6.823 Computer System Architecture
Out-of-Order Execution with ROB

http://csg.csail.mit.edu/6.823/

In this handout, we introduce a simple out-of-order processor using a reorder buffer. In this processor, instructions enter the reorder buffer (ROB) in-order, execute out-of-order, and commit in-order. A summary of the operation is given below. Some entries in the reorder buffer, register file, and rename table have been filled in as an example snapshot.

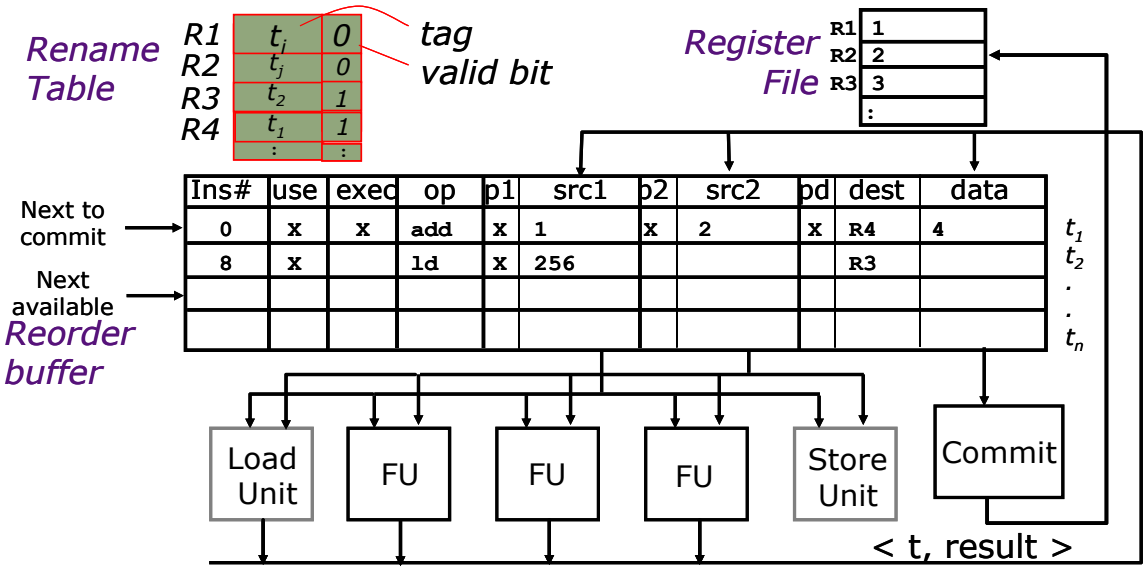

Figure H10-A

Fetch: Instruction is fetched from an instruction cache and queued in an instruction fetch buffer.

Decode: An instruction is decoded from the instruction fetch buffer, and rename table and register file are read simultaneously. If the rename table has the valid bit set for an operand (i.e. the bit is 1), then the ROB has to be checked for that operand. Otherwise the register file value can be used. The instruction is assigned a slot in the ROB (the ROB index is this instruction's tag). If the instruction writes a register, its tag is written to the destination register entry in the rename table.

Issue: The instruction is written to the ROB, with either a tag or a register value in each of its source operand fields. A tag will be used for each source register operand that has not yet been produced, and identifies the instruction in the ROB that will produce the needed result.

Execute: An instruction can begin executing when all of its operands are present.

Write-Back: When an instruction completes execution, the result, if any, will be written back to the *data* field in the reorder buffer and the *pd* bit will be set. Additionally, any dependent instructions in the reorder buffer will receive the value.

Commit: The instruction result, if any, will be written to the register file or memory in program order. If the instruction writes a register and the tag in the rename table for this register matches the tag of the result, the rename table's valid bit will be cleared. Finally, the instruction's entry in the reorder buffer will be deallocated.