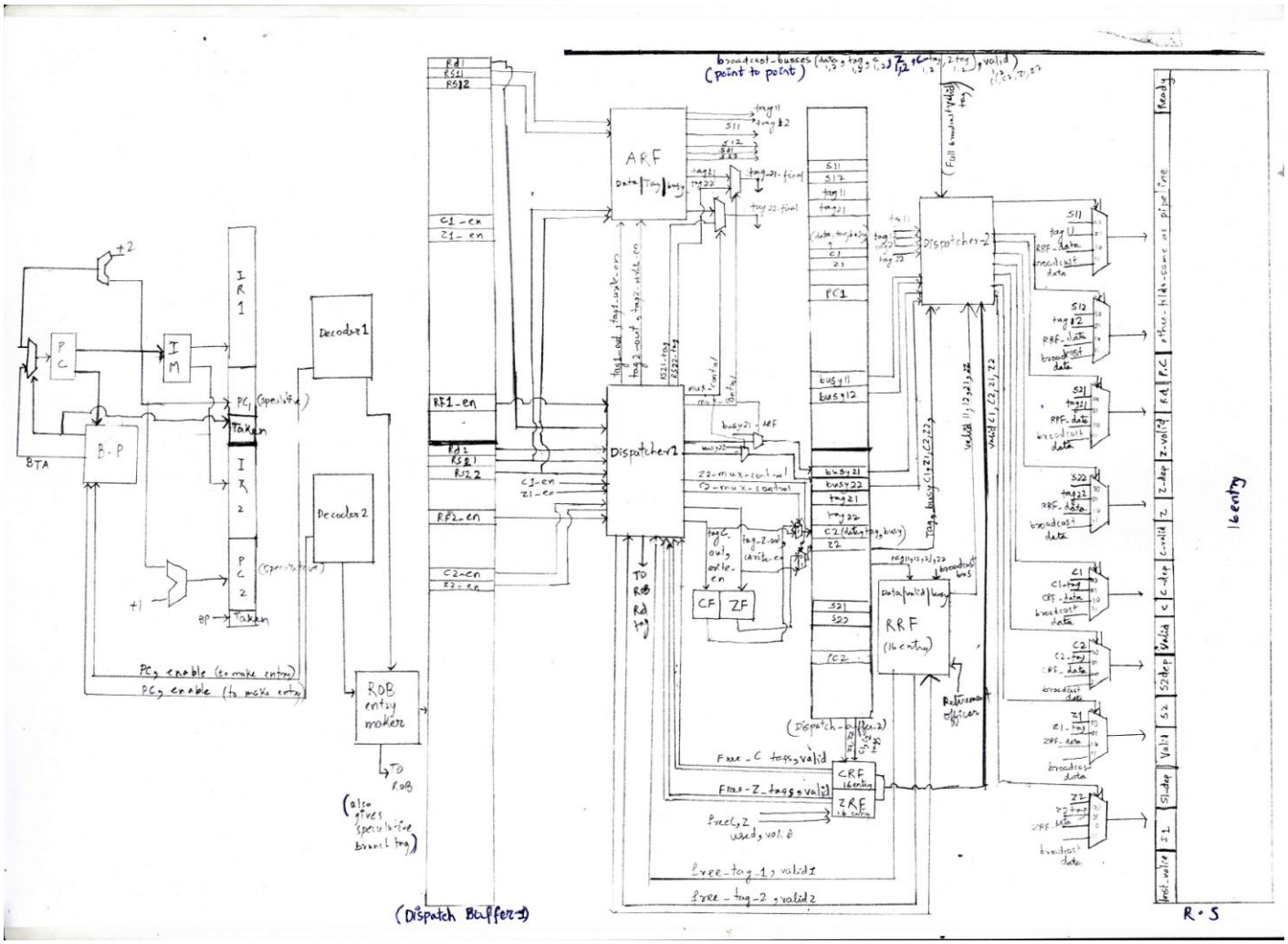
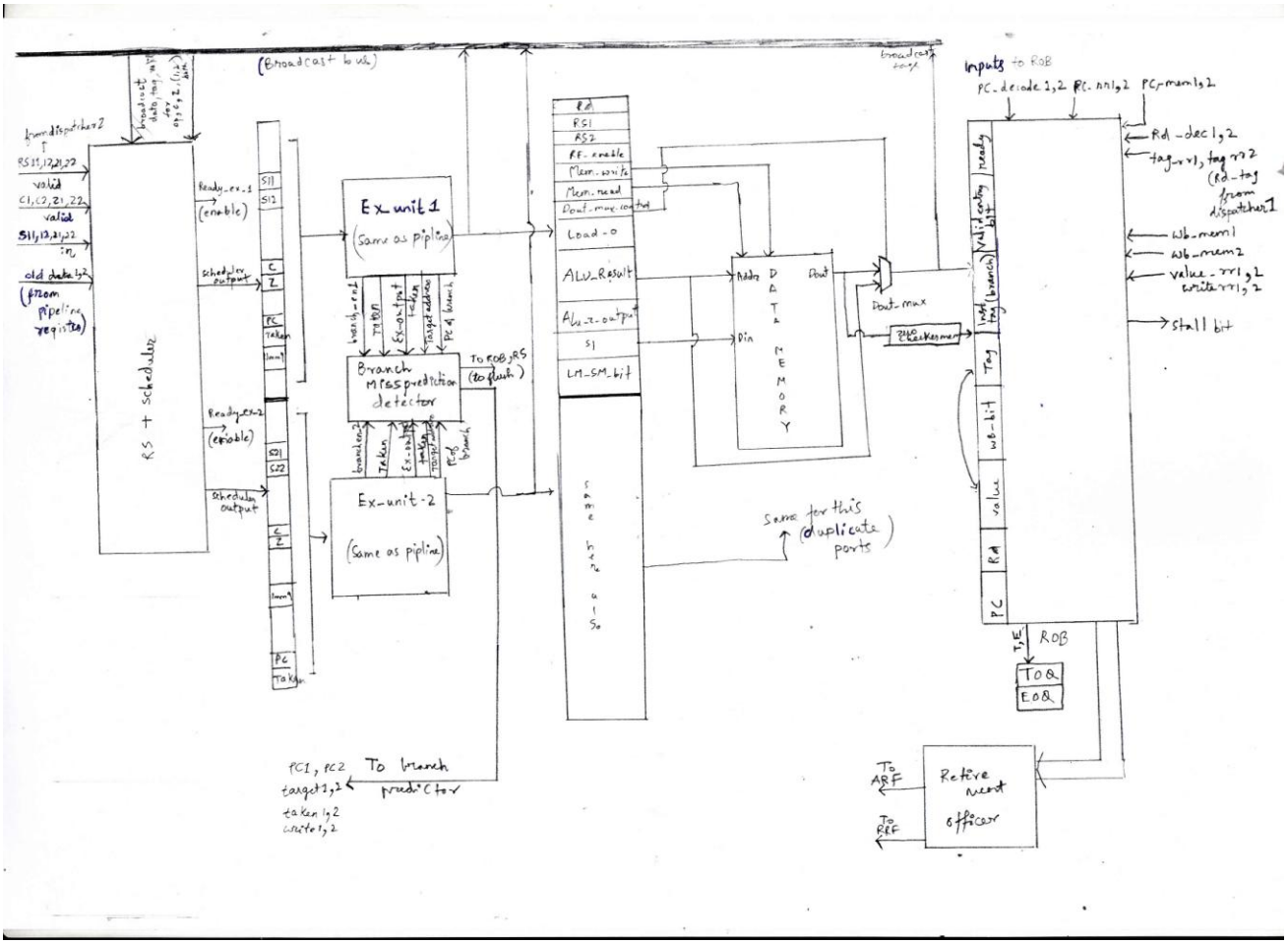
Two way fetch Out of Order Superscalar processor

Varunesh Goyal 140070006
Debarnab Mitrra 140070037
Dipti Ranjan Sahoo 140070038
Shyam Ramamoorty 140070042

1. Hardware design



Design diagram1: Fetch stage to Reservation Station



Design diagram2 : Reservation Station+ scheduler to ROB +retirement officer

2. Description of each stage and components

i) **Fetch stage**: - In the fetch stage the PC (which is also R7) is given to the instruction memory. At the same it is also given to the branch predictor. We have implemented a 32 entry fully associative branch predictor with 1 bit history and a valid bit stating if the entry in the branch predictor is valid or not.

PC	Target address	History Bit	valid

The branch predictor gets an entry from the decode stage and the execution stage as indicated in the hardware design diagram. As soon as the branches are decoded an entry in made in the branch predictor. After the execution of the branch instruction if it was wrongly predicted the target address as well as the history bit are changed.

ii) **Decode stage**: - The Instruction memory at every cycle gives out two instructions. These are decoded by two decoders which are same as the decoders used in the pipeline project. The decoder also talks to the ROB entry maker which makes entry for each instruction (in order) in the ROB and also gives a speculative branch tag. The decoder also makes entry in the branch predictor as said earlier. The decoded fields are stored in the Dispatch Buffer-1. Some of the fields are indicated in the diagram as Rd1, Rd2, RS11, Rs12, Rs21, Rs22, C1_en, c2_en, z1_en, z2_en, Rf1_en, Rf2_en. Rest of the fields are Mem_write1, Mem_read1, Dout_mux_control1, Carry_dep1, zero_dep1, alu_cntrl1, alu_ootput_mux_control1, RS11_dep, RS12_dep, Load_o1, Imm9_1, PC1, JAL_bit1, JLR_bit1, LM-SM_bit1, Mem_write2, Mem_read2, Dout_mux_control2, Carry_dep2, zero_dep2, alu_cntrl2, alu_ootput_mux_control2, RS21_dep, RS22_dep, Load_o2, Imm9_2, PC2, JAL_bit2, JLR_bit2, LM-SM_bit2.

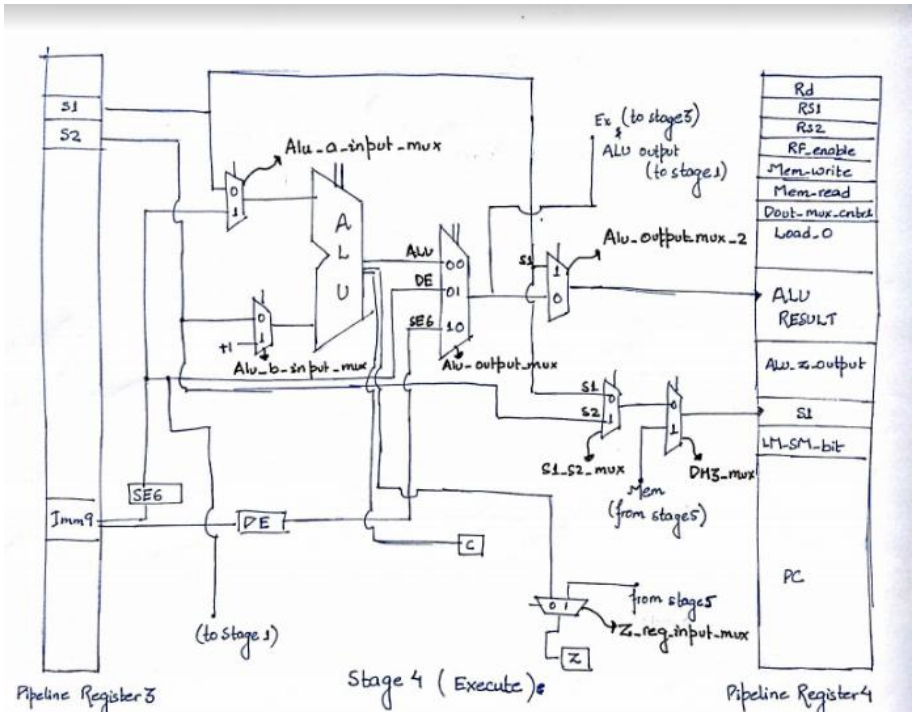
iii) **Dispatch stage 1, 2**: - The dispatch is carried out in 2 stages. In the first one, the ARF, C, Z is read and tags are assigned to the destination registers as well as C and Z if the instruction is depended on them. In the next stage, the rename tags (or the data if present in the RRF) from the RRF, CRF, ZRF are read.
Renaming is also done for C and Z registers as there may be many outstanding writes on these flag registers and other instruction may be willing to read them. In short there many be true as well as false data dependencies on C and Z flags and Renaming is required.
RRF – 16 entry
CRF – 16 entry
ZRF – 16 entry

Dispatcher1 - It checks for inter instruction dependency among the 2 simultaneously fetched instructions too and also receives free tags from the RRF, CRF, ZRF to assign to the destination registers.
Dispatcher2 - It looks at the busy bits and the tags from the ARF of the source operands, the entries in the RRF corresponding to the tags and also the broadcast bus and decides on whether to pass the operands or the tag. It basically gives the control signal to the muxes (as in the design diagram) at the head of the Reservation Station and decides which data should enter the reservation station.

The fields in the dispatch buffer-2 are same as in dispatch buffer1 expect a few new ones which have been indicated in the design diagram.

iv) **Reservation Station + scheduler**: - The dispatched instructions come and sit in the Reservation station (32 entry) and wait for the operands. The reservation station receives forwarded data from the execution and the memory stage against different rename tags (RRF, CRF, ZRF). It (RS) copies those forwarded data if some instruction is waiting for the corresponding tags. (Note: in the design diagram the forwarded data is indicated as a bus but it is actually a point to point communication link.) Once the instruction gets all the desired operands it is declared ready and issued by the scheduler to the OoO execution units. Only the upper pipeline lane has Data memory and is capable to execute memory instructions. So the scheduler issues memory instructions on the upper lane. (There was one more option of having multiple ports in Data memory thus giving the lower pipeline capability to execute memory instructions but this has not been implemented. Duplicating memory ports is costly!).
At every cycle the scheduler tries to issue two instructions but if it is unable in doing so it makes Ready_ex_1(or Ready_ex_2) signal low. These signals are the enables for the corresponding execution units.
The fields of the Reservation station are indicated in the hardware design diagram. 'other fields same as pipeline' indicates the remaining fields from the dispatch buffer-2 which are copied in the RS as it is which any intervention by the dispatchers.

v) **OoO EX units**: - The OoO Ex units receive the necessary control signals from the scheduler with all the necessary operands. They execute the instructions as dictated by control signals and the result gets stored in the next pipeline register. The inside of each of the execution unit is shown in the image below.



The above image is the execution unit of the pipeline project. Here we will be using the same unit. The only difference is the SE and ZE are before the RS and the Sign extended and Zero extended values are stored in the RS. Also the muxes at the input to the ALUs are not required as RS gives all the necessary operands.

The output of the Ex units is also fed to the branch misprediction unit which checks if the branch instruction is correctly executed or not and if its not then the instructions with branch tags after that particular branch instruction are invalidated in the ROB and the RS. Also the branch predictor entry is modified and the PC is changed to the output of EX unit at that stage.

vi) **Memory stage**: - Output of the EX stage is passed on to the memory stage. This stage is also same as the Pipeline project. The result from this stage is stored in ROB as stored in design diagram 2. A thing missed out in the design diagram 2 is that the memory stage also requires a branch misprediction unit as loads into R7 act as branch instructions. So output from this stage also has to make changes to the branch predictor as well as the flushing action to be taken.

vii) **ROB and retirement officer**: - The output from the memory stage is stored in the ROB (32 entry). The fields of the ROB are indicated in the design diagram2. There is a retirement officer which looks into the ROB and can retire at max 2 instructions every

cycle in inorder sense. The retired values go the ARF, C and Z flags. The RRF tags are freed. Also memory operations are initiated. ROB is a cyclic queue. A Top of Queue (TOQ) and a End of queue (EOQ) is maintained for that purpose.