CS 520 - Fall 2016 - Ghose
Project 2: Preliminary

3 to 4 students to a project team
Due: Monday, December 12

This is a preliminary specification to get you started. Further details will follow.

For Project 2, you need to extend the simulator that you have developed for Project 1 with a renaming mechanism that uses a unified register file, a centralized IQ and a ROB. There is no LSQ.

You can implement the IQ in any way within the simulator as long as wakeups and selection are implemented correctly. The wakeup signal for a function unit is generated one cycle before the function unit completes to support back-to-back execution. Ties for selection of a specific function unit are broken using a FIFO policy that selects the instruction dispatched earlier. You can implement the FIFO by adding an IQ entry field that holds the cycle in which the instruction was dispatched. Speculative execution is not supported.

The following function units are used and all function units, excepting the branch FU, has a writeback stage with a one cycle latency:

- A two-stage pipelined integer ALU (two stages, one cycle per stage) implementing all arithmetic instructions excepting a multiply. This function unit also implements the MOVC instruction by adding an implicit zero value to the literal in MOVC and writing it to the destination.
- A non-pipelined multiplication unit with a latency of 4 cycles that implements the multiply operation.
- A single cycle branch FU that computes the target address and decides whether to branch or not. This function unit also implements the JUMP and BAL instruction.
- A two-stage pipelined LSFU (one cycle per stage) implementing the LOAD and STORE instructions. LSFU generates memory address (stage 1 of LSFU), performs TLB lookup (LSFU 2nd stage) and then accesses cache when LOAD or STORE is at the head of the ROB. Assume Cache access performed by LSFU retrieves data in one cycle. There is no bypassing of earlier STORES by a later LOAD. When a LOAD completes, the result is written to the destination via the associated WB stage.

The unified register file has enough write ports to allow results from 3 FUs to be written to the unified register file. As a result is written to the unified register file, it is also forwarded to waiting instructions. Up to 3 wakeup signals can be sent per cycle and 3 results can be simultaneously forwarded per cycle. Only one commit per cycle is allowed.

You will need to implement all artifacts associated with renaming for a unified register file, including the rename table, the back-end register alias table, register allocation and deallocation operations.

Further details are forthcoming.