

## ISA Introduction:

- IITB-RISC is a 16-bit very simple computer developed for the teaching that is based on the Little Computer Architecture.
- IITB-RISC has 8 general purpose registers i.e. R0 to R7.
- This design have six stage pipeline architecture, namely
- Instruction Fetch (IF), Instruction Decode (ID), Register Read (RR), Execution (EX), Data memory (MEM) and Write Back (WB).
- Instruction set of IITB-RISC processor consists of three machine-code instruction formats namely Register (R), Immediate (I) and Jump (J).
- This architecture uses a condition code register which has two flags: Carry flag ( C ) and Zero flag (Z).
- This architecture is optimize for performance, it have hazard mitigation techniques i.e. hazard detection and forwarding technique.

## Pipelined Processor:

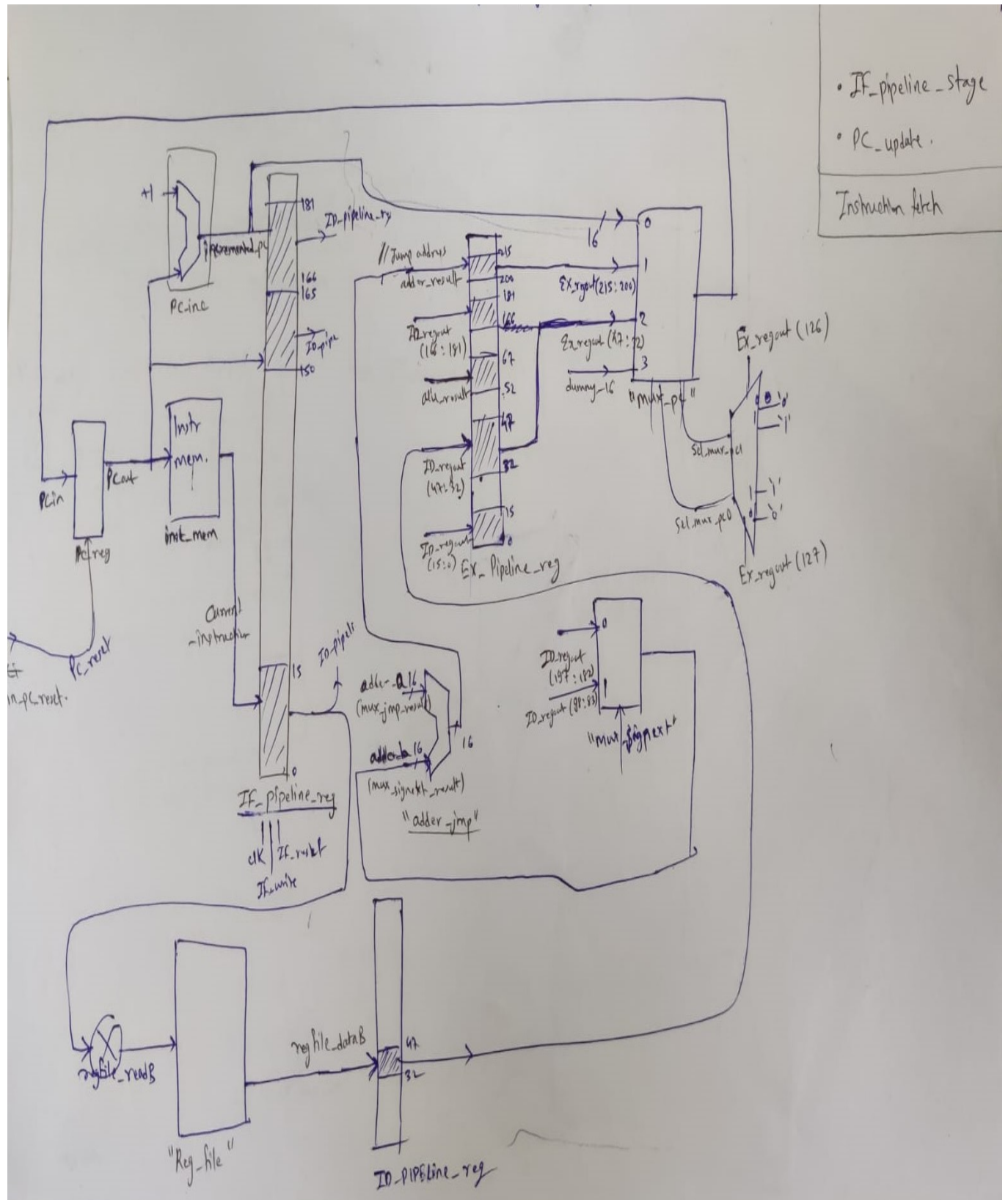
- Pipelined architecture improves the maximum operating frequency and throughput of processors by instruction level pipelining.
- In pipelining, divide datapath into nearly equal tasks, to be performed serially and requiring non-overlapping resources.
- Insert registers at task boundaries in the datapath. Registers pass the output data from one task as input data to the next task.
- Pipelining does not reduce the total time taken by an instruction, but it increases the number of instruction that can be executed together, and instruction throughput is increased.
- Even pipelining introduces latency in the output but the maximum frequency of operation is increased.

## Pipeline Hazards:

- Performance of any pipelined processor is degraded when an instruction depends on the result of previous instruction or any data which is not yet generated. In this case pipeline is stalled and processor send NOP instruction until the result for which the instruction was waiting is generated.
- In any pipelined architecture, three types of hazards occur they are control hazard, data hazard and structural hazard.
- To avoid these hazards, there is a need to forward data which is done by the forwarding unit. Hazards are resolved by Hazard detection and forwarding units.
- Compiler's understanding of how these units work can improve performance.

We have attached a README files in the folder, which will tells how to put data memory and instruction memory in the code and compile step by step using "do file".

# Instruction fetch stage:



• IF\_pipeline\_stage

• PC\_update

Instruction fetch

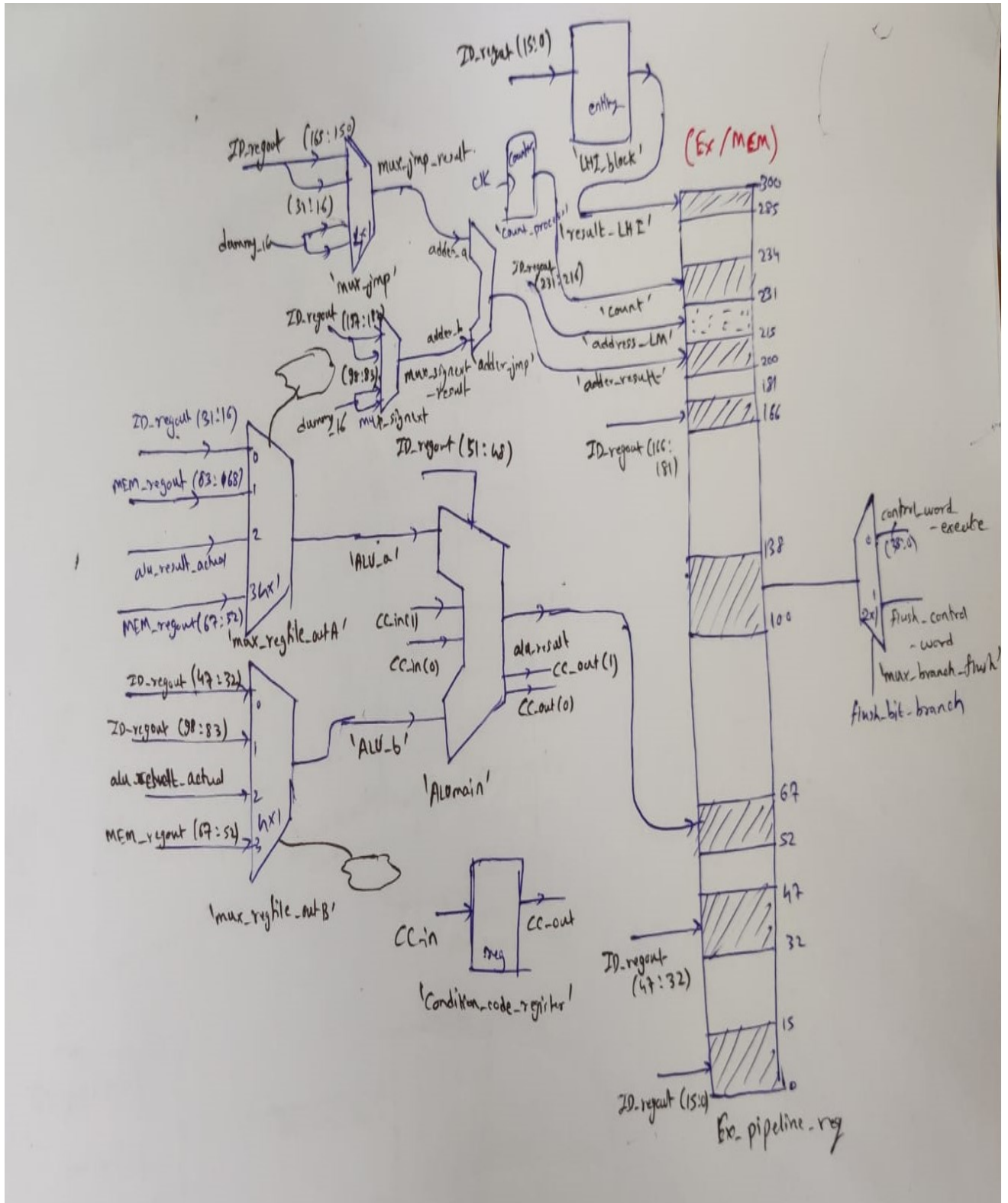
[illegible]



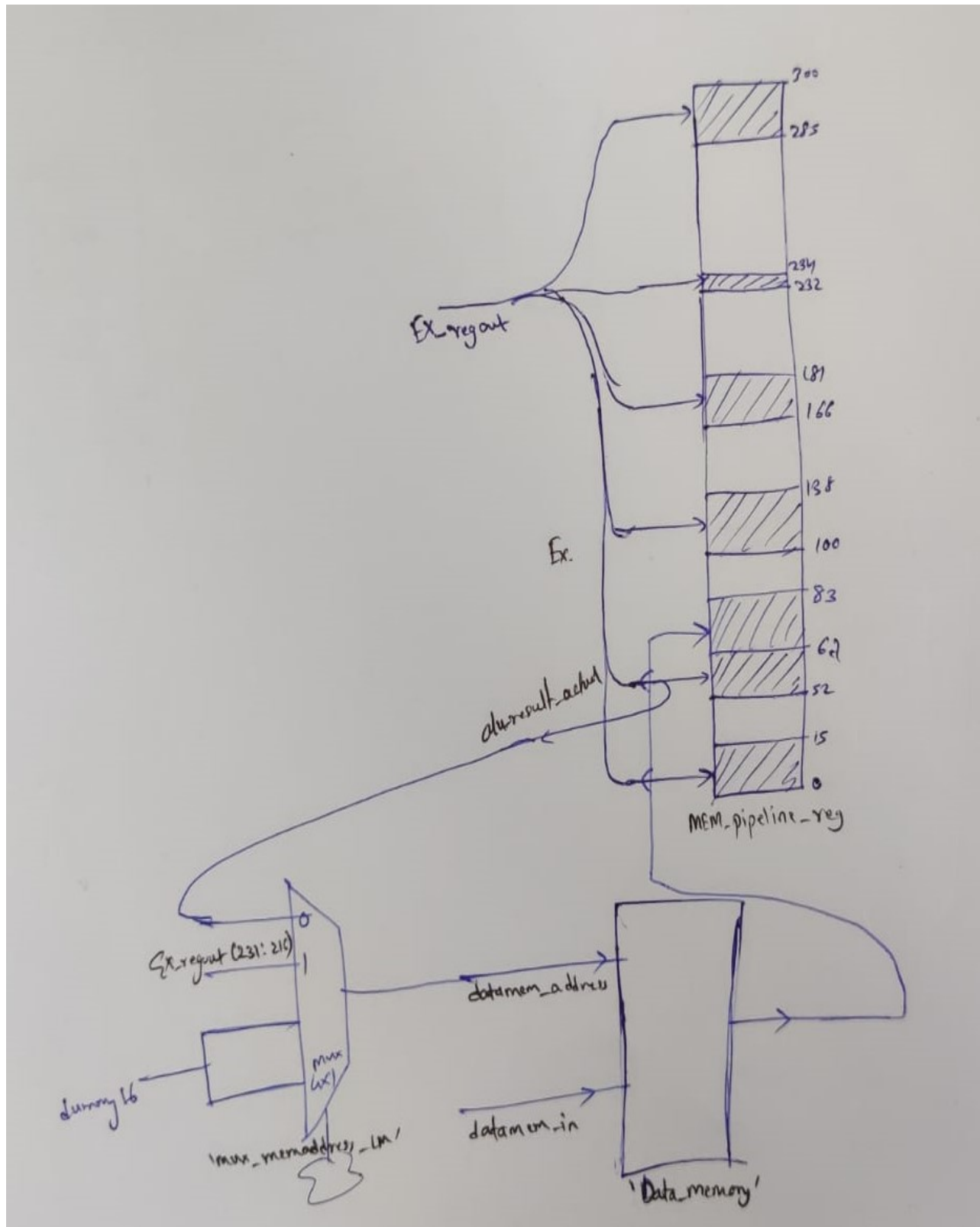
[illegible]

The diagram illustrates a 5-stage processor pipeline with the following components and connections:

- Registers:**
  - EX/MEM:** A vertical stack of 16 registers (0-15) on the right. Shaded registers are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15. Unshaded registers are 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31.
  - ID-regs:**
    - ID-regout (15:0) at the top.
    - ID-regout (16:150) and ID-regout (31:16) on the left.
    - ID-regout (17:17) and ID-regout (90:83) on the left.
    - ID-regout (31:16) on the left.
    - ID-regout (63:68) on the left.
    - ID-regout (67:52) on the left.
    - ID-regout (47:32) on the left.
    - ID-regout (30:83) on the left.
    - ID-regout (67:54) on the left.
    - ID-regout (47:32) on the left.
    - ID-regout (15:0) at the bottom.
  - MEM-regs:**
    - MEM-regout (63:68) on the left.
    - MEM-regout (67:52) on the left.
    - MEM-regout (67:54) on the left.
  - ALU-regs:**
    - ALU-a-1 and ALU-b-1 on the left.
    - ALU-main on the left.
  - Condition Code Register (CCR):** A register labeled 'Condition-code-register' with CC-in and CC-out ports.
- Multiplexers and Selectors:**
  - Mux-jmp:** Selects between ID-regout (15:0), ID-regout (16:150), and ID-regout (31:16) to produce mux-jmp-result.
  - Mux-sign:** Selects between ID-regout (17:17) and ID-regout (90:83) to produce mux-sign.
  - Mux-regfile-outA:** Selects between MEM-regout (63:68) and MEM-regout (67:52) to produce max-regfile-outA.
  - Mux-regfile-outB:** Selects between MEM-regout (67:54) and MEM-regout (67:52) to produce max-regfile-outB.
  - Mux-branch-flush:** Selects between control-word-execute (15:0) and flush-control-word to produce flush-bit-branch.
- Control Logic:**
  - Control:** A block that receives CLK and produces Count, Count-proc, and result-LH2.
  - Count:** A counter that produces Count.
  - Count-proc:** A block that produces Count-proc.
  - result-LH2:** A block that produces result-LH2.
  - address-LM:** A block that produces address-LM.
  - address-result:** A block that produces address-result.
- Connections:**
  - ID-regout (15:0) connects to the top of the EX/MEM register stack.
  - ID-regout (16:150) connects to the top of the EX/MEM register stack.
  - ID-regout (31:16) connects to the top of the EX/MEM register stack.
  - ID-regout (17:17) connects to the top of the EX/MEM register stack.
  - ID-regout (90:83) connects to the top of the EX/MEM register stack.
  - ID-regout (31:16) connects to the top of the EX/MEM register stack.
  - ID-regout (63:68) connects to the top of the EX/MEM register stack.
  - ID-regout (67:52) connects to the top of the EX/MEM register stack.
  - ID-regout (67:54) connects to the top of the EX/MEM register stack.
  - ID-regout (47:32) connects to the top of the EX/MEM register stack.
  - ID-regout (30:83) connects to the top of the EX/MEM register stack.
  - ID-regout (67:52) connects to the top of the EX/MEM register stack.
  - ID-regout (15:0) connects to the bottom of the EX/MEM register stack.
  - MEM-regout (63:68) connects to the top of the EX/MEM register stack.
  - MEM-regout (67:52) connects to the top of the EX/MEM register stack.
  - MEM-regout (67:54) connects to the top of the EX/MEM register stack.
  - ALU-a-1 connects to the top of the EX/MEM register stack.
  - ALU-b-1 connects to the top of the EX/MEM register stack.
  - ALU-main connects to the top of the EX/MEM register stack.
  - Condition-code-register connects to the top of the EX/MEM register stack.
  - Control connects to the top of the EX/MEM register stack.
  - Count connects to the top of the EX/MEM register stack.
  - Count-proc connects to the top of the EX/MEM register stack.
  - result-LH2 connects to the top of the EX/MEM register stack.
  - address-LM connects to the top of the EX/MEM register stack.
  - address-result connects to the top of the EX/MEM register stack.
  - Mux-jmp-result connects to the top of the EX/MEM register stack.
  - Mux-sign connects to the top of the EX/MEM register stack.
  - Mux-regfile-outA connects to the top of the EX/MEM register stack.
  - Mux-regfile-outB connects to the top of the EX/MEM register stack.
  - Mux-branch-flush connects to the top of the EX/MEM register stack.

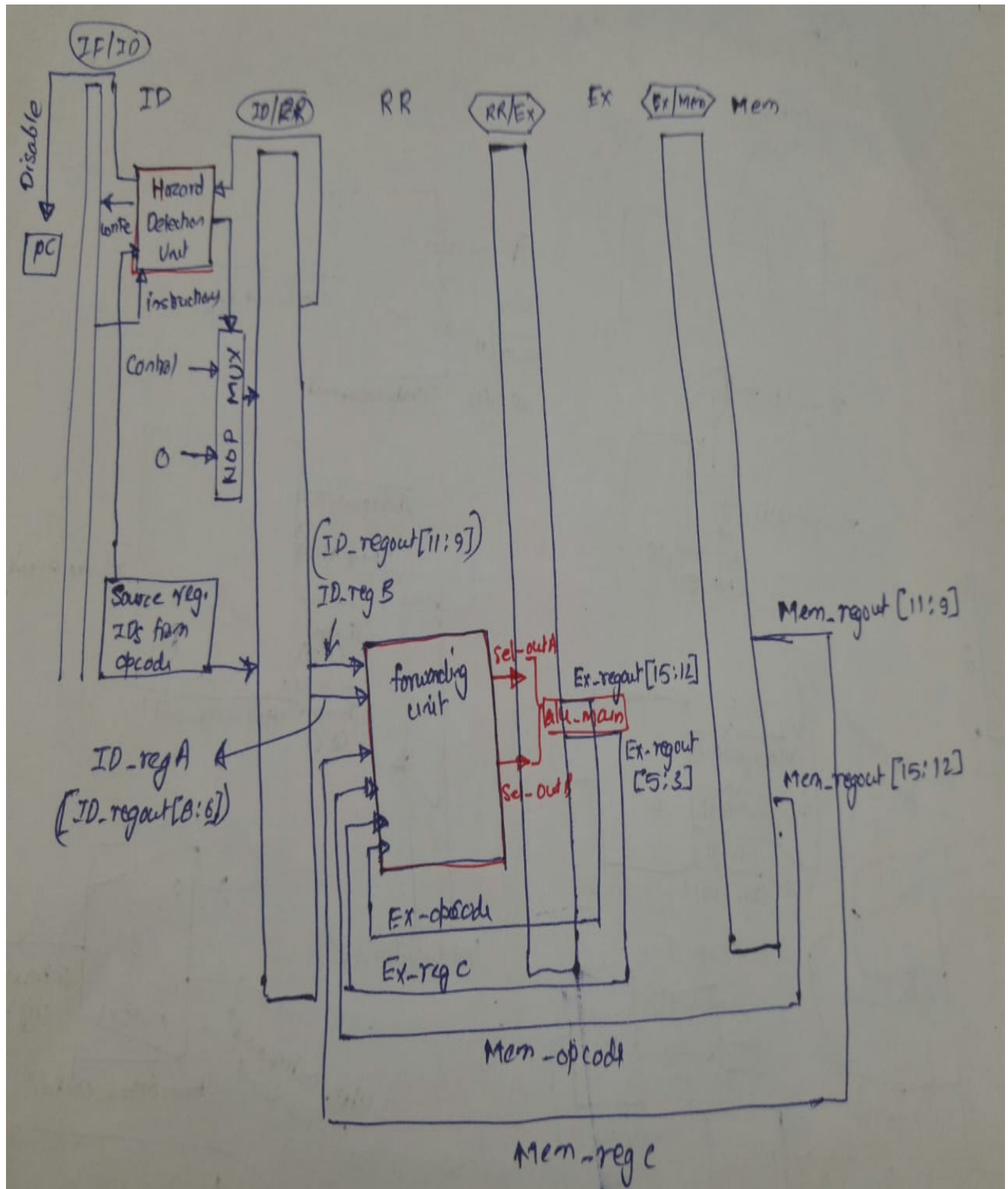


Memory stage:





Hazard detection and forwarding unit:





## Instruction sets:

Data Memory:	Instruction Memory:
X"0002", X"0005", X"0009", X"0004", X"0017", X"000C", X"0007", X"0008", X"0009", X"000A", X"000B"	"1100001011100100", "0001000010011000", "0001011001110000", "1000011101000010", "0001011010100000", "0010010001100000", "0000101000000111", "1100110010000010", "1001001000000010", "0001010011000000", "0100100110000000", "0100101110000001", "0001001110110000", "1101000010101010", "0000000000000000", "0000000000000000"

## Results of instruction set:

