CS4290/CS6290

Fall 2011

Prof. Hyesoon Kim
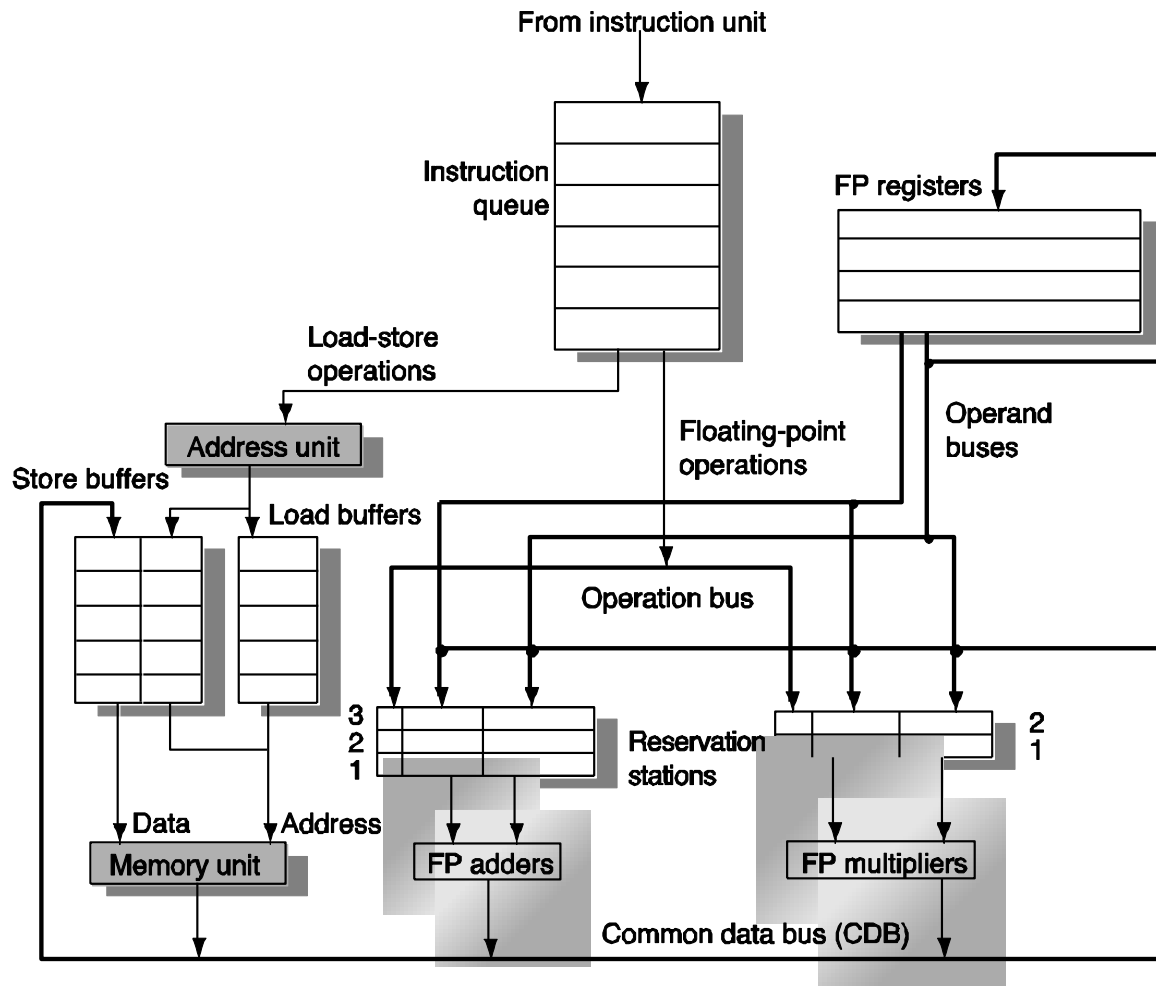
Georgia Tech | College of Computing

Thanks to Prof. Prvulovic and Prof. Loh

Tomasulo's Algorithm: The Picture

From instruction unit

Instruction queue

FP registers

Load-store operations

Address unit

Store buffers

Load buffers

Floating-point operations

Operand buses

Operation bus

3
2
1

Reservation stations

2
1

Data   Address

Memory unit

FP adders

FP multipliers

Common data bus (CDB)

Georgia Tech | College of Computing

Timing Example

- Kind of hard to keep track with previous table-based approach
- Simplified version to track timing only

> Load: 2 cycles
> Add: 2 cycles
> Mult: 10 cycles
> Divide: 40 cycles

Inst	Operands	Is	Exec	Wr	Comments
L.D	F6,34(R2)				
L.D	F2, 45(R3)				
MUL.D	F0,F2,F4				
SUB.D	F8,F2,F6				
DIV.D	F10,F0,F6				
ADD.D	F6,F8,F2				

Georgia Tech | College of Computing

Timing Example

- Kind of hard to keep track with previous table-based approach
- Simplified version to track timing only

Load: 2 cycles
Add: 2 cycles
Mult: 10 cycles
Divide: 40 cycles

Inst	Operands	Is	Exec	Wr	Comments
L.D	F6,34(R2)	1	2	4	
L.D	F2, 45(R3)	2	3	5	
MUL.D	F0,F2,F4	3	6	16	
SUB.D	F8,F2,F6	4	6	8	
DIV.D	F10,F0,F6	5	17	57	
ADD.D	F6,F8,F2	6	9	11	

Timing Example (2)

- This time, assume that there is a data forwarding from functional units to Exec

Load: 2 cycles
Add: 2 cycles
Mult: 10 cycles
Divide: 40 cycles

Inst	Operands	Is	Exec	Wr	Comments
L.D	F6,34(R2)				
L.D	F2, 45(R3)				
MUL.D	F0,F2,F4				
SUB.D	F8,F2,F6				
DIV.D	F10,F0,F6				
ADD.D	F6,F8,F2				

Timing Example (2)

- This time, assume that there is a data forwarding from functional units to Exec

Load: 2 cycles
Add: 2 cycles
Mult: 10 cycles
Divide: 40 cycles

Inst	Operands	Is	Exec	Wr	Comments
L.D	F6,34(R2)	1	2	4	
L.D	F2, 45(R3)	2	3	5	
MUL.D	F0,F2,F4	3	5	15	
SUB.D	F8,F2,F6	4	5	7	
DIV.D	F10,F0,F6	5	15	55	
ADD.D	F6,F8,F2	6	7	9	

Conflict Example

- No forwarding.

Load: 2 cycles
Add: **1** cycles
Mult: **4** cycles
Divide: 40 cycles

Inst	Operands	Is	Exec	Wr	Comments
L.D	F6,34(R2)				
ADD.D	F2,F0,F10				
SUB.D	F0,F6,F4				
MUL.D	F8,F2,F6				
LD.D	F10,28(R4)				
ADD.D	F6,F4,F12				

Conflict Example

- This time, assume you *can* bypass and execute in the same cycle (No forwarding)

Load: 2 cycles
Add: **1** cycles
Mult: **4** cycles
Divide: 40 cycles

Inst	Operands	Is	Exec	Wr	Comments
L.D	F6,34(R2)	1	2	4	
ADD.D	F2,F0,F10	2	3	5	
SUB.D	F0,F6,F4	3	5	6	
MUL.D	F8,F2,F6	4	6	10	
LD.D	F10,28(R4)	5	6	8	
ADD.D	F6,F4,F12	6	7	9	

Tomasulo Drawbacks

- Many associative stores (CDB) at high speed

- Performance limited by Common Data Bus
  - Multiple CDBs => more FU logic for parallel assoc stores

Out-of-Order Execution

- We're now executing instructions in data-flow order
  - Great! More performance

- But outside world can't know about this
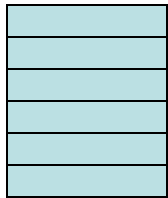 - Must maintain illusion of sequentiality

Re-Order Buffer (ROB)

- Separates architected vs. physical registers

- Tracks program order of all in-flight insts
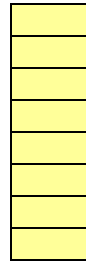 - Enables in-order completion or "commit"

Hardware Organization

Instruction Buffers

RAT

Architected Register File

Reservation Stations and ALUs

op	Qj	Qk	Vj	Vk
op	Qj	Qk	Vj	Vk
op	Qj	Qk	Vj	Vk
op	Qj	Qk	Vj	Vk

Add

op	Qj	Qk	Vj	Vk
op	Qj	Qk	Vj	Vk

Mult

ROB

← "head"

type	dest	value	fin
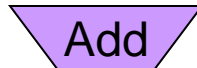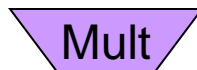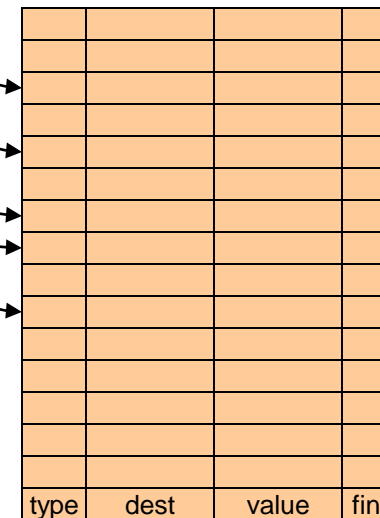
Issue

- Read inst from inst buffer
- Check if resources available:
  - Appropriate RS entry
  - ROB entry
- Read RAT, read (available) sources, update RAT
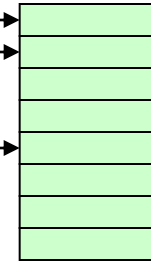- Write to RS and ROB

Instruction Buffers

RAT

Architected Register File

Reservation Stations and ALUs

op	Qj	Qk	Vj	Vk
op	Qj	Qk	Vj	Vk
op	Qj	Qk	Vj	Vk
op	Qj	Qk	Vj	Vk

Add

op	Qj	Qk	Vj	Vk
op	Qj	Qk	Vj	Vk

Mult

ROB

← "head"

type	dest	value	fin
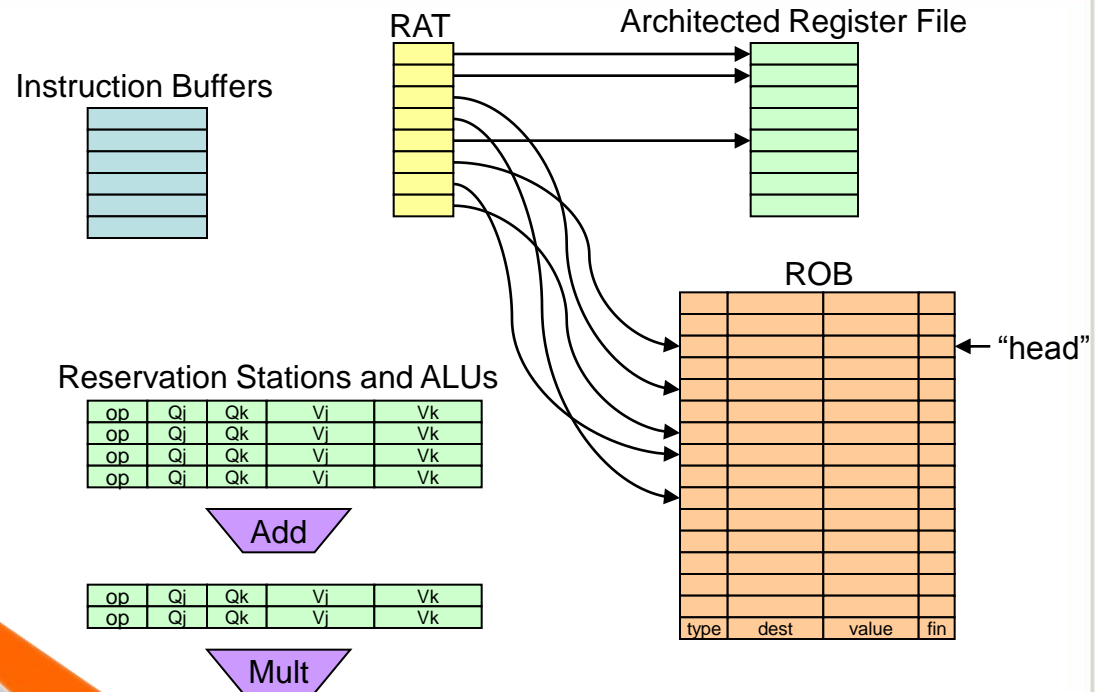
Stall issue if any needed resource not available

Exec

- Same as before
  - Wait for all operands to arrive
  - Compete to use functional unit
  - Execute!

Write Result

- Broadcast result on CDB
  - (any dependents will grab the value)
- Write result back to your **ROB** entry
  - The ARF holds the "official" register state, which we will only update in program order
  - Mark ready/finished bit in ROB (note that this inst has completed execution)
- Reservation station can be freed.

New: Commit

- When an inst is the oldest in the ROB
  - i.e., ROB-head points to it
- Write result (if ready/finished bit is set)
  - If register producing instruction: write to architected register file
  - If store: write to memory
    - Q: What about load?
- Advance ROB-head to next instruction

- This is what the outside world sees
  - And it's all in-order

Georgia Tech | College of Computing

Commit Illustrated

- Make instruction execution "visible" to the outside world
  - "Commit" the changes to the architected state

ROB

A	✓
B	✓
C	✓
D	✓
E	✓
F	✓
G	✓
H	✓
J	✓
K	✓

WB result

ARF
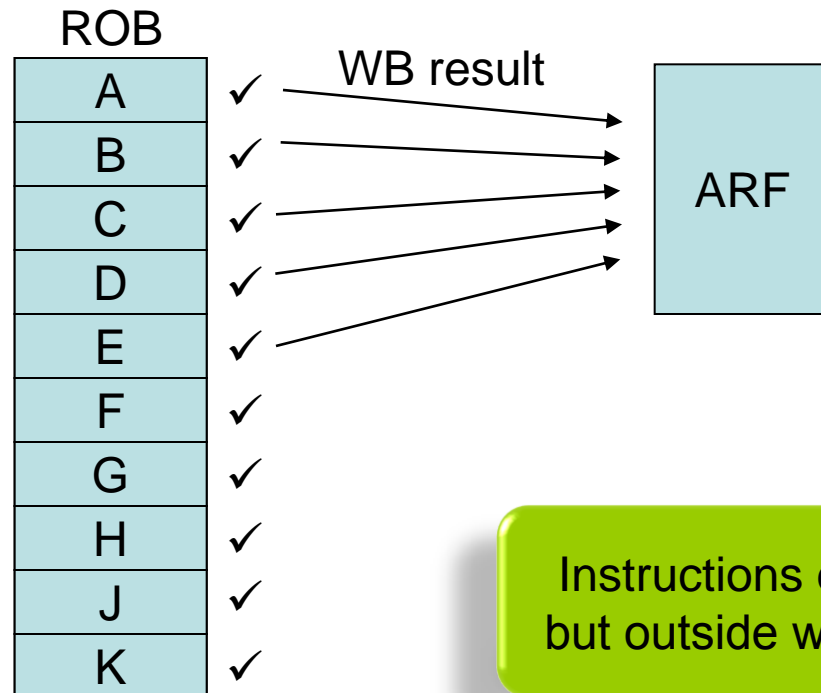
Outside World "sees":

A executed
B executed
C executed
D executed
E executed

Instructions execute out of program order, but outside world still "believes" it's in-order

Revisiting Register Renaming

ROB

ROB1	R1	R2+R3
ROB2	R3	R5-R6
ROB3	R1	ROB1*R7
ROB4	R1	R4+R8
ROB5	R2	R9+R3
ROB6		
ROB7		
ROB8		

RAT

R1	ROB4
R2	R2
R3	ROB2
	⋮

~~R1 = R2 + R3~~

~~R3 = R5 − R6~~

R1 = R1 * R7

R1 = R4 + R8

R2 = R9 + R3

If we issue R2=R9+R3 to the ROB now, R3 comes from ROB2

However, if R3=R5-R6 commits first…

Then update RAT so when we issue R2=R9+R3, it will read source from the ARF.

Georgia Tech | College of Computing

Example

Inst	Operands
DIV	R2, R3, R4
MUL	R1, R5, R6
ADD	R3, R7, R8
MUL	R1, R1, R3
SUB	R4, R1, R5
ADD	R1, R4, R2

Add: 1 cycles
Mult: 10 cycles
Divide: 40 cycles

Sequentially, this would take:

40+10+1+10+1+1 = 63 cycles

(+ other pipeline stages)

R1	-23
R2	16
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

In Detail

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

RS (Mul/Div)

ARF

R1	-23
R2	16
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ARF1
R2	ARF2
R3	ARF3
R4	ARF4
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

ROB1				
ROB2				
ROB3				
ROB4				
ROB5				
ROB6				

	I	E	W	C
1				
2				
3				
4				
5				
6				

Cycle:

Georgia Tech | College of Computing

In Detail

Add: 1 cycles
Mult: 10 cycles
Divide: 40 cycles

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

RS (Mul/Div)

DIV	ROB1			45	5

ARF

R1	-23
R2	16
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ARF1
R2	ARF2 ROB1
R3	ARF3
R4	ARF4
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

ROB1		R2		
ROB2				
ROB3				
ROB4				
ROB5				
ROB6				

	I	E	W	C
1	1			
2				
3				
4				
5				
6				

Cycle: 1

Georgia Tech College of Computing

In Detail

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

RS (Mul/Div)

DIV	ROB1			45	5
MUL	ROB2			3	4

ARF

R1	-23
R2	16
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ARF1 ROB2
R2	ROB1
R3	ARF3
R4	ARF4
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

ROB1	R2		
ROB2	R1		
ROB3			
ROB4			
ROB5			
ROB6			

	I	E	W	C
1	1	2		
2	2			
3				
4				
5				
6				

Cycle: 2

Georgia Tech | College of Computing

In Detail

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

ADD	ROB3			1	2

RS (Mul/Div)

DIV	ROB1			45	5
MUL	ROB2			3	4

ARF

R1	-23
R2	16
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ROB2
R2	ROB1
R3	ARF3 ROB3
R4	ARF4
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

ROB1	R2		
ROB2	R1		
ROB3	R3		
ROB4			
ROB5			
ROB6			

	I	E	W	C
1	1	2		
2	2	3		
3	3			
4				
5				
6				

Cycle: 3

Georgia Tech College of Computing

In Detail

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

ADD	ROB3			1	2

RS (Mul/Div)

DIV	ROB1			45	5
MUL	ROB2			3	4

ARF

R1	-23
R2	16
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ROB2
R2	ROB1
R3	ROB3
R4	ARF4
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

ROB1	R2		
ROB2	R1		
ROB3	R3		
ROB4			
ROB5			
ROB6			

	I	E	W	C
1	1	2		
2	2	3		
3	3	4		
4				
5				
6				

Cycle: 4

In Detail

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

ADD	ROB3			1	2

RS (Mul/Div)

DIV	ROB1			45	5
MUL	ROB2			3	4

ARF

R1	-23
R2	16
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ROB2
R2	ROB1
R3	ROB3
R4	ARF4
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

	Type	Dest	Value	Finished
ROB1		R2		
ROB2		R1		
ROB3		R3	3	Y
ROB4				
ROB5				
ROB6				

	I	E	W	C
1	1	2		
2	2	3		
3	3	4	5	
4				
5				
6				

Cycle: 5

In Detail

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

RS (Mul/Div)

DIV	ROB1			45	5
MUL	ROB2			3	4

ARF

R1	-23
R2	16
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ROB2
R2	ROB1
R3	ROB3
R4	ARF4
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

ROB1	R2		
ROB2	R1	12	Y
ROB3	R3	3	Y
ROB4			
ROB5			
ROB6			

	I	E	W	C
1	1	2		
2	2	3	13	
3	3	4	5	
4				
5				
6				

Cycle: 13

In Detail

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

RS (Mul/Div)

DIV	ROB1			45	5
MUL	ROB4			12	3

ARF

R1	-23
R2	16
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ROB2 ROB4
R2	ROB1
R3	ROB3
R4	ARF4
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

	Type	Value	Finished
ROB1	R2		
ROB2	R1	12	Y
ROB3	R3	3	Y
ROB4	R1		
ROB5			
ROB6			

	I	E	W	C
1	1	2		
2	2	3	13	
3	3	4	5	
4	14			
5				
6				

Cycle: 14

In Detail

Add: 1 cycles
Mult: 10 cycles
Divide: 40 cycles

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

SUB	ROB5	ROB4			3

RS (Mul/Div)

DIV	ROB1			45	5
MUL	ROB4			12	3

ARF

R1	-23
R2	16
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ROB4
R2	ROB1
R3	ROB3
R4	ARF4 ROB5
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

	Type	Value	Finished
ROB1	R2		
ROB2	R1	12	Y
ROB3	R3	3	Y
ROB4	R1		
ROB5	R4		
ROB6			

	I	E	W	C
1	1	2		
2	2	3	13	
3	3	4	5	
4	14	15		
5	15			
6				

Cycle: 15

In Detail

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

SUB	ROB5	ROB4		3
ADD	ROB6	ROB5	ROB1	

RS (Mul/Div)

DIV	ROB1		45	5
MUL	ROB4		12	3

ARF

R1	-23
R2	16
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	~~ROB4~~ ROB6
R2	ROB1
R3	ROB3
R4	ROB5
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

	Type	Value	Finished
ROB1	R2		
ROB2	R1	12	Y
ROB3	R3	3	Y
ROB4	R1		
ROB5	R4		
ROB6	R1		

	I	E	W	C
1	1	2		
2	2	3	13	
3	3	4	5	
4	14	15		
5	15			
6	16			

Cycle: 16

In Detail

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

SUB	ROB5	ROB4		3
ADD	ROB6	ROB5	ROB1	

RS (Mul/Div)

DIV	ROB1		45	5
MUL	ROB4		12	3

ARF

R1	-23
R2	16
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ROB6
R2	ROB1
R3	ROB3
R4	ROB5
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

ROB1	R2		
ROB2	R1	12	Y
ROB3	R3	3	Y
ROB4	R1		
ROB5	R4		
ROB6	R1		

	I	E	W	C
1	1	2		
2	2	3	13	
3	3	4	5	
4	14	15		
5	15			
6	16			

Cycle:

In Detail

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

SUB	ROB5			36	3
ADD	ROB6	ROB5	ROB1		

RS (Mul/Div)

DIV	ROB1			45	5

ARF

R1	-23
R2	16
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ROB6
R2	ROB1
R3	ROB3
R4	ROB5
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

	Type	Value	Finished
ROB1	R2		
ROB2	R1	12	Y
ROB3	R3	3	Y
ROB4	R1	36	Y
ROB5	R4		
ROB6	R1		

	I	E	W	C
1	1	2		
2	2	3	13	
3	3	4	5	
4	14	15	25	
5	15			
6	16			

Cycle: 25

Georgia Tech | College of Computing

In Detail

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

SUB	ROB5			36	3
ADD	ROB6	ROB5	ROB1		

RS (Mul/Div)

DIV	ROB1			45	5

ARF

R1	-23
R2	16
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ROB6
R2	ROB1
R3	ROB3
R4	ROB5
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

ROB1	R2		
ROB2	R1	12	Y
ROB3	R3	3	Y
ROB4	R1	36	Y
ROB5	R4		
ROB6	R1		

	I	E	W	C
1	1	2		
2	2	3	13	
3	3	4	5	
4	14	15	25	
5	15	26		
6	16			

Cycle: 26

In Detail

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

ADD	ROB6		ROB1	33	

RS (Mul/Div)

DIV	ROB1			45	5

ARF

R1	-23
R2	16
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ROB6
R2	ROB1
R3	ROB3
R4	ROB5
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

ROB1	R2		
ROB2	R1	12	Y
ROB3	R3	3	Y
ROB4	R1	36	Y
ROB5	R4	33	Y
ROB6	R1		

	I	E	W	C
1	1	2		
2	2	3	13	
3	3	4	5	
4	14	15	25	
5	15	26	27	
6	16			

Cycle: 27

In Detail

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

ADD	ROB6		33	9

RS (Mul/Div)

ARF

R1	-23
R2	16
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ROB6
R2	ROB1
R3	ROB3
R4	ROB5
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

ROB1	R2	9	Y
ROB2	R1	12	Y
ROB3	R3	3	Y
ROB4	R1	36	Y
ROB5	R4	33	Y
ROB6	R1		

	I	E	W	C
1	1	2	42	
2	2	3	13	
3	3	4	5	
4	14	15	25	
5	15	26	27	
6	16			

Cycle: 42

In Detail

#	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

ADD	ROB6		33	9

RS (Mul/Div)

ARF

R1	-23
R2	9
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ROB6
R2	ARF2
R3	ROB3
R4	ROB5
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

	Type	Value	Finished
ROB1			
ROB2	R1	12	Y
ROB3	R3	3	Y
ROB4	R1	36	Y
ROB5	R4	33	Y
ROB6	R1		

	I	E	W	C
1	1	2	42	43
2	2	3	13	
3	3	4	5	
4	14	15	25	
5	15	26	27	
6	16	43		

Cycle: 43

Georgia Tech | College of Computing

In Detail

Add: 1 cycles
Mult: 10 cycles
Divide: 40 cycles

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

ADD	ROB6		33	9

RS (Mul/Div)

ARF

R1	12
R2	9
R3	45
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ROB6
R2	ARF2
R3	ROB3
R4	ROB5
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

	Type	Dest	Value	Finished
ROB1				
ROB2				
ROB3		R3	3	Y
ROB4		R1	36	Y
ROB5		R4	33	Y
ROB6		R1		

	I	E	W	C
1	1	2	42	43
2	2	3	13	44
3	3	4	5	
4	14	15	25	
5	15	26	27	
6	16	43	44	

Cycle: 44

Georgia Tech | College of Computing

In Detail

Add: 1 cycles
Mult: 10 cycles
Divide: 40 cycles

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

RS (Mul/Div)

ARF

R1	12
R2	9
R3	3
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ROB6
R2	ARF2
R3	ARF3
R4	ROB5
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

ROB1			
ROB2			
ROB3			
ROB4	R1	36	Y
ROB5	R4	33	Y
ROB6	R1	42	Y

	I	E	W	C
1	1	2	42	43
2	2	3	13	44
3	3	4	5	45
4	14	15	25	
5	15	26	28	
6	16	43	44	

Cycle: 45

Georgia Tech | College of Computing

In Detail

Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

RS (Mul/Div)

ARF

R1	36
R2	9
R3	3
R4	5
R5	3
R6	4
R7	1
R8	2

RAT

R1	ROB6
R2	ARF2
R3	ARF3
R4	ROB5
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

	Type	Dest	Value	Finished
ROB1				
ROB2				
ROB3				
ROB4				
ROB5		R4	33	Y
ROB6		R1	42	Y

	I	E	W	C
1	1	2	42	43
2	2	3	13	44
3	3	4	5	45
4	14	15	25	46
5	15	26	28	
6	16	43	44	

Cycle: 46

In Detail

	Inst	Operands
1	DIV	R2, R3, R4
2	MUL	R1, R5, R6
3	ADD	R3, R7, R8
4	MUL	R1, R1, R3
5	SUB	R4, R1, R5
6	ADD	R1, R4, R2

Assume you can bypass and execute in the same cycle

RS fields:

Op	Dst-Tag	Tag1	Tag2	Val1	Val2

ROB fields:

Type	Dest	Value	Finished

RS (Adder)

RS (Mul/Div)

ARF

R1	36
R2	9
R3	3
R4	33
R5	3
R6	4
R7	1
R8	2

RAT

R1	ROB6
R2	ARF2
R3	ARF3
R4	ARF4
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

ROB

ROB1			
ROB2			
ROB3			
ROB4			
ROB5			
ROB6	R1	42	Y

	I	E	W	C
1	1	2	42	43
2	2	3	13	44
3	3	4	5	45
4	14	15	25	46
5	15	26	28	47
6	16	43	44	

Cycle: 47

Timing Example

- Assume you can bypass and execute in the same cycle

Add: 1 cycles
Mult: 10 cycles
Divide: 40 cycles

Inst	Operands	Is	Exec	Wr	Commit	Comments
DIV	R2, R3, R4					
MUL	R1, R5, R6					
ADD	R3, R7, R8					
MUL	R1, R1, R3					
SUB	R4, R1, R5					
ADD	R1, R4, R2					

Timing Example

- Assume you can bypass and execute in the same cycle

Add: 1 cycles
Mult: 10 cycles
Divide: 40 cycles

Inst	Operands	Is	Exec	Wr	Commit	Comments
DIV	R2, R3, R4	1	2	42	43	
MUL	R1, R5, R6	2	3	13	44	
ADD	R3, R7, R8	3	4	5	45	
MUL	R1, R1, R3	14	15	25	46	
SUB	R4, R1, R5	15	26	27	47	
ADD	R1, R4, R2	16	43	44	48	

Unified Reservation Stations (1)

- If MULT RS's are full, and we need to issue another MULT, then we have to stall
  - But there may be other RS's (e.g., add) that are available
  - Proper number of RS's per ALU needs to be matched to the program's inst distribution
    - But different programs have different distributions

Unified Reservation Stations (2)

RS (Adder)

RS (Mul/Div)

Add
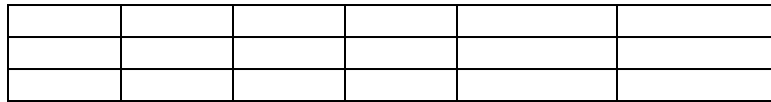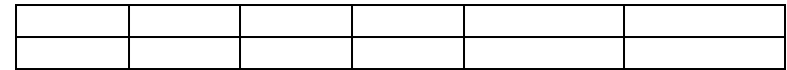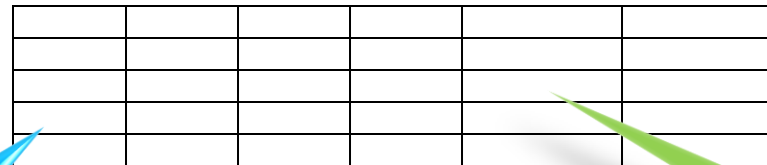
Mult

RS (Unified)

Add

Mult

Can hold 5 adds

Or any combination

Or 5 multiplies

Georgia Tech | College of Computing

Unified Reservation Stations (3)

- Arbitration and execution paths a little more complex (not too bad though)

RS Entries

ADD rdy	
MUL x	
ADD x	
ADD rdy	
MUL rdy	
MUL x	
ADD x	
ADD rdy	

Adder Arbiter

Multiplier Arbiter

For N functional units, RS needs
To support N read ports

Add

Mult

Ready add's compete for execution unit
Ready mul's do the same
Arbitration logic picks insts to execute
Insts go and do their thing

Georgia Tech | College of Computing
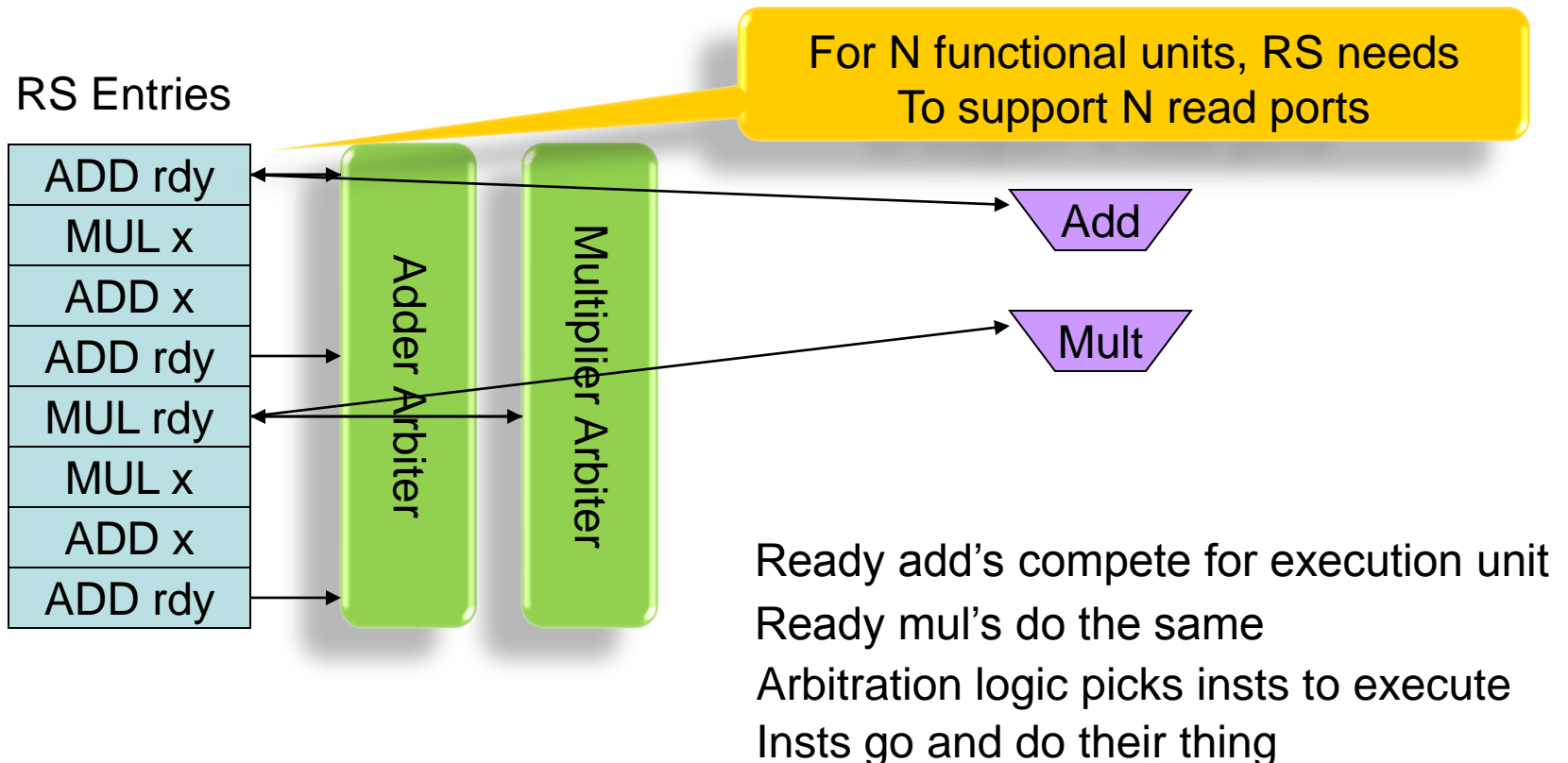
A Brief History of OOO

- OOO is a restricted form of data flow

- CDC 6600 (1964): scoreboard
- IBM 360/91 (1966) Tomasulo's algorithm
- Academic: Yale Patt's HPS (1985)
- Smith & Pleszkun: precise interrupt (1995)
- IBM/Motorola PowerPc 601 (1993)
- Fujitsu/HAP SPARC64(1995)
- Intel Pentium Pro (1995)

From wikipeida

Out-of-Order, but not Superscalar

- As described, this Tomasulo (+ROB) CPU can only maintain *sustained* throughput of 1 IPC

- Limitations:
  - Need superscalar fetch, decode, etc.
  - There's only one CDB, so only one inst per cycle can write-back its result to the ROB
  - Also must commit > 1 IPC

Getting > 1 IPC

- Must be able to issue > 1 IPC to RS/ROB
- Must be able to send > 1 IPC to functional units
  - original Tomasulo can do this already (if inst ready and FU available, go and execute!)
- Must be able to write-back > 1 IPC to ROB (and reservation stations)

Dual-Issue

- Need to check resource availability for two instructions (RS/ROB entries)
  - Depending on resources, may issue 0,1 or 2
- Read RAT/ARF/ROB for operands
  - Renaming is a little trickier (next slide)
- Update RS/ROB entries (not too hard)

Dual-Rename
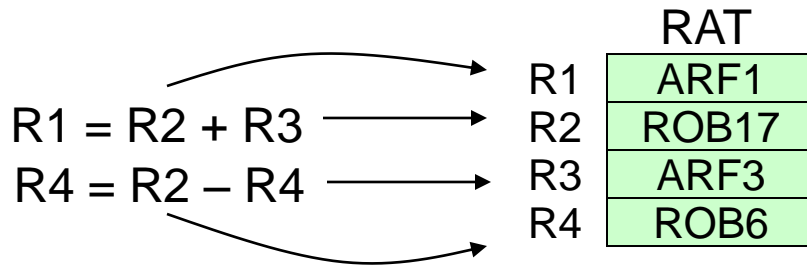
All registers renamed!

ROB21   ROB17, ARF3

ROB22   ROB17, ROB6

RAT

R1	ARF1
R2	ROB17
R3	ARF3
R4	ROB6

R1 = R2 + R3
R4 = R2 – R4

Read current mappings for all operands

New destinations are just in the next two ROB entries
To be allocated: ROB_tail, ROB_tail + 1

RAT

R1	ARF1
R2	ROB17
R3	ARF3
R4	ROB6

R1 = R2 + R3
R4 = R1 – R4
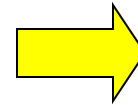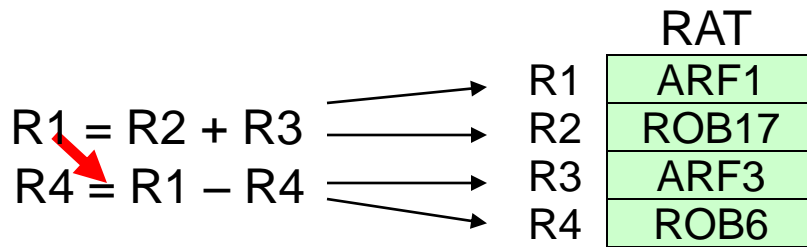
ROB21   ROB17, ARF3

ROB22   ARF1, ROB6

dst1   src2L   From RAT   New ROB entry   src2R   From RAT   New ROB entry

=?   =?
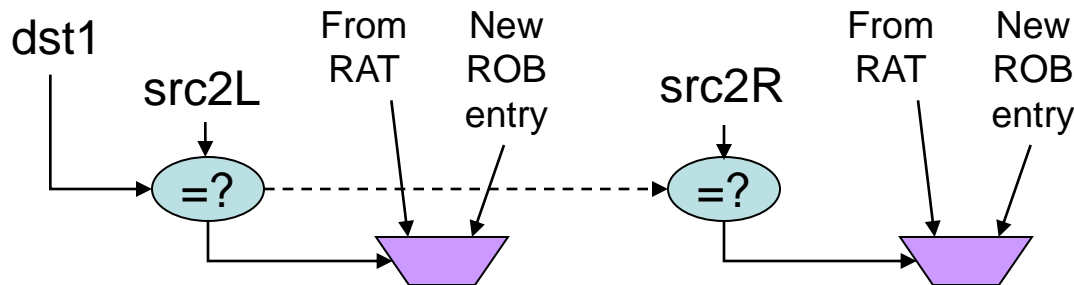
Need to check for RAW dependencies within your issue group

Multiple CDB's

RS Entries

D1, ROB3
E1, ROB4
A2, ROB10
A1, ROB7
B1, ROB19
C1, ROB2

Arbiters

ROB4   ROB7

CDB$_1$

CDB$_2$

FU
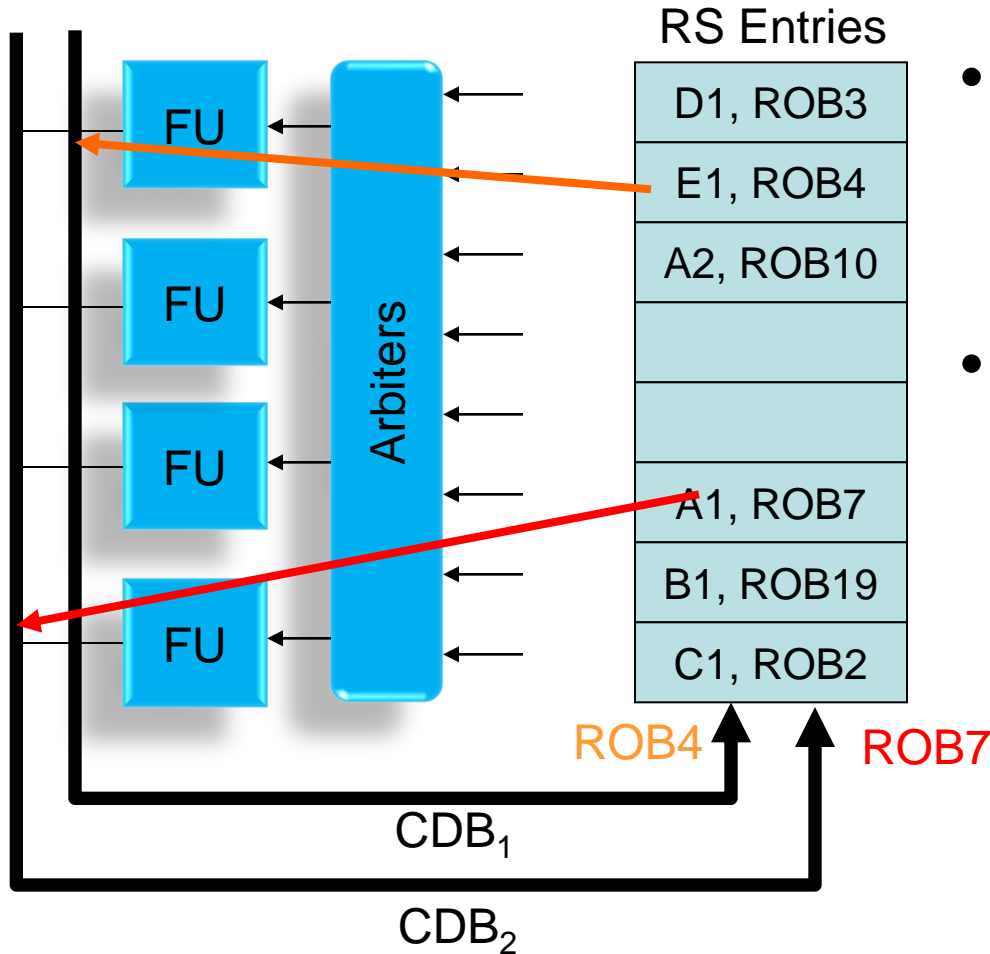
- It works (we do this today in CPUs)

- But there's a cost
  - each RS entry must compare each source with each CDB
  - more area, logic, power (all → $$$)

Georgia Tech | College of Computing

Committing > 1 IPC

- Must be able to write-back multiple results from ROB → ARF (or memory for stores)
  - ROB needs extra read ports
  - ARF needs extra write ports

Terminology is Inconsistent, Overloaded

- Issue, Dispatch, Commit, etc.
    - Text uses terms w.r.t. Tomasulo's algorithm
    - Other usage is different (many academics)

    - Issue/Alloc/Dispatch
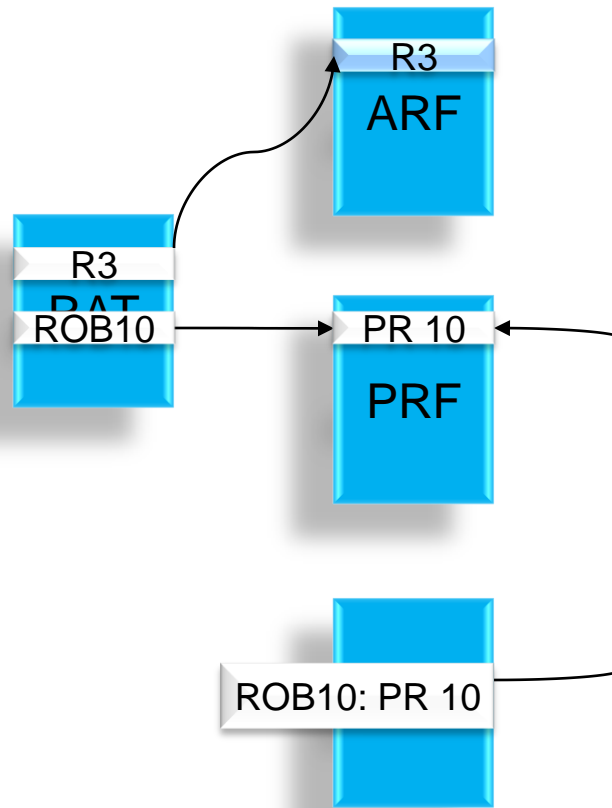   - Exec/Issue/Dispatch
    - Commit/Complete/Retire/Graduate

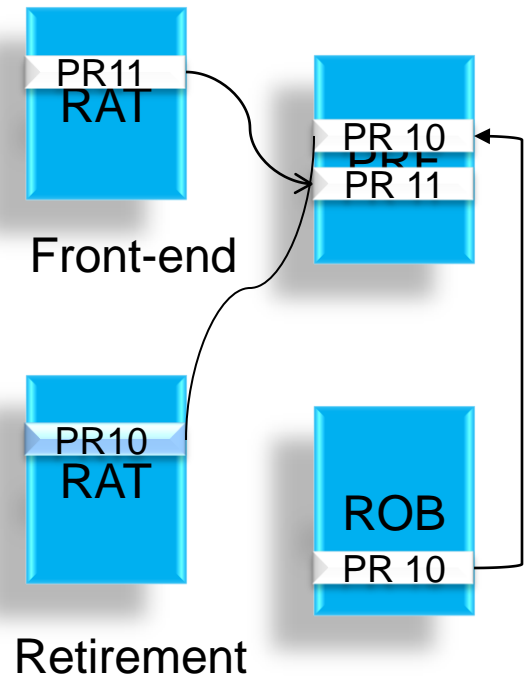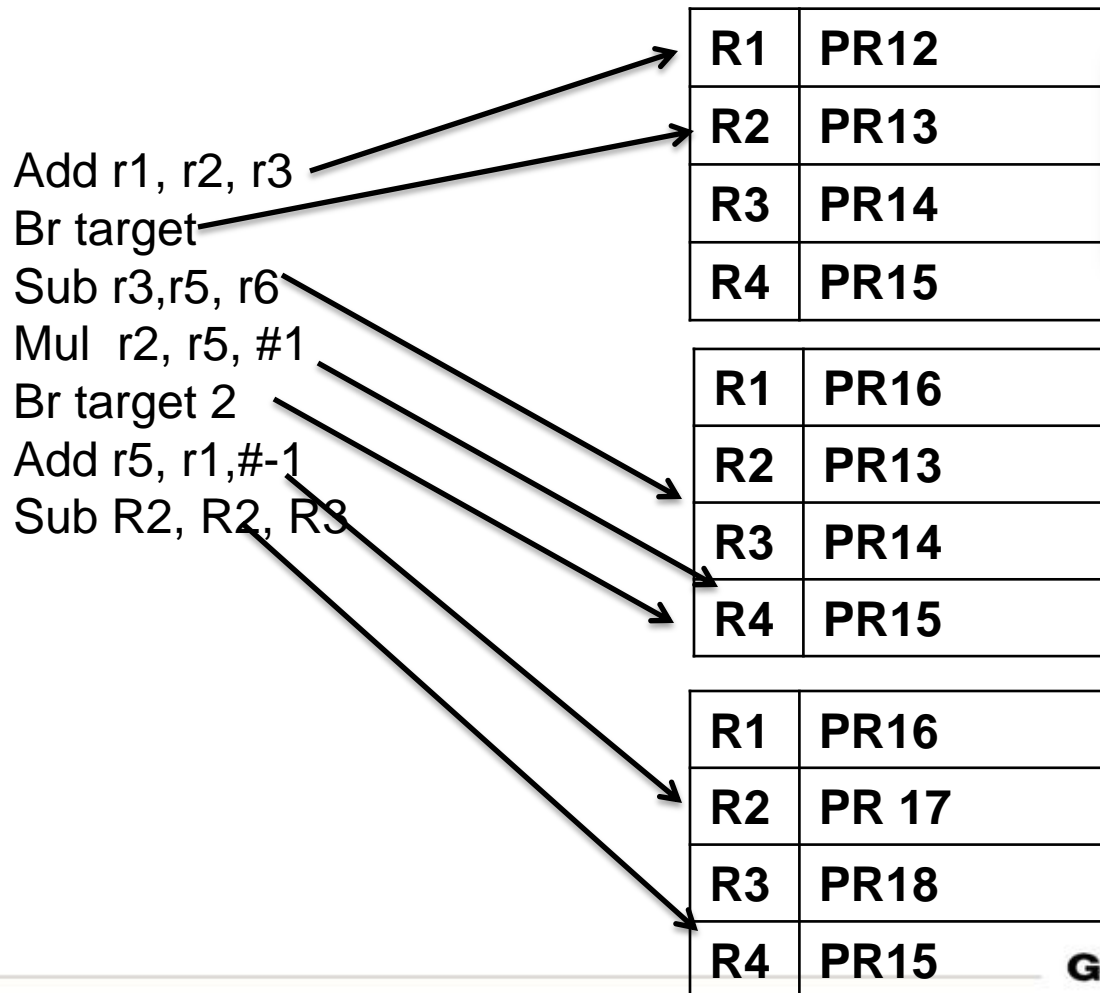   Blue: Intel's convention   Orange: some academics

ROB, PRF, ARF
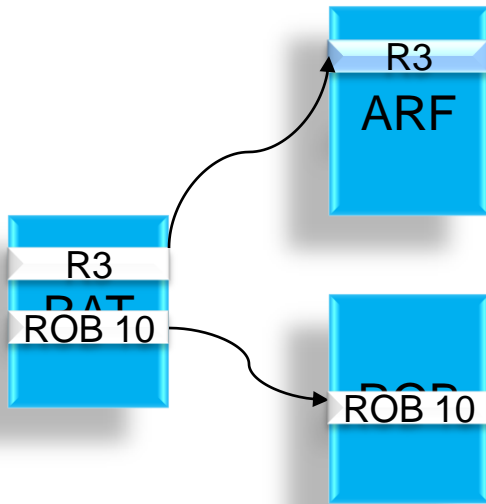
CASE 1

CASE 2
Pentium III

CASE 3
NetBurst

Check Point Mechanism

| fetch | decode | issue | exec | write | commit |

Add r1, r2, r3
Br target
Sub r3,r5, r6
Mul r2, r5, #1
Br target 2
Add r5, r1,#-1
Sub R2, R2, R3

R1	PR12
R2	PR13
R3	PR14
R4	PR15

R1	PR16
R2	PR13
R3	PR14
R4	PR15

R1	PR16
R2	PR 17
R3	PR18
R4	PR15

RAT

ARF

PRF

In order vs. OOO

fetch | decode | issue | exec | write | commit

fetch | decode | issue | exec | write | commit

In-order   Out of order   In-order