CS152/252A
Spring 2026

Out of Order Execution

Discussion 6

1 Out of Order Execution

1.1 Why is out of order execution useful?

1.2 Consider the following code-snippet. What limits its out of order performance, assuming there are only four architectural registers?

```
A: fmul f1, f0, f2
B: fadd f0, f3, f1
C: fmul f3, f2, f3
D: fadd f3, f3, f1
```

2 Register Renaming

Rewrite the code snippet to use infinite physical registers called P0, P1, P2, ... and update the rename table accordingly.

Old code snippet:

```
A: fmul f1, f0, f2
B: fadd f0, f3, f1
C: fmul f3, f2, f3
D: fadd f3, f3, f1
```

New code snippet:

```
A: fmul __, __, __
B: fadd __, __, __
C: fmul __, __, __
D: fadd __, __, __
```

Register Rename Table		
Floating Point Register	Initial Physical Register	Updated Physical Register
f0	P0	
f1	P1	
f2	P2	
f3	P3	

3 Tomasulo's Algorithm

A Brief Overview of Tomasulo's Algorithm

On instruction dispatch (in program order):

- Allocate reservation station (RS) entry
- If source register has "present" (P) bit set in register file (RF) entry
  - Copy value into tag/data field in RS and set P bit for operand
  - Otherwise, copy tag from RF into RS and clear P bit for operand
- Replace RF entry for destination register with tag assigned to RS entry (tagdest)

Prior to execution:

- For missing operands, monitor result bus for tag match; replace tag with value; set P
- When all operands are present, issue to functional unit

On completion:

- Broadcast <tagdest, result> on result bus for RF and other RS entries to consume
- Deallocate RS entry

Other notes for this question:

- At most one instruction is dispatched per cycle
  - Can begin execution the same cycle as dispatch if all operands are present
- Fully-pipelined functional units
  - 2-cycle floating-point add latency
  - 3-cycle floating-point multiply latency
- Broadcasting result takes another cycle
  - Result bus can broadcast two results simultaneously

3.1 Simulate execution of the code in Q2 using Tomasulo's Algorithm. The code is copied for convenience.

```
A: fmul f1, f0, f2    # V3
B: fadd f0, f3, f1    # V4
C: fmul f3, f2, f3    # V5
D: fadd f3, f3, f1    # V6
```

Cycle 1

Instruction:

Adder Reservation Stations

	p	tag/data	p	tag/data
T0				
T1				
T2				

Multiplier Reservation Stations

	p	tag/data	p	tag/data
T3				
T4				

Register File

	p	tag/data
f0	1	V0
f1		
f2	1	V1
f3	1	V2

Adder

Stage	Destination Tag
1	
2	

Multiplier

Stage	Destination Tag
1	
2	
3	

Cycle 2

Instruction:

Register File

	p	tag/data
f0	1	V0
f1		
f2	1	V1
f3	1	V2

Adder Reservation Stations

	p	tag/data	p	tag/data
T0				
T1				
T2				

Adder

Stage	Destination Tag
1	
2	

Multiplier Reservation Stations

	p	tag/data	p	tag/data
T3				
T4				

Multiplier

Stage	Destination Tag
1	
2	
3	

Cycle 3

Instruction:

Register File

	p	tag/data
f0	1	V0
f1		
f2	1	V1
f3	1	V2

Adder Reservation Stations

	p	tag/data	p	tag/data
T0				
T1				
T2				

Adder

Stage	Destination Tag
1	
2	

Multiplier Reservation Stations

	p	tag/data	p	tag/data
T3				
T4				

Multiplier

Stage	Destination Tag
1	
2	
3	

Cycle 4

Instruction:

Register File

	p	tag/data
f0	1	V0
f1		
f2	1	V1
f3	1	V2

Adder Reservation Stations

	p	tag/data	p	tag/data
T0				
T1				
T2				

Adder

Stage	Destination Tag
1	
2	

Multiplier Reservation Stations

	p	tag/data	p	tag/data
T3				
T4				

Multiplier

Stage	Destination Tag
1	
2	
3	

Cycle 5

Instruction:

Register File

	p	tag/data
f0	1	V0
f1		
f2	1	V1
f3	1	V2

Adder Reservation Stations

	p	tag/data	p	tag/data
T0				
T1				
T2				

Adder

Stage	Destination Tag
1	
2	

Multiplier Reservation Stations

	p	tag/data	p	tag/data
T3				
T4				

Multiplier

Stage	Destination Tag
1	
2	
3	

Cycle 6

Instruction:

Register File

	p	tag/data
f0	1	V0
f1		
f2	1	V1
f3	1	V2

Adder Reservation Stations

	p	tag/data	p	tag/data
T0				
T1				
T2				

Adder

Stage	Destination Tag
1	
2	

Multiplier Reservation Stations

	p	tag/data	p	tag/data
T3				
T4				

Multiplier

Stage	Destination Tag
1	
2	
3	

Cycle 7

Instruction:

Register File

	p	tag/data
f0	1	V0
f1		
f2	1	V1
f3	1	V2

Adder Reservation Stations

	p	tag/data	p	tag/data
T0				
T1				
T2				

Adder

Stage	Destination Tag
1	
2	

Multiplier Reservation Stations

	p	tag/data	p	tag/data
T3				
T4				

Multiplier

Stage	Destination Tag
1	
2	
3	

Cycle 8

Instruction:

Register File

	p	tag/data
f0	1	V0
f1		
f2	1	V1
f3	1	V2

Adder Reservation Stations

	p	tag/data	p	tag/data
T0				
T1				
T2				

Adder

Stage	Destination Tag
1	
2	

Multiplier Reservation Stations

	p	tag/data	p	tag/data
T3				
T4				

Multiplier

Stage	Destination Tag
1	
2	
3	

3.2 Why can't the reservation station entry for an instruction be deallocated immediately on issue?

3.3 Why are exceptions imprecise in this implementation?