**COMPUTER ORGANIZATION AND ARCHITECTURE**

**APEX SIMULATOR**

**A Project by –**

**Ishwar G. Khatri**

**Varun R. Kulkarni**

**Pratik Dobriyal**

**Guidance By:- Prof. Timothy Miller.**

**Computer Architecture Project Overview**

In project part-I we designed APEX simulator with 5 pipelined stages, which executes instructions cycle by cycle in program order.

Project part-II is just an extension of project part-I which implements APEX simulator with Issue queue and Reorder buffer. There is an additional data structure LSQ (load-store queue) which takes all load and store instructions, while all other instructions are transferred to issue queue.

Issue queue has 8 entries, load-store queue has 4 entries and Reorder Buffer has 16 entries. There are two fetch stages F1 and F2 wherein we fetch instructions from given input file. There are two decode stages D1. In first decode we deal with branch instructions while in second decode we deal with all other instructions. In decode 2 stage we separate out operands and opcodes of an instruction and we transfer this separated information to issue queue, load-store queue(in case of load-store instruction) and Reorder Buffer which is used for instruction commitment and sending updated results to required destinations of upcoming instructions. As we are using Reorder Buffer we can skip register renaming part as Reorder Buffer is actually an alternative for register renaming.

For instruction execution there are three separate functional execution units viz. Integer, Multiplication and Memory (for load-store instructions). Integer execution has 1 cycle latency, Multiplication has 4 cycle latency while Memory unit has 3 cycle latency.

Instructions are forwarded to issue queue after D2 stage considering their resources and destinations. Entry is also made in Reorder Buffer after the D2 stage where we note destination register, slot id of Reorder Buffer which is given to Register Rename Table. We are also implementing forwarding where the value is passed to the Issue queue, load-store queue for the required instructions.

Commits to ARF (Architectural Register File) are always done in program order. Whenever a commit has to be done, entry at head of RoB is checked for completion (execution is finished). If the destination register has got its result from execution stage, then this instruction is selected for retirement. In retirement stage we copy the register value of destination register and write it to ARF marking that entry as 0(valid) in rename table. So whenever a request comes in for this register, its value will be directly read from ARF.

Project requirement was discussed in detail and understood the functionality of each part. Each member has contributed equally to bring this project to its completion phase. All team members were together in entire project development process. Although we have worked, individually on some part of projects, all the team members are equally responsible. We have successfully managed to meet all the requirement of project, together.