Advanced Computer Architecture

Dynamic Scheduling using Tomasulo Algorithm

Prof. Roger Luis Uy
De La Salle University
College of Computer Studies

Tomasulo Algorithm

- Invented by Robert Tomasulo and used by the IBM 360/91 floating-point unit
- Key Idea: RAW hazard are avoided by executing an instruction only when its operands are available
- Key Idea: WAR & WAW hazard are avoided by using Register renaming

Tomasulo Algorithm

Consider the following example code sequence:

```
DIV.D F0, F2, F4
ADD.D F6, F0, F8
S.D F6, 0(R1)
SUB.D F8, F10, F14
MUL.D F6, F10, F8
```

Tomasulo Algorithm

With renaming:

```
DIV.D F0, F2, F4
ADD.D S, F0, F8
S.D 0(R1),S
SUB.D T, F10, F14
MUL.D F6, F10, T
```

Tomasulo Algorithm

- Motivation (i.e., problem of IBM 360)
  - 4 double-precision FP registers
  - Long memory accesses
  - Long floating-point delays

Tomasulo Algorithm

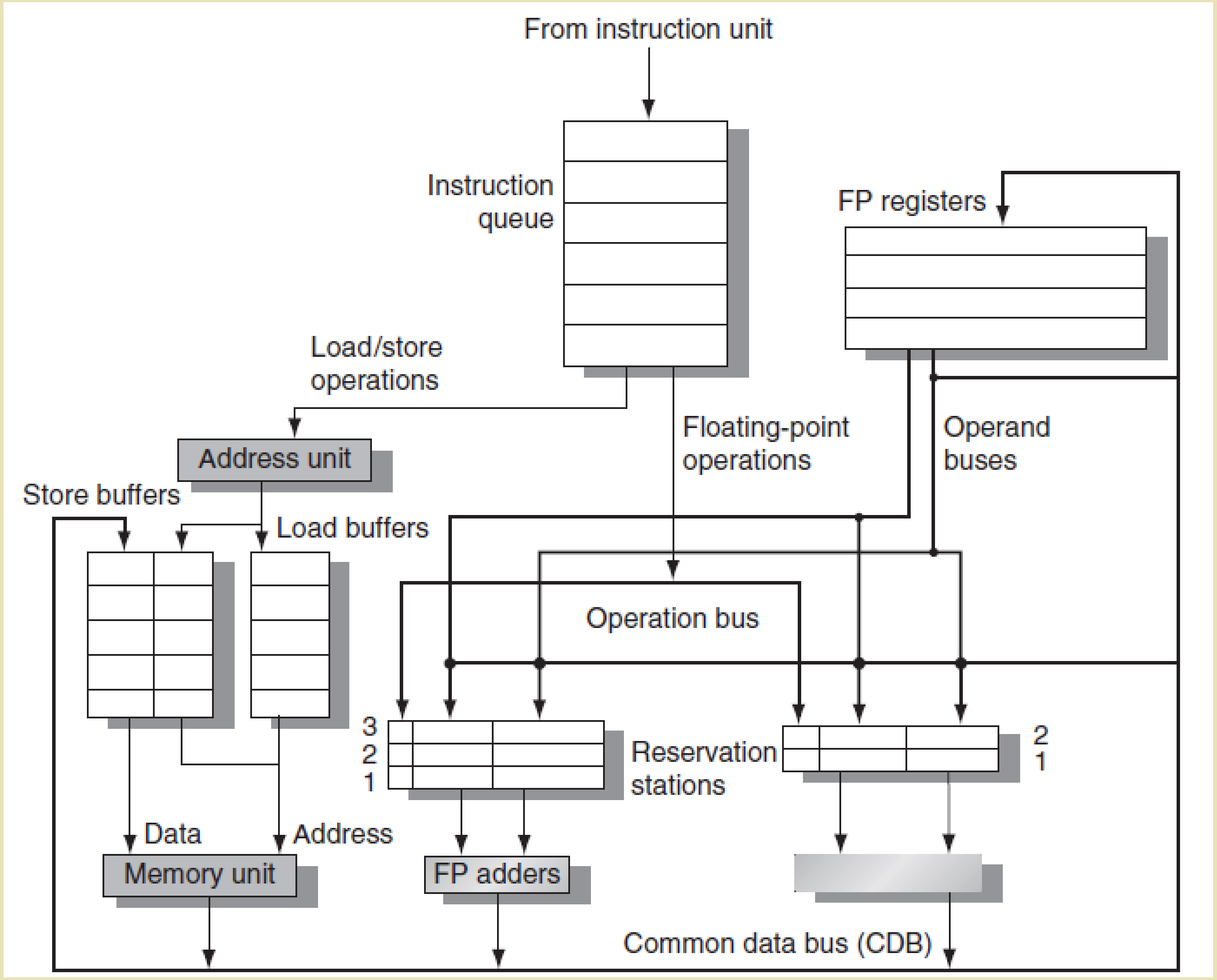
- IBM 360 uses pipelined functional units vs. multiple functional units
- The following floating-point operations could be accommodated by IBM 360:
  - 3 FP adder
  - 2 FP multiplier
  - Up to 6 FP loads & 3 FP stores could be outstanding via load data buffers & store data buffers

Tomasulo Algorithm

- Register renaming is done via *reservation station*
- Reservation station is a buffer for the *operands* of instructions waiting to issue
- Reservation station is used as follows:
  - Fetches / buffers an operand as soon as it is available
  - Pending instructions designate the reservation station that will provide their input
  - If successive writes to a register occur, only the last one is actually used to update the register

Tomasulo Algorithm

- 2 Significant differences between Tomasulo & Scoreboarding:
  - Hazard detection & execution control are decentralized for Tomasulo (via reservation stations) while scoreboarding is centralized
  - For Tomasulo, results are passed directly to functional units from the “buffered” reservation stations while scoreboarding is through the registers


From instruction unit
Instruction queue
FP registers
Load/store operations
Floating-point operations
Operand buses
Address unit
Store buffers
Load buffers
Operation bus
3
2
1
Reservation stations
2
1
Data
Address
Memory unit
FP adders
Common data bus (CDB)

Tomasulo Algorithm

- Basic structure of a Tomasulo-based FP unit for MIPS:
  - Reservation stations:
    - Instructions that have been issued & are awaiting execution at a functional unit
    - Operands for the instruction if they have already been computed or the source operands
    - Information needed to control the instruction once it has begun execution at the unit (i.e., used to detect & resolve hazards)

The Tomasulo Approach

- Basic structure of a Tomasulo-based FP unit for MIPS:
  - FP operation queue: instructions are fetched and placed in this queue
  - Load buffers: hold data coming from memory
  - Store buffers: hold addresses going to memory
  - FP registers & FU : a pair of busses
  - FP registers & store buffers: single bus
  - FU & memory: common data bus (CDB)
  - CDB: connected to everywhere except load buffers
  - Tag field: all buffers & reservation stations

The Tomasulo Approach

- Three cycles/steps for Tomasulo:
  - Issue
  - Execute
  - Write result

The Tomasulo Approach

- Issue cycle:
  - Get instruction from the FP operation queue
  - If FP operation, issue it if there is an empty reservation station
  - Operands are send from the registers to the reservation station
  - If load/store: issue if there is an available buffer
  - If no empty reservation station or empty buffer, then there is a structural hazard
  - Register renaming

The Tomasulo Approach

- Execute cycle:
  - If one more more of the operands is not available, monitor the CDB while waiting for the register to be computed
  - If an operand is available, placed into the corresponding reservation station
  - If all operands are available, execute the operation
  - Checks for RAW hazards

The Tomasulo Approach

- Write result cycle:
  - If result is available, write it on the CDB then from there to registers & reservation stations

The Tomasulo Approach

- Three major differences between scoreboarding & Tomasulo with regards to the cycles:
  - No checking of WAW & WAR hazards (register renaming)
  - CDB is used to broadcast results rather than waiting on the registers
  - Load & stores are treated as basic functional units

The Tomasulo Approach

- Concept of Tag field:
  - Tags are essentially names for an extended set of virtual registers used in renaming
  - Describes which reservation station contains the instruction that will produce a result needed as a source operand
  - Tag value of 0: indicate that the operand is already available in the registers

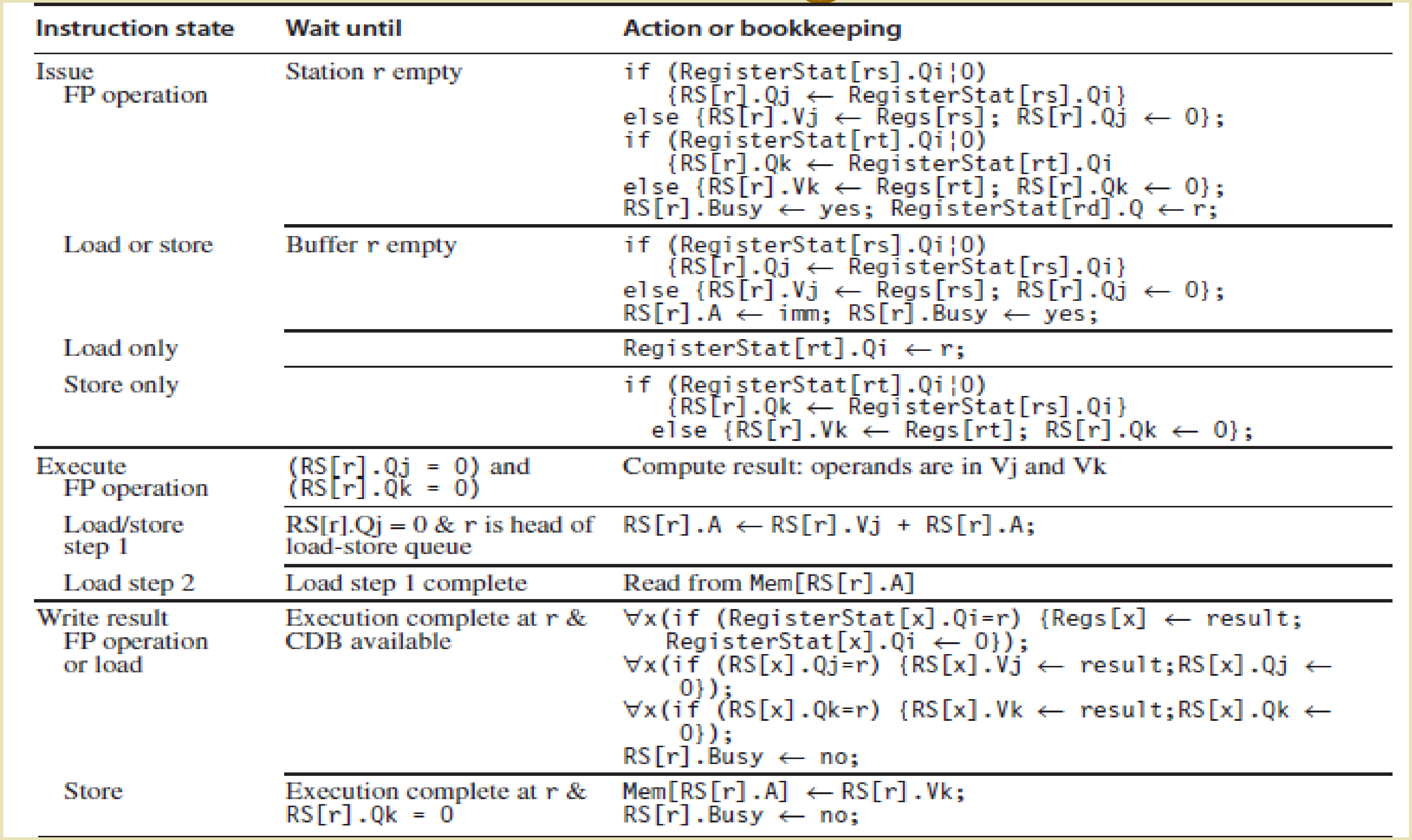
The Tomasulo Approach

- More reservation stations than actual registers numbers implies that WAW & WAR hazards can be reduced by renaming results using reservation station numbers
- Reservation station vs. Reorder buffer

Tomasulo Approach & MIPS64

- Three table for the Tomasulo:
  - Instruction Status
    - Op – The operation to perform on source operands S1 & S2
    - Qj, Qk – The reservation stations that will produce the corresponding source operand. A value of 0 indicates that the source operand is already available
    - Vj, Vk – The value of the source operands
  - Reservation stations
  - Register status
    - Qi – The number of the reservation station that contains the operation whose result should be stored into this register or into memory

Tomasulo Algorithm



Instruction state	Wait until	Action or bookkeeping

Issue
FP operation
	Station r empty	if (RegisterStat[rs].Qi¦0)
		{RS[r].Qj ← RegisterStat[rs].Qi}
		else {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0};
		if (RegisterStat[rt].Qi¦0)
		{RS[r].Qk ← RegisterStat[rt].Qi
		else {RS[r].Vk ← Regs[rt]; RS[r].Qk ← 0};
		RS[r].Busy ← yes; RegisterStat[rd].Q ← r;

Load or store	Buffer r empty	if (RegisterStat[rs].Qi¦0)
		{RS[r].Qj ← RegisterStat[rs].Qi}
		else {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0};
		RS[r].A ← imm; RS[r].Busy ← yes;

Load only		RegisterStat[rt].Qi ← r;

Store only		if (RegisterStat[rt].Qi¦0)
		{RS[r].Qk ← RegisterStat[rs].Qi}
		else {RS[r].Vk ← Regs[rt]; RS[r].Qk ← 0};

Execute FP operation	(RS[r].Qj = 0) and (RS[r].Qk = 0)	Compute result: operands are in Vj and Vk
Load/store step 1	RS[r].Qj = 0 & r is head of load-store queue	RS[r].A ← RS[r].Vj + RS[r].A;
Load step 2	Load step 1 complete	Read from Mem[RS[r].A]

Write result
FP operation
or load
	Execution complete at r &
	CDB available
		∀x(if (RegisterStat[x].Qi=r) {Regs[x] ← result;
		RegisterStat[x].Qi ← 0});
		∀x(if (RS[x].Qj=r) {RS[x].Vj ← result;RS[x].Qj ←
		0});
		∀x(if (RS[x].Qk=r) {RS[x].Vk ← result;RS[x].Qk ←
		0});
		RS[r].Busy ← no;

Store	Execution complete at r &
	RS[r].Qk = 0
		Mem[RS[r].A] ← RS[r].Vk;
		RS[r].Busy ← no;

Tomasulo Approach Example

- Show the status table using Tomasulo approach based on the code fragment below. Assume that FP add will take 2 clock cycles, multiply will take 10 clock cycles and divide will take 40 clock cycles

```
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
```

Tomasulo Example

Instruction status:

Instruction		j	k	Issue	*Exec* *Comp*	*Write* *Result*
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			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

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

Register result status:

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

Tomasulo Example Cycle 1

Instruction status:

Instruction		j	k	Issue	Exec Comp	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 Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

Register result status:

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

Tomasulo Example Cycle 2

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
L.D	F6	34+	R2	1		
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 Qj	RS 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	Exec Comp	Write Result
L.D	F6	34+	R2	1	3	
L.D	F2	45+	R3	2		
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	

Reservation Stations:

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

Register result status:

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

- **Note: registers names are removed ("renamed") in Reservation Stations; MUL.D issued vs. scoreboard**

Tomasulo Example Cycle 4

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
L.D	F6	34+	R2	1	3	4
L.D	F2	45+	R3	2	4	
MUL.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	4		
DIV.D	F10	F0	F6			
ADD.D	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	Yes	45+R3
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	Yes	SUBD	M(A1)			Load2
	Add2	No					
	Add3	No					
	Mult1	Yes	MUL.D		R(F4)	Load2	
	Mult2	No					

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
4	*FU*	Mult1	Load2		M(A1)	Add1				

Tomasulo Example Cycle 5

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
L.D	F6	34+	R2	1	3	4
L.D	F2	45+	R3	2	4	5
MUL.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	4		
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
2	Add1	Yes	SUB.D	M(A1)	M(A2)		
	Add2	No					
	Add3	No					
10	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
5	*FU*	Mult1	M(A2)		M(A1)	Add1	Mult2			

Tomasulo Example Cycle 6

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
L.D	F6	34+	R2	1	3	4
L.D	F2	45+	R3	2	4	5
MUL.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	4		
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
1	Add1	Yes	SUB.D	M(A1)	M(A2)		
	Add2	Yes	ADD.D		M(A2)	Add1	
	Add3	No					
9	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

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

- **Issue ADD.D here vs. scoreboard?**

Tomasulo Example Cycle 7

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
L.D	F6	34+	R2	1	3	4
L.D	F2	45+	R3	2	4	5
MUL.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	4	7	
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
0	Add1	Yes	SUB.D	M(A1)	M(A2)		
	Add2	Yes	ADD.D		M(A2)	Add1	
	Add3	No					
8	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
7	*FU*	Mult1	M(A2)		Add2	Add1	Mult2			

Tomasulo Example Cycle 8

Instruction status:

Instruction		j	k	*Exec* *Issue*	*Comp*	*Write* *Result*
L.D	F6	34+	R2	1	3	4
L.D	F2	45+	R3	2	4	5
MUL.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	4	7	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

				S1	*S2*	*RS*	*RS*
Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*
	Add1	No					
2	Add2	Yes	ADD.D	(M-M)	M(A2)		
	Add3	No					
7	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
8	*FU*	Mult1	M(A2)		Add2	(M-M)	Mult2			

Tomasulo Example Cycle 9

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
L.D	F6	34+	R2	1	3	4
L.D	F2	45+	R3	2	4	5
MUL.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	4	7	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
1	Add2	Yes	ADD.D	(M-M)	M(A2)		
	Add3	No					
6	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
9	*FU*	Mult1	M(A2)		Add2	(M-M)	Mult2			

Tomasulo Example Cycle 10

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
L.D	F6	34+	R2	1	3	4
L.D	F2	45+	R3	2	4	5
MUL.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	4	7	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	10	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
0	Add2	Yes	ADD.D	(M-M)	M(A2)		
	Add3	No					
5	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

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

Tomasulo Example Cycle 11

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
L.D	F6	34+	R2	1	3	4
L.D	F2	45+	R3	2	4	5
MUL.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	4	7	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
11	*FU*	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

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

Tomasulo Example Cycle 12

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
L.D	F6	34+	R2	1	3	4
L.D	F2	45+	R3	2	4	5
MUL.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	4	7	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
3	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
12	*FU*	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

Tomasulo Example Cycle 13

Instruction status:

Instruction		j	k	*Issue*	*Exec Comp*	*Write Result*
L.D	F6	34+	R2	1	3	4
L.D	F2	45+	R3	2	4	5
MUL.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	4	7	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
2	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
13	*FU*	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

Tomasulo Example Cycle 14

Instruction status:

Instruction		j	k	*Issue*	*Exec Comp*	*Write Result*
L.D	F6	34+	R2	1	3	4
L.D	F2	45+	R3	2	4	5
MUL.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	4	7	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
1	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
14	*FU*	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

Tomasulo Example Cycle 15

Instruction status:

Instruction		j	k	*Issue*	*Exec Comp*	*Write Result*
L.D	F6	34+	R2	1	3	4
L.D	F2	45+	R3	2	4	5
MUL.D	F0	F2	F4	3	15	
SUB.D	F8	F6	F2	4	7	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
0	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
15	*FU*	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

Tomasulo Example Cycle 16

Instruction status:

Instruction		j	k	*Issue*	*Exec Comp*	*Write Result*
L.D	F6	34+	R2	1	3	4
L.D	F2	45+	R3	2	4	5
MUL.D	F0	F2	F4	3	15	16
SUB.D	F8	F6	F2	4	7	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
40	Mult2	Yes	DIV.D	M*F4	M(A1)		

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
16	*FU*	M*F4	M(A2)		(M-M+M	(M-M)	Mult2			

(skip a couple of cycles)

Tomasulo Example Cycle 55

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
L.D	F6	34+	R2	1	3	4
L.D	F2	45+	R3	2	4	5
MUL.D	F0	F2	F4	3	15	16
SUB.D	F8	F6	F2	4	7	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
1	Mult2	Yes	DIV.D	M*F4	M(A1)		

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
55	*FU*	M*F4	M(A2)		(M-M+M	(M-M)	Mult2			

Tomasulo Example Cycle 56

Instruction status:

Instruction		j	k	*Issue*	*Exec Comp*	*Write Result*
L.D	F6	34+	R2	1	3	4
L.D	F2	45+	R3	2	4	5
MUL.D	F0	F2	F4	3	15	16
SUB.D	F8	F6	F2	4	7	8
DIV.D	F10	F0	F6	5	56	
ADD.D	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
0	Mult2	Yes	DIV.D	M*F4	M(A1)		

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
56	*FU*	M*F4	M(A2)		(M-M+M	(M-M)	Mult2			

Tomasulo Example Cycle 57

Instruction status:

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

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	Yes	DIV.D	M*F4	M(A1)		

Register result status:

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

- **Once again: In-order issue, out-of-order execution and completion.**

Compare to Scoreboard Cycle 62

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write Result		Issue	Exec Comp	Write Result
L.D	F6	34+	R2	1	2	3	4		1	3	4
L.D	F2	45+	R3	5	6	7	8		2	4	5
MUL.D	F0	F2	F4	6	9	19	20		3	15	16
SUB.D	F8	F6	F2	7	9	11	12		4	7	8
DIV.D	F10	F0	F6	8	21	61	62		5	56	57
ADD.D	F6	F8	F2	13	14	16	22		6	10	11

- **Why take longer on scoreboard?**
  - **Structural Hazards**
  - **Lack of forwarding**