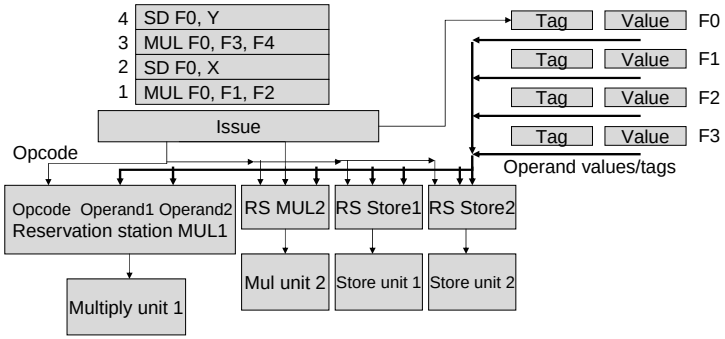
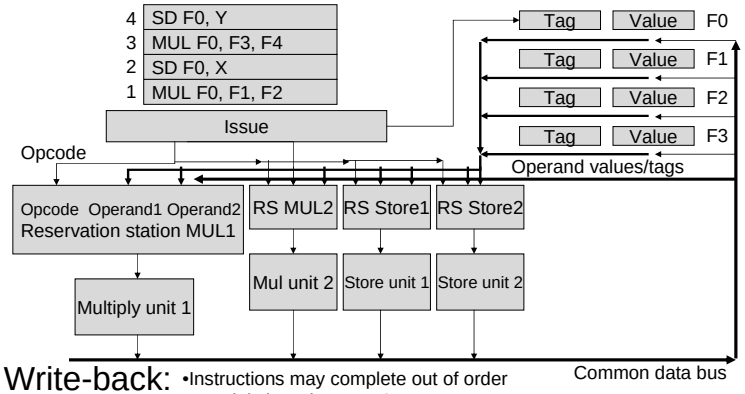
Tomasulo without Re-order Buffer

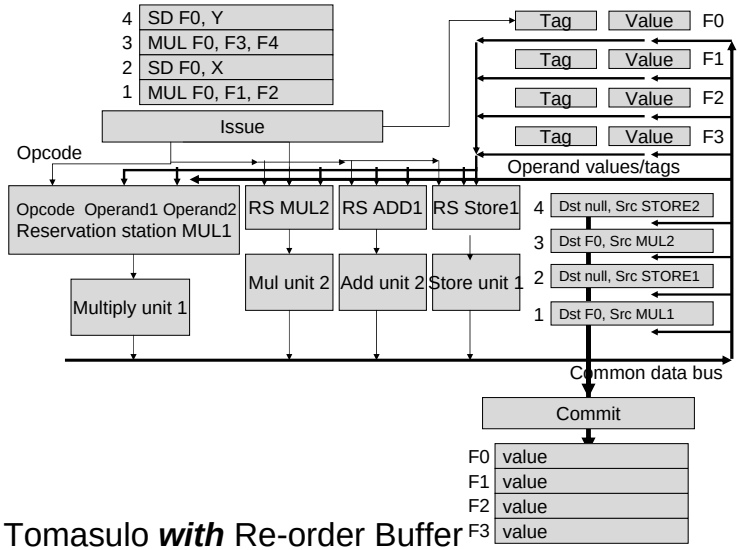


Issue:
- Each instruction is issued in order
- Issue unit collects operands from the two instruction's source registers
- Result may be a value, or, if value will be computed by an uncompleted instruction, the tag of the RS to which it was issued.
- **When instruction 1 is issued, F0 is updated to get result from MUL1**
- **When instruction 3 is issued, F0 is updated to get result from MUL2**

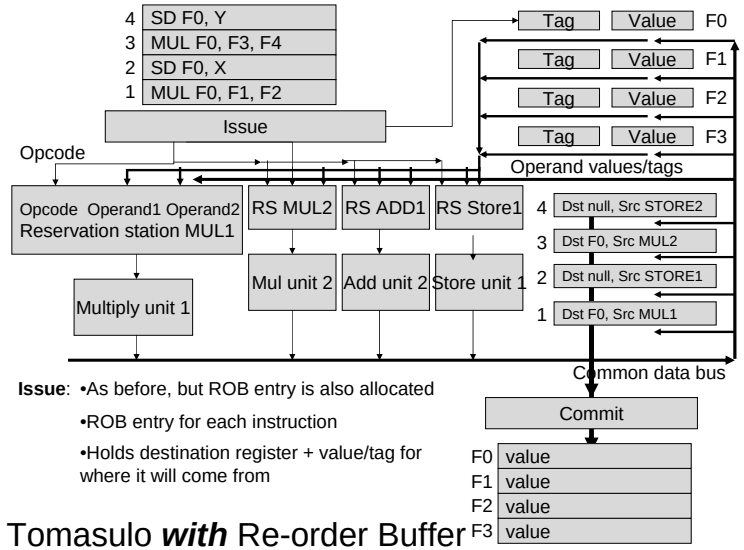
Tomasulo without Re-order Buffer



Write-back:
- Instructions may complete out of order
- Result is broadcast on CDB
- Carrying tag of RS to which instruction was originally issued
- All RSs and registers monitor CDB and collect value if tag matches
- Any RS which has both operands and whose FU is free fires.
- **When MUL1 completes result goes to store unit but not F0**

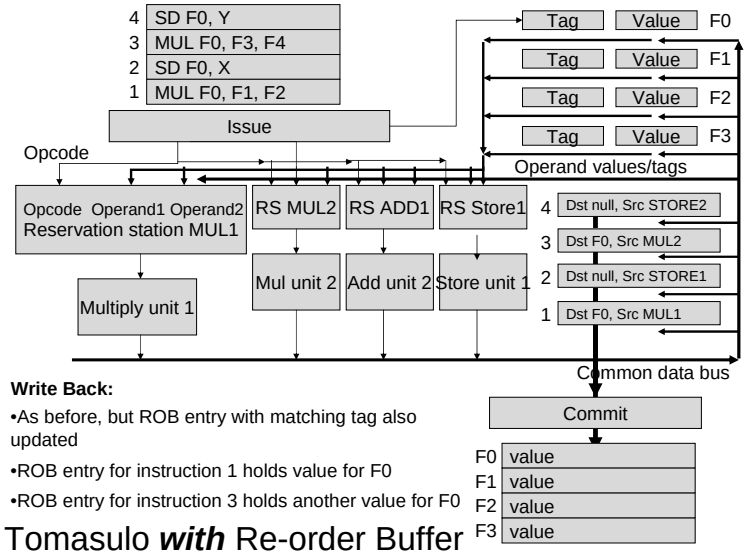


Tomasulo ***with*** Re-order Buffer



Issue:
- As before, but ROB entry is also allocated
- ROB entry for each instruction
- Holds destination register + value/tag for where it will come from

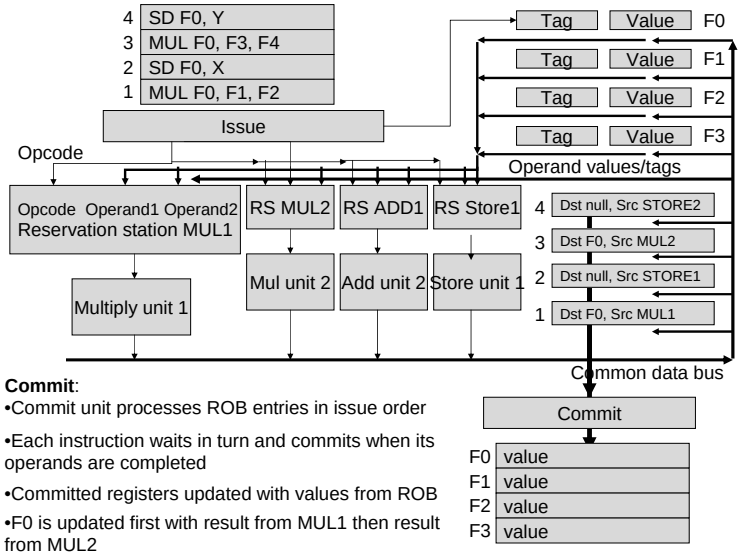
Tomasulo ***with*** Re-order Buffer



Write Back:

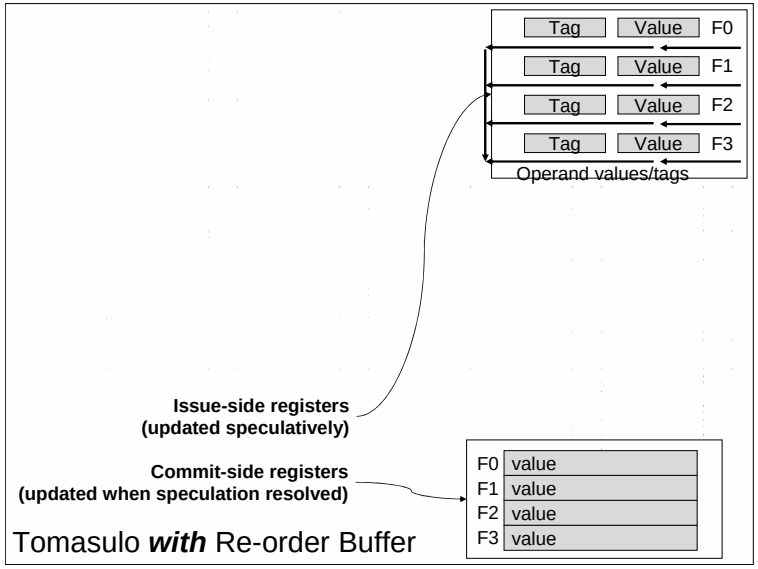
- As before, but ROB entry with matching tag also updated
- ROB entry for instruction 1 holds value for F0
- ROB entry for instruction 3 holds another value for F0

Tomasulo ***with*** Re-order Buffer

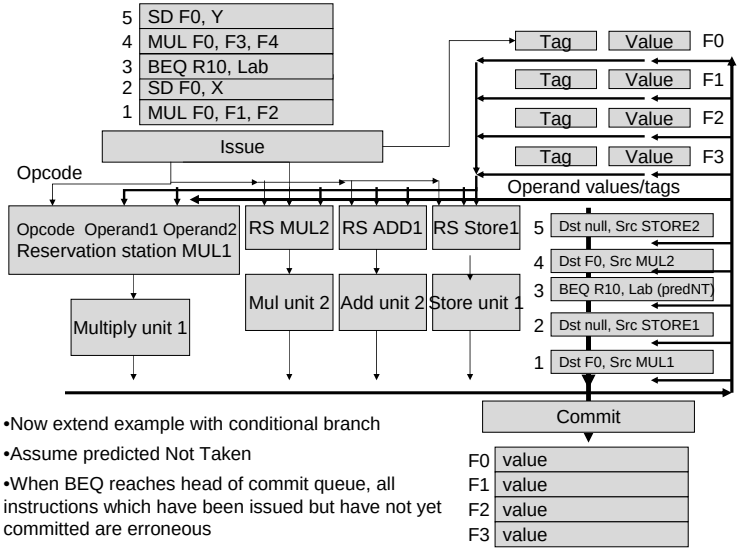


Commit:

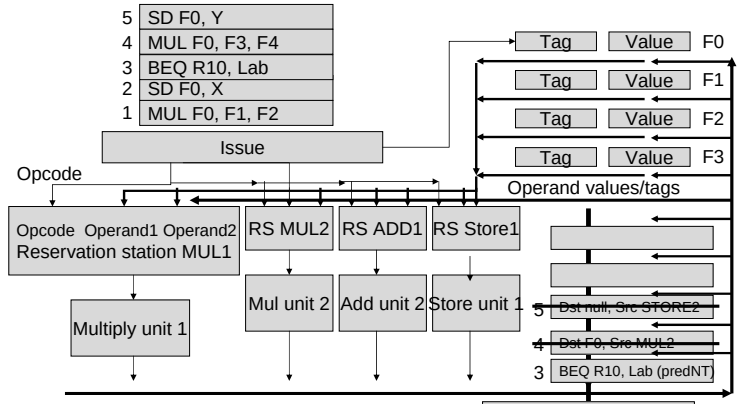
- Commit unit processes ROB entries in issue order
- Each instruction waits in turn and commits when its operands are completed
- Committed registers updated with values from ROB
- F0 is updated first with result from MUL1 then result from MUL2



Tomasulo ***with*** Re-order Buffer



- Now extend example with conditional branch
- Assume predicted Not Taken
- When BEQ reaches head of commit queue, all instructions which have been issued but have not yet committed are erroneous

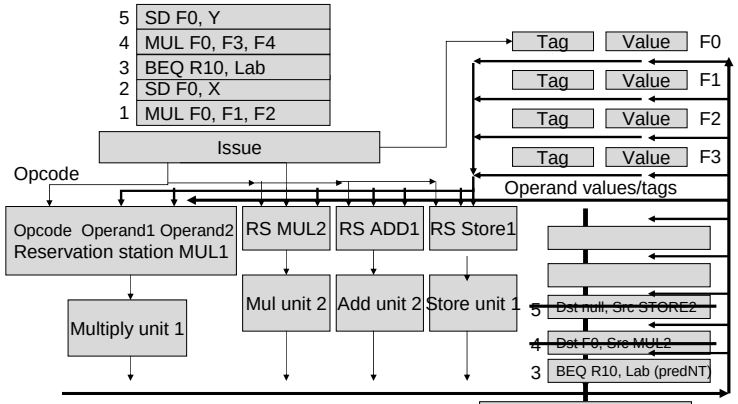


•**Misprediction**: all ROB entries are trashed

•Issue-side registers reset from commit-side registers

•Correct branch target instruction fetched and issued

F0	Value from MUL1
F1	value
F2	value
F3	value



•Committed F0 holds value from first MUL

•RS of uncompleted speculatively-executed instruction cannot be re-used until its FU (eg MUL2) completes

F0	Value from MUL1
F1	value
F2	value
F3	value