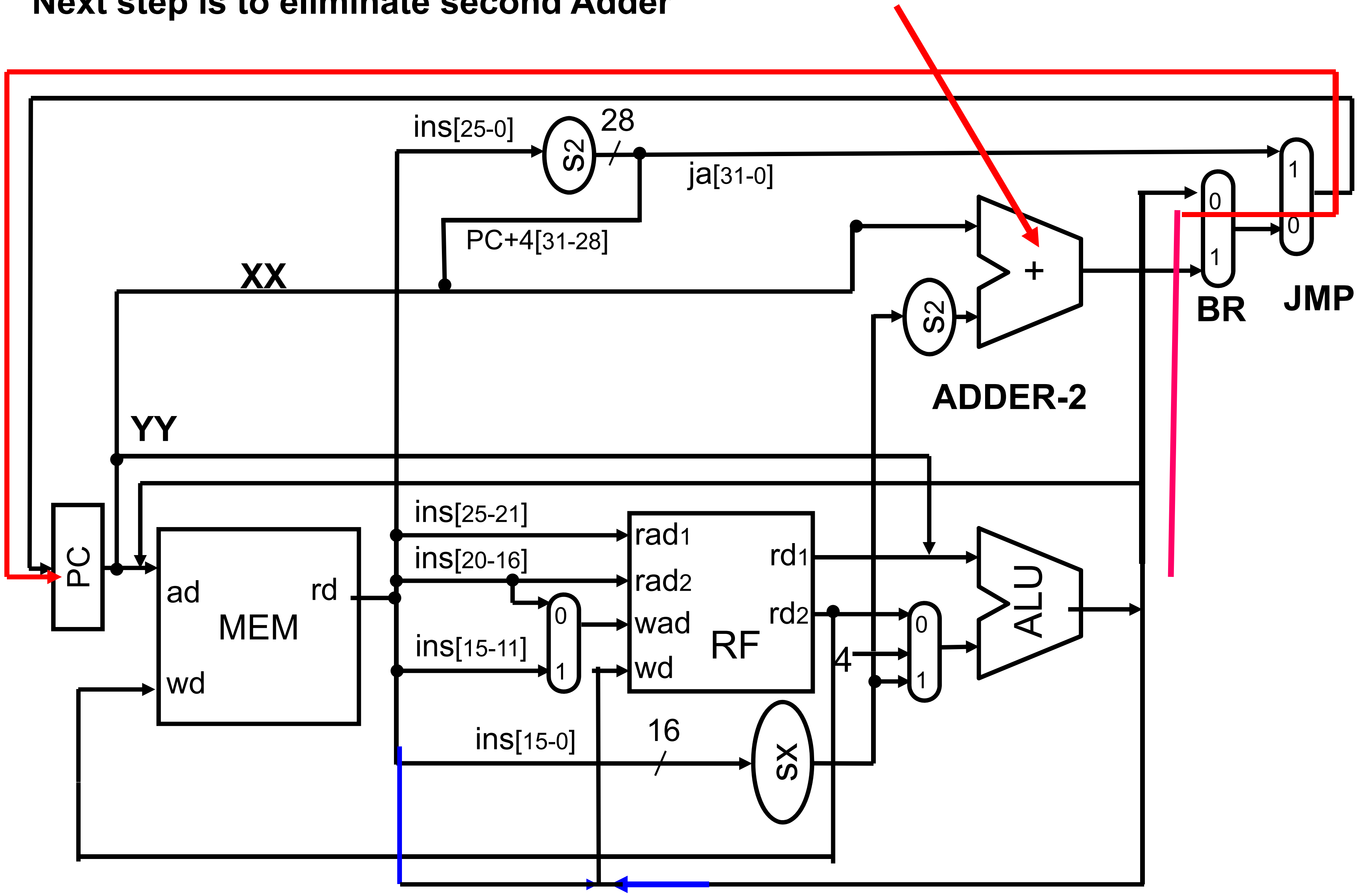
Rearrange after Elimination of First Adder

Now, XX input is connected to YY point as (PC+4) is coming through YY during a particular cycle

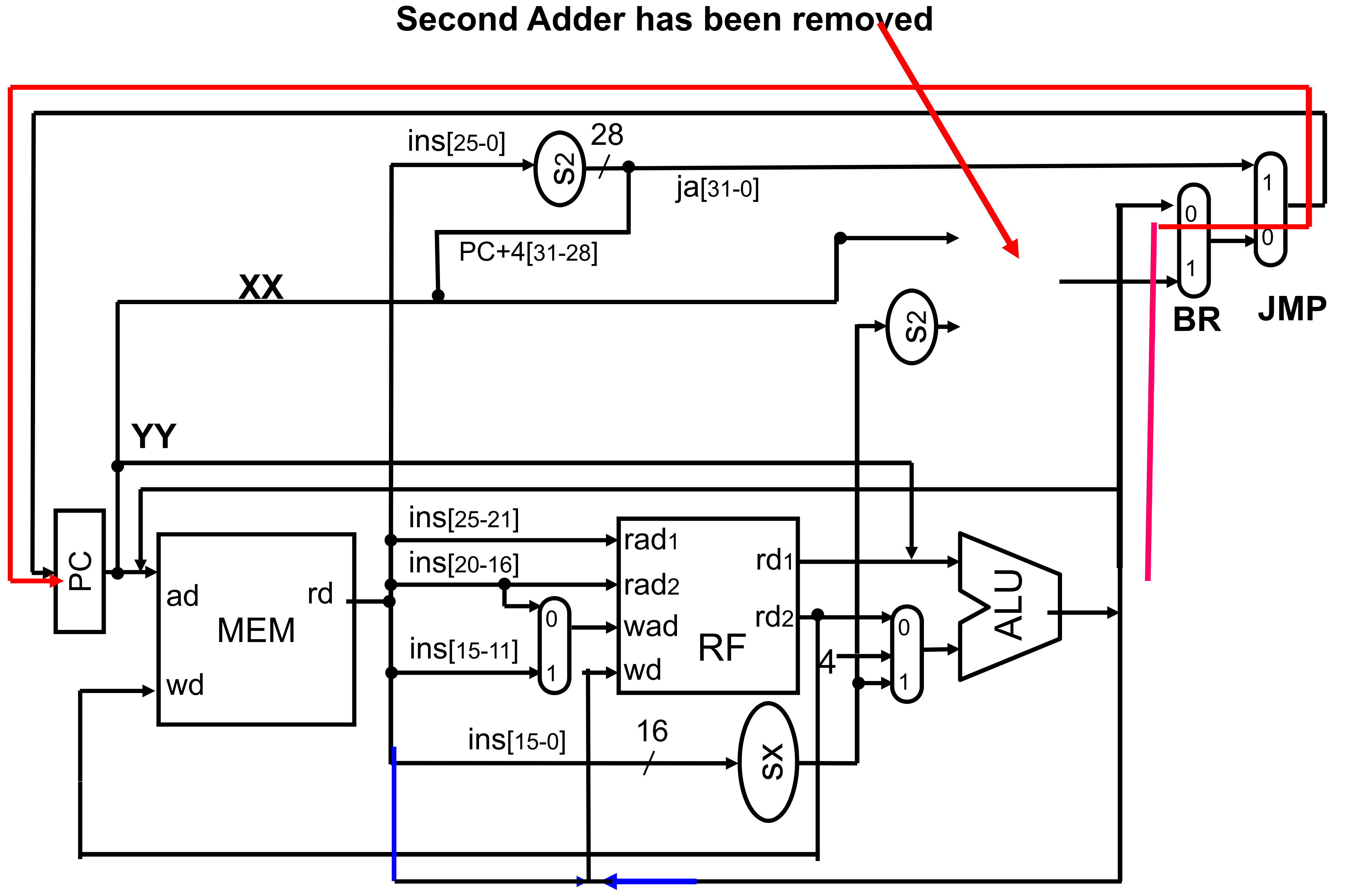


Elimination of Second Adder

Next step is to eliminate second Adder

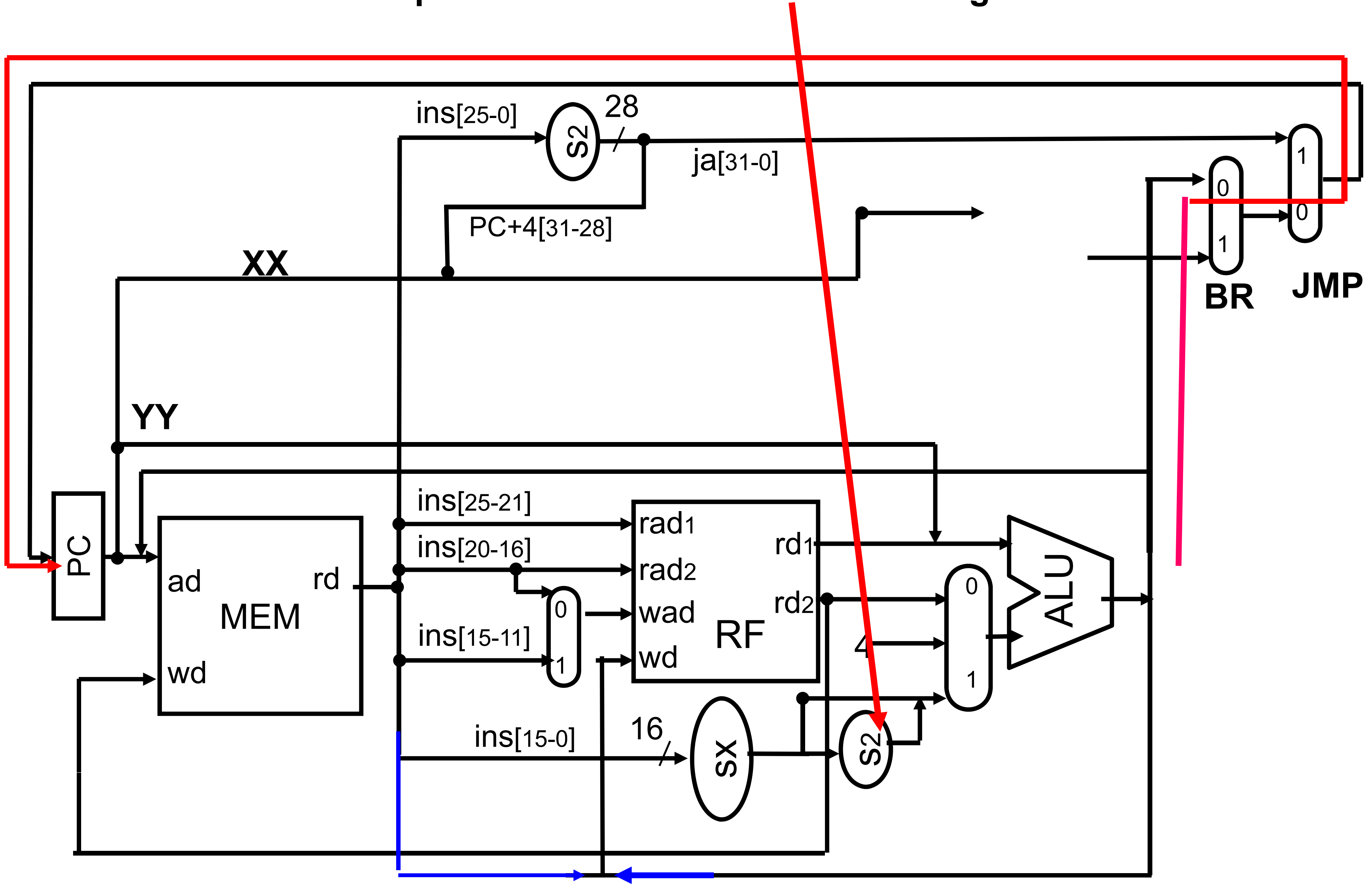


Elimination of Second Adder



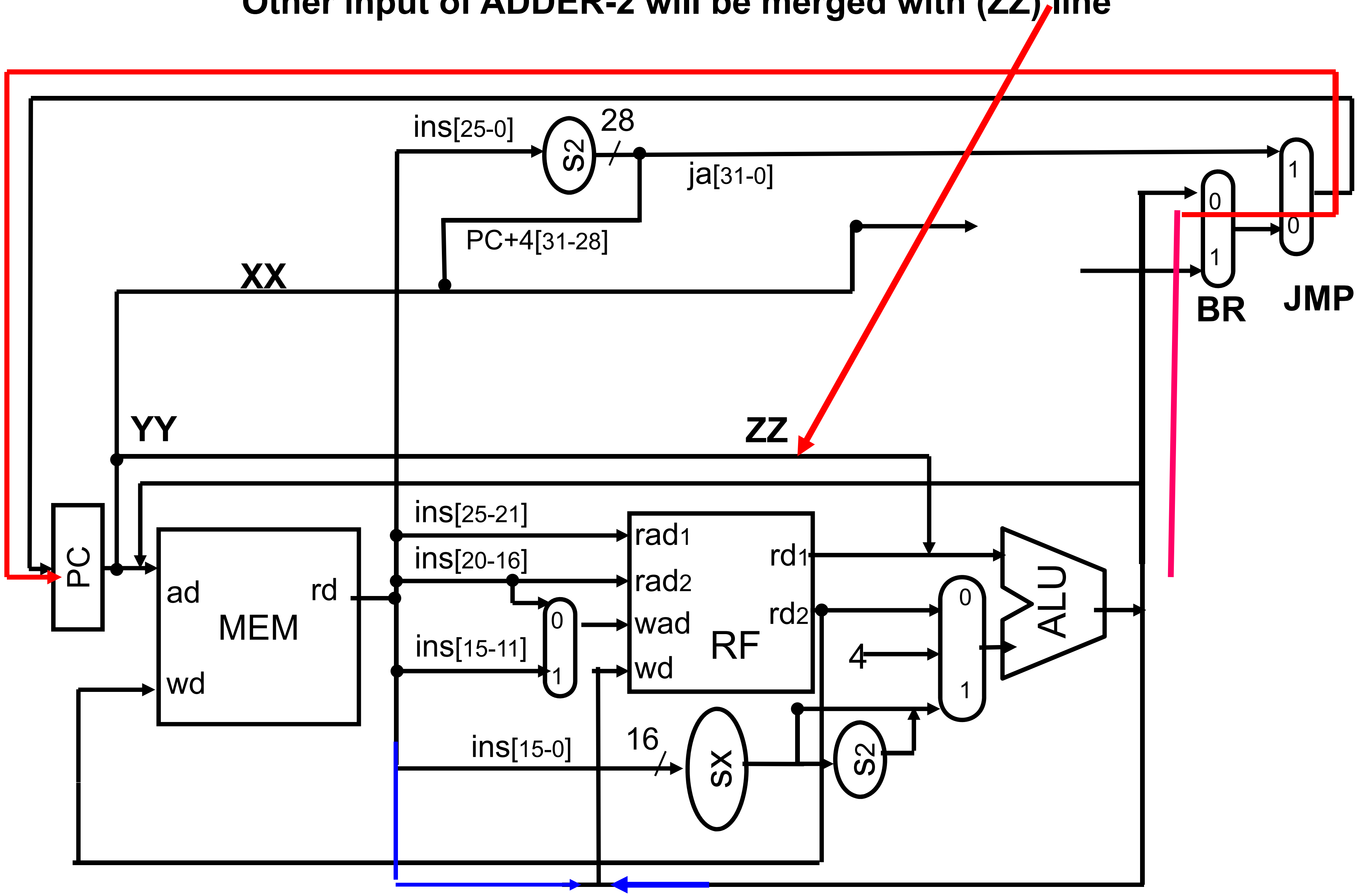
Elimination of Second Adder

S2 input of ADDER-2 has been rearranged



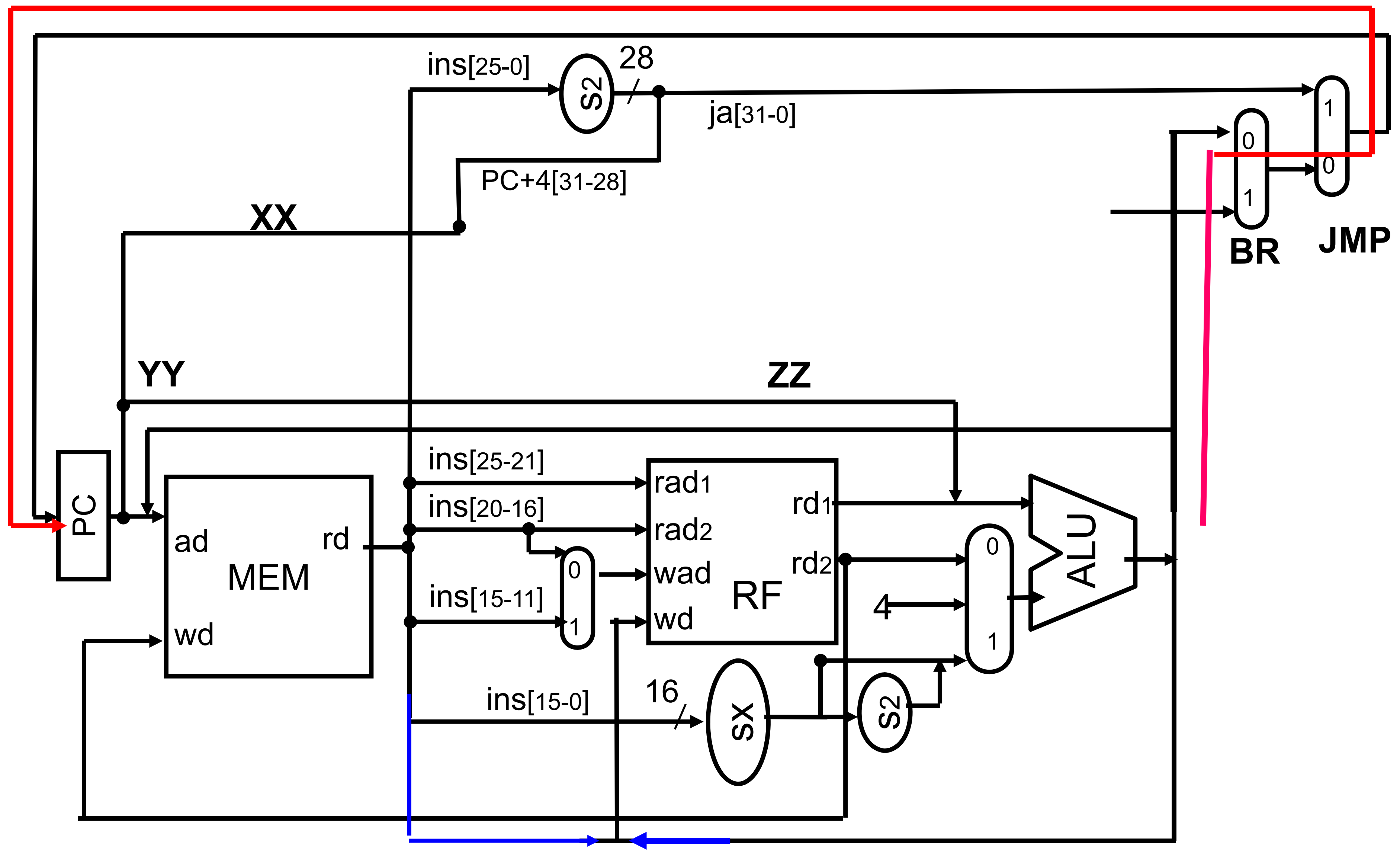
Elimination of Second Adder

Other input of ADDER-2 will be merged with (ZZ) line



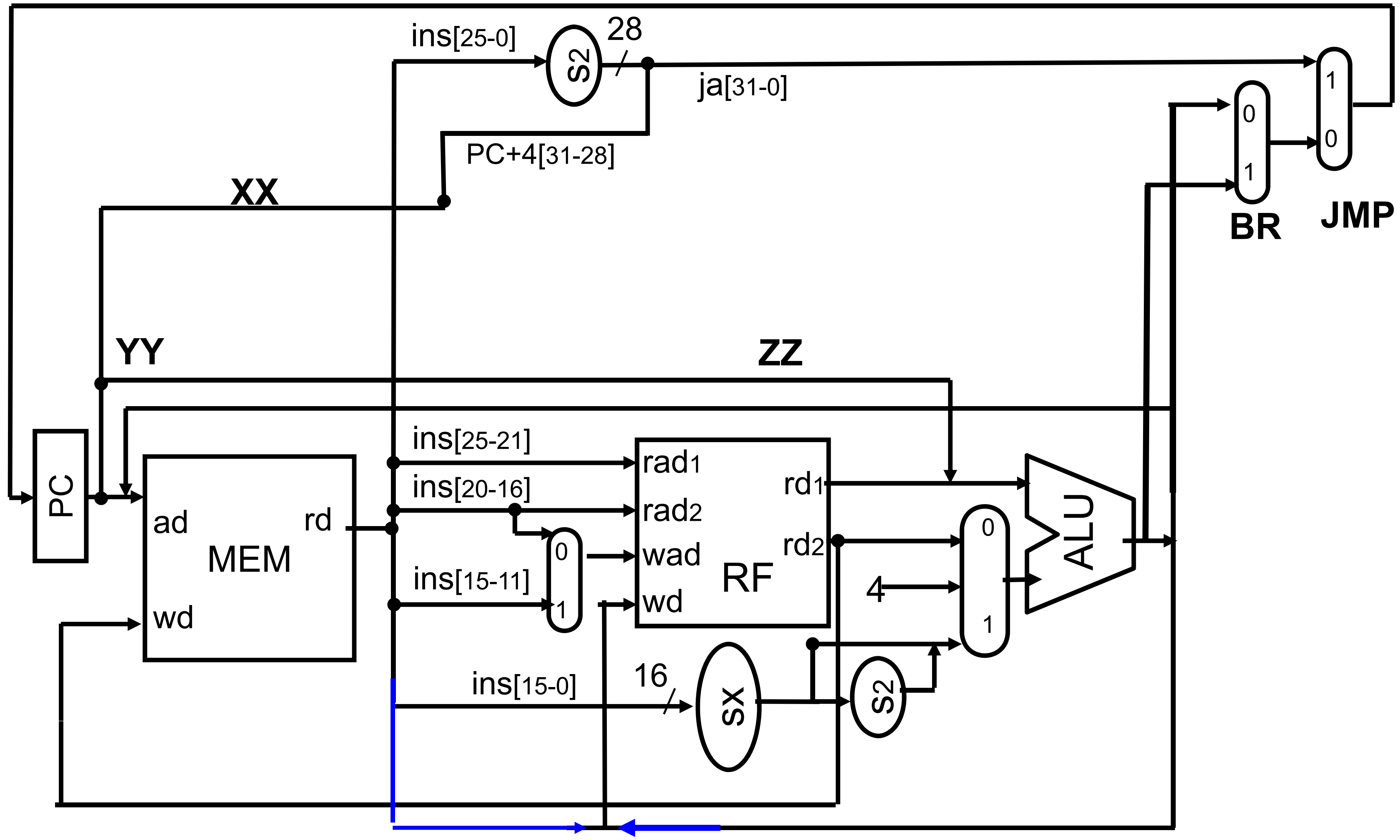
Elimination of Second Adder

Other input of ADDER-2 has been merged with (ZZ) line

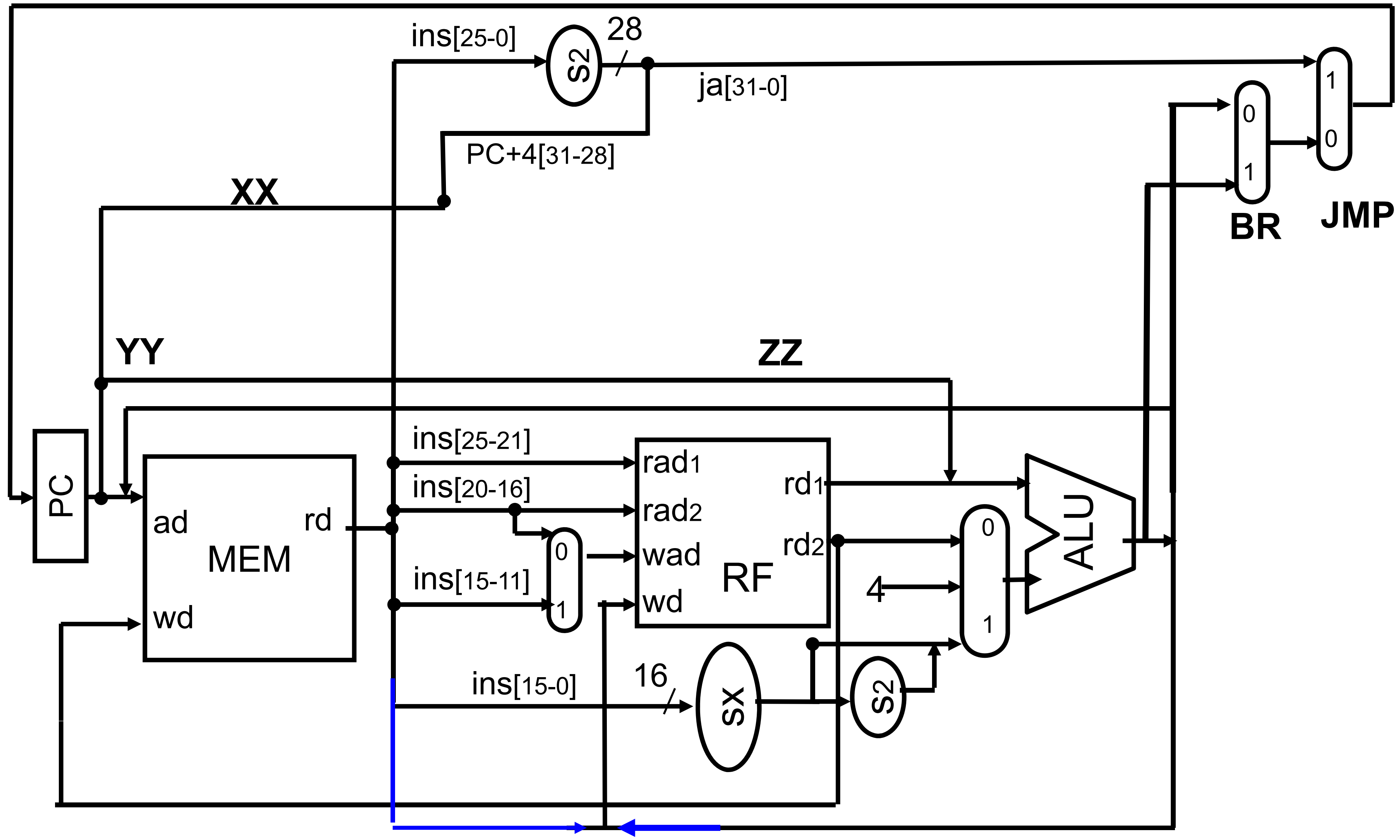


Elimination of Second Adder

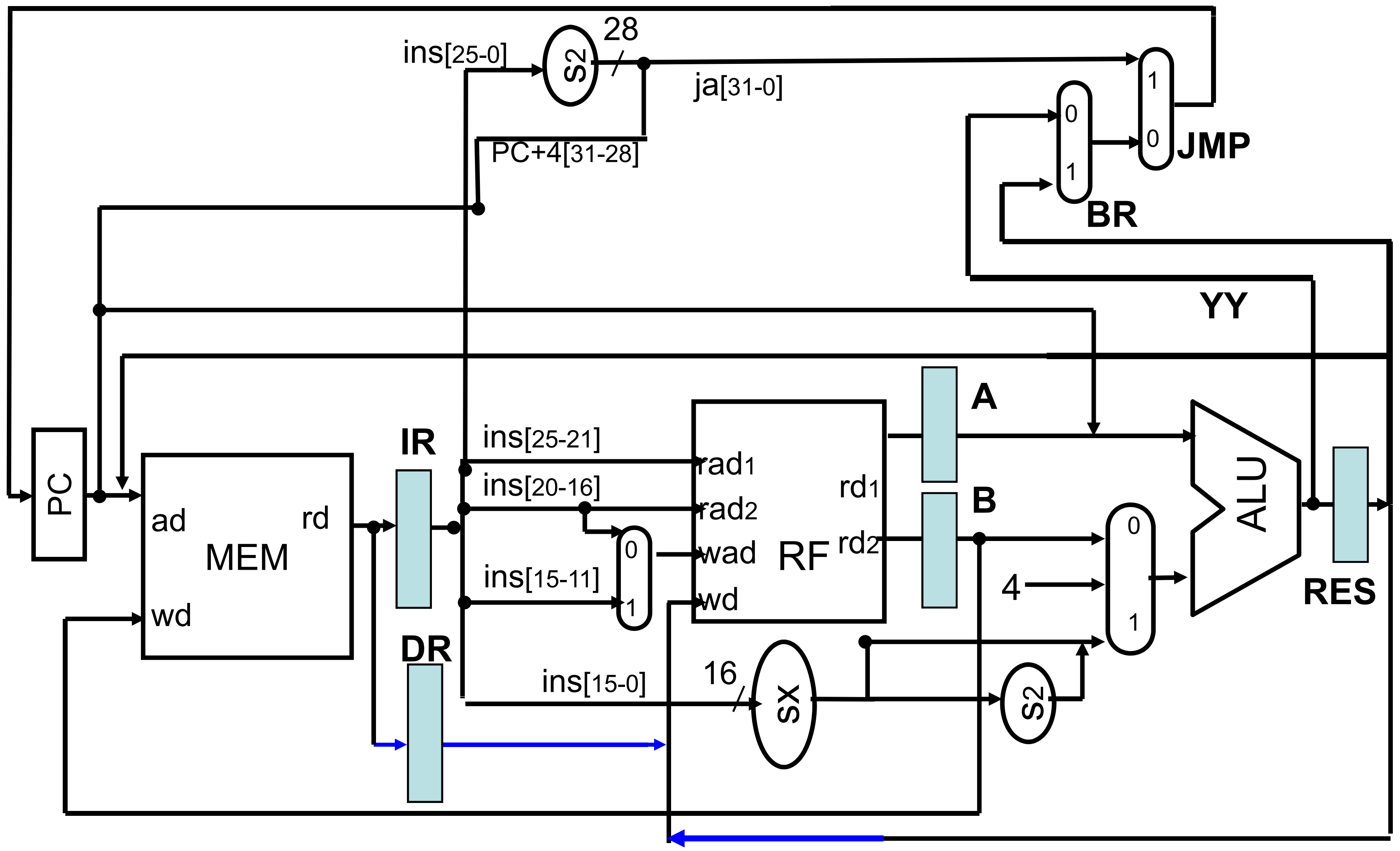
Output of ADDER-2 is replaced by ALU Output



Rearrange the Diagram



Introduce Registers



Introduce Registers

- We are reading instruction and data from the same memory. So we need two registers:-IR register for storing instruction and DR (Data Register) for storing data.

- We are reading two operands from Register File. So we need two registers (A and B) for storing operands.

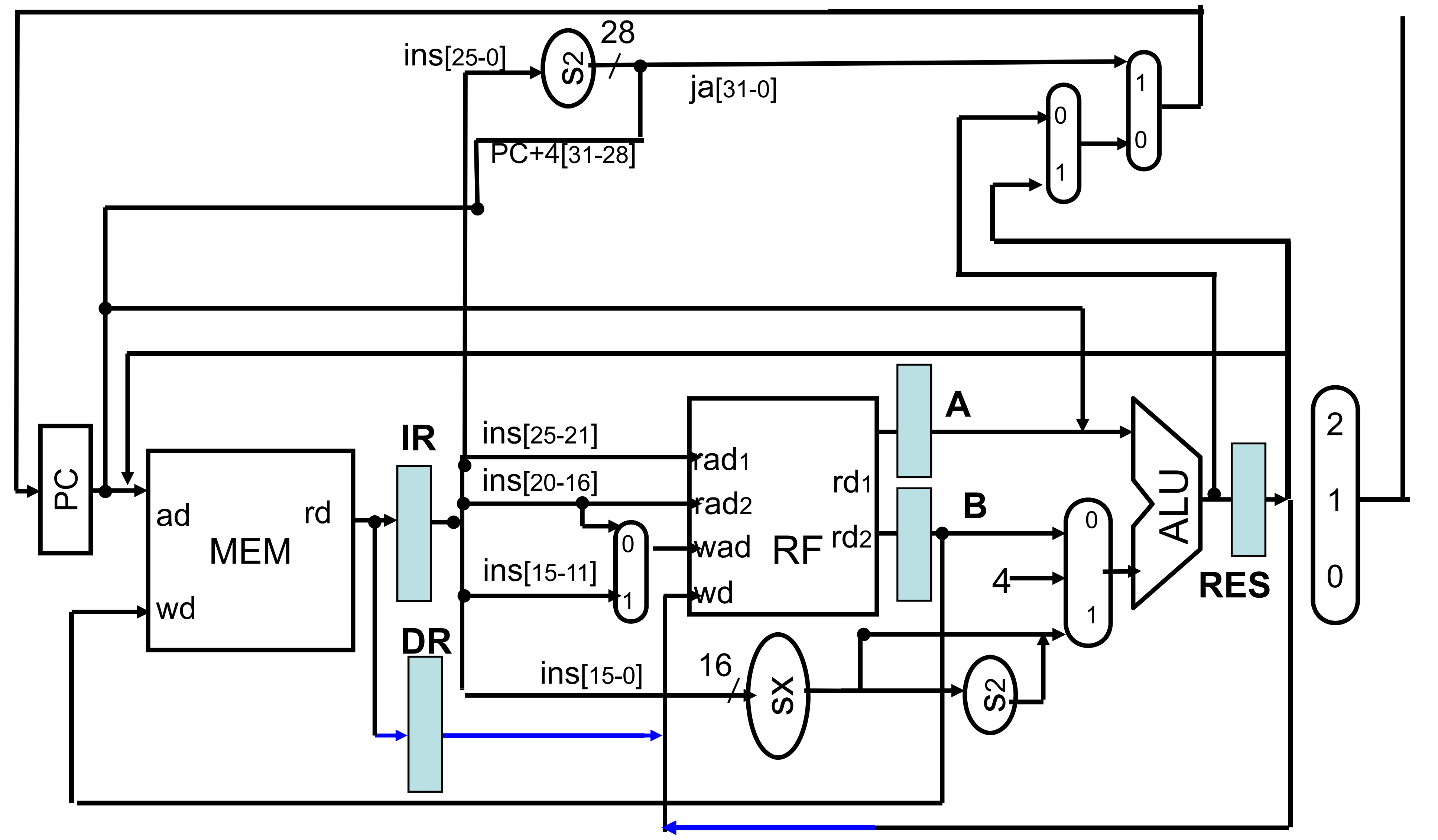
- The output of ALU be loaded in a register. We need one register at the output of the ALU.
- Two registers are placed at MEM Output UTPUT (IR and DR). We are placing two registers at RF OUTPUT (A and B) for Operand-1 and Operand-2.

- During the PC+4 calculation, address FROM ALU OUTPUT will flow through line YY.

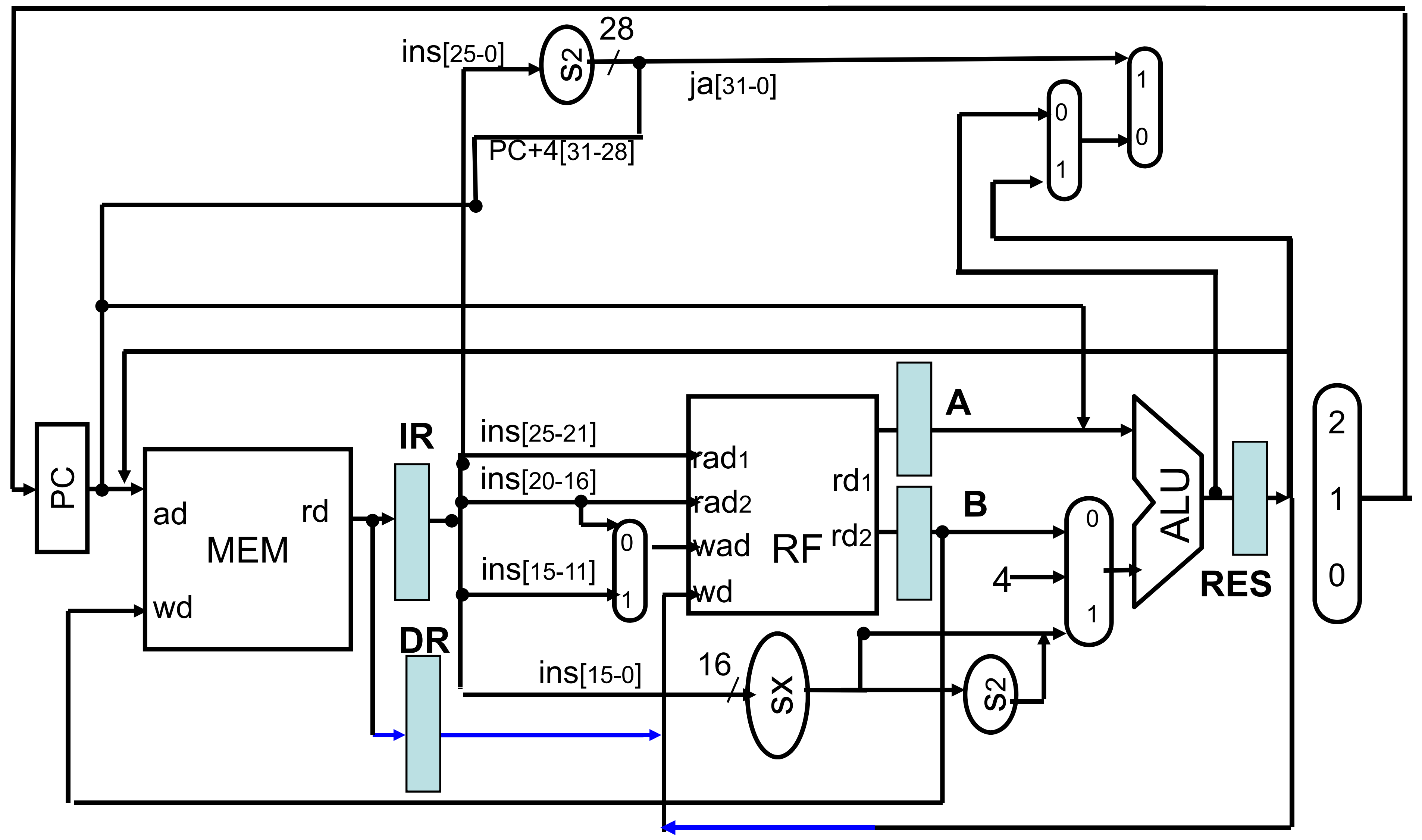
- One register “RES” is placed at output of ALU. RES will store output of R-TYPE instruction (ADD,SUB, SLT etc).

- Memory address after calculation for lw/sw will be resting in the “RES”. The effective branch address for BEQ instruction will be resting for the decision whether the branch is taken or not

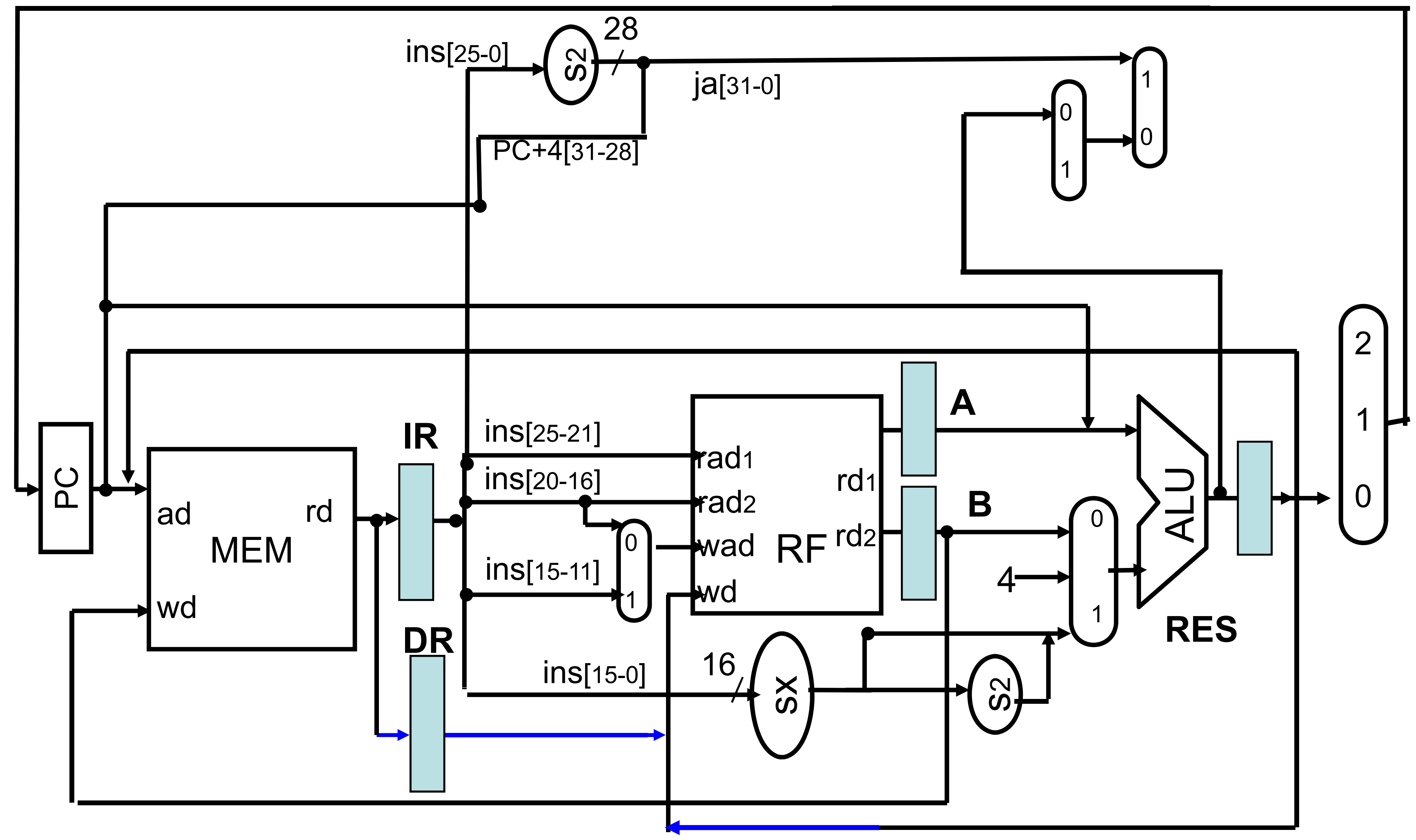
Rearrange PC Input Multiplexer



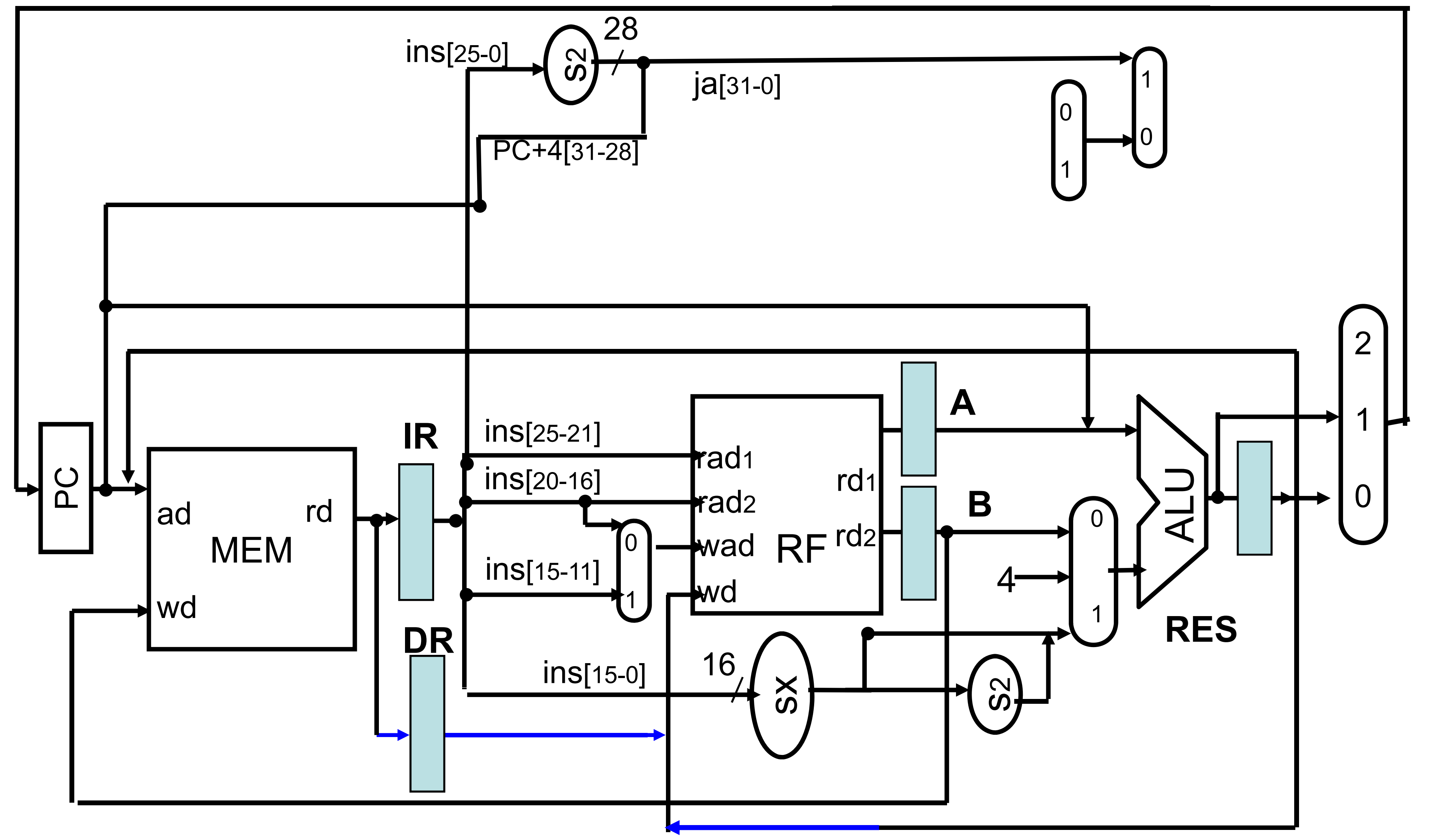
Rearrange PC Input Multiplexer



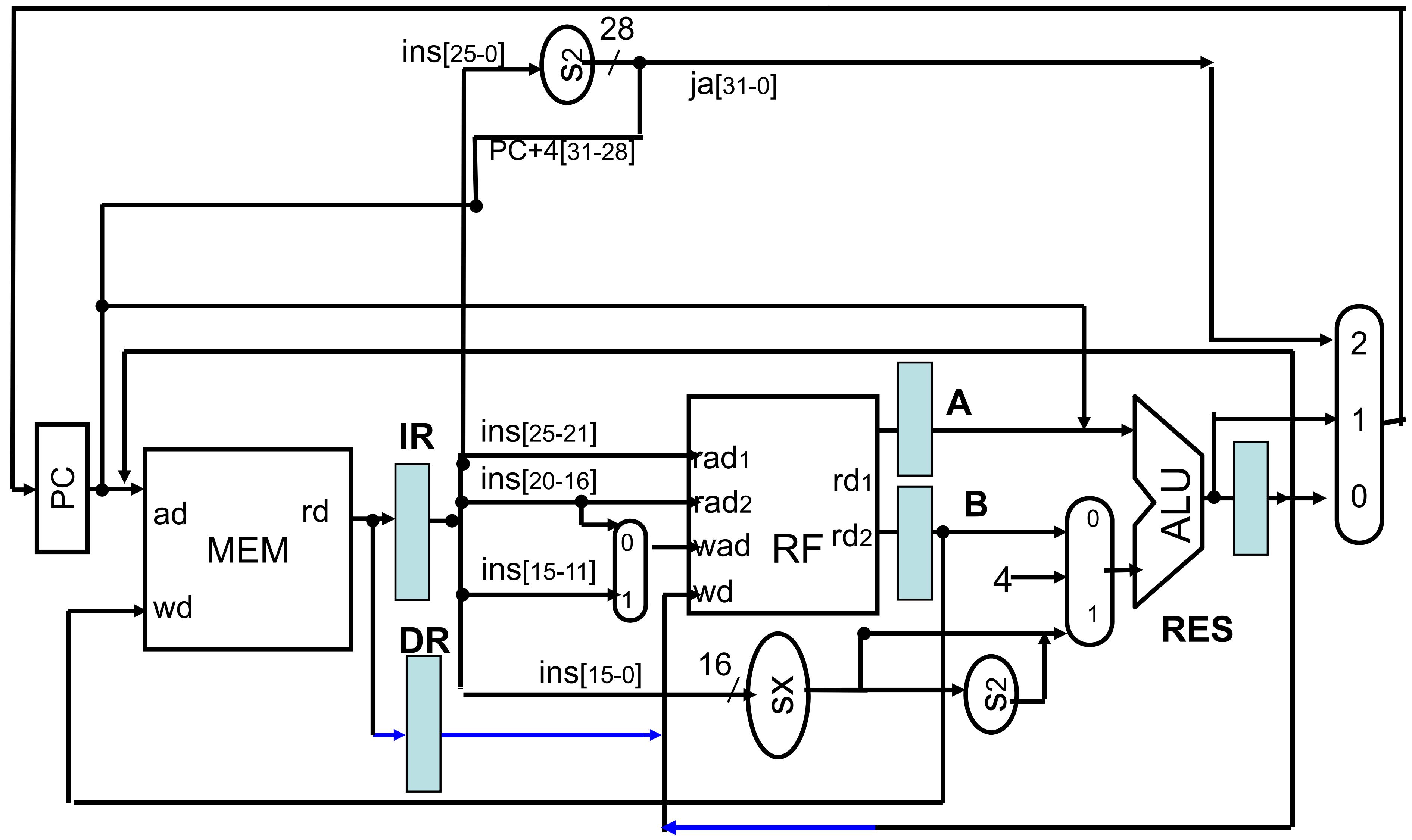
Rearrange PC Input Multiplexer



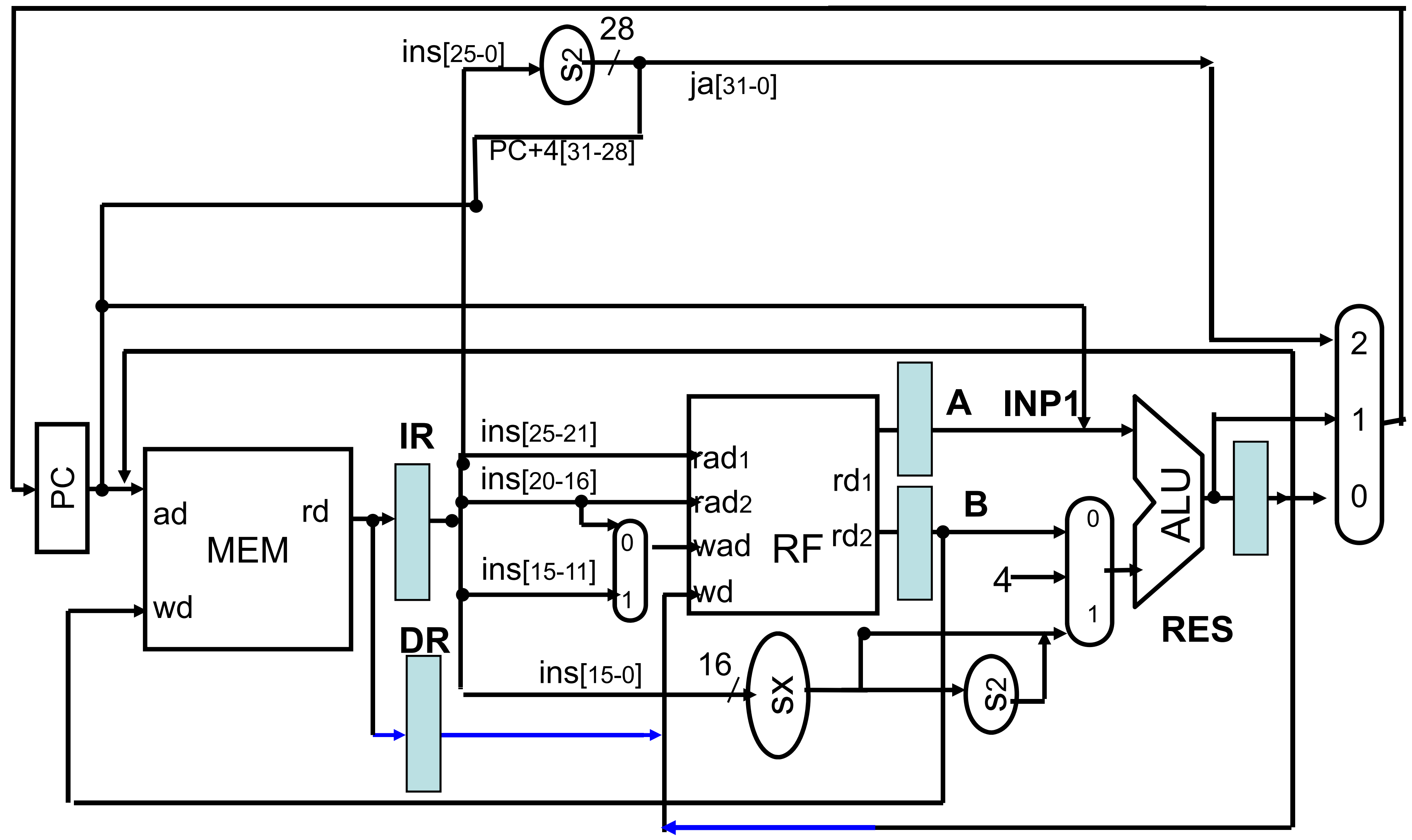
Rearrange PC Input Multiplexer



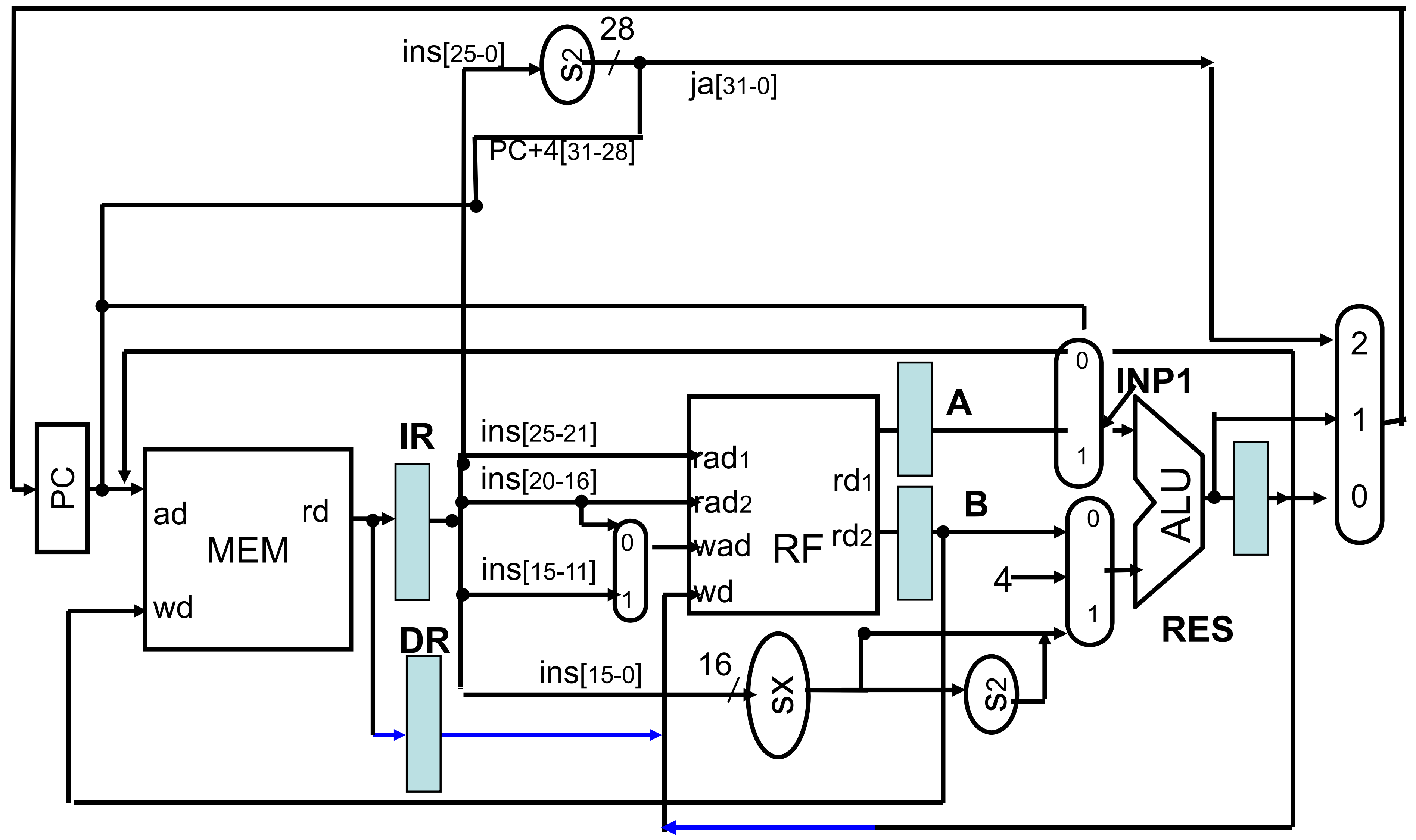
Rearrange PC Input Multiplexer



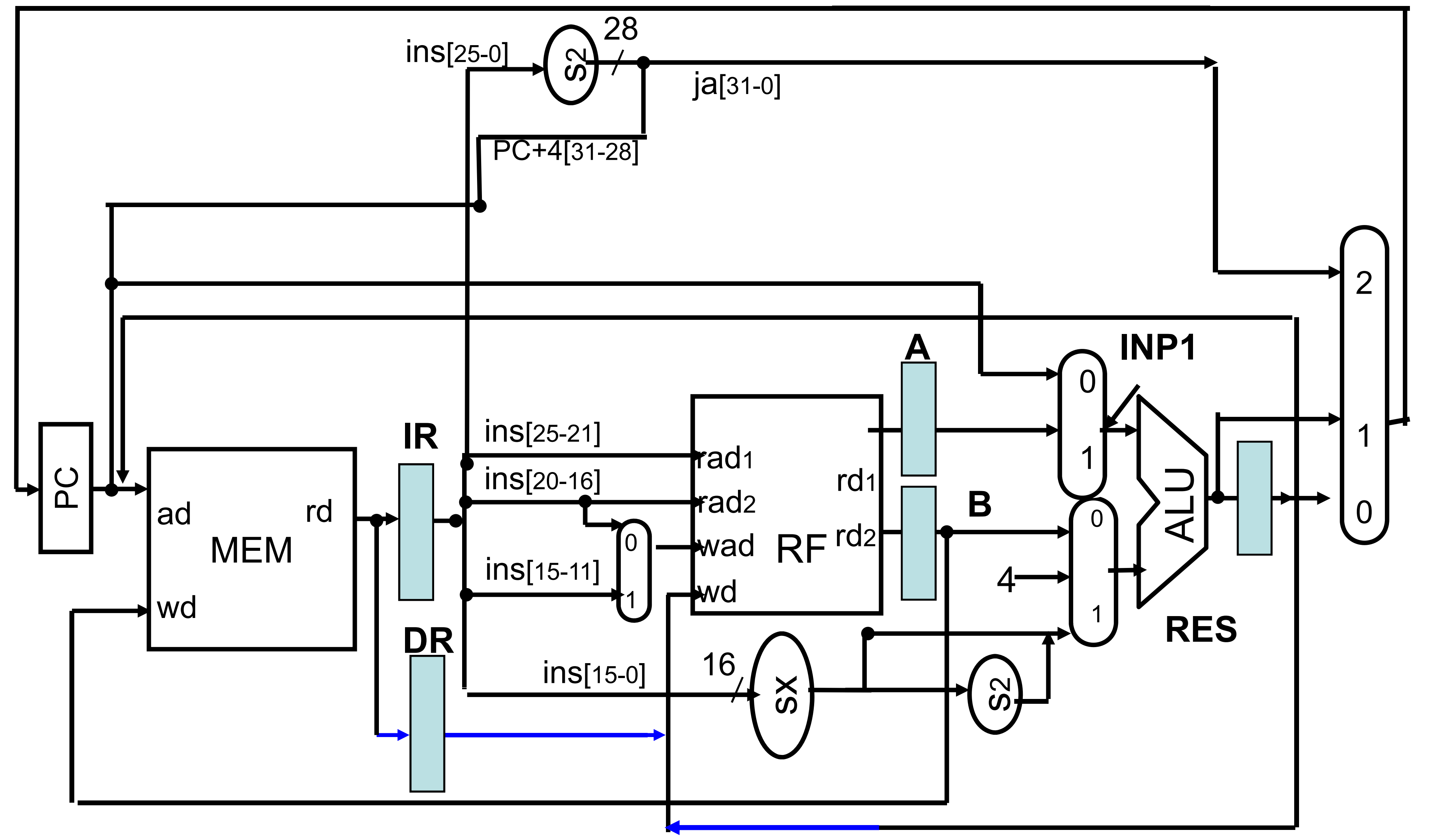
Introduce ALU INP1 Multiplexer



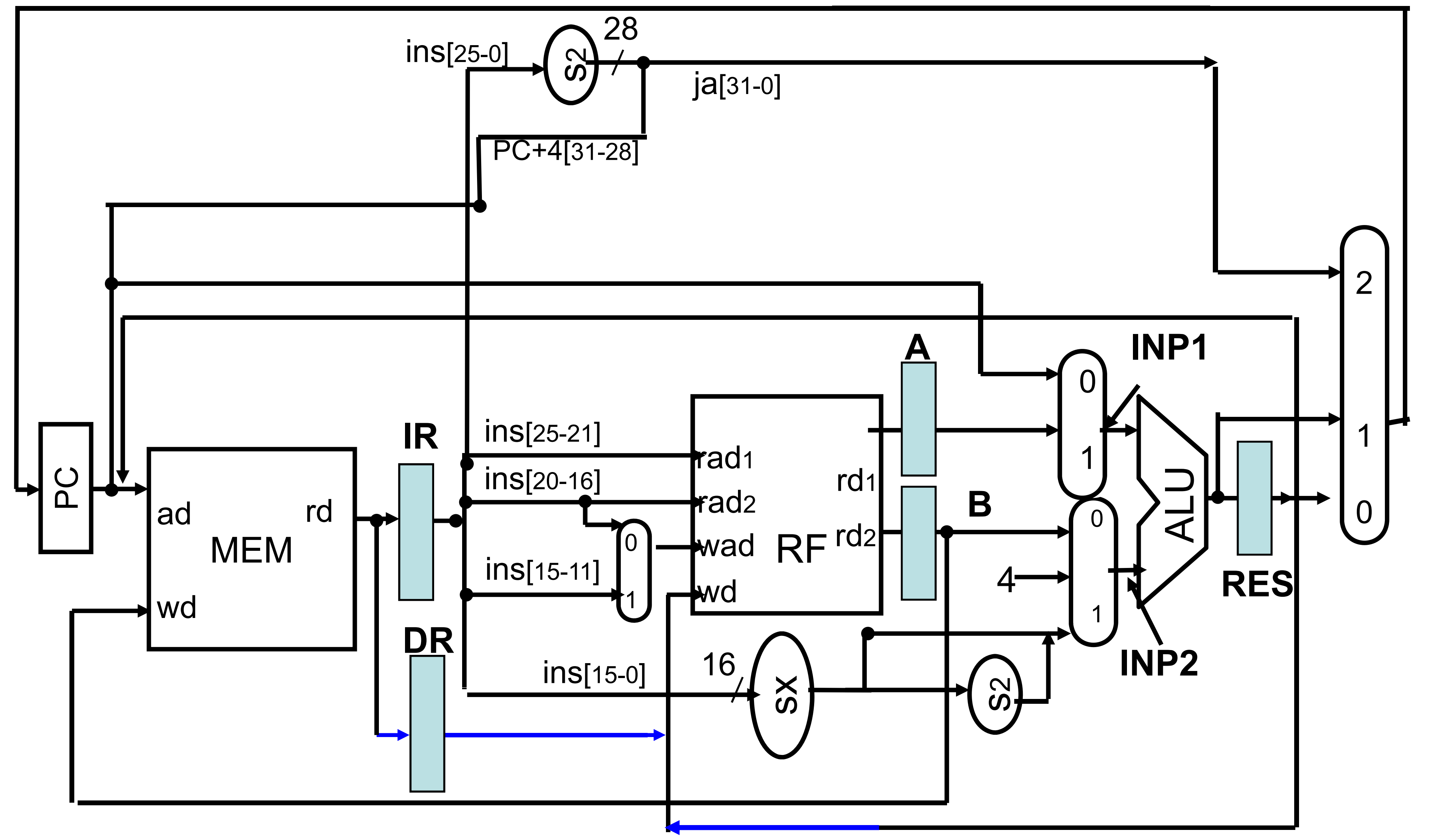
Introduce ALU INP1 Multiplexer



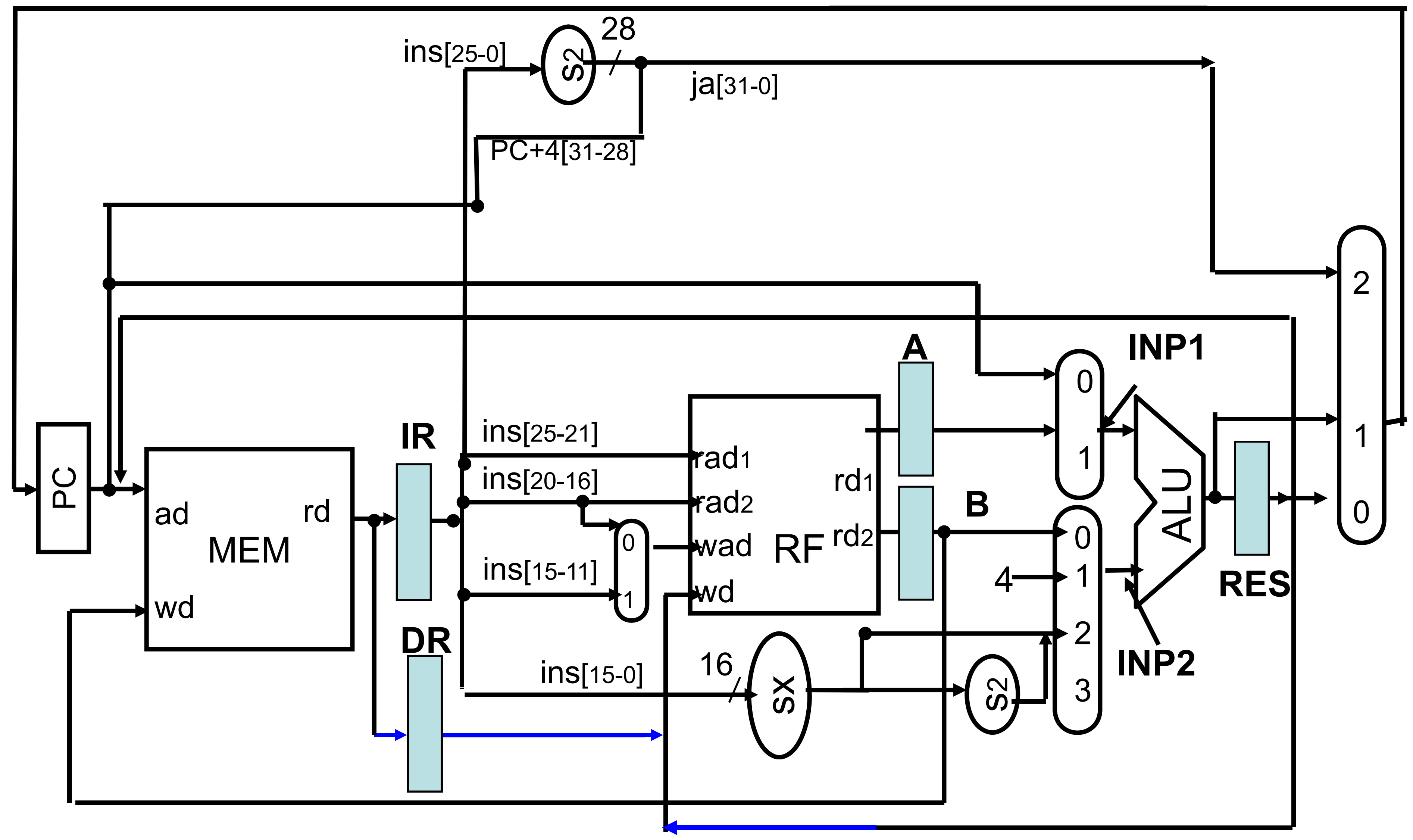
Introduce ALU INP1 Multiplexer



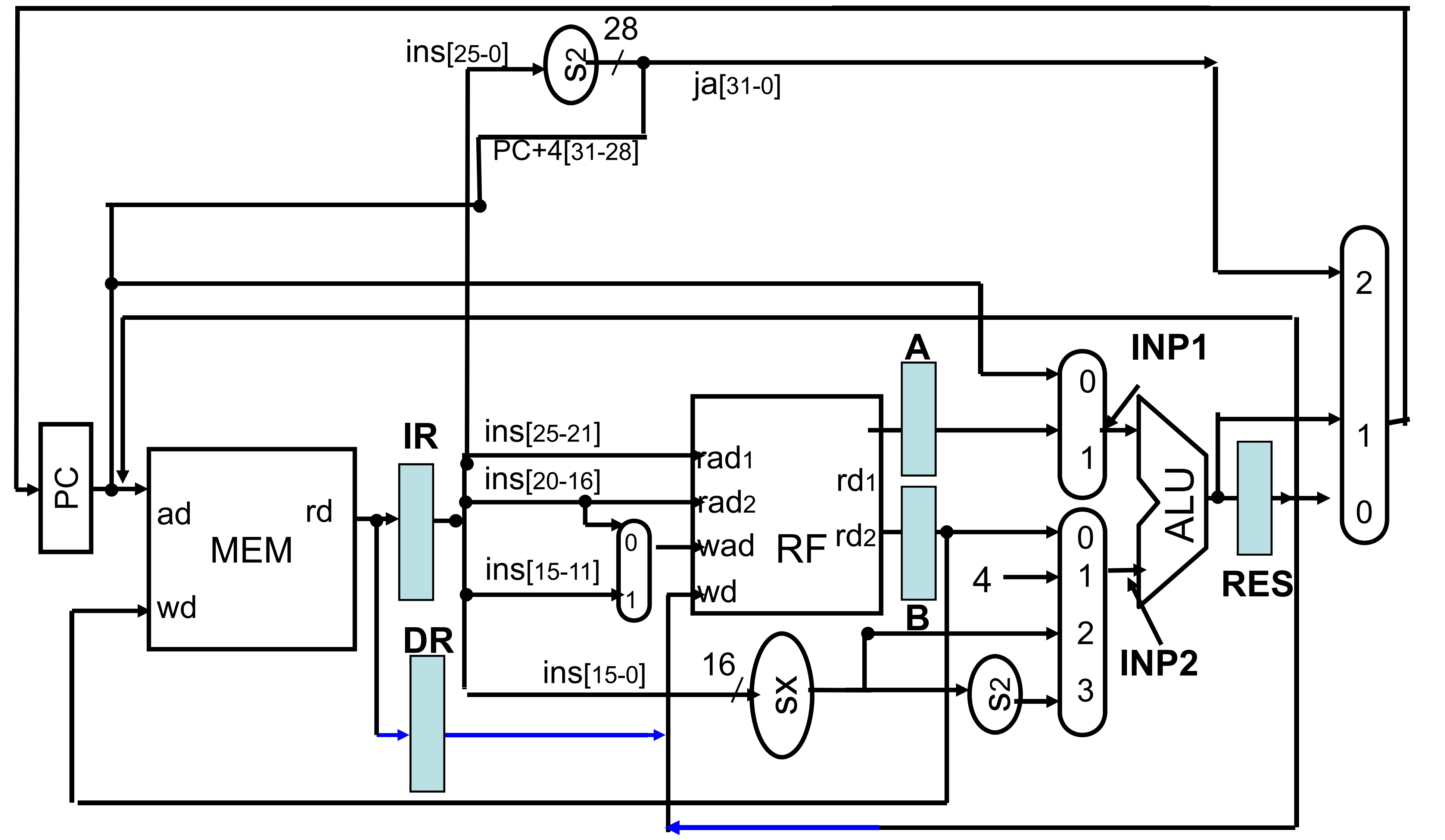
Introduce ALU INP2 Multiplexer



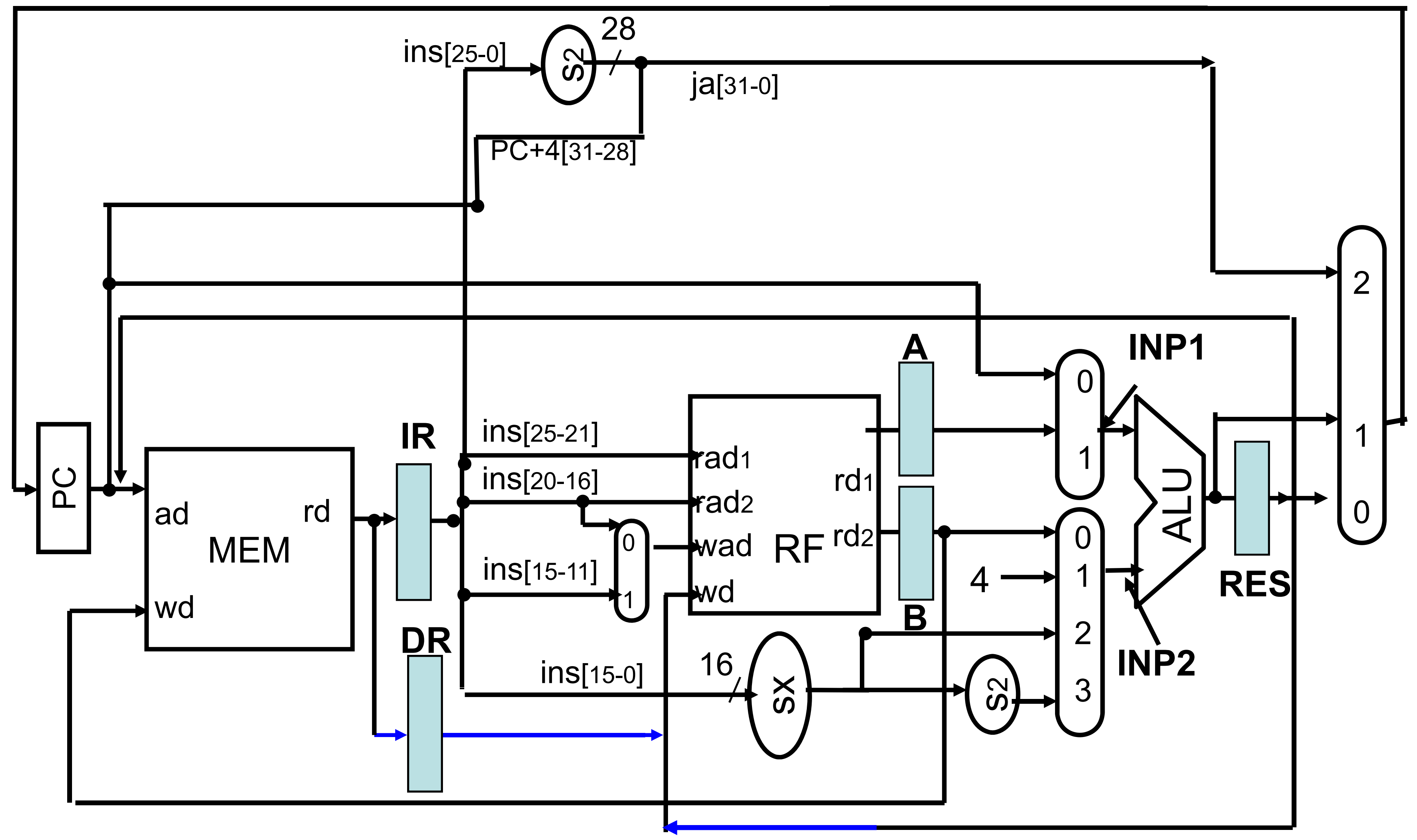
Introduce ALU INP2 Multiplexer



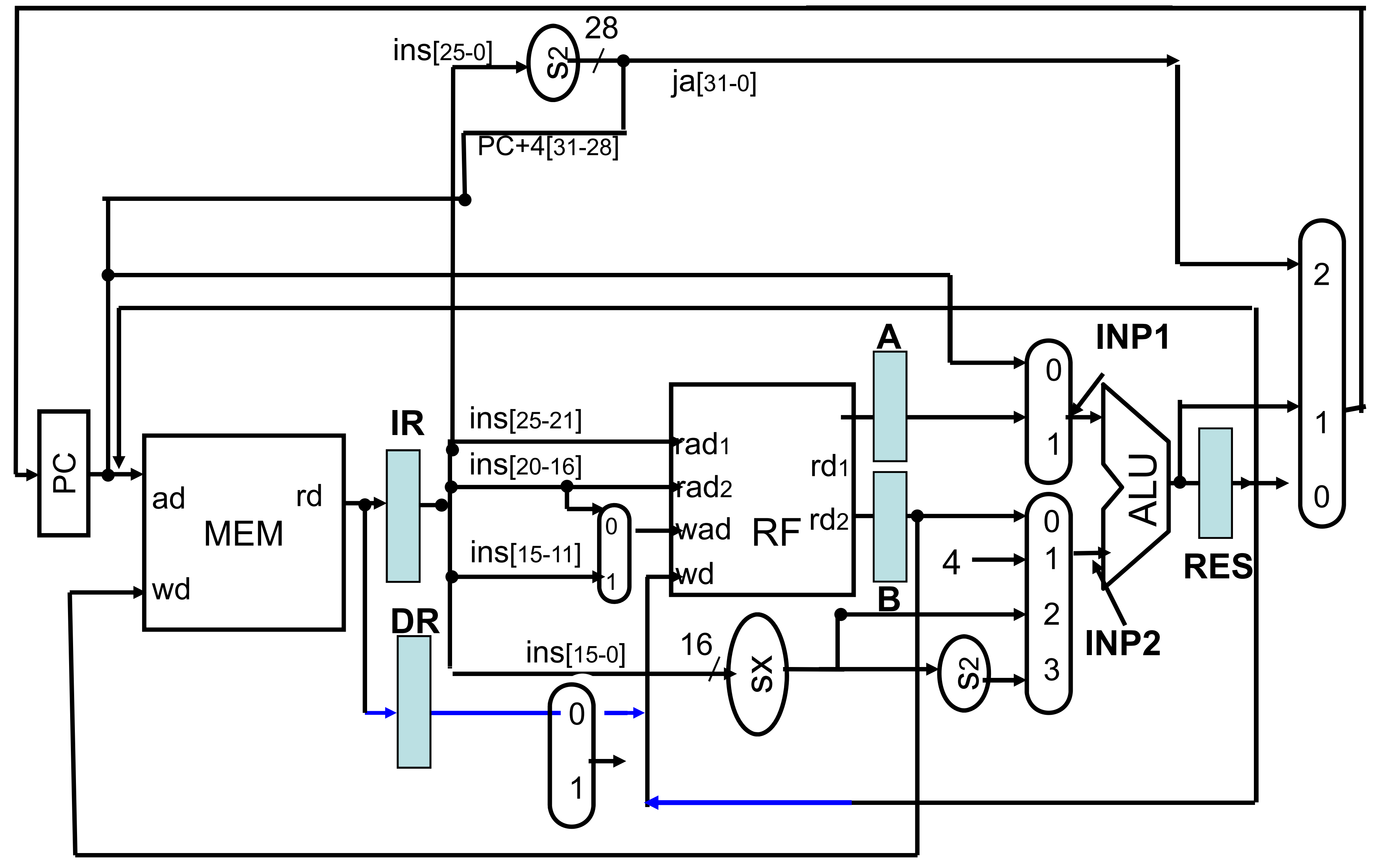
Introduce ALU INP2 Multiplexer



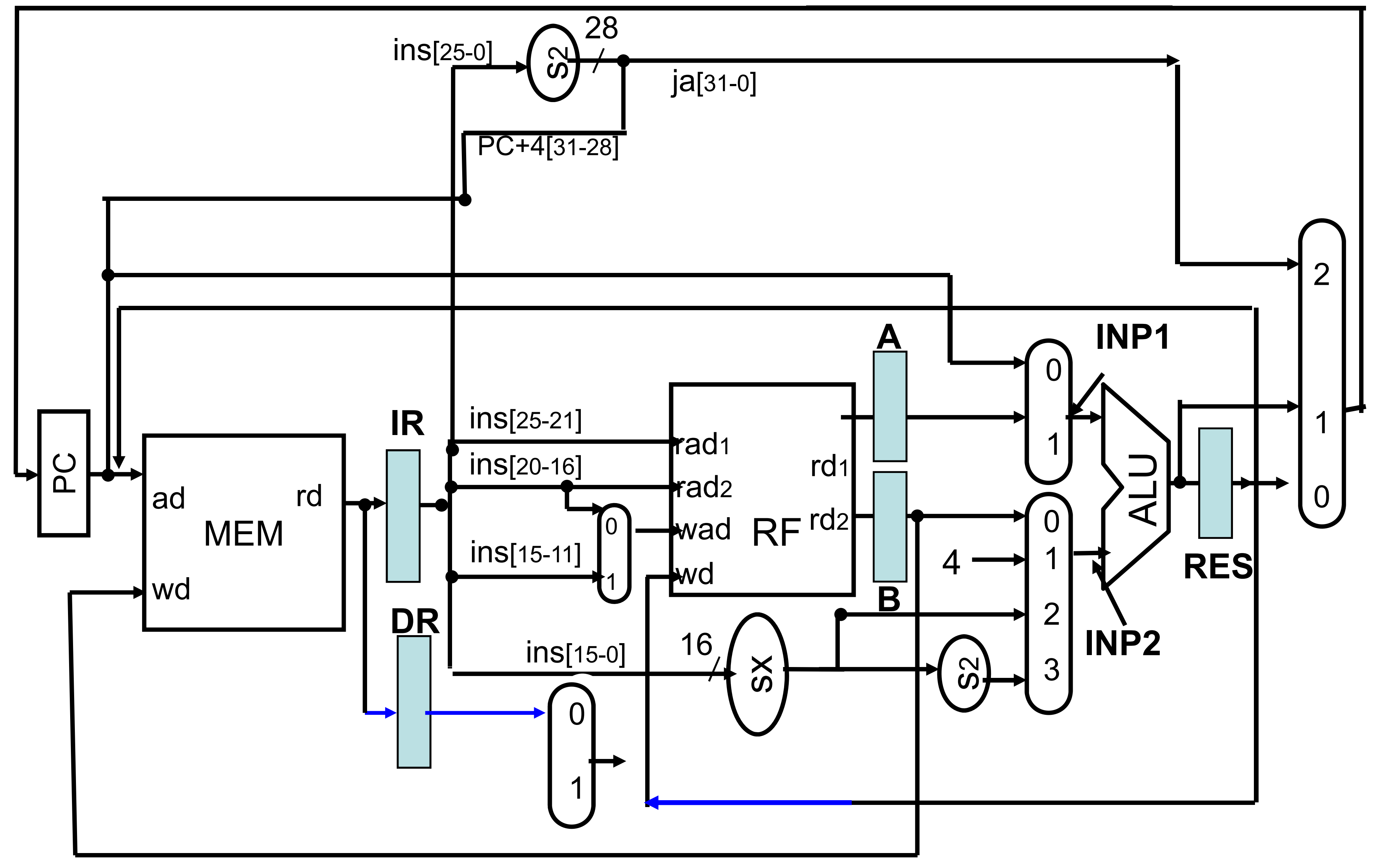
Introduce RF INP Multiplexer



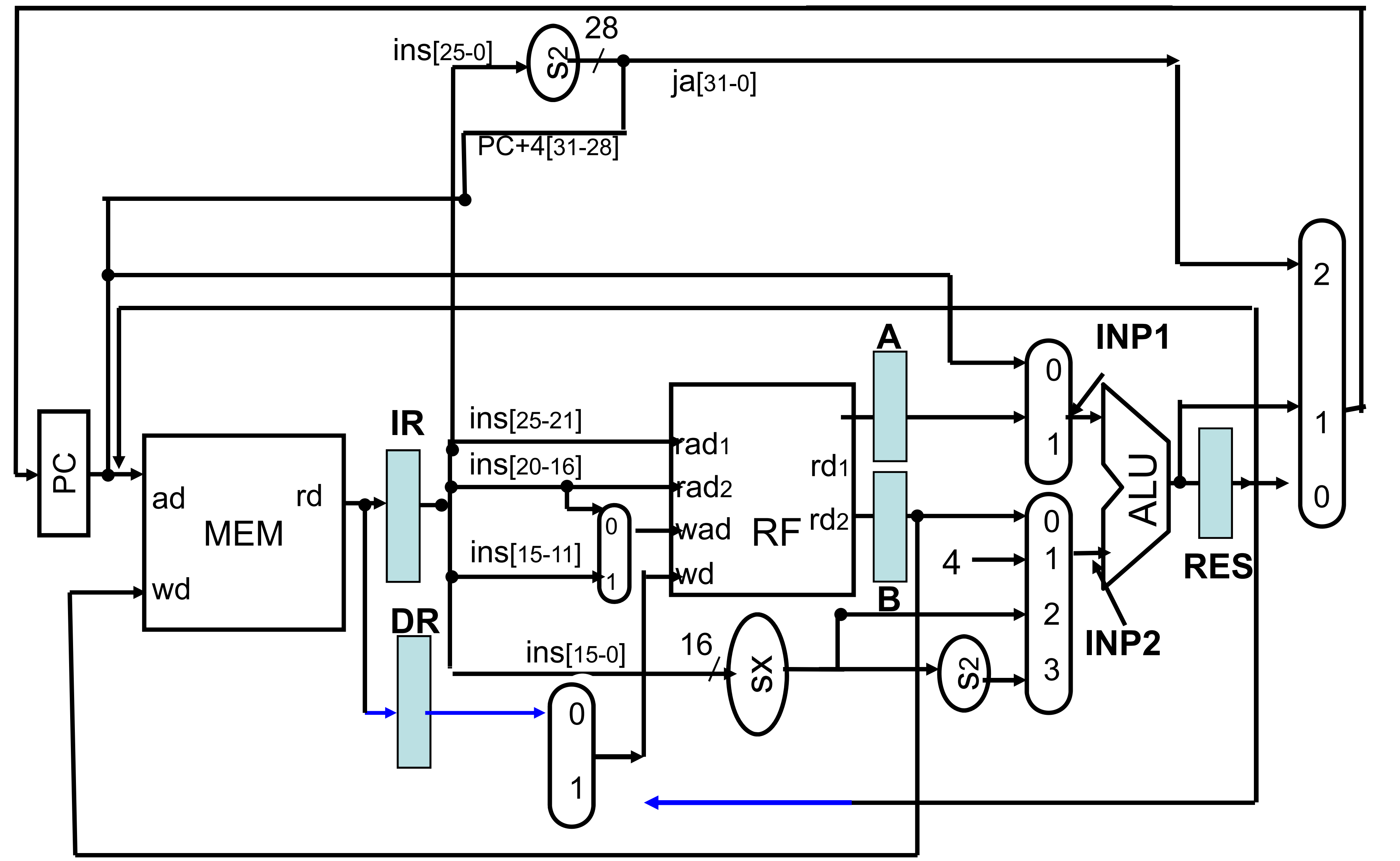
Introduce RF INP Multiplexer



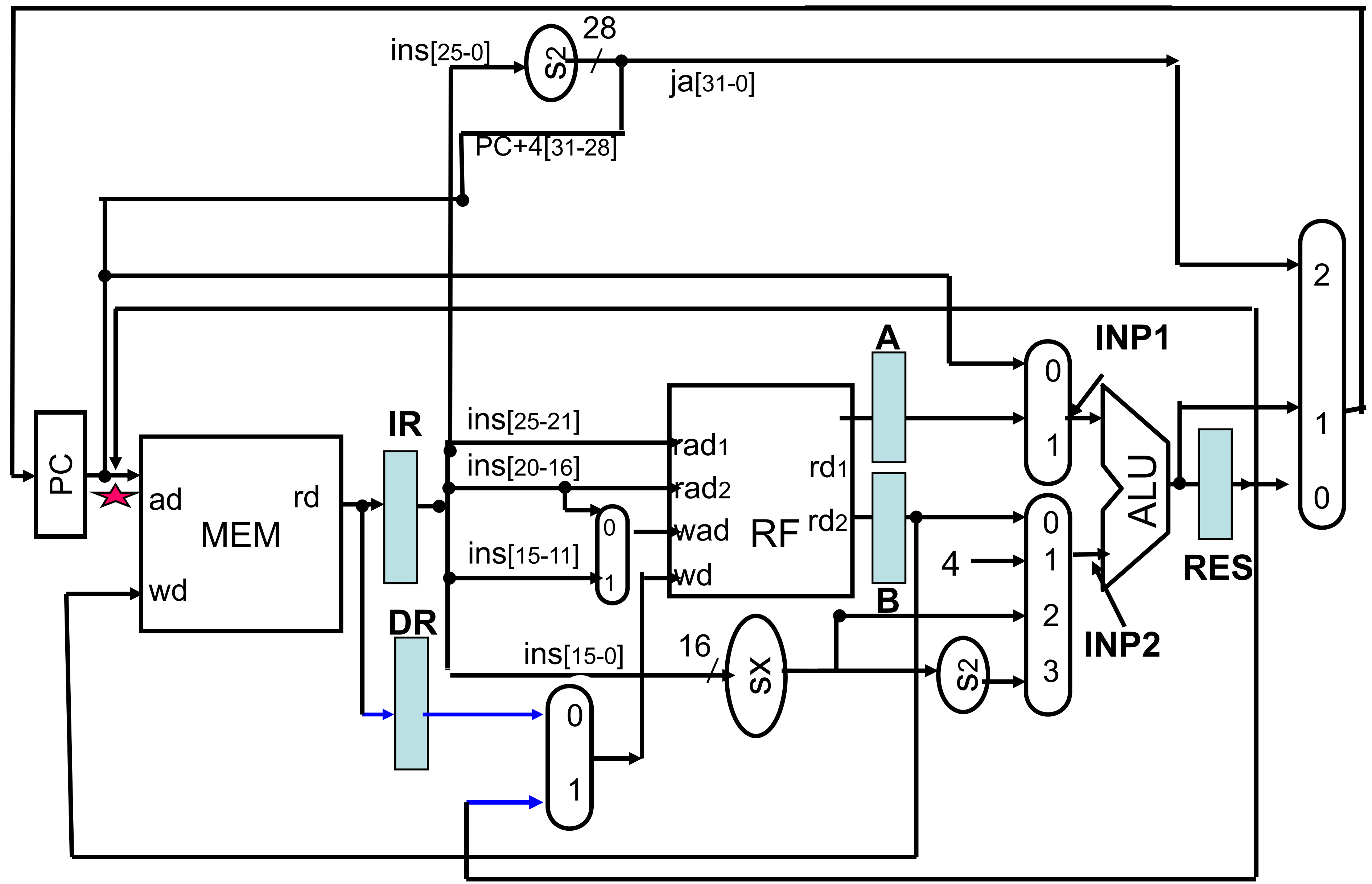
Introduce RF INP Multiplexer



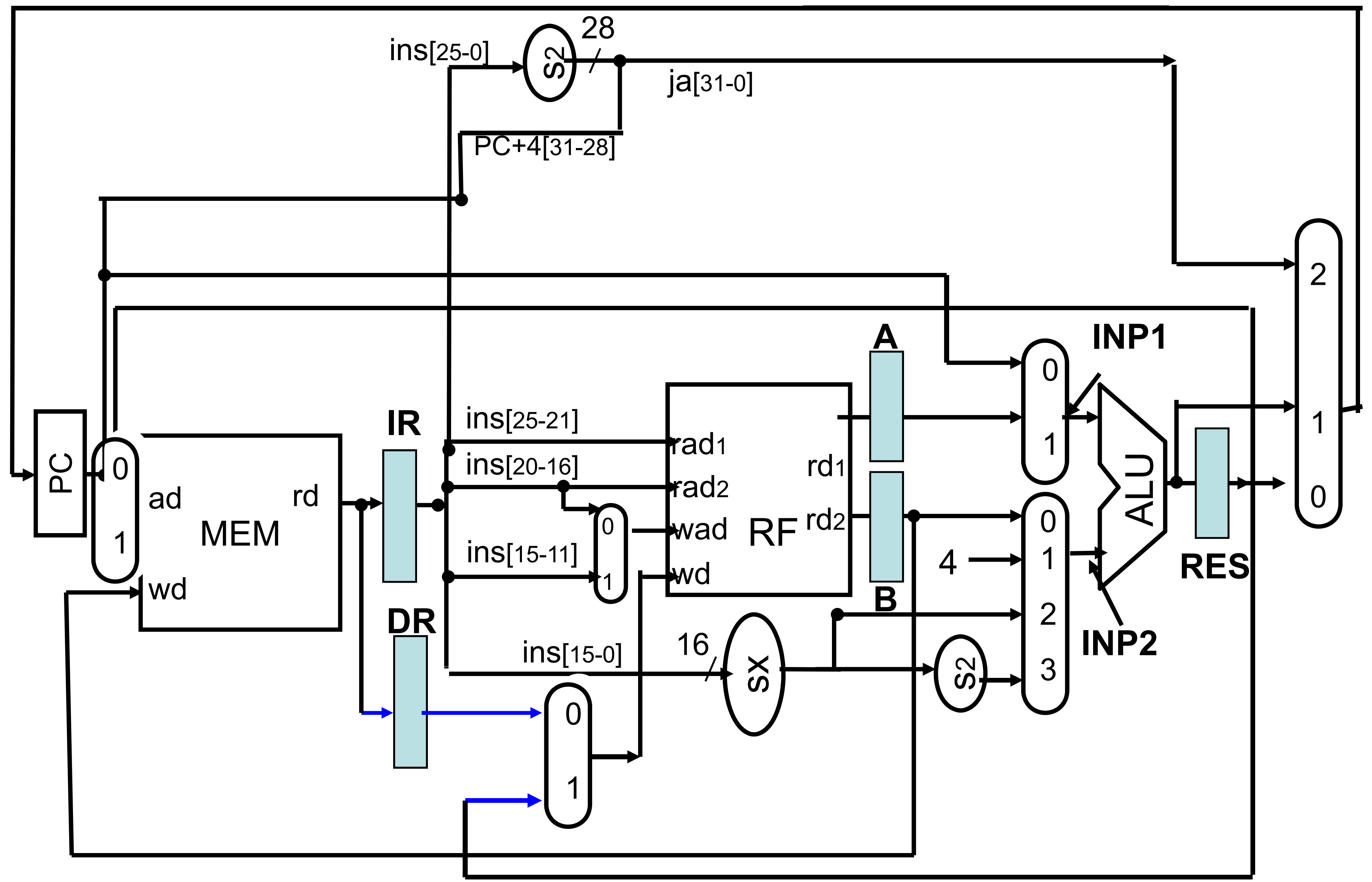
Introduce RF INP Multiplexer



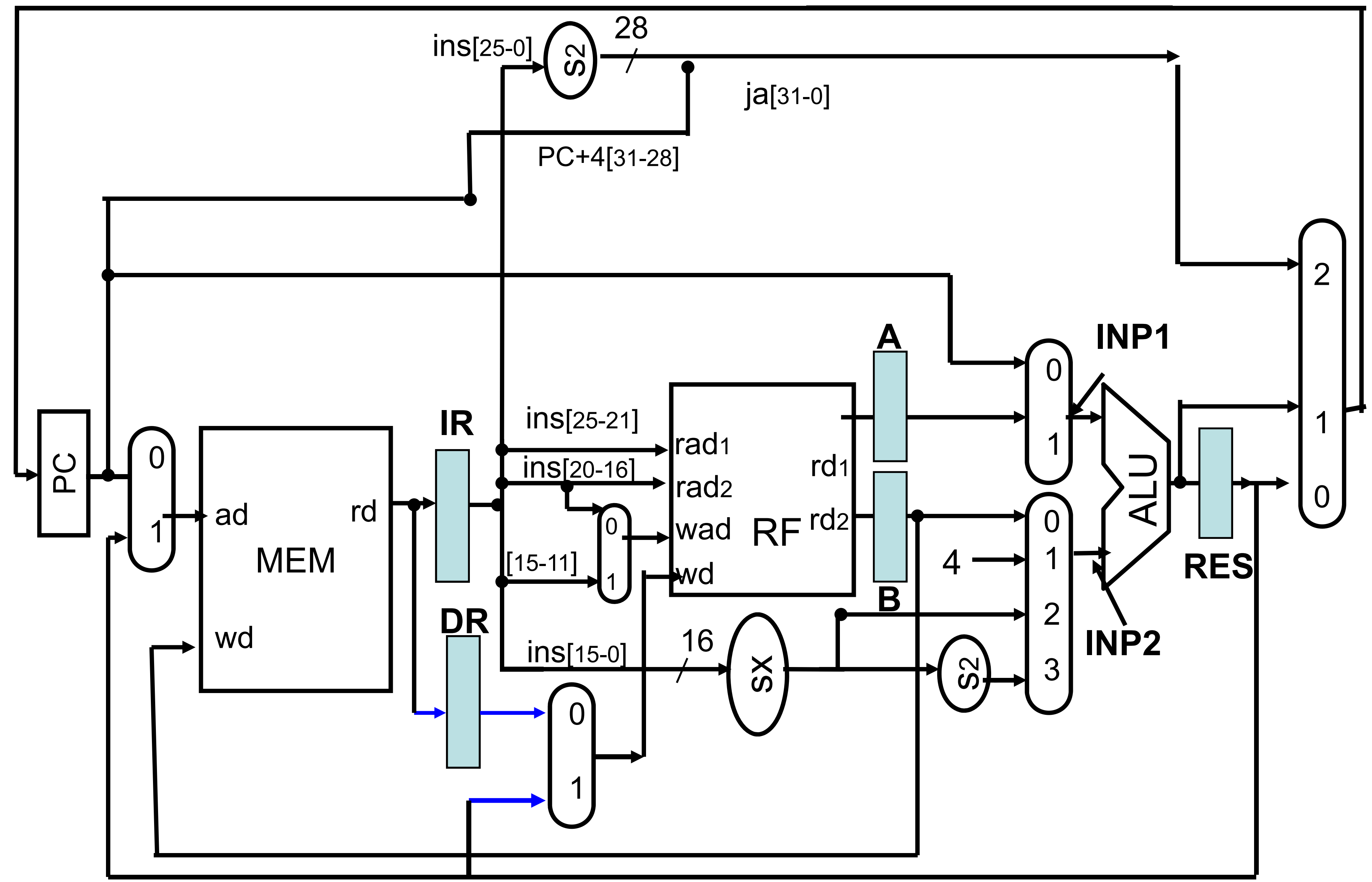
Introduce Mem INP Multiplexer



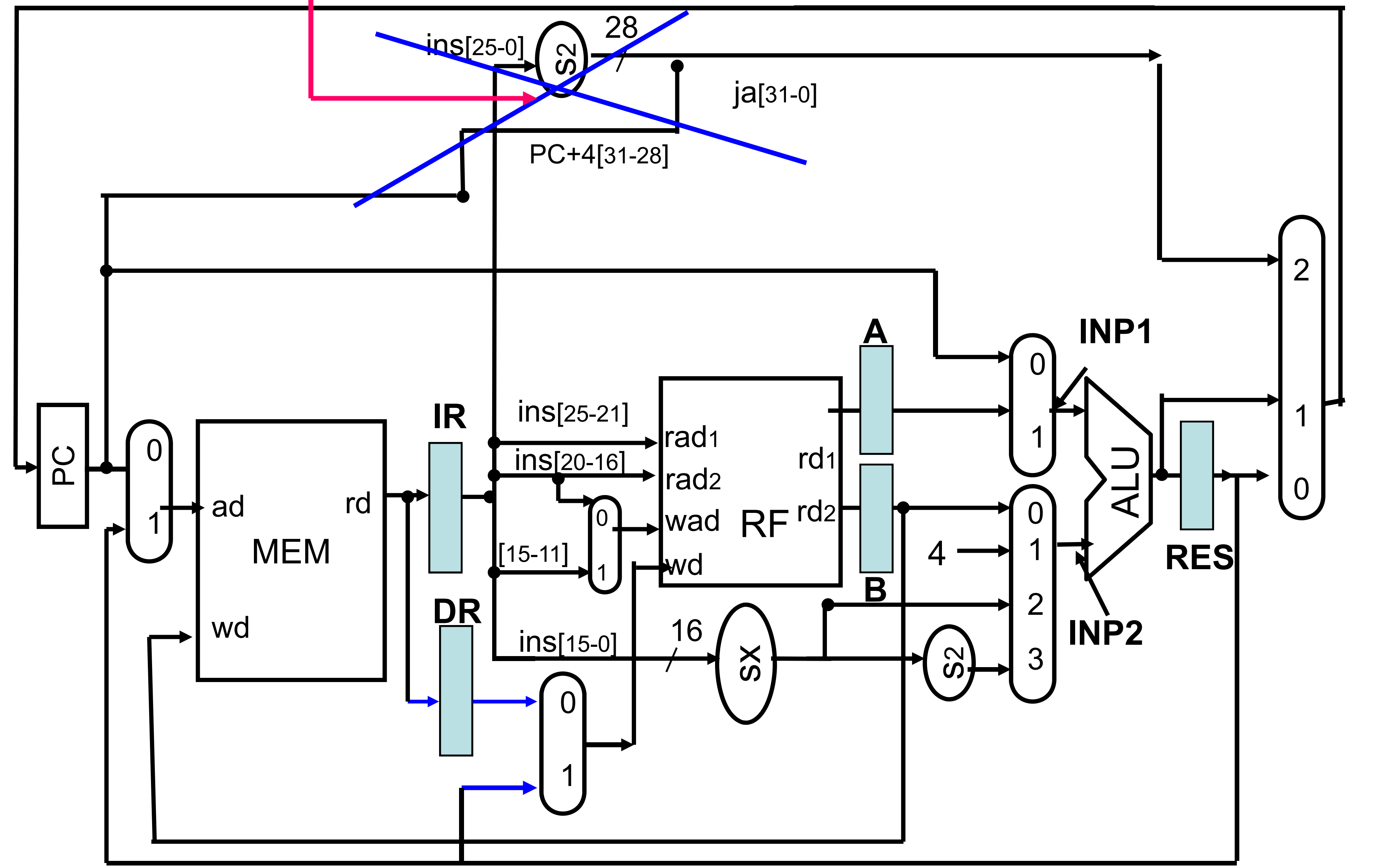
Introduce Mem INP Multiplexer



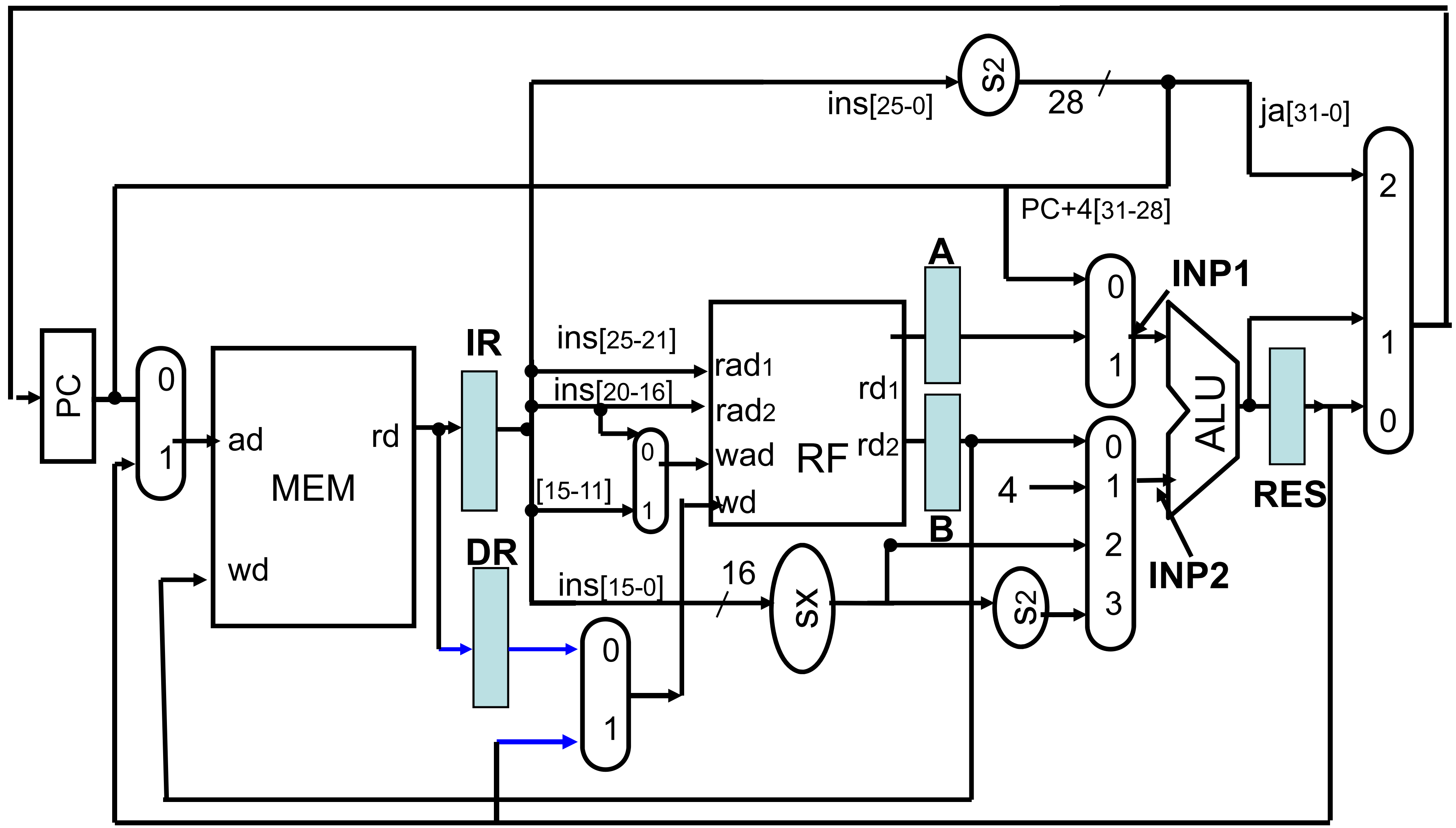
Introduce Mem INP Multiplexer



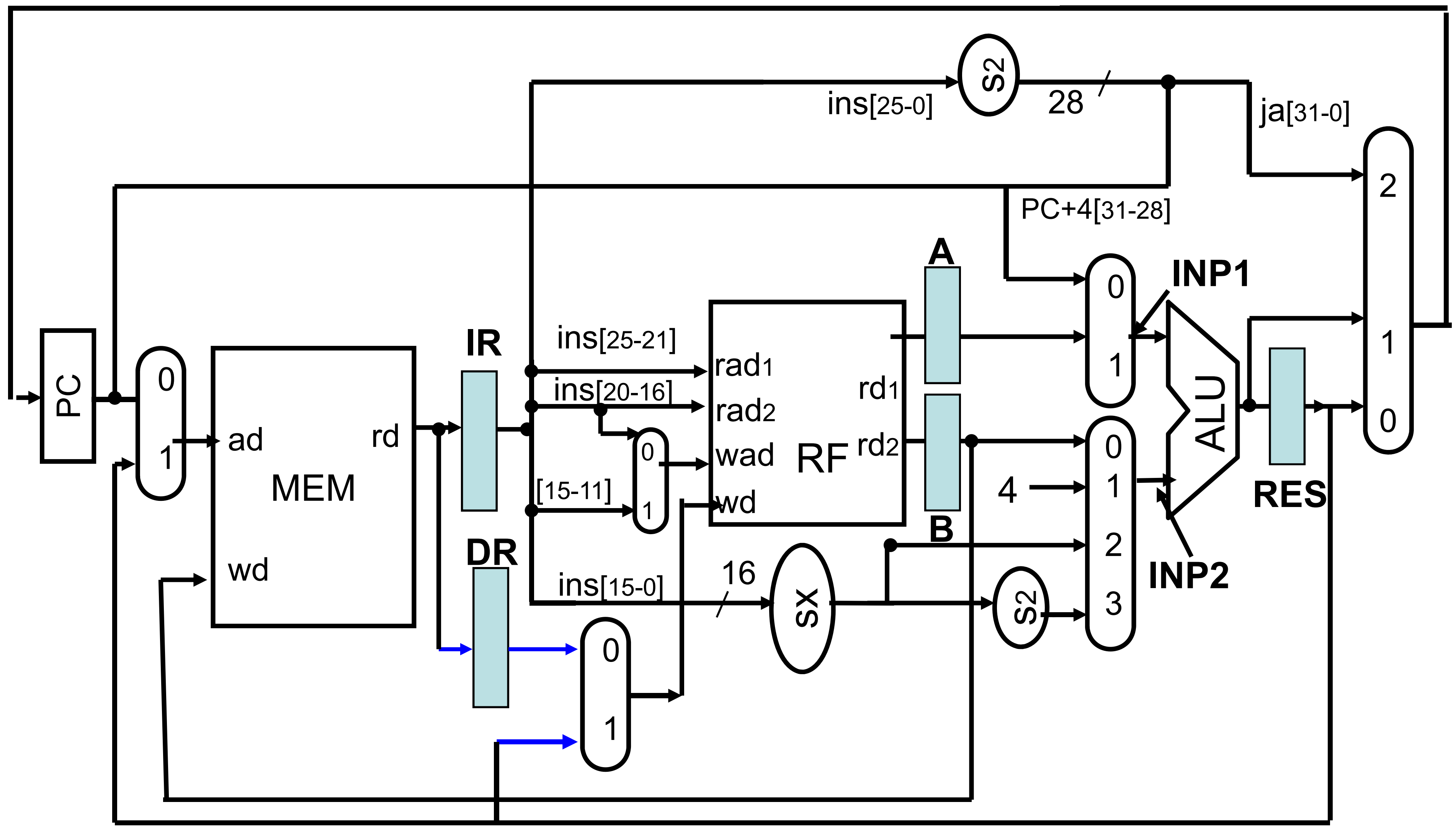
Rearrange Diagram



Rearrange Diagram



Complete Multi-Cycle DataPath



Summary

- Problems in Single cycle approach have been discussed.
- Determination of clock period for single cycle design and multi-cycle design
- Step-step design of Multi-cycle MIPS Processor

Assignments

1. Determine the clock period of Single Cycle Design of MIPS Processor
2. What are problems in the Single Cycle Design of MIPS Processor?
3. How do you rectify these problems in Single Cycle Design of MIPS Processor?
4. How is the clock period determined in Multi-Cycle Design of MIPS Processor?
5. Discuss the techniques for
   a). Merging Instruction memory and Data memory to make a single memory

   b). Elimination of adder-1 and adder-2 to minimize the Hardware cost in the Multi-cycle design

   c). Addition of stage registers

   d). Introduction of Multiplexers to avoid the conflicts in the cycle design.

Assignments

6. Why are the stage registers and multiplexers used in the Multi-cycle Design.
7. Derive the complete multi-cycle Data Design

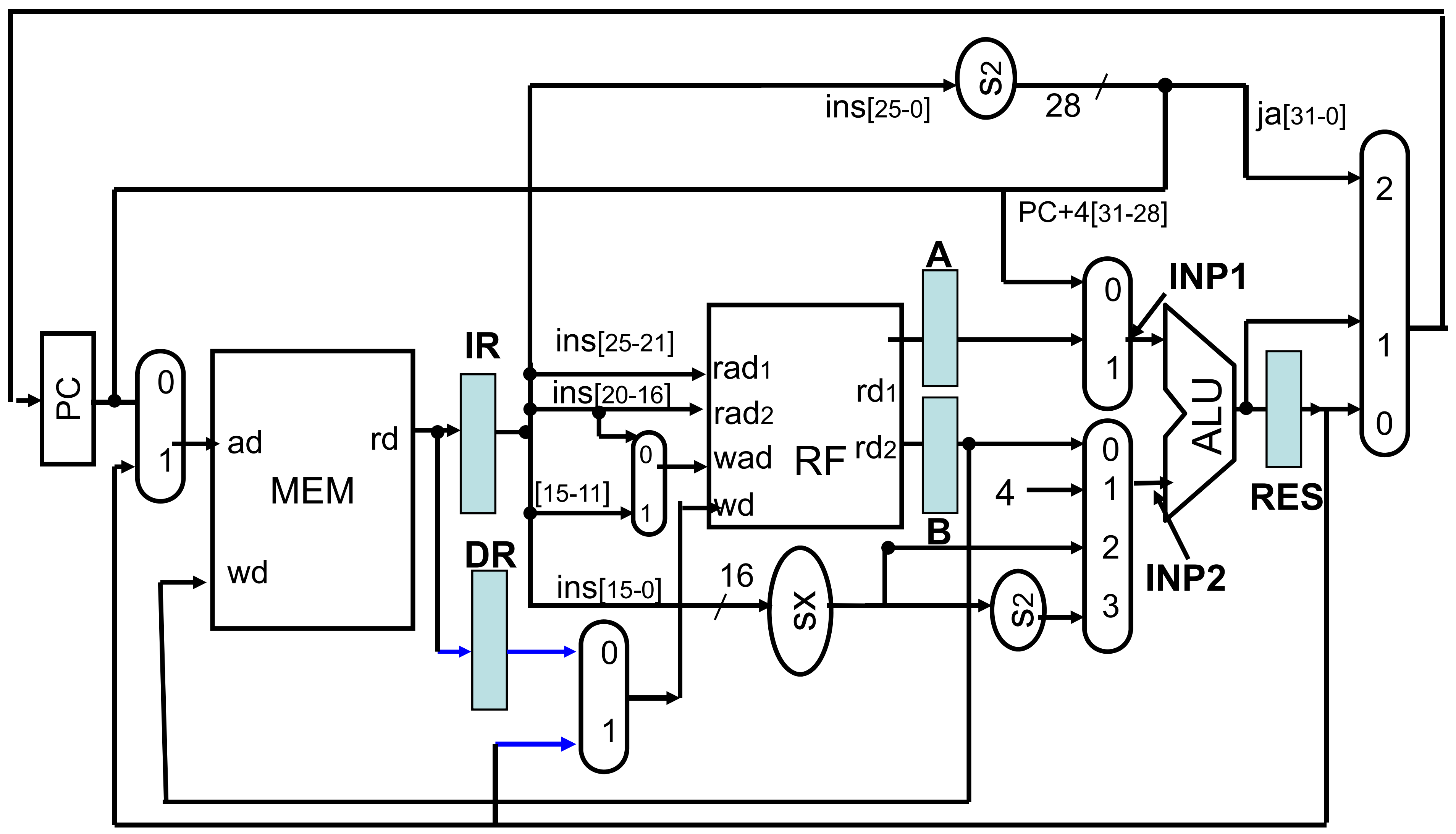
Assignments

6. Why are the stage registers and multiplexers used in the Multi-cycle Design.
7. Derive the complete multi-cycle Data Design

Complete Multi-Cycle DataPath

Complete Multi-Cycle DataPath

Complete Multi-Cycle Datapath implementing different MIPS instructions

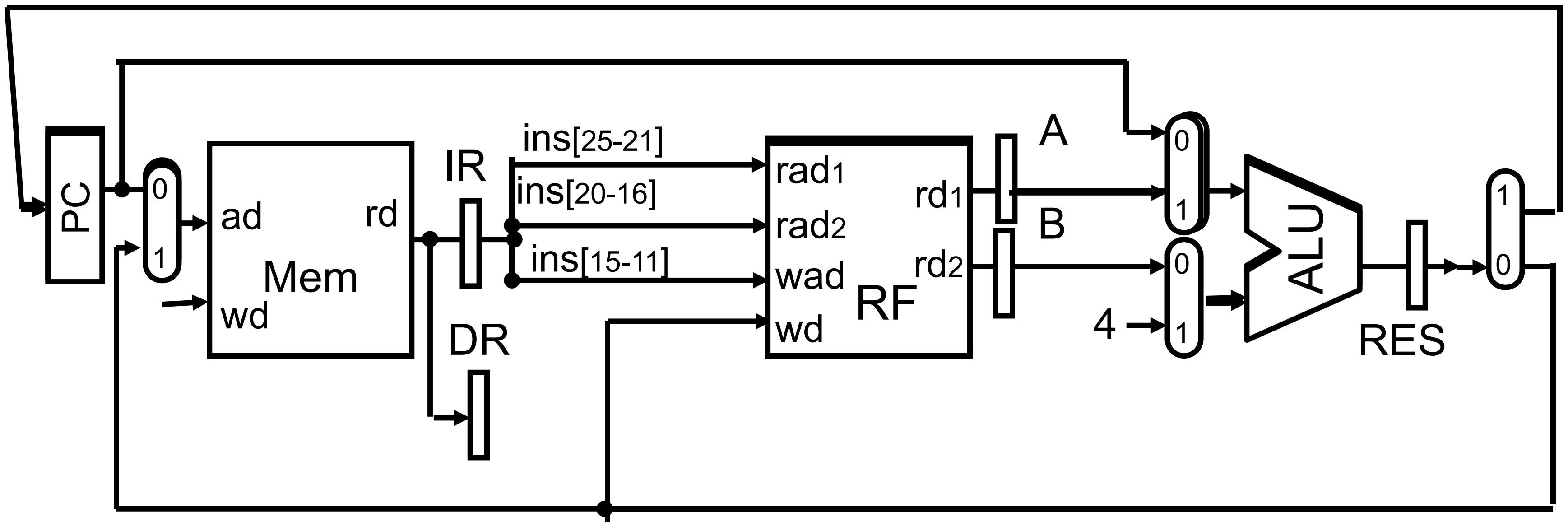


Next, We have presented the component wise datapath implementation of Each Instruction type

R-Type Instruction

	25-21	20-16	15-11		
op	rs	rt	rd	shamt	funct

According to PC address, Instruction is fetched from Memory and loaded into IR and is decoded. Simultaneously, PC value is replaced by (PC+4) in the first cycle. Two registers of RF are identified by 5 bit address lines Ins (25-21) and Ins (20-16) respectively and two source registers rs and rt are accessed and read. Through two ports rd1 and rd2, two operands (rs and rt) are fed to ALU respectively. As per opcode and funct bits, operation will be performed in ALU and Result will be loaded in 'RES' temporarily. Finally result will be loaded through 'wd' port into Register (rt) of RF identified by 5 address bit ins(15-11) supplied through ins(15-11).

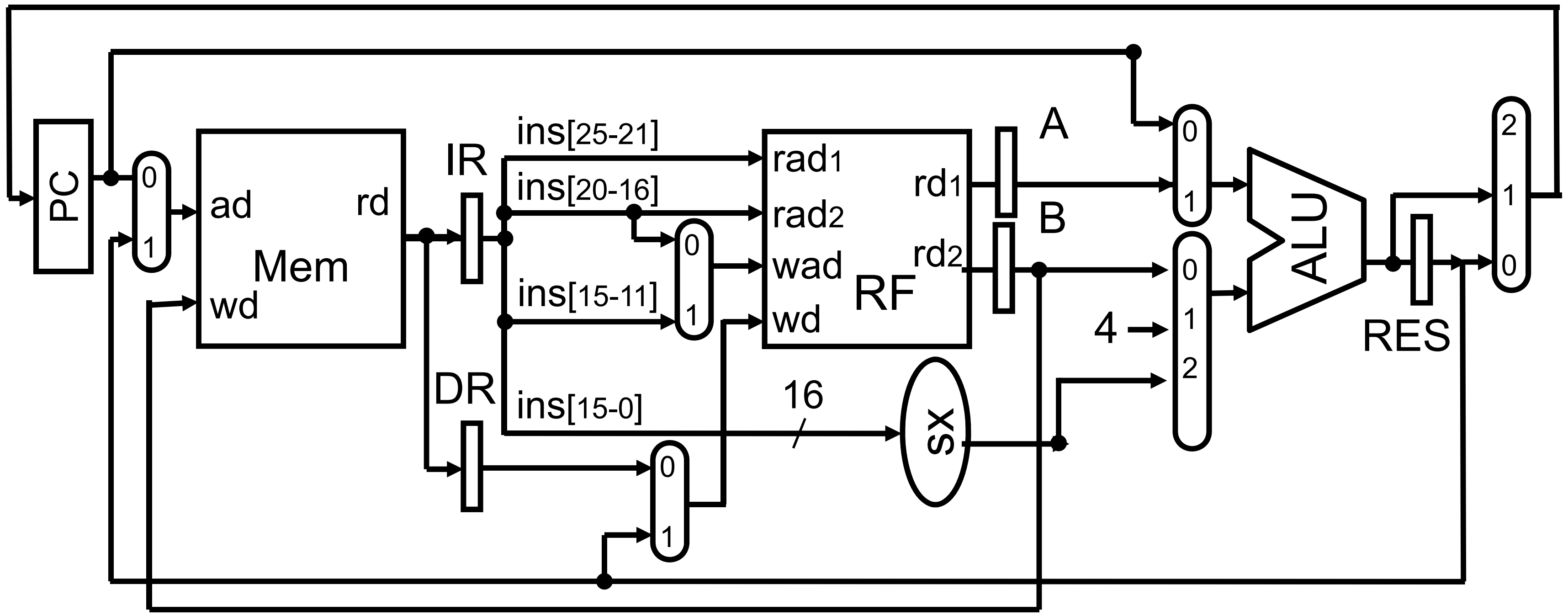


R-Type and SW/LW Instruction

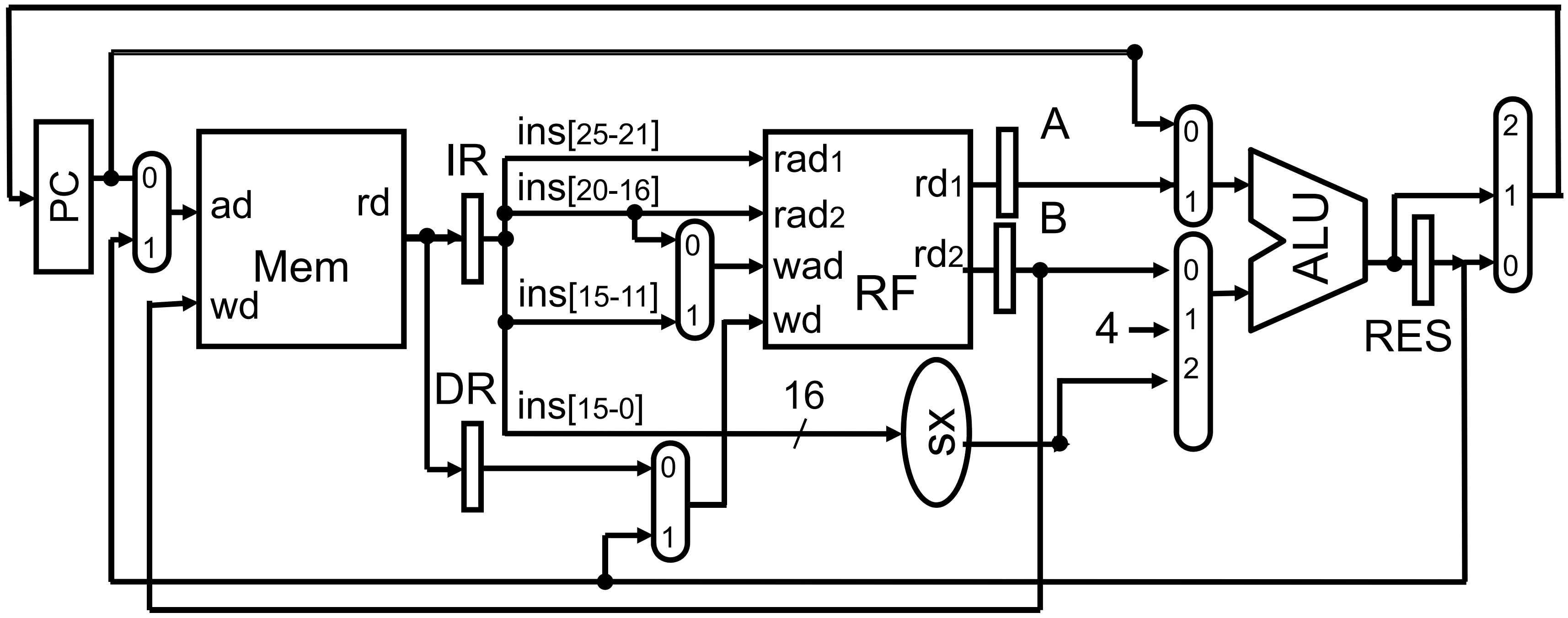
I- Type

	25-21	20-16	15-0
Op	rs	rt	16 bit number

According to PC address, Instruction is fetched from Memory and loaded into IR and is decoded. Simultaneously, PC value is replaced by (PC+4) in the first cycle. The “rs” register of RF is identified by 5 bit address lines Ins (25-21) and is fed to upper input of the ALU and 16 bit Ins (15-0) offset is fed to sign extended unit (SX) to convert 32-bit extended value which is fed to ALU through 3-input MUX (select input -2) and base address saved in registers rs and sign-extended value will be added in ALU to give effective address. Through two ports rd1 and rd2, two operands (rs and rt) are fed to ALU.



R-Type and LW/SW Instruction

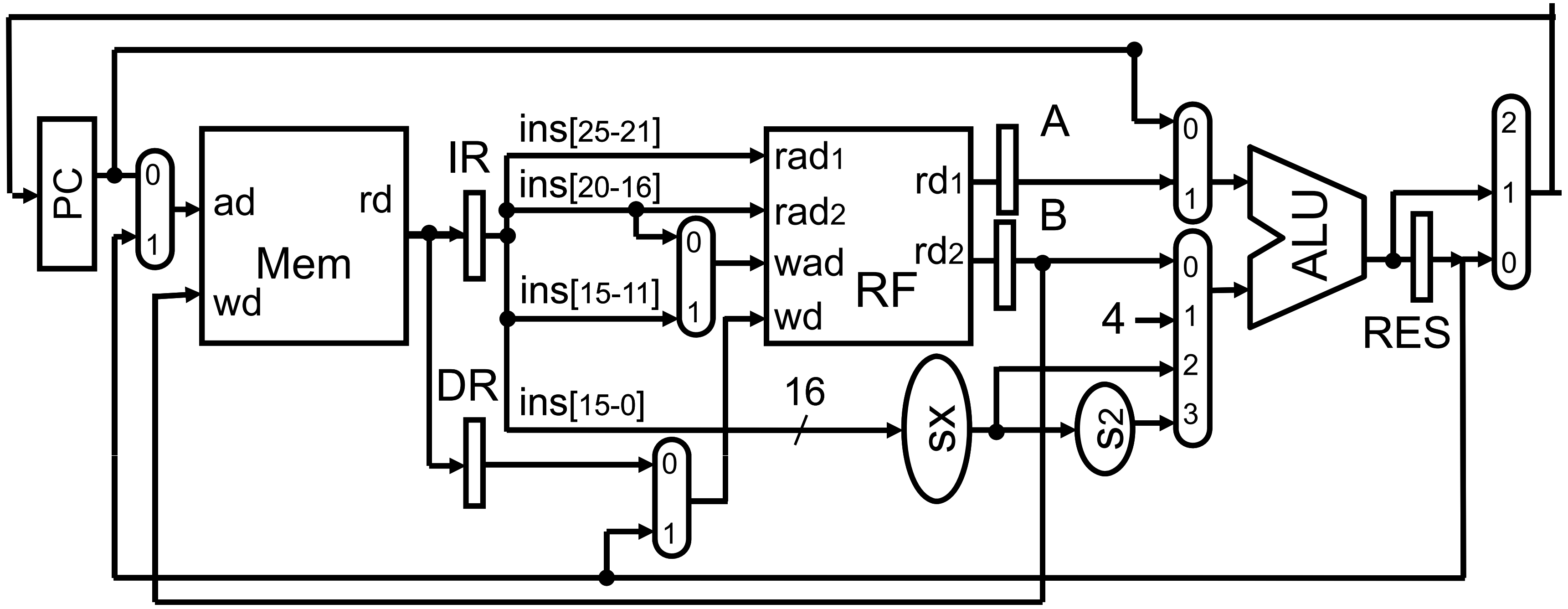


In case of SW instruction, the "rt" register of RF is identified by 5 bit address lines [Ins (20-16)] through rad2 port and Data from the register (rt) is loaded into B register through rd2 port of RF and finally data value is saved into memory location through wd port of the memory addressed by the effective address.

In case of LW instruction, the "rt" register of RF is identified by 5 bit address lines [Ins (20-16)] through wad port (MUX connected to wad port through select input=0). Data from the Memory location identified by the calculated effective address is loaded into the "DR" temporarily and Finally Data from DR register will be loaded into "rt" register of RF (MUX connected to wd port –select input=0).

Addition of BEQ Instruction **(BEQ R1, R2, OFFSET)**

According to PC address, Instruction is fetched from Memory and loaded into IR and is decoded. Simultaneously, PC value is replaced by (PC+4) in the first cycle.
Two registers of RF are identified by 5 bit address lines Ins (25-21) and Ins (20-16) respectively and two source registers rs and rt are accessed and read.
The operand-1 from RF register addressed by 5 Instruction bits (IR[25-21]) is loaded in ‘A’ and operand-2 from RF register addressed by 5 instruction bits 20-16) is loaded in the “B”. Effective address for branch will be calculated from (PC plus value [2 left of signed extended offset field) and will be loaded into ‘RES’ Register.



A and B operand will be compared and if Z-flag= 1, then branch address saved in “RES” will be loaded into PC register otherwise PC+4 will be loaded.

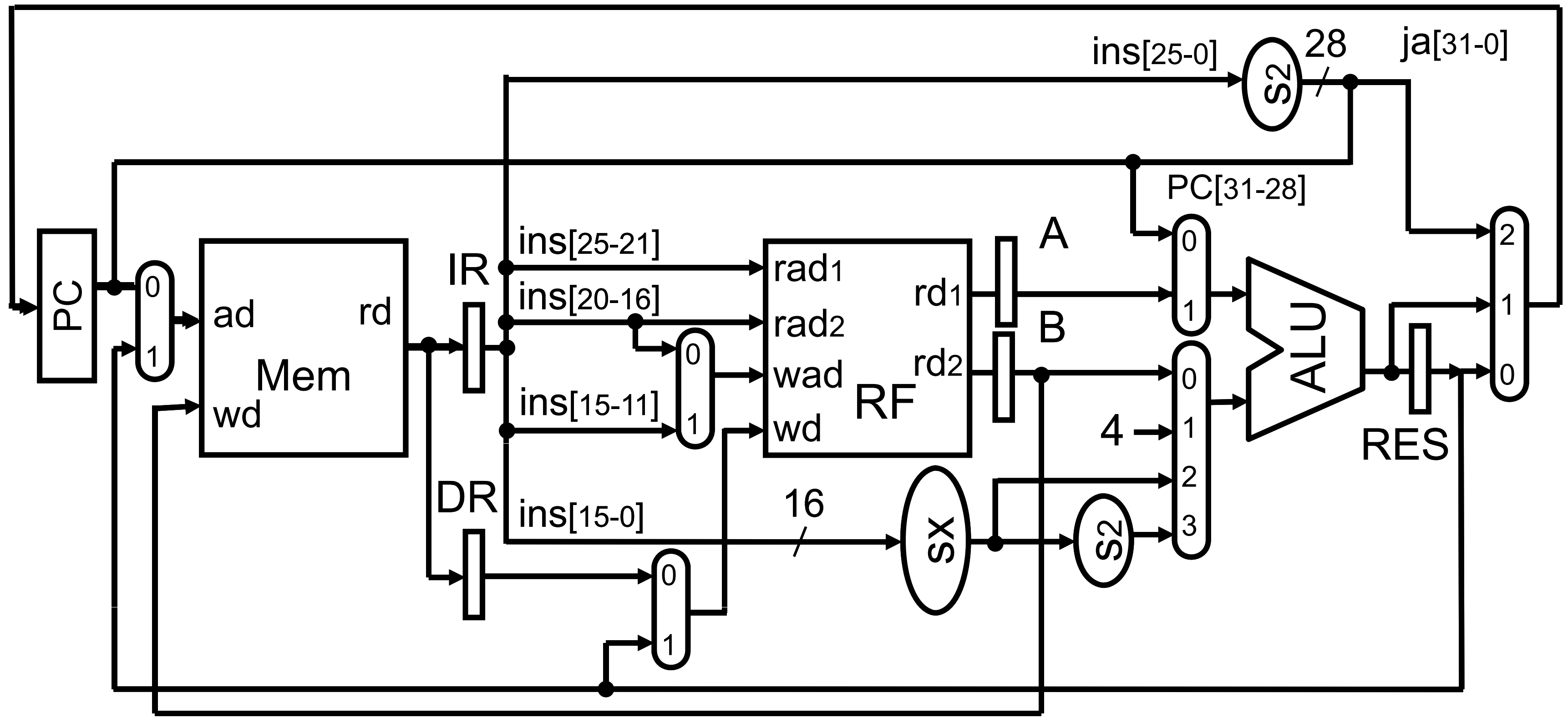
Adding "j" Instruction (J OFFSET)

	25-0
op	26 bit number

In cycle 1, Instruction is fetched from Memory based on address supplied by the PC and Instruction is saved in IR and PC is incremented by 4.

In cycle 2, JUMP will be calculated as follows...

	A31-A28	A27 ..... A2	A1 A0
JTA =	PC [31-28]	←26 bits from Instruction→	0 0



Assignments

1. Draw and explain the Multi-Cycle Datapath for implementing the following instructions

   a) R-type instruction

   b) sw/lw Instruction

   c) beq instruction

   d) ‘j’ type instruction

2. Draw the flow diagram and explain the operation the following instruction in terms of clock cycles.

   a) R-type instruction

   b) sw/lw Instruction

   c) beq instruction

   d) ‘j’ type instruction

Assignments

3. Draw and explain the controller state Transition Diagram for Multi-Cycle Datapath

4. Assign the control signal for the different components in Multi-Cycle Datapath

5. Discuss and tabulate the control signals for different micro-operation of the following group operations.

   a) PC Group

   b) Memory Group

   c) RF Group

   d) ALU Group

6. Draw the state transition diagram of the Multi-cycle Datapath controller in terms of control states and micro-operations.

Assignments

7. Tabulate the control states and signal values for different group operations in Multi-Cycle Datapath
8. Implement the control circuit for PC group operation in Multi-Cycle Datapath