**INTRO TO PROCESSOR ARCHITECTURE- PROJECT REPORT**

**-Jayant Duneja**

We have designed a 2 stage pipelined Mips Processor for this project. I have 3 modules in this processor

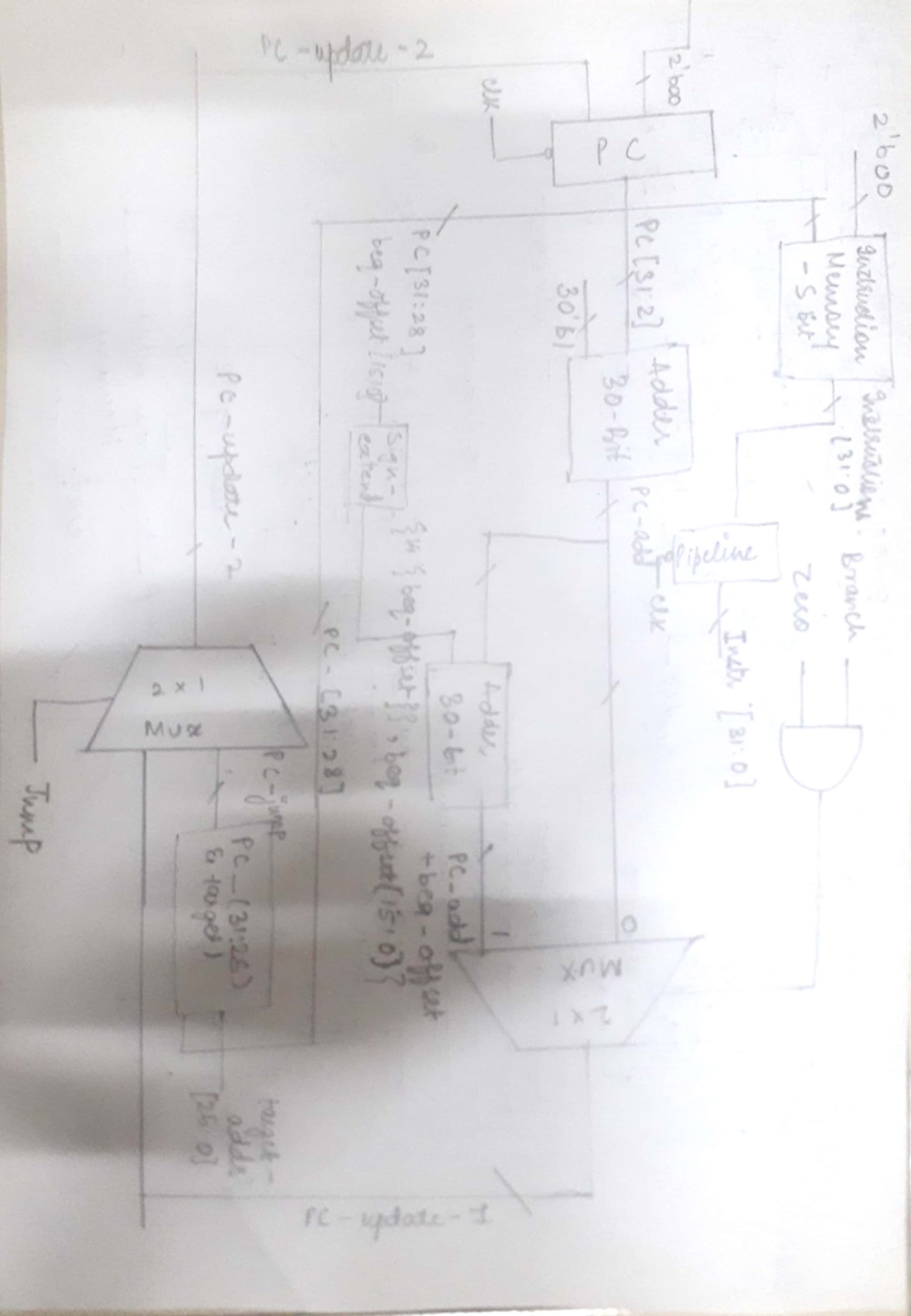
*ARCHITECTURAL DESIGN:*

We are implementing 15 instructions in the ALU

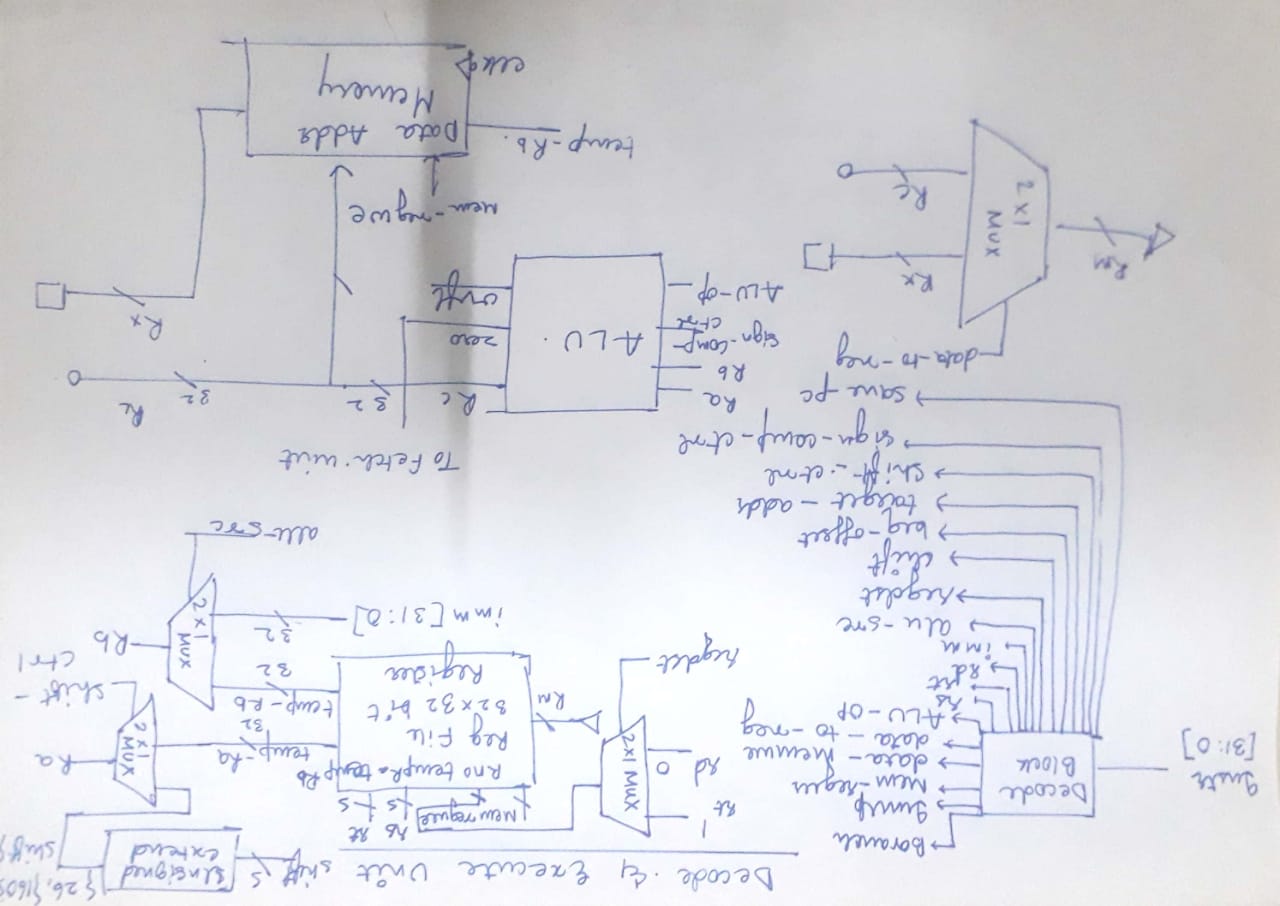
We have included 3 basic modules:

1. ALU stage
2. Decode stage
3. Fetch/Execute stage

* **DECODER**:The decoder will decode a total of 28 instructions. The decoder will give the alu\_op,control signals,registers addresses,jump target,address,beq\_offset etc.
* **FETCH/EXECUTE:** The fetch/execute will fetch the instruction and update the program counter. After this,we have also implemented the Execute stage which will update the registers at the negative edge of the clock.



*FETCH STAGE*



*DECODE/EXECUTE STAGE*

*MEMORY DESIGN:*

* Main memory has been designed on the basis of Harvard architecture
* We have separate Data memory and Instruction memory
* We have designed a Word based Data memory, i.e every memory block is 32 bytes and a byte based Instruction memory
* Instruction Memory consists of 128 blocks of size 1 byte each
* Data memory consists of 32 blocks of size 4 bytes each
* We have realised this memory in verilog by declaring arrays of the given size and both these memories were declared in the Fetch/Execute stage

*INSTRUCTIONS SUPPORTED:*

* We have implemented 28 instructions in total , 14 R type instructions,12 I type instructions,jump and JAL instruction.
* The 14 R-type are as follows: ADD,SUB,AND,OR,NOR,XOR,SLT,SLTU,SLL,SLLV,SRA,SRAV,SRL,SRLV
* The 12 I type instructions are as follows:

ADDI,ANDI,ORI,XORI,SLTI,SLTIU,LW,SW,BEQ,BNE,BGTZ,BLEZ

* Apart from that, we have implemented Jump and JAL(Jump and Load) instruction

*INSTRUCTIONS NOT SUPPORTED:*

* The following Instructions have not been implemented:

DIV ,MULT,MOD, NAND ,BGT, LB, LH, SB, SH, TRAP, MTLO ,MTHI, MFLO, JALR, BAL, EXT, LBE, LBU, MOVF, PAUSE, PREF, PREFE, WAIT, TLT, TLBP.

*PROCESSOR CLOCK FREQUENCY:*

* The length of the clock cycle is 10ns and hence the frequency is 100MHz

*PROS OF OUR DESIGN:*

* The design supports branch and jump instructions.
* We have implemented separate Data memory and Instruction Memory for this design

*CONS OF OUR DESIGN:*

* We have not included a branch delay slot and have placed no-op instruction after the branch instruction
* Only very basic and a limited number of instructions have been implemented
* I was facing an issue when I made the code modular, hence my project has all 3 modules in the same code