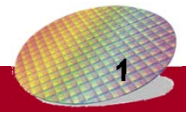
Chapter 3_3 Dynamic Scheduling



Sequential Execution using Simple Pipeline

- Simple Pipeline (in-order issue and execution) Limitation:
  - If an instruction is stalled, later instructions cannot proceed

```
DIV. D   F0, F2, F4
ADD.D    F10, F0, F8     <-- Stalled
SUB.D    F12, F8, F14    <-- Stalled as well
```

  - SUB. D cannot execute because ADD.D depends on DIV.D,
  - Although SUB.D does not depend on anything

- Reason for the stall
  - Both structural and data hazards are checked during ID stage by the control circuit
  - If there are hazards => Stalled

IF	ID	EXE	MEM	WB

```
DIV. D   F0, F2, F4
SUB.D    F12, F8, F14
```

has no dependence

It is possible to execute SUB.D without waiting DIV.D and SUB.D

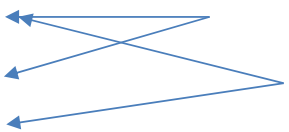
→ Out of order execution:

```
SUB.D    F12, F8, F14
DIV. D   F0, F2, F4
ADD.D    F10, F0, F8
```

Hazard introduced by out-of-order execution

- Out-of-order execution introduces the possibility of WAR and WAW hazard

```
DIV.D   F0, F2, F4
ADD.D   F6, F0, F8
SUB.D   F8, F10, F14
MUL.D   F6, F10, F8
```



Antidependence on F8

Output dependence on F6

- ADD.D & SUB.D -> antidependence, if SUB.D is executed before ADD.D, it will violate antidependence => WAR hazard on F8

```
ADD.D   F6, F0, F8        OOO          SUB.D   F8, F10, F14
SUB.D   F8, F10, F14      Execution    ADD.D   F6, F0, F8
```

WAR hazard on F8

- ADD.D & MUL.D -> output dependence, if MULD is executed before ADD.D, it will violate output dependence => WAW hazard

```
ADD.D   F6, F0, F8        OOO          MUL.D   F6, F10, F8
MUL.D   F6, F10, F8       Execution    ADD.D   F6, F0, F8
```

WAW hazard on F6

Register Renaming to eliminate WAR & WAW

- Use Register Renaming to solve WAR and WAW hazard
- Example:

(1) DIV.D F0,F2,F4
(2) ADD.D F6,F0, F8
(3) S.D F6,0(R1)
(4) SUB.D F8,F10,F14
(5) MUL.D F6,F10,F8

(2) ADD.D F6, F0, F8 (4) SUB.D F8, F10, F14	ADD and SUB Antidependence on F8	WAR hazard
(3) S.D F6, 0(R1) (5) MUL.D F6, F10, F8	S.D and MUL.D Antidependence on F6	WAR hazard
(2) ADD.D F6, F0, F8 (5) MUL.D F6, F10, F8	ADD and MUL Output dependence on F6	WAW hazard

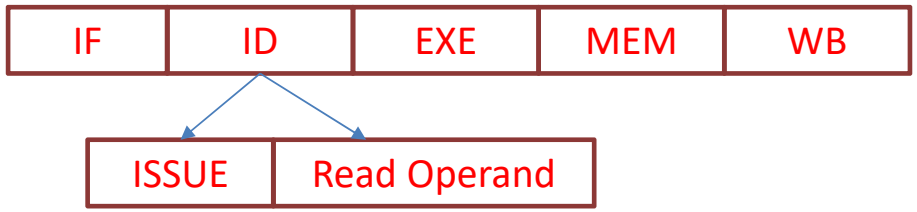
Eliminate WAR&WAW
=> Replace F6 with S and F8 with T

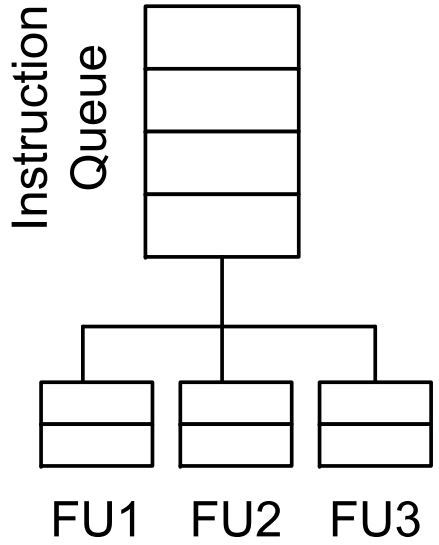
(1) DIV.D F0, F2, F4
(2) ADD.D S, F0, F8
(3) S.D S, 0(R1)
(4) SUB.D T, F10, F14
(5) MUL.D F6, F10, T

(2) ADD.D S, F0, F8 (4) SUB.D T, F10, F14	No WAR hazard
(3) S.D S, 0(R1) (5) MUL.D F6, F10, T	No WAR hazard
(2) ADD.D S, F0, F8 (5) MUL.D F6, F10, T	No WAW hazard

Dynamic Scheduling: Concept

- Dynamic Scheduling: Rearrange order of instructions to reduce stalls while maintaining data flow
- To allow out-of-order execution , Instruction Decode is divided into
  - Issue: Decode instruction, checking for any structural hazard
  - Read operand: Wait for the absence of data hazard

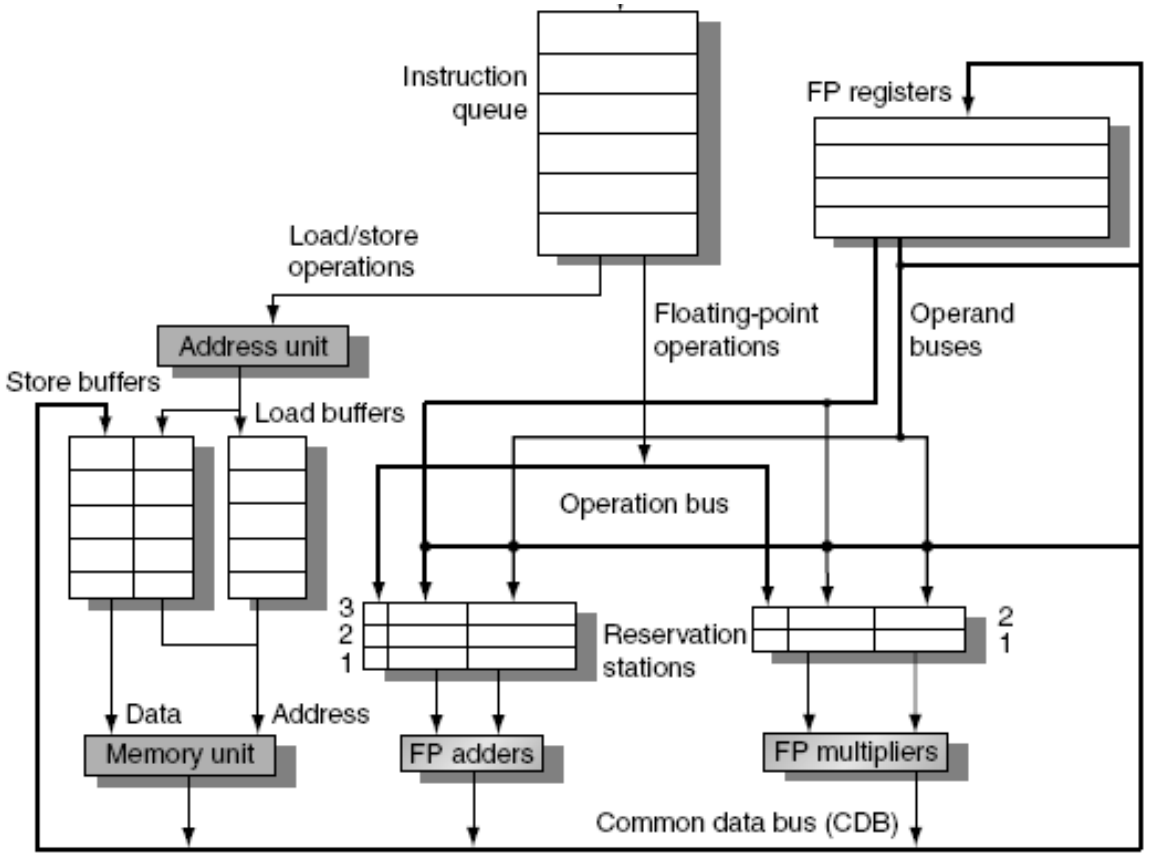




- Requires Multiple functional unit or pipelined functional units:
  - Multiple functional unit or pipelined functional units or both are needed to execute multiple instructions at the same time
  - If data are ready, functional units can start executions

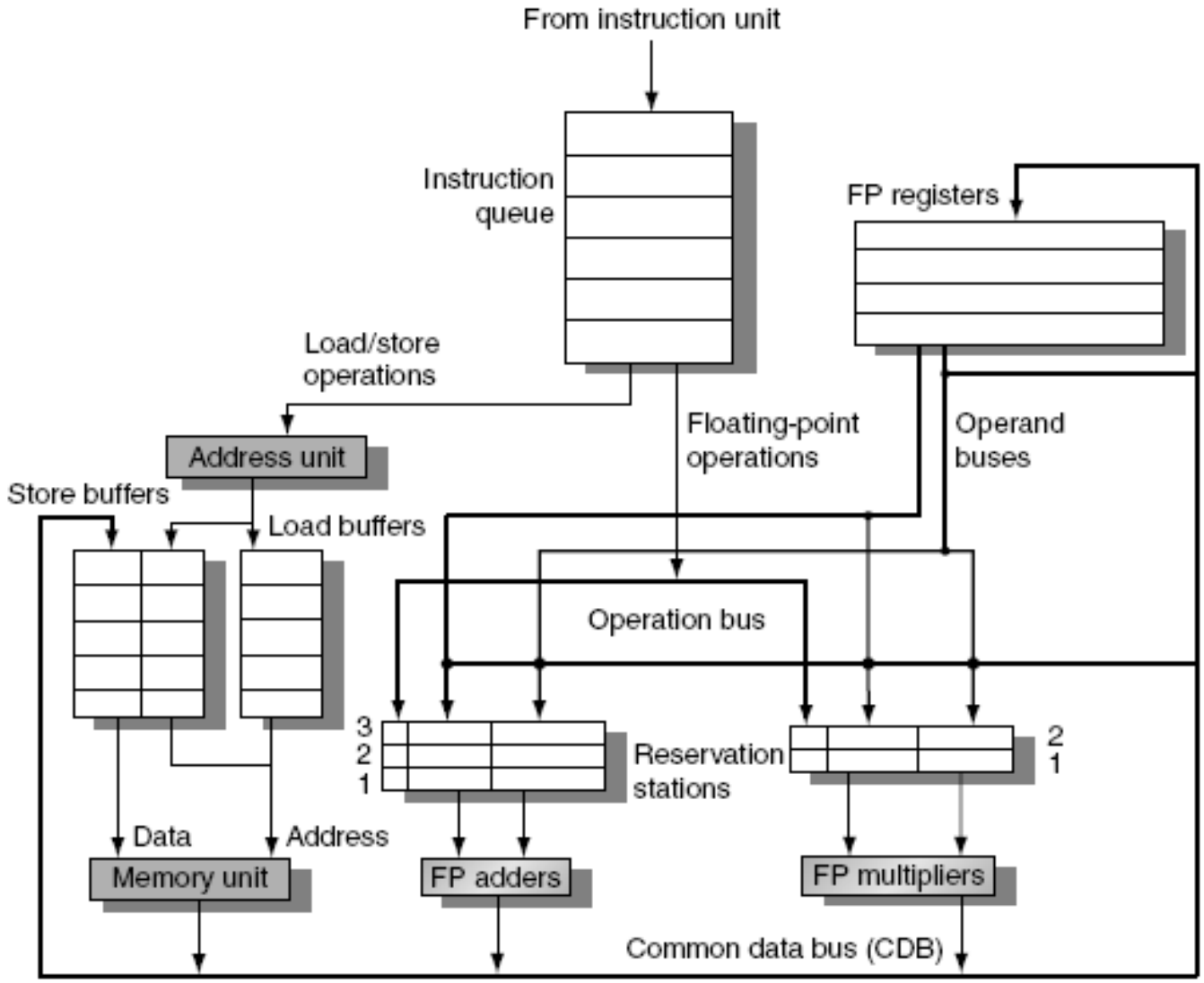
Dynamic Scheduling using Tomasulo's Algorithm

- Two methods: Scoreboarding and Tomasulo method
- Tomasulo is more advanced
- Introduces register renaming in hardware to minimizes WAW and WAR hazards
- Can be extended to handle speculation
- For IBM 360/91 about 3 years after CDC 6600
- Tomasulo's Algorithm
  - Control & buffers (called "reservation stations") distributed with functional units
  - Load and stores treated as functional units
  - Common data bus (CDB) broadcasts results to functional units
  - HW renaming of registers to avoid WAR, WAW hazards
  - Lead to Alpha 21264, HP 8000, MIPS 10000, Pentium III ...



Dynamic scheduling with Tomasulo's Algorithm

- Instructions are from Inst. Unit and stored in Inst. Queue
- Reservation stations fetches and buffers an operand as soon as it is available
  - eliminating the need to get the operand from a register
  - Registers in instructions replaced by pointers to reservation station buffer
- Results from either FP unit or load unit are put on CDB
- Then, results are passed to
  - FP registers
  - Reservation Station
  - Store buffer



Reservation Station and Register result status Components

- Busy – Indicates reservation station and FU is busy
- Op – Operation to perform in the unit (e.g., + or –)
- Q_j, Q_k – Reservation stations producing source registers (if =0, it means operations are available in V_j, V_k)
- V_j, V_k – Value of source operands
- A (Address) – Address. Used to hold memory address for a load or store

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

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

- Register result status – Indicates which functional unit will write each register,

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

Three steps in Tomasulo's Algorithm

- Issue
  - Get next instruction from FIFO queue
  - If there is available RS, issue the instruction to the RS with operand values if available
    - If operand values not available, keep track of the FU that will produce the operands
- Execute
  - When operand becomes available, store it in any reservation stations waiting for it
  - When all operands are ready, execute the instruction
  - No instruction allowed to initiate execution until all branches that proceed it in program order have completed (no speculative execution)
- Write result
  - Write result on CDB into reservation stations and store buffers
    - Stores must wait until address and value are received

See example in next slide

Tomasulo Cycle 0

Instruction status

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2			
LD	F2	45+	R3			
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Assume
Load need 2 clock cycles
Add need 2 clock cycles
Mult need 10 clock cycles
Div need 40 clock cycles

Reservation Stations

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

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

Tomasulo Cycle 1

Instruction status

Instruction		j	k	Issue	Exe complete	Write Result
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
Load1	Yes	34+R2
Load2	No	
Load3	No	

Load1 is set to busy

Reservation Stations

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

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

F6 stores the results of Load1

Tomasulo Cycle 2

Instruction status

Instruction		j	k	Issue	Exe complete	Write Result
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
Load1	Yes	34+R2
Load2	Yes	45+R3
Load3	No	

Load2 is set to busy

Reservation Stations

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					

Can have multiple loads outstanding

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
2	FU		Load2		Load1					

F2 stores the results of Load2

Registers names are removed ("renamed") in Reservation Stations

Tomasulo Cycle 3

MUL is issued to RS (Mult1)

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	
LD	F2	45+	R3	2		
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

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

Load1 completing; what is waiting for Load1?

F6

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	Yes	MULTD		R(F4)	Load2	
0	Mult2	No					

← Mult1 wait for Load2's results

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

F0 stores the results of Mult1

Tomasulo Cycle 4

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	Yes	45+R3
Load3	No	

SUBD is issued to Add1. Wait for Load2

Load1 is completed

Load2 completing; what is waiting for it?

Mult1 & Add1(=SUBD) & F2

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

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

F8 stores the results of Add1

Tomasulo Cycle 5

DIVD is issued to Mult2. It is waiting for the results of Mult1

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Mult1 & Add1(=SUBD) start executing

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

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
5	FU	Mult1	M(45+R3)		M(34+R2)	Add1	Mult2			

Results of Load2 is put into F2

F10 is waiting for the result of Mult2

Tomasulo Cycle 6

ADDD is issued to Add2. It is waiting for the result of Add1

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Mult1 keep executing

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

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

F6 stores the results of Add2

Tomasulo Cycle 7

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

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

Mult1 keep executing

Add1 is completing; what is waiting for it?

Add2 and F8

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

Tomasulo Cycle 8

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

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

Add2 start execution

Mult1 keep executing

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

F8 stores the result of Add1

Tomasulo Cycle 9

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Add2 & Mult1 keep executing

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

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

Tomasulo Cycle 10

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

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

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

Add2 is completing; what is waiting for it? **F6**

Tomasulo Cycle 11

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Mult1 keep executing

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					
4	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
0	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
11	FU	Mult1	M(45+R3)		(M-M)+M()	M()-M()	Mult2			

Write result of Add2 to F6

Tomasulo Cycle 12

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Mult1 keep executing

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					
3	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
0	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
12	FU	Mult1	M(45+R3)		(M-M)+M()	M()-M()	Mult2			

- **Note: all quick instructions complete already**

Tomasulo Cycle 13

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Mult1 keep executing

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					
2	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
0	Mult2	Yes	DIVD		M(34+R2)	Mult1	

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

Tomasulo Cycle 14

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Mult1 keep executing

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					
1	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
0	Mult2	Yes	DIVD		M(34+R2)	Mult1	

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

Tomasulo Cycle 15

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	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	Yes	MULTD	M(45+R3)	R(F4)		
0	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Mult1 keep executing

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
15	FU	Mult1	M(45+R3)		(M-M)+M()	M()-M()	Mult2			

Mult1 completing; what is waiting for it? Mult2

Tomasulo Cycle 16

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	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	Yes					
40	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Mult2 start executing

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
16	FU	M*F4	M(45+R3)		(M-M)+M()	M()-M()	Mult2			

- **Note: Just waiting for divide**

Tomasulo Cycle 55

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	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	Yes					
1	Mult2	Yes	DIVD	M*F4	M(34+R2)		

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
55	FU	M*F4	M(45+R3)		(M-M)+M()	M()-M()	Mult2			

Tomasulo Cycle 56

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5	56	
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	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	Yes					
0	Mult2	Yes	DIVD	M*F4	M(34+R2)		

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
56	FU	M*F4	M(45+R3)		(M-M)+M()	M()-M()	Mult2			

Mult2 completing; what is waiting for it? F10

Tomasulo Cycle 57

Instruction		j	k	Issue	Exe complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5	56	57
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	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					

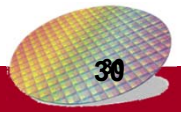
Mult2 start executing

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
57	FU	M*F4	M(45+R3)		(M)+M()	M()-()	M*F4/M			

- **Again, in-order issue, out-of-order execution, completion**

Summary: Dynamic Scheduling using Tomasulo's Algorithm

- Dynamic scheduling implies:
  - In-order issue (to tell the order of instructions)
  - Out-of-order execution
  - Out-of-order completion
- Tomasulo is more advanced (and more complicated) because
  - Introduces register renaming in hardware to minimizes WAW and WAR hazards
  - Can be extended to handle speculation (See Section 3.6)
- Pros
  - Distribution of hazard detection logic because of distributed RS and CDB
  - Elimination of stalls for WAW and WAR hazards because registering renaming using RS and storing operands into RS as soon as they are available
- Cons
  - Complexity: delays of 360/91, MIPS 10000, IBM 620?
  - Need many CDBs at high speed
  - Performance limited by Common Data Bus
    - Multiple CDBs can improve performance, but higher hardware cost



Backup slides