

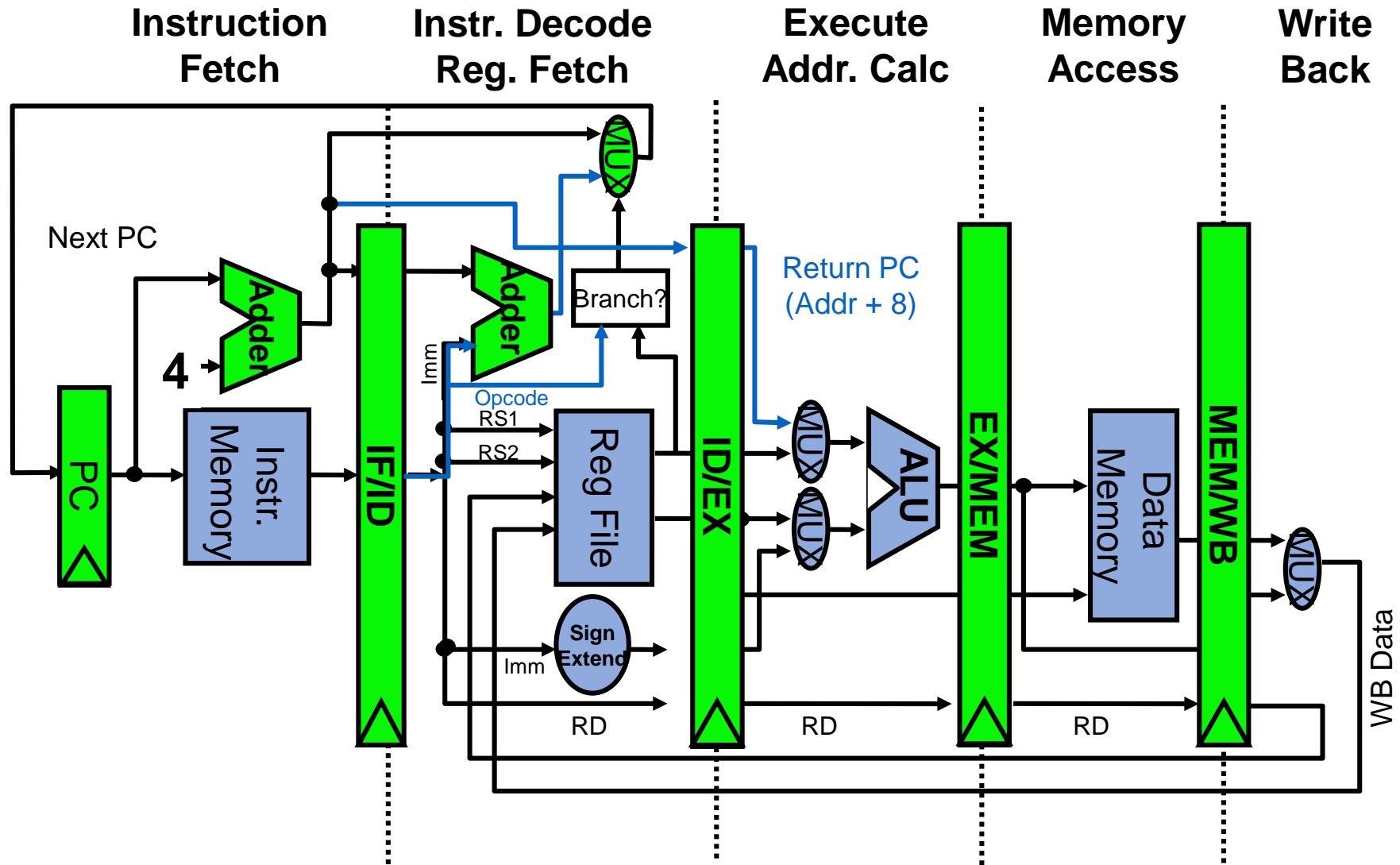
# **CS425**

# **Computer Systems Architecture**

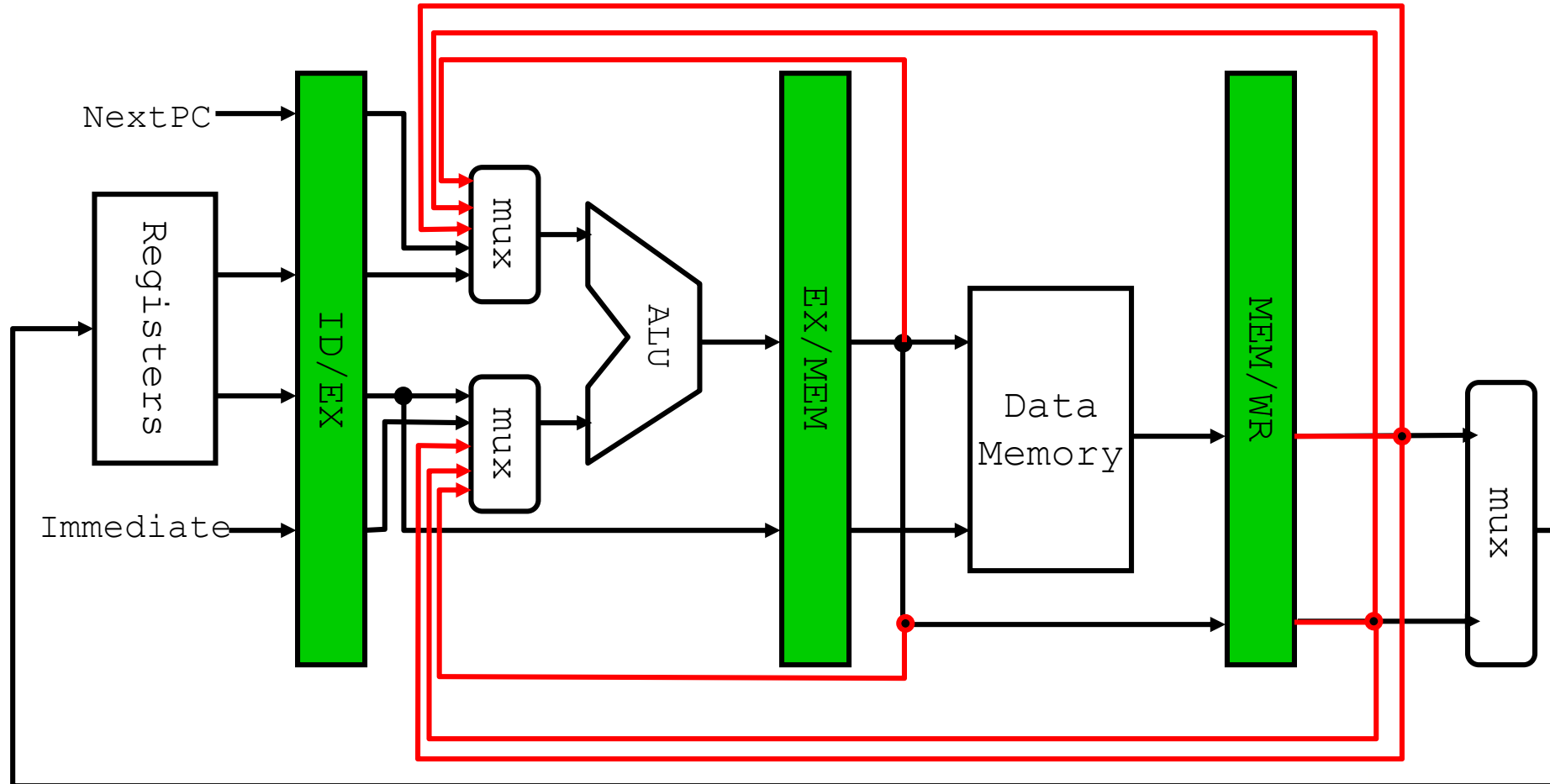
**Fall 2019**

**Dynamic Instruction Scheduling:  
Tomasulo**

# DLX Processor



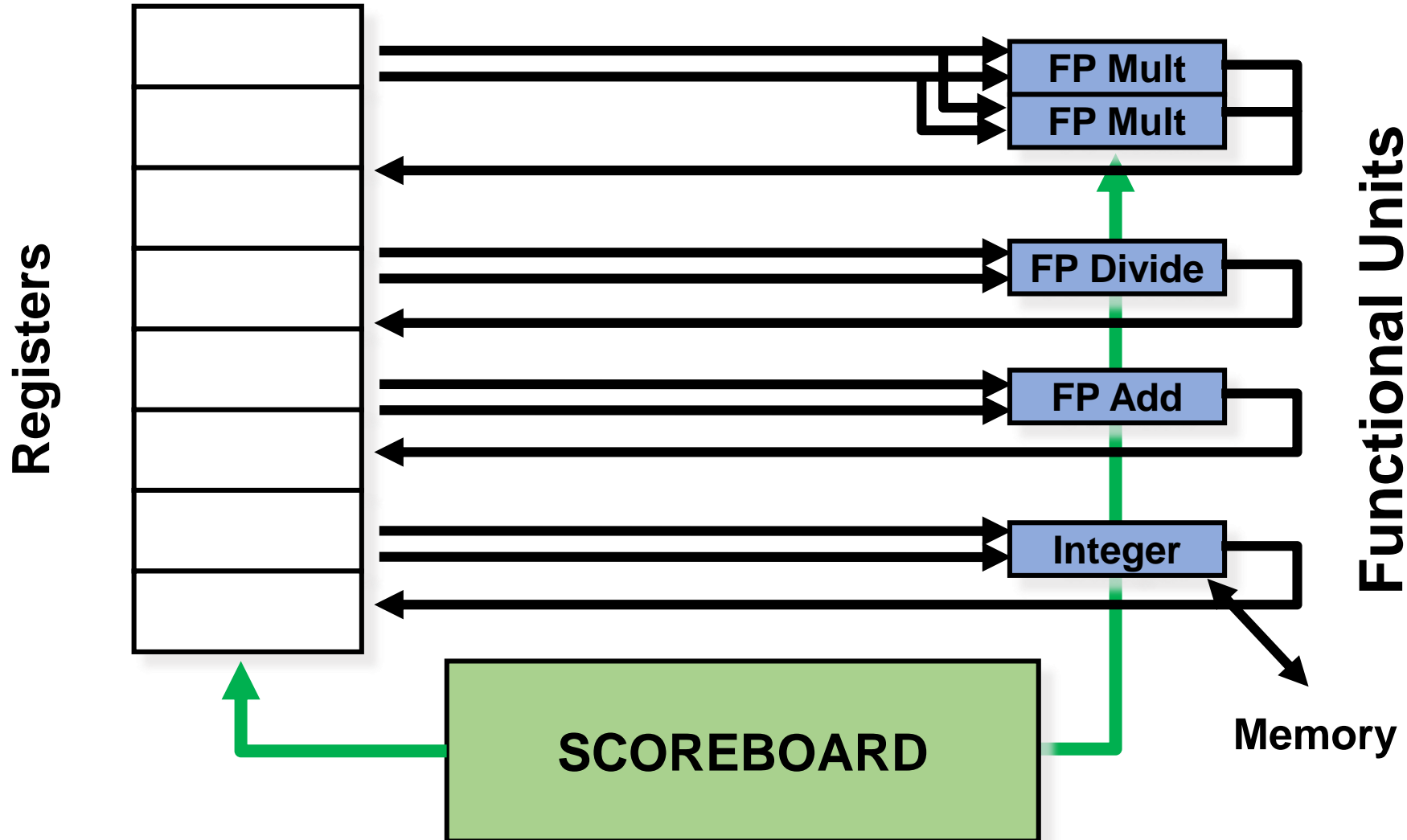
# HW Change for Forwarding



**What circuit detects and resolves this hazard?**

**Why we need forwarding lines for both inputs of the ALU?**

# Scoreboard Architecture (CDC 6600)



Instruction to scoreboard  $\Rightarrow$   
data dependences  $\Rightarrow$   
hazard detection and  
resolution centralized

**Scoreboard:** decides when  
the instruction can **execute**  
and when it can **write its  
result**

# CDC 6600 Scoreboard

- Speedup 1.7 for FORTRAN programs; 2.5 by hand (outdated measurements)
- Limitations of 6600 scoreboard:
  - No forwarding hardware
  - Limited to instructions in basic block (small window)
  - Small number of functional units (structural hazards), especially integer/load store units
    - Do not issue on structural hazards
  - Wait for WAR hazards
  - Prevent WAW hazards

# Another Dynamic Algorithm: Tomasulo

- IBM 360/91 3 years later than CDC 6600 (1966)
- **Goal:** High Performance without special compilers
- Differences between IBM 360 & CDC 6600 ISA
  - IBM has 4 FP registers vs. 8 in CDC 6600
  - IBM has memory-register ops
- Why study this?
  - lead to Alpha 21264, HP 8000, MIPS 10000, Pentium II, PowerPC 604...

# Register Renaming

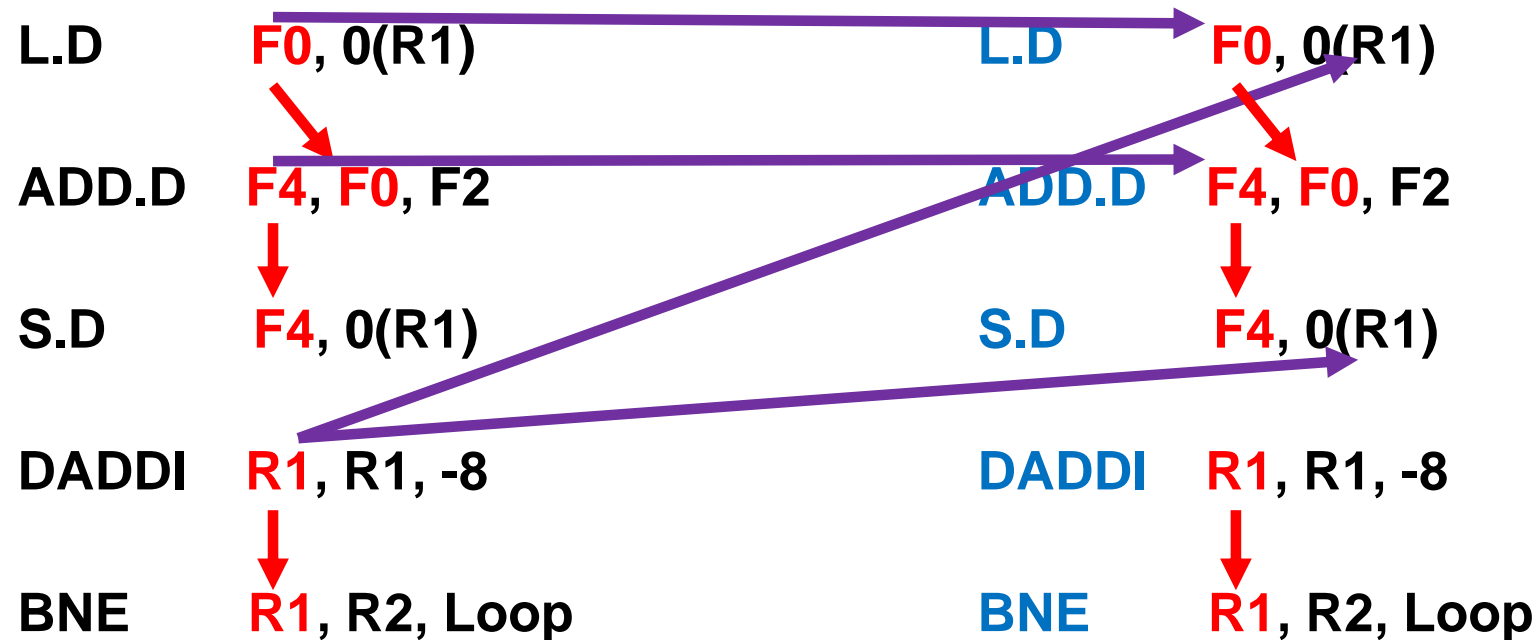


- What happens with branches?
- Tomasulo can handle renaming across branches

# Dependencies Between Instructions

- **(True) Data Dependences**: two instructions are data dependent when there is a chain of RAW hazards between them.

Loop:



*What happens in next iteration?*



# Scoreboard Example: Cycle 17

## Instruction status:

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

**WAR hazard!**

**Why not write ADDD result?**

## Functional unit status:

Time	Name	Busy	Op	dest <i>Fi</i>	<i>S1</i> <i>Fj</i>	<i>S2</i> <i>Fk</i>	<i>FU</i> <i>Qj</i>	<i>FU</i> <i>Qk</i>	<i>Fj?</i> <i>Rj</i>	<i>Fk?</i> <i>Rk</i>
	Integer	No								
2	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

## Register result status:

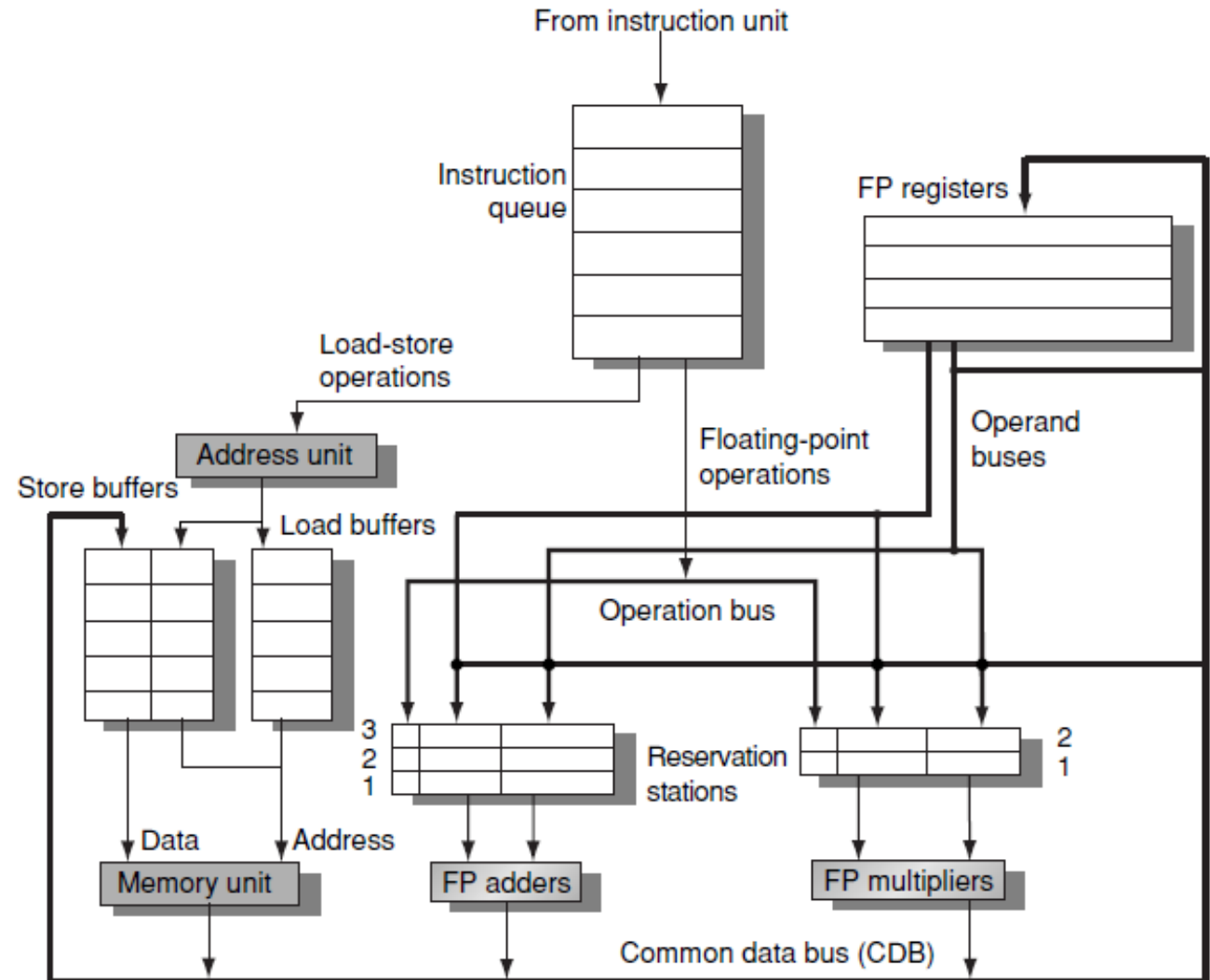
Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
17									
<i>FU</i>	Mult1			Add		Divide			

# Tomasulo Algorithm vs. Scoreboard

- FU buffers called “**reservation stations (RS)**” and hold the register/variable values needed for execution
- The source registers are replaced with the actual values or pointers to the RSs that will produce the values. This is effectively **register renaming**.
  - avoids WAR, WAW hazards
  - more RSs than real registers (can do optimizations compilers can't !)
- Results arrive to the FUs from the RSs, **not via register file**, but over the **Common Data Bus** that broadcasts results to all FUs: bypassing of results
- Load and Stores use separate FUs with RSs as well
- Integer instructions can go past branches, allowing FP ops beyond basic block in FP queue
- Control logic and buffers are **distributed** with the Functional Units (FUs) vs. **centralized** in Scoreboard

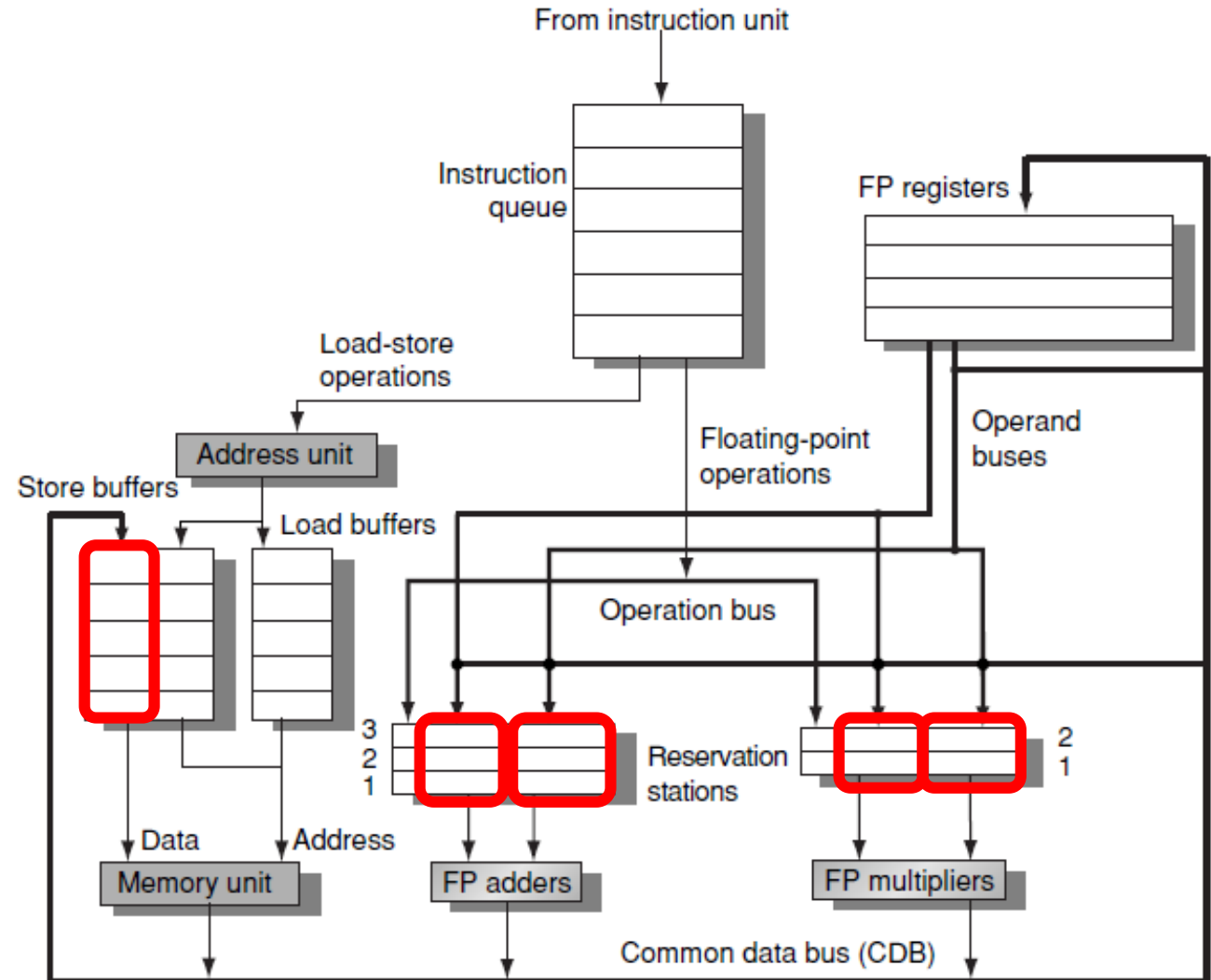
# Tomasulo Organization (1/3)

- Read operands from RF or CDB on issue
- CDB: no need for multi-ported RF
- Multiple instructions maybe ready for Write Result or Execution:
  - source of structural hazards if FUs or CDB is not enough
- Hazards on memory (effective address, next lectures)



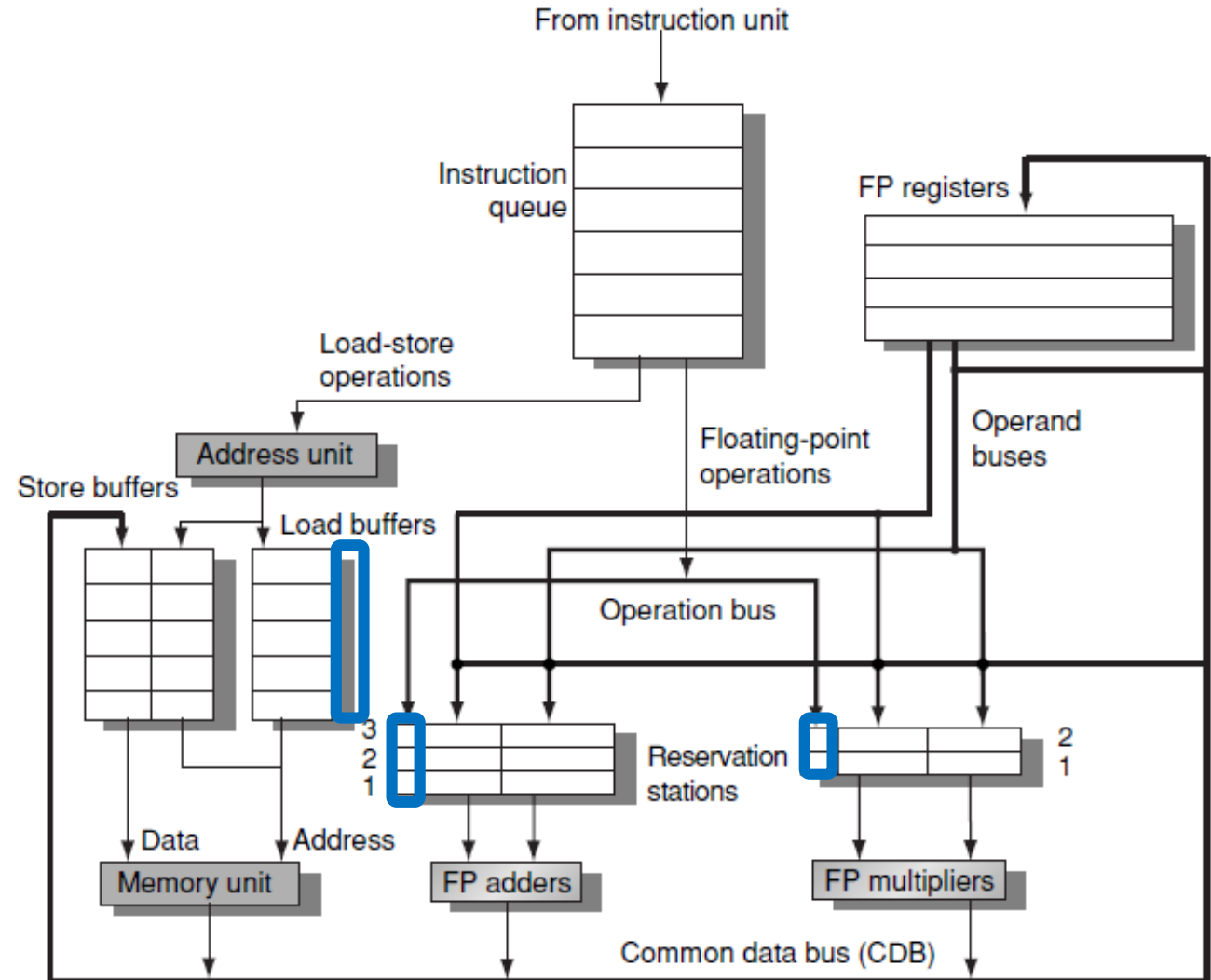
# Tomasulo Organization (2/3)

- Register values or tags of FU that will produce the results



# Tomasulo Organization (3/3)

- Ten tags: 4-bit tags
  - 3 Adders
  - 2 Multipliers
  - 5 Load buffers



# Three Stages of the Tomasulo Algorithm

- **Issue (dispatch):** get a new instruction from the instruction queue
  - If there is a vacant reservation station (no structural hazard) then issue instruction and send operand values or keep track of the FU/RS that will produce the operand (register renaming)
- **Execution:** execute on the functional unit (FU)
  - when both operands are ready in the RS start execution
  - when not ready watch the Common Data Bus for the expected result
- **Write result (commit):** end of execution (WB)
  - write the result on the Common Data Bus so that all waiting FUs/RSs get the result, write to register file, mark reservation station available

# Common Data Bus

- Typical data bus: data + destination (“go to” bus)
- Common data bus: data + source/res. station tag (“come from” bus).
  - masters send to all slaves
  - does arbitration and the **broadcast**
  - 64 bits of data + 4 bits of Functional Unit source address
  - write result if it matches the expected Functional Unit (result producer)

# Tomasulo Data Structures

- **Instruction status:** the stage of each instruction
- **Reservation station status:**
  - **Busy:** marks if the RS is busy or not
  - **Op:** the operation to be executed (e.g., + or –)
  - **V<sub>j</sub>, V<sub>k</sub>:** the value of the source register
    - Store buffers have a single V field, value to store
  - **Q<sub>j</sub>, Q<sub>k</sub>:** the reservation stations which produce the source registers (value to be written)
    - Note: No ready flags as in Scoreboard; Q<sub>j</sub>=0, Q<sub>k</sub>=0 means ready
    - Store buffers only have Q<sub>i</sub> for RS producing result
- **Register Result Status:** show which FU/RS will write each register. Empty when no pending instructions write this register.



# Tomasulo Example

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec</i>	<i>Write</i>	<i>Comp</i>	<i>Result</i>	Busy	Address
L.D	F6	34+	R2					Load1	No
L.D	F2	45+	R3					Load2	No
MULT.D	F0	F2	F4					Load3	No
SUB.D	F8	F6	F2						
DIV.D	F10	F0	F6						
ADD.D	F6	F8	F2						

## Reservation Stations:

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

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
0	FU								

# Tomasulo Example: Cycle 1

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec</i>	<i>Write</i>
			<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	
LD	F2	45+	R3		
MULTD	F0	F2	F4		
SUBD	F8	F6	F2		
DIVD	F10	F0	F6		
ADDD	F6	F8	F2		

	Busy	Address
2 Load1	Yes	34+R2
Load2	No	
Load3	No	

## 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>
1				Load1					

# Tomasulo Example: Cycle 2

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec</i>	<i>Write</i>	<i>Comp</i>	<i>Result</i>
LD	F6	34+	R2	1			
LD	F2	45+	R3	2			
MULTD	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

	Busy	Address
1 Load1	Yes	34+R2
2 Load2	Yes	45+R3
Load3	No	

**Note: we can have multiple loads outstanding**

## Reservation Stations:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<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
2		Load2							
				Load1					

# Tomasulo Example: Cycle 3

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec Comp</i>	<i>Write Result</i>	Busy	Address
LD	F6	34+	R2	1	3	0 Load1	Yes 34+R2
LD	F2	45+	R3	2		1 Load2	Yes 45+R3
MULTD	F0	F2	F4	3		Load3	No
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Load1 completes

## Reservation Stations:

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

**Note:** registers names are removed (“renamed”) in Reservation Stations; MULTD issued vs. scoreboard

## Register result status:

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

# Tomasulo Example: Cycle 4

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec Comp</i>	<i>Write Result</i>	Busy	Address
LD	F6	34+	R2	1	3	4	Load1
LD	F2	45+	R3	2	4		0 Load2
MULTD	F0	F2	F4	3			Load3
SUBD	F8	F6	F2	4			
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Load2 completes

## 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	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
4	FU								
	Mult1	Load2			Add1				
Reg File	M(A1)								

# Tomasulo Example: Cycle 5

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec Comp</i>	<i>Write Result</i>	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				
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2					

## Reservation Stations:

Time	Name	Busy	Op	<i>S1 Vj</i>	<i>S2 Vk</i>	<i>RS Qj</i>	<i>RS 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	FU								
	Mult1				Add1	Mult2			
Reg File	M(A2)			M(A1)					

# Tomasulo Example: Cycle 6

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec Comp</i>	<i>Write Result</i>	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				
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6				

## Reservation Stations:

Time	Name	Busy	Op	<i>S1 Vj</i>	<i>S2 Vk</i>	<i>RS Qj</i>	<i>RS 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	

Issue ADDD here vs. scoreboard?

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
6	FU								
	Mult1			Add2	Add1	Mult2			
Reg File	M(A2)			M(A1)					

# Tomasulo Example: Cycle 7

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec Comp</i>	<i>Write Result</i>	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 Vj</i>	<i>S2 Vk</i>	<i>RS Qj</i>	<i>RS 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	

Add1 completes;  
what is waiting for it?

## 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	FU								
	Mult1			Add2	Add1	Mult2			
	Reg File								
		M(A2)		M(A1)					



# Tomasulo Example: Cycle 8

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec Comp</i>	<i>Write Result</i>	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				

## 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					
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			Add2		Mult2			
	Reg File								
		M(A2)		M(A1)	(M-M)				

# Tomasulo Example: Cycle 9

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec Comp</i>	<i>Write Result</i>	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				

## 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					
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			Add2		Mult2			
Reg File	M(A2)		M(A1)			(M-M)			

# Tomasulo Example: Cycle 10

## Instruction status:

Instruction	$j$	$k$	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			Load3
SUBD	F8	F6	F2	4	7	8	
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	6	10		

## Reservation Stations:

Time	Name	Busy	Op	$S1$ $V_j$	$S2$ $V_k$	$RS$ $Q_j$	$RS$ $Q_k$
	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	

Add2 completes;  
what is waiting for it?

## Register result status:

Clock	$F0$	$F2$	$F4$	$F6$	$F8$	$F10$	$F12$	...	$F30$
10	FU								
	Mult1			Add2		Mult2			
	Reg File								
		M(A2)		M(A1)	(M-M)				

# Tomasulo Example: Cycle 11

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec Comp</i>	<i>Write Result</i>	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 Vj</i>	<i>S2 Vk</i>	<i>RS Qj</i>	<i>RS Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Write result of ADDD here vs. scoreboard?

All quick instructions complete in this cycle!

## 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>
11	FU								
	Mult1						Mult2		
Reg File	M(A2)		(M-M+N)			(M-M)			

# Tomasulo Example: Cycle 12

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec Comp</i>	<i>Write Result</i>	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 Vj</i>	<i>S2 Vk</i>	<i>RS Qj</i>	<i>RS 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						Mult2		
Reg File	M(A2)		(M-M+M)		(M-M)				

# Tomasulo Example: Cycle 13

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec Comp</i>	<i>Write Result</i>	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					
2	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>
13	FU								
	Mult1						Mult2		
Reg File	M(A2)		(M-M+M)		(M-M)				

# Tomasulo Example: Cycle 14

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec Comp</i>	<i>Write Result</i>	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 Vj</i>	<i>S2 Vk</i>	<i>RS Qj</i>	<i>RS 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	FU								
	Mult1						Mult2		
Reg File	M(A2)		(M-M+M)		(M-M)				

# Tomasulo Example: Cycle 15

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec Comp</i>	<i>Write Result</i>	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		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 Vj</i>	<i>S2 Vk</i>	<i>RS Qj</i>	<i>RS 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	FU								
	Mult1						Mult2		
Reg File	M(A2)		(M-M+M)		(M-M)				



# Tomasulo Example: Cycle 16

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec Comp</i>	<i>Write Result</i>	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				
ADDD	F6	F8	F2	6	10	11		

## Reservation Stations:

Time	Name	Busy	Op	<i>S1 Vj</i>	<i>S2 Vk</i>	<i>RS Qj</i>	<i>RS Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
40	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>
16	FU								
	Reg File								
	M*F4	M(A2)	(M-M+M)	(M-M)					

**After a few clock cycles...**

# Tomasulo Example: Cycle 55

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec Comp</i>	<i>Write Result</i>	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				
ADDD	F6	F8	F2	6	10	11		

## Reservation Stations:

Time	Name	Busy	Op	<i>S1 Vj</i>	<i>S2 Vk</i>	<i>RS Qj</i>	<i>RS 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 <span style="float: right;">Mult2</span>								
Reg File	M*F4	M(A2)	(M-M+M)	(M-M)					

# Tomasulo Example: Cycle 56

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec Comp</i>	<i>Write Result</i>	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 Vj</i>	<i>S2 Vk</i>	<i>RS Qj</i>	<i>RS Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
0	Mult2	Yes	DIVD	M*F4	M(A1)		

Mult2 is completing;  
what is waiting for it?

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
56	FU								
	Reg File								
	M*F4	M(A2)		(M-M+M)	(M-M)				

# 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	S1 V <sub>j</sub>	S2 V <sub>k</sub>	RS Q <sub>j</sub>	RS Q <sub>k</sub>
Add1		No					
Add2		No					
Add3		No					
Mult1		No					
Mult2		Yes	DIVD	M*F4	M(A1)		

In-order issue?

Out-of-order execute?

Out-of-order commit?

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
56	FU								
	Result								
Reg File	M*F4	M(A2)	(M-M+M)	(M-M)					

# 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 did it take longer on scoreboard/6600?
  - Structural Hazards
  - Lack of forwarding
  - WAR, WAW hazards cause stalls

# Tomasulo Loop Example

```
while (R1 > 0) { MEM[R1] = MEM[R1] * F2; R1 -= 8; }
```

```
Loop:      LD          F0      0      R1
           MULTD       F4      F0      F2
           S.D         F4      0      R1
           SUBI        R1      R1      #8
           BNEZ        R1      Loop
```

- Assume MULTD takes 4 clocks and R1 = 80 at start
- Assume **first load takes 8 clocks** (cache miss), second load takes 1 clock (hit) and stores take 1 clock (hit)
- We will show clocks for SUBI, BNEZ
- Reality: integer instructions move ahead/faster

# Loop Example

## Instruction status:

<i>ITER</i>	<i>Instruction</i>	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>CompResult</i>	<i>Exec</i>	<i>Write</i>	<i>Busy</i>	<i>Addr</i>	<i>Fu</i>
1	LD	F0	0	R1		Load1	No			
1	MULTD	F4	F0	F2		Load2	No			
1	SD	F4	0	R1		Load3	No			
2	LD	F0	0	R1		Store1	No			
2	MULTD	F4	F0	F2		Store2	No			
2	SD	F4	0	R1		Store3	No			

## Reservation Stations:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	<i>Code:</i>
	Add1	No						LD
	Add2	No						MULTD
	Add3	No						SD
	Mult1	No						SUBI
	Mult2	No						BNEZ

## Register result status

<i>Clock</i>	<i>R1</i>	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	<i>...</i>	<i>F30</i>
0	80	<i>Fu</i>								



# Loop Example: Cycle 1

## Instruction status:

ITER	Instruction	<i>j</i>	<i>k</i>	Exec Write		Issue	CompResult	<sup>8</sup> Busy	Addr	Fu
				<i>S1</i>	<i>S2</i>					
1	LD	F0	0	R1		1		Yes	80	
1	MULTD	F4	F0	F2				No		
1	SD	F4	0	R1				No		
2	LD	F0	0	R1				No		
2	MULTD	F4	F0	F2				No		
2	SD	F4	0	R1				No		

## Reservation Stations:

Time	Name	Busy	Op	V <sub>j</sub>	V <sub>k</sub>	Q <sub>j</sub>	Q <sub>k</sub>	Code:
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2
	Add3	No						SD F4 0 R1
	Mult1	No						SUBI R1 R1 #8
	Mult2	No						BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30	
1	80	Fu Load1									

# Loop Example: Cycle 2

## Instruction status:

ITER	Instruction	<i>j</i>	<i>k</i>	Exec Write		Issue	Comp	Result	<sup>7</sup> Busy	Addr	Fu
				<i>S1</i>	<i>S2</i>						
1	LD	F0	0	R1		1		Load1	Yes	80	
1	MULTD	F4	F0	F2		2		Load2	No		
1	SD	F4	0	R1				Load3	No		
2	LD	F0	0	R1				Store1	No		
2	MULTD	F4	F0	F2				Store2	No		
2	SD	F4	0	R1				Store3	No		

## Reservation Stations:

Time	Name	Busy	Op	V <sub>j</sub>	V <sub>k</sub>	Q <sub>j</sub>	Q <sub>k</sub>	Code:
	Add1	No						LD
	Add2	No						MULTD
	Add3	No						SD
	Mult1	Yes	Multd		R(F2)	Load1		SUBI
	Mult2	No						BNEZ

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
2	80	Fu	Load1	Mult1						

# Loop Example: Cycle 3

## Instruction status:

ITER	Instruction	<i>j</i>	<i>k</i>	Issue	Comp	Result	Busy	Addr	<i>Fu</i>
1	LD	F0	0	R1	1		Yes	80	
1	MULTD	F4	F0	F2	2		No		
1	SD	F4	0	R1	3		No		
2	LD	F0	0	R1			Yes	80	Mult1
2	MULTD	F4	F0	F2			No		
2	SD	F4	0	R1			No		

## Reservation Stations:

Time	Name	Busy	Op	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	Code:
	Add1	No						LD
	Add2	No						MULTD
	Add3	No						SD
	Mult1	Yes	Multd		R(F2)	Load1		SUBI
	Mult2	No						BNEZ

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
3	80	<i>Fu</i> Load1		Mult1						

Implicit renaming sets up “data-flow” graph

# Loop Example: Cycle 4

## Instruction status:

ITER	Instruction	j	k	Exec Write		Issue	Comp	Result	5	Addr	Fu
				Busy	Addr						
1	LD	F0	0	R1		1			Yes	80	
1	MULTD	F4	F0	F2		2			No		
1	SD	F4	0	R1		3			No		
2	LD	F0	0	R1					Yes	80	Mult1
2	MULTD	F4	F0	F2					No		
2	SD	F4	0	R1					No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	S1	S2	RS	Qj	Qk	Code:
	Add1	No									LD
	Add2	No									MULTD
	Add3	No									SD
	Mult1	Yes	Multd			R(F2)	Load1				SUBI
	Mult2	No									BNEZ

**Dispatching  
SUBI Instruction**

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
4	80	Fu	Load1	Mult1						

# Loop Example: Cycle 5

## Instruction status:

ITER	Instruction	j	k	Exec Write		Issue	Comp	Result	4 Busy	Addr	Fu
				S1	S2						
1	LD	F0	0	R1		1			Yes	80	
1	MULTD	F4	F0	F2		2			No		
1	SD	F4	0	R1		3			No		
2	LD	F0	0	R1					Yes	80	Mult1
2	MULTD	F4	F0	F2					No		
2	SD	F4	0	R1					No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2
	Add3	No						SD F4 0 R1
	Mult1	Yes	Multd		R(F2)	Load1		SUBI R1 R1 #8
	Mult2	No						BNEZ R1 Loop

**And, BNEZ instruction**

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
5	72	Fu	Load1	Mult1						

# Loop Example: Cycle 6

## Instruction status:

ITER	Instruction	j	k	Exec Write		Issue	Comp	Result	Busy	Addr	Fu
				Issue	Comp						
1	LD	F0	0	R1		1		Load1	Yes	80	
1	MULTD	F4	F0	F2		2		Load2	Yes	72	
1	SD	F4	0	R1		3		Load3	No		
2	LD	F0	0	R1		6		Store1	Yes	80	Mult1
2	MULTD	F4	F0	F2				Store2	No		
2	SD	F4	0	R1				Store3	No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:	Fu	Addr	Fu
	Add1	No						LD	F0	0	R1
	Add2	No						MULTD	F4	F0	F2
	Add3	No						SD	F4	0	R1
	Mult1	Yes	Multd		R(F2)	Load1		SUBI	R1	R1	#8
	Mult2	No						BNEZ	R1	Loop	

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
6	72	Fu	Load2	Mult1						

**Note: F0 never sees Load from location 80**

**Load2 waits for cache-line behind Load1**

# Loop Example: Cycle 7

## Instruction status:

ITER	Instruction	j	k	Exec Write		Issue	Comp	Result	Busy	Addr	Fu
				Issue	Comp						
1	LD	F0	0	R1	1			Load1	Yes	80	
1	MULTD	F4	F0	F2	2			Load2	Yes	72	
1	SD	F4	0	R1	3			Load3	No		
2	LD	F0	0	R1	6			Store1	Yes	80	Mult1
2	MULTD	F4	F0	F2	7			Store2	No		
2	SD	F4	0	R1				Store3	No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	S1	S2	RS	Qj	Qk	Code:
	Add1	No									LD
	Add2	No									MULTD
	Add3	No									SD
	Mult1	Yes	Multd			R(F2)	Load1				SUBI
	Mult2	Yes	Multd			R(F2)	Load2				BNEZ

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
7	72	Fu	Load2	Mult2						

Register file completely detached from computation

First and second iteration completely overlapped

# Loop Example: Cycle 8

## Instruction status:

ITER	Instruction	j	k	Exec Write		Issue	Comp	Result	Busy	Addr	Fu
				Issue	Comp						
1	LD	F0	0	R1	1				Yes	80	
1	MULTD	F4	F0	F2	2				Yes	72	
1	SD	F4	0	R1	3				No		
2	LD	F0	0	R1	6				Yes	80	Mult1
2	MULTD	F4	F0	F2	7				Yes	72	Mult2
2	SD	F4	0	R1	8				No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	S1	S2	RS	Code:
	Add1	No							LD F0 0 R1
	Add2	No							MULTD F4 F0 F2
	Add3	No							SD F4 0 R1
	Mult1	Yes	Multd		R(F2)	Load1			SUBI R1 R1 #8
	Mult2	Yes	Multd		R(F2)	Load2			BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
8	72	Fu	Load2	Mult2						



# Loop Example: Cycle 9

*Instruction status:*

ITER	Instruction	j	k	Exec Write		Busy <sup>0</sup>	Addr	Fu		
				Issue	CompResult					
1	LD	F0	0	R1	1	9	Load1	Yes	80	
1	MULTD	F4	F0	F2	2		Load2	Yes	72	
1	SD	F4	0	R1	3		Load3	No		
2	LD	F0	0	R1	6		Store1	Yes	80	Mult1
2	MULTD	F4	F0	F2	7		Store2	Yes	72	Mult2
2	SD	F4	0	R1	8		Store3	No		

*Reservation Stations:*

Time	Name	Busy	Op	Vj	Vk	S1	S2	RS	Code:	Fu
	Add1	No							LD	F0 0 R1
	Add2	No							MULTD	F4 F0 F2
	Add3	No							SD	F4 0 R1
	Mult1	Yes	Multd			R(F2)	Load1		SUBI	R1 R1 #8
	Mult2	Yes	Multd			R(F2)	Load2		BNEZ	R1 Loop

**Load1 completing:  
who is waiting?**

**Dispatching SUBI**

*Register result status*

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
9	72	Fu	Load2	Mult2						

# Loop Example: Cycle 10

*Instruction status:*

ITER	Instruction	j	k	Exec Write			Busy	Addr	Fu
				Issue	Comp	Result			
1	LD	F0	0	R1	1	9	10	Load1	No
1	MULTD	F4	F0	F2	2			Load2	Yes 72
1	SD	F4	0	R1	3			Load3	No
2	LD	F0	0	R1	6	10		Store1	Yes 80 Mult1
2	MULTD	F4	F0	F2	7			Store2	Yes 72 Mult2
2	SD	F4	0	R1	8			Store3	No

*Reservation Stations:*

Time	Name	Busy	Op	Vj	Vk	S1	S2	RS	Qj	Qk	Code:	
												Qj
	Add1	No									LD	F0 0 R1
	Add2	No									MULTD	F4 F0 F2
	Add3	No									SD	F4 0 R1
4	Mult1	Yes	Multd	M[80]	R(F2)						SUBI	R1 R1 #8
	Mult2	Yes	Multd		R(F2)	Load2					BNEZ	R1 Loop

**Load2 completing:  
who is waiting?**

**Dispatching BNEZ**

*Register result status*

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
10	64	Fu	Load2	Mult2						

# Loop Example: Cycle 11

## Instruction status:

ITER	Instruction	j	k	Exec Write			Busy	Addr	Fu
				Issue	Comp	Result			
1	LD	F0	0	R1	1	9	10	Load1	No
1	MULTD	F4	F0	F2	2			Load2	No
1	SD	F4	0	R1	3			Load3	Yes 64
2	LD	F0	0	R1	6	10	11	Store1	Yes 80 Mult1
2	MULTD	F4	F0	F2	7			Store2	Yes 72 Mult2
2	SD	F4	0	R1	8			Store3	No

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	S1	S2	RS	Qj	Qk	Code:	
												Qj
	Add1	No									LD	F0 0 R1
	Add2	No									MULTD	F4 F0 F2
	Add3	No									SD	F4 0 R1
3	Mult1	Yes	Multd	M[80]	R(F2)						SUBI	R1 R1 #8
4	Mult2	Yes	Multd	M[72]	R(F2)						BNEZ	R1 Loop

Next load in sequence

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
11	64	Fu	Load3	Mult2						

# Loop Example: Cycle 12

*Instruction status:*

<i>ITER</i>	<i>Instruction</i>	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</i>	<i>Exec</i>	<i>Write</i>	<i>Busy</i>	<i>Addr</i>	<i>Fu</i>
1	LD	F0	0	R1	1	9	10	Load1	No		
1	MULTD	F4	F0	F2	2			Load2	No		
1	SD	F4	0	R1	3			Load3	Yes	64	
2	LD	F0	0	R1	6	10	11	Store1	Yes	80	Mult1
2	MULTD	F4	F0	F2	7			Store2	Yes	72	Mult2
2	SD	F4	0	R1	8			Store3	No		

*Reservation Stations:*

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	<i>Code:</i>
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2
	Add3	No						SD F4 0 R1
2	Mult1	Yes	Multd	M[80]	R(F2)			SUBI R1 R1 #8
3	Mult2	Yes	Multd	M[72]	R(F2)			BNEZ R1 Loop

**Why not issue third multiply?**  
**Structural hazard**

*Register result status*

<i>Clock</i>	<i>R1</i>	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	<i>...</i>	<i>F30</i>
12	64	<i>Fu</i>	Load3	Mult2						

# Loop Example: Cycle 13

## Instruction status:

ITER	Instruction	j	k	Exec			Busy	Addr	Fu
				Issue	Comp	Result			
1	LD	F0	0	R1	1	9	10	Load1	No
1	MULTD	F4	F0	F2	2			Load2	No
1	SD	F4	0	R1	3			Load3	Yes 64
2	LD	F0	0	R1	6	10	11	Store1	Yes 80 Mult1
2	MULTD	F4	F0	F2	7			Store2	Yes 72 Mult2
2	SD	F4	0	R1	8			Store3	No

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:	S1	S2	RS
	Add1	No						LD	F0	0	R1
	Add2	No						MULTD	F4	F0	F2
	Add3	No						SD	F4	0	R1
1	Mult1	Yes	Multd	M[80]	R(F2)			SUBI	R1	R1	#8
2	Mult2	Yes	Multd	M[72]	R(F2)			BNEZ	R1	Loop	

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
13	64	Fu	Load3	Mult2						

# Loop Example: Cycle 14

*Instruction status:*

ITER	Instruction	j	k	Exec Write			Busy	Addr	Fu
				Issue	Comp	Result			
1	LD	F0	0	R1	1	9	10	Load1	No
1	MULTD	F4	F0	F2	2	14		Load2	No
1	SD	F4	0	R1	3			Load3	Yes 64
2	LD	F0	0	R1	6	10	11	Store1	Yes 80 Mult1
2	MULTD	F4	F0	F2	7			Store2	Yes 72 Mult2
2	SD	F4	0	R1	8			Store3	No

*Reservation Stations:*

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:			
									S1	S2	RS
	Add1	No						LD	F0	0	R1
	Add2	No						MULTD	F4	F0	F2
	Add3	No						SD	F4	0	R1
0	Mult1	Yes	Multd	M[80]	R(F2)			SUBI	R1	R1	#8
1	Mult2	Yes	Multd	M[72]	R(F2)			BNEZ	R1	Loop	

**Mult1 completing:  
Who is waiting?**

*Register result status*

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
14	64	Fu	Load3	Mult2						

# Loop Example: Cycle 15

*Instruction status:*

ITER	Instruction	j	k	Exec Write			Busy	Addr	Fu
				Issue	Comp	Result			
1	LD	F0	0	R1	1	9	10	Load1	No
1	MULTD	F4	F0	F2	2	14	15	Load2	No
1	SD	F4	0	R1	3			Load3	Yes 64
2	LD	F0	0	R1	6	10	11	Store1	Yes 80 [80]*R2
2	MULTD	F4	F0	F2	7	15		Store2	Yes 72 Mult2
2	SD	F4	0	R1	8			Store3	No

*Reservation Stations:*

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:			
									S1	S2	RS
	Add1	No						LD	F0	0	R1
	Add2	No						MULTD	F4	F0	F2
	Add3	No						SD	F4	0	R1
	Mult1	No						SUBI	R1	R1	#8
0	Mult2	Yes	Multd	M[72]	R(F2)			BNEZ	R1	Loop	

**Mult2 completing:  
Who is waiting?**

*Register result status*

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
15	64	Fu	Load3	Mult2						

# Loop Example: Cycle 16

## Instruction status:

ITER	Instruction	j	k	Exec Write			Busy	Addr	Fu	
				Issue	Comp	Result				
1	LD	F0	0	R1	1	9	10	Load1	No	
1	MULTD	F4	F0	F2	2	14	15	Load2	No	
1	SD	F4	0	R1	3	16		Load3	Yes	64
2	LD	F0	0	R1	6	10	11	Store1	Yes	80
2	MULTD	F4	F0	F2	7	15	16	Store2	Yes	72
2	SD	F4	0	R1	8			Store3	No	

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	S1	S2	RS	Code:
	Add1	No							LD
	Add2	No							MULTD
	Add3	No							SD
	Mult1	Yes	Multd		R(F2)	Load3			SUBI
	Mult2	No							BNEZ

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
16	64	Fu	Load3		Mult1					



# Loop Example: Cycle 17

## Instruction status:

ITER	Instruction	j	k	Exec Write			Busy	Addr	Fu		
				Issue	Comp	Result					
1	LD	F0	0	R1	1	9	10	Load1	No		
1	MULTD	F4	F0	F2	2	14	15	Load2	No		
1	SD	F4	0	R1	3	16	17	Load3	Yes	64	
2	LD	F0	0	R1	6	10	11	Store1	No		
2	MULTD	F4	F0	F2	7	15	16	Store2	Yes	72	[72]*R2
2	SD	F4	0	R1	8	17		Store3	Yes	64	Mult1

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	S1	S2	RS	Qj	Qk	Code:			
												Qj	Qk	
	Add1	No									LD	F0	0	R1
	Add2	No									MULTD	F4	F0	F2
	Add3	No									SD	F4	0	R1
	Mult1	Yes	Multd			R(F2)	Load3				SUBI	R1	R1	#8
	Mult2	No									BNEZ	R1	Loop	

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
17	64	Fu	Load3	Mult1						

# Loop Example: Cycle 18

## Instruction status:

ITER	Instruction	<i>j</i>	<i>k</i>	Exec Write			<i>Busy</i>	<i>Addr</i>	<i>Fu</i>
				<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
1	LD	F0	0	R1	1	9	10	Load1	No
1	MULTD	F4	F0	F2	2	14	15	Load2	No
1	SD	F4	0	R1	3	16	17	Load3	Yes 64
2	LD	F0	0	R1	6	10	11	Store1	No
2	MULTD	F4	F0	F2	7	15	16	Store2	No
2	SD	F4	0	R1	8	17	18	Store3	Yes 64 Mult1

## Reservation Stations:

Time	Name	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	Code:	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
	Add1	No						LD	F0	0	R1					
	Add2	No						MULTD	F4	F0	F2					
	Add3	No						SD	F4	0	R1					
	Mult1	Yes	Multd		R(F2)	Load3		SUBI	R1	R1	#8					
	Mult2	No						BNEZ	R1	Loop						

## Register result status

Clock	R1	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
18	64	<i>Fu</i>	Load3	Mult1						

# Tomasulo overlaps iterations of loops?

- Register renaming
  - Multiple iterations use different physical destinations for registers (dynamic loop unrolling).
- Reservation stations
  - Permit instruction issue to advance past integer control flow operations

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

## Tomasulo

Pipelined Functional Units

(6 load, 3 store, 3 +, 2 x/÷)

window size:  $\leq 14$  instructions

No issue on structural hazard

WAR: renaming avoids them

WAW: renaming avoids them

Broadcast results from FU

Control: reservation stations

## Scoreboard

Multiple Functional Units

(1 load/store, 1 +, 2 x, 1 ÷)

$\leq 5$  instructions

same

stall completion

stall issue

Write/read registers

Central scoreboard

# Tomasulo Drawbacks

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