CS 520 - Fall 2017 - Ghose

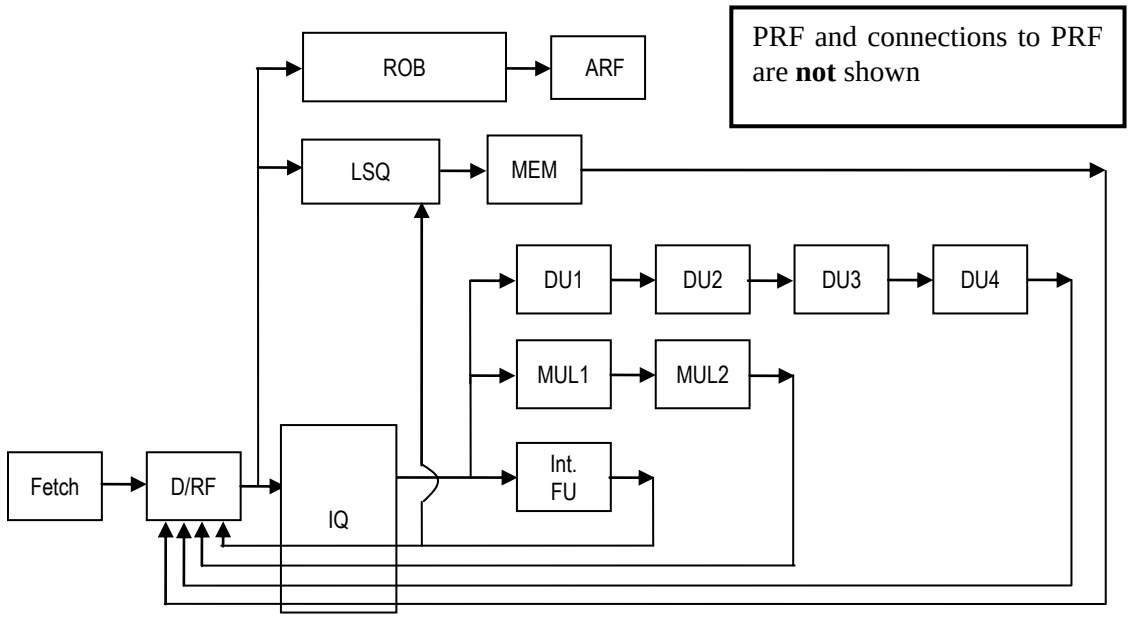
Programming Project 3: Out-of-order processor of simulator for APEX

THIS IS A TEAM ASSIGNMENT - **3 STUDENTS TO A TEAM**

DUE: Monday, December 4 midnight - STRICT DEADLINE

All demos to be completed by Friday, December 8.

For this project, you are required to implement the simulator for an out-of-order processor organized as in Variation 2 of the notes. The processor to be simulated is organized as follows:



The processor to be simulated has the following features:

- It has the same instructions as the processor you simulated for Project 2. The function units are also the same as in Project 2.
- It uses register renaming with an IQ, ROB and LSQ. A physical register file is used, along with an architectural register file. The issue queue entry holds literal operands and all register operand values.
- LOADs can bypass earlier STOREs in the LSQ when the bypassing conditions hold. A LOAD can also pick up the queued value to be stored from the LSQ entry of the nearest preceding STORE if they both target the same memory location.
- Memory addresses are calculated on the IntFU and written to the LSQ entry for LOADs and STOREs. Memory accesses take place from the head of the LSQ using the stage MEM, which is not pipelined and has a 3-cycle delay. Each memory access thus takes 3 cycles.

- Four forwarding/broadcast buses exist to the IQ for simultaneous use by the IntFU, DIV FU, MUL FU and MEM in a single cycle if needed. The physical register file, IQ, LSQ and ROB each have enough ports to support as many concurrent reads and writes that are needed in a single cycle.
- All decoding, renaming and dispatch steps are performed in a single D/RF stage.
- The processor supports **back-to-back execution**, with the tag (which is a physical register address) broadcasted one cycle before the result is available.
- As in Project 2, the target address of the branches, JUMP and JAL are computed in the IntFU.
- A field is added to every physical register to hold the flag value produced along with the result held in the physical register. For project 2, the ONLY flag that you have to deal with is the Z flag, so you will need an 1-bit extension for each physical register. As an example, when you have a SUB instruction that uses physical register P5 as an destination, store the result of the SUB in P5 and hold the Z flag produced by the SUB was a zero or not in the 1-bit extension of P5. The rename table will need an entry for the Z flag.

The configuration details on the number/capacities of the key datapath components are as:

Architectural registers: 16 (Numbered 0 to 15), Physical registers: 32 (Numbered 0 to 31)

IQ capacity : 16 entries, ROB capacity: 32 entries, LSQ capacity: 32 entries

All other functional requirements for the simulator are as in Project 2. The "display" command needs to print the contents of all major components that are simulated (rename table, LSQ, IQ, ROB, PRF, Fetch PC, elapsed cycles, CPI up to the last simulated cycle etc.

SPECIAL REQUIREMENT: You need to submit a detailed document on what each individual team member contributed towards this project.