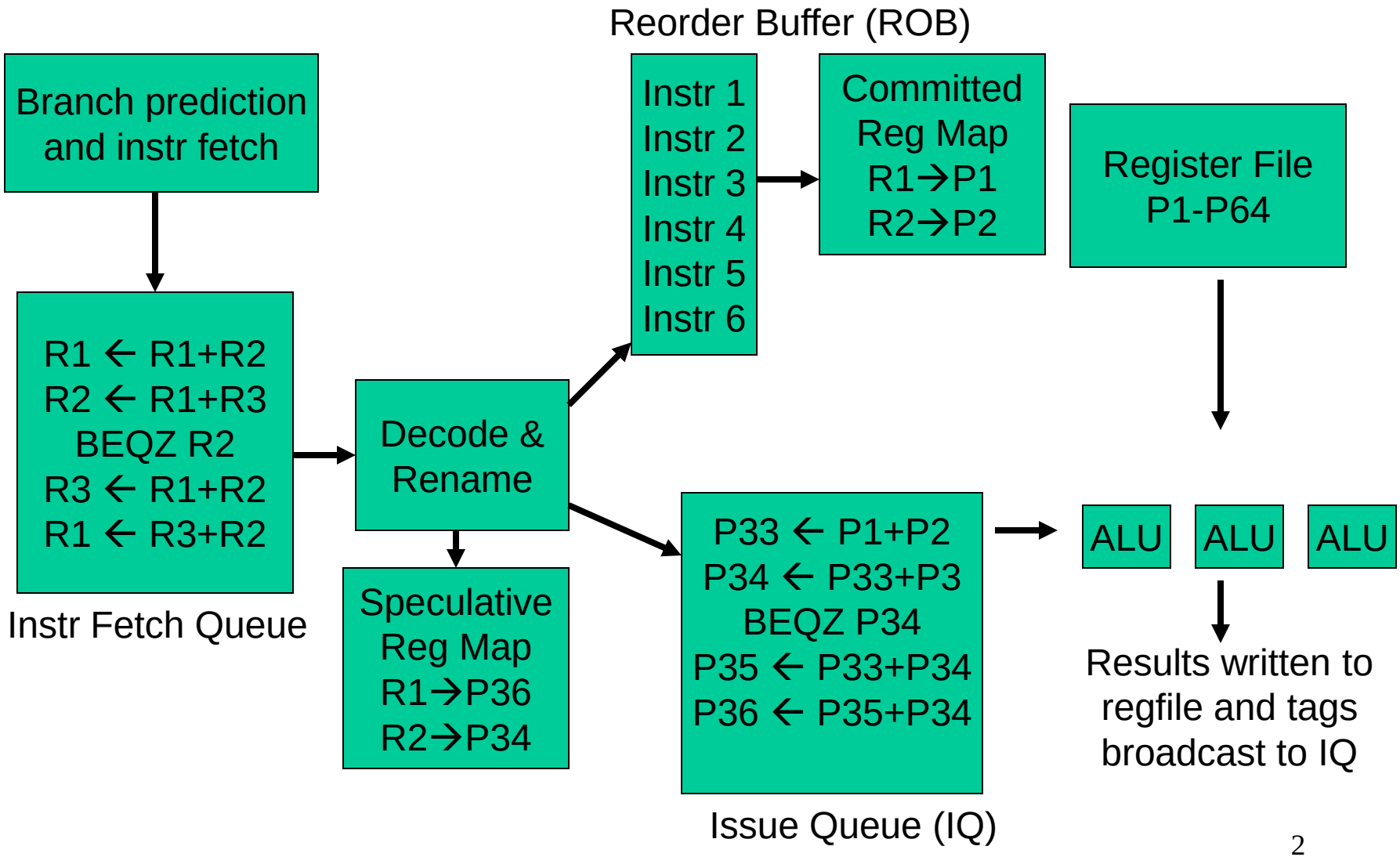
Lecture: Out-of-order Processors

- Topics: more ooo design details, timing, load-store queue

The Alpha 21264 Out-of-Order Implementation



Additional Details

- When does the decode stage stall? When we either run out of registers, or ROB entries, or issue queue entries
- Issue width: the number of instructions handled by each stage in a cycle. High issue width ➔ high peak ILP
- Window size: the number of in-flight instructions in the pipeline. Large window size ➔ high ILP
- No more WAR and WAW hazards because of rename registers – must only worry about RAW hazards

Branch Mispredict Recovery

- On a branch mispredict, must roll back the processor state: throw away IFQ contents, ROB/IQ contents after branch
- Committed map table is correct and need not be fixed
- The speculative map table needs to go back to an earlier state
- To facilitate this spec-map-table rollback, it is checkpointed at every branch

Waking Up a Dependent

- In an in-order pipeline, an instruction leaves the decode stage when it is known that the inputs can be correctly received, not when the inputs are computed

- Similarly, an instruction leaves the issue queue before its inputs are known, i.e., wakeup is speculative based on the expected latency of the producer instruction

Out-of-Order Loads/Stores

Ld	R1 ← [R2]
Ld	R3 ← [R4]
St	R5 → [R6]
Ld	R7 ← [R8]
Ld	R9←[R10]

What if the issue queue also had load/store instructions?
Can we continue executing instructions out-of-order?

Memory Dependence Checking

Ld	0x abcdef
Ld	
St	
Ld	
Ld	0x abcdef
St	0x abcd00
Ld	0x abc000
Ld	0x abcd00

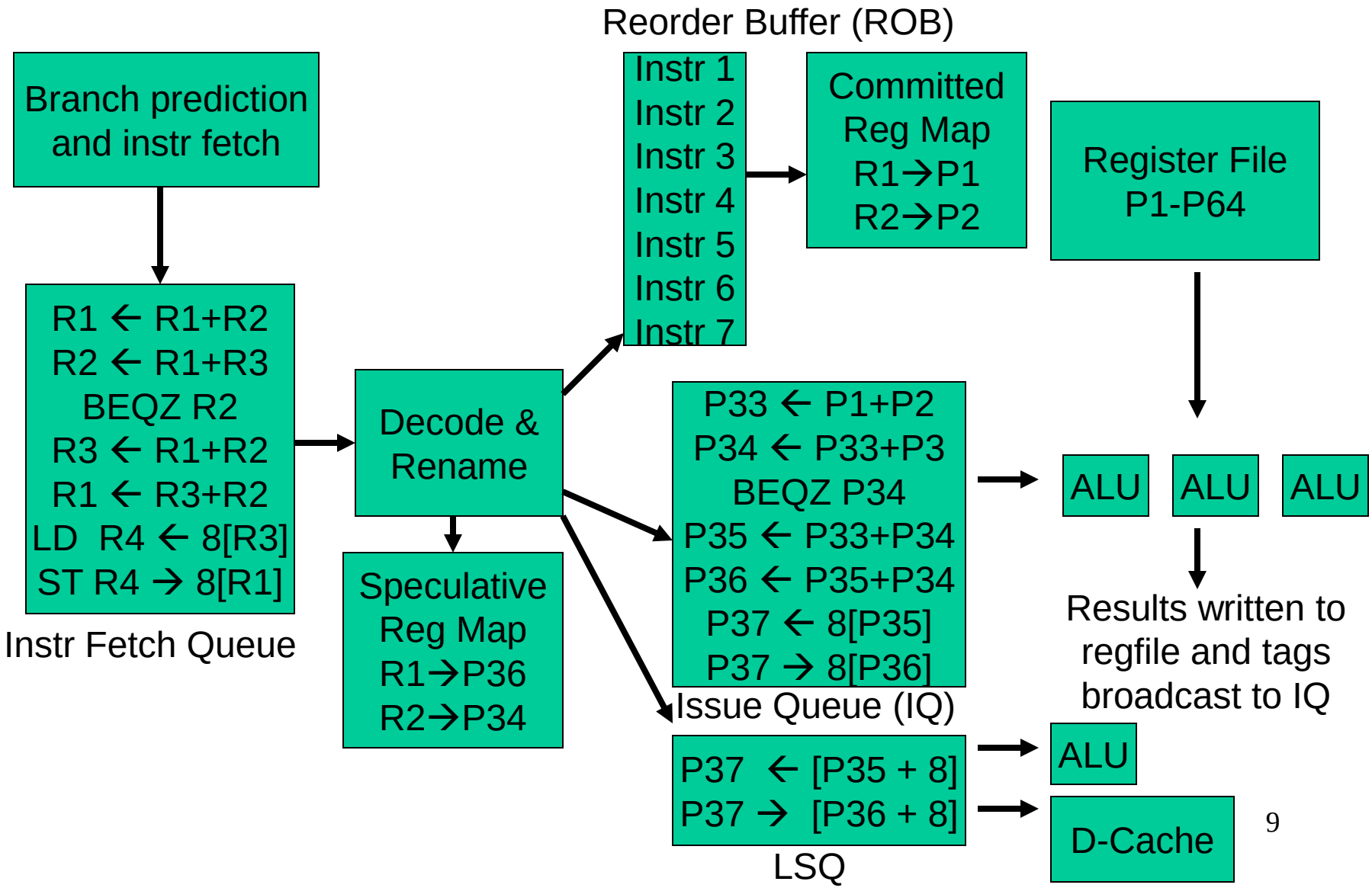
- The issue queue checks for register dependences and executes instructions as soon as registers are ready
- Loads/stores access memory as well – must check for RAW, WAW, and WAR hazards for memory as well
- Hence, first check for register dependences to compute effective addresses; then check for memory dependences

Memory Dependence Checking

Ld	0x abcdef
Ld	
St	
Ld	
Ld	0x abcdef
St	0x abcd00
Ld	0x abc000
Ld	0x abcd00

- Load and store addresses are maintained in program order in the Load/Store Queue (LSQ)
- Loads can issue if they are guaranteed to not have true dependences with earlier stores
- Stores can issue only if we are ready to modify memory (can not recover if an earlier instr raises an exception)

The Alpha 21264 Out-of-Order Implementation



Title

- Bullet