Lecture 6: Scoreboarding and Tomasulo Algorithm

History

- 1966: scoreboarding in CDC6600, implementing limited dynamic scheduling
- Three years later: Tomasulo in IBM 360/91, introducing register renaming and reservation station
- Now appearing in todays Dec Alpha, SGI MIPS, SUN UltraSparc, Intel Pentium, IBM PowerPC, and others

Scoreboarding Overview

Basic idea:

- Use scoreboard to track data (RAW) dependence through register

Main points of design:

- Instructions are sent to FU unit if there is no outstanding name dependence
- RAW data dependence is tracked and enforced by scoreboard
- Register values are passed through the register file; a child instruction starts execution after the last parent finishes execution
- Pipeline stalls if any name dependence (WAR or WAW) is detected

Machine Correctness

E(D,P) = E(S,P) if

1. E(D,P) and E(S,P) execute the same set of instructions
2. **For any inst *i*, *i* receives the outputs in E(D,P) of its parents in E(S,P)**
3. In E(D,P) any register or memory word receives the output of inst *j*, where *j* is the last instruction writes to the register or memory word in E(S,P)

Scoreboarding merit: Be able to execute independent instructions in parallel without violating statement 2.

Four Stages of Scoreboard Control

Fetch first, then

1. Issue—decode instructions & check for structural hazards
   - Wait conditions: (1) the required FU is free; (2) no other instruction writes to the same register destination (to avoid WAW)
   - Actions: (1) the instruction proceeds to the FU; (2) scoreboard updates its internal data structure

   If an instruction is stalled at this stage, no other instructions can proceed
2. Read operands—wait until no data hazards, then read operands
   - Wait conditions: all source operands are available
   - Actions: the function unit reads register operands and start execution the next cycle

Four Stages of Scoreboard Control

3. Execution—operate on operands (EX)
   - Actions: The functional unit begins execution upon receiving operands. When the result is ready, it notifies the scoreboard that it has completed execution.
4. Write result—finish execution (WB)
   - Wait condition: no other instruction/FU is going to read the register destination of the instruction
   - Actions: Write the register and update the scoreboard

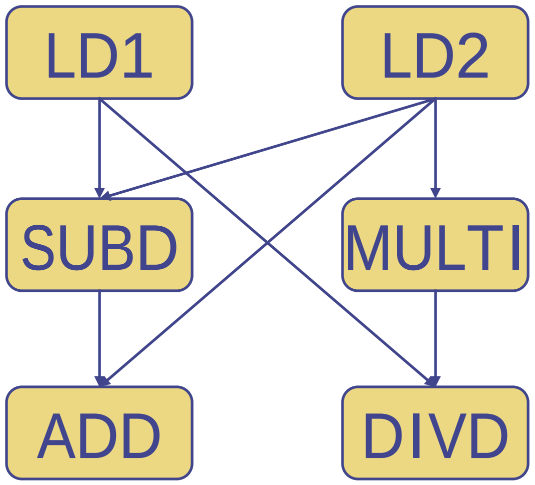
   WAR Example:

```
DIVD   F0,F2,F4
ADDD   F10,F0,F8
SUBD   F8,F8,F14
```

   CDC 6600 scoreboard would stall SUBD until ADDD reads operands

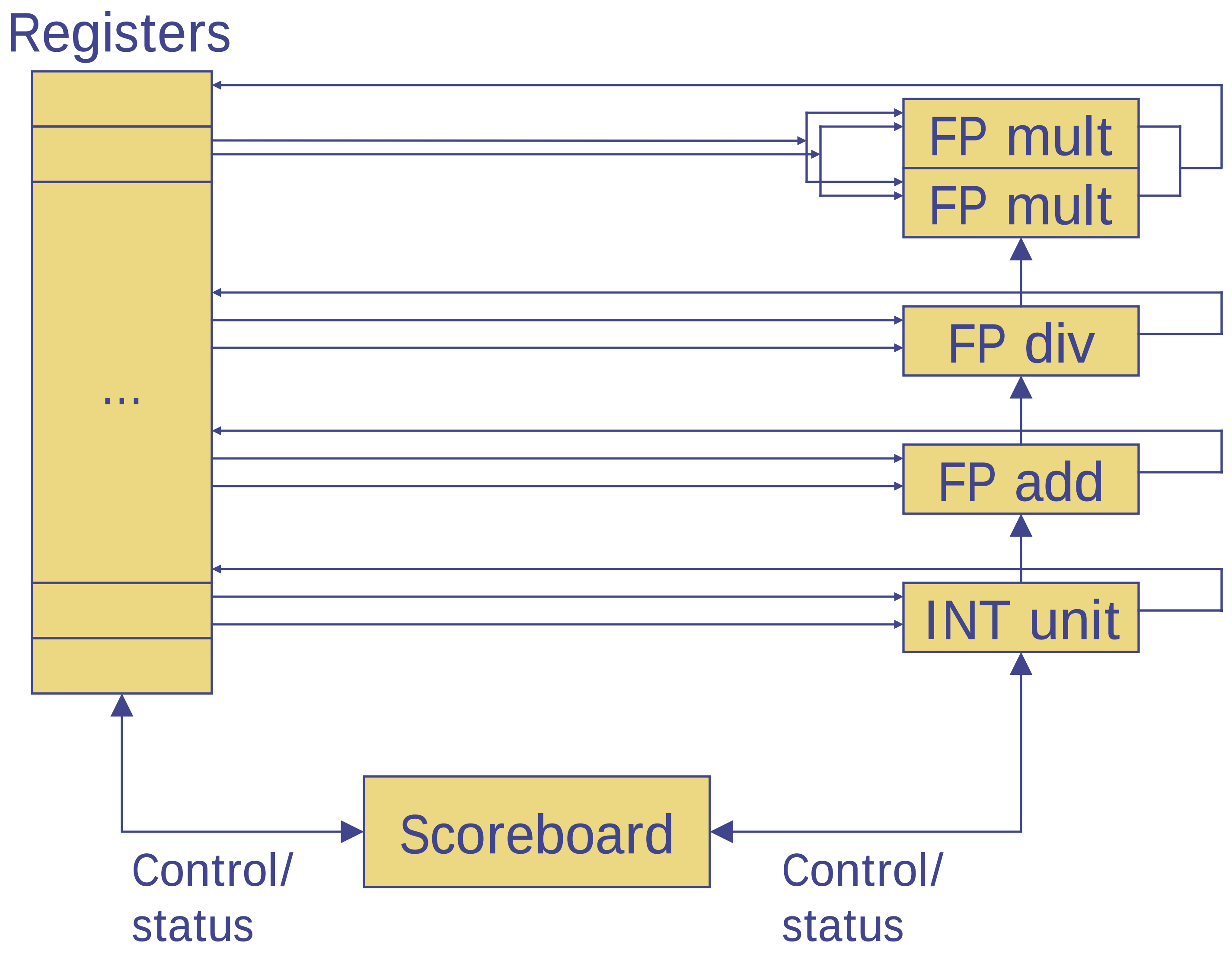
Code Example

```
LD     F6,34(R2)
LD     F2,45(R3)
MULTI F0,F2,F4
SUBD  F8,F6,F2
DIVD  F10,F0,F6
ADD   F6,F8,F2
```



Operation latencies: load/store 2 cycles,
Add/sub 2 cycles, Mult 10 cycles, divide 40 cycle

Scoreboard Connections



Three Parts of the Scoreboard

1. Instruction status—which of 4 steps the instruction is in

2. Functional unit status—Indicates the state of the functional unit (FU). 9 fields for each functional unit
   - Busy—Indicates whether the unit is busy or not
   - Op—Operation to perform in the unit (e.g., + or –)
   - Fi—Destination register
   - Fj, Fk—Source-register numbers
   - Qj, Qk—Functional units producing source registers Fj, Fk
   - Rj, Rk—Flags indicating when Fj, Fk are ready

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

Scoreboard Example

Instruction status					*Read*	*Executi*	*Write*
Instruction		*j*	*k*	*Issue*	*operanc*	*complet*	*Result*
LD	F6	34+	R2				
LD	F2	45+	R3				
MULT	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

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

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

Scoreboard Example Cycle 1

Instruction status

Instruction		j	k	Issue	Read operand	Executic complet	Write 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				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

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

Scoreboard Example Cycle 2

Instruction status					Read	Executic	Write
Instruction		*j*	*k*	*Issue*	*operand*	*complet*	*Result*
LD	F6	34+	R2	1	2		
LD	F2	45+	R3				
MULT	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Functional unit status				*dest*	*S1*	*S2*	*FU for j*	*FU for k*	*Fj?*	*Fk?*
Time	*Name*	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
	Integer	Yes	Load	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

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

- **Issue 2nd LD?**

Scoreboard Example Cycle 3

Instruction status					Read	Executic	Write
Instruction		*j*	*k*	*Issue*	*operand*	*complet*	*Result*
LD	F6	34+	R2	1	2	3	
LD	F2	45+	R3				
MULT	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Functional unit status				*dest*	*S1*	*S2*	*FU for j*	*FU for k*	*Fj?*	*Fk?*
Time	*Name*	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
	Integer	Yes	Load	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

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

- **Issue MULT?**

Scoreboard Example Cycle 4

Instruction status

Instruction		j	k	Issue	Read operand	Executic complet	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3				
MULT	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

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

Scoreboard Example Cycle 5

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

Functional unit status				*dest*	*S1*	*S2*	*FU for j*	*FU for k*	*Fj?*	*Fk?*
Time	*Name*	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
	Integer	Yes	Load	F2		R3				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

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

Scoreboard Example Cycle 6

Instruction status

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

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F2		R3				Yes
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

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

Scoreboard Example Cycle 7

Instruction status

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

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F2		R3				Yes
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2		Integer	Yes	No
	Divide	No								

Register result status

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

- **Read multiply operands?**

Scoreboard Example Cycle 8a

Instruction status

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

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F2		R3				Yes
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2		Integer	Yes	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

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

Scoreboard Example Cycle 8b

Instruction status

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

Functional unit status

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

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
8	FU	Mult1				Add	Divide			

Scoreboard Example Cycle 9

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

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

Register result status

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
9	*FU*	Mult1				Add	Divide			

- **Read operands for MULT & SUBD? Issue ADDD?**

Scoreboard Example Cycle 11

Instruction status

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

Functional unit status

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

Register result status

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

Scoreboard Example Cycle 12

Instruction status

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

Functional unit status

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

Register result status

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
12	*FU*	Mult1					Divide			

- **Read operands for DIVD?**

Scoreboard Example Cycle 13

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

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								
6	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

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

Scoreboard Example Cycle 14

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

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								
5	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
2	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

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

Scoreboard Example Cycle 15

Instruction status

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

Functional unit status

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

Register result status

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

Scoreboard Example Cycle 16

Instruction status

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

Functional unit status

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

Register result status

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
16	*FU*	Mult1			Add		Divide			

Scoreboard Example Cycle 17

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

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								
2	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

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

- **Write result of ADDD?**

Scoreboard Example Cycle 18

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

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								
1	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

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

Scoreboard Example Cycle 19

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

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								
0	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
19	*FU*	Mult1			Add		Divide			

Scoreboard Example Cycle 20

Instruction status

Instruction		*j*	*k*	*Issue*	*Read operand*	*Executic complet*	*Write 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			
ADDD	F6	F8	F2	13	14	16	

Functional unit status

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

Register result status

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
20	*FU*				Add		Divide			

Scoreboard Example Cycle 21

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

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	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
21	*FU*				Add		Divide			

Scoreboard Example Cycle 22

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

Functional unit status				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								
40	Divide	Yes	Div	F10	F0	F6			Yes	Yes

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

Scoreboard Example Cycle 61

Instruction status					Read	Executic	Write
Instruction		*j*	*k*	*Issue*	*operand*	*complet*	*Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULT	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8	21	61	
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	Yes	Div	F10	F0	F6			Yes	Yes

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

Scoreboard Example Cycle 62

Instruction status					Read	Executic	Write
Instruction		*j*	*k*	*Issue*	*operand*	*complet*	*Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULT	F0	F2	F4	6	9	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*									

Scoreboard Scheduling

Inst	Issue	Read operands	Execution complete	Write Result
LD	1	2	3	4
LD	5	6	7	8
MULT	6	9	19	20
SUBD	7	9	11	12
DIVD	8	21	61	62
ADDD	13	14	16	22

CDC 6600 Scoreboard

- Speedup 1.7 from compiler; 2.5 by hand BUT slow memory (no cache) limits benefit
- First implement for dynamic scheduling (though limited)
- Limitations of 6600 scoreboard as for dynamic scheduling
  - Stall on name dependence (WAR and WAW), which is not really necessary
  - Instruction parallelism is limited by # of function units
  - No forwarding hardware

Tomasulo Overview

Basic Idea:

- Remove name dependence by renaming register in execution
- Introduce tag-broadcasting in instruction scheduling

Main point of design

- Instructions are decoded and then renamed
- Renamed instructions are sent to reservation stations
- Reservation stations track and enforce register data (RAW) dependences
- A child instruction can start execution after the last parent finishes writeback and does broadcasting; in this case, register values are passed through broadcasting
- Prevent an early register write from overwriting the value of a later register write to enforce name dependence (WAR and WAW)

Three Stages of Tomasulo Algorithm

After fetch and decode,

1. Issue—get instruction from FP Op Queue

   If reservation station free (no structural hazard), control issues instr & 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

Issue: build dependence for new inst
Writeback: Wakeup dependent instructions

Issue Stage and Renaming Table

- Renames its two source registers (source renaming)
- Assigns it to a free RS
- Updates Renaming table (dest renaming)
- Also decodes the inst and read register values in parallel

How would the following inst be renamed?

```
ADD $16, $8, $9
ADD $17, $16, $16
```

Execute Stage

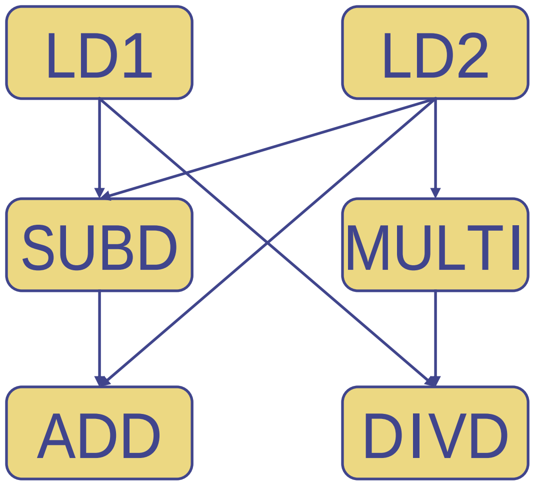
- Only "ready" instructions can join the competition
- There is a select logic to select instructions for FU execution
  - Some policy may be used, e.g. age based
- Non-ready instructions can be "waken up" during writeback of its parent inst

Writeback and Common Data Bus

- Normal data bus: data + destination ("go to" bus)
- Common data bus: data + source ("come from" bus)
  - 64 bits of data + 4 bits of source index (tag)
  - Does the broadcast to every instruction in the fly
- Child instructions do tag matching and update their ready bits and value fields (if the tag matches theirs)

Code Example

```
LD     F6,34(R2)
LD     F2,45(R3)
MULTI F0,F2,F4
SUBD  F8,F6,F2
DIVD  F10,F0,F6
ADD   F6,F8,F2
```



Operation latencies: load/store 2 cycles,
Add/sub 2 cycles, Mult 10 cycles, divide 40 cycle

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			
MULT	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*									

Tomasulo Example Cycle 1

Instruction status

Instruction	j	k		Issue	Execution complete	Write 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

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k 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		
MULT	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					

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		
MULT	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

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS for j Qj*	*RS for k 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: registers names are removed (“renamed”) in Reservation Stations**
- **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		
MULT	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			
MULT	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			
MULT	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			

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			
MULT	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

Instruction		j	k	Issue	Execution complete	Write 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			Load3	No	
SUBD	F8	F6	F2	4	7	8			
DIVD	F10	F0	F6	5					
ADDD	F6	F8	F2	6					

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k 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	

Register result statu

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			
MULT	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()萠()	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()萠()	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			
MULT	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()蓢()	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()蓢()	Mult2			

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

Tomasulo Example Cycle 11

Instruction status

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

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

Register result statu

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
11	FU	Mult1	M(45+R3)		(M-M)+M()	M()ŠM()	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			
SUB	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADD	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	MULT	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			

Tomasulo Example Cycle 13

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				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									
	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萠)+M()	M()萠()	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			
MULT	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									
	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萠)+M()	M()萠()	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			
MULT	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萠)+M()	M()萠()	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			
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									
	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萠)+M()	M()萠()	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萠)+M()	M()萠()	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			
MULT	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朚)+M()	M()朚()	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			
MULT	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萠)+M()	M()萠()	M*F4/M			

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

Review Dependences

How are dependences are enforced or removed in Tomasulo Algorithm?

- Data dependences (RAW)
- Antidependence (WAR)
- Output Dependence (WAW)

Dependences can be through register or memory.

Data Dependence Through Register

For any inst *i*, *i* receives the outputs in E(D,P) of its parents in E(S,P)

Assume i is dependent on k through register Rx, how does i receives k's output?

1. If k.WriteResult $\rightarrow$ i.Issue:
2. If i.Issue $\rightarrow$ k.WriteResult:
3. If i.Issue $\leftrightarrow$ k.WriteResult:

In all cases, i must receive the right value.

Name Dependence Through Register

For any inst *i*, *i* receives the outputs in E(D,P) of its parents in E(S,P)

In E(D,P) any register or memory word receives the output of inst *j*, where *j* is the last instruction writes to the register or memory word in E(S,P)

Assume *i* is prior to *j*.

1. WAR dependence, i reads some Rx and j writes to it

What would happen if j.WriteResult → i.Issue or j.WriteResult → i.Execute?

2. WAW dependence, both i and j writes to some Rx

What would happen to the register content if j.WriteResult → i.WriteResult?

What is Tag

Tag is a modern name

In Tomasulo, RS or load/store buffer index is used as tag.

- Renaming assign new inst a unique tag
- RS stores tags to preserve dependences
- CDB broadcasts tag with data for data passing and wakeup

- *Why tag is so chosen?*
- *What does tag really represent?*
- *What can be used as tag?*

Tomasulo Summary

- Reservations stations:
  - Increases effective register number
  - Distributes scheduling logic
- Register renaming: Avoids WAR and WAW dependence
- Tag + Data broadcasting for waking up child instructions
- Pros: can be effectively combined with speculative execution
- Cons: CDB broadcasting adds one-cycle delay