Instruction-level parallelism: Scoreboard

Dr. Tao Xie

These slides are adapted from notes by Dr. David Patterson (UCB) and Dr. Xiao Qin (Auburn).

HW Scheme: Dynamic Scheduling

- Static Scheduling

  Compiler techniques for scheduling

  - separate dependent instructions
  - minimize the number of hazard and stalls

  e.g.: static branch prediction

- Dynamic Scheduling

  1. Uses hardware to rearrange instructions to reduce stalls
  2. Works when can't know real dependence at compile time
  3. Compiler simpler
  4. Code for one pipeline runs well on another pipeline

The Key idea of Dynamic Scheduling

- Key Idea: Allow instructions behind stall to proceed. => Instructions executing in parallel. There are multiple execution units, so use them.

```
DIVD  F0, F2, F4
ADDD  F10, F0, F8
SUBD  F12, F8, F14
```

ADDD stalls. Will SUBD stall?

Even though ADDD stalls, the SUBD has no dependencies and can run.

  - Enables out-of-order execution => out-of-order completion

Dynamic pipeline scheduling overcomes the limitations of in-order pipelined execution by allowing out-of-order instruction execution.

Dynamic Scheduling with a Scoreboard

- The scoreboard is a centralized hardware mechanism
  - In order to execute an instruction as soon as its operands are available and no hazard conditions prevent it.

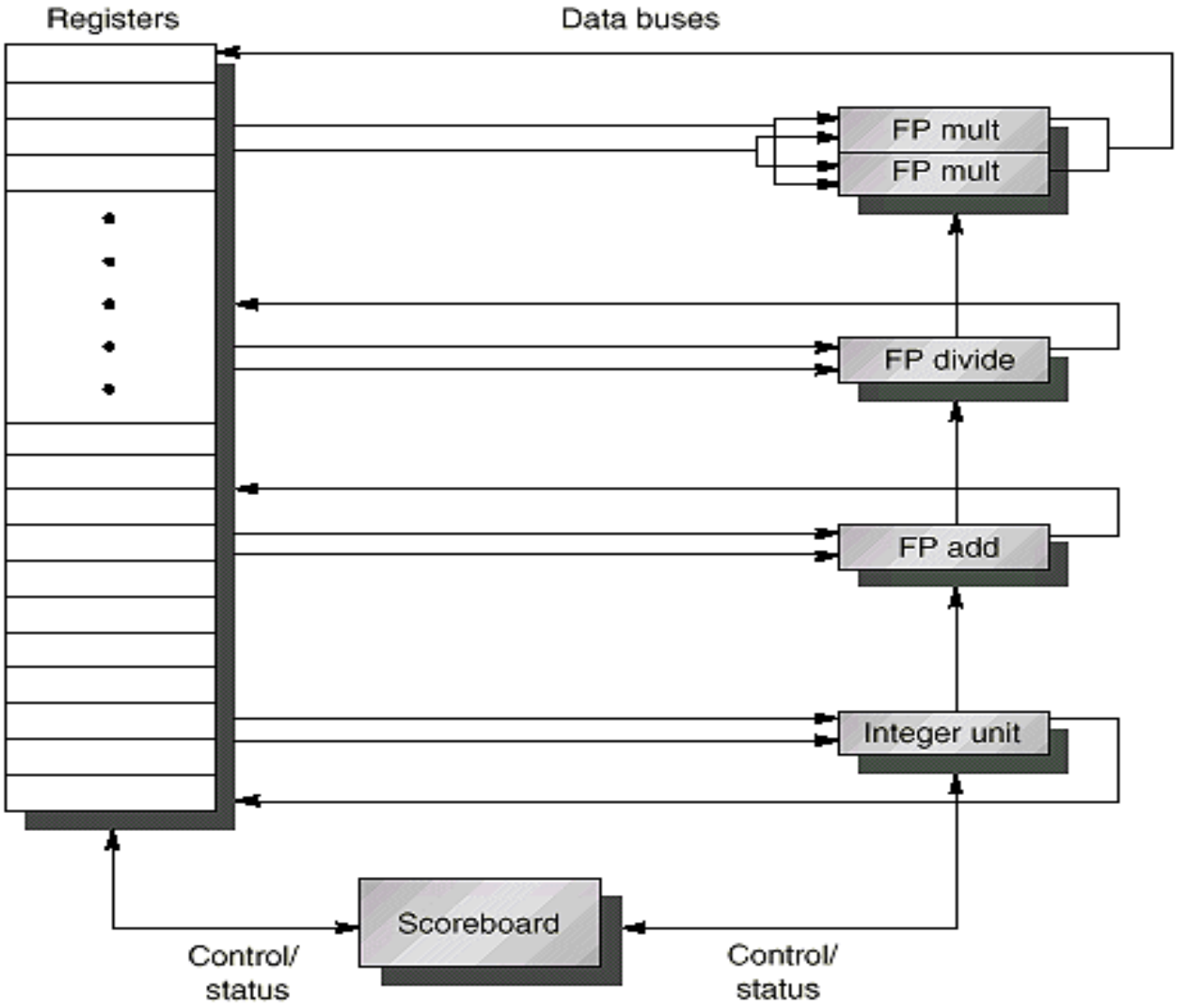
- It dynamically constructs the dependency graph by hardware for a window of instructions as they are issued in program order.

- A scoreboard is a "data structure" that provides the information necessary for all pieces of the processor to work together.

The Key idea of Scoreboards

- 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
- Scoreboards allow instruction to execute whenever 1 & 2 hold, not waiting for prior instructions.
- We will use In order issue, out of order execution, out of order commit (also called completion)
- First used in CDC6600 in 1963. Our example modified here for MIPS.
- CDC had 4 FP units, 5 memory reference units, 7 integer units.
- MIPS has 2 FP multipliers, 1 FP adder, 1 FP divider, 1 integer unit.

Typical Scoreboard Structure



2 FP multipliers, 1 FP adder, 1 FP divider, 1 integer unit

Using A Scoreboard: 4 stages

1. Issue —decode instructions & check for structural & WAW hazards (ID1)

Always done in program order

If a functional unit for the instruction is free (*no structural hazards*) and no other active instruction has the same destination register (*no WAW*), the scoreboard issues the instruction to the functional unit and updates its internal data structure.

If a structural or WAW hazard exists, then the instruction issue stalls, and no further instructions will issue until these hazards are cleared.

2. Read operands —wait until no data hazards, then read operands (ID2)

Can be done out of program order

A source operand is available if no earlier issued active instruction is going to write it, or if the register containing the operand is being written by a currently active functional unit (*no RAW*).

When the source operands are available, the scoreboard tells the functional unit to proceed to read the operands from the registers and begin execution. The scoreboard resolves RAW hazards dynamically in this step, and instructions may be sent into execution out of order.

Using A Scoreboard: 4 stages (cont.)

3. Execution — operate on operands (EX)

   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)

   Once the scoreboard is aware that the functional unit has completed execution, the scoreboard checks for *WAR* hazards. If none, it writes results. If WAR, then it stalls the instruction.

   Example:

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

   Scoreboard would stall SUBD until ADDD reads operands

Using A Scoreboard: 3 parts

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. Set to No after operands are read.

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

A Scoreboard Example

The following code is run on the MIPS with a scoreboard given earlier with:

Functional Unit (FU)	# of FUs	EX cycles
Integer	1	1
Floating Point Multiply	2	10
Floating Point add	1	2
Floating point Divide	1	40

All functional units are not pipelined
(similar to CDC6600)

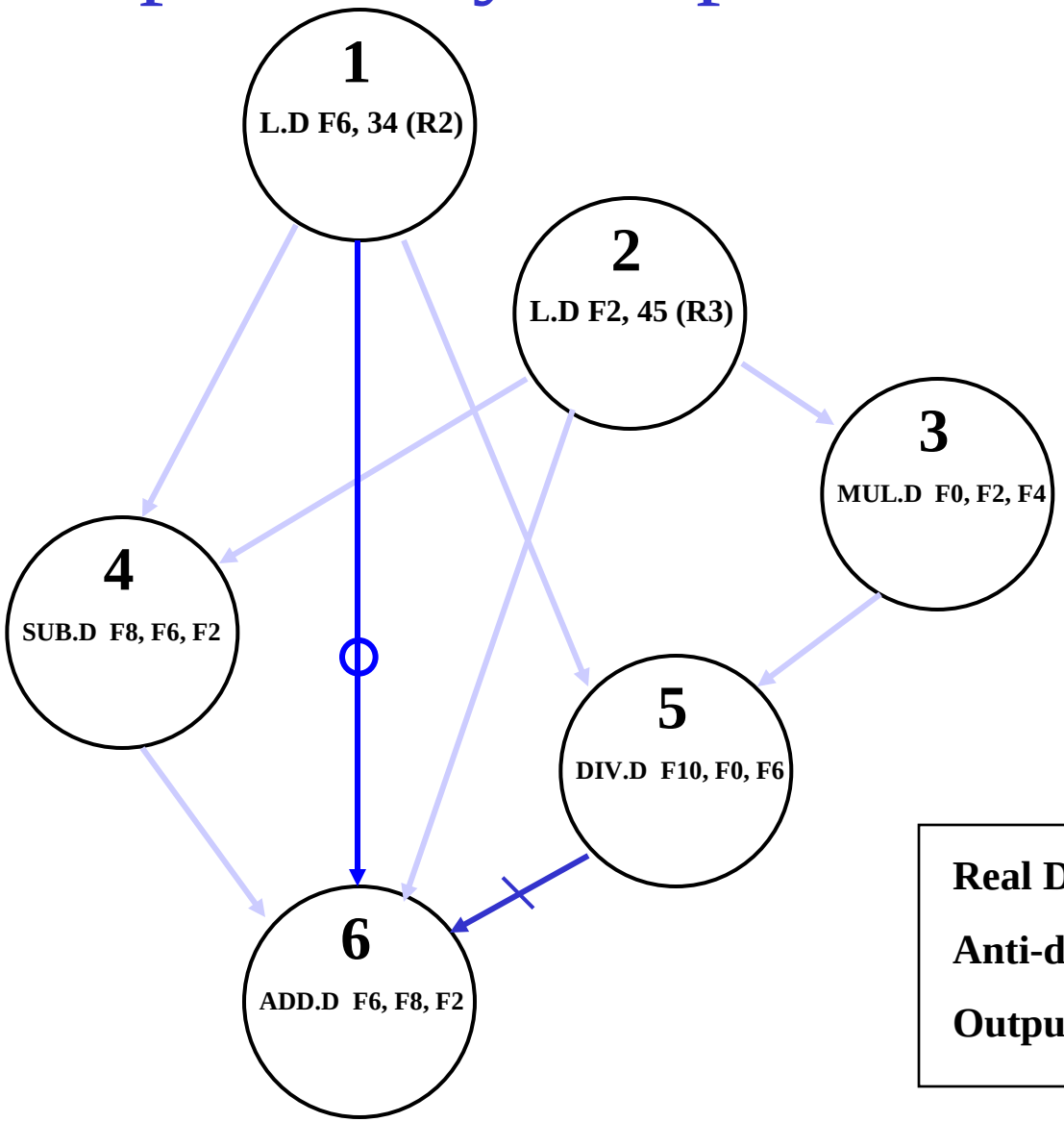
```
L.D      F6, 34(R2)
L.D      F2, 45(R3)
MUL.D    F0, F2, F4
SUB.D    F8, F6, F2
DIV.D    F10, F0, F6
ADD.D    F6, F8, F2
```

Real Data Dependence (RAW)

Anti-dependence (WAR)

Output Dependence (WAW)

Dependency Graph For Example Code



Example Code

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

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

Output Dependence:
(1, 6)

Anti-dependence:
(5, 6)

Real Data Dependence (RAW)

Anti-dependence (WAR)

Output Dependence (WAW)

Scoreboard Example

Instruction status

Instruction		j	k	Issue	Read operanc	Executi complet	Write 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

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	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				

Issue LD #1

Shows in which cycle the operation occurred.

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

Instruction		j	k	Issue	Read operand	Executic complet	Write 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				

LD #2 can't issue since ?

integer unit is busy.

MULT can't issue because

we require in-order issue.

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
2	FU				Integer					

Scoreboard Example Cycle 3

Instruction status

Instruction		*j*	*k*	*Issue*	*Read operand*	*Execution complete*	*Write 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

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*
3	*FU*				Integer					

Scoreboard Example Cycle 4

Instruction status

Instruction		*j*	*k*	*Issue*	*Read operand*	*Execution complete*	*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

Issue LD #2 since integer unit is now free.

Instruction status

Instruction		j	k	Issue	Read operand	Executio complet	Write 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

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	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

Issue MULT.

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

Can MULT read its operands?

(F2) because LD #2 hasn't finished.

Instruction status

Instruction		*j*	*k*	*Issue*	*Read operand*	*Executio 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				

Scoreboard Example Cycle 8a

DIVD issues.
MULT and SUBD both waiting for F2.

Instruction status

Instruction		j	k	Issue	Read operand	Execution complete	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

LD #2 writes F2.

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

Instruction		j	k	Issue	Read operand	Executio 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		
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Now MULT and SUBD can both read F2.
How can both instructions do this at the same time??

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								
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			

Scoreboard Example Cycle 11

ADDD can't start because?

add unit is busy.

Instruction status

Instruction		j	k	Issue	Read operand	Executio 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

SUBD finishes.
DIVD waiting for F0.

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			

Scoreboard Example Cycle 13

ADDD issues.

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			

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								
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

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								
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*	*Executio 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*	*Executio 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

Can ADDD write? Why?

No! Because of DIVD. WAR!

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								
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			

Scoreboard Example Cycle 18

Nothing Happens!!

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								
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

MULT completes execution.

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	
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								
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

MULT writes.

Instruction status

Instruction		j	k	Issue	Read operand	Executio 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

DIVD loads operands

Instruction status

Instruction		*j*	*k*	*Issue*	*Read operand*	*Executio 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	21		
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*
21 *FU*				Add		Divide			

Scoreboard Example Cycle 22

Now ADDD can write ?

Yes! Since WAR removed.

Instruction status

Instruction		j	k	Issue	Read operand	Executio 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	21		
ADDD	F6	F8	F2	13	14	16	22

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	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

DIVD completes execution

Instruction status

Instruction		j	k	Issue	Read operand	Executio 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	21	61	
ADDD	F6	F8	F2	13	14	16	22

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	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

DONE!!

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	21	61	62
ADDD	F6	F8	F2	13	14	16	22

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	No								
0	Divide	No								

Register result status

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

Detailed Scoreboard Pipeline Control

Instruction status	Wait until	Bookkeeping
Issue	Not Busy(FU) and not Result(D)	Busy(FU)← yes; Op(FU)← op; Fi(FU)← `D'; Fj(FU)← `S1'; Fk(FU)← `S2'; Qj← Result('S1'); Qk← Result(`S2'); Rj← not Qj; Rk← not Qk; Result('D')← FU;
Read operands	Rj and Rk	Rj← No; Rk← No
Execution complete	Functional unit done	
Write result	∀f((Fj(f)≠Fi(FU) or Rj(f)=No) & (Fk(f) ≠Fi(FU)or Rk(f)=No))	∀f(if Qj(f)=FU then Rj(f)← Yes); ∀f(if Qk(f)=FU then Rj(f)← Yes); Result(Fi(FU))← 0; Busy(FU)← No

Avoid Structural and WAW Hazards

Avoid RAW Hazards

Avoid WAR Hazards

Limitations of Scoreboard

- The amount of parallelism available among the instructions (chosen from the same basic block)
- The number of score entries (The size of the scoreboard determines the size of the window)
- The number and types of functional units (Structural hazards increase when dynamic scheduling is used)
- The presence of anti-dependence and output dependences lead to WAR and WAW stalls.

Summary

- Techniques to deal with data hazards in instruction pipelines by:
  - Result forwarding to reduce or eliminate RAW hazards
  - Hazard detection hardware to stall the pipeline during hazards
  - Compiler-based static scheduling to separate the dependent instructions minimizing actual hazard-prevention stalls in scheduled code.
  - Uses a hardware-based mechanism to rearrange instruction execution order to reduce stalls dynamically at runtime (dynamic scheduling)
    - Better dynamic exploitation of instruction-level parallelism (ILP)
    - We learned scoreboard techniques today
    - We will learn another technique Tomasulo next.