counter-track out-of-order implementation

The counter-based OoO execution uses counters to track instruction dependency in OoO by replacing the traditional valid and tag registers with read and write counter registers and the reservation station's with a single count register, incremented or decremented by adders and subtractors. Instead of tagging values and broadcasting validity via the Common Data Bus (CDB), you maintain read and write counters for each register. The counter-based system tracks how many instructions are reading from or writing to a register, with increment/decrement operations ensuring proper synchronization. A single count register tracks the number of unresolved dependencies for each instruction. The instruction becomes ready for execution when this count reaches zero.

Advantages over traditional tomasolu's approach

1. Improved scalability and Simplified Dependency Tracking

- Multiple processes can have separate sizeable reservation stations which reduces the number of rows to read during write back, this reduces contention on the common data bus(CBD)
- Using counters eliminates the need for explicit tags and broadcasting.

2. Reduced Hardware Complexity (Potentially)

- Fewer fields (valid, tag, and value) in the reservation stations may reduce hardware resource usage.
- A single counter replaces multiple registers, simplifying operand tracking.

Proof of concept

This approach eliminates the need for traditional value and tag-based dependency resolution by using read and write counters in the Register Alias Table (RAT) and a count-based mechanism in the Reservation Stations (RS).

Instruction to test

Mul R1, R2, -> R3

Add R3, R4, -> R5

Add R2, R6, -> R7

Add R8, R9, -> R10

Mul R7, R10, -> R11

Add R5, R11, -> R5

The RAT tracks each register with the following fields:

- **Read Count:** Number of active instructions reading this register.
- **Write Count:** Number of active instructions writing to this register.

The value field can be read directly from the register files. This enables multiples processes to run multiple independent short RAT and reservation station. This makes the RAT and reservation station sizeable for search which solves the common databus bottle neck found in traditional tomasolu's algorithm.

The reservation station tracks:

- **S1, S2:** Source registers.

- **R:** Destination register.
- **Count:** Number of unresolved dependencies for the instruction. It sums the write count field of the source registers with the read count field of the destination register from the RAT and stores in the count field.

Analysis of the Example Sequence

The first instruction (MUL R1, R2 -> R3) has no unresolved dependencies (Count = 0) and executes immediately.

RAT and RS

register	Read count	Write count		S1	S2	R	count
R1	1	0		R1	R2	R3	0
R2	1	0					
R3	0	1					

The second instruction (ADD R3, R4 -> R5) depends on instr1 to read r3 value (Count = 1) .

register	Read count	Write count		S1	S2	R	count
R3	1	1		R1	R2	R3	0
R4	1	0		R3	R4	R5	1
R5	0	1					

The 3rd instruction (ADD R2, R6 -> R7) has no unresolved dependencies (Count = 0) and executes immediately.

register	Read count	Write count		S1	S2	R	count
R2	2	0		R1	R2	R3	0
R6	1	0		R3	R4	R5	1

R7	0	1		R2	R6	R7	0

The fourth instruction (ADD R8, R9 -> R10) has no unresolved dependencies (Count = 0) and executes immediately.

register	Read count	Write count		S1	S2	R	count
R8	1	0		R1	R2	R3	0
R9	1	0		R3	R4	R5	1
R10	0	1		R2	R6	R7	0
				R8	R9	R10	0

The fifth instruction (MUL R7, R10 -> R11) depends on the 3rd and 4th instr to read r7 and r10 (Count = 2).

register	Read count	Write count		S1	S2	R	count
R7	1	1		R1	R2	R3	0
R10	1	1		R3	R4	R5	1
R11	0	1		R2	R6	R7	0
				R8	R9	R10	0
				R7	R10	R11	2

The sixth instruction (ADD R5, R11 -> R5) depends on the 2nd and 5th instruction to read r5 and r11 (Count = 2) and executes immediately.

register	Read count	Write count		S1	S2	R	count
R5	1	2		R1	R2	R3	0
R11	1	1		R3	R4	R5	1
				R2	R6	R7	0
				R8	R9	R10	0
				R7	R10	R11	2
				R5	R11	R5	2

Final RAT for counter track vs tomasolus algorithm

Counter track				**Tomasolu**			
register	Read count	Write count		register	valid	tag	value
R1	1			R1	1		1
R2	2			R2	1		2
R3	1	1		R3	1		3
R4	1			R4	1		4
R5	1	2		R5	0	d	
R6	1			R6	1		6
R7	1	1		R7	0	b	
R8	1			R8	1		8
R9	1			R9	1		9
R10	1	1		R10	0	c	10
R11	1			R11	0	y	11

Final RS for counter track vs tomasolus algorithm

Counter track					**Tomasolu**					
S1	S2	R	count		s/n	valid	tag	value	valid	tag	value
R1	R2	R3	0		a	1	~	2	1	~	4
R3	R4	R5	1		b	1	~	2	1	~	6
R2	R6	R7	0		c	1	~	8	1	~	9
R8	R9	R10	0		d	0	a		0	y	
R7	R10	R11	2		x	1	~	1	1	~	2
R5	R11	R5	2		y	0	b		0	c	

After execution:

The first instruction (MUL R1, R2 -> R3) decrements read count for R1 and R2, and write count for R3 in the RAT. In the RS the second instr. Is decremented by 1, and second instr. executes immediately.

register	Read count	Write count		S1	S2	R	count
R1	0	0					
R2	1	0		R3	R4	R5	0
R3	1	0		R2	R6	R7	0
				R8	R9	R10	0
				R7	R10	R11	2
				R5	R11	R5	2

The second instruction (ADD R3, R4 -> R5) decrements read count for R3 and R4, and write count for R5 in the RAT. In the RS the sixth instr. Is decremented by 1.

register	Read count	Write count		S1	S2	R	count
R3	0	0					
R4	0	0					
R5	1	1		R2	R6	R7	0
				R8	R9	R10	0
				R7	R10	R11	2
				R5	R11	R5	1

The third instruction (ADD R2, R6 -> R7) decrements read count for R2 and R6, and write count for R7 in the RAT. In the RS the fifth instr. Is decremented by 1.

register	Read count	Write count		S1	S2	R	count
R2	0	0					
R6	0	0					

R7	1	0					
				R8	R9	R10	0
				R7	R10	R11	1
				R5	R11	R5	1

The fourth instruction (ADD R8, R9 -> R10) decrements read count for R8 and R9, and write count for R10 in the RAT. In the RS the fifth instr. Is decremented by 1.

register	Read count	Write count		S1	S2	R	count
R8	0	0					
R9	0	0					
R10	1	0					
				R7	R10	R11	0
				R5	R11	R5	1

The fifth instruction (MUL R7, R10 -> R11) decrements read count for R7 and R10, and write count for R11 in the RAT. In the RS the sixth instr. Is decremented by 1.

register	Read count	Write count		S1	S2	R	count
R7	0	0					
R10	0	0					
R11	1	0					
				R5	R11	R5	0

The sixth instruction (ADD R5, R11 -> R5) decrements read count for R5 and R11, and write count for R5 in the RAT.

register	Read count	Write count		S1	S2	R	count
R5	0	0					

R11	0	0					