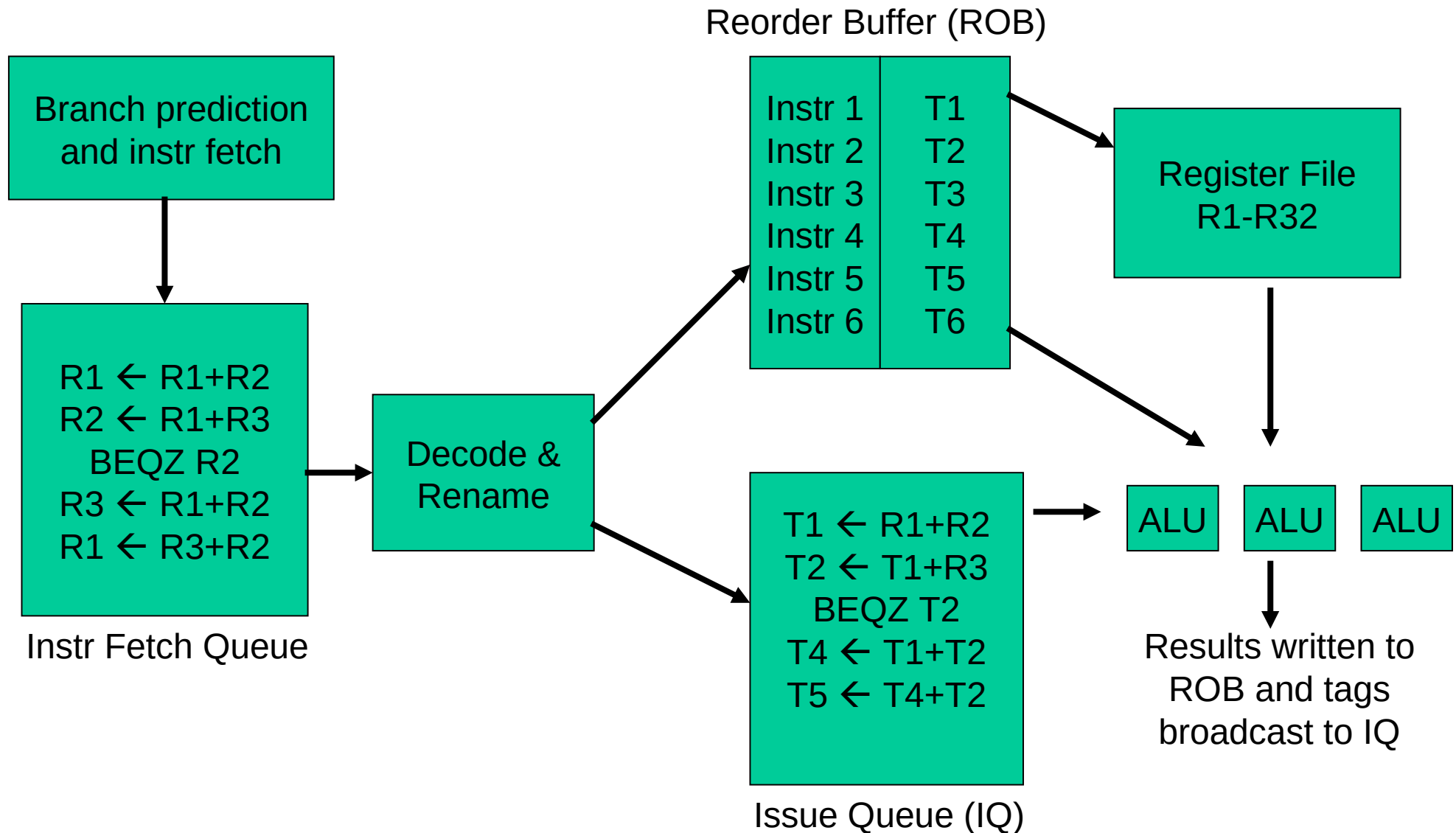


# CS/EE 3810: Computer Organization

## Lecture 14: Out-of-order execution

Anton Burtsev  
November, 2022

# An Out-of-Order Processor Implementation

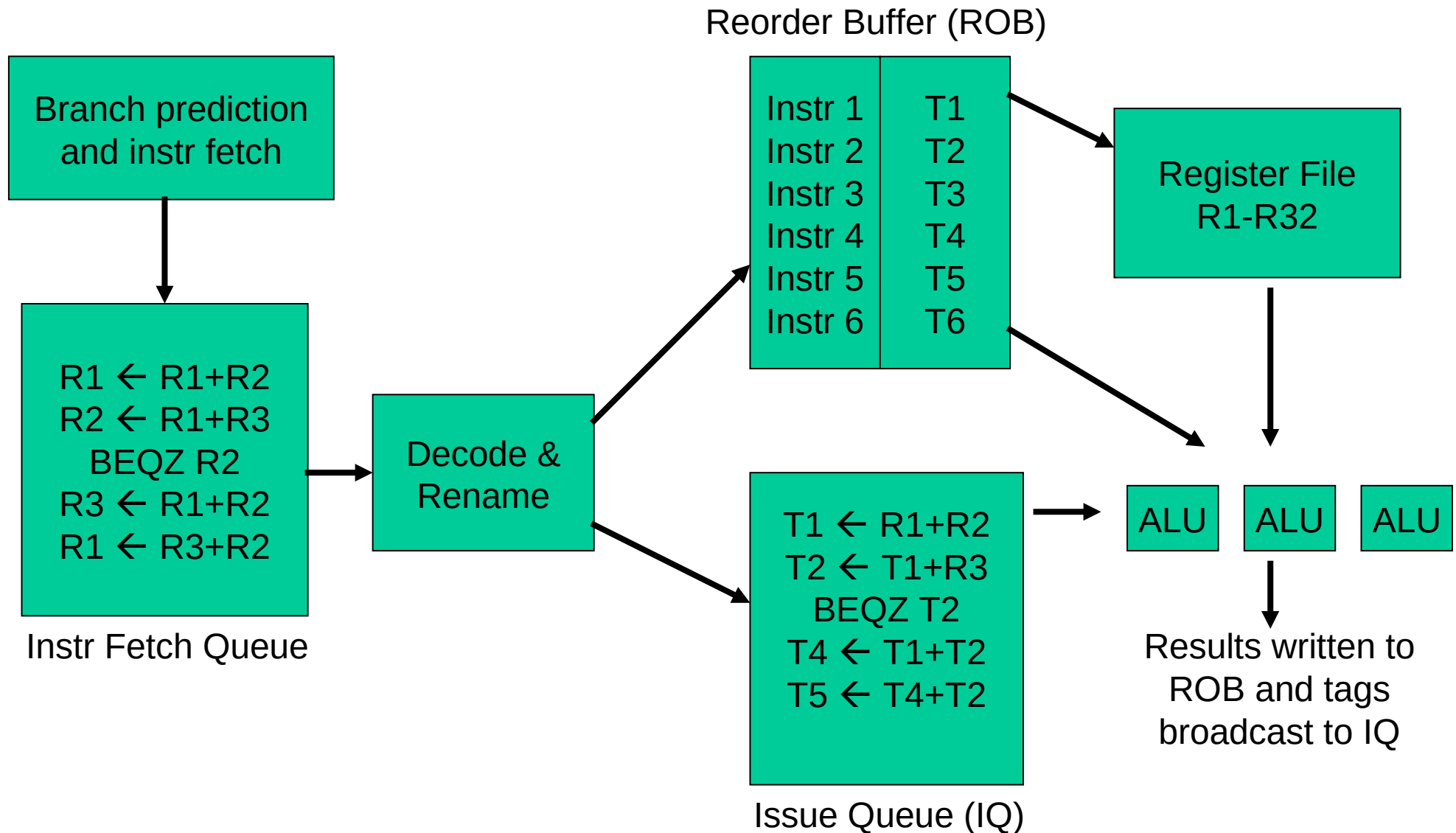


## Example Code

---

Completion times	with in-order	with ooo
ADD R1, R2, R3	5	5
ADD R4, R1, R2	6	6
LW R5, 8(R4)	7	7
ADD R7, R6, R5	9	9
ADD R8, R7, R5	10	10
LW R9, 16(R4)	11	7
ADD R10, R6, R9	13	9
ADD R11, R10, R9	14	10

# An Out-of-Order Processor Implementation



# Design Details - I

- Instructions enter the pipeline in order
- No need for branch delay slots if prediction happens in time
- Instructions leave the pipeline in order – all instructions that enter also get placed in the ROB – the process of an instruction leaving the ROB (in order) is called commit – an instruction commits only if it and all instructions before it have completed successfully (without an exception)
- To preserve precise exceptions, a result is written into the register file only when the instruction commits – until then, the result is saved in a temporary register in the ROB

# Design Details - II

- Instructions get renamed and placed in the issue queue – some operands are available (T1-T6; R1-R32), while others are being produced by instructions in flight (T1-T6)
- As instructions finish, they write results into the ROB (T1-T6) and broadcast the operand tag (T1-T6) to the issue queue – instructions now know if their operands are ready
- When a ready instruction issues, it reads its operands from T1-T6 and R1-R32 and executes (out-of-order execution)
- Can you have WAW or WAR hazards? By using more names (T1-T6), name dependences can be avoided

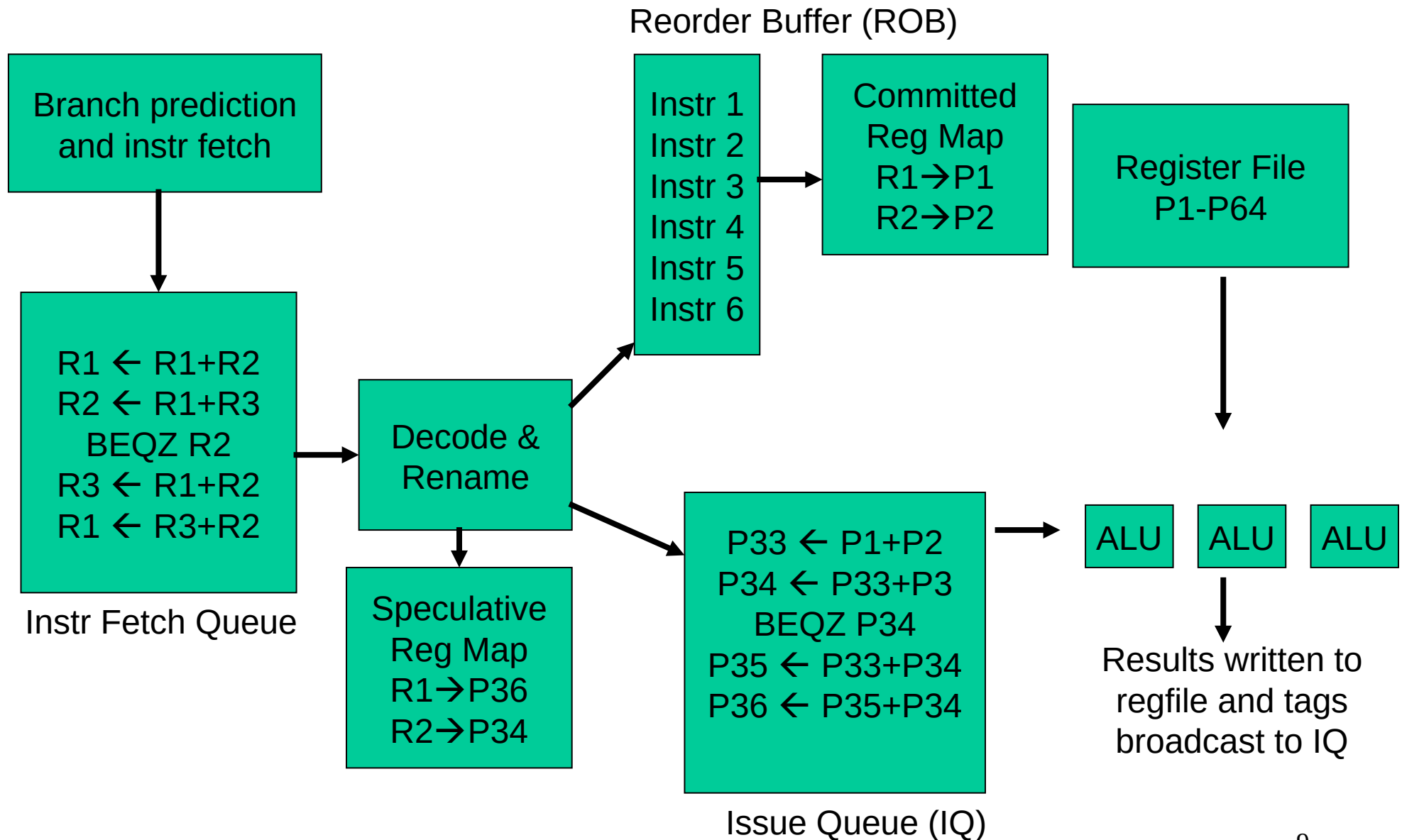
# Design Details - III

- If instr-3 raises an exception, wait until it reaches the top of the ROB – at this point, R1-R32 contain results for all instructions up to instr-3 – save registers, save PC of instr-3, and service the exception
- If branch is a mispredict, flush all instructions after the branch and start on the correct path – mispredicted instrs will not have updated registers (the branch cannot commit until it has completed and the flush happens as soon as the branch completes)
- Potential problems: ?

# Alpha 21264 Out-of-Order Implementation



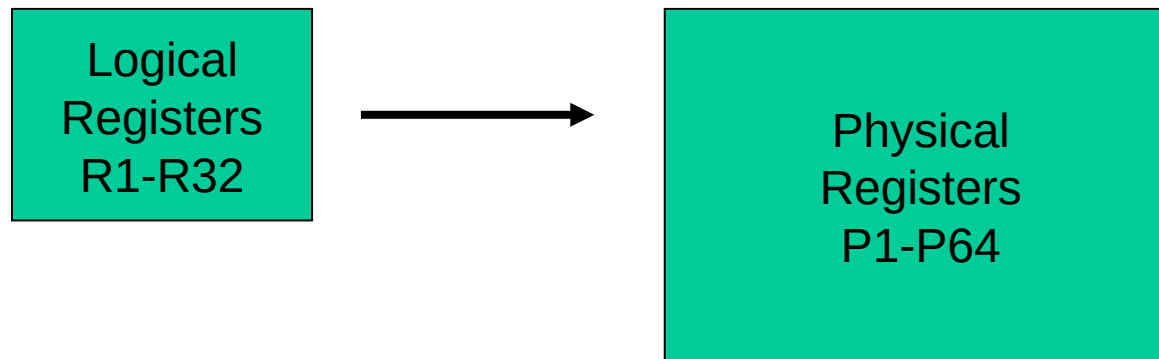
# The Alpha 21264 Out-of-Order Implementation



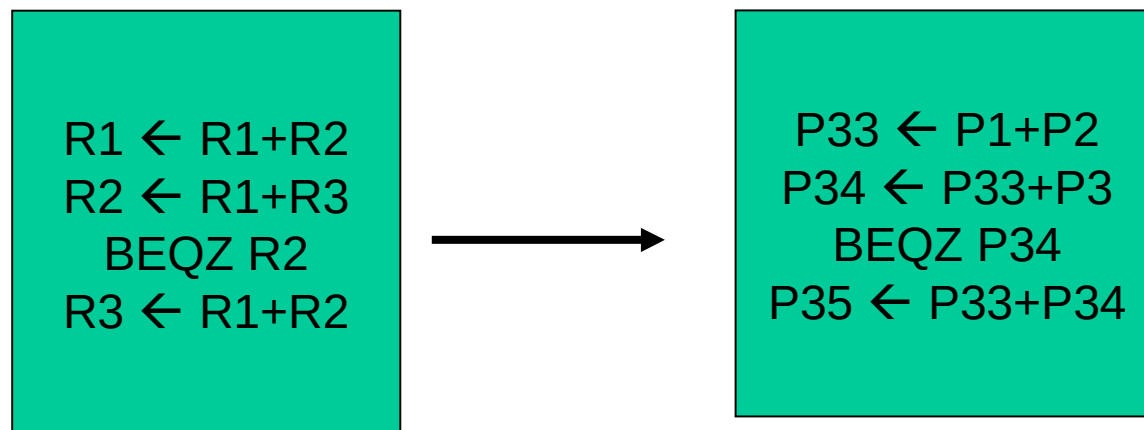
# Logical and physical registers

# Managing Register Names

Temporary values are stored in the register file and not the ROB



At the start, R1-R32 can be found in P1-P32  
Instructions stop entering the pipeline when P64 is assigned



What happens on commit?

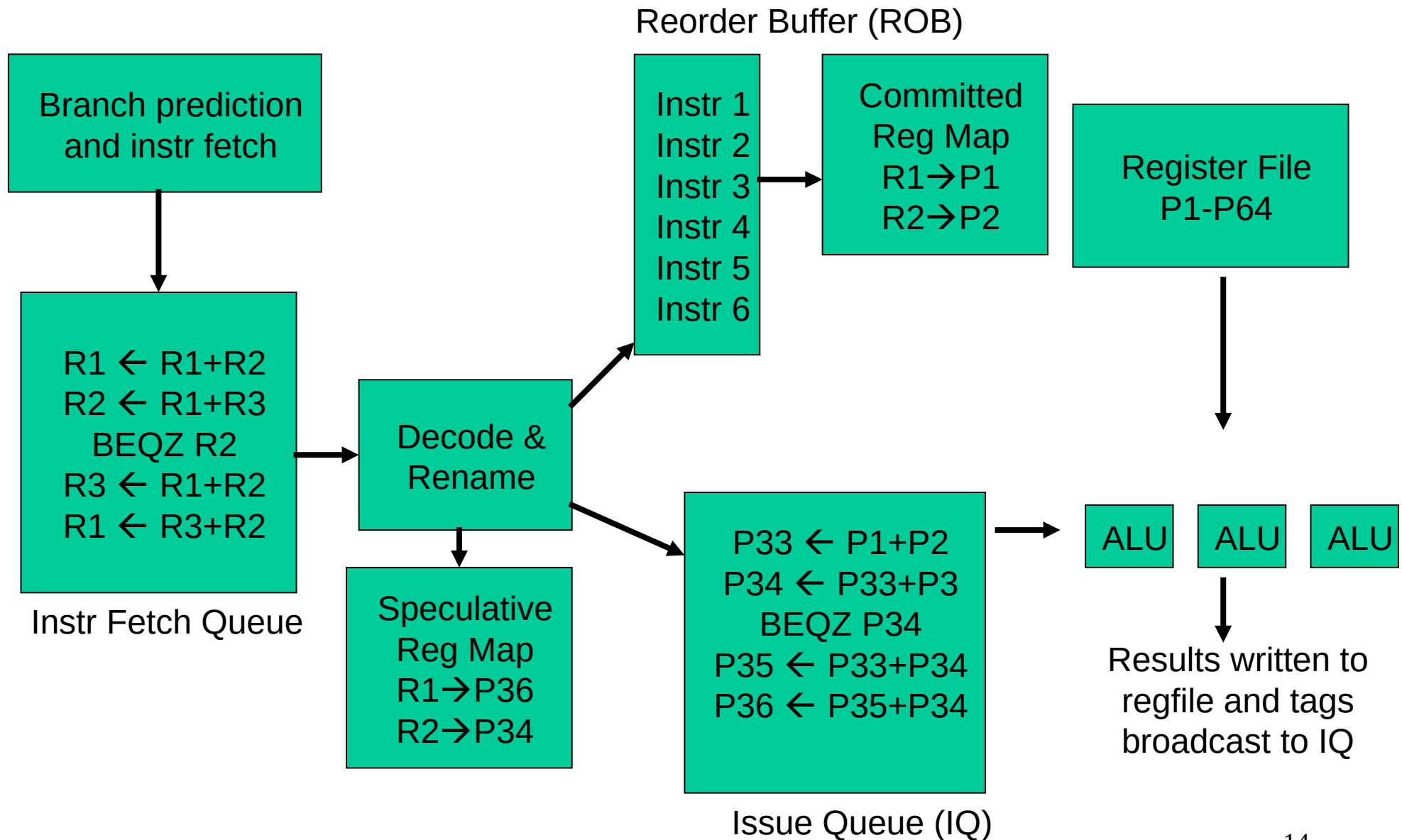
# The Commit Process

- On commit, no copy is required
- The register map table is updated – the “committed” value of R1 is now in P33 and not P1 – on an exception, P33 is copied to memory and not P1
- An instruction in the issue queue need not modify its input operand when the producer commits
- When instruction-1 commits, we no longer have any use for P1 – it is put in a free pool and a new instruction can now enter the pipeline → for every instr that commits, a new instr can enter the pipeline → number of in-flight instrs is a constant = number of extra (rename) registers

# 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

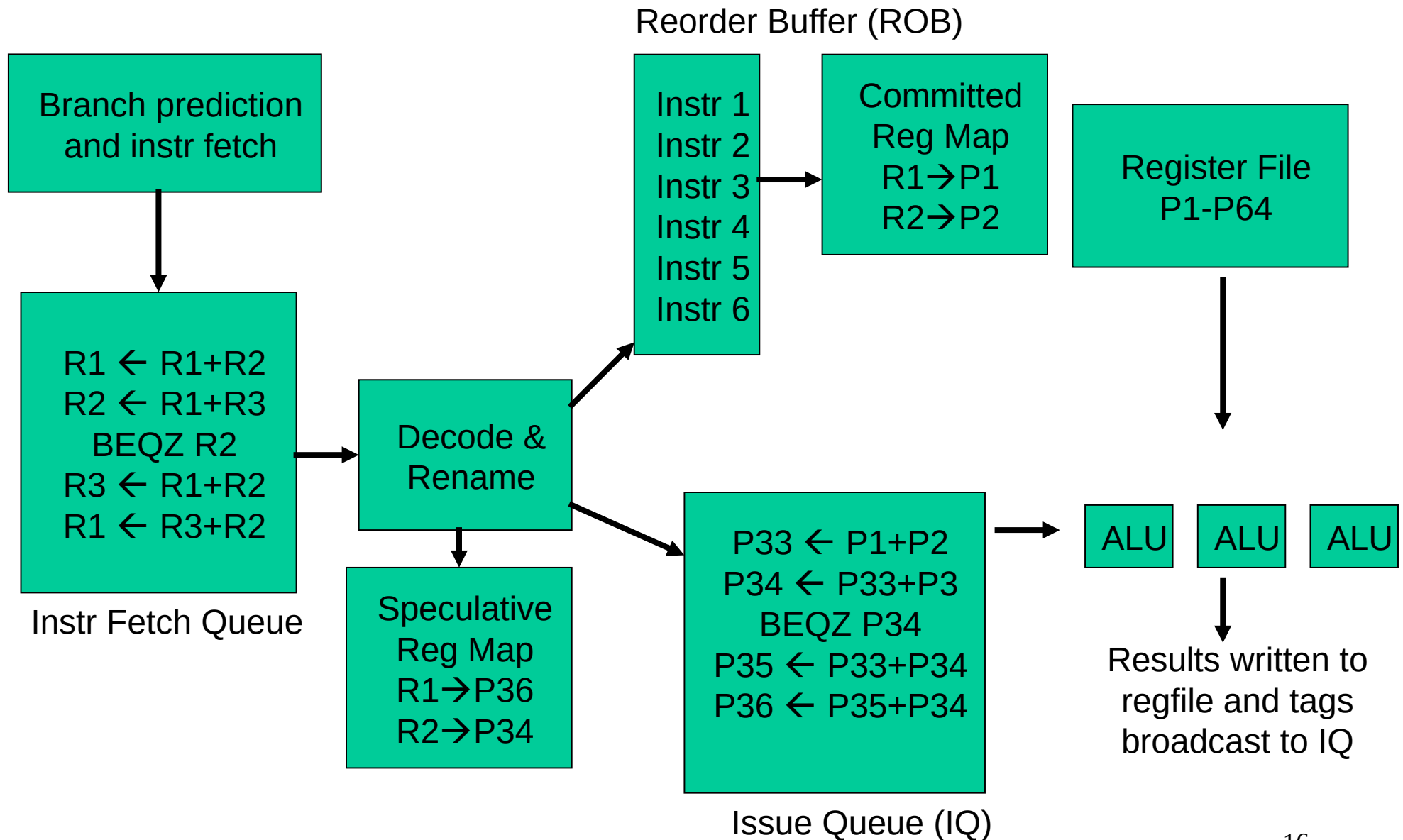
# The Alpha 21264 Out-of-Order Implementation



# 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

# The Alpha 21264 Out-of-Order Implementation





Thank you!