

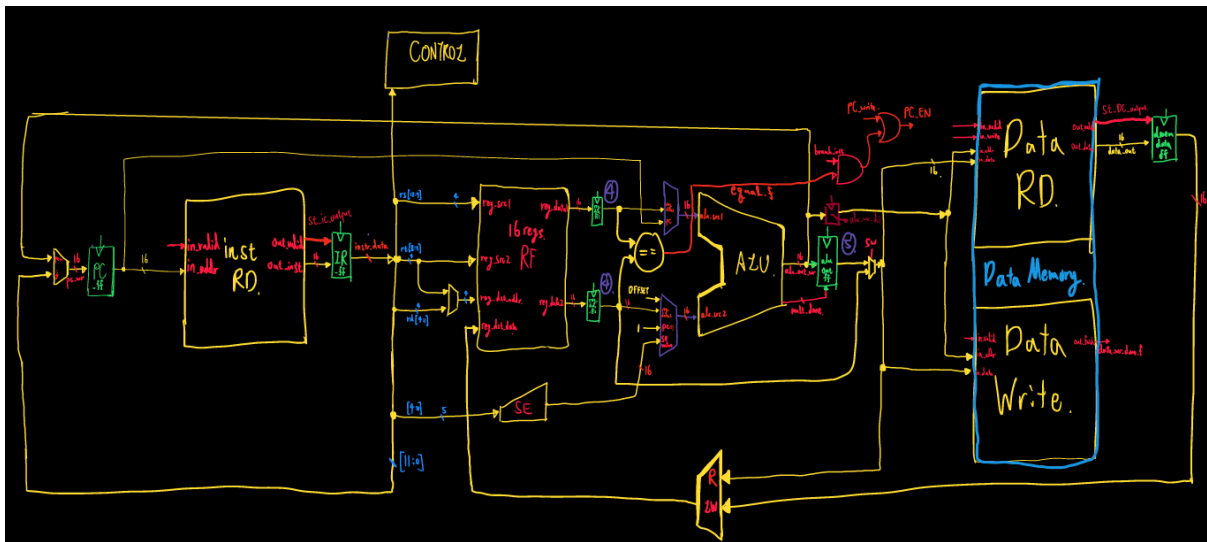
The diagram illustrates a 5-stage MIPS processor pipeline. The stages are:

- IF_WAIT.MEM**: Instruction Fetch and Memory Access. It receives the instruction from the PC and the register file. It outputs the instruction to the ID stage and the register file address to the MEM_WAIT stage.
- ID**: Instruction Decode. It receives the instruction from the IF_WAIT.MEM stage. It outputs the register file addresses to the EXE stage and the ALU to the MEM_WAIT stage.
- EXE**: Execute. It receives the instruction from the ID stage. It outputs the ALU result to the MEM_WAIT stage and the register file address to the MEM_WAIT stage.
- MEM_WAIT**: Memory Access. It receives the instruction from the EXE stage. It outputs the data to the MEM.WB stage and the register file address to the MEM.WB stage.
- MEM.WB**: Memory Write Back. It receives the instruction from the MEM_WAIT stage. It outputs the data to the register file and the register file address to the MEM_WAIT stage.

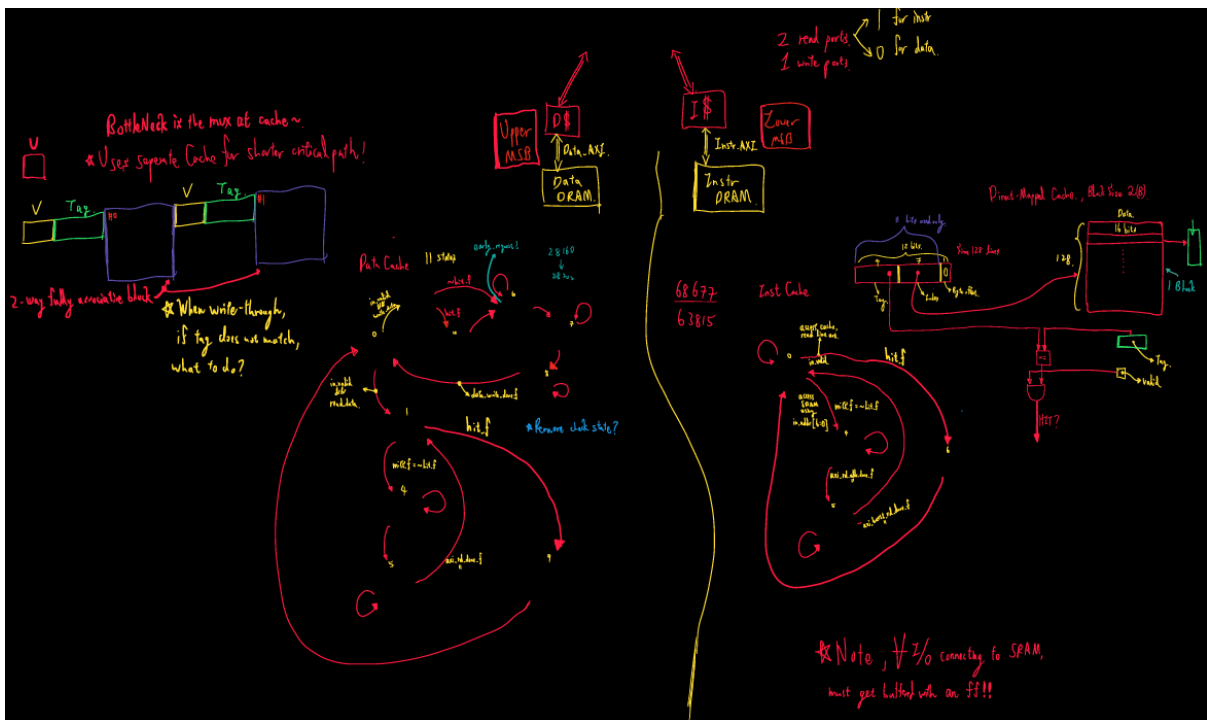
The diagram also shows the flow of data and control signals between the stages and the register file. Key components include the PC (Program Counter), ALU, and various buffers and multiplexers. The diagram is annotated with handwritten notes in blue and red ink, detailing the operations and data flow at each stage.

Due to the complexity of pipeline hazards, I chose to design the whole cpu in a multicycle fashion which is much easier.

Datapath Of CPU



Instruction Read and Data read interfaces to access memory, also share execution units through a single ALU.



I use 2 128x16 blocks of fully-associative block for better hit rate, which might provide great improvement when the pattern has lots of beq or j-type instruction.