Speculation to greater ILP

- 3 components of HW-based speculation:
  - Dynamic branch prediction to choose which instructions to execute
  - Speculation to allow execution of instructions before control dependences are resolved
    - Ability to undo effects of incorrectly speculated sequence
  - Dynamic scheduling to deal with scheduling of different combinations of basic blocks

Adding Speculation to Tomasulo

- Must separate execution from allowing instruction to finish or “commit”
  - This additional step called instruction commit
- When an instruction is no longer speculative, allow it to update the register file or memory
- Requires additional set of buffers to hold results of instructions that have finished execution but have not committed
  - This reorder buffer (ROB) is also used to pass results among instructions that may be speculated

Reorder Buffer (ROB)

- In Tomasulo’s algorithm, once an instruction writes its result, any subsequently issued instructions will find result in the register file
- With speculation, the register file is not updated until the instruction commits
  - Know definitively that the instruction should execute
- Thus, the ROB supplies operands in interval between completion of instruction execution and instruction commit
  - ROB is a source of operands for instructions, just as reservation stations (RS) provide operands in Tomasulo’s algorithm
  - ROB extends architecture registers like RS

Reorder Buffer Entry

- Each entry in the ROB contains four fields:
  1. Instruction type
    - A branch (has no destination result)
    - A store (has a memory address destination)
    - A register operation (ALU operation or load, which has register destinations)
  2. Destination
    - Register number (for loads and ALU operations) or memory address (for stores) where the instruction result should be written
Reorder Buffer Entry (cont.)

- Each entry in the ROB contains four fields:
  3. Value
     - Value of instruction result until the instruction commits
  4. Ready
     - Indicates that instruction has completed execution, and the value is ready
  
Recall: 4 Steps of Speculative Tomasulo Algorithm

1. Issue—get instruction from FP Op Queue
   If reservation station and reorder buffer slot free, issue instr & send operands & reorder buffer no. for destination (this stage sometimes called "dispatch")
2. Execution—operate on operands (EX)
   When both operands ready then execute; if not ready, watch CDB for result; when both in reservation station, execute; checks RAW (sometimes called "issue")
3. Write result—finish execution (WB)
   Write on Common Data Bus to all awaiting FUs & reorder buffer; mark reservation station available.
4. Commit—update register with reorder result
   When instr at head of reorder buffer & result present, update register with result (or store to memory) and remove instr from reorder buffer. Mispredicted branch flushes reorder buffer (sometimes called "graduation")

Reorder Buffer operation

- Holds instructions in FIFO order, exactly as issued
- When instructions complete, results placed into ROB
  - Supplies operands to other instruction between execution complete & commit ⇒ more registers like RS
  - Tag results with ROB buffer number instead of reservation station
- Instructions commit ⇒ values at head of ROB placed in registers
- As a result, easy to undo speculated instructions on mispredicted branches or on exceptions

Tomasulo With Reorder buffer:

In the diagram, the reorder buffer is shown with instructions and their corresponding values. The steps of the Tomasulo algorithm are illustrated, showing the flow of instructions through the pipeline stages of issue, execution, write result, and commit.
Avoiding Memory Hazards

- **WAW and WAR hazards** through memory are eliminated with speculation because actual updating of memory occurs in order, when a store is at head of the ROB, and hence, no earlier loads or stores can still be pending.
- **RAW hazards** through memory are maintained by two restrictions:
  1. not allowing a load to initiate the second step of its execution if any active ROB entry occupied by a store has a Destination field that matches the value of the A field of the load, and
  2. maintaining the program order for the computation of an effective address of a load with respect to all earlier stores.
- These restrictions ensure that any load that accesses a memory location written to by an earlier store cannot perform the memory access until the store has written the data.

Exceptions and Interrupts

- IBM 360/91 invented “imprecise interrupts”
  - Computer stopped at this PC; its likely close to this address
  - Not so popular with programmers
  - Also, what about Virtual Memory? (Not in IBM 360)
- Technique for both precise interrupts/exceptions and speculation: in-order completion and in-order commit
  - If we speculate and are wrong, need to back up and restart execution to point at which we predicted incorrectly
  - This is exactly same as need to do with precise exceptions
- Exceptions are handled by not recognizing the exception until instruction that caused it is ready to commit in ROB
  - If a speculated instruction raises an exception, the exception is recorded in the ROB
  - This is why reorder buffers in all new processors