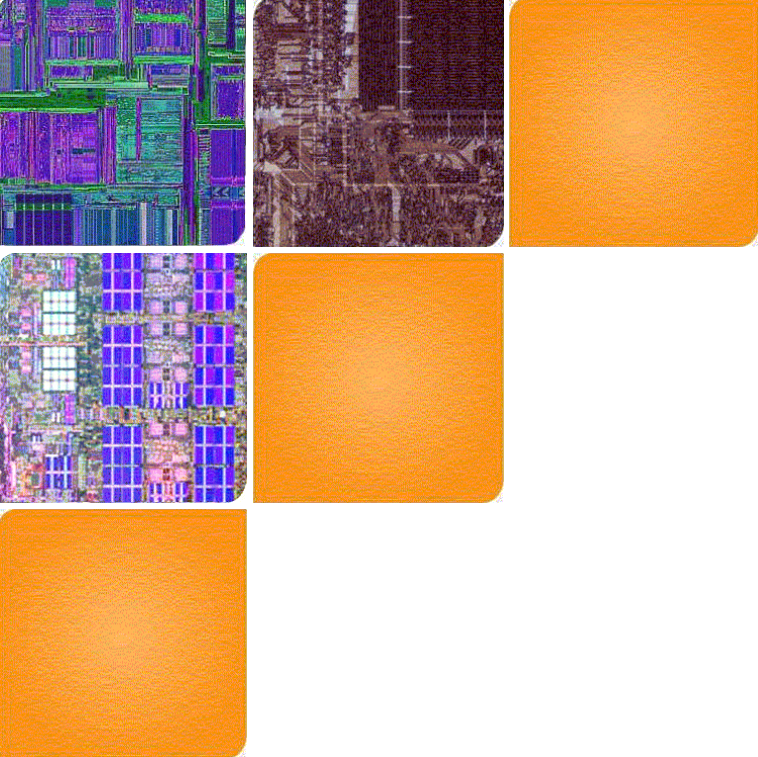


**Homework 1 is out!**  
**Due Tuesday next week!**

**Georgia  
Tech**



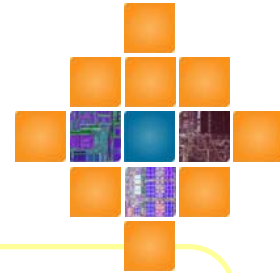
College of  
Computing



# CS6290

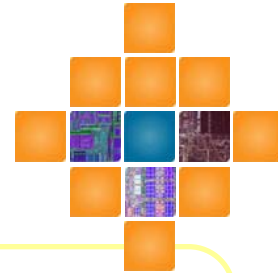
## Reorder Buffer





# 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



# Remember the Toll Booth?



5s

5s

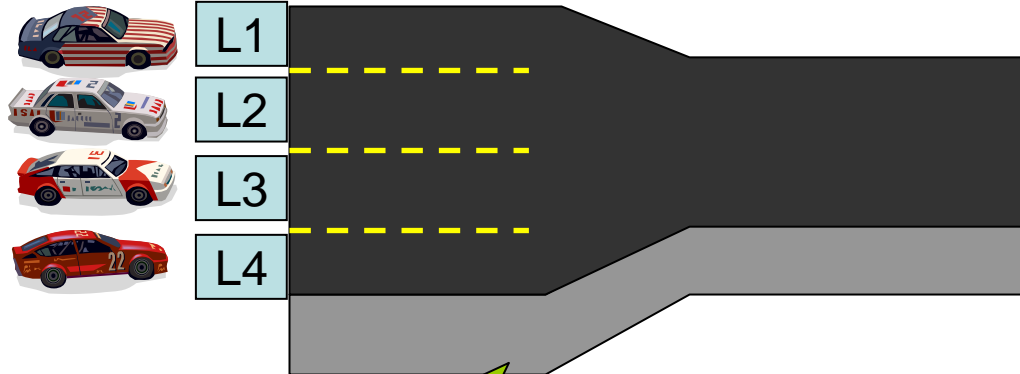
30s

5s

One-at-a-time = 45s

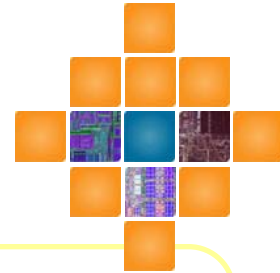
With a "4-Issue" Toll Booth

OOO = 30s



Hands toll-booth agent a \$100 bill; takes a while to count the change

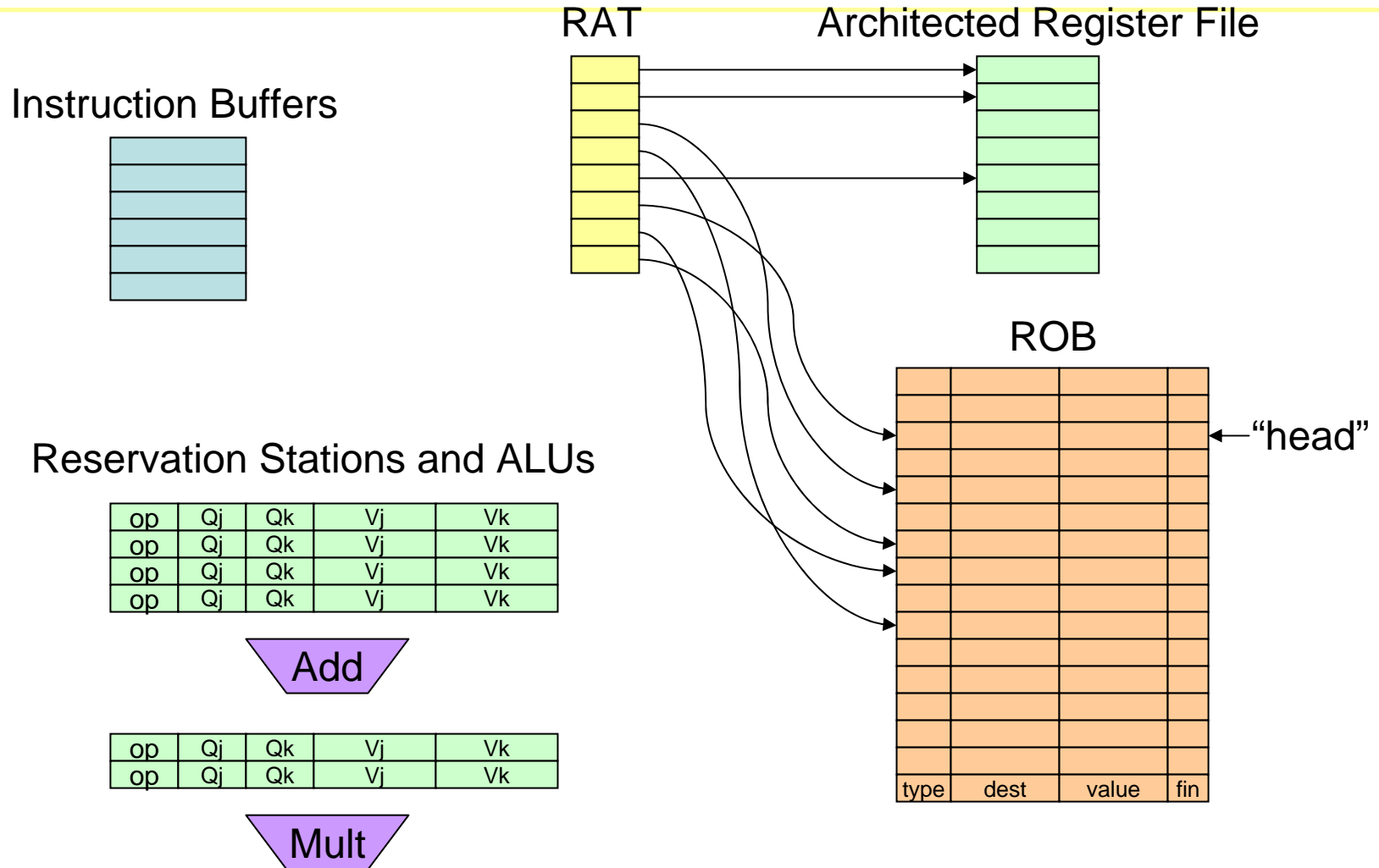
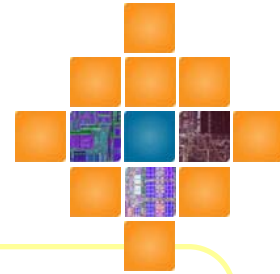
We'll add the equivalent of the "shoulder" to the CPU: the Re-Order Buffer (ROB)

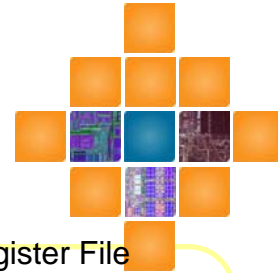


# Re-Order Buffer (ROB)

- Separates architected vs. physical registers
- Tracks program order of all in-flight insts
  - Enables in-order completion or “commit”

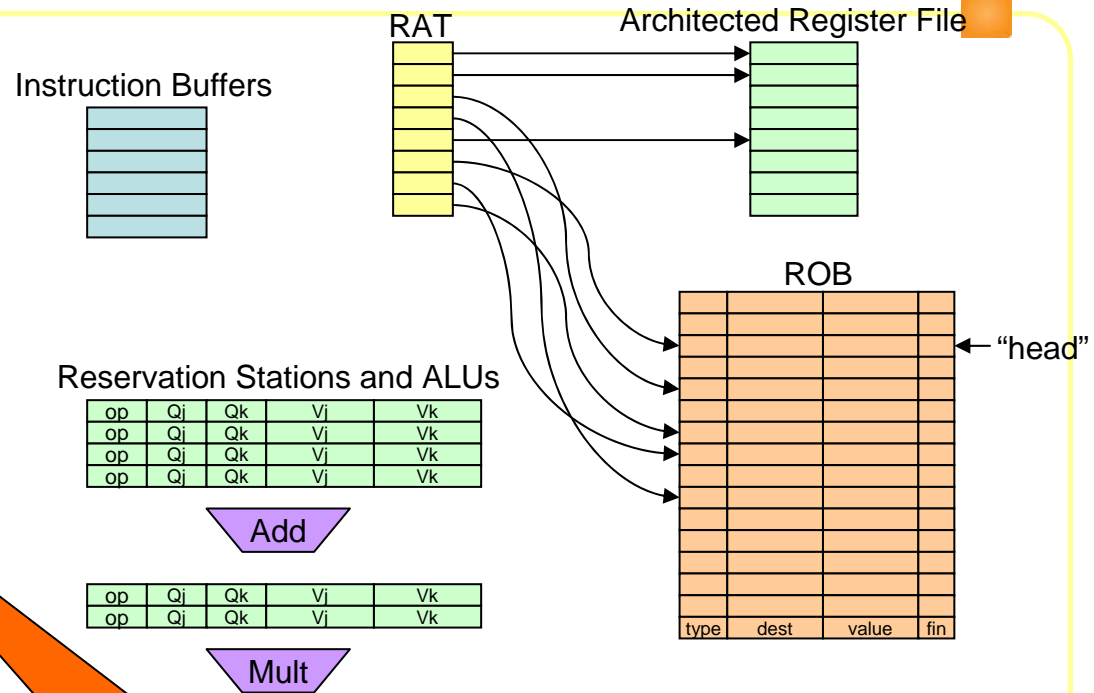
# Hardware Organization





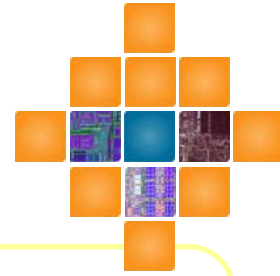
# Issue

- 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

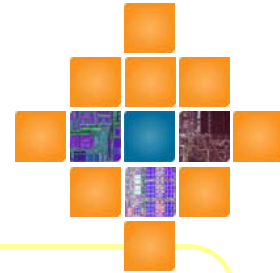


Stall issue if any needed resource not available

# Exec

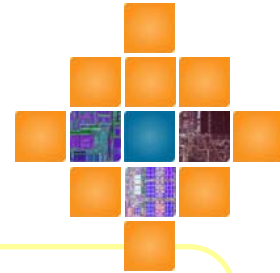


- Same as before
  - Wait for all operands to arrive
  - Compete to use functional unit
  - Execute!



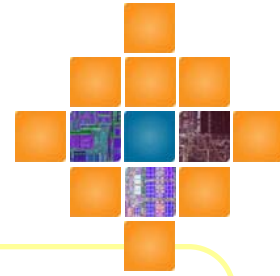
# Write Result

- 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)



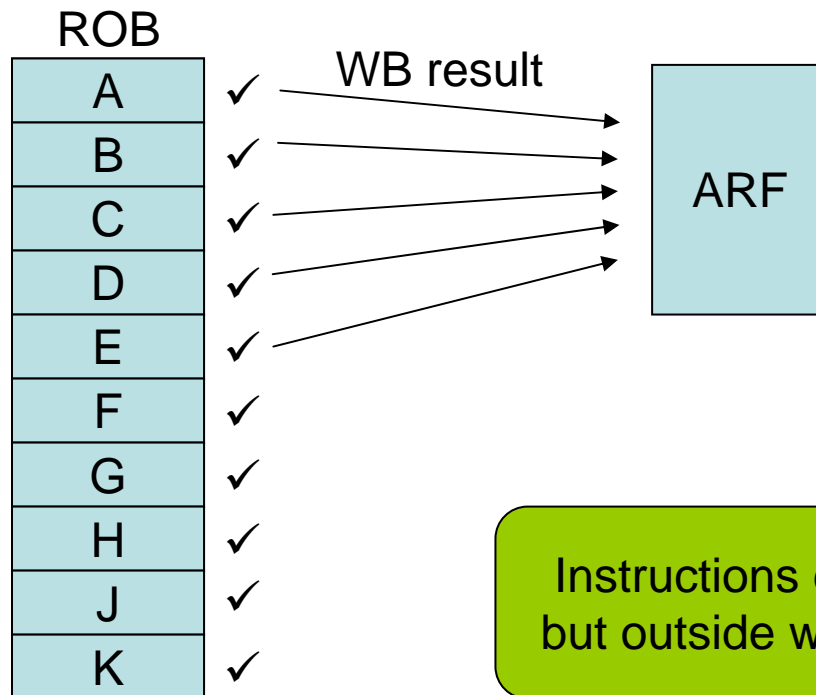
# New: Commit

- 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
- Advance ROB-head to next instruction
  
- This is what the outside world sees
  - And it's all in-order



# Commit Illustrated

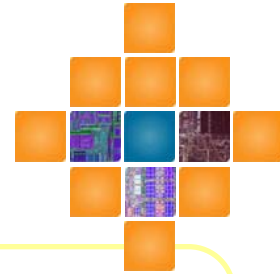
- Make instruction execution “visible” to the outside world
  - “Commit” the changes to the architected state



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
	⋮

An arrow points from the ROB2 row of the first table to the R3 row of the second table.

~~$R1 = R2 + R3$~~

~~$R3 = R5 - R6$~~

$R1 = R1 * R7$

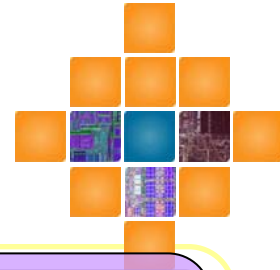
$R1 = R4 + R8$

$R2 = R9 + R3$

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.



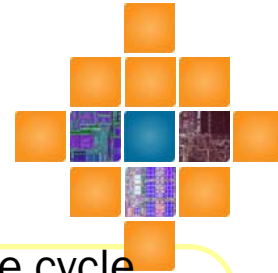
# Example

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

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

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

Sequentially, this would take:  
 $40+10+2+10+2+2 = 66$  cycles  
(+ other pipeline stages)



# In Detail

Assume you can bypass and execute in the same cycle

	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

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	<del>ARF2</del> ROB1
R3	ARF3
R4	ARF4
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

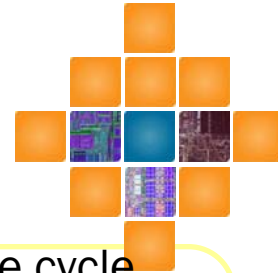
ROB

ROB1		R2		
ROB2				
ROB3				
ROB4				
ROB5				
ROB6				

I E W C

1	1			
2				
3				
4				
5				
6				

Cycle: 1



# 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	<del>ARF1</del> ROB2
R2	ROB1
R3	ARF3
R4	ARF4
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

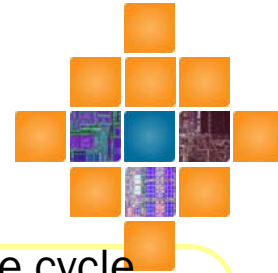
ROB

ROB1		R2		
ROB2		R1		
ROB3				
ROB4				
ROB5				
ROB6				

I E W C

1	1	2		
2	2			
3				
4				
5				
6				

Cycle: 2



# In Detail

Assume you can bypass and execute in the same cycle

	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

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

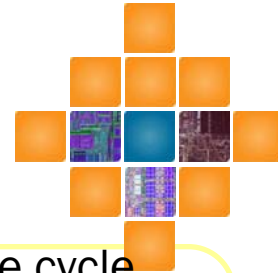
ROB1		R2		
ROB2		R1		
ROB3		R3		
ROB4				
ROB5				
ROB6				

I E W C

1	1	2		
2	2	3		
3	3			
4				
5				
6				

Cycle:

3



# In Detail

Assume you can bypass and execute in the same cycle

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

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

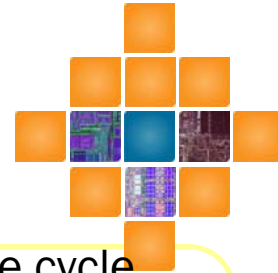
ROB

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

Assume you can bypass and execute in the same cycle

	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

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

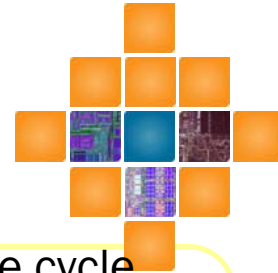
ROB1		R2		
ROB2		R1		
ROB3		R3	3	Y
ROB4				
ROB5				
ROB6				

I E W C

1	1	2		
2	2	3		
3	3	4	6	
4				
5				
6				

Cycle:

6



# In Detail

Assume you can bypass and execute in the same cycle

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

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

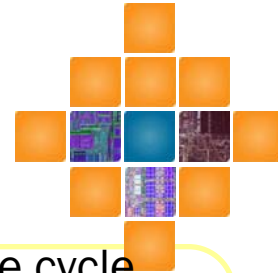
ROB

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	6	
4				
5				
6				

Cycle: 13



# In Detail

Assume you can bypass and execute in the same cycle

	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

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	<del>ROB2</del> ROB4
R2	ROB1
R3	ROB3
R4	ARF4
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

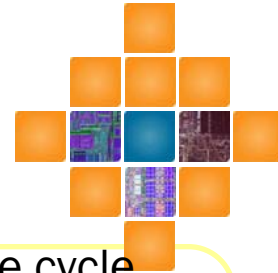
ROB

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

I E W C

1	1	2		
2	2	3	13	
3	3	4	6	
4	14			
5				
6				

Cycle: 14



# In Detail

Assume you can bypass and execute in the same cycle

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

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	<del>ARF4</del> ROB5
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

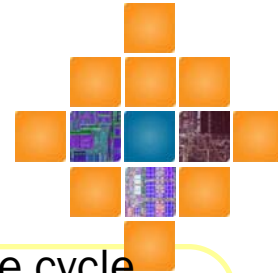
ROB

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

I E W C

1	1	2		
2	2	3	13	
3	3	4	6	
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	<del>ROB4</del> ROB6
R2	ROB1
R3	ROB3
R4	ROB5
R5	ARF5
R6	ARF6
R7	ARF7
R8	ARF8

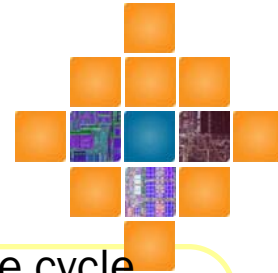
ROB

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	6	
4	14	15		
5	15			
6	16			

Cycle: 16



# In Detail

Assume you can bypass and execute in the same cycle

	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

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

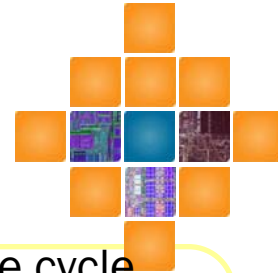
ROB

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	6	
4	14	15		
5	15			
6	16			

Cycle:



# In Detail

Assume you can bypass and execute in the same cycle

	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

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

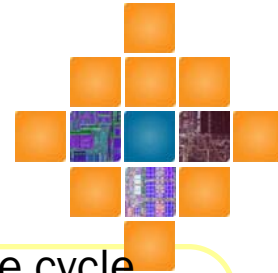
ROB

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	6	
4	14	15	25	
5	15			
6	16			

Cycle: 26



# In Detail

Assume you can bypass and execute in the same cycle

	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

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

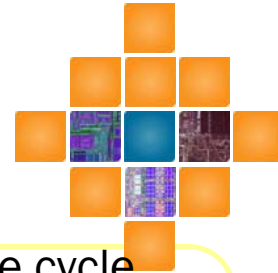
ROB

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	6	
4	14	15	25	
5	15	26		
6	16			

Cycle: 28



# 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

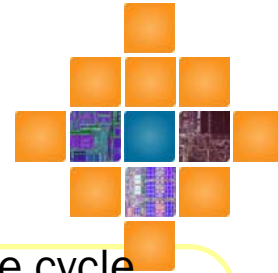
ROB

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	6	
4	14	15	25	
5	15	26	28	
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)

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

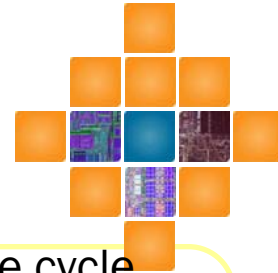
ROB

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	6	
4	14	15	25	
5	15	26	28	
6	16			

Cycle: 43



# In Detail

Assume you can bypass and execute in the same cycle

	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

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

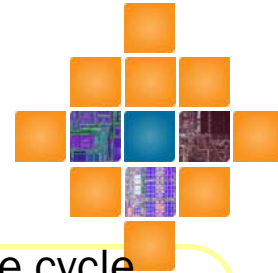
ROB

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	6	
4	14	15	25	
5	15	26	28	
6	16	43		

Cycle: 44



# 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	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

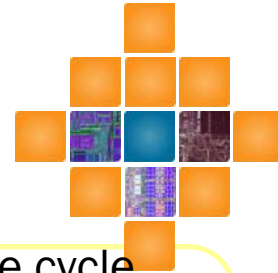
ROB

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	6	
4	14	15	25	
5	15	26	28	
6	16	43		

Cycle: 45



# 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	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

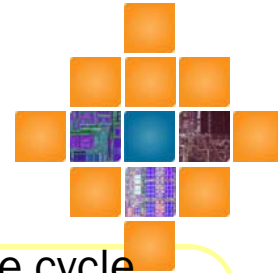
ROB

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	6	45
4	14	15	25	
5	15	26	28	
6	16	43	45	

Cycle: 46



# In Detail

Assume you can bypass and execute in the same cycle

	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

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

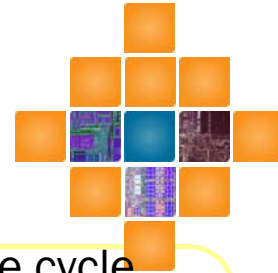
ROB

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	6	45
4	14	15	25	46
5	15	26	28	
6	16	43	45	

Cycle: 47



# In Detail

Assume you can bypass and execute in the same cycle

	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

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

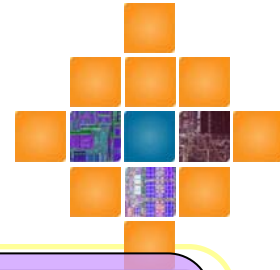
ROB1			
ROB2			
ROB3			
ROB4			
ROB5			
ROB6	R1	42	Y

I E W C

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

Cycle: 

48
----



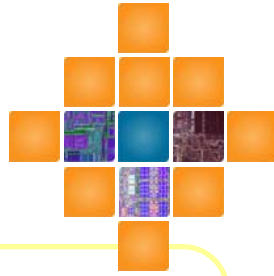
# Timing Example

- Assume you can bypass and execute in the same cycle

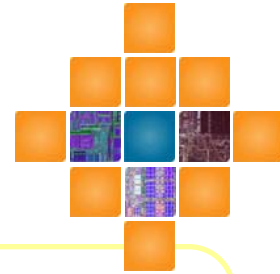
Add: 1 cycles  
Mult: 10 cycles  
Divide: 40 cycles

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					

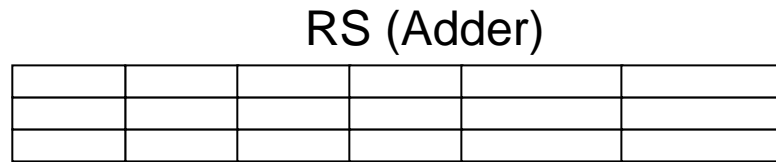
# Unified Reservation Stations (1)



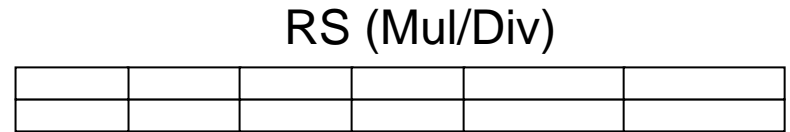
- If MULT RS's are full, and we need to issue another MULT, then we have to stall
  - But there may be other RS's (e.g., add) that are available
  - Proper number of RS's per ALU needs to be matched to the program's inst distribution
    - But different programs have different distributions



# Unified Reservation Stations (2)

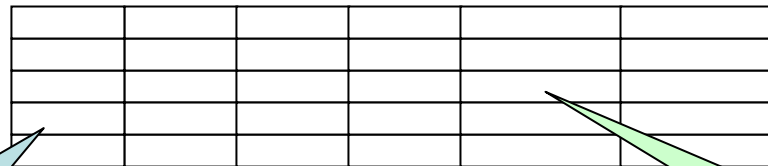


Add



Mult

RS (Unified)



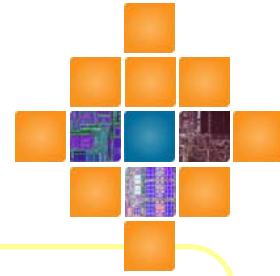
Can hold 5 adds

Add

Mult

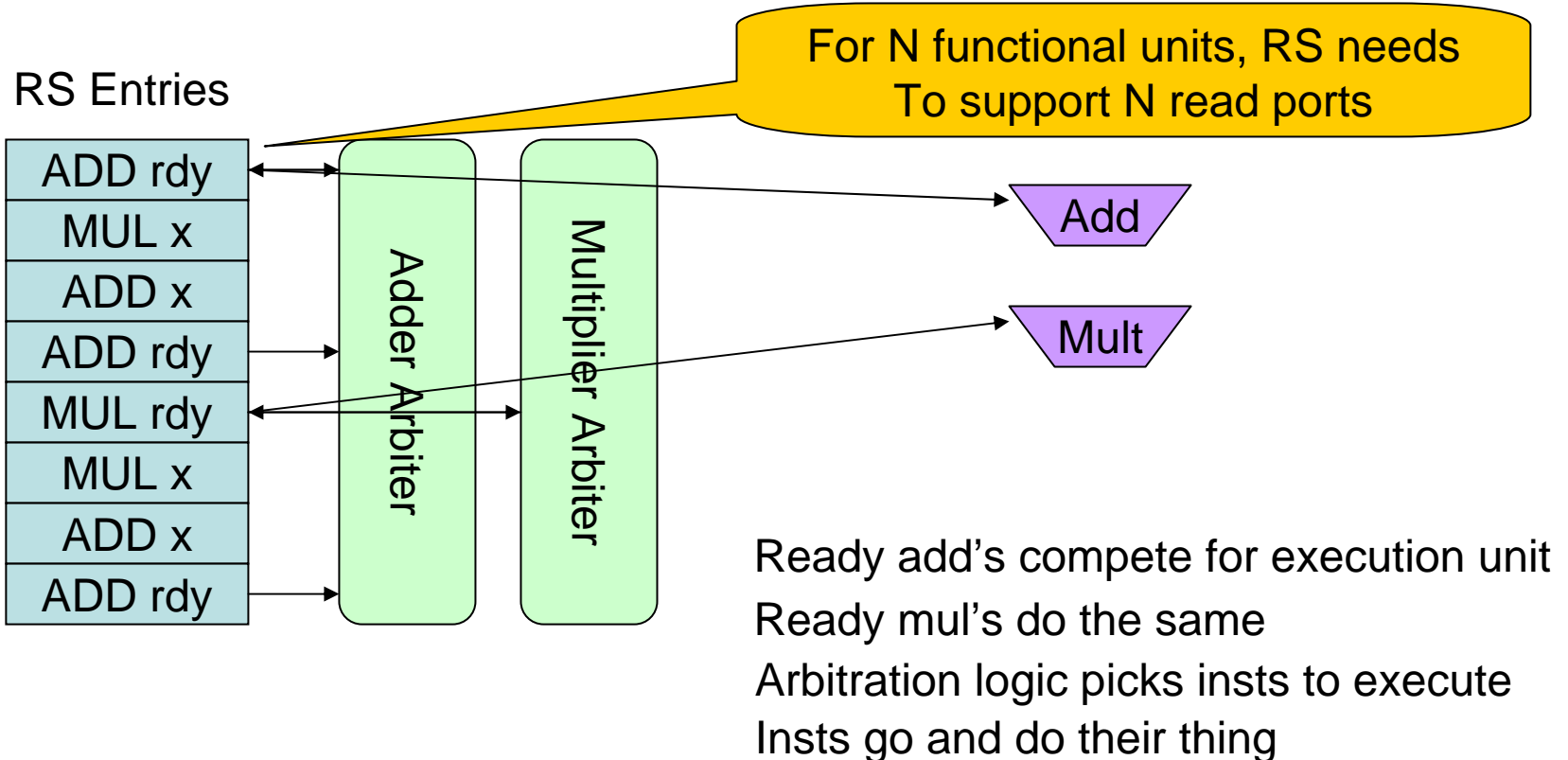
Or 5 multiplies

Or any combination

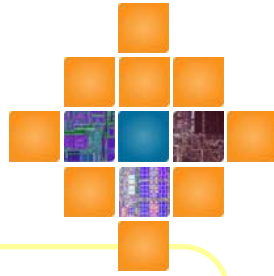


# Unified Reservation Stations (3)

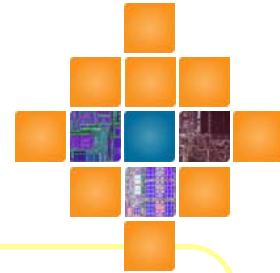
- Arbitration and execution paths a little more complex (not too bad though)



# Out-of-Order, but not Superscalar

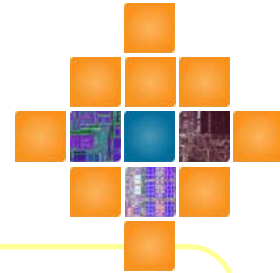


- As described, this Tomasulo (+ROB) CPU can only maintain *sustained* throughput of 1 IPC
- Limitations:
  - Need superscalar fetch, decode, etc.
  - There's only one CDB, so only one inst per cycle can write-back its result to the ROB
  - Also must commit  $> 1$  IPC



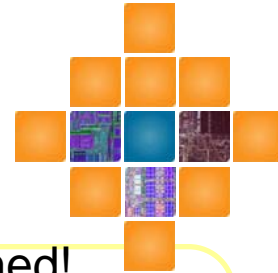
# Getting $> 1$ IPC

- 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)

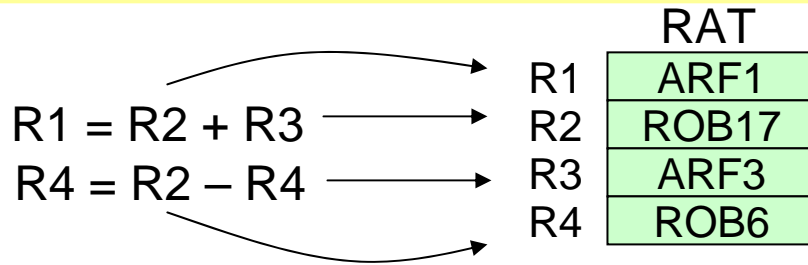


# Dual-Issue

- 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!

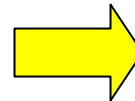
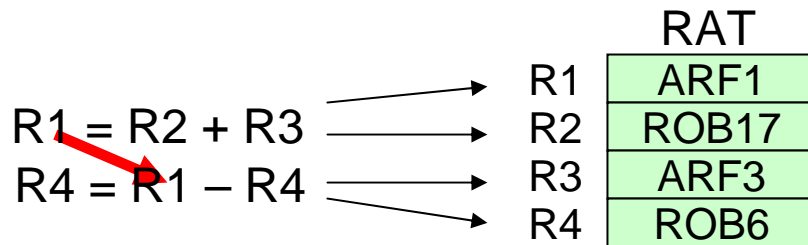
ROB21    ROB17, ARF3

ROB22    ROB17, ROB6

Read current mappings for all operands

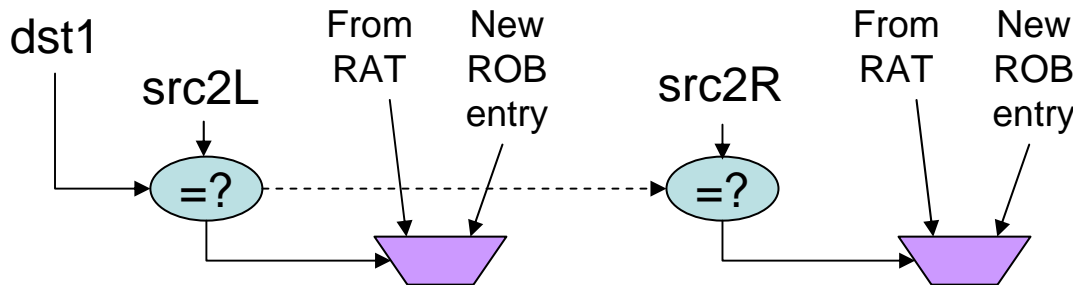
New destinations are just in the next two ROB entries

To be allocated: ROB-tail, ROB-tail + 1

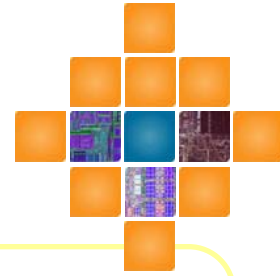


ROB21    ROB17, ARF3

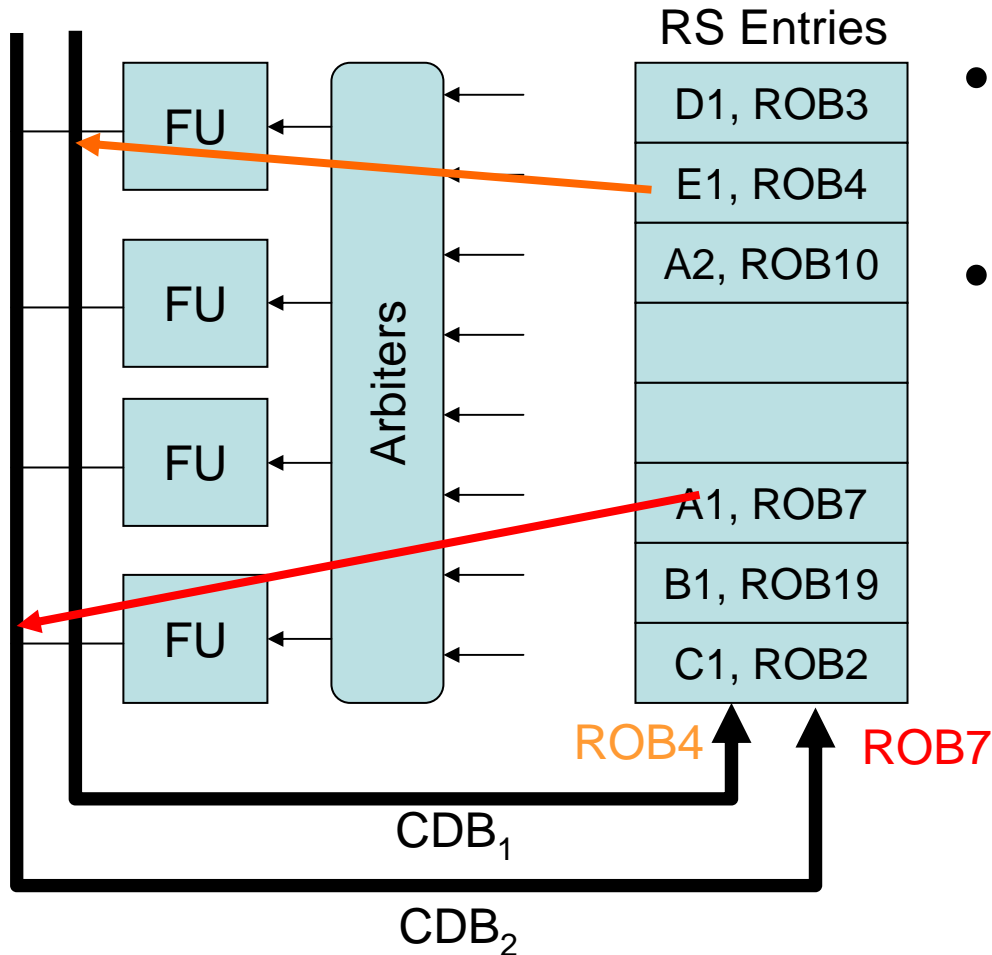
ROB22    ~~ARF1~~, ROB6



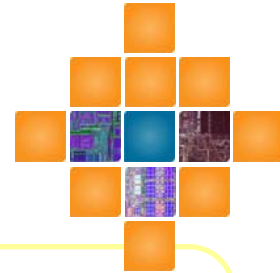
Need to check for RAW dependencies within your issue group



# Multiple CDB's



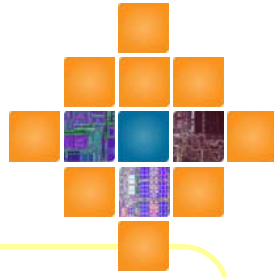
- It works (we do this today in CPUs)
- But there's a cost
  - each RS entry must compare each source with each CDB
  - more area, logic, power (all → \$\$\$)



# 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