



Tomasulo's Algorithm is an example of *dynamic scheduling*. In dynamic scheduling the ID - WB stages of the five-stage RISC pipeline are split into three stages to allow for out-of-order execution:

1. *Issue* - decodes instructions and checks for structural hazards. Instructions are issued in-order through a FIFO queue to maintain correct data flow. If there is not a free reservation station of the appropriate type, the instruction queue stalls.
2. *Read operands* - waits until no data hazards, then read operands
3. *Write result* - send the result to the CDB to be grabbed by any waiting register or reservation stations

All instructions pass through the issue stage in order, but instructions stalling on operands can be bypassed by later instructions whose operands are available.

RAW hazards are handled by delaying instructions in reservation stations until all their operands are available.

WAR and WAW hazards are handled by renaming registers in instructions by reservation station numbers.

Load and Store instructions to different memory addresses can be done in any order, but the relative order of a Store and accesses to the same memory location must be maintained. One way to perform *dynamic disambiguation* of memory references, is to perform effective address calculations of Loads and Stores in program order in the issue stage.

- Before issuing a Load from the instruction queue, make sure that its effective address does not match the address of any Store instruction in the Store buffers. If there is a match, stall the instruction queue until, the corresponding Store completes. (Alternatively, the Store could forward the value to the corresponding Load)
- Before issuing a Store from the instruction queue, make sure that its effective address does not match the address of any Store or Load instructions in the Store or Load buffers.