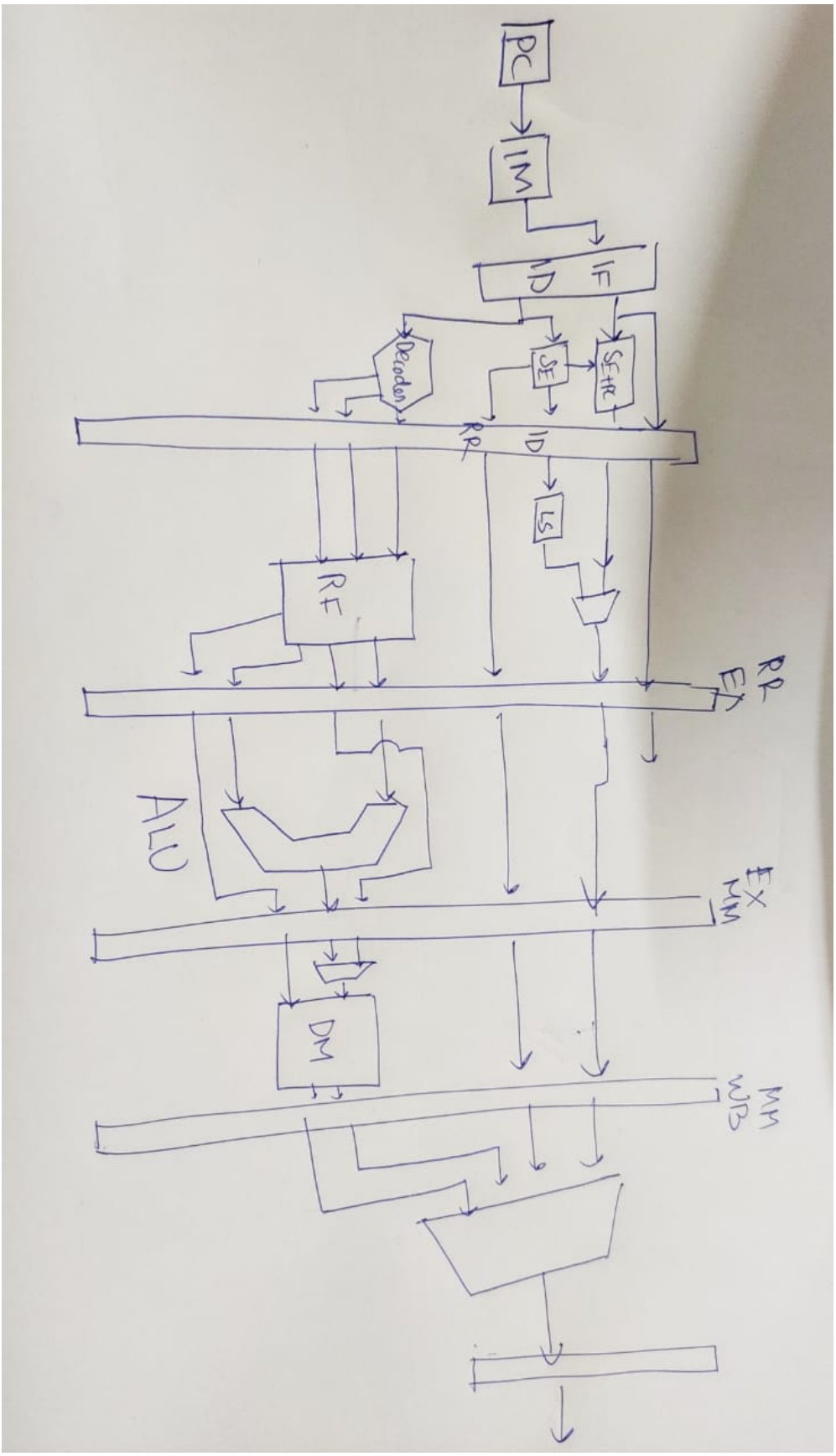
EE 739 Project 1 Report

Team members: Parvik Dave (190070044), E Abhishek (190070022), Mitali Meratwal (190070033), Parth Sankhe (190070043)

Datapath Diagram


PC
IM
ID
IF
Decoder
SE
SE+RF
RR
ID
LS
RF
RR
EX
ALU
EX
MM
DM
MM
WB

We have implemented a 6-stage pipelined processor, which is a 16-bit processor supporting a total of 19 instructions. Our VHDL files include:

- alu.vhd: This file contains the code for the ALU. The ALU can perform 3 kinds of operations: ADD, NAND, and XOR. The XOR function of the ALU is used as to compare the two inputs fed to the ALU. In order to have a fast and efficient implementation, we have implemented a Kogge-Stone Adder (KS Adder), which has significantly less propagation delay as compared to a normal ripple carry adder, for example
- decoder.vhd: This file contains the logic for the decoder, which is essential in the 2nd stage, which is the Instruction Decode stage
- intermediate_reg.vhd: This file contains the logic for all the intermediate registers required to store the signals between two stages
- reg_file.vhd: This file defines the register file which will store registers R0 through R7
- RAM: Data and address are 16 bits in size. So, there are 2^16 blocks of memory each with size 16 bits. The other two inputs are clock and write enable. If clock goes high and write is enabled then we write into the memory.

- ROM:

- main.vhd: This is the main file. It contains the entire pipeline process, and basically defines the processor itself. An instruction goes through 6 stages inside the main file, emulating an actual processor

Instruction Fetch

1. PC : Stores the PC value of the current instruction.
2. IM : Instruction Memory from which the instruction is obtained.
3. PC++ : It increments the PC value by 1 for the next possible instruction.

Instruction Decode

1. SE : It sign extends the 6-bit or 9-bit immediate data to 16 bit. It also observes the extra LLI bit in which case it only packs the immediate data by zeros.
2. SE+PC : It adds the SE value and the PC value of the current instruction.
3. Decoder: It gives the control signals according to the current instruction.
4. MUX : Steers input between PC+1 and (SE+PC) value. Chooses (SE+PC) when JAL instruction.

In decoder.vhd

1. AR1, AR2, AR3 represent the Ra, Rb, Rc operands in the instruction.
2. LM, SM, LW, BEQ represent whether the instruction is LM or SM or LW or BEQ.

3. WB_mux(mux control signal) represents ALU output for 00, LHI for 01,PC++ for 10, and LM,LW for 11.
4. ALU_C(mux control signal) represents ADD for 10, NAND for 00, and a comparator for 01.

Link to the GitHub repository: here