Advanced Computer Architecture

Tomasulo's Dynamic Instruction Scheduling Algorithm

Lecture's Overview

❑ ***Previous Lecture:***

➔ Overcoming data hazard through dynamic scheduling

- Memory dependencies hardest to determine

➔ HW exploiting ILP

- Works when cannot know dependence at run time
- Code for one machine runs well on another

➔ The scoreboard scheduling algorithm

- Key idea of Scoreboard: Allow instructions behind stall to proceed
- Enables out-of-order execution => out-of-order completion
- ID stage checked both for structural and data hazards

❑ ***This Lecture***

➔ Overcoming data hazard through dynamic scheduling

➔ The Tomasulo scheduling algorithm

HW Schemes: Instruction Parallelism

- Why in HW at run time?
  - ➔ Works when can’t know real dependence at compile time
  - ➔ Compiler simpler
  - ➔ Code for one machine runs well on another
- Key idea: Allow instructions behind stall to proceed
  - ➔ Enables out-of-order execution => out-of-order completion
  - ➔ ID stage checked both for structural and data hazards
- 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

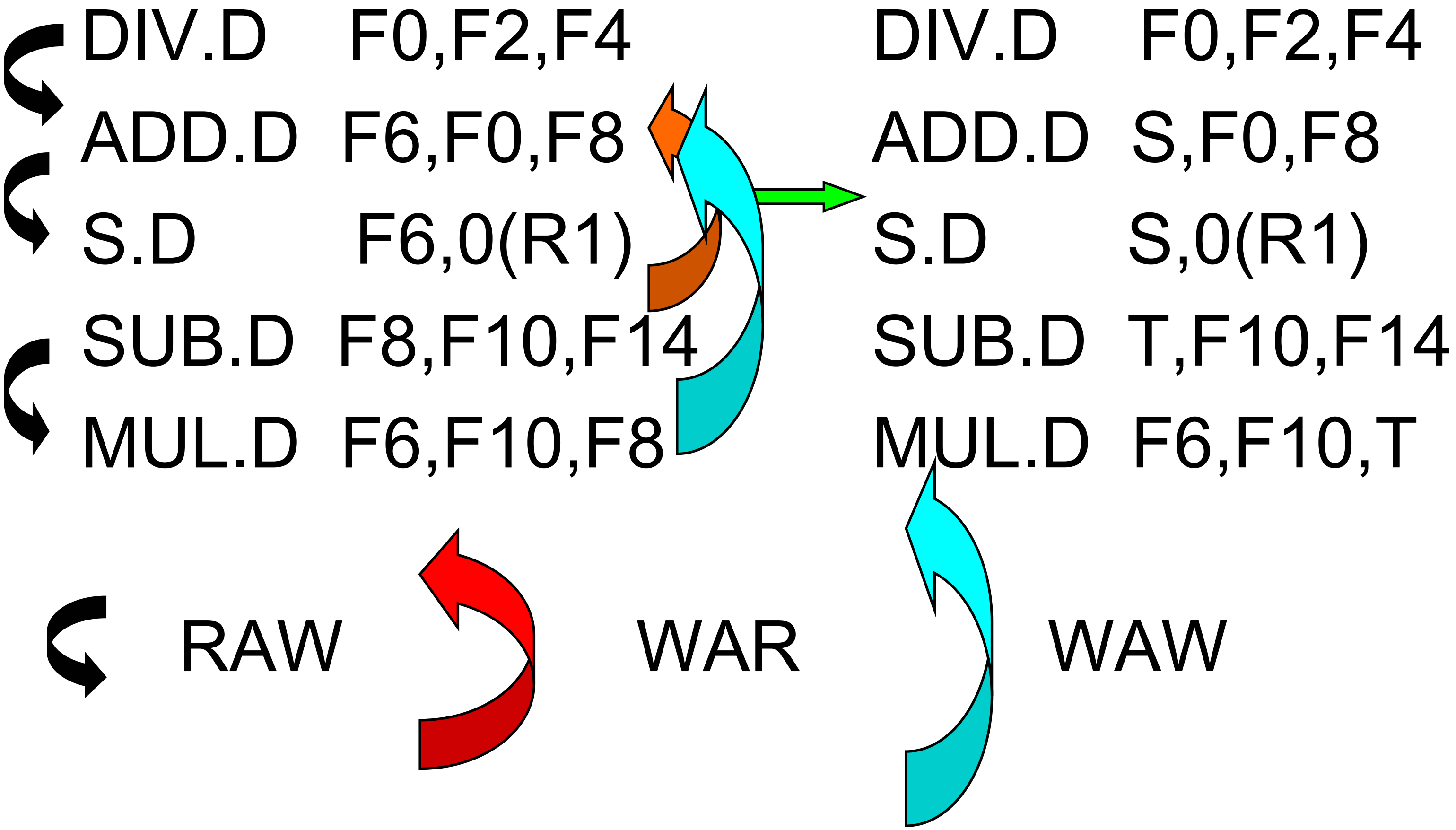
Scoreboard Summary

- Speedup 1.7 from compiler; 2.5 by hand BUT slow memory (no cache)
- Limitations of 6600 scoreboard
  - ➔ No forwarding (First write register then read it)
  - ➔ Limited to instructions in basic block (small *window*)
  - ➔ Number of functional units(structural hazards)
  - ➔ Wait for WAR and WAW hazards

Tomasulo Algorithm vs. Scoreboard

- ❑ Tomasulo algorithm lead to Alpha 21264, HP 8000, MIPS 10000, Pentium II, PowerPC 604, …
- ❑ Many variations compared with scoreboard, though the key concept of register renaming to avoid WAR and WAW hazards is the common one
- ❑ Control & buffers distributed with Function Units (FU) vs. centralized in scoreboard;
  - ➔ FU buffers called "reservation stations"; have pending operands
- ❑ Registers in instructions replaced by values or pointers to reservation stations(RS); called register renaming ;
  - ➔ avoids WAR, WAW hazards
  - ➔ More reservation stations than registers, so can do optimizations compilers cannot perform
- ❑ Results to FU from reservation stations , not through registers, over Common Data Bus that broadcasts results to all function units
- ❑ Load and Stores treated as function units with reservation stations as well
- ❑ Issuing instructions can go past branches, allowing FP operations beyond basic block in FP queue

Renaming eliminates WAR and WAW hazards

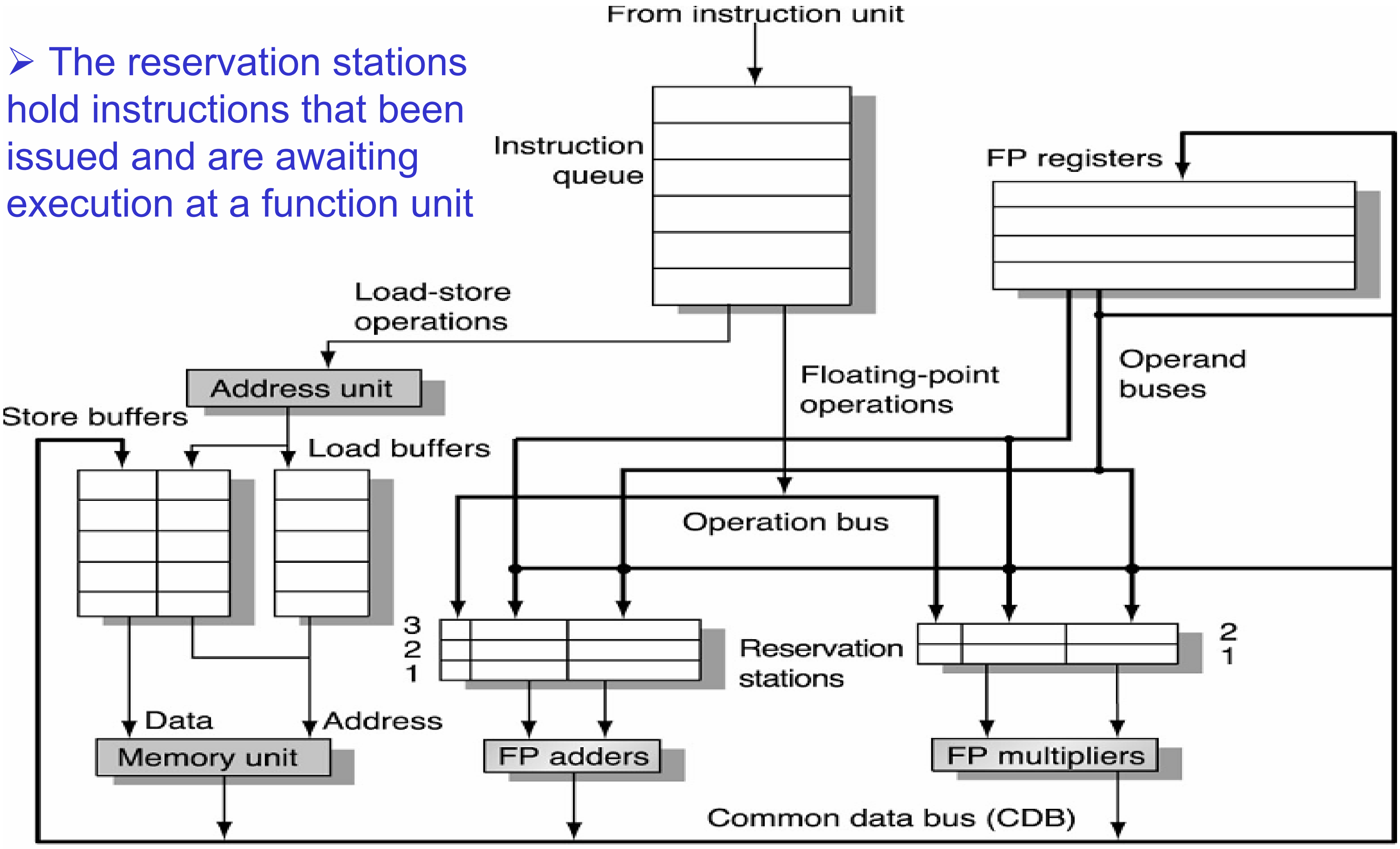


```
DIV.D   F0,F2,F4
ADD.D   F6,F0,F8
S.D     F6,0(R1)
SUB.D   F8,F10,F14
MUL.D   F6,F10,F8
```

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

Tomasulo Organization

- The reservation stations hold instructions that been issued and are awaiting execution at a function unit



- All results from FP func. units and loads are broadcasted on the CBD

Three Stages of Tomasulo Algorithm

1. Issue—get instruction from FP Operation Queue

   If reservation station free (no structural hazard), control issues instruction & sends operands (renames registers).

2. Execution—operate on operands (EX)

   When both operands ready then execute;
   if not ready, watch Common Data Bus for result

3. Write result—finish execution (WB)

   Write on Common Data Bus to all awaiting units;
   mark reservation station available

- ❑ Normal data bus: data + destination ("go to" bus)
- ❑ Common data bus: data + source ("come from" bus)
  - ➔ 64 bits of data + 4 bits of Functional Unit source address
  - ➔ Write if matches expected Functional Unit (produces result)
  - ➔ Does the broadcast

Reservation Station Components

Op—Operation to perform in the unit (e.g., + or –)

Vj, Vk—Value of Source operands

- ➔ Store buffers has V field, result to be stored

Qj, Qk—Reservation stations producing source registers (value to be written)

- ➔ Note: No ready flags as in Scoreboard; Qj,Qk=0 => ready
- ➔ Store buffers only have Qi for RS producing result

Busy—Indicates reservation station or function unit is busy

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

Instruction state	Wait until	Action or bookkeeping

Issue
FP Operation
	Station r empty	`if (Register Stat[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]Q.i}`
		`else {RS[r].Vk← Regs[rt]; RS[r].Qk← 0};`
		`RS[r].Busy← yes; RegisterStat[rd].Qi=r;`

Load or Store	Buffer r empty	`if (Register Stat[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 (Register Stat[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	RS[r].A<>0	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 Example Cycle 0

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

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										

Latency: *load* 1, *add* 2, *multiply* 10 and *divide* 40 clock cycles

Tomasulo Example Cycle 1

Instruction status					Execution	Write
Instruction	j	k		Issue	complete	Result
LD	F6	34+	R2	1		
LD	F2	45+	R3			
MULT	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	

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

Register result status

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

Tomasulo Example Cycle 2

Instruction status					Execution	Write
Instruction		j	k	Issue	complete	Result
LD	F6	34+	R2	1		
LD	F2	45+	R3	2		
MUL	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	

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

Register result status

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

Note: Unlike 6600, can have multiple loads outstanding

Tomasulo Example Cycle 3

Instruction status

Instruction		j	k	Issue	Execution complete	Write Result
LD	F6	34+	R2	1	3	
LD	F2	45+	R3	2		
MUL	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	

Reservation Stations

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

- Note: register names are removed (“renamed”) in Reservation Stations; MULT issued vs. scoreboard
- Load1 completing; what is waiting for 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	4			Load2	Yes	45+R3		
MULTD	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4								
DIVD	F10	F0	F6									
ADDD	F6	F8	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									
		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				

- Load2 completing; what is waiting for it?

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	4	5		Load2	No	
MUL	F0	F2	F4	3				Load3	No	
SUBD	F8	F6	F2	4						
DIVD	F10	F0	F6	5						
ADDD	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)		
	0	Add2	No					
		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
5			FU	Mult1	M(45+R3)		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	4	5		Load2	No			
MUL	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4								
DIVD	F10	F0	F6	5								
ADDD	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)						
	0	Add2	Yes	ADDD		M(45+R3)	Add1					
		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*
6			*FU*	Mult1	M(45+R3)		Add2	Add1	Mult2			

- Issue ADDD here vs. scoreboard?

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	4	5		Load2	No			
MUL	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4	7							
DIVD	F10	F0	F6	5								
ADDD	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)						
	0	Add2	Yes	ADDD		M(45+R3)	Add1					
		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*
7			*FU*	Mult1	M(45+R3)		Add2	Add1	Mult2			

- Add1 completing; what is waiting for it?

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	4	5		Load2	No			
MUL	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6								
Reservation Stations					*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	MULT	M(45+R3)	R(F4)						
	0	Mult2	Yes	DIVD		M(34+R2)	Mult1					
Register result s												
Clock				*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
8			*FU*	Mult1	M(45+R3)		Add2	M()-M()	Mult2			

Tomasulo Example Cycle 9

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	4	5		Load2	No			
MUL	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6								
Reservation Stations					*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					
Register result status												
Clock				*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
9			*FU*	Mult1	M(45+R3)		Add2	M()–M()	Mult2			

Tomasulo Example Cycle 10

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	4	5		Load2	No	
MUL	F0	F2	F4	3				Load3	No	
SUBD	F8	F6	F2	4	7	8				
DIVD	F10	F0	F6	5						
ADDD	F6	F8	F2	6	10					

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	

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

- Add2 completing; what is waiting for it?

Tomasulo Example Cycle 11

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	4	5		Load2	No			
MUL	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10	11						
Reservation Stations					*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	MULT	M(45+R3)	R(F4)						
	0	Mult2	Yes	DIVD		M(34+R2)	Mult1					
Register result s												
Clock				*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
11			*FU*	Mult1	M(45+R3)		(M-M)+M()	M()ĞM()	Mult2			

- Write result of ADDD here vs. scoreboard?

Tomasulo Example Cycle 12

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	4	5		Load2	No			
MUL	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4	6	7						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10	11						
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					
Register result status												
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 Example Cycle 13

Instruction status					Execution	Write
Instruction		j	k	Issue	complete	Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MUL	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	

Reservation Stations				S1	S2	RS for j	RS for k
Time	Name	Busy	Op	Vj	Vk	Qj	Qk
0	Add1	No					
0	Add2	No					
	Add3	No					
2	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
13		FU	Mult1	M(45+R3)		(M–M)+M()	M()–M()	Mult2			

Tomasulo Example Cycle 14

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	4	5		Load2	No			
MUL	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10	11						
Reservation Stations					S1	S2	RS for j	RS for k				
	Time	Name	Bus	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					
Register result status												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
14			FU	Mult1	M(45+R3)		(M–M)+M()	M()–M()	Mult2			

Tomasulo Example Cycle 15

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	4	5		Load2	No			
MUL	F0	F2	F4	3	15			Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10	11						
Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*				
	0	Add1	No									
	0	Add2	No									
		Add3	No									
	0	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*
15			*FU*	Mult1	M(45+R3)		(M–M)+M()	M()–M()	Mult2			

- Mult1 completing; what is waiting for it?

Tomasulo Example Cycle 16

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	4	5		Load2	No	
MUL	F0	F2	F4	3	15	16		Load3	No	
SUBD	F8	F6	F2	4	7	8				
DIVD	F10	F0	F6	5						
ADDD	F6	F8	F2	6	10	11				

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

- Note: Just waiting for divide

Tomasulo Example Cycle 55

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	4	5		Load2	No			
MULT	F0	F2	F4	3	15	16		Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10	11						
Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*				
	0	Add1	No									
	0	Add2	No									
		Add3	No									
	0	Mult1	No									
	1	Mult2	Yes	DIVD	M*F4	M(34+R2)						
Register result status												
Clock				*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
55			*FU*	M*F4	M(45+R3)		(M–M)+M()	M()–M()	Mult2			

Tomasulo Example Cycle 56

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	4	5		Load2	No			
MUL	F0	F2	F4	3	15	16		Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5	56							
ADDD	F6	F8	F2	6	10	11						
Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*				
	0	Add1	No									
	0	Add2	No									
		Add3	No									
	0	Mult1	No									
	0	Mult2	Yes	DIVD	M*F4	M(34+R2)						
Register result status												
Clock				*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
56			*FU*	M*F4	M(45+R3)		(M–M)+M()	M()–M()	Mult2			

- Mult 2 completing; what is waiting for it?

Tomasulo Example Cycle 57

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	4	5		Load2	No			
MUL	F0	F2	F4	3	15	16		Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5	56	57						
ADDD	F6	F8	F2	6	10	11						
Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*				
	0	Add1	No									
	0	Add2	No									
		Add3	No									
	0	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			

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

Compare to Scoreboard Cycle 62

Instruction status					Read	Executi	Write
Instruction		j	k	Issue	operan	comple	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				dest	S1	S2	FU for j	FU for k	Fj?	Fk?
Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
0	Divide	No								

Register result status

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

- Why takes longer on Scoreboard/6600?

Tomasulo vs. Scoreboard

	Tomasulo (IBM 360/91)	Scoreboard (CDC 6600)
Functional Units	Pipelined (6 load, 3 store, 3 +, 2 x/÷)	Multiple (1 load/store, 1 + , 2 x, 1 ÷)
Window size	14 instructions	5 instructions
Structural hazard	No issue	No issue
WAR	Renaming avoids	Stall completion
WAW	Renaming avoids	Stall completion
Operands	Broadcast results from FU	Write/read registers
Control	Reservation stations	Central scoreboard

Tomasulo Drawbacks

❑ Circuit complexity

❑ Many associative stores (CDB) at high speed

❑ Performance limited by Common Data Bus

➔ Multiple CDBs → more FU logic for parallel associative stores

Tomasulo Loop Example

```
Loop:  LD         F0    0     R1
       MULTD      F4    F0    F2
       SD         F4    0     R1
       SUBI       R1    R1    #8
       BNEZ       R1    Loop
```

- Assume Multiply takes 4 clocks
- Assume first load takes 8 clocks (cache miss?), second load takes 4 clocks (hit)
- To be clear, will show clocks for SUBI, BNEZ
- Reality, integer instructions ahead

Loop Example Cycle 0

Instruction status

					Executioi	Write
Instruction	j	k	iteration	Issue	complete	Result
LD	F0	0	R1	1		
MULT	F4	F0	F2	1		
SD	F4	0	R1	1		
LD	F0	0	R1	2		
MULT	F4	F0	F2	2		
SD	F4	0	R1	2		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	Qi
Store1	No	
Store2	No	
Store3	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					

Code:

```
LD   F0   0  R1
MULT F4   F0 F2
SD   F4   0  R1
SUBI R1   R1 #8
BNEZ R1   Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
0	80	Qi									

Loop Example Cycle 1

Instruction status

						Executio	Write
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*
LD	F0	0	R1	1	1		
MULT	F4	F0	F2	1			
SD	F4	0	R1	1			
LD	F0	0	R1	2			
MULT	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address
Load1	Yes	80
Load2	No	
Load3	No	

	Busy	Qi
Store1	No	
Store2	No	
Store3	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					

Code:

```
LD    F0    0   R1
MULT  F4    F0  F2
SD    F4    0   R1
SUBI  R1    R1  #8
BNEZ  R1    Loop
```

Register result status

Clock	**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
1	**80**	*Qi*	Load1								

Loop Example Cycle 2

Instruction status

Instruction		j	k	iteration	Issue	Executio	Write
						complete	Result
LD	F0	0	R1	1	1		
MULT	F4	F0	F2	1	2		
SD	F4	0	R1	1			
LD	F0	0	R1	2			
MULT	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address
Load1	Yes	80
Load2	No	
Load3	No	

	Busy	Qi
Store1	No	
Store2	No	
Store3	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		R(F2)	Load1	
0	Mult2	No					

Code:

```
LD    F0   0   R1
MULT  F4   F0  F2
SD    F4   0   R1
SUBI  R1   R1  #8
BNEZ  R1   Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
2	80	Qi	Load1		Mult1						

Loop Example Cycle 3

Instruction status

					Executio	Write
Instruction	j	k	iteration	Issue	complete	Result
LD	F0	0	R1	1	1		
MULT	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0	R1	2			
MULT	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address	Qi
Load1	Yes	80	
Load2	No		
Load3	No		
Store1	Yes	80	Mult1
Store2	No		
Store3	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		R(F2)	Load1	
0	Mult2	No					

Code:

LD	F0	0	R1
MULT	F4	F0	F2
SD	F4	0	R1
SUBI	R1	R1	#8
BNEZ	R1	Loop	

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
3	**80**	Qi	Load1		Mult1						

- Note: MULT1 has no registers names in RS

Loop Example Cycle 4

Instruction status

						Executio	*Write*
Instruction	*j*	*k*		*iteration*	*Issue*	*complete*	*Result*
LD	F0	0	R1	1	1		
MULT	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0	R1	2			
MULT	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address	
Load1	Yes	80	
Load2	No		
Load3	No		Qi
Store1	Yes	80	Mult1
Store2	No		
Store3	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		R(F2)	Load1	
0	Mult2	No					

Code:

LD	F0	0	R1
MULT	F4	F0	F2
SD	F4	0	R1
SUBI	R1	R1	#8
BNEZ	R1	Loop	

Register result status

Clock	**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
4	**72**	*Qi*	Load1		Mult1						

Loop Example Cycle 5

Instruction status

Instruction	j	k	iteration	Issue	*Executio complete*	*Write Result*
LD F0	0	R1	1	1		
MULT F4	F0	F2	1	2		
SD F4	0	R1	1	3		
LD F0	0	R1	2			
MULT F4	F0	F2	2			
SD F4	0	R1	2			

	Busy	Address	Qi
Load1	Yes	80	
Load2	No		
Load3	No		
Store1	Yes	80	Mult1
Store2	No		
Store3	No		

Reservation Stations

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS for j Qj*	*RS for k Qk*
0	Add1	No					
0	Add2	No					
0	Add3	No					
0	Mult1	Yes	MULTD		R(F2)	Load1	
0	Mult2	No					

Code:

LD	F0	0	R1
MULT	F4	F0	F2
SD	F4	0	R1
SUBI	R1	R1	#8
BNEZ	R1	Loop	

Register result status

Clock	**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
5	**72**	*Qi*	Load1		Mult1						

Loop Example Cycle 6

Instruction status						Executio	Write					
Instruction		j	k	iteration	Issue	complete	Result		Busy	Address		
LD	F0	0	R1	1	1			Load1	Yes	80		
MULT	F4	F0	F2	1	2			Load2	Yes	72		
SD	F4	0	R1	1	3			Load3	No		Qi	
LD	F0	0	R1	2	6			Store1	Yes	80	Mult1	
MULT	F4	F0	F2	2				Store2	No			
SD	F4	0	R1	2				Store3	No			
Reservation Stations					S1	S2	RS for j	RS for k				
	Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD		R(F2)	Load1		SUBI	R1	R1	#8
	0	Mult2	No						BNEZ	R1	Loop	
Register result status												
Clock		**R1**		F0	F2	F4	F6	F8	F10	F12	...	F30
6		**72**	Qi	Load2		Mult1						

- Note: F0 never sees Load1 result

Loop Example Cycle 7

Instruction status						Executio	Write					
Instruction		j	k	iteration	Issue	complete	Result		Busy	Address		
LD	F0	0	R1	1	1			Load1	Yes	80		
MULT	F4	F0	F2	1	2			Load2	Yes	72		
SD	F4	0	R1	1	3			Load3	No		Qi	
LD	F0	0	R1	2	6			Store1	Yes	80	Mult1	
MULT	F4	F0	F2	2	7			Store2	No			
SD	F4	0	R1	2				Store3	No			
Reservation Stations					S1	S2	RS for j	RS for k				
	Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD		R(F2)	Load1		SUBI	R1	R1	#8
	0	Mult2	Yes	MULTD		R(F2)	Load2		BNEZ	R1	Loop	
Register result status												
Clock		R1		F0	F2	F4	F6	F8	F10	F12	...	F30
7		72	Qi	Load2		Mult2						

- Note: MULT2 has no registers names in RS

Loop Example Cycle 8

Instruction status

Instruction		j	k	iteration	Issue	Executio	Write
						complete	Result
LD	F0	0	R1	1	1		
MULT	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0	R1	2	6		
MULT	F4	F0	F2	2	7		
SD	F4	0	R1	2	8		

	Busy	Address	Qi
Load1	Yes	80	
Load2	Yes	72	
Load3	No		
Store1	Yes	80	Mult1
Store2	Yes	72	Mult2
Store3	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		R(F2)	Load1	
0	Mult2	Yes	MULTD		R(F2)	Load2	

Code:

LD	F0	0	R1
MULT	F4	F0	F2
SD	F4	0	R1
SUBI	R1	R1	#8
BNEZ	R1	Loop	

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
8	**72**	Qi	Load2		Mult2						

Loop Example Cycle 9

Instruction status

						Executio	*Write*
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*
LD	F0	0	R1	1	1	9	
MULT	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0	R1	2	6		
MULT	F4	F0	F2	2	7		
SD	F4	0	R1	2	8		

	Busy	Address	Qi
Load1	Yes	80	
Load2	Yes	72	
Load3	No		
Store1	Yes	80	Mult1
Store2	Yes	72	Mult2
Store3	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		R(F2)	Load1	
0	Mult2	Yes	MULTD		R(F2)	Load2	

Code:

LD	F0	0	R1
MULT	F4	F0	F2
SD	F4	0	R1
SUBI	R1	R1	#8
BNEZ	R1	Loop	

Register result status

Clock	**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
9	**64**	*Qi*	Load2		Mult2						

- Load1 completing; what is waiting for it?

Loop Example Cycle 10

Instruction status

Instruction		j	k	iteration	Issue	Executio complete	Write Result
LD	F0	0	R1	1	1	9	10
MULT	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0	R1	2	6	10	
MULT	F4	F0	F2	2	7		
SD	F4	0	R1	2	8		

	Busy	Address	Qi
Load1	No		
Load2	Yes	72	
Load3	No		
Store1	Yes	80	Mult1
Store2	Yes	72	Mult2
Store3	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					
4	Mult1	Yes	MULTD	M(80)	R(F2)		
0	Mult2	Yes	MULTD		R(F2)	Load2	

Code:

```
LD    F0    0   R1
MULT  F4    F0  F2
SD    F4    0   R1
SUBI  R1    R1  #8
BNEZ  R1    Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
10	64	Qi	Load2		Mult2						

- Load2 completing; what is waiting for it?

Loop Example Cycle 11

Instruction status						*Executio*	*Write*					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2			Load2	No			
SD	F4	0	R1	1	3			Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	Yes	80	Mult1	
MULT	F4	F0	F2	2	7			Store2	Yes	72	Mult2	
SD	F4	0	R1	2	8			Store3	No			
Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	3	Mult1	Yes	MULTD	M(80)	R(F2)			SUBI	R1	R1	#8
	4	Mult2	Yes	MULTD	M(72)	R(F2)			BNEZ	R1	Loop	
Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
11		**64**	*Qi*	Load3		Mult2						

Loop Example Cycle 12

Instruction status						*Executio*	*Write*					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2			Load2	No			
SD	F4	0	R1	1	3			Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	Yes	80	Mult1	
MULT	F4	F0	F2	2	7			Store2	Yes	72	Mult2	
SD	F4	0	R1	2	8			Store3	No			
Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	2	Mult1	Yes	MULTD	M(80)	R(F2)			SUBI	R1	R1	#8
	3	Mult2	Yes	MULTD	M(72)	R(F2)			BNEZ	R1	Loop	
Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
12		**64**	*Qi*	Load3		Mult2						

Loop Example Cycle 13

Instruction status						Executioi	Write					
Instruction		j	k	iteration	Issue	complete	Result		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2			Load2	No			
SD	F4	0	R1	1	3			Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	Yes	80	Mult1	
MULT	F4	F0	F2	2	7			Store2	Yes	72	Mult2	
SD	F4	0	R1	2	8			Store3	No			
Reservation Stations					S1	S2	RS for j	RS for k				
	Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	1	Mult1	Yes	MULTD	M(80)	R(F2)			SUBI	R1	R1	#8
	2	Mult2	Yes	MULTD	M(72)	R(F2)			BNEZ	R1	Loop	
Register result status												
Clock		**R1**		F0	F2	F4	F6	F8	F10	F12	…	F30
13		**64**	Qi	Load3		Mult2						

Loop Example Cycle 14

Instruction status						Executio	Write					
Instruction		j	k	iteration	Issue	complete	Result		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2	14		Load2	No			
SD	F4	0	R1	1	3			Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	Yes	80	Mult1	
MULT	F4	F0	F2	2	7			Store2	Yes	72	Mult2	
SD	F4	0	R1	2	8			Store3	No			
Reservation Stations					S1	S2	RS for j	RS for k				
	Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD	M(80)	R(F2)			SUBI	R1	R1	#8
	1	Mult2	Yes	MULTD	M(72)	R(F2)			BNEZ	R1	Loop	
Register result status												
Clock		R1		F0	F2	F4	F6	F8	F10	F12	...	F30
14		64	Qi	Load3		Mult2						

- Mult1 completing; what is waiting for it?

Loop Example Cycle 15

Instruction status						*Executior*	*Write*					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2	14	15	Load2	No			
SD	F4	0	R1	1	3			Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	Yes	80	M(80)*R(F	
MULT	F4	F0	F2	2	7	15		Store2	Yes	72	Mult2	
SD	F4	0	R1	2	8			Store3	No			
Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	No						SUBI	R1	R1	#8
	0	Mult2	Yes	MULTD	M(72)	R(F2)			BNEZ	R1	Loop	
Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
15		**64**	*Qi*	Load3		Mult2						

- Mult2 completing; what is waiting for it?

Loop Example Cycle 16

Instruction status						*Executioi*	*Write*					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2	14	15	Load2	No			
SD	F4	0	R1	1	3			Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	Yes	80	M(80)*R(F	
MULT	F4	F0	F2	2	7	15	16	Store2	Yes	72	M(72)*R(7	
SD	F4	0	R1	2	8			Store3	No			
Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD		R(F2)	Load3		SUBI	R1	R1	#8
	0	Mult2	No						BNEZ	R1	Loop	
Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
16		**64**	*Qi*	Load3		Mult1						

Loop Example Cycle 17

Instruction status						Executioi	Write					
Instruction		j	k	iteration	Issue	complete	Result		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2	14	15	Load2	No			
SD	F4	0	R1	1	3			Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	Yes	80	M(80)*R(F	
MULT	F4	F0	F2	2	7	15	16	Store2	Yes	72	M(72)*R(7	
SD	F4	0	R1	2	8			Store3	Yes	64	Mult1	
Reservation Stations					S1	S2	RS for j	RS for k				
	Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD		R(F2)	Load3		SUBI	R1	R1	#8
	0	Mult2	No						BNEZ	R1	Loop	
Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
17		**64**	*Qi*	Load3		Mult1						

Loop Example Cycle 18

Instruction status						Executioi	Write					
Instruction		j	k	iteration	Issue	complete	Result		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2	14	15	Load2	No			
SD	F4	0	R1	1	3	18		Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	Yes	80	M(80)*R(F	
MULT	F4	F0	F2	2	7	15	16	Store2	Yes	72	M(72)*R(7	
SD	F4	0	R1	2	8			Store3	Yes	64	Mult1	
Reservation Stations					S1	S2	RS for j	RS for k				
	Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD		R(F2)	Load3		SUBI	R1	R1	#8
	0	Mult2	No						BNEZ	R1	Loop	
Register result status												
Clock		R1		F0	F2	F4	F6	F8	F10	F12	...	F30
18		**56**	Qi	Load3		Mult1						

Loop Example Cycle 19

Instruction status						Executioi	Write					
Instruction		j	k	iteration	Issue	complete	Result		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2	14	15	Load2	No			
SD	F4	0	R1	1	3	18	19	Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	No			
MULT	F4	F0	F2	2	7	15	16	Store2	Yes	72	M(72)*R(7	
SD	F4	0	R1	2	8			Store3	Yes	64	Mult1	
Reservation Stations					S1	S2	RS for j	RS for k				
	Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD		R(F2)	Load3		SUBI	R1	R1	#8
	0	Mult2	No						BNEZ	R1	Loop	
Register result status												
Clock		R1		F0	F2	F4	F6	F8	F10	F12	...	F30
19		56	Qi	Load3		Mult1						

Loop Example Cycle 20

Instruction status					*Executior*	*Write*						
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2	14	15	Load2	No			
SD	F4	0	R1	1	3	18	19	Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	No			
MULT	F4	F0	F2	2	7	15	16	Store2	Yes	72	M(72)*R(7	
SD	F4	0	R1	2	8	20		Store3	Yes	64	Mult1	
Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD		R(F2)	Load3		SUBI	R1	R1	#8
	0	Mult2	No						BNEZ	R1	Loop	
Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
20		**56**	*Qi*	Load3		Mult1						

Loop Example Cycle 21

Instruction status						*Executioi*	*Write*					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2	14	15	Load2	No			
SD	F4	0	R1	1	3	18	19	Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	No			
MULT	F4	F0	F2	2	7	15	16	Store2	No			
SD	F4	0	R1	2	8	20	21	Store3	Yes	64	Mult1	
Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD		R(F2)	Load3		SUBI	R1	R1	#8
	0	Mult2	No						BNEZ	R1	Loop	
Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
21		**56**	*Qi*	Load3		Mult1						

Conclusion

- ❑ ***Summary***
  - ➔ The Tomasulo dynamic instruction scheduling algorithm
    - Reservations stations: renaming to larger set of registers + buffering source operands
    - Prevents registers as bottleneck
    - Avoids WAR, WAW hazards of Scoreboard
    - Allows loop unrolling in HW
    - HW exploiting ILP
    - Not limited to basic blocks (integer units gets ahead, beyond branches)
    - Helps cache misses as well
- ❑ ***Next Lecture***
  - ➔ Control hazards and ILP
  - ➔ Reducing branch penalties with dynamic H/W prediction

Reading assignment includes section 3.2 in the textbook