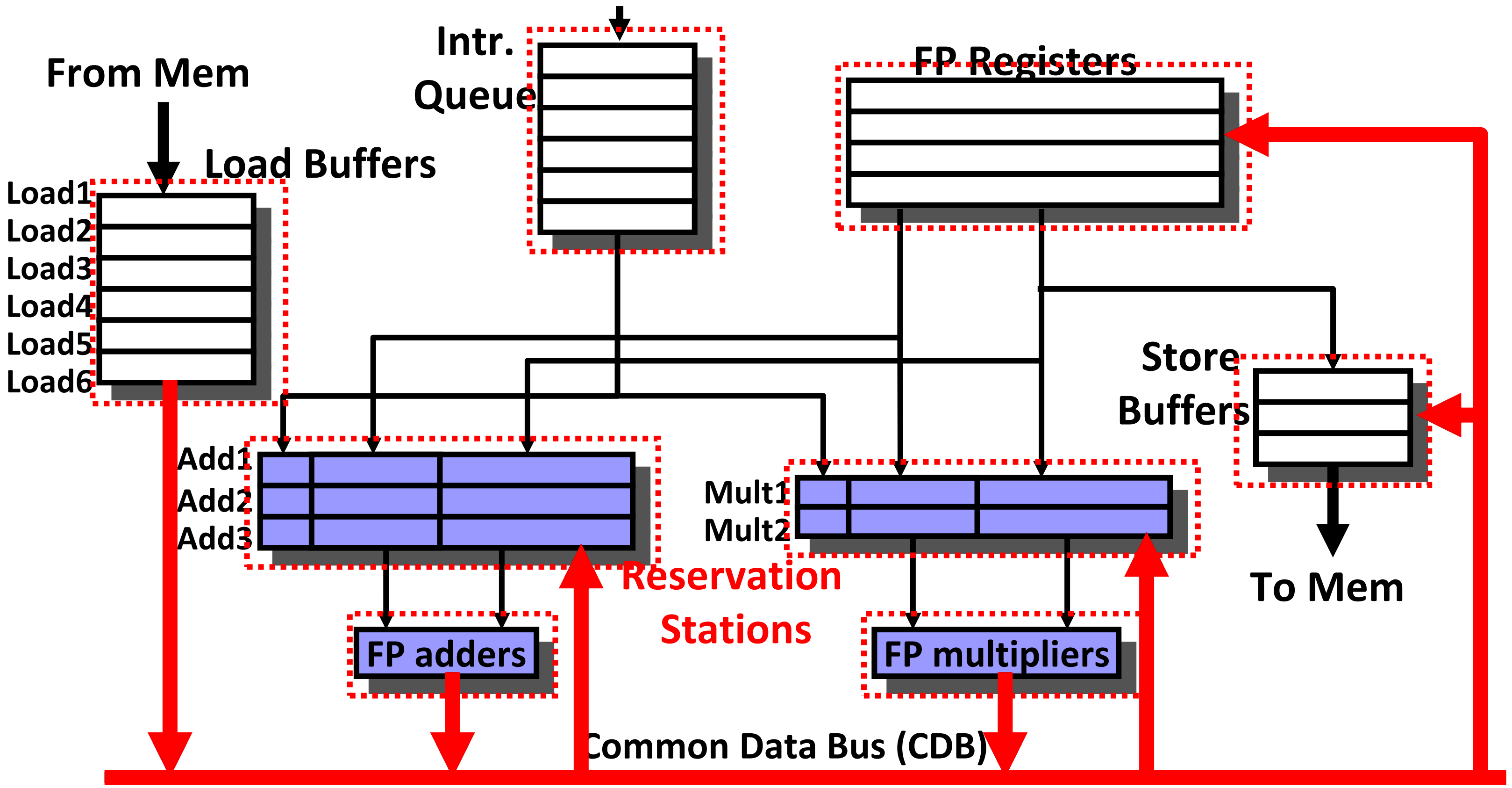
Tomasulo with ROB(Re-Order Buffer)

汪东升 (Prof. Dongsheng Wang)

wds@tsinghua.edu.cn

清华大学计算机系

Tomasulo-based FPU for MIPS



Scoreboard vs Tomasulo

	Scoreboard	**Tomasulo**
Window size:	≤ 5 instructions	≤ 14 instructions
Structural hazard:	stall pipeline	No issue
WAR dependency	stall completion	renaming avoids
WAW dependency:	stall pipeline	renaming avoids
Results forwarding:	Write/read registers	Broadcast from FU
Control structure:	central scoreboard	distributed reservation stations

Example

In-order

RAW

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LD R1 X	IF	ID	LD1	LD2	LD3	LD4	WB									
LD R2 Y		IF	ID	LD1	LD2	LD3	LD4	WB								
ADD R3 R1 R2			IF	ID	Stall	Stall	Stall	Stall	Add1	Add2	WB					
SUB R3 R5 R6				IF	Stall	Stall	Stall	Stall	ID	Sub1	Sub2	WB				
MUL R4 R1 R1									IF	ID	Mul1	Mul2	Mul3	Mul4	WB	
DIV R7 R5 R6										IF	ID	Div1	Div2	Div3	Div4	WB

RAW – Stall the pipeline

Out-of-order with Scoreboard

WAW

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LD R1 X	I	RO	LD1	LD2	LD3	LD4	**WB**								
LD R2 Y		I	RO	LD1	LD2	LD3	LD4								
ADD R3 R1 R2			I	RO	RO	RO	RO	**RO**	Add1	Add2	**WB**				
SUB R3 R5 R6				I	I	I	I	RO	Sub1	Sub2	**WB**				
MUL R4 R1 R1								I	**RO**	Mul1	Mul2	Mul3	Mul4	WB	
DIV R7 R5 R6									I	RO	Div1	Div2	Div3	Div4	WB

RAW – ADD stalled, SUB could be issued

Out-of-order with Tomasulo

	1	2	3	4	5	6	7	8	9	10	11	12
LD R1 X	I	LD1	LD2	LD3	LD4	CDB						
LD R2 Y		I	LD1	LD2	LD3	LD4	**CDB**					
ADD R3 R1 R2			I	RS	RS	RS	RS	Add1	Add2	CDB		
SUB R3 R5 R6				I	Sub1	Sub2	**CDB**	CDB				
MUL R4 R1 R1					I	**RS**	Mul1	Mul2	Mul3	Mul4	**CDB**	
DIV R7 R5 R6						I	Div1	Div2	Div3	Div4	**CDB**	CDB

RAW – ADD stalled, SUB can be issued

LD – 4 cycles
Add/Sub – 2 cycles
Mul/Div – 2 cycles

Assuming no structural Hazards

Example

Tsinghua University

In-order

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LD R1 X	IF	ID	LD1	LD2	LD3	LD4	WB									
LD R2 Y		IF	ID	LD1	LD2	LD3	LD4	WB								
ADD R3 R1 R2			IF	ID	**Stall**	**Stall**	**Stall**	**Stall**	Add1	Add2	WB					
SUB R3 R5 R6				IF	**Stall**	**Stall**	**Stall**	**Stall**	ID	Sub1	Sub2	WB				
MUL R4 R1 R1									IF	ID	Mul1	Mul2	Mul3	Mul4	WB	
DIV R7 R5 R6										IF	ID	Div1	Div2	Div3	Div4	WB

Out-of-order with Scoreboard

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LD R1 X	I	RO	LD1	LD2	LD3	LD4	**WB**								
LD R2 Y		I	RO	LD1	LD2	LD3	LD4								
ADD R3 R1 R2			I	**RO**	**RO**	**RO**	**RO**								
SUB R3 R5 R6				**I**	**I**	**I**	**I**								
MUL R4 R1 R1								I	**RO**	Mul1	Mul2	Mul3	Mul4	WB	
DIV R7 R5 R6									I	RO	Div1	Div2	Div3	Div4	WB

WAW

WAW – SUB cannot be issued
Stall the pipeline

Out-of-order with Tomasulo

	1	2	3	4	5	6	7	8	9	10	11	12
LD R1 X	I	LD1	LD2	LD3	LD4	CDB						
LD R2 Y		I	LD1	LD2	LD3	LD4	**CDB**					
ADD R3 R1 R2			I	**RS**	**RS**	**RS**	**RS**					
SUB R3 R5 R6				I	Sub1	Sub2	**CDB**	CDB				
MUL R4 R1 R1					I	**RS**	Mul1	Mul2	Mul3	Mul4	**CDB**	
DIV R7 R5 R6						I	Div1	Div2	Div3	Div4	**CDB**	CDB

WAW – Allowed by register renaming in RS

LD – 4 cycles
Add/Sub – 2 cycles
Mul/Div – 2 cycles

Assuming no structural Hazards

Example

In-order

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LD R1 X	IF	ID	LD1	LD2	LD3	LD4	WB									
LD R2 Y		IF	ID	LD1	LD2	LD3	LD4	WB								
ADD R3 R1 R2			IF	ID	**Stall**	**Stall**	**Stall**	**Stall**	Add1	Add2	WB					
SUB R3 R5 R6				IF	**Stall**	**Stall**	**Stall**	**Stall**	ID	Sub1	Sub2	WB				
MUL R4 R1 R1									IF	ID	Mul1	Mul2	Mul3	Mul4	WB	
DIV R7 R5 R6										IF	ID	Div1	Div2	Div3	Div4	WB

Out-of-order with Scoreboard

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LD R1 X	I	RO	LD1	LD2	LD3	LD4	**WB**								
LD R2 Y		I	RO	LD1	LD2	LD3	LD4	**WB**							
ADD R3 R1 R2			I	**RO**	**RO**	**RO**	**RO**	**RO**	Add1	Add2	**WB**				
SUB R3 R5 R6				**I**	**I**	**I**	**I**	RO	Sub1	Sub2	**WB**				
MUL R4 R1 R1								I	**RO**	Mul1	Mul2				
DIV R7 R5 R6									I	RO	Div1				

2 instrs. can finish at the same time (assuming enough ports in the Register bank)

Out-of-order with Tomasulo

	1	2	3	4	5	6	7	8	9	10	11	
LD R1 X	I	LD1	LD2	LD3	LD4	CDB						
LD R2 Y		I	LD1	LD2	LD3	LD4	**CDB**					
ADD R3 R1 R2			I	**RS**	**RS**	**RS**	**RS**	Add1	Add2	CDB		
SUB R3 R5 R6				I	Sub1	Sub2	**CDB**	CDB				
MUL R4 R1 R1					I	**RS**	Mul1	Mul2	Mul3	Mul4	**CDB**	
DIV R7 R5 R6						I	Div1	Div2	Div3	Div4	**CDB**	CDB

CDB limits finishing instrs. to one/cycle

LD – 4 cycles
Add/Sub – 2 cycles
Mul/Div – 2 cycles

Assuming no structural Hazards

Review Tomasulo

- 引入动态调度的动机
  - 在没有专用编译器的情况下，提高系统性能
  - 解决编译时无法判定的部分相关问题
  - **Scoreboard** 和**Tomasulo**
- **Tomasulo** 主要贡献
  - **Dynamic scheduling**
  - **Register renaming---**消除了**WAW**，**WAR** 相关
- 算法的主要缺陷
  - 复杂
  - 要求高速**CDB**（**Common Data Bus**）
  - 性能受限于**CDB**

- 动态硬件方案可以用硬件进行循环展开
- 如何处理分支?
  - 我们可以用硬件做循环展开必须可以解决分支指令问题
- 如何处理精确中断?
  - Out-of-order execution ⇒ out-of-order completion!

关于异常处理

- 乱序完成加大了实现精确中断的难度
  - 在前面指令还没有完成时，寄存器文件中可能会有后面指令的运行结果.
  - 如果这些前面的指令执行时有中断产生，怎么办？
  - 例如：

```
DIVD F10, F0, F2
SUBD F4, F6, F8
ADDD F12, F14, F16
```

- 需要**"rollback"** 寄存器文件到原来的状态:
  - 精确中断的含义是其返回地址为：
    - 该地址之前的所有指令都已完成
    - 其后的指令还都没有完成
- 实现精确中断的技术：顺序完成（或提交）
  - 即提交指令完成的顺序必须与指令发射的顺序相同

控制相关的动态解决技术

- 控制相关：由条件转移或程序中断引起的相关，也称全局相关。控制相关对流水线的吞吐率和效率影响相对于数据相关要大得多
  - 条件指令在一般程序中所占的比例相当大
  - 中断虽然在程序中所占的比例不大，但中断发生在程序中的哪一条指令，发生在一条指令执行过程中的哪一个功能段都是不确定的
- 处理好条件转移和中断引起的控制相关是很重要的。
- 关键问题：
  - 要确保流水线能够正常工作
  - 减少因断流引起的吞吐率和效率的下降

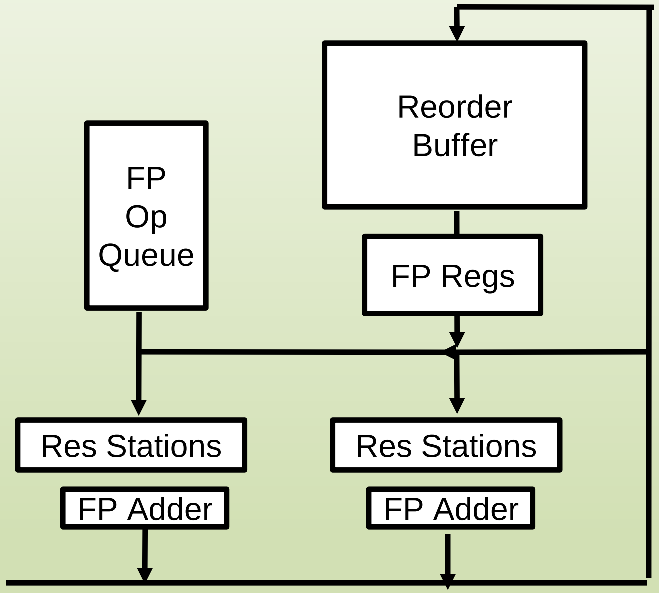
Dynamic Branch Prediction

- 动态分支预测：预测分支的方向在程序运行时刻动态确定
- 需解决的关键问题是：
  - 如何记录转移历史信息
  - 如何根据所记录的转移历史信息，预测转移的方向
- 主要方法
  - 基于**BPB(Branch Prediction Buffer)**或**BHT(Branch History Table)**的方法
    - 1-bit BHT和2-bit BHT
    - Correlating Branch Predictors
    - Tournament Predictors: Adaptively Combining Local and Global Predictors
  - **High Performance Instruction Delivery**
    - BPB
    - Integrated Instruction Fetch Units
    - Return Address Predictors

- **Performance = *f*(accuracy, cost of misprediction)**
  - **Misprediction $\Rightarrow$ Flush Reorder Buffer**

硬件支持精确中断/分支预测

- 需要硬件缓存没有提交的指令结果: ***reorder buffer*** ***（ROB）***
  - **3** 个域: **指令类型,目的地址, 值**
  - **Reorder buffer** 可以作为操作数源 **=>** 就像有更多的寄存器（与**RS**类似）
  - 当程序执行阶段完成后，用**ROB**的编号代替**RS**中的值
  - 增加指令提交阶段
  - **ROB**提供执行完成阶段和提交阶段的操作数
  - 一旦操作数提交，结果就写入寄存器
  - 这样，在预测失败时，容易恢复推断执行的指令，或发生异常时，容易恢复状态



Technique for both precise interrupts/exceptions and speculation: in-order completion or commit

Tomasulo with ROB

1. Issue—get instruction from FP Op Queue
   - 如果RS和ROB有空闲单元就发射指令。如果寄存器或ROB中源操作数可用，就将其发送到RS，目的地址的ROB编号也发送给RS (this stage sometimes called "dispatch")
   - 否则，指令发射停顿
2. Execution—operate on operands (EX)
   - 当操作数就绪后，开始执行。如果没有就绪，监测CDB，检查RAW相关（ISSUE）
3. Write result—finish execution (WB)
   - 将运算结果通过CDB传送给所有等待结果的FU以及ROB单元，标识RS可用
4. Commit—update register with reorder result
   - 按ROB表中顺序，如果结果已有，就更新寄存器（或存储器），并将该指令从ROB表中删除
   - 预测失败或有中断时，刷新ROB

```
LD      F6, 34(R2)
LD      F2, 45(R3)
MULT    F0, F2, F4
SUBD    F8, F6, F2
DIVD    F10, F0, F6
ADDD    F6, F8, F2
```

Tomasulo With Reorder Buffer - Cycle 0

Reservation Stations

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

	Busy	Address
Load1		
Load2		
Load3		

Reorder Buffer

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

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

Tomasulo With Reorder Buffer - Cycle 1

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	F0, F2, F4
SUBD	F8, F6, F2
DIVD	F10, F0, F6
ADDD	F6, F8, F2

Reservation Stations

R S

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

	Busy	Address
Load1	Yes	34+Regs[R2]
Load2		
Load3		

Reorder Buffer

ROB

Entry	Busy	Instruction	State	Destination	Value
1	Yes	LD F6, 34(R2)	Issue	F6	
2					
3					
4					
5					
6					
7					
8					
9					
10					

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #				#1					
Busy	no	no	no	Yes	no	no	no		no

Tomasulo With Reorder Buffer - Cycle 2

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	F0, F2, F4
SUBD	F8, F6, F2
DIVD	F10, F0, F6
ADDD	F6, F8, F2

Reservation Stations

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

	Busy	Address
Load1	Yes	34+Regs[R2]
Load2	Yes	45+Regs[R3]
Load3		

Reorder Buffer

	Entry	Busy	Instruction	State	Destination	Value
head →	1	Yes	LD F6, 34(R2)	Ex1	F6	
tail →	2	Yes	LD F2, 45(R3)	Issue	F2	
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #		#2		#1					
Busy	no	Yes	no	Yes	no	no	no		no

Tomasulo With Reorder Buffer - Cycle 3

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	F0, F2, F4
SUBD	F8, F6, F2
DIVD	F10, F0, F6
ADDD	F6, F8, F2

Reservation Stations

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	Yes	Mult		Regs[F4]	#2		#3
0	Mult2	No						

	Busy	Address
Load1	No	
Load2	Yes	45+Regs[R3]
Load3		

Reorder Buffer

	Entry	Busy	Instruction	State	Destination	Value
head →	1	Yes	LD F6, 34(R2)	write	F6	Mem[load1]
	2	Yes	LD F2, 45(R3)	Ex1	F2	
tail →	3	Yes	MULT F0, F2, F4	Issue	F0	
	4					
	5					
	6					
	7					
	8					
	9					
	10					

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #	#3	#2		#1					
Busy	Yes	Yes	no	Yes	no	no	no		no

Tomasulo With Reorder Buffer - Cycle 4

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	F0, F2, F4
SUBD	F8, F6, F2
DIVD	F10, F0, F6
ADDD	F6, F8, F2

Reservation Stations

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						

	Busy	Address
Load1	No	
Load2	No	
Load3		

Reorder Buffer

	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					

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

Tsinghua University

Tomasulo With Reorder Buffer - Cycle 5

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	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	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		

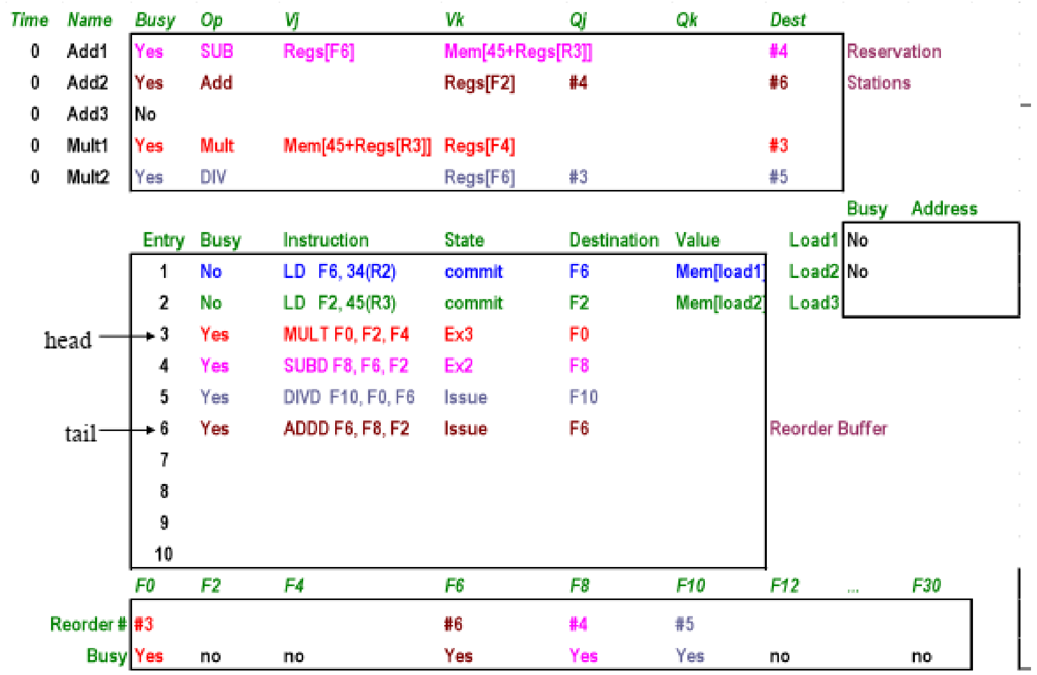
	Entry	Busy	Instruction	State	Destination	Value
	1	No	LD F6, 34(R2)	commit	F6	Mem[load1]
	2	No	LD F2, 45(R3)	commit	F2	Mem[load2]
head →	3	Yes	MULT F0, F2, F4	Ex2	F0	
	4	Yes	SUBD F8, F6, F2	Ex1	F8	
tail →	5	Yes	DIVD F10, F0, F6	Issue	F10	
	6					
	7					
	8					
	9					
	10					

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

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	F0, F2, F4
SUBD	F8, F6, F2
DIVD	F10, F0, F6
ADDD	F6, F8, F2



Reservation Stations

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

	Busy	Address
Load1	No	
Load2	No	
Load3		

Reorder Buffer

	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					

	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

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	F0, F2, F4
SUBD	F8, F6, F2
DIVD	F10, F0, F6
ADDD	F6, F8, F2

Reservation Stations

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

	Busy	Address
Load1	No	
Load2	No	
Load3		

Reorder Buffer

	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					

	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

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	F0, F2, F4
SUBD	F8, F6, F2
DIVD	F10, F0, F6
ADDD	F6, F8, F2

Reservation Stations

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

	Busy	Address
Load1	No	
Load2	No	
Load3		

Reorder Buffer

	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					

	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

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	F0, F2, F4
SUBD	F8, F6, F2
DIVD	F10, F0, F6
ADDD	F6, F8, F2

Reservation Stations

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

	Busy	Address
Load1	No	
Load2	No	
Load3		

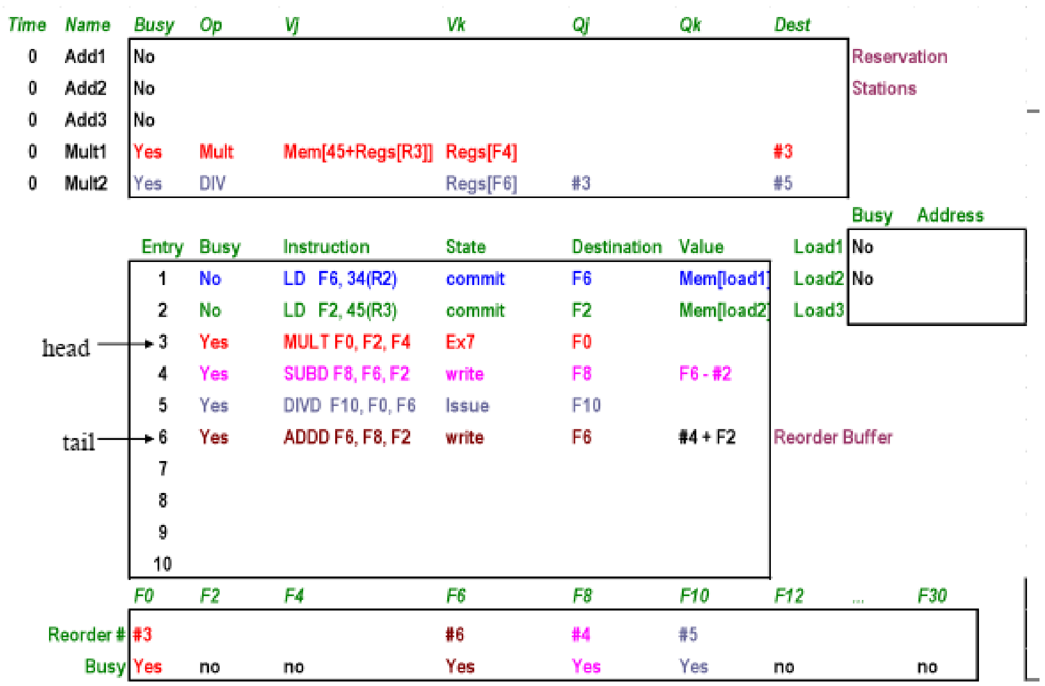
Reorder Buffer

	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					

	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

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	F0, F2, F4
SUBD	F8, F6, F2
DIVD	F10, F0, F6
ADDD	F6, F8, F2



Reservation Stations

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

	Busy	Address
Load1	No	
Load2	No	
Load3		

Reorder Buffer

	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					

	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

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	F0, F2, F4
SUBD	F8, F6, F2
DIVD	F10, F0, F6
ADDD	F6, F8, F2

Reservation Stations

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

	Busy	Address
Load1	No	
Load2	No	
Load3		

Reorder Buffer

	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					

	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

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	F0, F2, F4
SUBD	F8, F6, F2
DIVD	F10, F0, F6
ADDD	F6, F8, F2

Reservation Stations

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

	Busy	Address
Load1	No	
Load2	No	
Load3		

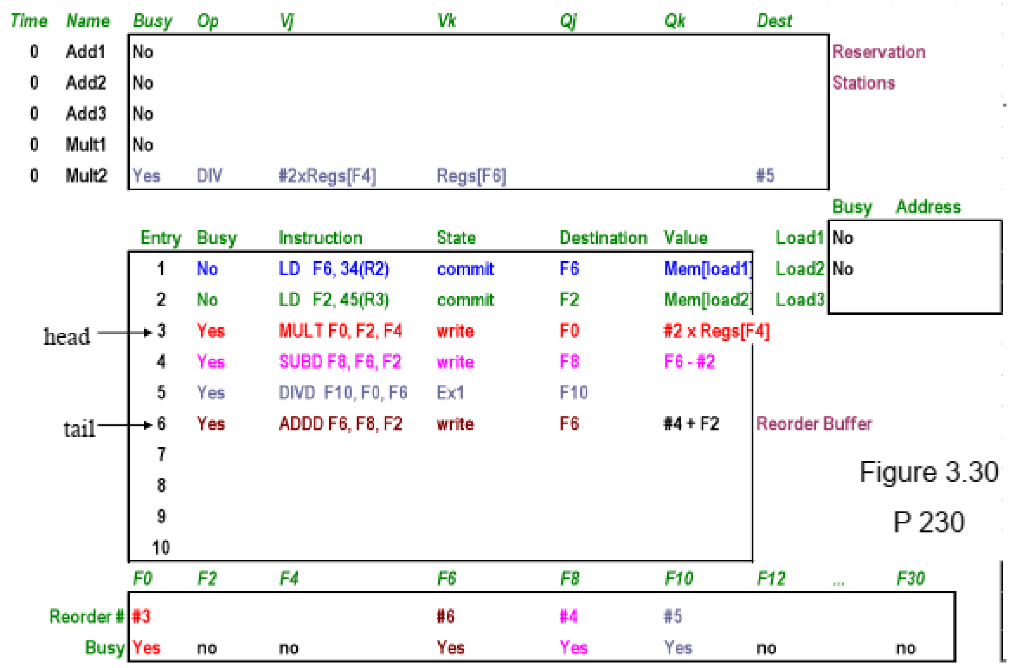
Reorder Buffer

	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					

	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

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	F0, F2, F4
SUBD	F8, F6, F2
DIVD	F10, F0, F6
ADDD	F6, F8, F2



Reservation Stations

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

	Busy	Address
Load1	No	
Load2	No	
Load3		

Reorder Buffer

	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	write	F0	#2 x Regs[F4]
	4	Yes	SUBD F8, F6, F2	write	F8	F6 - #2
	5	Yes	DIVD F10, F0, F6	Ex1	F10	
tail	6	Yes	ADDD F6, F8, F2	write	F6	#4 + F2
	7					
	8					
	9					
	10					

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

Figure 3.30

P 230

写到ROB，DIVD开始执行 @Clock13

Tomasulo With Reorder Buffer - Cycle 14

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	F0, F2, F4
SUBD	F8, F6, F2
DIVD	F10, F0, F6
ADDD	F6, F8, F2

Reservation Stations

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

	Busy	Address
Load1	No	
Load2	No	
Load3		

Reorder Buffer

	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]
head →	4	Yes	SUBD F8, F6, F2	write	F8	F6 - #2
	5	Yes	DIVD F10, F0, F6	Ex2	F10	
tail →	6	Yes	ADDD F6, F8, F2	write	F6	#4 + F2
	7					
	8					
	9					
	10					

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

Tomasulo With Reorder Buffer - Cycle 15

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	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	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

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	F0, F2, F4
SUBD	F8, F6, F2
DIVD	F10, F0, F6
ADDD	F6, F8, F2

Reservation Stations

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

	Busy	Address
Load1	No	
Load2	No	
Load3		

Reorder Buffer

	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					

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

Need 36 more EX cycles for DIV to finish…

Tomasulo With Reorder Buffer: Summary

LD	F6, 34(R2)
LD	F2, 45(R3)
MULT	F0, F2, F4
SUBD	F8, F6, F2
DIVD	F10, F0, F6
ADDD	F6, F8, F2

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

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