Computer Architecture

Spring 2016

Lecture 13: Speculation I

Shuai Wang

Department of Computer Science and Technology

Nanjing University

[Slides adapted from CS 246, Harvard University]

Review: Scoreboading

- **Dynamic scheduling named after CDC6600 scoreboard**
- **Scoreboard monitors instruction issue, operand read, execution, and write result**
  - In-order issue, avoids structural hazards and WAW hazards at issue stage
  - Read operand only when both source operands are ready, avoids RAW hazards
  - Allow out-of-order execution and out-of-order completion
  - Write result stage delays writing register to avoid WAR hazards, if any
- **Performance is limited by**
  - # scoreboard entries, # and types of FUs
  - parallelism within basic block
  - Presence of WAW and WAR dependences

Review: Tomasulo

- **Reservations stations: *implicit register renaming* to larger set of registers + buffering source operands**
  - Prevents registers as bottleneck
  - Avoids WAR, WAW hazards of Scoreboard
  - Allows loop unrolling in HW
- **Not limited to basic blocks (integer units gets ahead, beyond branches)**
- **Today, helps cache misses as well**
  - Do not stall for L1 Data cache miss (insufficient ILP for L2 miss?)
- **Lasting Contributions**
  - Dynamic scheduling
  - Register renaming
  - Load/store disambiguation
- **360/91 descendants are Pentium III; PowerPC 604; MIPS R10000; HP-PA 8000; Alpha 21264**

What about Precise Exception?

- Tomasulo had:

  In-order issue, out-of-order execution, and out-of order completion

- Need to “fix” the out-of-order completion aspect so that we can find precise breakpoint in instruction stream.

Relationship between Precise Interrupts and Speculation

- Speculation is a form of guessing.
- Important for branch prediction:
  - Need to "take our best shot" at predicting branch direction.
- If we speculate and are wrong, need to back up and restart execution to point at which we predicted incorrectly:
  - This is exactly same as precise exceptions!
- Technique for both precise interrupts/exceptions and speculation: *in-order completion or commit*

Solving both problems with one solution

- Need the ability to squash/restart *any* instruction
  - Gives us precise state
    - Need for memory ops (page faults, etc)
    - Need for FP ops (divide by 0)
  - Gives us ability to recover mis-speculations
    - Need for branches

- Providing precise state solves both these problems

How to get precise state?

- Imprecise state
  - As we've said this is a bad idea
  - For speculation it is unacceptable
- Force in-order completion at WB (stall when necessary)
- Precise state in software: save recovery info for traps
  - Traps on all faulting memory, FP, and mispredicted branch ops?
- Precise state in hardware: save recovery info online

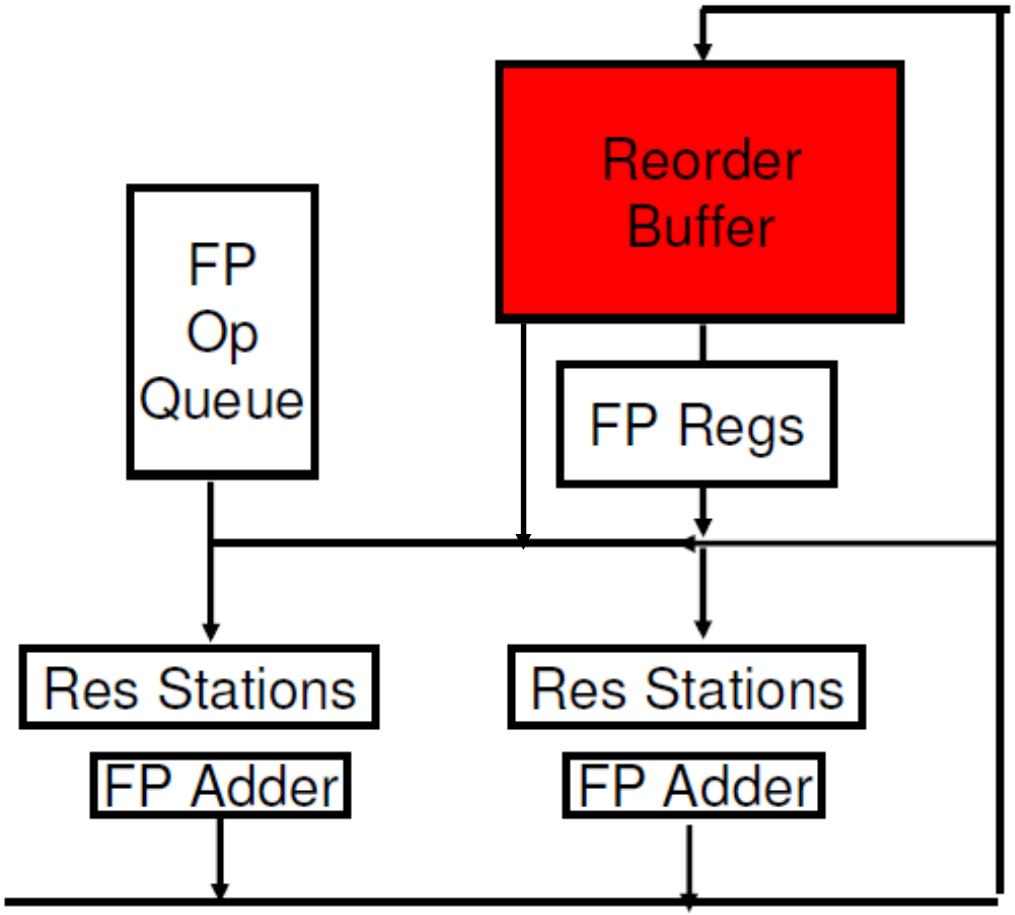
Solution: Writeback and Commit

- Allow out of order issue/writeback
  - Require in-order commit when instruction is no longer speculative
  - Prevent speculative changes from changing state
    - e.g. memory write or register write
- Collect pre-commit instructions
  - in a reorder buffer
    - holds completed but not committed instruction
  - Effectively contains a set of virtual registers
    - similar to a reservation station
    - and becomes a bypass (forwarding) source

Reorder Buffer: HW buffer for results of uncommitted instructions

- Need HW buffer for results of uncommitted instructions: ***Reorder Buffer (ROB)***
  - 3 major fields: instr, destination, value
  - Use reorder buffer number instead of reservation station when execution completes
  - Supplies operands between execution complete & commit
    - Reorder buffer can be operand source => more registers like RS
  - Instructions commit
    - Once instruction commits, result is put into register
  - As a result, easy to undo speculated instructions on mispredicted branches or exceptions

Speculative Tomasulo with ROB



Four Steps of Speculative Tomasulo Algorithm

- **Issue** - get instruction from FP Op Queue
  - If reservation station and reorder buffer slot free, issue instruction & send operands & reorder buffer number for destination (this stage sometimes called “dispatch”.)
  - if RS or ROB is full, instruction issue is stalled.

- **Execution** - operate on operands (EX)
  - When both operands ready then execute;
  - if not ready, watch CDB for result;
  - when both in reservation station, execute;
  - checks RAW (sometimes called “issue”.)

Four Steps of Speculative Tomasulo Algorithm

- **Write result** - finish execution (WB)
  - Write on Common Data Bus to all awaiting reservation stations & reorder buffer;
  - mark reservation station available.

- **Commit** - update register with reorder result
  - When instr at head of reorder buffer & result present, update register with result (or store to memory) and remove instr from reorder buffer.
  - Mispredicted branch flushes reorder buffer (sometimes called “graduation”.)

Tomasulo With Reorder Buffer - Cycle 0

Instruction		*j*	*k*
LD	F6	34+	R2
LD	F2	45+	R3
MULTD	F0	F2	F4
SUBD	F8	F6	F2
DIVD	F10	F0	F6
ADDD	F6	F8	F2

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	No						
0	Mult2	No						

Reservation Stations

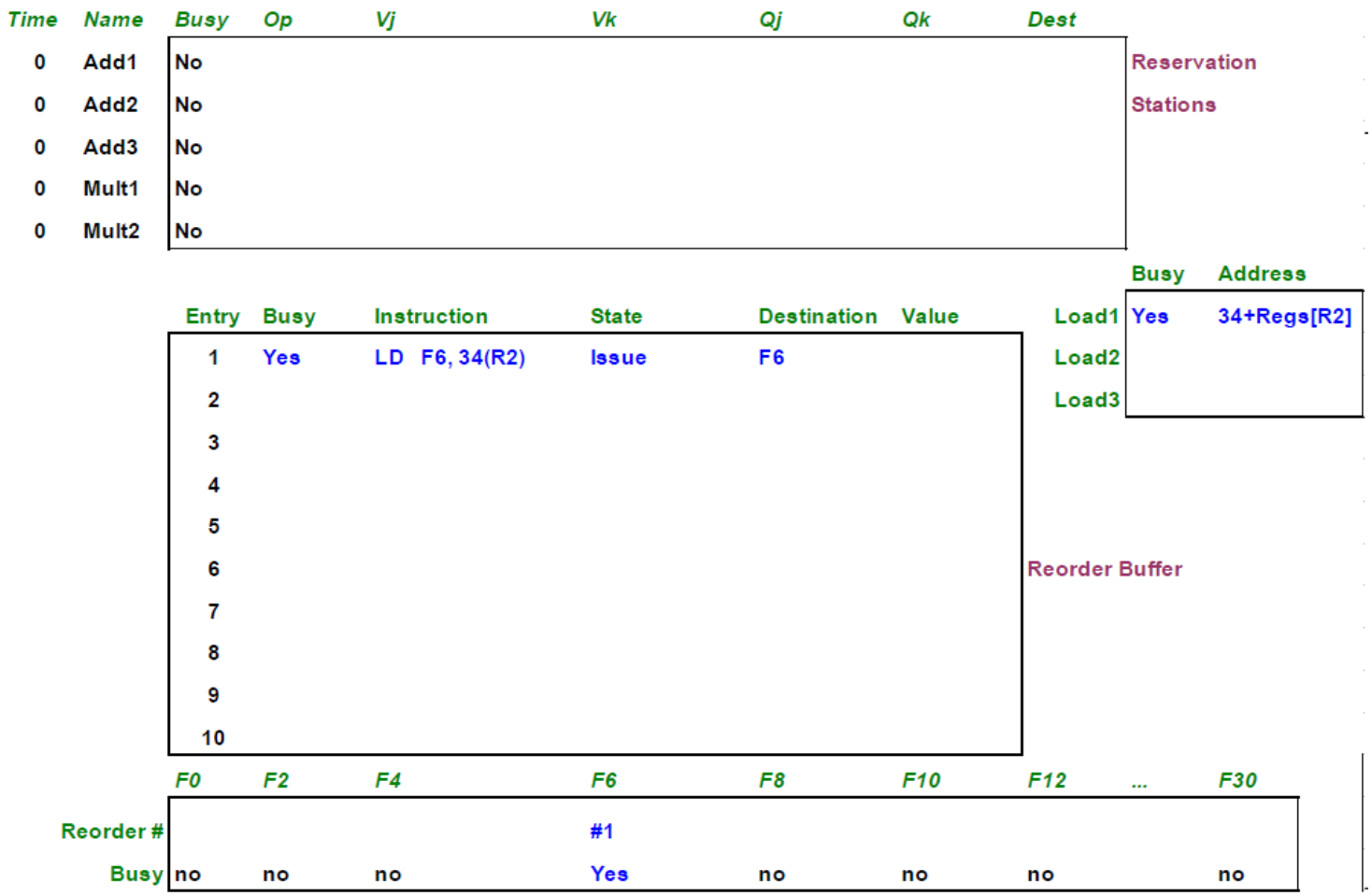
	Busy	Address
Load1		
Load2		
Load3		

Entry	Busy	Instruction	State	Destination	Value
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

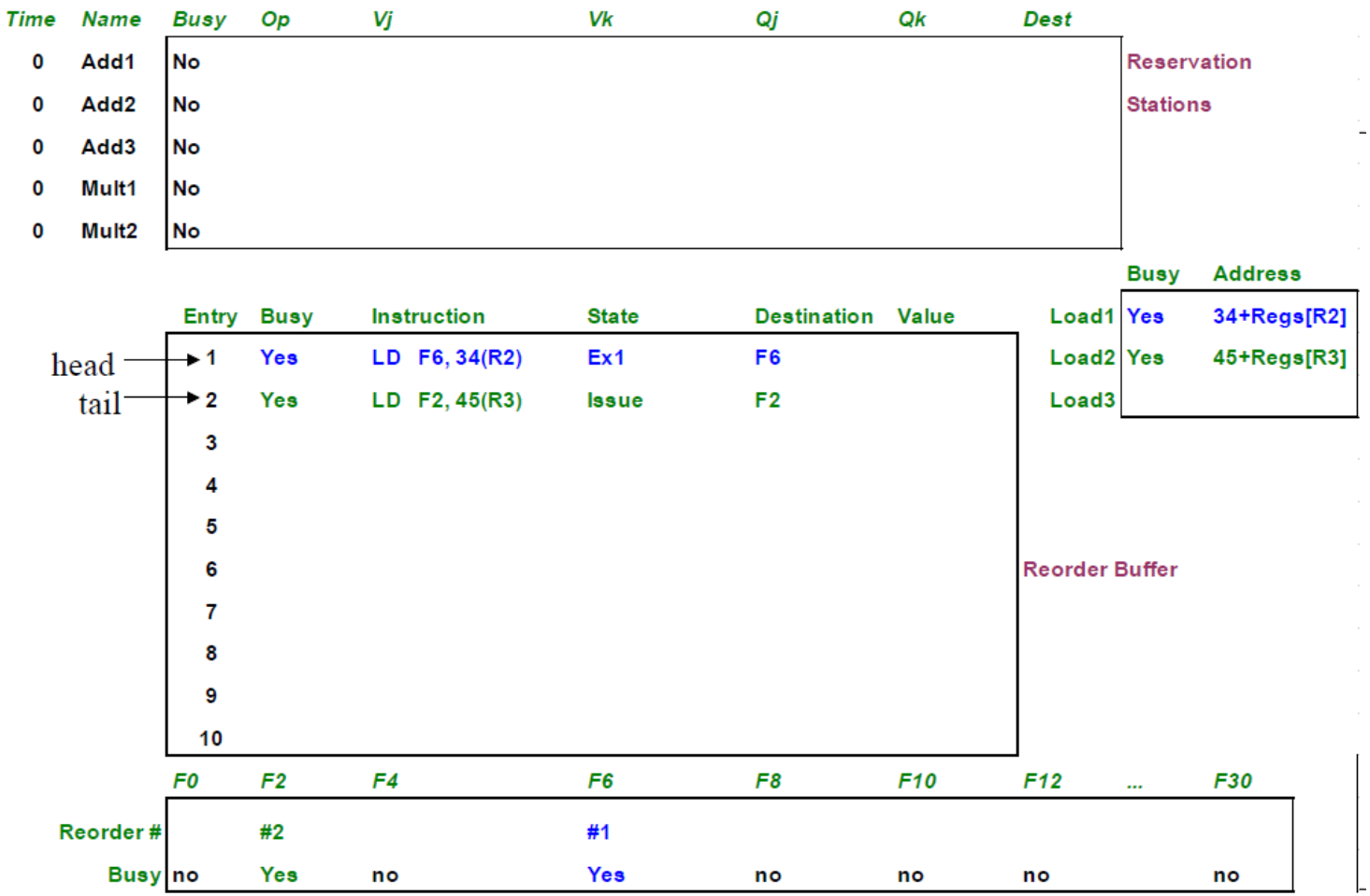
Reorder Buffer

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #									
Busy	no	no	no	no	no	no	no		no

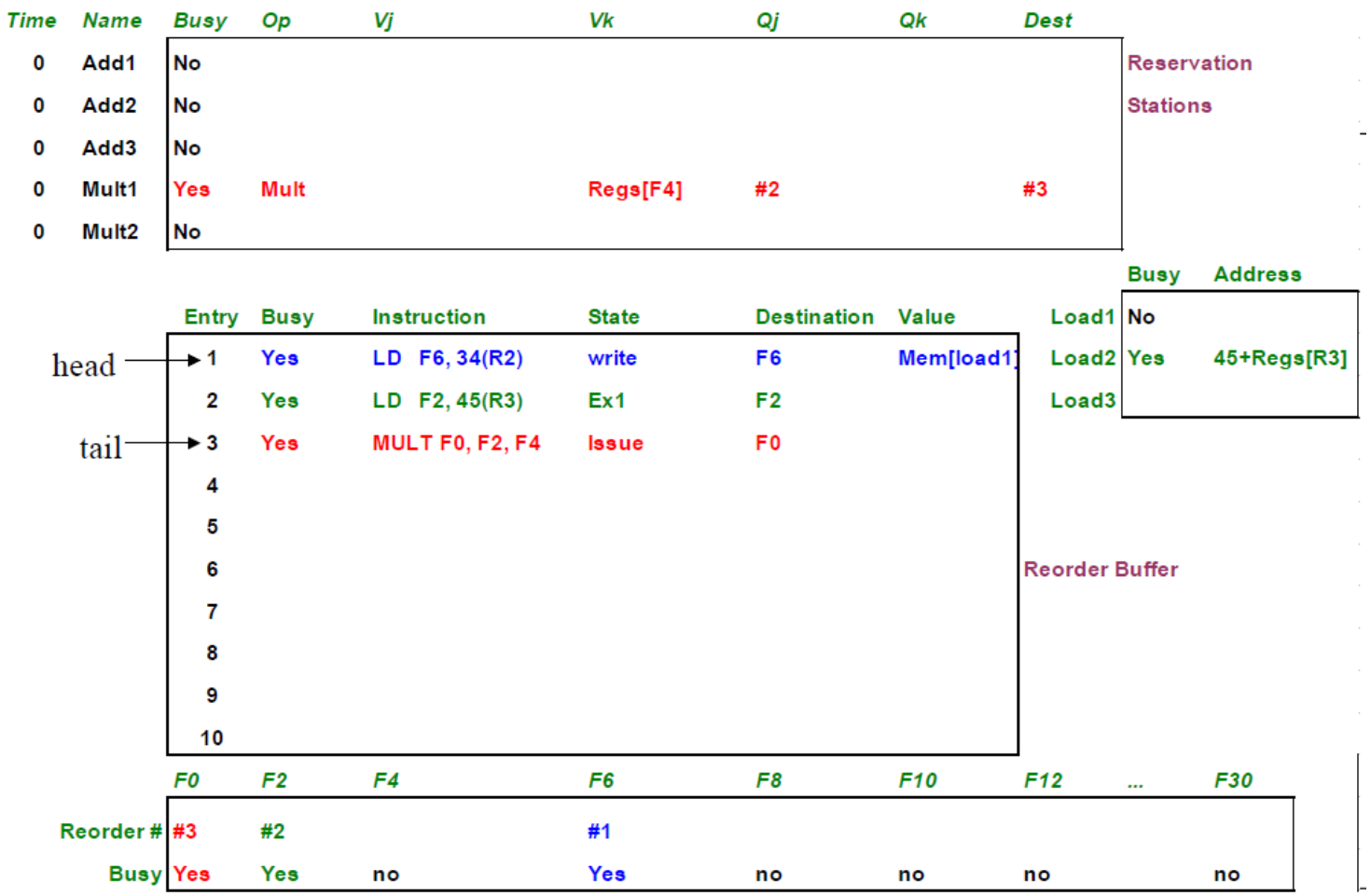
Tomasulo With Reorder Buffer - Cycle 1



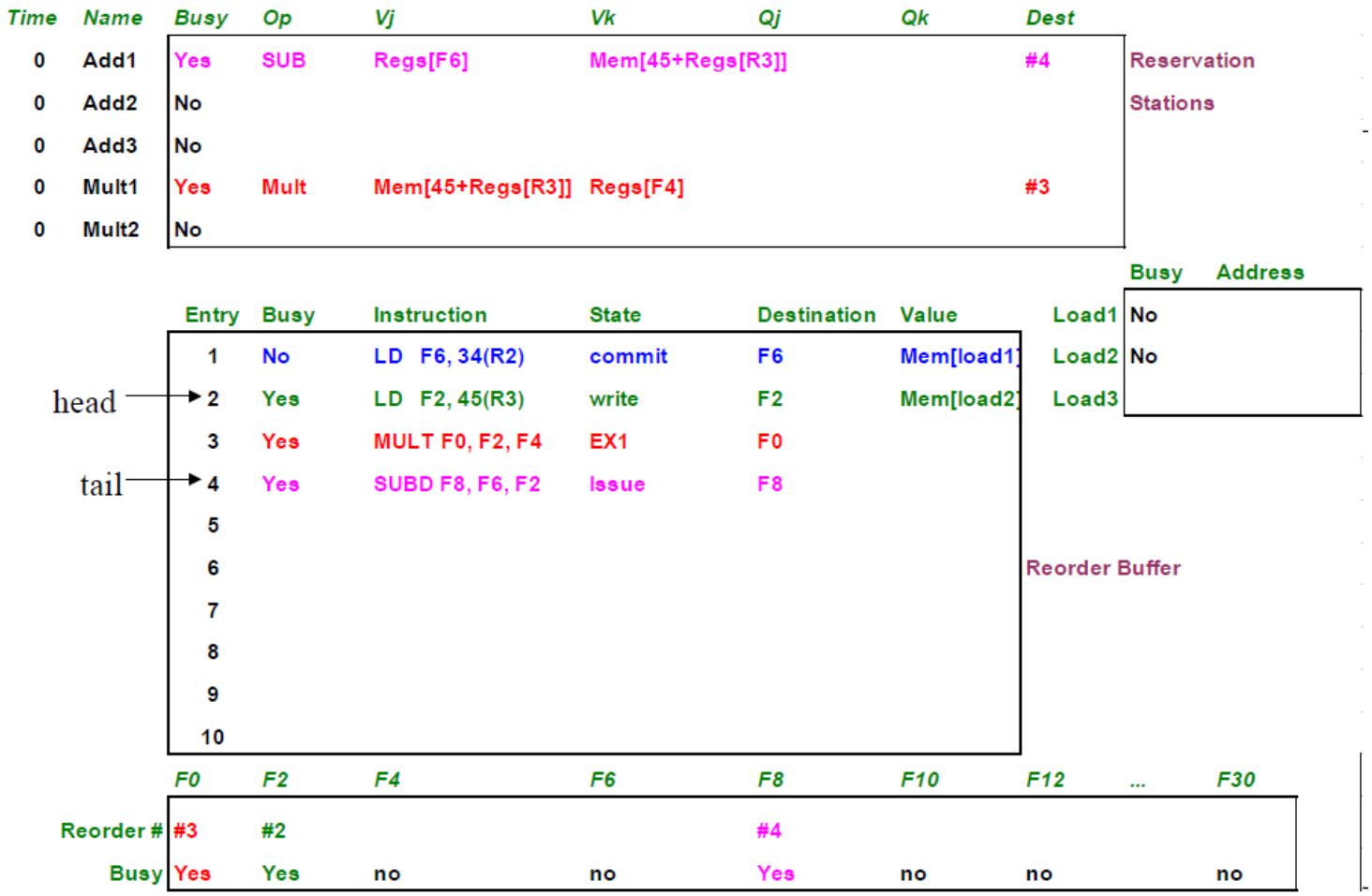
Tomasulo With Reorder Buffer - Cycle 2



Tomasulo With Reorder Buffer - Cycle 3



Tomasulo With Reorder Buffer - Cycle 4



Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	Yes	SUB	Regs[F6]	Mem[45+Regs[R3]]			#4
0	Add2	No						
0	Add3	No						
0	Mult1	Yes	Mult	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	No						

Reservation Stations

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Destination	Value
	1	No	LD F6, 34(R2)	commit	F6	Mem[load1]
head →	2	Yes	LD F2, 45(R3)	write	F2	Mem[load2]
	3	Yes	MULT F0, F2, F4	EX1	F0	
tail →	4	Yes	SUBD F8, F6, F2	Issue	F8	
	5					
	6					
	7					
	8					
	9					
	10					

Reorder Buffer

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #	#3	#2			#4				
Busy	Yes	Yes	no	no	Yes	no	no		no

Tomasulo With Reorder Buffer - Cycle 5

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	Yes	SUB	Regs[F6]	Mem[45+Regs[R3]]			#4
0	Add2	No						
0	Add3	No						
0	Mult1	Yes	Mult	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Stations

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Destination	Value
	1	No	LD F6, 34(R2)	commit	F6	Mem[load1]
	2	No	LD F2, 45(R3)	commit	F2	Mem[load2]
head →	3	Yes	MULT F0, F2, F4	Ex2	F0	
	4	Yes	SUBD F8, F6, F2	Ex1	F8	
tail →	5	Yes	DIVD F10, F0, F6	Issue	F10	
	6					
	7					
	8					
	9					
	10					

Reorder Buffer

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #	#3				#4	#5			
Busy	Yes	no	no	no	Yes	Yes	no		no

Tomasulo With Reorder Buffer - Cycle 6

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	Yes	SUB	Regs[F6]	Mem[45+Regs[R3]]			#4
0	Add2	Yes	Add		Regs[F2]	#4		#6
0	Add3	No						
0	Mult1	Yes	Mult	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Stations

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Destination	Value
	1	No	LD F6, 34(R2)	commit	F6	Mem[load1]
	2	No	LD F2, 45(R3)	commit	F2	Mem[load2]
head →	3	Yes	MULT F0, F2, F4	Ex3	F0	
	4	Yes	SUBD F8, F6, F2	Ex2	F8	
	5	Yes	DIVD F10, F0, F6	Issue	F10	
tail →	6	Yes	ADDD F6, F8, F2	Issue	F6	
	7					
	8					
	9					
	10					

Reorder Buffer

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #	#3			#6	#4	#5			
Busy	Yes	no	no	Yes	Yes	Yes	no		no

Tomasulo With Reorder Buffer - Cycle 7

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	Yes	Add	#4	Regs[F2]			#6
0	Add3	No						
0	Mult1	Yes	Mult	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Stations

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Destination	Value
	1	No	LD F6, 34(R2)	commit	F6	Mem[load1]
	2	No	LD F2, 45(R3)	commit	F2	Mem[load2]
head →	3	Yes	MULT F0, F2, F4	Ex4	F0	
	4	Yes	SUBD F8, F6, F2	write	F8	F6 - #2
	5	Yes	DIVD F10, F0, F6	Issue	F10	
tail →	6	Yes	ADDD F6, F8, F2	EX1	F6	
	7					
	8					
	9					
	10					

Reorder Buffer

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #	#3			#6	#4	#5			
Busy	Yes	no	no	Yes	Yes	Yes	no		no

Tomasulo With Reorder Buffer - Cycle 8

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	Yes	Add	#4	Regs[F2]			#6
0	Add3	No						
0	Mult1	Yes	Mult	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Stations

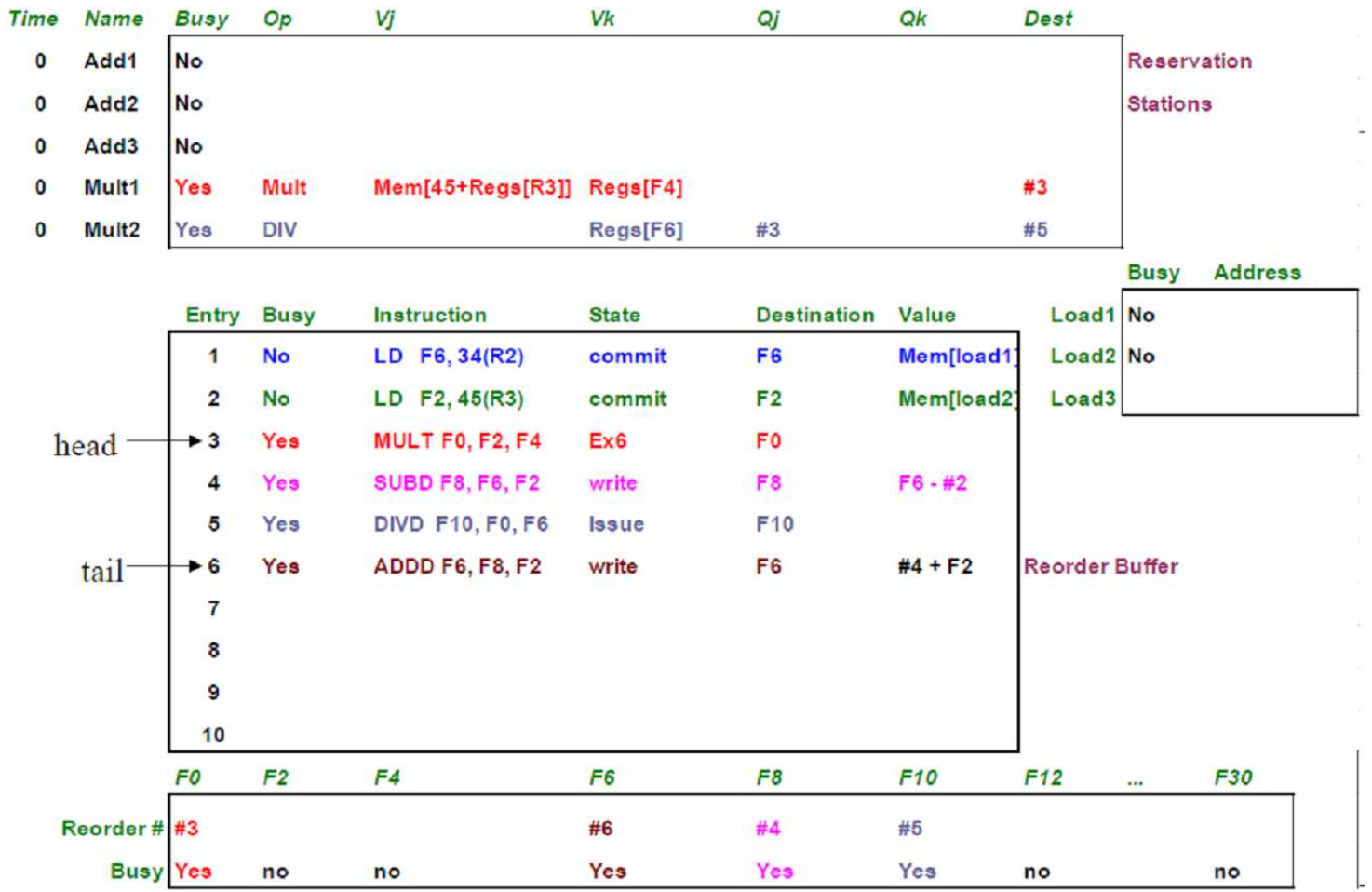
	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Destination	Value
	1	No	LD F6, 34(R2)	commit	F6	Mem[load1]
	2	No	LD F2, 45(R3)	commit	F2	Mem[load2]
head →	3	Yes	MULT F0, F2, F4	Ex5	F0	
	4	Yes	SUBD F8, F6, F2	write	F8	F6 - #2
	5	Yes	DIVD F10, F0, F6	Issue	F10	
tail →	6	Yes	ADDD F6, F8, F2	Ex2	F6	
	7					
	8					
	9					
	10					

Reorder Buffer

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #	#3			#6	#4	#5			
Busy	Yes	no	no	Yes	Yes	Yes	no		no

Tomasulo With Reorder Buffer - Cycle 9



Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	Yes	Mult	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Stations

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Destination	Value
	1	No	LD F6, 34(R2)	commit	F6	Mem[load1]
	2	No	LD F2, 45(R3)	commit	F2	Mem[load2]
head →	3	Yes	MULT F0, F2, F4	Ex6	F0	
	4	Yes	SUBD F8, F6, F2	write	F8	F6 - #2
	5	Yes	DIVD F10, F0, F6	Issue	F10	
tail →	6	Yes	ADDD F6, F8, F2	write	F6	#4 + F2
	7					
	8					
	9					
	10					

Reorder Buffer

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #	#3			#6	#4	#5			
Busy	Yes	no	no	Yes	Yes	Yes	no		no

Tomasulo With Reorder Buffer - Cycle 10

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	Yes	Mult	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Stations

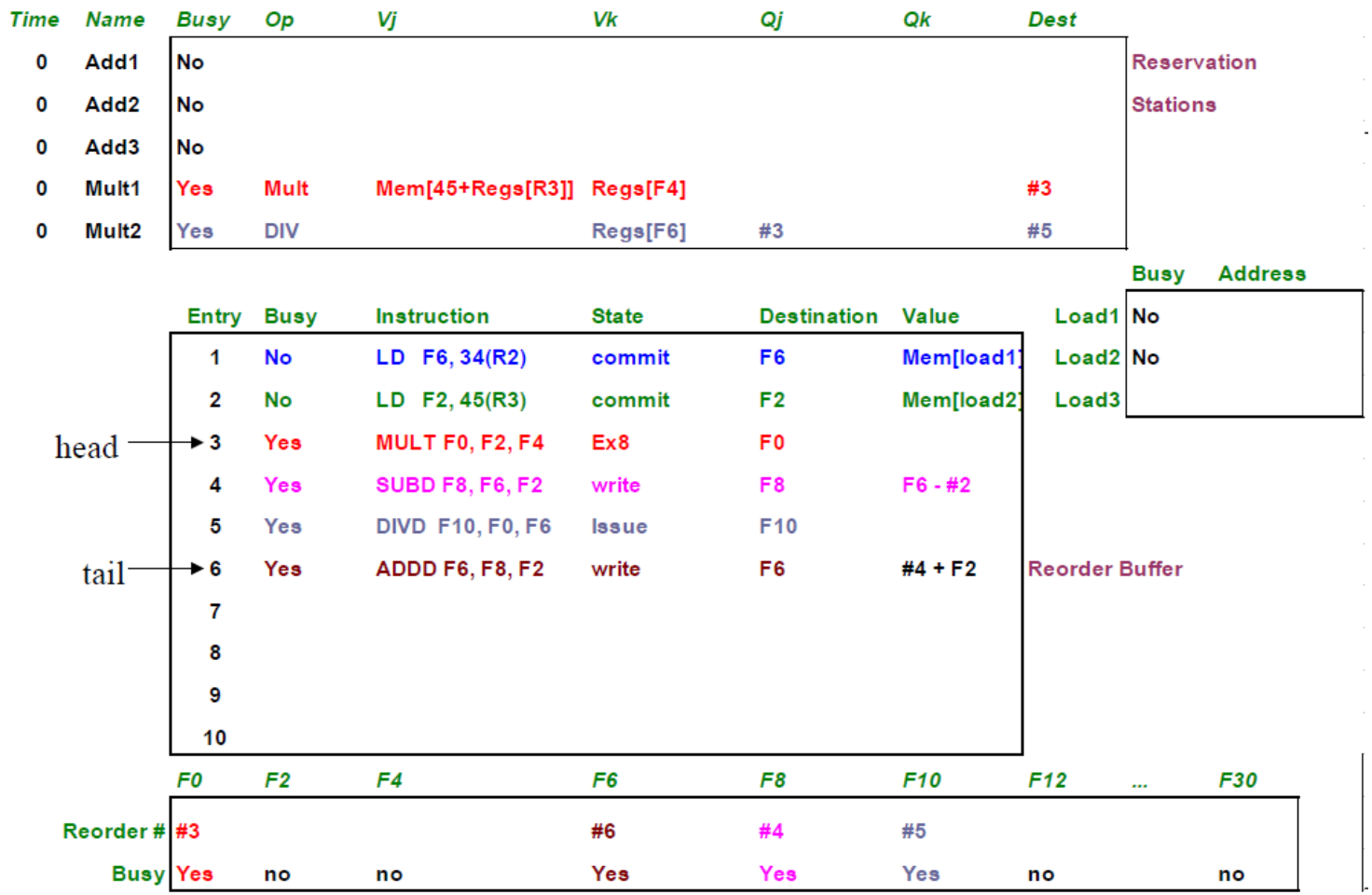
	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Destination	Value
	1	No	LD F6, 34(R2)	commit	F6	Mem[load1]
	2	No	LD F2, 45(R3)	commit	F2	Mem[load2]
head →	3	Yes	MULT F0, F2, F4	Ex7	F0	
	4	Yes	SUBD F8, F6, F2	write	F8	F6 - #2
	5	Yes	DIVD F10, F0, F6	Issue	F10	
tail →	6	Yes	ADDD F6, F8, F2	write	F6	#4 + F2
	7					
	8					
	9					
	10					

Reorder Buffer

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #	#3			#6	#4	#5			
Busy	Yes	no	no	Yes	Yes	Yes	no		no

Tomasulo With Reorder Buffer - Cycle 11



Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	Yes	Mult	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Stations

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Destination	Value
	1	No	LD F6, 34(R2)	commit	F6	Mem[load1]
	2	No	LD F2, 45(R3)	commit	F2	Mem[load2]
head →	3	Yes	MULT F0, F2, F4	Ex8	F0	
	4	Yes	SUBD F8, F6, F2	write	F8	F6 - #2
	5	Yes	DIVD F10, F0, F6	Issue	F10	
tail →	6	Yes	ADDD F6, F8, F2	write	F6	#4 + F2
	7					
	8					
	9					
	10					

Reorder Buffer

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #	#3			#6	#4	#5			
Busy	Yes	no	no	Yes	Yes	Yes	no		no

Tomasulo With Reorder Buffer - Cycle 12

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	Yes	Mult	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Stations

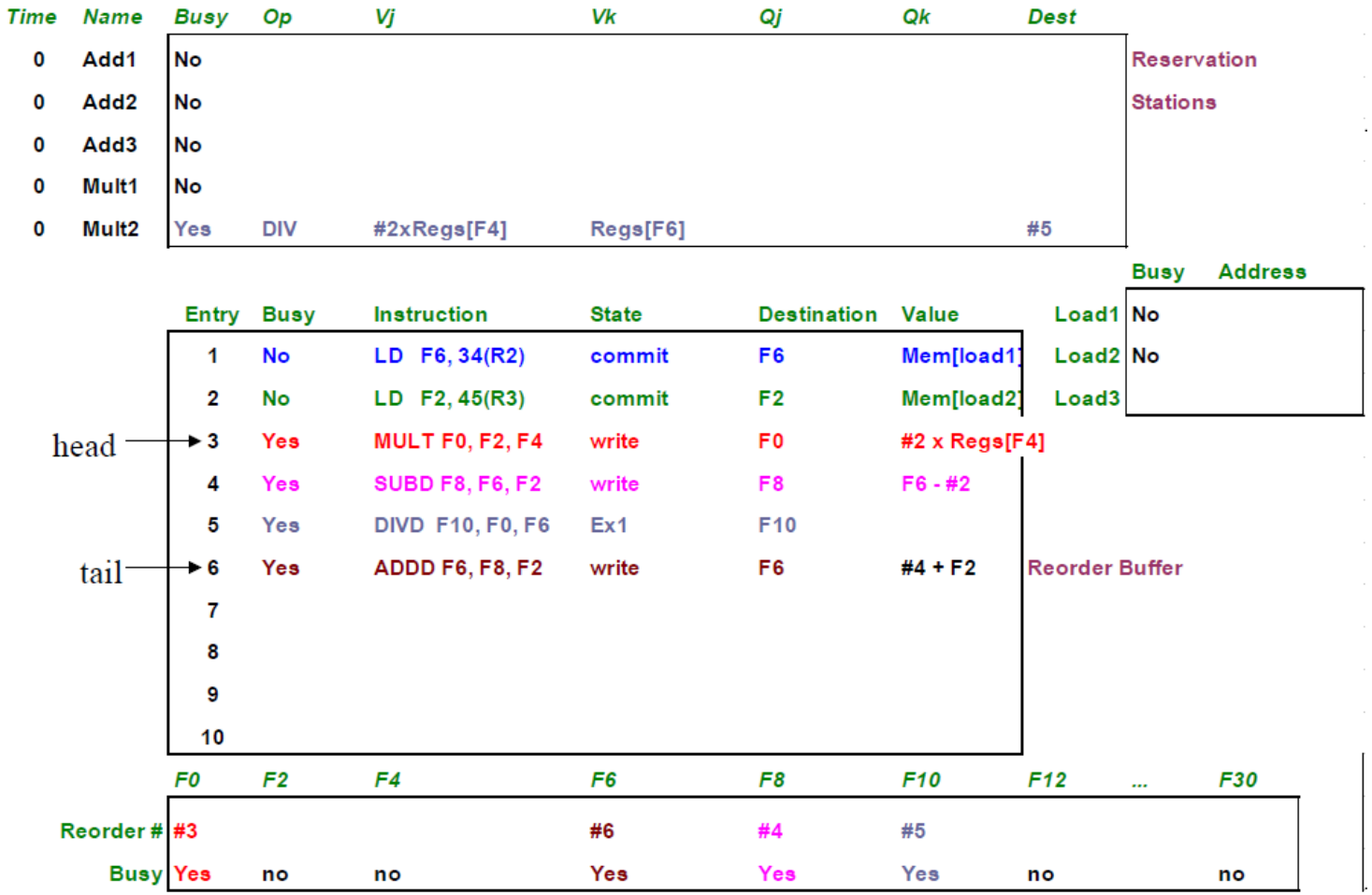
	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Destination	Value
	1	No	LD F6, 34(R2)	commit	F6	Mem[load1]
	2	No	LD F2, 45(R3)	commit	F2	Mem[load2]
head →	3	Yes	MULT F0, F2, F4	Ex9	F0	
	4	Yes	SUBD F8, F6, F2	write	F8	F6 - #2
	5	Yes	DIVD F10, F0, F6	Issue	F10	
tail →	6	Yes	ADDD F6, F8, F2	write	F6	#4 + F2
	7					
	8					
	9					
	10					

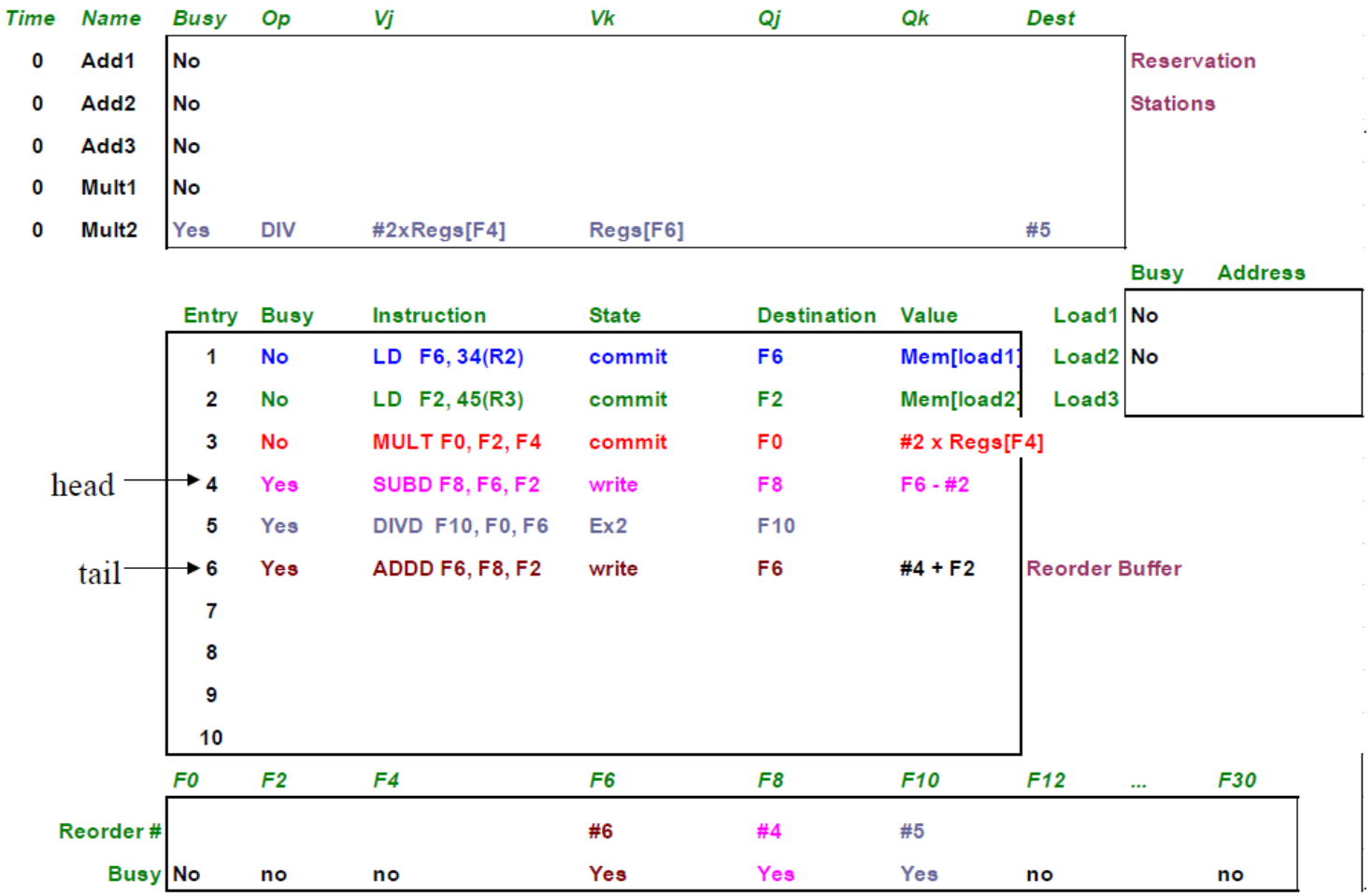
Reorder Buffer

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #	#3			#6	#4	#5			
Busy	Yes	no	no	Yes	Yes	Yes	no		no

Tomasulo With Reorder Buffer - Cycle 13



Tomasulo With Reorder Buffer - Cycle 14



Tomasulo With Reorder Buffer - Cycle 15

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	No						
0	Mult2	Yes	DIV	#2xRegs[F4]	Regs[F6]			#5

Reservation Stations

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Destination	Value
	1	No	LD F6, 34(R2)	commit	F6	Mem[load1]
	2	No	LD F2, 45(R3)	commit	F2	Mem[load2]
	3	No	MULT F0, F2, F4	commit	F0	#2 x Regs[F4]
	4	No	SUBD F8, F6, F2	commit	F8	F6 - #2
head →	5	Yes	DIVD F10, F0, F6	Ex3	F10	
tail →	6	Yes	ADDD F6, F8, F2	write	F6	#4 + F2
	7					
	8					
	9					
	10					

Reorder Buffer

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #				#6		#5			
Busy	no	no	no	Yes	no	Yes	no		no

Tomasulo With Reorder Buffer - Cycle 16

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	No						
0	Mult2	Yes	DIV	#2xRegs[F4]	Regs[F6]			#5

Reservation Stations

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Destination	Value
	1	No	LD F6, 34(R2)	commit	F6	Mem[load1]
	2	No	LD F2, 45(R3)	commit	F2	Mem[load2]
	3	No	MULT F0, F2, F4	commit	F0	#2 x Regs[F4]
	4	No	SUBD F8, F6, F2	commit	F8	F6 - #2
head →	5	Yes	DIVD F10, F0, F6	Ex4	F10	
tail →	6	Yes	ADDD F6, F8, F2	write	F6	#4 + F2
	7					
	8					
	9					
	10					

Reorder Buffer

Need 36 more EX cycles for DIV to finish…

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #				#6		#5			
Busy	no	no	no	Yes	no	Yes	no		no

Tomasulo With Reorder Buffer: Summary

Instruction	Issue	Exec Comp	Writeback	Commit
LD F6, 34(R2)	1	2	3	4
LD F2, 45(R3)	2	3	4	5
MULT F0, F2, F4	3	12	13	14
SUBD F8, F6, F2	4	6	7	15
DIVD F10, F0, F6	5	52	53	54
ADDD F6, F8, F2	6	8	9	55

In-order Issue/Commit, Out-of-Order Execution/Writeback

Precise State with ROB

- ROB maintains precise state and allows speculation
  - Waits until precise condition reaches retire/commit stage
  - Or until branch is noted mispredicted
    - Clear ROB, RS, and register status table (Flush)
    - Service exception/Restart from True Branch target

- Need to do similar things with memory ops
  - Called Memory Ordering Buffer (MOB)
    - Completed stores write to MOB then complete (write to memory) in-order (when they reach head of buffer)

Tomasulo + ROB Summary

- Many implementations are very similar
  - Pentium III, PowerPC, etc

- Some limitations
  - Too many value copy operations
    - Register file => RS => ROB => Register File
  - Too many muxes/busses (CDB)
    - Values are coming from everywhere to everywhere else!
  - Reservation Stations mix values (data) and tags (control)
    - Slows down the max clock frequency