

---

# CS152 – Computer Architecture and Engineering

## Lecture 17 – Advanced Pipelining: Tomasulo Algorithm

2003-10-23

Dave Patterson

([www.cs.berkeley.edu/~patterson](http://www.cs.berkeley.edu/~patterson))

[www-inst.eecs.berkeley.edu/~cs152/](http://www-inst.eecs.berkeley.edu/~cs152/)



# Scoreboard Review

---

- HW exploiting ILP (Instruction Level Parallelism)
  - Works when can't know dependence at compile time.
  - Code for one machine runs well on another
- Key idea of Scoreboard: Allow instructions behind stall to proceed (Decode => Issue instruction & read operands)
  - Enables out-of-order execution => out-of-order completion (but in order execution)
  - ID stage checked both for structural & data dependencies
  - Original version didn't handle forwarding.
  - No automatic register renaming = WAW, WAR stalls



# Another Dynamic Algorithm: Tomasulo Algorithm

- For IBM 360/91 about 3 years after CDC 6600 (1966)
- Goal: High Performance without special compilers
- Differences between IBM 360 & CDC 6600 ISA
  - IBM has only 2 register specifiers/instr vs. 3 in CDC 6600
  - IBM has 4 FP registers vs. 8 in CDC 6600
  - IBM has memory-register ops
- Why Study? lead to Alpha 21264, HP 8000, MIPS 10000, Pentium Pro, Pentium 4, PowerPC 604, ...

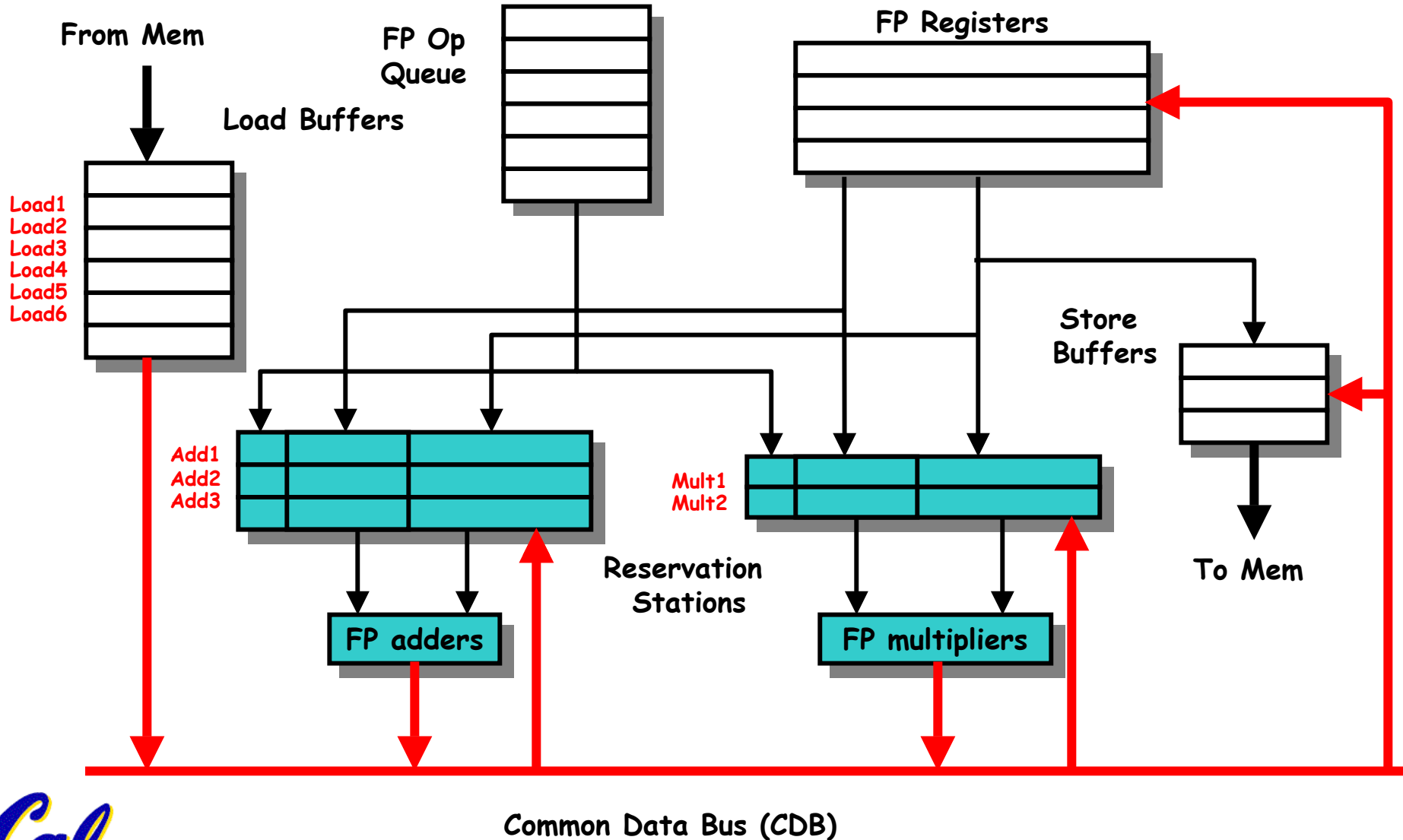


# Tomasulo Algorithm vs. Scoreboard

- Control & buffers **distributed** with Function Units (FU) vs. centralized in scoreboard;
  - FU buffers called “**reservation stations**” (RS);  
have pending operands
- Registers in instructions replaced by values or pointers to reservation stations(RS); called **register renaming** ;
  - avoids WAR, WAW hazards
  - More reservation stations than registers, so can do optimizations compilers can't
- Results to FU from RS, **not through registers**, over **Common Data Bus** that broadcasts results to all FUs
- Load and Stores treated as FUs with RSs as well
- Integer instructions can go past branches,  
allowing FP ops beyond basic block in FP queue



# Tomasulo Organization



# Reservation Station Components

---

**Busy:** Indicates reservation station or FU is busy (*like CDC*)

**Op:** Operation to perform in the unit (+, −, ...) (*like CDC*)

**Vj, Vk:** **Value** of Source operands

- Store buffers has V field, result to be stored
- (CDC scoreboard had 3 register numbers, not their values)

**Qj, Qk:** Reservation stations producing source registers (value to be written) (*like CDC*)

- Note: No register ready flags as in Scoreboard;  
Qj, Qk=0 => **ready** (nothing is writing them)
- Store buffers only have Qi for RS producing result

**Register result status**—Indicates which functional unit will write each register, if one exists. Blank when no pending instructions that will write that register. (*like CDC*)



# Three Stages of Tomasulo Algorithm

---

## 1. Issue—get instruction from FP Op Queue

If reservation station free (no structural hazard), control issues instr & sends operands (renames registers).

## 2. Execution—operate on operands (EX)

When both operands ready then execute;  
if not ready, watch Common Data Bus for result

## 3. Write result—finish execution (WB)

Write on Common Data Bus to **all** awaiting units;  
mark reservation station available

- Normal data bus: data + destination (“go to” bus)
- Common data bus: data + source (“come from” bus)
  - 64 bits of data + 4 bits of Functional Unit source address
  - Write if matches expected Functional Unit (produces result)
  - Does the broadcast



# Detailed Tomasulo Pipeline Control

Instruction status	Wait until	Action or bookkeeping
Issue	Station or buffer empty <i>(No structural hazard)</i>	<pre> RS[r].Busy ← yes; Register[D].Qi ← r; if (Register[S1].Qi ≠ 0) {RS[r].Qj ← Register[S1].Qi} else {RS[r].Vj ← S1; RS[r].Qj ← 0}; if (Register[S2].Qi ≠ 0) {RS[r].Qk ← Register[S2].Qi} else {RS[r].Vk ← S2; RS[r].Qk ← 0}; <i>(Mark RS busy and D to be written by RS If RAW on S1 or S2, copy RS into Q, else copy data into V and set Q to 0)</i> </pre>
Execute	(RS[r].Qj=0) and (RS[r].Qk=0) <i>(No RAW hazard)</i>	None (operands are in Vj and Vk)
Write result	Execution completed at r and CDB available <i>(No structural hazard on CDB)</i>	<pre> RS[r].Busy ← No ∀x (if (Register[x].Qi=r) {Fx ← result; Register[x].Qi ← 0}); ∀x (if (RS[x].Qj=r) {RS[x].Vj ← result; RS[x].Qj ← 0}); ∀x (if (RS[x].Qk=r) {RS[x].Vk ← result; RS[x].Qk ← 0}); ∀x (if (Store[x].Qi=r) {Store[x].V ← result; Store[x].Qi ← 0}); <i>(Mark RS free. For all other RS, if waiting for result, write and reset Q)</i> </pre>

D = destination, S1 and S2 = source register numbers, and r is the reservation station or buffer that D is assigned to. RS is the reservation-station data structure. The value returned by a reservation station or by the load unit is called result. Register is the register data structure (control, not the register file), while Store is the store-buffer data structure.



# Administrivia

---

- Design full cache, but only simulation on Friday 10/24; demo board Friday 10/31
- Thur 11/6: Design Doc for Final Project due
  - Deep pipeline? Superscalar? Out-of-order?
  - Read section 4.2 from CA:AQA 2/e
- Fri 11/14: Demo Project modules
- Wed 11/19: 5:30 PM Midterm 2 in 1 LeConte
- Tues 11/22: Field trip to Xilinx
- CS 152 Project week: 12/1 to 12/5
  - Mon: TA Project demo, Tue: 30 min Presentation, Wed: Processor racing, Fri: Written report



# Tomasulo Example

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>		Busy	Address
			<i>Issue</i>	<i>Comp Result</i>		
LD	F6	34+	R2		Load1	No
LD	F2	45+	R3		Load2	No
MULTD	F0	F2	F4		Load3	No
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

## Reservation Stations:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

## Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
0	<i>FU</i>								



# Tomasulo Example Cycle 1

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Exec	Write	Comp	Result	Busy	Address
LD	F6	34+	R2	1				Yes	34+R2
LD	F2	45+	R3					No	
MULTD	F0	F2	F4					No	
SUBD	F8	F6	F2						
DIVD	F10	F0	F6						
ADDD	F6	F8	F2						

## Reservation Stations:

Time	Name	Busy	Op	<i>S1</i> <i>Vj</i>	<i>S2</i> <i>Vk</i>	<i>RS</i> <i>Qj</i>	<i>RS</i> <i>Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
1				Load1					



# Tomasulo Example Cycle 2

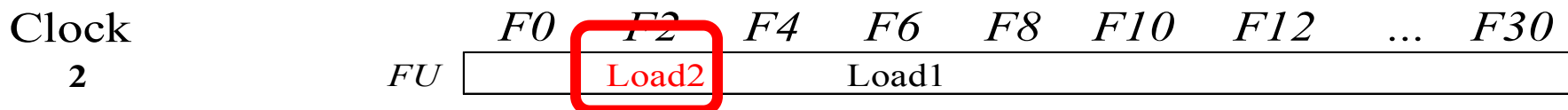
## Instruction status:

Instruction	<i>j</i>	<i>k</i>			Busy	Address
			<i>Issue</i>	<i>Exec Write</i> <i>Comp Result</i>		
LD	F6	34+	R2	1	Yes	34+R2
LD	F2	45+	R3	2	Yes	45+R3
MULTD	F0	F2	F4		No	
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

## Reservation Stations:

Time	Name	Busy	Op	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

## Register result status:



Note: Unlike 6600, can have multiple loads outstanding  
(Assume latency for loads is 2 clock cycles)



# Tomasulo Example Cycle 3

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>		Busy	Address		
			<i>Issue</i>	<i>Comp Result</i>				
LD	F6	34+	R2	1	3	Load1	Yes	34+R2
LD	F2	45+	R3	2		Load2	Yes	45+R3
MULTD	F0	F2	F4	3		Load3	No	
SUBD	F8	F6	F2					
DIVD	F10	F0	F6					
ADDD	F6	F8	F2					

## Reservation Stations:

Time	Name	Busy	Op	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		R(F4)	Load2	
	Mult2	No					

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
3	Mult1	Load2			Load1				

- Note: registers names are removed ("renamed") in Reservation Stations; MULT issued vs. scoreboard
- Load1 completing; what is waiting for Load1?



# Tomasulo Example Cycle 4

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Exec Comp	Write Result	Load	Busy	Address
LD	F6	34+	R2	1	3	4	No	
LD	F2	45+	R3	2	4		Yes	45+R3
MULTD	F0	F2	F4	3			No	
SUBD	F8	F6	F2	4			No	
DIVD	F10	F0	F6				No	
ADDD	F6	F8	F2				No	

## Reservation Stations:

Time	Name	Busy	Op	<i>S1</i> <i>Vj</i>	<i>S2</i> <i>Vk</i>	<i>RS</i> <i>Qj</i>	<i>RS</i> <i>Qk</i>
Add1		Yes	SUBD	M(A1)			Load2
Add2		No					
Add3		No					
Mult1		Yes	MULTD		R(F4)	Load2	
Mult2		No					

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
4	Mult1	Load2		M(A1)	Add1				

• Load2 completing; what is waiting for Load2?



# Tomasulo Example Cycle 5

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4				
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2					

## Reservation Stations:

Time	Name	Busy	Op	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
2	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	No					
	Add3	No					
10	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
5	Mult1	M(A2)		M(A1)	Add1	Mult2			



# Tomasulo Example Cycle 6

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4				
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6				

## Reservation Stations:

Time	Name	Busy	Op	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
1	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	Yes	ADDD		M(A2)	Add1	
	Add3	No					
9	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
6	FU	Mult1	M(A2)		Add2	Add1	Mult2		

• Issue ADDD here vs. scoreboard?



# Tomasulo Example Cycle 7

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Exec Comp	Write Result	Load	Busy	Address
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7			
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6				

## Reservation Stations:

Time	Name	Busy	Op	<i>S1</i> <i>Vj</i>	<i>S2</i> <i>Vk</i>	<i>RS</i> <i>Qj</i>	<i>RS</i> <i>Qk</i>
0	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	Yes	ADDD		M(A2)	Add1	
	Add3	No					
8	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
7	<i>FU</i>	Mult1	M(A2)		Add2	Add1	Mult2		

- Add1 completing; what is waiting for it?



# Tomasulo Example Cycle 8

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6				

## Reservation Stations:

Time	Name	Busy	Op	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
	Add1	No					
2	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
7	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
8	FU								
	Mult1	M(A2)		Add2	(M-M)	Mult2			



# Tomasulo Example Cycle 9

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6				

## Reservation Stations:

Time	Name	Busy	<i>Op</i>	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
	Add1	No					
1	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
6	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
9	FU								
	Mult1	M(A2)		Add2	(M-M)	Mult2			



# Tomasulo Example Cycle 10

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10			

## Reservation Stations:

Time	Name	Busy	Op	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
	Add1	No					
0	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
5	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
10	FU								
	Mult1	M(A2)		Add2	(M-M)	Mult2			

- Add2 completing; what is waiting for it?



# Tomasulo Example Cycle 11

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Exec	Write	Result	Busy	Address
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

## Reservation Stations:

Time	Name	Busy	Op	<i>S1</i> <i>Vj</i>	<i>S2</i> <i>Vk</i>	<i>RS</i> <i>Qj</i>	<i>RS</i> <i>Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
11	FU	Mult1	M(A2)		(M-M+N)	(M-M)	Mult2		

- Write result of ADDD here vs. scoreboard?
- All quick instructions complete by this cycle!



# Tomasulo Example Cycle 12

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

## Reservation Stations:

Time	Name	Busy	Op	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
3	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
12	FU								
	Mult1	M(A2)		(M-M+N	(M-M)	Mult2			



# Tomasulo Example Cycle 13

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Exec	Write	Busy	Address
				Comp	Result		
LD	F6	34+	R2	1	3	4	Load1
LD	F2	45+	R3	2	4	5	Load2
MULTD	F0	F2	F4	3			Load3
SUBD	F8	F6	F2	4	7	8	
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	6	10	11	

## Reservation Stations:

Time	Name	Busy	Op	<i>S1</i> <i>Vj</i>	<i>S2</i> <i>Vk</i>	<i>RS</i> <i>Qj</i>	<i>RS</i> <i>Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
2	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
13	Mult1	M(A2)		(M-M+N)	(M-M)	Mult2			



# Tomasulo Example Cycle 14

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

## Reservation Stations:

Time	Name	Busy	<i>Op</i>	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
1	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
14	Mult1	M(A2)		(M-M+N)	(M-M)	Mult2			



# Tomasulo Example Cycle 15

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3	15		Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

## Reservation Stations:

Time	Name	Busy	<i>Op</i>	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
0	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
15	<i>FU</i>	Mult1	M(A2)		(M-M+N)	(M-M)	Mult2		



# Tomasulo Example Cycle 16

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Exec	Write	Busy	Address	
				Comp	Result			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3	15	16	Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

## Reservation Stations:

Time	Name	Busy	Op	<i>S1</i> <i>Vj</i>	<i>S2</i> <i>Vk</i>	<i>RS</i> <i>Qj</i>	<i>RS</i> <i>Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
40	Mult2	Yes	DIVD	M*F4	M(A1)		

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
16	FU	M*F4	M(A2)		(M-M+N	(M-M)	Mult2		



---

Faster than light computation  
(skip a couple of cycles)



# Tomasulo Example Cycle 55

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3	15	16	Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

## Reservation Stations:

Time	Name	Busy	<i>Op</i>	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
1	Mult2	Yes	DIVD	M*F4	M(A1)		

## Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
55	FU	M*F4	M(A2)		(M-M+N	(M-M)	Mult2		



# Tomasulo Example Cycle 56

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Exec	Write	Result	Busy	Address
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3	15	16	Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5	56			
ADDD	F6	F8	F2	6	10	11		

## Reservation Stations:

Time	Name	Busy	Op	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
0	Mult2	Yes	DIVD	M*F4	M(A1)		

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
56	M*F4	M(A2)		(M-M+N	(M-M)	Mult2			

- Mult2 is completing; what is waiting for it?



# Tomasulo Example Cycle 57

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Exec Comp	Write Result	Busy	Address
LD	F6	34+	R2	1	3	4	Load1
LD	F2	45+	R3	2	4	5	Load2
MULTD	F0	F2	F4	3	15	16	Load3
SUBD	F8	F6	F2	4	7	8	
DIVD	F10	F0	F6	5	56	57	
ADDD	F6	F8	F2	6	10	11	

## Reservation Stations:

Time	Name	Busy	Op	<i>S1</i> <i>Vj</i>	<i>S2</i> <i>Vk</i>	<i>RS</i> <i>Qj</i>	<i>RS</i> <i>Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
0	Mult2	Yes	DIVD	M*F4	M(A1)		

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
56	M*F4	M(A2)		(M-M+N	(M-M)	Mult2			

- Once again: In-order issue, out-of-order execution and completion.



# Compare to Scoreboard Cycle 62

*Instruction status:*

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>			<i>Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4	1	3	4
LD	F2	45+	R3	5	6	7	8	2	4	5
MULTD	F0	F2	F4	6	9	19	20	3	15	16
SUBD	F8	F6	F2	7	9	11	12	4	7	8
DIVD	F10	F0	F6	8	21	61	62	5	56	57
ADDD	F6	F8	F2	13	14	16	22	6	10	11

- Why take longer on scoreboard/6600?
  - Structural Hazards (multiple load)
  - WAW, WAR control hazards vs. renaming registers
  - Lack of forwarding (must write and read register)
  - 4 steps vs. 3 steps of control



# Tomasulo v. Scoreboard (IBM 360/91 v. CDC 6600)

Pipelined Functional Units (6 load, 3 store, 3 +, 2 x/÷) ÷)	Multiple Functional Units (1 load/store, 1 +, 2 x, 1 ÷)
window size: ~ 14 instructions	~ 5 instructions
No issue on structural hazard	same
WAR: renaming avoids	stall completion
WAW: renaming avoids	stall issue
Broadcast results from FU	Write/read registers
Control: reservation stations	central scoreboard



# Tomasulo Analysis

---

- Complexity
  - delays of 360/91, MIPS 10000, IBM 620, Alpha 21264, ...
- Many associative stores (CDB) at high speed
- Performance limited by Common Data Bus
  - Multiple CDBs => more FU logic for parallel associative stores



# PRS: State Example 2

What is Instruction Status at end of Clock 5?

*Instruction status:*

Instruction	<i>j</i>	<i>k</i>
<b>LD</b> <b>F2</b> <b>0</b> <b>R1</b>		
<b>MULTD</b> <b>F4</b> <b>F2</b> <b>F0</b>		
<b>LD</b> <b>F6</b> <b>0</b> <b>R2</b>		
<b>ADDD</b> <b>F6</b> <b>F4</b> <b>F6</b>		
<b>SD</b> <b>F6</b> <b>R2</b>		

1)	2)	3)	4)	5)	6)
Exec. Complete	Exec. Complete	Wrote Result	Wrote Result	Wrote Result	none
Issued	Read Operands	Read Operands	Read Operands	Read Operands	done
Not Issued	Issued	Issued	Exec. Complete	Exec. Complete	etc
Not Issued	Not Issued	Not Issued	Issued	Issued	above
Not Issued	Not Issued	Not Issued	Not Issued	Issued	



# PRS: Tomasulo Example 2

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec Comp</i>	<i>Write Result</i>		Busy	Address	RS?
LD	F2	0	R1	1		Load1	No		
MULTD	F4	F2	F0			Load2	No		
LD	F6	0	R2			Load3	No		
ADDD	F6	F4	F6			Store1	No		
SD		F6	R2			Store2	No		
						Store3	No		

## Reservation Stations:

Time	Name	Busy	Op	<i>S1</i> <i>Vj</i>	<i>S2</i> <i>Vk</i>	<i>RS</i> <i>Qj</i>	<i>RS</i> <i>Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

## Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	...
0	<i>FU</i>						



# PRS: State Example 2

What is Instruction Status at end of Clock 10?

*Instruction status:*

Instruction		<i>j</i>	<i>k</i>
LD	F2	0	R1
MULTD	F4	F2	F0
LD	F6	0	R2
ADDD	F6	F4	F6
SD		F6	R2

	1)	2)	3)	4)	5)
LD	Wrote Result	Wrote Result	Wrote Result	Wrote Result	Wrote Result
MULTD	Read Operands	Read Operands	Read Operands	Read Operands	Read Operands
LD	Exec. Complete	Wrote Result	Wrote Result	Wrote Result	Wrote Result
ADDD	Not Issued	Issued	Issued	Read Operands	Read Operands
SD	Not Issued	Not Issued	Issued	Issued	Read Operands



# PRS: State Example 2

What is Instruction Status at end of Clock 17?

*Instruction status:*

Instruction	<i>j</i>	<i>k</i>
LD F2 0 R1		
MULTD F4 F2 F0		
LD F6 0 R2		
ADDD F6 F4 F6		
SD F6 R2		

1)	2)	3)	4)	5)
Wrote Result	Wrote Result	Wrote Result	Wrote Result	Wrote Result
Exec. Complete	Wrote Result	Wrote Result	Wrote Result	Wrote Result
Wrote Result	Wrote Result	Wrote Result	Wrote Result	Wrote Result
Issued	Issued	Read Operands	Exec. Complete	Wrote Result
Issued	Issued	Issued	Issued	Read Operands



# Summary

---

- Reservations stations: 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)
- Helps cache misses as well
- Lasting Contributions
  - Dynamic scheduling
  - Register renaming
  - Load/store disambiguation
- 360/91 descendants are Pentium 4; PowerPC 604; MIPS R10000; HP-PA 8000; Alpha 21264

