OOO execution with Tomasulo's algorithm

Tomasulo's algorithm intuition

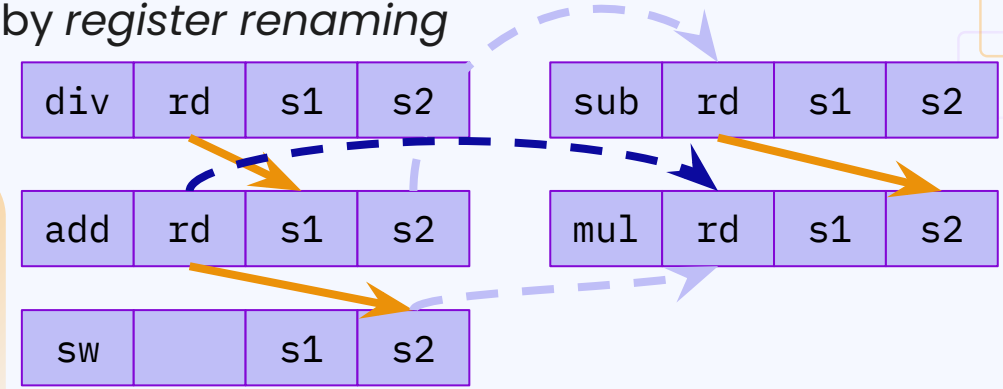
Developed by Robert Tomasulo for IBM 360/91

Minimize RAW hazards by tracking data dependences and reordering

Minimize WAR and WAW hazards by *register renaming*

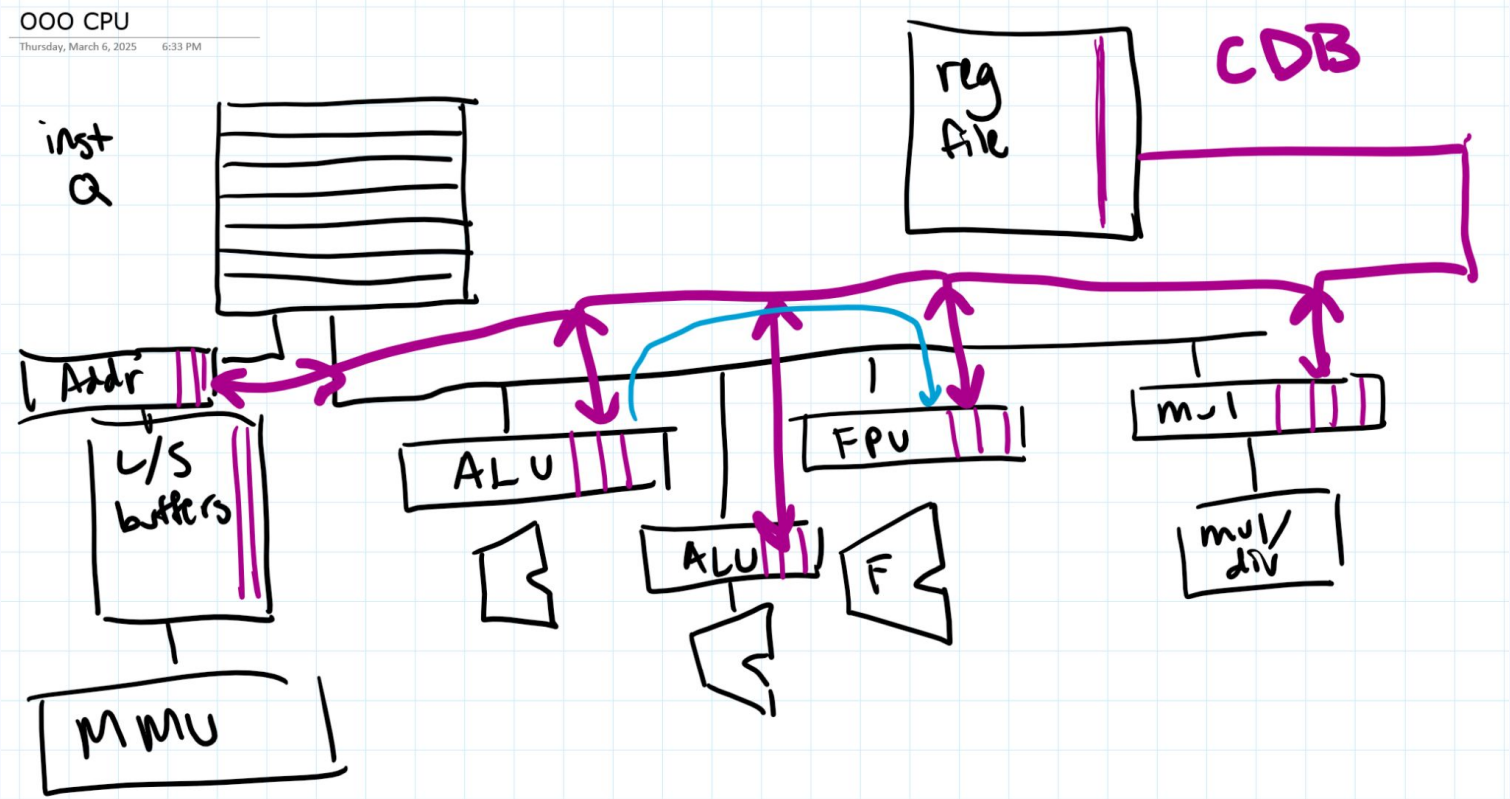
```
div t0, t1, t2
add t3, t0, t4
sw t3, 0(s0)
sub t4, t5, t6
mul t3, t5, t4
```

What if the hardware could track dependences and temporarily store results?



now the div might have enough time to write to t0 before add needs it!

The OOO CPU(sketch of H&P fig 3.6)



Three stages of execution

Issue

- If reservation station available, issue to station; tracking source data (available and copied over or waiting on other reservation station)
- If dest is register, update register file tags to say data will come from this res. station

Execute

- If operands available, execute instr; otherwise wait

Write

- Once result is ready, send on CDB (update registers, reservation station sources waiting on result)

Worksheets for today

Link

Execute latencies for our examples:

Mul/div: 3 cycles

Load/store: 3 cycles (1 for addr, 2 for load/store)

ALU: 1 cycle

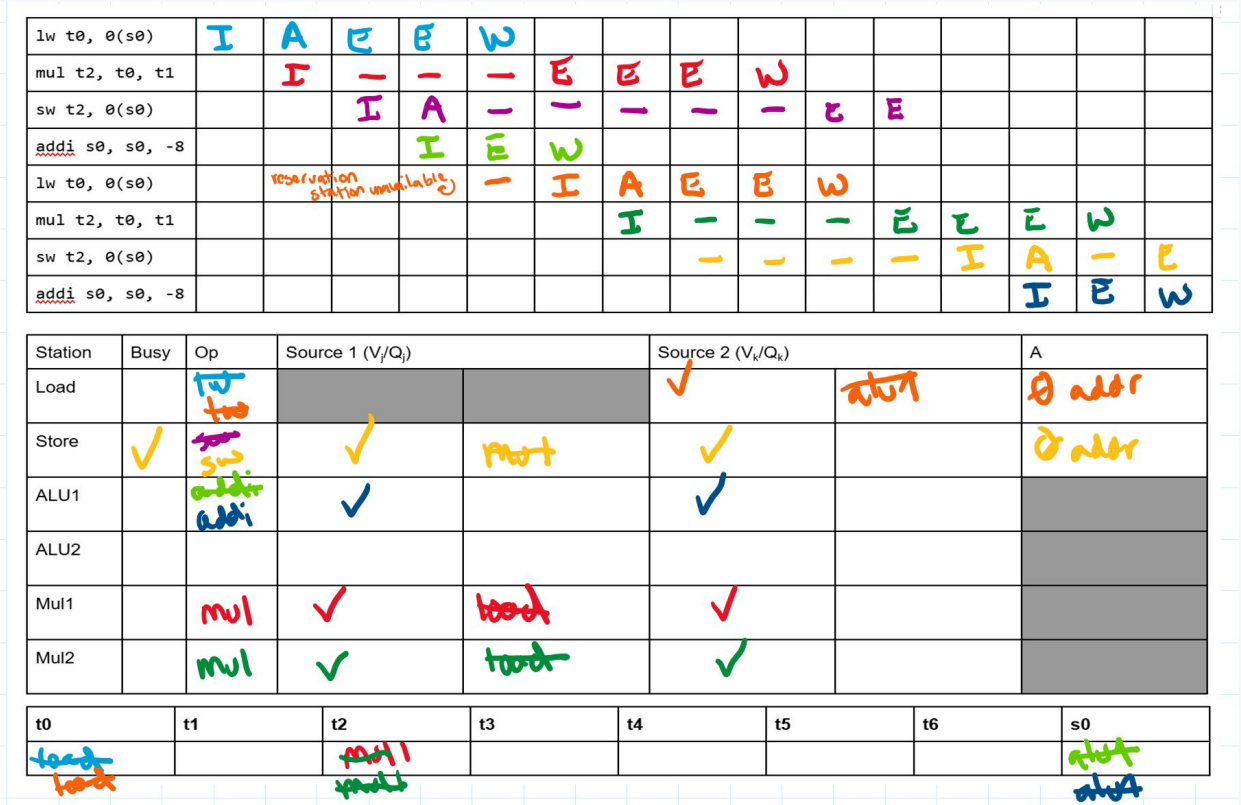
Example 1

div t0, t1, t2	I	E	E	E	W										
add t3, t0, t4		I	–	–	←	E	W								
sw t3, 0(s0)			I	A	←	–	←	E	E						
sub t4, t5, t6				I	E	W									
mul t3, t5, t4					I	←	E	E	E	W					

Station	Busy	Op	Source 1 (V_j/Q_j)		Source 2 (V_k/Q_k)		A
Load							
Store		~~sw~~	√ (from CDB)	alu1	√ (from s0)		0 addr
ALU1		~~addi~~	√ Copied from CDB	~~mul1~~	√ (C from t4)		
ALU2		~~sub~~	√		√		
Mul1		~~mul~~	√ (copied from t1)		√ (copied from t2)		
Mul2		~~mul~~	√		√ (from CDB)	alu2	

t0 √	t1	t2	t3 √	t4 √	t5	t6	s0
~~mul1~~			~~alu1~~ mul2	~~alu2~~			

Example 2: unrolled loop



lw t0, 0(s0)	I	A	E	E	W										
mul t2, t0, t1		I	–	–	–	E	E	E	W						
sw t2, 0(s0)			I	A	–	–	–	–	–	E	E				
addi s0, s0, -8				I	E	W									
lw t0, 0(s0)		reservation station unavailable			–	I	A	E	E	W					
mul t2, t0, t1							I	–	–	–	E	E	E	W	
sw t2, 0(s0)								–	–	–	–	I	A	–	E
addi s0, s0, -8													I	E	W

Station	Busy	Op	Source 1 (V_j/Q_j)		Source 2 (V_k/Q_k)		A
Load		lw			✓	alu1	0 addr
Store	✓	~~lw~~ sw	✓	mul1	✓		0 addr
ALU1		addi	✓		✓		
ALU2							
Mul1		mul	✓	~~load~~	✓		
Mul2		mul	✓	~~load~~	✓		

t0	t1	t2	t3	t4	t5	t6	s0
~~load~~ load		~~mul1~~ mul2					alu1 alu2