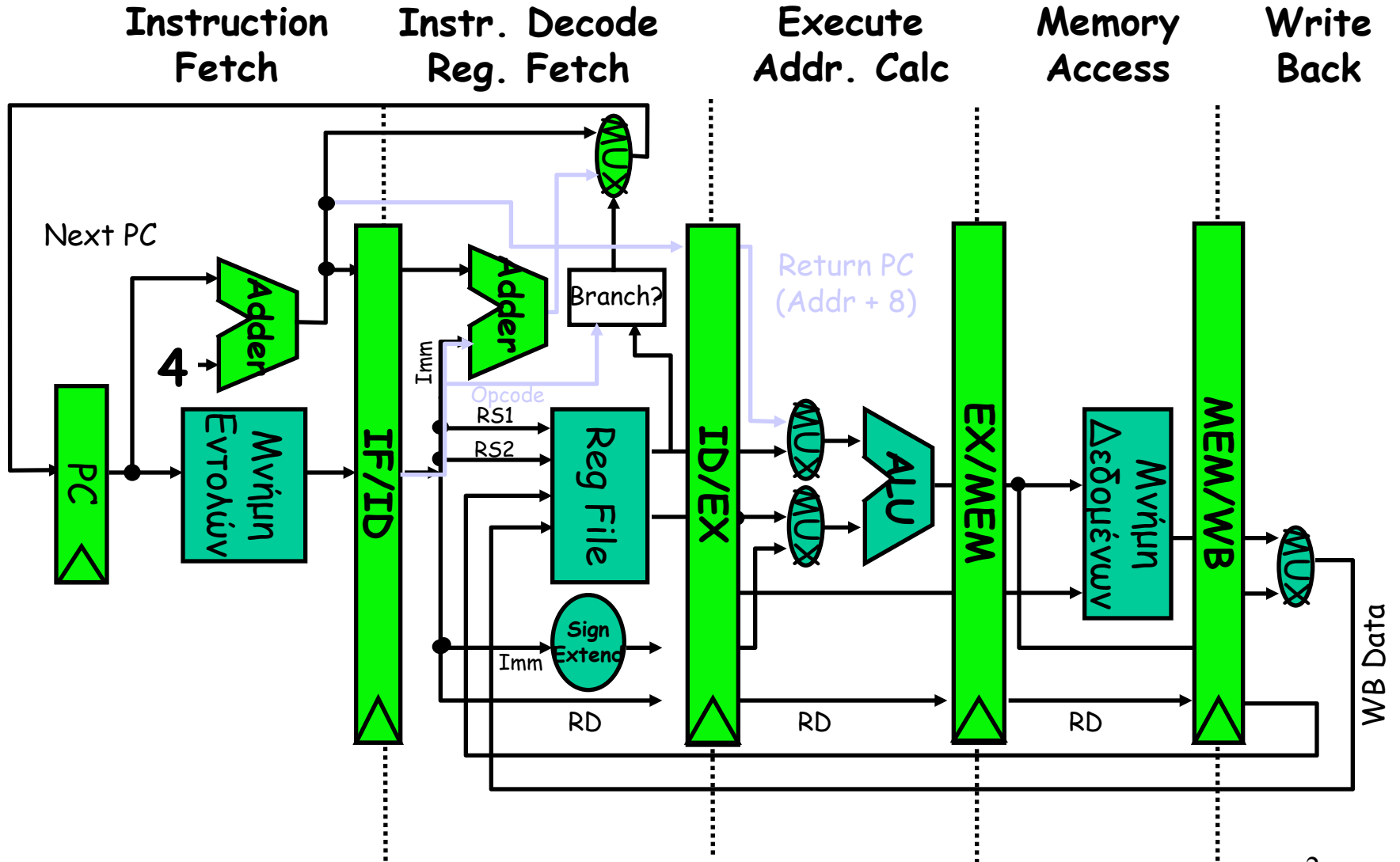


HY425  
Αρχιτεκτονική Υπολογιστών

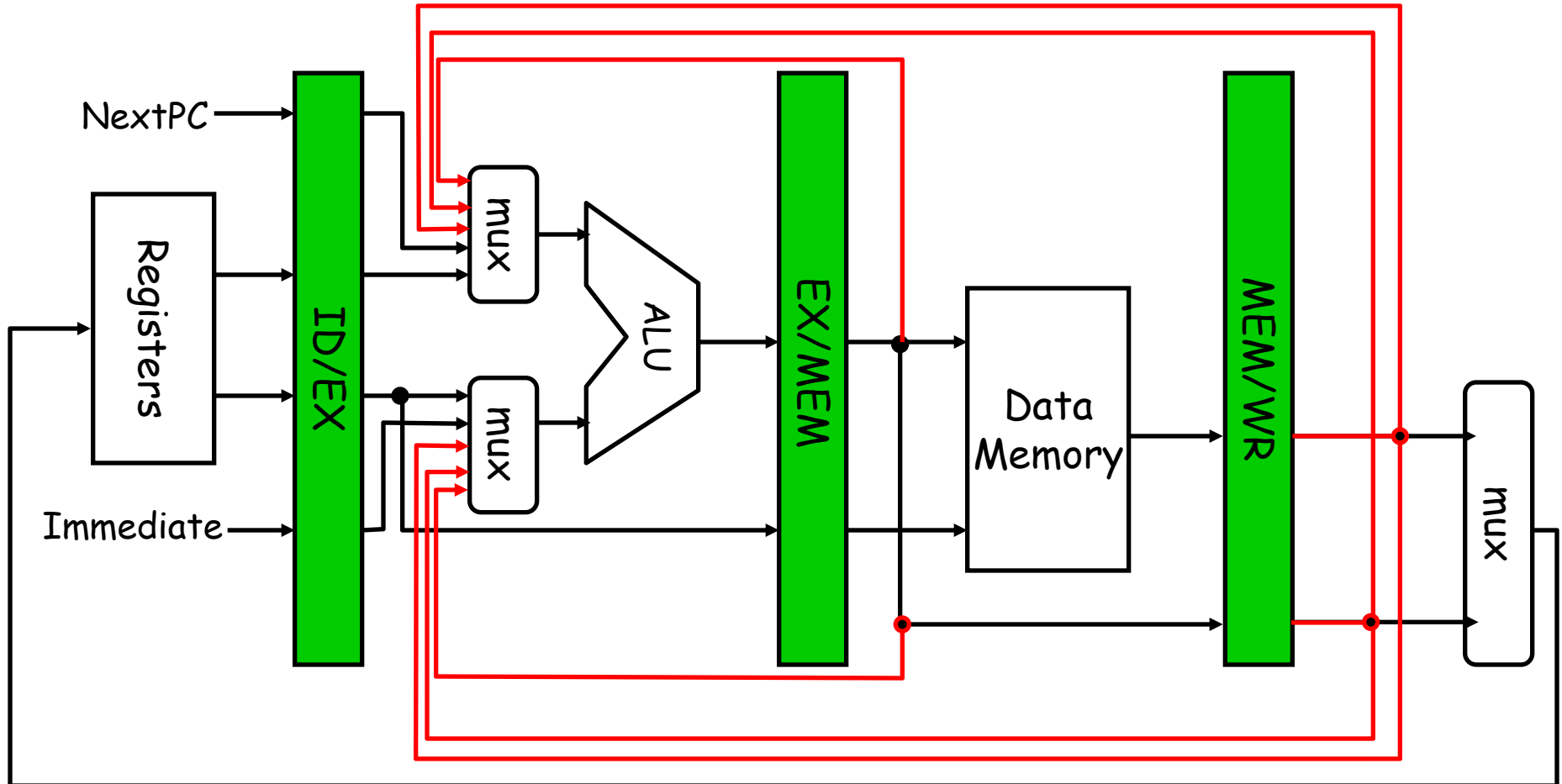
Tomasulo  
Register Renaming

Βασίλης Παπαευσταθίου  
Ιάκωβος Μαυροειδής

# Επεξεργαστής DLX



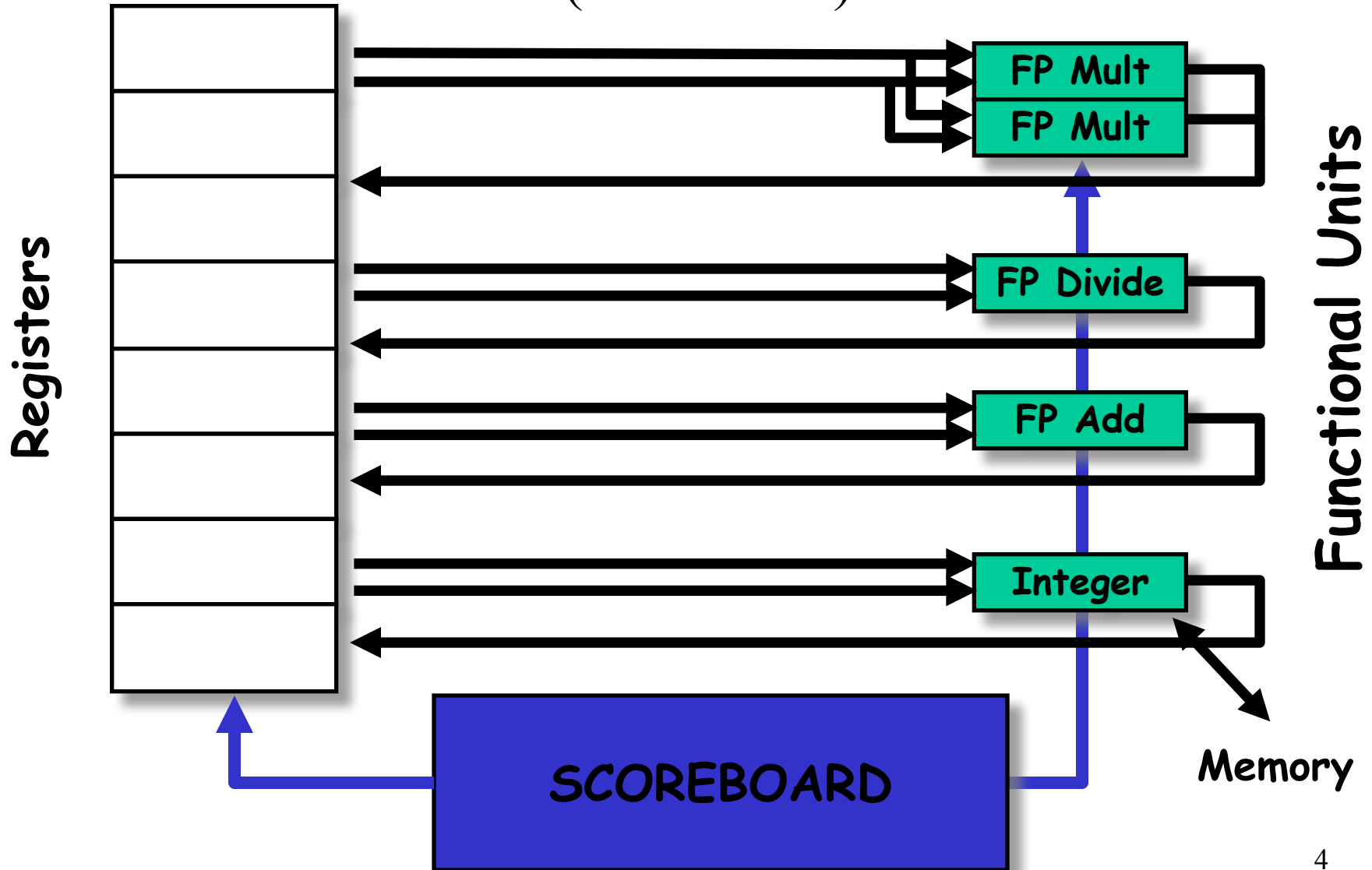
# HW Change for Forwarding



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

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

# Αρχιτεκτονική Scoreboard (CDC 6600)



## CDC 6600 Scoreboard

- Speedup 1.7 from compiler; 2.5 by hand  
BUT slow memory (no cache) limits benefit
- 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 Algorithm

- IBM 360/91 3 χρόνια μετά από CDC 6600 (1966)
- Στόχος: High Performance without special compilers
- Διαφορές μεταξύ IBM 360 & CDC 6600 ISA
  - IBM has 4 FP registers vs. 8 in CDC 6600
  - IBM has memory-register ops
- Γιατί το μελετάμε? lead to Alpha 21264, HP 8000, MIPS 10000, Pentium II, PowerPC 604, ...

# Register Renaming



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

# Εξαρτήσεις Μεταξύ Εντολών

• **(True) Data Dependences** : Δύο εντολές είναι data dependent όταν υπάρχει μία αλυσίδα από RAW hazards μεταξύ τους.

**Loop:**

L.D      **F0**, 0(R1)  
          ↓  
ADD.D    **F4**, **F0**, F2  
          ↓  
S.D      **F4**, 0(R1)  
  
DADDUI   **R1**, R1, -8  
          ↓  
BNE      **R1**, R2, Loop

*Τι εκτελεί το παραπάνω πρόγραμμα;*

S.D      **F0**, 100(R4)  
          ↓ **???**  
L.D      **F1**, 20(R6)



# Παράδειγμα Scoreboard : Κύκλος 17

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>
L.D	F6	34+ R2	1	2	3	4
L.D	F2	45+ R3	5	6	7	8
MULT.D	F0	F2 F4	6	9		
SUB.D	F8	F6 F2	7	9	11	12
DIV.D	F10	F0 F6	8			
ADD.D	F6	F8 F2	13	14	16	

**WAR Hazard!**

## 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	Mult1			Add		Divide			

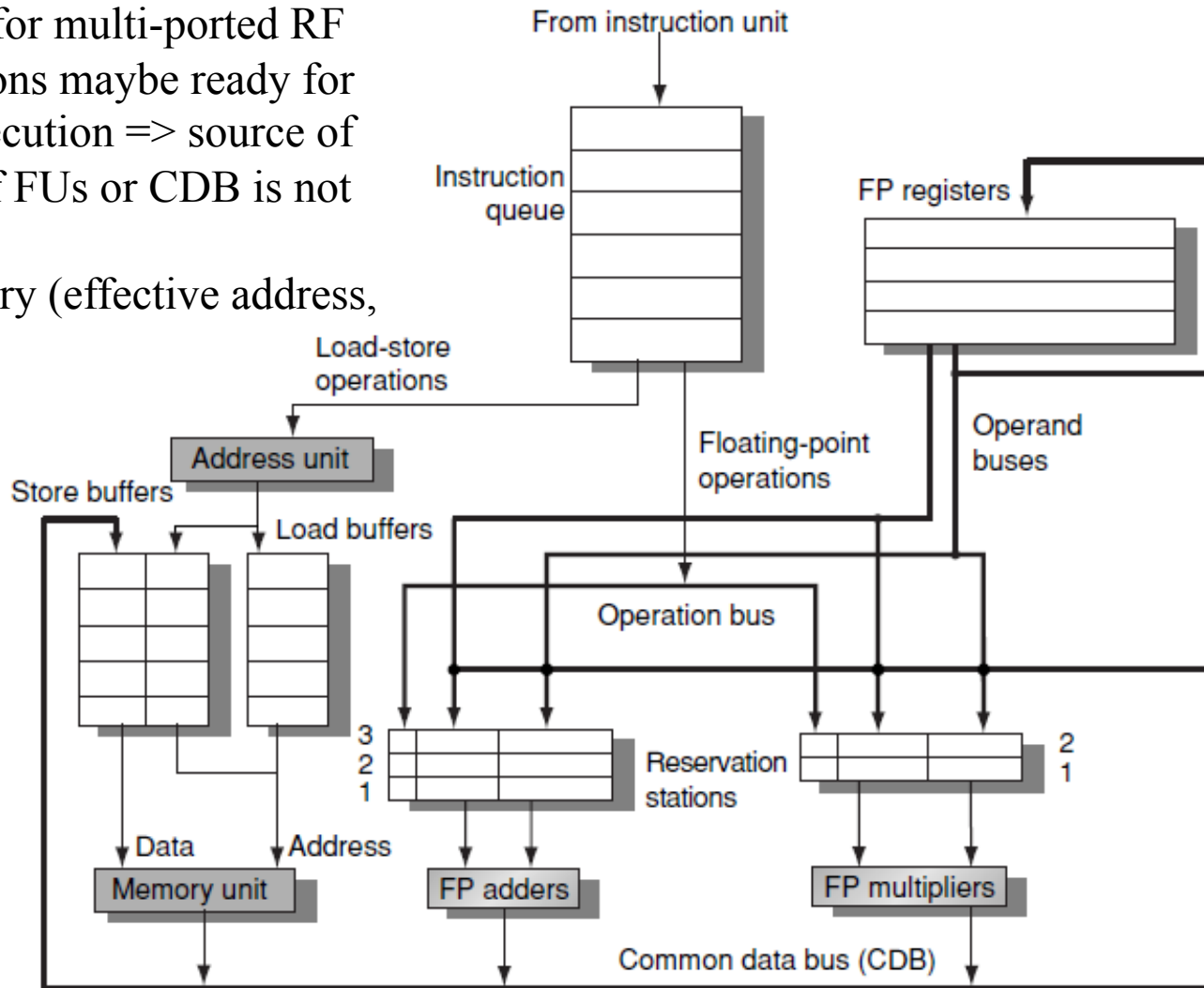
- Γιατί να μην γραφτούν τα αποτελέσματα της ADD???

# Tomasulo Algorithm vs. Scoreboard

- FU buffers ονομάζονται “**reservation stations**” και έχουν τις τιμές των μεταβλητών προς εκτέλεση.
- Οι Source Registers στις εντολές αντικαθίστονται από τιμές ή pointers στα reservation stations(RS), ονομάζεται **register renaming**:
  - αποφεύγει WAR, WAW hazards
  - Περισσότερα reservation stations από πραγματικούς registers (so can do optimizations compilers can't)
- Αποτελέσματα στα FU από τα RS, **όχι μέσω του register file**, πάνω από το **Common Data Bus** που κάνει broadcasts τα αποτελέσματα σε όλα τα FUs
  - Bypassing of results
- 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
- Control λογική & buffers **κατανεμημένα** με τα Function Units (FU) vs. κεντριοποιημένα στο scoreboard

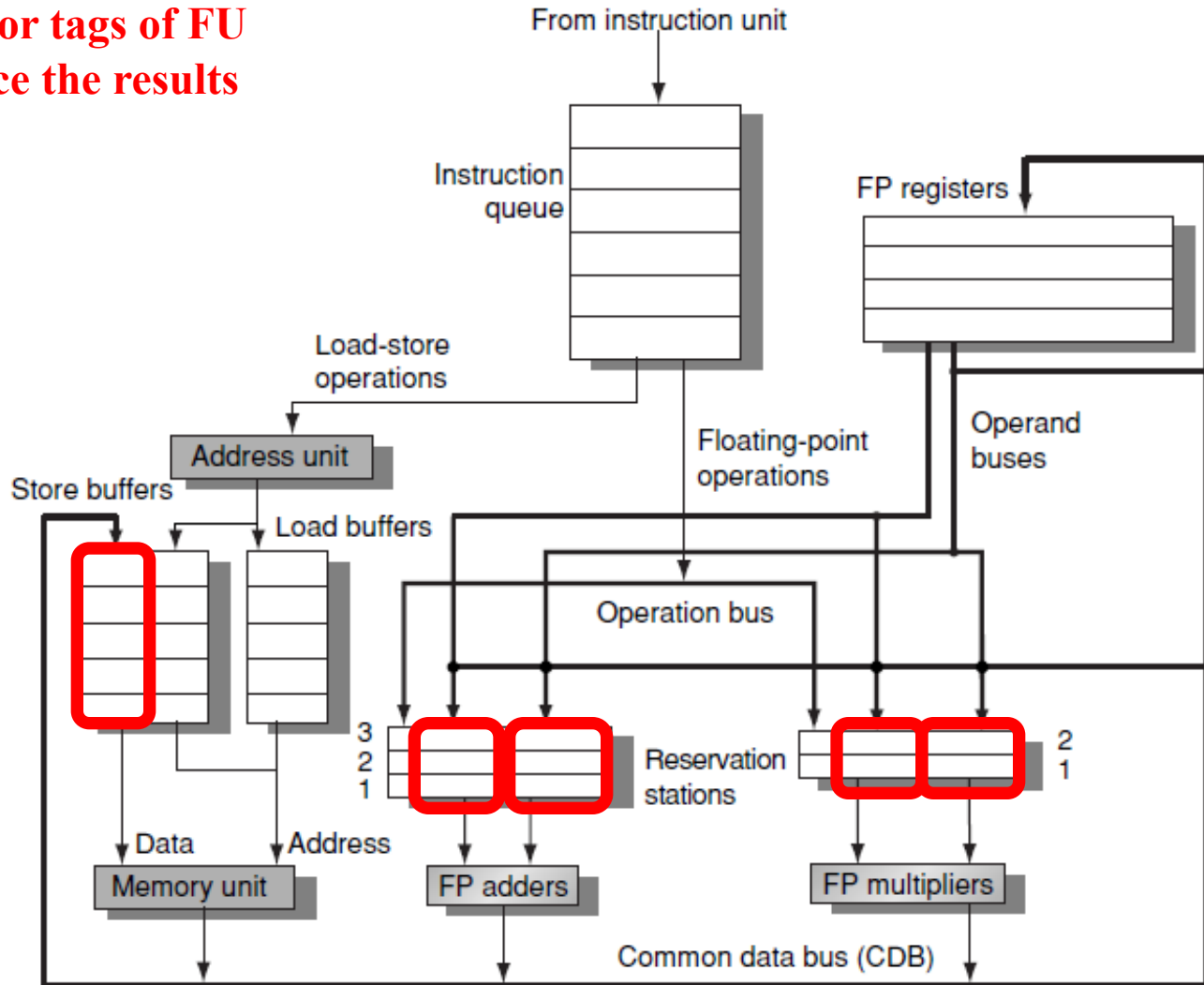
# Tomasulo Organization

- 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)



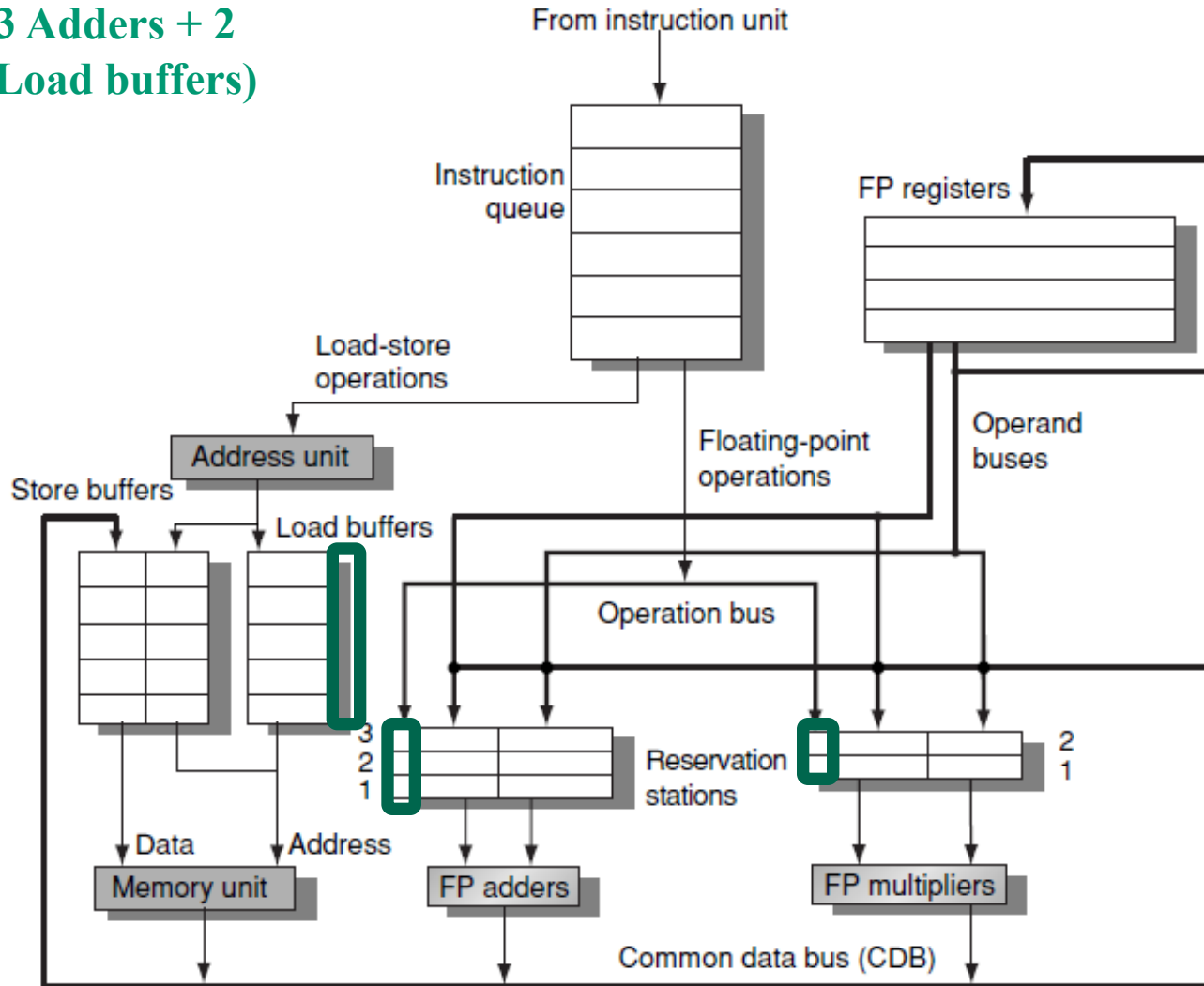
# Tomasulo Organization

Register values or tags of FU  
that will produce the results



# Tomasulo Organization

Ten 4-bit tags (3 Adders + 2 Multipliers + 5 Load buffers)



# Three Stages of Tomasulo Algorithm

## 1. **Issue (dispatch)**—πάρε εντολή από την FP Ορ ουρά

Αν υπάρχει ελεύθερο reservation station (no structural hazard), issue instr & send operand values or keep track of FU that will produce the operand (register renaming).

## 2. **Execution**—εκτέλεση στην αριθμητική μονάδα (EX)

Όταν και οι τιμές των δύο μεταβλητών είναι έτοιμες άρχισε εκτέλεση.  
Αν δεν είναι έτοιμες, watch Common Data Bus for result

## 3. **Write result (commit)** —Τέλος εκτέλεσης (WB)

Γράψε αποτέλεσμα στο Common Data Bus για όλες τις μονάδες και registers που περιμένουν, mark reservation station available

- Συνηθισμένο data bus: data + destination (“go to” bus)
- **Common data bus**: data + **source/res. station tag** (“**come from**” bus).
  - Ένας Master πολλοί Slaves
  - 64 bits of data + 4 bits of Functional Unit **source** address
  - Write if matches expected Functional Unit (produces result)
  - Does the broadcast

# Reservation Station Components

## Reservation Station

**Busy:** Δείχνει αν το reservation station είναι busy ή όχι.

**Op:** Λειτουργία προς εκτέλεση στην μονάδα (e.g., + or −)

**V<sub>j</sub>, V<sub>k</sub>:** Τιμή του Source Register

- Store buffers έχουν ένα πεδίο V field, τιμή προς αποθήκευση

**Q<sub>j</sub>, Q<sub>k</sub>:** Reservation stations που παράγουν source registers (value to be written)

- Note: No ready flags as in Scoreboard; Q<sub>j</sub>, Q<sub>k</sub>=0 => ready
- Store buffers only have Q<sub>i</sub> for RS producing result

## Register result status

Δείχνει ποιο functional unit θα γράψει κάθε register, αν υπάρχει κανένα. Blank when no pending instructions that will write that register.

# Tomasulo Example

## Instruction status:

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

<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				2 Load1	Yes 34+R2
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:

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>	<i>F12</i>	...	<i>F30</i>
1				Load1					

# Tomasulo Example Cycle 2

## Instruction status:

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

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>
2		Load2		Load1					

Note: Unlike 6600, can have multiple loads outstanding

# Tomasulo Example Cycle 3

## Instruction status:

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

## 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		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

# Tomasulo Example Cycle 4

## 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		0 Load2
MULTD	F0	F2	F4	3			Load3
SUBD	F8	F6	F2	4			
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	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)								

- Load2 completing

# Tomasulo Example Cycle 5

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

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

- Issue ADDD here vs. scoreboard?

# Tomasulo Example Cycle 7

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

- Add1 completing; what is waiting for it?

# Tomasulo Example Cycle 8

## 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			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>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					
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	<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		

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

- Add2 completing; what is waiting for it?

# Tomasulo Example Cycle 11

## Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Exec	Write	Comp	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					Mult2			
	Reg File	M(A2)			(M-M+N)		(M-M)			

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

# Tomasulo Example Cycle 12

## 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>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					Mult2				
	Reg File									
	M(A2)				(M-M+M) (M-M)					

# 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>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					
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>	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>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	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>	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	15		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>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	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>	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	15	16	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>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					
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)					



Faster than light computation  
(skip a couple of cycles)

# Tomasulo Example Cycle 55

## 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>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								
	Reg File								
	M*F4	M(A2)		(M-M+M)	(M-M)				

# Tomasulo Example Cycle 56

## 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	15	16	Load3
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	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
56	FU								
	Reg File								
	M*F4	M(A2)	(M-M+M)	(M-M)					

- 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					
	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>
56	FU Result								
Reg File	M*F4 M(A2)		(M-M+M)		(M-M)				

- 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
  - Lack of forwarding
  - WAR, WAW hazards cause stalls

# Tomasulo Loop Example

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

<b>Loop:L.D</b>	<b>F0</b>	<b>0</b>	<b>R1</b>
<b>MULT.D</b>	<b>F4</b>	<b>F0</b>	<b>F2</b>
<b>S.D</b>	<b>F4</b>	<b>0</b>	<b>R1</b>
<b>SUBI</b>	<b>R1</b>	<b>R1</b>	<b>#8</b>
<b>BNEZ</b>	<b>R1</b>	<b>Loop</b>	

- Assume Multiply takes 4 clocks
- Assume first load takes 8 clocks (cache miss), second load takes 1 clock (hit) and stores take 1 clock (hit)
- To be clear, will show clocks for SUBI, BNEZ
- Reality: integer instructions ahead
- Assume R1 = 80 in the first iteration

# Loop Example

*Instruction status:*

*Exec Write*

<i>ITER</i>	<i>Instruction</i>		<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</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      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*

<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:*

*Exec Write*

<i>ITER</i>	<i>Instruction</i>	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>CompResult</i>	<i>Busy</i>	<i>Addr</i>	<i>Fu</i>
1	LD	F0	0	R1	1	8 Load1	Yes	80
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 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*

<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>
1	80	<i>Fu</i>	Load1							



# Loop Example Cycle 2

*Instruction status:*

*Exec Write*

<i>ITER</i>	<i>Instruction</i>	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</i>	<i>Busy</i>	<i>Addr</i>	<i>Fu</i>
1	LD	F0	0	R1	1		7 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:*

*S1 S2 RS*

<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
Mult1		Yes	Multd		R(F2)	Load1		SUBI R1 R1 #8
Mult2		No						BNEZ R1 Loop

*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>
2	80	<i>Fu</i>	Load1		Mult1					

# Loop Example Cycle 3

*Instruction status:*

*Exec Write*

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

- Implicit renaming sets up “DataFlow” graph <sup>42</sup>

# Loop Example Cycle 4

*Instruction status:*

*Exec Write*

ITER	Instruction	<i>j</i>	<i>k</i>	Issue	Comp	Result	Busy	Addr	<i>Fu</i>		
1	LD	F0	0	R1	1		5	Load1	Yes	80	
1	MULTD	F4	F0	F2	2			Load2	No		
1	SD	F4	0	R1	3			Load3	No		
2	LD	F0	0	R1				Store1	Yes	80	Mult1
2	MULTD	F4	F0	F2				Store2	No		
2	SD	F4	0	R1				Store3	No		

*Reservation Stations:*

Time	Name	Busy	Op	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	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	

*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>
4	80	<i>Fu</i>	Load1			Mult1				

- Dispatching SUBI Instruction

# Loop Example Cycle 5

*Instruction status:*

*Exec Write*

<i>ITER</i>	<i>Instruction</i>	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</i>	<i>Busy</i>	<i>Addr</i>	<i>Fu</i>
1	LD	F0	0	R1	1		4 Load1	Yes	80
1	MULTD	F4	F0	F2	2		Load2	No	
1	SD	F4	0	R1	3		Load3	No	
2	LD	F0	0	R1			Store1	Yes	80
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 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*

<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>
5	72	<i>Fu</i>	Load1		Mult1					

- And, BNEZ instruction

# Loop Example Cycle 6

*Instruction status:*

*Exec Write*

<i>ITER</i>	<i>Instruction</i>	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</i>	<i>Busy</i>	<i>Addr</i>	<i>Fu</i>
1	LD	F0	0	R1	1		3 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:*

<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
Mult1	Yes	Multd			R(F2)	Load1		SUBI R1 R1 #8
Mult2	No							BNEZ R1 Loop

*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>
6	72	<i>Fu</i>	Load2	Mult1						

- Notice that F0 never sees Load from location 80
- Load2 waits for cache line behind Load1

# Loop Example Cycle 7

*Instruction status:*

*Exec Write*

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		Yes	72	
1	SD	F4	0	R1	3		No		
2	LD	F0	0	R1	6		Yes	80	Mult1
2	MULTD	F4	F0	F2	7		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 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	<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	72	<i>Fu</i>	Load2	Mult2						

- Register file completely detached from computation
- First and Second iteration completely overlapped

# Loop Example Cycle 8

*Instruction status:*

*Exec Write*

<i>ITER</i>	<i>Instruction</i>	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</i>	<i>Busy</i>	<i>Addr</i>	<i>Fu</i>
1	LD	F0	0	R1	1		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	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
	Mult1	Yes	Multd		R(F2)	Load1		SUBI R1 R1 #8
	Mult2	Yes	Multd		R(F2)	Load2		BNEZ R1 Loop

*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>
8	72	<i>Fu</i>	Load2		Mult2					

# Loop Example Cycle 9

*Instruction status:*

*Exec Write*

<i>ITER</i>	<i>Instruction</i>	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>CompResult</i>	<i>Busy</i>	<i>Addr</i>	<i>Fu</i>
1	LD	F0	0	R1	1 9	0 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:*

<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
	Mult1	Yes	Multd		R(F2)	Load1		SUBI R1 R1 #8
	Mult2	Yes	Multd		R(F2)	Load2		BNEZ R1 Loop

*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>
9	72	<i>Fu</i>	Load2	Mult2						

- Load1 completing: who is waiting?
- Note: Dispatching SUBI



# Loop Example Cycle 10

*Instruction status:*

*Exec Write*

<i>ITER</i>	<i>Instruction</i>	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</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	Yes 72
1	SD	F4	0	R1	3			Load3	No
2	LD	F0	0	R1	6	10		Store1	Yes 80
2	MULTD	F4	F0	F2	7			Store2	Yes 72
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
4	Mult1	Yes	Multd	M[80]	R(F2)			SUBI R1 R1 #8
	Mult2	Yes	Multd		R(F2)	Load2		BNEZ R1 Loop

*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>
10	64	<i>Fu</i>	Load2			Mult2				

- Load2 completing: who is waiting?
- Note: Dispatching BNEZ

# Loop Example Cycle 11

*Instruction status:*

*Exec Write*

<i>ITER</i>	<i>Instruction</i>	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</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
2	MULTD	F4	F0	F2	7			Store2	Yes 72
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
3	Mult1	Yes	Multd	M[80]	R(F2)			SUBI R1 R1 #8
4	Mult2	Yes	Multd	M[72]	R(F2)			BNEZ R1 Loop

*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>
11	64	<i>Fu</i>	Load3			Mult2				

- Next load in sequence

# Loop Example Cycle 12

*Instruction status:*

*Exec Write*

<i>ITER</i>	<i>Instruction</i>	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</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
2	MULTD	F4	F0	F2	7			Store2	Yes 72
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

*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				

- Why not issue third multiply? Structural hazard

# Loop Example Cycle 13

*Instruction status:*

*Exec Write*

<i>ITER</i>	<i>Instruction</i>	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</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
2	MULTD	F4	F0	F2	7			Store2	Yes 72
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
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*

<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>
13	64	<i>Fu</i>	Load3			Mult2				

# Loop Example Cycle 14

*Instruction status:*

*Exec Write*

<i>ITER</i>	<i>Instruction</i>	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</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	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:*

<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
0	Mult1	Yes	Multd	M[80]	R(F2)			SUBI R1 R1 #8
1	Mult2	Yes	Multd	M[72]	R(F2)			BNEZ R1 Loop

*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>
14	64	<i>Fu</i>	Load3			Mult2				

- Mult1 completing. Who is waiting?

# Loop Example Cycle 15

*Instruction status:*

*Exec Write*

ITER	Instruction	<i>j</i>	<i>k</i>	Issue	Comp	Result	Busy	Addr	<i>Fu</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			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	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	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
0	Mult2	Yes	Multd	M[72]	R(F2)			BNEZ R1 Loop

*Register result status*

Clock	R1	<i>Fu</i>	<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	64		Load3		Mult2						

- Mult2 completing. Who is waiting?

# Loop Example Cycle 16

*Instruction status:*

*Exec Write*

<i>ITER</i>	<i>Instruction</i>	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</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	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 [80]*R2
2	MULTD	F4	F0	F2	7	15	16	Store2	Yes 72 [72]*R2
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
Mult1	Yes	Multd			R(F2)	Load3		SUBI R1 R1 #8
Mult2	No							BNEZ R1 Loop

*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>
16	64	<i>Fu</i>	Load3		Mult1					

# Loop Example Cycle 17

*Instruction status:*

*Exec Write*

<i>ITER</i>	<i>Instruction</i>	<i>j</i>	<i>k</i>	<i>Issue CompResult</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	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:*

<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
Mult1	Yes	Multd			R(F2)	Load3		SUBI R1 R1 #8
Mult2	No							BNEZ R1 Loop

*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>
17	64	<i>Fu</i>	Load3			Mult1				



# Loop Example Cycle 18

*Instruction status:*

*Exec Write*

<i>ITER</i>	<i>Instruction</i>	<i>j</i>	<i>k</i>	<i>Issue CompResult</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	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:*

*S1 S2 RS*

<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
Mult1	Yes	Multd			R(F2)	Load3		SUBI R1 R1 #8
Mult2	No							BNEZ R1 Loop

*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>
18	64	<i>Fu</i>	Load3		Mult1					

## Why can Tomasulo overlap 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 v. Scoreboard

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

Pipelined Functional Units

(6 load, 3 store, 3 +, 2 x/÷)

window size:  $\leq 14$  instructions

No issue on structural hazard

WAR: renaming avoids

WAW: renaming avoids

Broadcast results from FU

Control: reservation stations

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 assoc stores