

Lecture 6 Dynamic Scheduling

Dynamic vs. Static Scheduling

- Data hazards in a program to cause a processor to stall.
- With **static scheduling** the **compiler** tries to reorder these instructions during **compile time** to reduce pipeline stalls.
 - Uses less hardware
 - Can use more powerful algorithms
- With **dynamic scheduling** the **hardware** tries to rearrange the instructions during **run-time** to reduce pipeline stalls.
 - Simpler compiler
 - Handles dependencies not known at compile time
 - Allows code compiled for a different machine to run efficiently.

Out-Of-Order Execution

- In our previous model of DLX, all instructions executed in the order that they appear
- This can lead to unnecessary stalls

DIVD	F0, F2, F4
ADDD	F10, F0, F8
SUBD	F12, F8, F14
- SUBD stalls waiting for the ADDD to go first, even though SUBD does not have a data dependency.
- With out-of-order execution, the SUBD is allowed to be executed before the add
 - This can lead to out-of order completion, which can cause WAW and WAR hazards

Scoreboarding

- The scoreboard implements a centralized control scheme that
 - Detects all resource and data hazards
 - Allows instructions to execute out-of-order when no resource hazards or data dependencies
- First implemented in 1964 by the CDC 6600, which had 18 separate functional units
 - 4 FP units (2 multiply, 1 add, 1 divide)
 - 7 memory units (5 loads, 2 stores)
 - 7 integer units (add, shift, logical, compare, etc.)
- Dynamic DLX (much simpler)
 - 2 FP multiply (10 EX cycles)
 - 1 FP add (2 EX cycles)
 - 1 FP divide (40 EX cycles)
 - 1 integer unit (1 EX cycle)

Out-of-Order Execution

- Out-of-order execution divides ID stage:
 1. **Issue**—decode instructions, check for structural hazards
 2. **Read operands**—wait until no data hazards, then read operands
- Scoreboards allow instruction to execute whenever 1 & 2 hold, not waiting for prior instructions
- CDC 6600: *In order issue*, out of order execution, out of order commit (also called completion)

Scoreboard Implications

- Out-of-order completion can lead to WAR and WAW hazards?
- Solution for WAW
 - Detect WAW hazard before reading operands
 - Stall write until other instruction completes
- Solutions for WAR
 - Detect WAR hazards before writing back to the register files and stall the write back
- This scoreboard does not take advantage of forwarding, since it waits until both results are written back to the register file
- Scoreboard replaces ID, EX, WB with 4 stages

Four Stages of Scoreboard Control

- **Issue (ID1)**
 - decode instructions
 - check for structural and WAW hazards
 - stall until structural and WAW hazards are resolved
- **Read operands (ID2)**
 - wait until no RAW hazards
 - then read operands
- **Execution (EX)**
 - operate on operands
 - may be multiple cycles - notify scoreboard when done
- **Write result (WB)**
 - finish execution
 - stall if WAR hazard

Three Parts of the Scoreboard

1. **Instruction status**—which of 4 steps the instruction is in: ID1, ID2, EX, or WB.
2. **Functional unit status**—Indicates the state of the functional unit (FU). 9 fields for each functional unit
 - Busy**—Indicates whether the unit is busy or not
 - Op**—Operation to perform in the unit (e.g., + or -)
 - Fi**—Destination register
 - Fj, Fk**—Source-register numbers
 - Qj, Qk**—Functional units producing source registers Fj, Fk
 - Rj, Rk**—Flags indicating when Fj, Fk are ready
3. **Register result status**—Indicates which functional unit will write each register, if one exists. Blank when no pending instructions will write that register

Scoreboard Example for DLX

Instruction status				Read				Executi				Write			
Instruction	j	k		Issue	operand	complete	Result	Issue	operand	complete	Result	Issue	operand	complete	Result
LD	F6	34+	R2												
LD	F2	45+	R3												
MULT	F0	F2	F4												
SUBD	F8	F6	F2												
DIVD	F10	F0	F6												
ADD	F6	F8	F2												

Functional unit status		dest				S1	S2	FU for j				FU for k				Fj?	Fk?		
Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No																	
	Mult1	No																	
	Mult2	No																	
	Add	No																	
	Divide	No																	

Register result status		F0	F2	F4	F6	F8	F10	F12	...	F30
Clock	1									
	FU									

Scoreboard Example Cycle 1

Instruction status				Read				Executi				Write			
Instruction	j	k		Issue	operand	complete	Result	Issue	operand	complete	Result	Issue	operand	complete	Result
LD	F6	34+	R2	1											
LD	F2	45+	R3												
MULT	F0	F2	F4												
SUBD	F8	F6	F2												
DIVD	F10	F0	F6												
ADD	F6	F8	F2												

Functional unit status		dest				S1	S2	FU for j				FU for k				Fj?	Fk?		
Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	Load	F6															Yes
	Mult1	No																	
	Mult2	No																	
	Add	No																	
	Divide	No																	

Register result status		F0	F2	F4	F6	F8	F10	F12	...	F30
Clock	1									
	FU									Integer

Scoreboard Example Cycle 2

Instruction status				Read				Executi				Write			
Instruction	j	k		Issue	operand	complete	Result	Issue	operand	complete	Result	Issue	operand	complete	Result
LD	F6	34+	R2	1											
LD	F2	45+	R3	2											
MULT	F0	F2	F4												
SUBD	F8	F6	F2												
DIVD	F10	F0	F6												
ADD	F6	F8	F2												

Functional unit status		dest				S1	S2	FU for j				FU for k				Fj?	Fk?		
Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	Load	F6															Yes
	Mult1	No																	
	Mult2	No																	
	Add	No																	
	Divide	No																	

Register result status		F0	F2	F4	F6	F8	F10	F12	...	F30
Clock	2									
	FU									Integer

- **Issue 2nd LD?**

Scoreboard Example Cycle 3

Instruction status				Read				Executi				Write			
Instruction	j	k		Issue	operand	complete	Result	Issue	operand	complete	Result	Issue	operand	complete	Result
LD	F6	34+	R2	1											
LD	F2	45+	R3	2											
MULT	F0	F2	F4	3											
SUBD	F8	F6	F2												
DIVD	F10	F0	F6												
ADD	F6	F8	F2												

Functional unit status		dest				S1	S2	FU for j				FU for k				Fj?	Fk?		
Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	Load	F6															Yes
	Mult1	No																	
	Mult2	No																	
	Add	No																	
	Divide	No																	

Register result status		F0	F2	F4	F6	F8	F10	F12	...	F30
Clock	3									
	FU									Integer

- **Issue MULT?**

Scoreboard Example Cycle 4

Instruction status				Read				Executiv Write			
Instruction	j	k		Issue	operand	complete	Result				
LD	F6	34+	R2	1	2	3	4				
LD	F2	45+	R3								
MULT	F0	F2	F4								
SUBD	F8	F6	F2								
DIVD	F10	F0	F6								
ADD	F6	F8	F2								

Functional unit status											
Time	Name	Busy	Op	dest	S1	S2	FU for j	FU for k	Fj?	Fk?	Rk?
	Integer	Yes	Load	F6	Fj	Fk	Qj	Qk	Rj	Rk	Yes
	Mult1	No									
	Mult2	No									
	Add	No									
	Divide	No									

Register result status											
Clock	F0	F2	F4	F6	F8	F10	F12	...	F30	FU	
4										Integer	

Scoreboard Example Cycle 5

Instruction status				Read				Executiv Write			
Instruction	j	k		Issue	operand	complete	Result				
LD	F6	34+	R2	1	2	3	4				
LD	F2	45+	R3	5							
MULT	F0	F2	F4								
SUBD	F8	F6	F2								
DIVD	F10	F0	F6								
ADD	F6	F8	F2								

Functional unit status											
Time	Name	Busy	Op	dest	S1	S2	FU for j	FU for k	Fj?	Fk?	Rk?
	Integer	Yes	Load	F2	Fj	Fk	Qj	Qk	Rj	Rk	Yes
	Mult1	No									
	Mult2	No									
	Add	No									
	Divide	No									

Register result status											
Clock	F0	F2	F4	F6	F8	F10	F12	...	F30	FU	
5										Integer	

Scoreboard Example Cycle 6

Instruction status				Read				Executiv Write			
Instruction	j	k		Issue	operand	complete	Result				
LD	F6	34+	R2	1	2	3	4				
LD	F2	45+	R3	5	6						
MULT	F0	F2	F4	6							
SUBD	F8	F6	F2								
DIVD	F10	F0	F6								
ADD	F6	F8	F2								

Functional unit status											
Time	Name	Busy	Op	dest	S1	S2	FU for j	FU for k	Fj?	Fk?	Rk?
	Integer	Yes	Load	F2	Fj	Fk	Qj	Qk	Rj	Rk	Yes
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes	
	Mult2	No									
	Add	No									
	Divide	No									

Register result status											
Clock	F0	F2	F4	F6	F8	F10	F12	...	F30	FU	
6										Mult1 Integer	

Scoreboard Example Cycle 7

Instruction status				Read				Executiv Write			
Instruction	j	k		Issue	operand	complete	Result				
LD	F6	34+	R2	1	2	3	4				
LD	F2	45+	R3	5	6	7					
MULT	F0	F2	F4	6							
SUBD	F8	F6	F2	7							
DIVD	F10	F0	F6								
ADD	F6	F8	F2								

Functional unit status											
Time	Name	Busy	Op	dest	S1	S2	FU for j	FU for k	Fj?	Fk?	Rk?
	Integer	Yes	Load	F2	Fj	Fk	Qj	Qk	Rj	Rk	Yes
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes	
	Mult2	No									
	Add	Yes	Sub	F8	F6	F2		Integer	Yes	No	
	Divide	No									

Register result status											
Clock	F0	F2	F4	F6	F8	F10	F12	...	F30	FU	
7										Mult1 Integer Add	

- Read multiply operands?

Scoreboard Example Cycle 8a

Instruction status				Read				Executiv Write			
Instruction	j	k		Issue	operand	complete	Result				
LD	F6	34+	R2	1	2	3	4				
LD	F2	45+	R3	5	6	7					
MULT	F0	F2	F4	6							
SUBD	F8	F6	F2	7							
DIVD	F10	F0	F6	8							
ADD	F6	F8	F2								

Functional unit status											
Time	Name	Busy	Op	dest	S1	S2	FU for j	FU for k	Fj?	Fk?	Rk?
	Integer	Yes	Load	F2	Fj	Fk	Qj	Qk	Rj	Rk	Yes
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes	
	Mult2	No									
	Add	Yes	Sub	F8	F6	F2		Integer	Yes	No	
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes	

Register result status											
Clock	F0	F2	F4	F6	F8	F10	F12	...	F30	FU	
8										Mult1 Integer Add Divide	

Scoreboard Example Cycle 8b

Instruction status				Read				Executiv Write			
Instruction	j	k		Issue	operand	complete	Result				
LD	F6	34+	R2	1	2	3	4				
LD	F2	45+	R3	5	6	7	8				
MULT	F0	F2	F4	6							
SUBD	F8	F6	F2	7							
DIVD	F10	F0	F6	8							
ADD	F6	F8	F2								

Functional unit status											
Time	Name	Busy	Op	dest	S1	S2	FU for j	FU for k	Fj?	Fk?	Rk?
	Integer	No									
	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes	
	Mult2	No									
	Add	Yes	Sub	F8	F6	F2			Yes	Yes	
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes	

Register result status											
Clock	F0	F2	F4	F6	F8	F10	F12	...	F30	FU	
8										Mult1 Integer Add Divide	

Scoreboard Example Cycle 9

Instruction status				Read		Executiv		Write	
Instruction	j	k		Issue	operand complet	Result			
LD	F6	34+	R2	1	2	3	4		
LD	F2	45+	R3	5	6	7	8		
MULT	F0	F2	F4	6	9				
SUBD	F8	F6	F2	7	9				
DIVD	F10	F0	F6	8					
ADD	F6	F8	F2						

Functional unit status		dest		S1	S2	FU for j		FU for k		Fj?	Fk?
Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk	
No	Integer	No									
10	Mult1	Yes	Mult	F0	F2	F4				Yes	Yes
No	Mult2	No									
2	Add	Yes	Sub	F8	F6	F2				Yes	Yes
Divide	Divide	Yes	Div	F10	F0	F6	Mult1			No	Yes

Register result status		Clock										
		F0	F2	F4	F6	F8	F10	F12	...	F30		
9	FU	Mult1				Add	Divide					

Scoreboard Example Cycle 11

Instruction status				Read		Executiv		Write	
Instruction	j	k		Issue	operand complet	Result			
LD	F6	34+	R2	1	2	3	4		
LD	F2	45+	R3	5	6	7	8		
MULT	F0	F2	F4	6	9				
SUBD	F8	F6	F2	7	9	11			
DIVD	F10	F0	F6	8					
ADD	F6	F8	F2						

Functional unit status		dest		S1	S2	FU for j		FU for k		Fj?	Fk?
Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk	
No	Integer	No									
8	Mult1	Yes	Mult	F0	F2	F4				Yes	Yes
No	Mult2	No									
0	Add	Yes	Sub	F8	F6	F2				Yes	Yes
Divide	Divide	Yes	Div	F10	F0	F6	Mult1			No	Yes

Register result status		Clock										
		F0	F2	F4	F6	F8	F10	F12	...	F30		
11	FU	Mult1				Add	Divide					

Scoreboard Example Cycle 12

Instruction status				Read		Executiv		Write	
Instruction	j	k		Issue	operand complet	Result			
LD	F6	34+	R2	1	2	3	4		
LD	F2	45+	R3	5	6	7	8		
MULT	F0	F2	F4	6	9				
SUBD	F8	F6	F2	7	9	11	12		
DIVD	F10	F0	F6	8					
ADD	F6	F8	F2						

Functional unit status		dest		S1	S2	FU for j		FU for k		Fj?	Fk?
Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk	
No	Integer	No									
7	Mult1	Yes	Mult	F0	F2	F4				Yes	Yes
No	Mult2	No									
No	Add	No									
Divide	Divide	Yes	Div	F10	F0	F6	Mult1			No	Yes

Register result status		Clock										
		F0	F2	F4	F6	F8	F10	F12	...	F30		
12	FU	Mult1				Divide						

- Read operands for DIVD?

Scoreboard Example Cycle 13

Instruction status				Read		Executiv		Write	
Instruction	j	k		Issue	operand complet	Result			
LD	F6	34+	R2	1	2	3	4		
LD	F2	45+	R3	5	6	7	8		
MULT	F0	F2	F4	6	9				
SUBD	F8	F6	F2	7	9	11	12		
DIVD	F10	F0	F6	8					
ADD	F6	F8	F2						

Functional unit status		dest		S1	S2	FU for j		FU for k		Fj?	Fk?
Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk	
No	Integer	No									
6	Mult1	Yes	Mult	F0	F2	F4				Yes	Yes
No	Mult2	No									
Yes	Add	Yes	Add	F6	F8	F2				Yes	Yes
Divide	Divide	Yes	Div	F10	F0	F6	Mult1			No	Yes

Register result status		Clock										
		F0	F2	F4	F6	F8	F10	F12	...	F30		
13	FU	Mult1			Add	Divide						

Scoreboard Example Cycle 14

Instruction status				Read		Executiv		Write	
Instruction	j	k		Issue	operand complet	Result			
LD	F6	34+	R2	1	2	3	4		
LD	F2	45+	R3	5	6	7	8		
MULT	F0	F2	F4	6	9				
SUBD	F8	F6	F2	7	9	11	12		
DIVD	F10	F0	F6	8					
ADD	F6	F8	F2						

Functional unit status		dest		S1	S2	FU for j		FU for k		Fj?	Fk?
Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk	
No	Integer	No									
5	Mult1	Yes	Mult	F0	F2	F4				Yes	Yes
No	Mult2	No									
2	Add	Yes	Add	F6	F8	F2				Yes	Yes
Divide	Divide	Yes	Div	F10	F0	F6	Mult1			No	Yes

Register result status		Clock										
		F0	F2	F4	F6	F8	F10	F12	...	F30		
14	FU	Mult1			Add	Divide						

Scoreboard Example Cycle 15

Instruction status				Read		Executiv		Write	
Instruction	j	k		Issue	operand complet	Result			
LD	F6	34+	R2	1	2	3	4		
LD	F2	45+	R3	5	6	7	8		
MULT	F0	F2	F4	6	9				
SUBD	F8	F6	F2	7	9	11	12		
DIVD	F10	F0	F6	8					
ADD	F6	F8	F2						

Functional unit status		dest		S1	S2	FU for j		FU for k		Fj?	Fk?
Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk	
No	Integer	No									
4	Mult1	Yes	Mult	F0	F2	F4				Yes	Yes
No	Mult2	No									
1	Add	Yes	Add	F6	F8	F2				Yes	Yes
Divide	Divide	Yes	Div	F10	F0	F6	Mult1			No	Yes

Register result status		Clock										
		F0	F2	F4	F6	F8	F10	F12	...	F30		
15	FU	Mult1			Add	Divide						

Scoreboard Example Cycle 16

Instruction status				Read		Executiv		Write	
Instruction	j	k		Issue	operand	complete	Result		
LD	F6	34+	R2	1	2	3	4		
LD	F2	45+	R3	5	6	7	8		
MULT	F0	F2	F4	6	9				
SUBD	F8	F6	F2	7	9	11	12		
DIVD	F10	F0	F6	8					
ADD	F6	F8	F2	13	14	16			

Functional unit status										
Time Name	Busy	Op	dest	S1	S2	FU for j		FU for k		Fk?
			Fi	Fj	Fk	Qj	Qk	Rj	Rk	
Integer	No									
Mult1	Yes	Mult	F0	F2	F4				Yes	Yes
Mult2	No									
Add	Yes	Add	F6	F8	F2				Yes	Yes
Divide	Yes	Div	F10	F0	F6	Mult1			No	Yes

Register result status									
Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
16	FU		Mult1	Add	Divide				

Scoreboard Example Cycle 17

Instruction status				Read		Executiv		Write	
Instruction	j	k		Issue	operand	complete	Result		
LD	F6	34+	R2	1	2	3	4		
LD	F2	45+	R3	5	6	7	8		
MULT	F0	F2	F4	6	9				
SUBD	F8	F6	F2	7	9	11	12		
DIVD	F10	F0	F6	8					
ADD	F6	F8	F2	13	14	16			

Functional unit status										
Time Name	Busy	Op	dest	S1	S2	FU for j		FU for k		Fk?
			Fi	Fj	Fk	Qj	Qk	Rj	Rk	
Integer	No									
Mult1	Yes	Mult	F0	F2	F4				Yes	Yes
Mult2	No									
Add	Yes	Add	F6	F8	F2				Yes	Yes
Divide	Yes	Div	F10	F0	F6	Mult1			No	Yes

Register result status									
Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
17	FU		Mult1	Add	Divide				

- Write result of ADD?

Scoreboard Example Cycle 18

Instruction status				Read		Executiv		Write	
Instruction	j	k		Issue	operand	complete	Result		
LD	F6	34+	R2	1	2	3	4		
LD	F2	45+	R3	5	6	7	8		
MULT	F0	F2	F4	6	9				
SUBD	F8	F6	F2	7	9	11	12		
DIVD	F10	F0	F6	8					
ADD	F6	F8	F2	13	14	16			

Functional unit status										
Time Name	Busy	Op	dest	S1	S2	FU for j		FU for k		Fk?
			Fi	Fj	Fk	Qj	Qk	Rj	Rk	
Integer	No									
Mult1	Yes	Mult	F0	F2	F4				Yes	Yes
Mult2	No									
Add	Yes	Add	F6	F8	F2				Yes	Yes
Divide	Yes	Div	F10	F0	F6	Mult1			No	Yes

Register result status									
Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
18	FU		Mult1	Add	Divide				

Scoreboard Example Cycle 19

Instruction status				Read		Executiv		Write	
Instruction	j	k		Issue	operand	complete	Result		
LD	F6	34+	R2	1	2	3	4		
LD	F2	45+	R3	5	6	7	8		
MULT	F0	F2	F4	6	9	19	20		
SUBD	F8	F6	F2	7	9	11	12		
DIVD	F10	F0	F6	8					
ADD	F6	F8	F2	13	14	16			

Functional unit status										
Time Name	Busy	Op	dest	S1	S2	FU for j		FU for k		Fk?
			Fi	Fj	Fk	Qj	Qk	Rj	Rk	
Integer	No									
Mult1	Yes	Mult	F0	F2	F4				Yes	Yes
Mult2	No									
Add	Yes	Add	F6	F8	F2				Yes	Yes
Divide	Yes	Div	F10	F0	F6	Mult1			No	Yes

Register result status									
Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
19	FU		Mult1	Add	Divide				

Scoreboard Example Cycle 20

Instruction status				Read		Executiv		Write	
Instruction	j	k		Issue	operand	complete	Result		
LD	F6	34+	R2	1	2	3	4		
LD	F2	45+	R3	5	6	7	8		
MULT	F0	F2	F4	6	9	19	20		
SUBD	F8	F6	F2	7	9	11	12		
DIVD	F10	F0	F6	8					
ADD	F6	F8	F2	13	14	16			

Functional unit status										
Time Name	Busy	Op	dest	S1	S2	FU for j		FU for k		Fk?
			Fi	Fj	Fk	Qj	Qk	Rj	Rk	
Integer	No									
Mult1	No									
Mult2	No									
Add	Yes	Add	F6	F8	F2				Yes	Yes
Divide	Yes	Div	F10	F0	F6				Yes	Yes

Register result status									
Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
20	FU			Add	Divide				

Scoreboard Example Cycle 21

Instruction status				Read		Executiv		Write	
Instruction	j	k		Issue	operand	complete	Result		
LD	F6	34+	R2	1	2	3	4		
LD	F2	45+	R3	5	6	7	8		
MULT	F0	F2	F4	6	9	19	20		
SUBD	F8	F6	F2	7	9	11	12		
DIVD	F10	F0	F6	8	21				
ADD	F6	F8	F2	13	14	16			

Functional unit status										
Time Name	Busy	Op	dest	S1	S2	FU for j		FU for k		Fk?
			Fi	Fj	Fk	Qj	Qk	Rj	Rk	
Integer	No									
Mult1	No									
Mult2	No									
Add	Yes	Add	F6	F8	F2				Yes	Yes
Divide	Yes	Div	F10	F0	F6				Yes	Yes

Register result status									
Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
21	FU			Add	Divide				

Scoreboard Example Cycle 22

Instruction status				Read				Executic Write			
Instruction	j	k		Issue	operand	complete	Result				
LD	F6	34+	R2	1	2	3	4				
LD	F2	45+	R3	5	6	7	8				
MULT	F0	F2	F4	6	9	19	20				
SUBD	F8	F6	F2	7	9	11	12				
DIVD	F10	F0	F6	8	21						
ADDD	F6	F8	F2	13	14	16	22				

Functional unit status										
Time	Name	Busy	Op	dest	S1	S2	FU for j	FU for k	Fj?	Fk?
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
40	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status									
Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
22									

Scoreboard Example Cycle 61

Instruction status				Read				Executic Write			
Instruction	j	k		Issue	operand	complete	Result				
LD	F6	34+	R2	1	2	3	4				
LD	F2	45+	R3	5	6	7	8				
MULT	F0	F2	F4	6	9	19	20				
SUBD	F8	F6	F2	7	9	11	12				
DIVD	F10	F0	F6	8	21						
ADDD	F6	F8	F2	13	14	16	22				

Functional unit status										
Time	Name	Busy	Op	dest	S1	S2	FU for j	FU for k	Fj?	Fk?
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
0	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status									
Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
61									

Scoreboard Example Cycle 62

Instruction status				Read				Executic Write			
Instruction	j	k		Issue	operand	complete	Result				
LD	F6	34+	R2	1	2	3	4				
LD	F2	45+	R3	5	6	7	8				
MULT	F0	F2	F4	6	9	19	20				
SUBD	F8	F6	F2	7	9	11	12				
DIVD	F10	F0	F6	8	21	61	62				
ADDD	F6	F8	F2	13	14	16	22				

Functional unit status										
Time	Name	Busy	Op	dest	S1	S2	FU for j	FU for k	Fj?	Fk?
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
0	Divide	No								

Register result status									
Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
62									

CDC 6000 Scoreboard Summary

- **Speedup from scoreboard**
 - 1.7 for FORTRAN programs
 - 2.5 for hand-coded assembly language programs
 - Effects of modern compilers?
- **Hardware**
 - Scoreboard hardware approximately same as one FPU
 - Main cost was buses (4x's normal amount)
 - Could be more severe for modern processors
- **Limitations**
 - No forwarding logic
 - Limited to instructions in basic block
 - Stalls for WAW hazards
 - Wait for WAR hazards before WB

Tomasulo Algorithm for Dynamic Scheduling

- For IBM 360/91 in 1967 - about 3 years after CDC 6600
- **Goal: High performance without special compilers**
- **Differences between IBM 360 & CDC 6600**
 - IBM has only 2 register specifiers/instr vs. 3 in CDC 6600
 - IBM has register-memory instructions
 - IBM has 4 FP registers vs. 8 in CDC 6600
 - IBM has pipelined functional units (3 adds, 2 multiplies)
- **Tomasulo algorithm is designed to handle name dependencies (WAW and WAR hazards) efficiently**

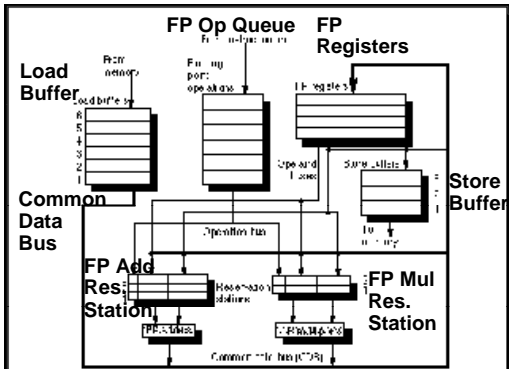
```

SUB    F1, F2, F0
DIVF  F2, F3, F2
ADDF  F3, F0, F0
MULF F3, F1, F1
        
```

Tomasulo Algorithm

- **Differences from Scoreboarding**
 - Distributed hazard detection and control (through **reservation stations**)
 - Results are bypassed to function units
 - Common data bus (CDB) broadcasts results to all FUs.
 - HW renaming of registers to avoid WAR, WAW hazards
 - Load and Stores treated as FUs as well
 - Registers in instructions replaced by pointers to reservation station buffers
- Lead to concepts used in Alpha 21264, HP 8000, MIPS 10000, Pentium II, PowerPC 604, ...

Tomasulo Organization



Reservation Station Components

- Op**—Operation to perform in the unit (e.g., + or -)
- Qj, Qk**—Reservation stations producing source registers
- Vj, Vk**—Value of Source operands
- Rj, Rk**—Flags indicating when Vj, Vk are ready
- Busy**—Indicates reservation station and FU is busy

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

Three Stages of Tomasulo Algorithm

- 1. Issue**—get instruction from FP Op Queue
If reservation station free, the algorithm issues instr & sends operands (renames registers).
- 2. Execution**—operate on operands (EX)
When both operands ready then execute; if not ready, watch CDB for result
- 3. Write result**—finish execution (WB)
Write on Common Data Bus to all awaiting units; mark reservation station available.

Tomasulo Example Cycle 0

Instruction status				Execution		Write		Busy		Address	
Instruction	j	k		Issue	complete	Result			Load1	Load2	Load3
LD	F6	34+	R2						No	No	No
LD	F2	45+	R3						No	No	No
MULT	F0	F2	F4						No	No	No
SUBD	F8	F6	F2						No	No	No
DIVD	F10	F0	F6						No	No	No
ADD	F6	F8	F2						No	No	No

Reservation Stations		S1		S2		RS for j		RS for k	
Time	Name	Busy	Op	Vj	Vk	Qj	Qk		
0	Add1	No							
0	Add2	No							
0	Add3	No							
0	Mult1	No							
0	Mult2	No							

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

Tomasulo Example Cycle 1

Instruction status				Execution		Write		Busy		Address	
Instruction	j	k		Issue	complete	Result			Load1	Load2	Load3
LD	F6	34+	R2	1					Yes	No	No
LD	F2	45+	R3						No	No	No
MULT	F0	F2	F4						No	No	No
SUBD	F8	F6	F2						No	No	No
DIVD	F10	F0	F6						No	No	No
ADD	F6	F8	F2						No	No	No

Reservation Stations		S1		S2		RS for j		RS for k	
Time	Name	Busy	Op	Vj	Vk	Qj	Qk		
0	Add1	No							
0	Add2	No							
0	Add3	No							
0	Mult1	No							
0	Mult2	No							

Register result status		FU											
Clock		F0	F2	F4	F6	F8	F10	F12	...	F30			
1													

Tomasulo Example Cycle 2

Instruction status				Execution		Write		Busy		Address	
Instruction	j	k		Issue	complete	Result			Load1	Load2	Load3
LD	F6	34+	R2	1					Yes	No	No
LD	F2	45+	R3	2					Yes	45+R3	No
MULT	F0	F2	F4						No	No	No
SUBD	F8	F6	F2						No	No	No
DIVD	F10	F0	F6						No	No	No
ADD	F6	F8	F2						No	No	No

Reservation Stations		S1		S2		RS for j		RS for k	
Time	Name	Busy	Op	Vj	Vk	Qj	Qk		
0	Add1	No							
0	Add2	No							
0	Add3	No							
0	Mult1	No							
0	Mult2	No							

Register result status		FU											
Clock		F0	F2	F4	F6	F8	F10	F12	...	F30			
2			Load2										

Tomasulo Example Cycle 3

Instruction status		Execution			Write			
Instruction	j	k	Issue	complete	Result	Busy	Address	
LD F6	34+	R2	1	3		Load1	Yes 34+R2	
LD F2	45+	R3	2			Load2	Yes 45+R3	
MULT F0	F2	F4	3			Load3	No	
SUBD F8	F6	F2						
DIVD F10	F0	F6						
ADD F8	F8	F2						

Reservation Stations		S1		S2		RS for j		RS for k	
Time	Name	Busy	Op	Vj	Vk	Oj	Ok		
0	Add1	No							
0	Add2	No							
0	Add3	No							
0	Mult1	Yes	MULTD		R(F4)		Load2		
0	Mult2	No							

Register result status												
Clock		F0	F2	F4	F6	F8	F10	F12	...	F30		
3	FU		Mult1	Load2		Load1						

Tomasulo Example Cycle 4

Instruction status		Execution			Write			
Instruction	j	k	Issue	complete	Result	Busy	Address	
LD F6	34+	R2	1	3	4	Load1	No	
LD F2	45+	R3	2			Load2	Yes 45+R3	
MULT F0	F2	F4	3			Load3	No	
SUBD F8	F6	F2	4					
DIVD F10	F0	F6	5					
ADD F8	F8	F2	6					

Reservation Stations		S1		S2		RS for j		RS for k	
Time	Name	Busy	Op	Vj	Vk	Oj	Ok		
0	Add1	Yes	SUBD		M(34+R2)		Load2		
0	Add2	No							
0	Add3	No							
0	Mult1	Yes	MULTD		R(F4)		Load2		
0	Mult2	No							

Register result status												
Clock		F0	F2	F4	F6	F8	F10	F12	...	F30		
4	FU		Mult1	Load2		M(34+R2)	Add1					

Tomasulo Example Cycle 5

Instruction status		Execution			Write			
Instruction	j	k	Issue	complete	Result	Busy	Address	
LD F6	34+	R2	1	3	4	Load1	No	
LD F2	45+	R3	2	5		Load2	Yes 45+R3	
MULT F0	F2	F4	3			Load3	No	
SUBD F8	F6	F2	4					
DIVD F10	F0	F6	5					
ADD F8	F8	F2	6					

Reservation Stations		S1		S2		RS for j		RS for k	
Time	Name	Busy	Op	Vj	Vk	Oj	Ok		
0	Add1	Yes	SUBD		M(34+R2)		Load2		
0	Add2	No							
0	Add3	No							
0	Mult1	Yes	MULTD		R(F4)		Load2		
0	Mult2	Yes	DIVD		M(34+R2)		Mult1		

Register result status												
Clock		F0	F2	F4	F6	F8	F10	F12	...	F30		
5	FU		Mult1	Load2		M(34+R2)	Add1	Mult2				

Tomasulo Example Cycle 6

Instruction status		Execution			Write			
Instruction	j	k	Issue	complete	Result	Busy	Address	
LD F6	34+	R2	1	3	4	Load1	No	
LD F2	45+	R3	2	5	6	Load2	No	
MULT F0	F2	F4	3			Load3	No	
SUBD F8	F6	F2	4					
DIVD F10	F0	F6	5					
ADD F8	F8	F2	6					

Reservation Stations		S1		S2		RS for j		RS for k	
Time	Name	Busy	Op	Vj	Vk	Oj	Ok		
2	Add1	Yes	SUBD		M(34+R2)		M(45+R3)		
0	Add2	Yes	ADD		M(45+R3)		Add1		
0	Add3	No							
10	Mult1	Yes	MULTD		M(45+R3)		R(F4)		
0	Mult2	Yes	DIVD		M(34+R2)		Mult1		

Register result status												
Clock		F0	F2	F4	F6	F8	F10	F12	...	F30		
6	FU		Mult1	M(45+R3)		Add2	Add1	Mult2				

Tomasulo Example Cycle 7

Instruction status		Execution			Write			
Instruction	j	k	Issue	complete	Result	Busy	Address	
LD F6	34+	R2	1	3	4	Load1	No	
LD F2	45+	R3	2	5	6	Load2	No	
MULT F0	F2	F4	3			Load3	No	
SUBD F8	F6	F2	4					
DIVD F10	F0	F6	5					
ADD F8	F8	F2	6					

Reservation Stations		S1		S2		RS for j		RS for k	
Time	Name	Busy	Op	Vj	Vk	Oj	Ok		
1	Add1	Yes	SUBD		M(34+R2)		M(45+R3)		
0	Add2	Yes	ADD		M(45+R3)		Add1		
0	Add3	No							
9	Mult1	Yes	MULTD		M(45+R3)		R(F4)		
0	Mult2	Yes	DIVD		M(34+R2)		Mult1		

Register result status												
Clock		F0	F2	F4	F6	F8	F10	F12	...	F30		
7	FU		Mult1	M(45+R3)		Add2	Add1	Mult2				

Tomasulo Example Cycle 8

Instruction status		Execution			Write			
Instruction	j	k	Issue	complete	Result	Busy	Address	
LD F6	34+	R2	1	3	4	Load1	No	
LD F2	45+	R3	2	5	6	Load2	No	
MULT F0	F2	F4	3			Load3	No	
SUBD F8	F6	F2	4					
DIVD F10	F0	F6	5					
ADD F8	F8	F2	6					

Reservation Stations		S1		S2		RS for j		RS for k	
Time	Name	Busy	Op	Vj	Vk	Oj	Ok		
0	Add1	Yes	SUBD		M(34+R2)		M(45+R3)		
0	Add2	Yes	ADD		M(45+R3)		Add1		
0	Add3	No							
8	Mult1	Yes	MULTD		M(45+R3)		R(F4)		
0	Mult2	Yes	DIVD		M(34+R2)		Mult1		

Register result status												
Clock		F0	F2	F4	F6	F8	F10	F12	...	F30		
8	FU		Mult1	M(45+R3)		Add2	Add1	Mult2				

Tomasulo Example Cycle 9

Instruction status		Execution			Write		Busy		Address	
Instruction	j	k	Issue	complete	Result	Load1	Load2	Load3	Busy	Address
LD	F6	34+	R2	1	3	4			No	
LD	F2	45+	R3	2	5	6			No	
MULT	F0	F2	F4	3					No	
SUBD	F8	F6	F2	4	8	9				
DIVD	F10	F0	F6	5						
ADD	F6	F8	F2	6						

Reservation Stations		S1		S2		RS for j		RS for k	
Time	Name	Busy	Op	Vj	Vk	Oj	Ok	Ok	Ok
0	Add1	No							
0	Add2	Yes	ADD	M()–M()	M(45+R3)				
0	Add3	No							
7	Mult1	Yes	MULT	M(45+R3)	R(F4)				
0	Mult2	Yes	DIVD	M(34+R2)	Mult1				

Register result status											
Clock	FU	F0	F2	F4	F6	F8	F10	F12	...	F30	
9		Mult1	M(45+R3)		Add2	M()–M()	Mult2				

Tomasulo Example Cycle 10

Instruction status		Execution			Write		Busy		Address	
Instruction	j	k	Issue	complete	Result	Load1	Load2	Load3	Busy	Address
LD	F6	34+	R2	1	3	4			No	
LD	F2	45+	R3	2	5	6			No	
MULT	F0	F2	F4	3					No	
SUBD	F8	F6	F2	4	8	9				
DIVD	F10	F0	F6	5						
ADD	F6	F8	F2	6						

Reservation Stations		S1		S2		RS for j		RS for k	
Time	Name	Busy	Op	Vj	Vk	Oj	Ok	Ok	Ok
0	Add1	No							
2	Add2	Yes	ADD	M()–M()	M(45+R3)				
0	Add3	No							
6	Mult1	Yes	MULT	M(45+R3)	R(F4)				
0	Mult2	Yes	DIVD	M(34+R2)	Mult1				

Register result status											
Clock	FU	F0	F2	F4	F6	F8	F10	F12	...	F30	
10		Mult1	M(45+R3)		Add2	M()–M()	Mult2				

Tomasulo Example Cycle 11

Instruction status		Execution			Write		Busy		Address	
Instruction	j	k	Issue	complete	Result	Load1	Load2	Load3	Busy	Address
LD	F6	34+	R2	1	3	4			No	
LD	F2	45+	R3	2	5	6			No	
MULT	F0	F2	F4	3					No	
SUBD	F8	F6	F2	4	8	9				
DIVD	F10	F0	F6	5						
ADD	F6	F8	F2	6						

Reservation Stations		S1		S2		RS for j		RS for k	
Time	Name	Busy	Op	Vj	Vk	Oj	Ok	Ok	Ok
0	Add1	No							
1	Add2	Yes	ADD	M()–M()	M(45+R3)				
0	Add3	No							
5	Mult1	Yes	MULT	M(45+R3)	R(F4)				
0	Mult2	Yes	DIVD	M(34+R2)	Mult1				

Register result status											
Clock	FU	F0	F2	F4	F6	F8	F10	F12	...	F30	
11		Mult1	M(45+R3)		Add2	M()–M()	Mult2				

Tomasulo Example Cycle 12

Instruction status		Execution			Write		Busy		Address	
Instruction	j	k	Issue	complete	Result	Load1	Load2	Load3	Busy	Address
LD	F6	34+	R2	1	3	4			No	
LD	F2	45+	R3	2	5	6			No	
MULT	F0	F2	F4	3					No	
SUBD	F8	F6	F2	4	8	9				
DIVD	F10	F0	F6	5						
ADD	F6	F8	F2	6	12	13				

Reservation Stations		S1		S2		RS for j		RS for k	
Time	Name	Busy	Op	Vj	Vk	Oj	Ok	Ok	Ok
0	Add1	No							
0	Add2	Yes	ADD	M()–M()	M(45+R3)				
0	Add3	No							
4	Mult1	Yes	MULT	M(45+R3)	R(F4)				
0	Mult2	Yes	DIVD	M(34+R2)	Mult1				

Register result status											
Clock	FU	F0	F2	F4	F6	F8	F10	F12	...	F30	
12		Mult1	M(45+R3)		Add2	M()–M()	Mult2				

Tomasulo Example Cycle 13

Instruction status		Execution			Write		Busy		Address	
Instruction	j	k	Issue	complete	Result	Load1	Load2	Load3	Busy	Address
LD	F6	34+	R2	1	3	4			No	
LD	F2	45+	R3	2	5	6			No	
MULT	F0	F2	F4	3					No	
SUBD	F8	F6	F2	4	8	9				
DIVD	F10	F0	F6	5						
ADD	F6	F8	F2	6	12	13				

Reservation Stations		S1		S2		RS for j		RS for k	
Time	Name	Busy	Op	Vj	Vk	Oj	Ok	Ok	Ok
0	Add1	No							
0	Add2	No							
0	Add3	No							
3	Mult1	Yes	MULT	M(45+R3)	R(F4)				
0	Mult2	Yes	DIVD	M(34+R2)	Mult1				

Register result status											
Clock	FU	F0	F2	F4	F6	F8	F10	F12	...	F30	
13		Mult1	M(45+R3)		(M–M)+M()	M()–M()	Mult2				

Tomasulo Example Cycle 14

Instruction status		Execution			Write		Busy		Address	
Instruction	j	k	Issue	complete	Result	Load1	Load2	Load3	Busy	Address
LD	F6	34+	R2	1	3	4			No	
LD	F2	45+	R3	2	5	6			No	
MULT	F0	F2	F4	3					No	
SUBD	F8	F6	F2	4	8	9				
DIVD	F10	F0	F6	5						
ADD	F6	F8	F2	6	12	13				

Reservation Stations		S1		S2		RS for j		RS for k	
Time	Name	Busy	Op	Vj	Vk	Oj	Ok	Ok	Ok
0	Add1	No							
0	Add2	No							
0	Add3	No							
2	Mult1	Yes	MULT	M(45+R3)	R(F4)				
0	Mult2	Yes	DIVD	M(34+R2)	Mult1				

Register result status											
Clock	FU	F0	F2	F4	F6	F8	F10	F12	...	F30	
14		Mult1	M(45+R3)		(M–M)+M()	M()–M()	Mult2				

Tomasulo Example Cycle 15

Instruction status		Execution			Write		Busy		Address	
Instruction	j	k	Issue	complete	Result	Load1	Load2	Load3	Busy	Address
LD	F6	34+	R2	1	3	4			No	
LD	F2	45+	R3	2	5	6			No	
MULT	F0	F2	F4	3	16	17			No	
SUBD	F8	F6	F2	4	8	9			No	
DIVD	F10	F0	F6	5					No	
ADD	F6	F8	F2	6	12	13			No	

Reservation Stations		S1	S2	RS for j	RS for k		
Time	Name	Busy	Op	Vj	Vk	Oj	Ok
0	Add1	No					
0	Add2	No					
0	Add3	No					
1	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
0	Mult2	Yes	DIVD	M(34+R2)	Mult1		

Register result status		F0	F2	F4	F6	F8	F10	F12	...	F30
Clock	15	FU	Mult1	M(45+R3)	(M-M)+M)	M(34+R2)	M(34+R2)	M(34+R2)	M(34+R2)	M(34+R2)

Tomasulo Example Cycle 16

Instruction status		Execution			Write		Busy		Address	
Instruction	j	k	Issue	complete	Result	Load1	Load2	Load3	Busy	Address
LD	F6	34+	R2	1	3	4			No	
LD	F2	45+	R3	2	5	6			No	
MULT	F0	F2	F4	3	16	17			No	
SUBD	F8	F6	F2	4	8	9			No	
DIVD	F10	F0	F6	5					No	
ADD	F6	F8	F2	6	12	13			No	

Reservation Stations		S1	S2	RS for j	RS for k		
Time	Name	Busy	Op	Vj	Vk	Oj	Ok
0	Add1	No					
0	Add2	No					
0	Add3	No					
0	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
0	Mult2	Yes	DIVD	M(34+R2)	Mult1		

Register result status		F0	F2	F4	F6	F8	F10	F12	...	F30
Clock	16	FU	Mult1	M(45+R3)	(M-M)+M)	M(34+R2)	M(34+R2)	M(34+R2)	M(34+R2)	M(34+R2)

Tomasulo Example Cycle 17

Instruction status		Execution			Write		Busy		Address	
Instruction	j	k	Issue	complete	Result	Load1	Load2	Load3	Busy	Address
LD	F6	34+	R2	1	3	4			No	
LD	F2	45+	R3	2	5	6			No	
MULT	F0	F2	F4	3	16	17			No	
SUBD	F8	F6	F2	4	8	9			No	
DIVD	F10	F0	F6	5					No	
ADD	F6	F8	F2	6	12	13			No	

Reservation Stations		S1	S2	RS for j	RS for k		
Time	Name	Busy	Op	Vj	Vk	Oj	Ok
0	Add1	No					
0	Add2	No					
0	Add3	No					
0	Mult1	No					
0	Mult2	Yes	DIVD	M*F4	M(34+R2)		

Register result status		F0	F2	F4	F6	F8	F10	F12	...	F30
Clock	17	FU	M*F4	M(45+R3)	(M-M)+M)	M(34+R2)	M(34+R2)	M(34+R2)	M(34+R2)	M(34+R2)

Tomasulo Example Cycle 18

Instruction status		Execution			Write		Busy		Address	
Instruction	j	k	Issue	complete	Result	Load1	Load2	Load3	Busy	Address
LD	F6	34+	R2	1	3	4			No	
LD	F2	45+	R3	2	5	6			No	
MULT	F0	F2	F4	3	16	17			No	
SUBD	F8	F6	F2	4	8	9			No	
DIVD	F10	F0	F6	5					No	
ADD	F6	F8	F2	6	12	13			No	

Reservation Stations		S1	S2	RS for j	RS for k		
Time	Name	Busy	Op	Vj	Vk	Oj	Ok
0	Add1	No					
0	Add2	No					
0	Add3	No					
0	Mult1	No					
40	Mult2	Yes	DIVD	M*F4	M(34+R2)		

Register result status		F0	F2	F4	F6	F8	F10	F12	...	F30
Clock	18	FU	M*F4	M(45+R3)	(M-M)+M)	M(34+R2)	M(34+R2)	M(34+R2)	M(34+R2)	M(34+R2)

Tomasulo Example Cycle 57

Instruction status		Execution			Write		Busy		Address	
Instruction	j	k	Issue	complete	Result	Load1	Load2	Load3	Busy	Address
LD	F6	34+	R2	1	3	4			No	
LD	F2	45+	R3	2	5	6			No	
MULT	F0	F2	F4	3	16	17			No	
SUBD	F8	F6	F2	4	8	9			No	
DIVD	F10	F0	F6	5					No	
ADD	F6	F8	F2	6	12	13			No	

Reservation Stations		S1	S2	RS for j	RS for k		
Time	Name	Busy	Op	Vj	Vk	Oj	Ok
0	Add1	No					
0	Add2	No					
0	Add3	No					
0	Mult1	No					
1	Mult2	Yes	DIVD	M*F4	M(34+R2)		

Register result status		F0	F2	F4	F6	F8	F10	F12	...	F30
Clock	57	FU	M*F4	M(45+R3)	(M-M)+M)	M(34+R2)	M(34+R2)	M(34+R2)	M(34+R2)	M(34+R2)

Tomasulo Example Cycle 58

Instruction status		Execution			Write		Busy		Address	
Instruction	j	k	Issue	complete	Result	Load1	Load2	Load3	Busy	Address
LD	F6	34+	R2	1	3	4			No	
LD	F2	45+	R3	2	5	6			No	
MULT	F0	F2	F4	3	16	17			No	
SUBD	F8	F6	F2	4	8	9			No	
DIVD	F10	F0	F6	5					No	
ADD	F6	F8	F2	6	12	13			No	

Reservation Stations		S1	S2	RS for j	RS for k		
Time	Name	Busy	Op	Vj	Vk	Oj	Ok
0	Add1	No					
0	Add2	No					
0	Add3	No					
0	Mult1	No					
0	Mult2	Yes	DIVD	M*F4	M(34+R2)		

Register result status		F0	F2	F4	F6	F8	F10	F12	...	F30
Clock	58	FU	M*F4	M(45+R3)	(M-M)+M)	M(34+R2)	M(34+R2)	M(34+R2)	M(34+R2)	M(34+R2)

Tomasulo Example Cycle 59

Instruction	status	j	k	Issue	Execution complete	Write Result	Load1	Busy	Address
LD	F6	34+	R2	1	3	4		No	
LD	F2	45+	R3	2	5	6	Load2	No	
MULT	F0	F2	F4	3	16	17	Load3	No	
SUBD	F8	F6	F2	4	8	9			
DIVD	F10	F0	F6	5	58	59			
ADD	F8	F2	F2	6	12	13			

Reservation Stations	S1	S2	RS for j	RS for k			
Time	Name	Busy	Op	Vj	Vk	Oj	Ok
0	Add1	No					
0	Add2	No					
0	Add3	No					
0	Mult1	No					
0	Mult2	No					

Register result status	F0	F2	F4	F6	F8	F10	F12	...	F30
Clock									
59	FU	M*F4	M(45+R3)	(M-M)+M	M0-M0	M*F4M			

Tomasulo Summary

- **Advantages**
 - Prevents register from being the bottleneck
 - Eliminates WAR, WAW hazards
 - Allows loop unrolling in HW
- **Common Data Bus**
 - Broadcasts results to multiple instructions
 - Central bottleneck
- **Lasting Contributions**
 - Dynamic scheduling
 - Register renaming
 - Load/store disambiguation

Branch Prediction Paper

- The six Fortran benchmarks used in this study include:
 - ADVAN: Solves partial differential equations
 - SCI2: Performs matrix inversion
 - SINCOS: Converts points from polar to Cartesian Coordinates
 - SORTST: Sorts a list of 10,000 integers
 - Gibson: IA program to compile instructions
 - Tblnk: Processes a linked list
- Last four programs are highly unpredictable
- Programs compiled for the CDC Cyber 170 architecture => supercomputer
- Don't distinguish conditional and unconditional (always taken) branches.

Static Branch Prediction Strategies

- **Strategy 1: Predict that all branches are taken.**
 - Accuracy varied from 57.4% to 99.4%, average = 77%
 - High **program sensitivity**
 - Performance can be improved if compiler knows this is the strategy used
- **Strategy 1a: Predict that all branches with certain opcodes will be taken, predict that the others will not be.**
 - Branch if **negative, equal, and greater than or equal** were always predicted as taken
 - All others were predicted as not taken
 - Accuracy varied from 65.4% to 98.5%, average = 87%
 - Results are probably optimistic - why?
 - Performance is again compiler dependent

Static Branch Prediction Strategies

- **Strategy 3: Predict that all backward branches will be taken and that all forward branches will not be taken.**
 - Accuracy varied from 35.2% to 99.2%, average = 80%
 - What is the motivation behind this strategy?
 - Why might this strategy be more difficult to implement than the previous two strategies?
 - Strategy is very program sensitive - when would this technique work well?
 - Performance is much less compiler dependent

Dynamic Branch Prediction Strategies

- **Strategy 2: Predict that a branch will be decided the same way it was on its last execution. If it has not been previously executed, predict that it will be taken.**
 - Accuracy varied from 76.2% to 99.4%, average = 90%
 - Why is this technique not often implemented?
 - When does this strategy do poorly?
 - Less program sensitivity
- **Strategy 4: Maintain a table of the m most recent branches not taken. Predict only branches found in the table are not taken.**
 - Requires a search of m-value (i.e., m parallel compares)
 - Performance dependent on table size
 - Accuracy varied from 61.5% to 98.9%, average = 85% (m=2)
 - Accuracy varied from 76.2% to 98.9%, average = 90% (m=8)

Dynamic Branch Prediction Strategies

- Strategy 5: Maintain a **history bit** in the instruction cache to tell what happened the last time this instruction was executed. Branches predicted based on last execution
 - If a branch has not been executed yet, it is predicted to be taken - by initializing this history bit accordingly.
 - Accuracy varied from 76.1% to 98.1%, average = 90% for a 64-word instruction cache
 - What are advantages and disadvantages of this scheme?
- Strategy 6: Hash the branch address to m bits and access a 2^m word RAM containing history bits. Predict according to the history bits.
 - Accuracy varies from 76.2% to 98.9%, average = 90%, using an XOR hash with 16 word table ($m=4$)
 - What type of hash is used in the book?
 - What are the advantages and disadvantages of this scheme?

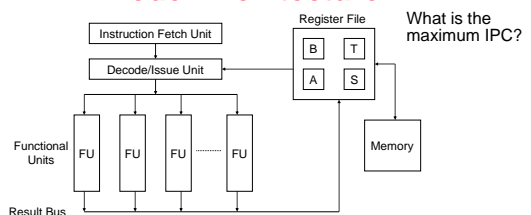
Dynamic Branch Prediction Strategies

- Strategy 7 : Like strategy 6, but uses two's complement counters instead of a single history bit.
 - Increment counter when taken, decrement when not taken
 - Predict not taken if the counter's sign bit is 1
 - Accuracy varies from 80.1% to 99.4%, average = 94%, using a 2-bit counter with 16 word table ($m=4$)
 - Increasing counter size beyond 2 does not help much
 - Where is this scheme likely to be very effective?
- Questions on paper?
 - What are other techniques for branch prediction?
 - What aspects of branch prediction does the paper fail to consider in detail?
 - Are advances in technology likely to change results from the paper? Why or why not?

Instruction Issue Logic for High-Performance Interruptible Pipelined Processors

- Goal: Develop instruction issue logic that
 - Resolves dependencies dynamically
 - Guarantees precise interrupts
 - Does not have significant hardware requirements
- Precise interrupts
 - Instructions before the faulting instruction are allowed to complete
 - Instructions after the faulting instruction can be restarted from scratch - should not change processor state before the exception is taken.
 - Needed to correctly handle page faults
 - add R1, R2, R3
 - mul R1, R2, R4
 - load R5, 200(R6)
 - add R1, R5, R7
 - mul R1, R1, R6

Model Architecture



- The register file consists of
 - 8 Address Registers (A)
 - 64 Address-Save Registers (B)
 - 8 Scalar Registers (S)
 - 64 Scalar-Save Registers (T)
 - Functional units only get values from A and S registers
 - Values in B and T registers can be quickly transferred to A and S registers

Standard Tomasulo's Algorithm

- With standard Tomasulo's algorithm
 - Reservation stations are used to store instructions until all data dependencies have been resolved.
 - Each destination (sink) register is assigned a tag that identifies the result that must be written into it.
 - Since any registers in the register file can be a sink, they all have unique tags.
 - Each register requires associative comparison logic to perform tag-matching (144 tag-matching units for model architecture).

Source Operand 1			Source Operand 2			Destination
Ready	Tag	Contents	Ready	Tag	Contents	Register

Extended Tomasulo's Algorithm

- To reduce the amount of tag-matching, a separate tag unit is used.
 - Only assign tags to sink registers that are currently active
 - Tags from all active registers are consolidated in a Tag Unit (TU)
 - If a source register is busy, the TU provides the current tag for a register to the reservation station.
 - Instruction issue blocks if there are no tags available (i.e., the TU is full).
 - All registers are updated directly by the TU.
- To ensure correction operation
 - Only the latest tag for each register is used by subsequent instructions.
 - Only the latest instruction updates the busy bit of the register
- The reservations now have the following fields

Source Operand 1			Source Operand 2			Destination
Ready	Tag	Contents	Ready	Tag	Contents	TU Slot

Example of the Tag Unit

Tag No.	Register No.	Tag Free	Latest Copy
1	A0	N	Y
2	S0	N	Y
3	NIL (S4)	Y (N)	N/A (Y)
4	S4	N	Y (N)
5	S0	N	N
6	S3	N	Y

I1: Add S4, S0, S7

(1) Get a new tag for S4 (tag no. 3)

(2) S7 has valid contents, so it is read from the register file

(3) S0 does not have valid contents, so it gets tag no. 2

Reservation Station for I1	Source Operand 1		Source Operand 2		Destination		
	No	2	N/A	Yes	N/A	S7	3

Merging the Reservation Stations and TAG Unit

- Instead of having separate reservation stations for each functional unit, they can all be combined into a common RS pool.
 - What are the advantages and disadvantages of this approach?
- Since any entry in the TU either in a functional unit or a reservation station, the TU and RS pool can be combined (RSTU).
- When an instruction issues it obtains a tag from the RSTU and reserves a reservation station.
- An entry for the combined unit is as follows:

Tag Unit Info			Source Operand 1			Source Operand 2			Destination
Tag No.	Tag Free	Latest Copy	Ready	Tag	Contents	Ready	Tag	Contents	Register

Register Update Unit and Precise Interrupts

- To implement precise interrupts, the RSTU is modified so that it commits results in order.
- If the RSTU is implemented as a queue, it is known as a Register Update Unit (RUU).
 - Determines which instructions to issue to function units, reserves result bus, and dispatches instructions
 - Determines which instructions can commit
 - Monitors the result bus to resolve dependencies
 - Provides tags to and accepts in instructions from the decode and issue logic

Modified Architecture with RUU

