CS510 Computer Architecture

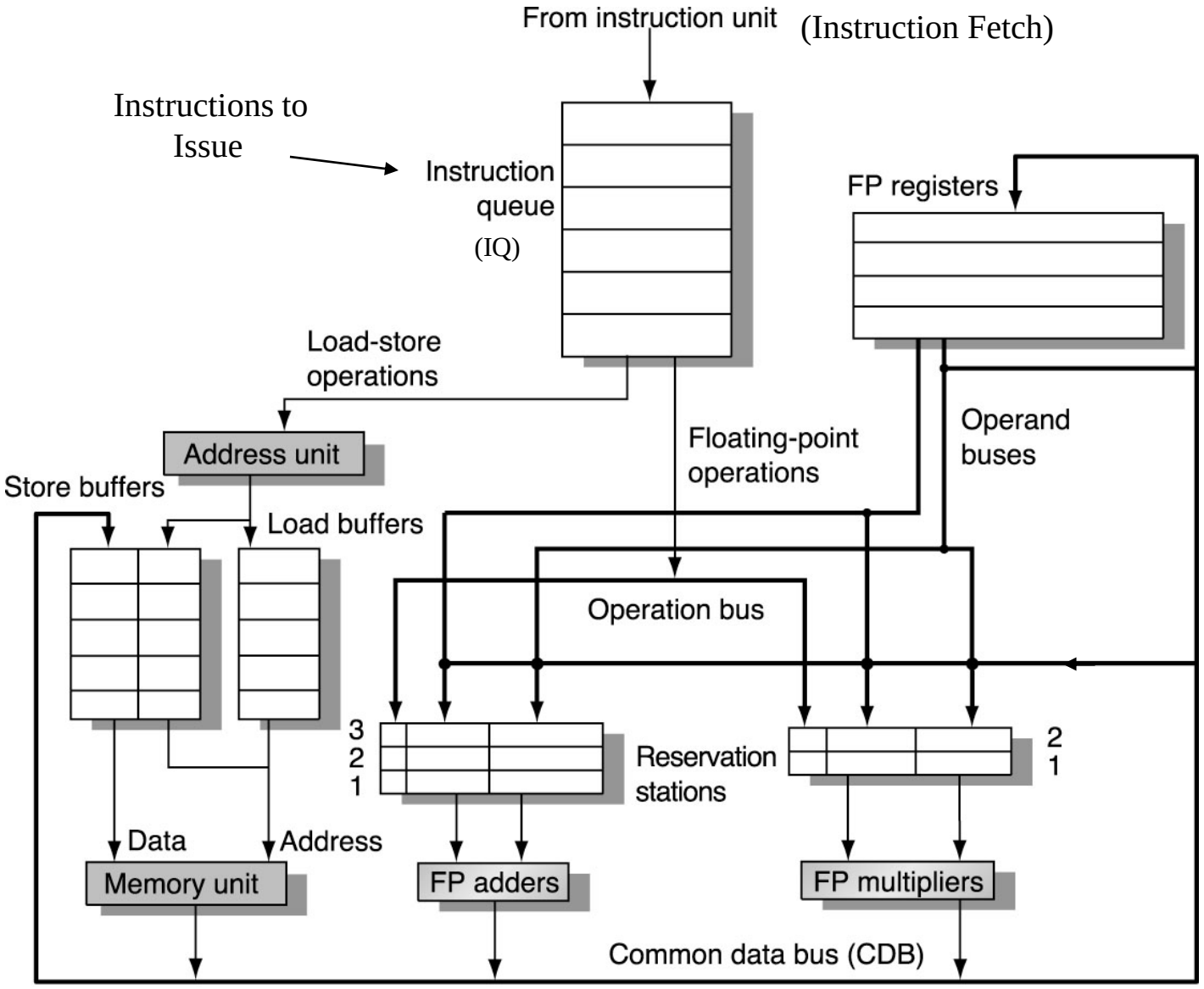
Lecture 10: Dynamic Scheduling II

Soontae Kim

Spring 2017

School of Computing, KAIST

Dynamic Scheduling: The Tomasulo Approach



The basic structure of a MIPS floating-point unit using Tomasulo's algorithm

Reservation Station Fields

- **Op** **Operation to perform in the unit (e.g., + or –)**
- **Vj, Vk** **Values of Source operands S1 and S2**
  - **Store buffers have a single V field indicating result to be stored.** operand가 available하면 vj,vk에 바로 들어가고, 그렇지 않을 경우 Qj,Qk가 set된다.
- **Qj, Qk** **Reservation stations producing source registers.**
  - **No ready flags; Qj,Qk=0 => ready.**
  - **Store buffers only have Qi for RS producing a result.**
- **A:** **Address information for loads or stores. Initially immediate field of instruction then effective address when calculated.**
- **Busy:** **Indicates reservation station is busy.**
- **Register result status:** **Qi Indicates which Reservation Station will write each register, if one exists.**
  - **Blank (or 0) when no pending instruction (i.e. RS) exist that will write to that register.**

Three Stages of Tomasulo Algorithm

1 **Issue:** Get instruction from Instruction Queue (IQ).

- Instruction issued to a free reservation station (RS) (no structural hazard) in FIFO order. if there is not a empty reservation station, then there is a structural hazard and the instruction stalls untill a station or buffer is freed
- Selected RS is marked busy.
- Control sends available instruction operands values (from ISA registers) to the assigned RS.
- Operands not available yet are renamed to RSs that will produce the operand (register renaming). (Dynamic construction of data dependence graph)

Always done in program order

2 **Execution (EX):** Operate on operands.

- When both operands are ready then start executing on assigned FU.
- If all operands are not ready, watch Common Data Bus (CDB) for needed result (forwarding done via CDB). (i.e. wait on any remaining operands, no RAW)

3 **Write result (WB):** Finish execution.

- Write result on Common Data Bus (CDB) to all awaiting units (RSs)
- Mark reservation station as available.

- Normal data bus: data + destination ("go to" bus).
- Common Data Bus (CDB): data + source ("come from" bus):
  - 64 bits for data + 4 bits for source (RSs and load buffers).
  - Write data to waiting RS if source matches expected RS (that produces result).
  - Do the result forwarding via broadcast to waiting RSs.

Can be done out of program order

Including destination register

Tomasulo Algorithm

- **Control & buffers *distributed* with Functional Units (FUs)**
  - **FU buffers are called *"reservation stations"* which have pending instructions and operands and other instruction status info (including data dependences).**
  - **Reservations stations are sometimes referred to as "physical registers" or "renaming registers" as opposed to architecture or ISA registers specified by the ISA.** (internal register) (logical registers)
- **ISA Registers in instructions are replaced by either values (if available) or pointers (renamed) to reservation stations that will supply the value later:**
  - **This process is called *register renaming*.**
    - **Register renaming eliminates WAR, WAW hazards.**
  - **More reservation stations than ISA registers are possible, leading to optimizations that compilers can't achieve and prevents the number of ISA registers from becoming a bottleneck.**
- **Instruction results go (forwarded) from RSs to RSs , *not through registers*, over *Common Data Bus (CDB)* that broadcasts results to all waiting RSs (dependant instructions.**
- **Loads and Stores are treated as FUs with RSs as well.**

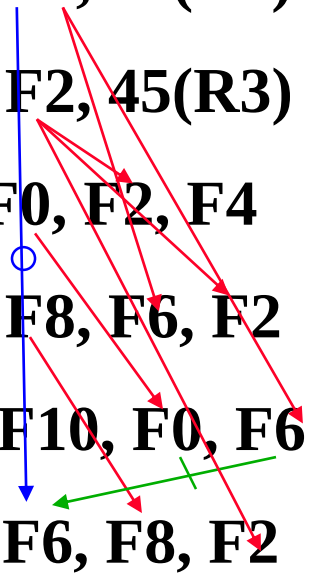
if a load and a store access the same address
1) the load is before the store in program order and interchanging them results in a WAR hazard
2) the store is before the load in program order and interchanging them results in a RAW hazard
3) the store A is before the store A in program order and interchanging them results in a WAR hazard
to determine if a load can be executed at a given time, the processor can check whether any uncompleted store that precedes the load in program order shares the same data memory address as the load. Similarly, a store must wait until there are no unexecuted loads or stores that are earlier in program order and share the same data memory address

to preserve exception behavior, no instruction is allowed to initiate execution until all branches that precede the instruction in program order have completed. this restriction guarantees that an instruction that causes an exception during execution really would have been executed.
this means that the processor must know that the branch prediction was correct before allowing an instruction after the branch to begin execution

Tomasulo Approach Example

	# of RSs	EX Cycles
Integer	1	1
Floating Point Multiply/divide	2	10/40
Floating Point add	3	2

L.D F6, 34(R2)
L.D F2, 45(R3)
MUL. D F0, F2, F4
SUB.D F8, F6, F2
DIV.D F10, F0, F6
ADD.D F6, F8, F2

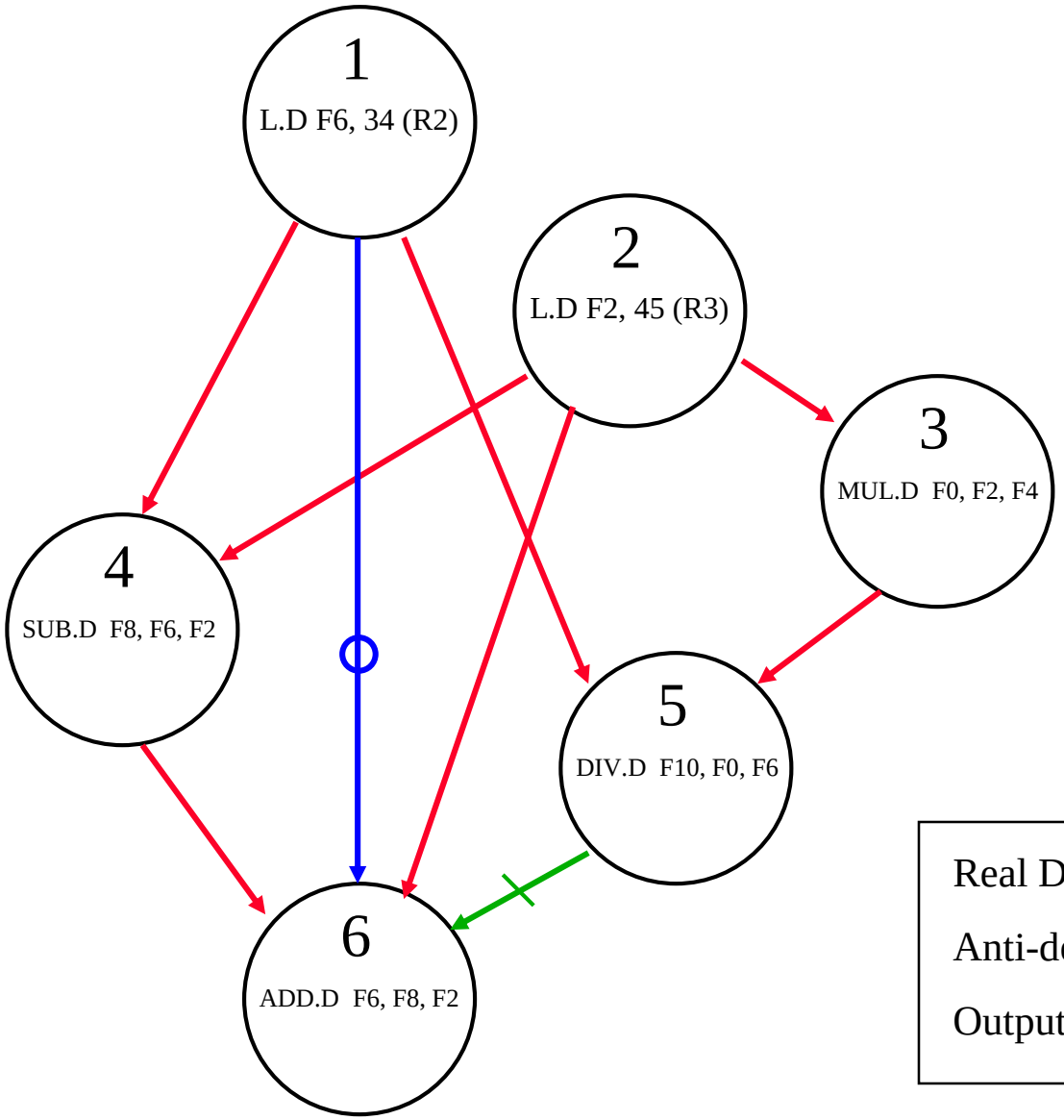


Pipelined Functional Units

True Data Dependence (RAW)
Anti-dependence (WAR)
Output Dependence (WAW)

L.D processing takes two cycles: EX, MEM

Dependency Graph For Example Code



Example Code

1	L.D	F6, 34(R2)
2	L.D	F2, 45(R3)
3	MUL.D	F0, F2, F4
4	SUB.D	F8, F6, F2
5	DIV.D	F10, F0, F6
6	ADD.D	F6, F8, F2

Data Dependence:
(1, 4) (1, 5) (2, 3) (2, 4)
(2, 6) (3, 5) (4, 6)

Output Dependence:
(1, 6)

Anti-dependence:
(5, 6)

Real Data Dependence (RAW)

Anti-dependence (WAR)

Output Dependence (WAW)

Tomasulo Example: Cycle 0

(i.e at end of cycle 0)

FP EX Cycles : Add = 2 cycles, Multiply = 10, Divide = 40

Instruction status					Execution	Write						
Instruction		j	k	Issue	Start/complete	Result			Busy	Address		
L.D	F6	34+	R2					Load1	No			
L.D	F2	45+	R3					Load2	No			
MUL.D	F0	F2	F4					Load3	No			
SUB.D	F8	F6	F2									
DIV.D	F10	F0	F6									
ADD.D	F6	F8	F2									

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

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

Tomasulo Example Cycle 1

FP EX Cycles : Add = 2 cycles, Multiply = 10, Divide = 40

Instruction status

Instruction		j	k	Issue	Execution Start/complete	Write Result
L.D	F6	34+	R2	1		
L.D	F2	45+	R3			
MUL.D	F0	F2	F4			
SUB.D	F8	F6	F2			
DIV.D	F10	F0	F6			
ADD.D	F6	F8	F2			

	Busy	Address
Load1	Yes	34+R2
Load2	No	
Load3	No	

Reservation Stations

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

Register result status

Clock 1

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

Tomasulo Example: Cycle 2

Instruction status

Instruction		j	k	Issue	Execution Start/complete	Write Result
L.D	F6	34+	R2	1	2/	
L.D	F2	45+	R3	2		
MUL.D	F0	F2	F4			
SUB.D	F8	F6	F2			
DIV.D	F10	F0	F6			
ADD.D	F6	F8	F2			

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

Reservation Stations

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

Register result status

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

Tomasulo Example: Cycle 3

Instruction status

Instruction		j	k	Issue	Execution Start/complete	Write Result
L.D	F6	34+	R2	1	2/3	
L.D	F2	45+	R3	2	3/	
MUL.D	F0	F2	F4	3		
SUB.D	F8	F6	F2			
DIV.D	F10	F0	F6			
ADD.D	F6	F8	F2			

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

Load processing takes 2 cycles (EX, Mem)

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULT.D		R(F4)	Load2	
	Mult2	No					

Register result status

Clock 3

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

Tomasulo Example: Cycle 4

Instruction status					Execution	Write						
Instruction		j	k	Issue	Start/complete	Result			Busy	Address		
L.D	F6	34+	R2	1	2/3	4		Load1	No			
L.D	F2	45+	R3	2	3/4			Load2	Yes	45+R3		
MUL.D	F0	F2	F4	3				Load3	No			
SUB.D	F8	F6	F2	4								
DIV.D	F10	F0	F6									
ADD.D	F6	F8	F2									

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

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

- Load2 completing; what is waiting for it?

Tomasulo Example: Cycle 5

FP EX Cycles : Add = 2 cycles, Multiply = 10, Divide = 40

Instruction status												
Instruction		*j*	*k*	*Issue*	*Execution Start/complete*	*Write Result*			Busy	Address		
L.D	F6	34+	R2	1	2/3	4		Load1	No			
L.D	F2	45+	R3	2	3/4	5		Load2	No			
MUL.D	F0	F2	F4	3				Load3	No			
SUB.D	F8	F6	F2	4								
DIV.D	F10	F0	F6	5								
ADD.D	F6	F8	F2									

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)						
		Add2	No									
		Add3	No									
	10	Mult1	Yes	MULTD	M(45+R3)	R(F4)						
		Mult2	Yes	DIVD		M(34+R2)	Mult1					

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

Load2 result forwarded via CDB to Add1, Mult1

SUB.D, MUL.D execution will start execution in next cycle

Tomasulo Example Cycle 6

FP EX Cycles : Add = 2 cycles, Multiply = 10, Divide = 40

Instruction status					Execution	Write						
Instruction		j	k	Issue	Start/complete	Result			Busy	Address		
L.D	F6	34+	R2	1	2/3	4		Load1	No			
L.D	F2	45+	R3	2	3/4	5		Load2	No			
MUL.D	F0	F2	F4	3	6/			Load3	No			
SUB.D	F8	F6	F2	4	6/							
DIV.D	F10	F0	F6	5								
ADD.D	F6	F8	F2	6								

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)						
		Add2	Yes	ADDD		M(45+R3)	Add1					
		Add3	No									
	9	Mult1	Yes	MULTD	M(45+R3)	R(F4)						
		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			

Add1 and Mult1 start to execute

Tomasulo Example: Cycle 7

FP EX Cycles : Add = 2 cycles, Multiply = 10, Divide = 40

Instruction status					Execution	Write						
Instruction		j	k	Issue	Start/complete	Result			Busy	Address		
L.D	F6	34+	R2	1	2/3	4		Load1	No			
L.D	F2	45+	R3	2	3/4	5		Load2	No			
MUL.D	F0	F2	F4	3	6/			Load3	No			
SUB.D	F8	F6	F2	4	6/7							
DIV.D	F10	F0	F6	5								
ADD.D	F6	F8	F2	6								

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)						
		Add2	Yes	ADDD		M(45+R3)	Add1					
		Add3	No									
	8	Mult1	Yes	MULTD	M(45+R3)	R(F4)						
		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			

Add1 completing; what is waiting for it?

Tomasulo Example: Cycle 10

Instruction status												
Instruction		*j*	*k*	*Issue*	*Execution Start/complete*	*Write Result*			Busy	Address		
L.D	F6	34+	R2	1	2/3	4		Load1	No			
L.D	F2	45+	R3	2	3/4	5		Load2	No			
MUL.D	F0	F2	F4	3	6/			Load3	No			
SUB.D	F8	F6	F2	4	6/7	8						
DIV.D	F10	F0	F6	5								
ADD.D	F6	F8	F2	6	9/10							

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

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

- Add2 completed execution

Tomasulo Example: Cycle 11

Instruction status					Execution	Write					
Instruction		j	k	Issue	Start/complete	Result			Busy	Address	
L.D	F6	34+	R2	1	2/3	4		Load1	No		
L.D	F2	45+	R3	2	3/4	5		Load2	No		
MUL.D	F0	F2	F4	3	6/			Load3	No		
SUB.D	F8	F6	F2	4	6/7	8					
DIV.D	F10	F0	F6	5							
ADD.D	F6	F8	F2	6	9/10	11					

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

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

- Write back result of ADD.D in this cycle

Tomasulo Example: Cycle 15

Instruction status											
Instruction		*j*	*k*	*Issue*	*Execution Start/complete*	*Write Result*			Busy	Address	
L.D	F6	34+	R2	1	2/3	4		Load1	No		
L.D	F2	45+	R3	2	3/4	5		Load2	No		
MUL.D	F0	F2	F4	3	6/15			Load3	No		
SUB.D	F8	F6	F2	4	6/7	8					
DIV.D	F10	F0	F6	5							
ADD.D	F6	F8	F2	6	9/10	11					

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

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

- Mult1 completed execution; what is waiting for it?

Tomasulo Example: Cycle 16

FP EX Cycles : Add = 2 cycles, Multiply = 10, Divide = 40

Instruction status					Execution	Write					
Instruction		j	k	Issue	Start/complete	Result			Busy	Address	
L.D	F6	34+	R2	1	2/3	4		Load1	No		
L.D	F2	45+	R3	2	3/4	5		Load2	No		
MUL.D	F0	F2	F4	3	6/15	16		Load3	No		
SUB.D	F8	F6	F2	4	6/7	8					
DIV.D	F10	F0	F6	5							
ADD.D	F6	F8	F2	6	9/10	11					

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

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

Only Divide instruction remains

DIV.D execution will start in next cycle (17)

Tomasulo Example: Cycle 57

Instruction status

Instruction		j	k	Issue	Execution Start/complete	Write Result
L.D	F6	34+	R2	1	2/3	4
L.D	F2	45+	R3	2	3/4	5
MUL.D	F0	F2	F4	3	6/15	16
SUB.D	F8	F6	F2	4	6/7	8
DIV.D	F10	F0	F6	5	17/56	57
ADD.D	F6	F8	F2	6	9/10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations

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

Register result status

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

Instruction Block done

- Again we have:
  - In-oder issue,
  - Out-of-order execution, completion