# Advanced Computer Architectures

(High Performance Processors and Systems)

# Dynamic Scheduling: Scoreboard

Politecnico di Milano v1

Alessandro Verosimile < alessandro.verosimile@polimi.it> Marco D. Santambrogio <marco.santambrogio@polimi.it>

### Instruction Level Parallelism

- Two strategies to support ILP:
  - Dynamic Scheduling: Depend on the hardware to locate parallelism
  - Static Scheduling: Rely on software for identifying potential parallelism
- Hardware intensive approaches dominate desktop and server markets

### Instruction Level Parallelism

- Two strategies to support ILP:
  - Dynamic Scheduling: Depend on the hardware to locate parallelism
  - Static Scheduling: Rely on software for identifying potential parallelism
- Hardware intensive approaches dominate desktop and server markets

# Key Idea: dynamic scheduling

- Problem:
  - data dependences that cannot be hidden with bypassing or forwarding cause hardware stalls of the pipeline
- Solution: allow instructions behind a stall to proceed
  - HW rearranges the instruction execution to reduce stalls
- Enables out-of-order execution and completion (commit)
  - Out-of order execution introduces possibility of WAR, WAW data hazards.
- First implemented in CDC6600 (1963)

### A Data Structure for Correct Issues Keeps track of the status of Functional Units

Name	Busy	Ор	Dest	Src1	Src2
Int					
Mem					
Add1					
Add2					
Add3					
Mult1					
Mult2					
Div					

The instruction i at the Issue stage consults this table

FU available? check the busy column

RAW? search the dest column for i's sources

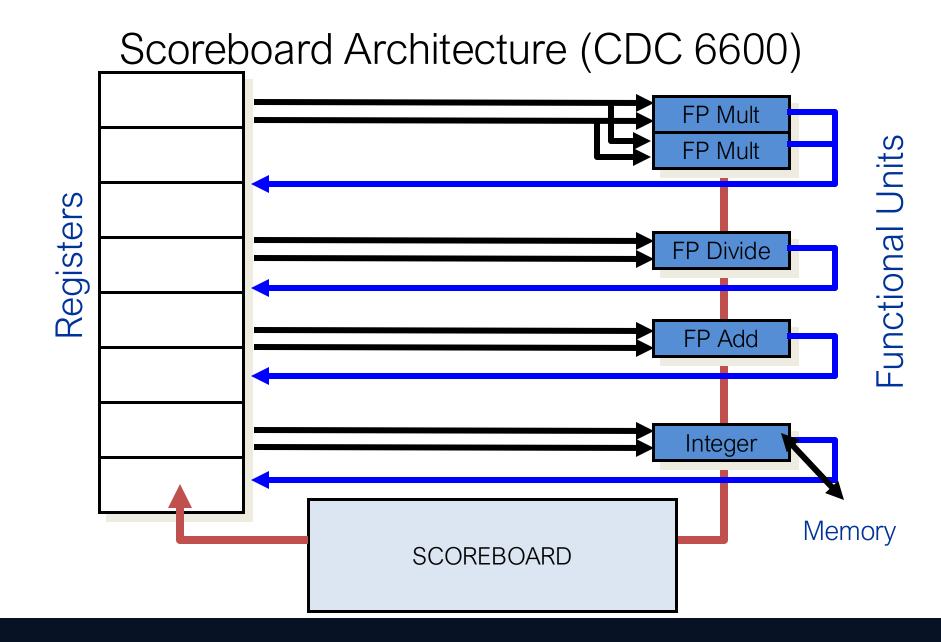
WAR? search the source columns for i's destination

WAW? search the dest column for i's destination

An entry is added to the table if no hazard is detected; An entry is removed from the table after Write-Back

### CDC6600 Scoreboard

- Instructions dispatched in-order to functional units provided no structural hazard or WAW
  - Stall on structural hazard, no functional units available
  - Only one pending write to any register
- Instructions wait for input operands (RAW hazards) before execution
  - Can execute out-of-order
- Instructions wait for output register to be read by preceding instructions (WAR)
  - Result held in functional unit until register free



# Scoreboard Operation

- Scoreboard centralizes hazard management
  - Every instruction goes through the scoreboard
  - Scoreboard determines when the instruction can read its operands and begin execution
  - Monitors changes in hardware and decides when a stalled instruction can execute
  - Controls when instructions can write results
- New pipeline

11	D	EX	WB		
Issue	Read Regs	Execution	Write		

### Scoreboard Scheme

- ID stage splitted in two parts:
  - Issue (decode and check structural hazard)
  - Read Operands (wait until no data hazards)
- In-order issue BUT out-of-order read-operands
- Scoreboard allows instructions without dependencies to execute

### Issue

Decode instructions & check for structural hazards.

- ✓ Instructions issued in program order (for hazard checking)
- ✓ If a functional unit for the instruction is free and no other active instruction has the same destination register (WAW), the scoreboard issues the instruction to the functional unit and updates its internal data structure.
- If a structural or a 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

A source operand is available if:

- no earlier issued active instruction will write it or
- A functional unit is writing its value in a register

When the source operands are available, the scoreboard tells the functional unit to proceed to read the operands from the registers and begin execution.

RAW hazards are resolved dynamically in this step, and instructions may be sent into execution out of order.

No forwarding of data in this model

### 3. Execution

Operate on operands
The functional unit begins execution upon receiving operands. When the result is ready, it notifies the scoreboard that it has completed execution.

### FUs are characterized by:

- latency (the effective time used to complete one operation)
- Initiation interval (the number of cycles that must elapse between issuing two operations to the same functional unit).

# 4. Write result Finish execution

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.

Assume we can overlap issue and write

```
DIVD F0, F2, F4
```

```
DIVD FO, F2, F4
```

```
DIVD F0, F2, F4

ADDD F6, F0, F8 war

SUBD F8, F8, F14

MULD F6, F10, F8
```

### The scoreboard would stall:

SUBD in the WB stage, waiting that ADDD reads F0 and F8 and

### The scoreboard would stall:

- SUBD in the WB stage, waiting that ADDD reads F0 and F8 and
- MULD in the issue stage until ADDD writes F6.

The scoreboard would stall:

- SUBD in the WB stage, waiting that ADDD reads F0 and F8 and
- MULD in the issue stage until ADDD writes F6.

Can be solved through register renaming

# Scoreboard Implications

- Solution for WAW:
  - Detect hazard and stall issue of new instruction until the other instruction completes
- No register renaming
- Need to have multiple instructions in execution phase → Multiple execution units or pipelined execution units
- Scoreboard keeps track of dependences and state of operations

# Scoreboard Implications

- Solution for WAW:
  - Detect hazard and stall issue of new instruction until the other instruction completes
- No register renaming
- Need to have multiple instructions in execution phase → Multiple execution units or pipelined execution units
- Scoreboard keeps track of dependences and state of operations

### Scoreboard structure: three parts

### 1. Instruction status

### 2. Functional Unit status

Indicates the state of the functional unit (FU):

Busy – Indicates whether the unit is busy or not

Op - The operation to perform in the unit (+,-, etc.)

Fi - Destination register

Fj, Fk – Source register numbers

Qj, Qk – Functional units producing source registers

Rj, Rk – Flags indicating when Fj, Fk are ready

### 3. Register result status

Indicates which functional unit will write each register. Blank if no pending instructions will write that register.

### Detailed Scoreboard Pipeline Control

Instruction status	Wait until	Bookkeeping
Issue	Not busy (FU) and not result(D)	Busy(FU) $\leftarrow$ yes; Op(FU) $\leftarrow$ op; Fi(FU) $\leftarrow$ `D'; Fj(FU) $\leftarrow$ `S1'; Fk(FU) $\leftarrow$ `S2'; Qj $\leftarrow$ Result( 'S1'); Qk $\leftarrow$ Result( `S2'); Rj $\leftarrow$ not Qj; Rk $\leftarrow$ not Qk; Result( 'D') $\leftarrow$ 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))	$\forall$ f(if Qj(f)=FU then Rj(f) $\leftarrow$ Yes); $\forall$ f(if Qk(f)=FU then Rk(f) $\leftarrow$ Yes); Result(Fi(FU)) $\leftarrow$ 0; Busy(FU) $\leftarrow$ No

# Scoreboard Example

dest

```
Instruction status:
                               Read Exec Write
                        Issue Oper Comp Result
   Instruction
   LD
            F6
                34 + R2
   LD
            F2
                45 + R3
   MULTD
                F2
                   F4
            F0
   SUBD
            F8
                    F2
   DIVD
            F10
                F0
                    F6
   ADDD
            F6
                F8
                    F2
```

#### Functional unit status:

Time Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Řj	Rk
Integer	No								
Mult1	No								
Mult2	No								
Add	No								
Divide	No								

*S*2

FU

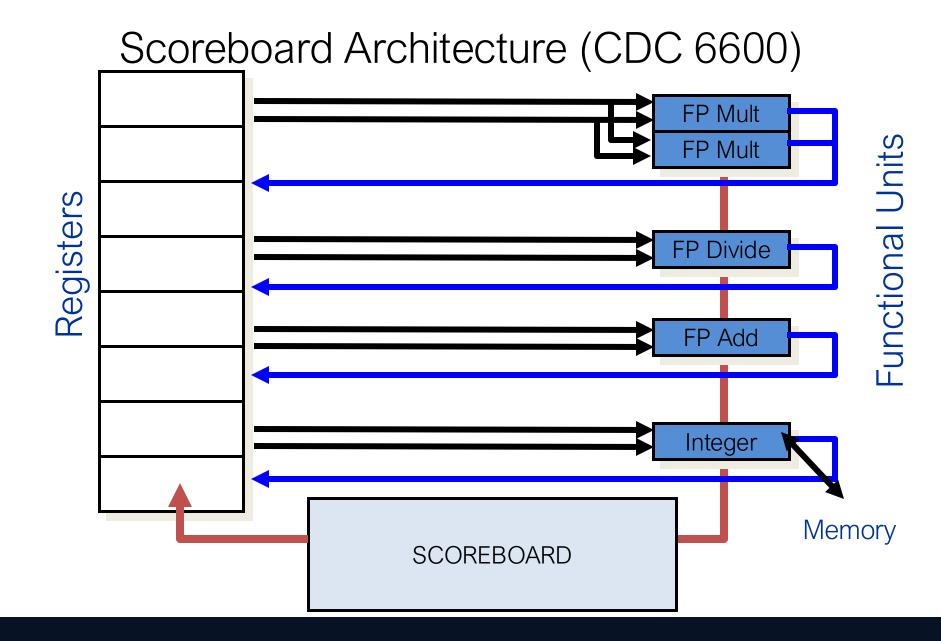
FU

Fi?

Fk?

SI

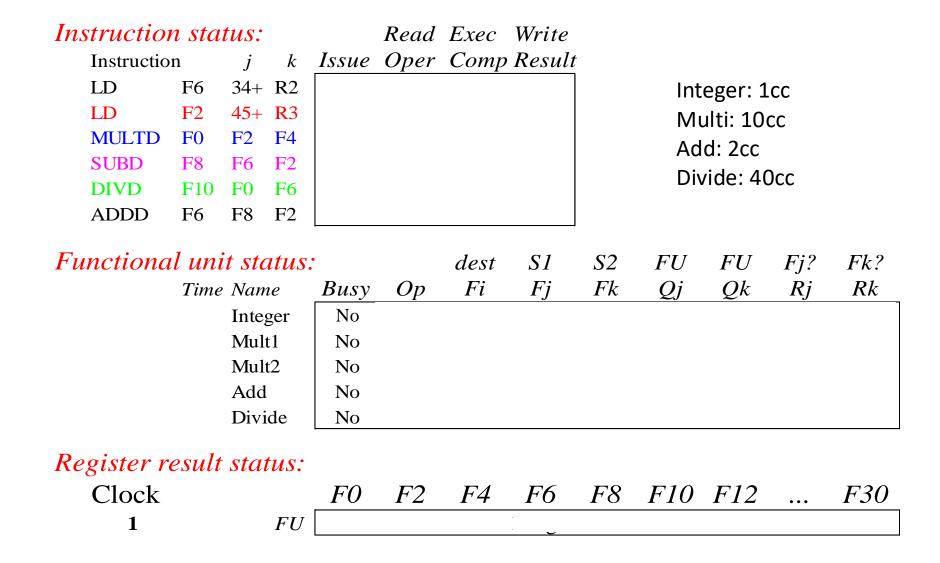
### Register result status:



### **Execution Process**

- Issue
  - Functional unit is free (structural)
  - Active instructions do not have same Rd (WAW)
- Read Operands
  - Checks availability of source operands
  - Resolves RAW hazards dynamically (out-of-order execution)
- Execution
  - Functional unit begins execution when operands arrive
  - Notifies the scoreboard when it has completed execution
- Write result
  - Scoreboard checks WAR hazards
  - Stalls the completing instruction if necessary

### Scoreboard Example: Bootstrap



Fk?

Fi?

# Scoreboard Example: Cycle 1

Instructio	n sta	tus:			Read	Exec	Write
Instruction	on	j	k	Issue	Oper	Comp	Result
LD	F6	34+	R2	1			
LD	F2	45+	R3				
MULTD	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
$\Delta$ DDD	F6	FΩ	F2				

Integer: 1cc Multi: 10cc Add: 2cc

Divide: 40cc

FII

#### Functional unit status:

t tillit bittitib.			CCBI	<b>D</b> 1	~ -	1 0	1	<b>-</b> j ·	1
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								

SI

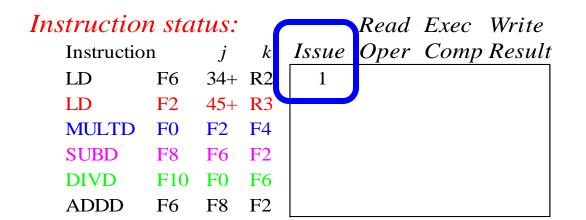
S2

FII

### Register result status:

Clock		F0	F2	<i>F4</i>	<i>F6</i>	F8	F10	<i>F12</i>	•••	F30
1	FU $igl[$				Integer					

dest



Integer: 1cc Multi: 10cc

Add: 2cc

Divide: 40cc

Functional unit status	•		dest	S1	<i>S2</i>	FU	FU	Fi?	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 1 

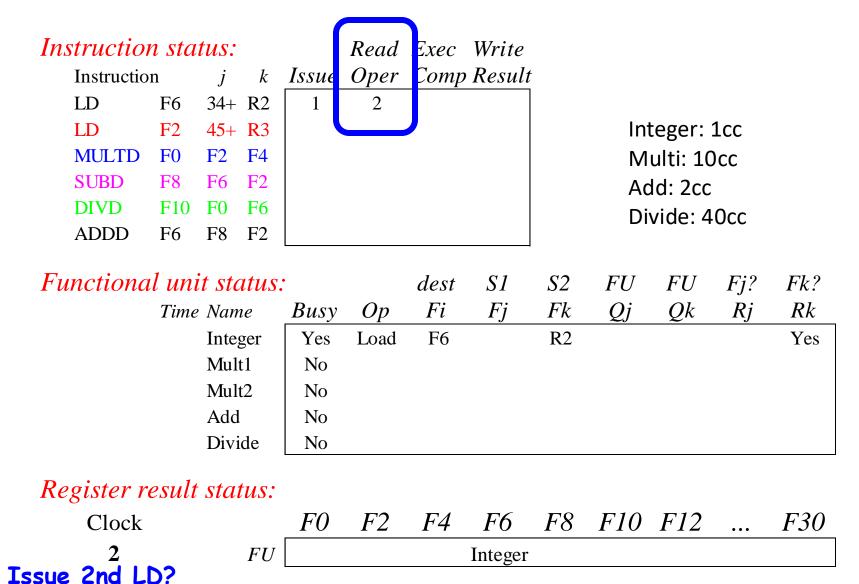
 F0
 F2
 F4
 F6
 F8
 F10
 F12
 ...
 F30

 FU
 Integer
 Integer

Instruction			Read	Exec	Write							
Instructio	n	j	k	Issue	Oper	Comp	Result					
LD	F6	34+	R2	1	2							
LD	F2	45+	R3						Int	teger:	1cc	
MULTD	F0	F2	F4						M	ulti: 10	)cc	
SUBD	F8	F6	F2						Ac	ld: 2cc		
DIVD	F10	F0	F6						Di	vide: 4	0cc	
ADDD	F6	F8	F2						٥.			
Functiona	it sta	atus.	•		dest	S1	<i>S</i> 2	FU	FU	Fj?	Fk?	
	Time	Nan	<i>ie</i>	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
		Inte	ger	Yes	Load	F6		R2				Yes
		Mul	t1	No								
		Mul	t2	No								
		Add		No								
		Divi	de	No								
Register re	esult	sta	tus:									
Clock				F0	<i>F</i> 2	<i>F4</i>	<i>F6</i>	F8	F10	<i>F12</i>	•••	F30

Integer

FU



Integer Pipeline Full - Cannot exec 2<sup>nd</sup> Load - Issue stalls

Instruction	n sta	tus:			Read	Exec	Write
Instructio	n	j	k	Issue	Oper	Comp	Result
LD	F6	34+	R2	1	2	3	
LD	F2	45+	R3				
MULTD	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Integer: 1cc Multi: 10cc Add: 2cc

Divide: 40cc

Fi?

Fk?

#### Functional unit status:

								J	
Time Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	Load	F6		R2				No
Mult1	No								
Mult2	No								
Add	No								
Divide	No								

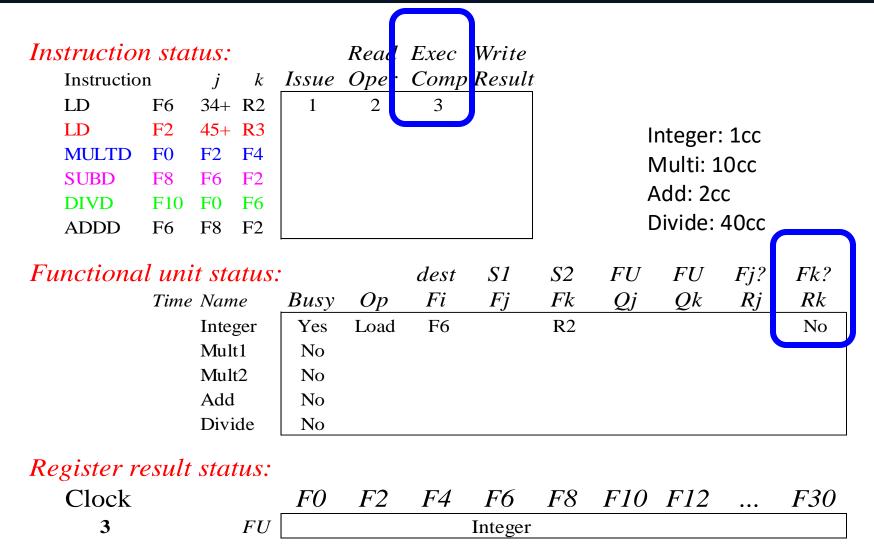
SI

*S*2

FU

### Register result status:

dest



**Issue MULT?** Issue stalls

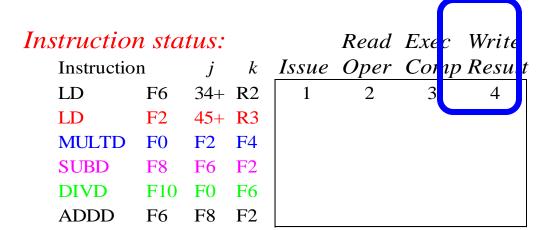
Load execution completes in one clock cycle

In	struction status:					Read	Exec	Write					
	Instructio	n	$\dot{j}$	k	Issue	Oper	Comp	Result					
	LD	F6	34+	R2	1	2	3	4					
	LD	F2	45+	R3						I	nteger:	1cc	
	<b>MULTD</b>	F0	F2	F4							∕lulti: 1		
	SUBD	F8	F6	F2						_			
	DIVD	F10	F0	F6							Add: 2c		
	ADDD	F6	F8	F2							Divide:	40cc	
Functional unit status:					•		dest	S1	<i>S</i> 2	FU	FU	Fj?	Fk?
		Time	Nan	ıe	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
			Inte	ger	No								
			Mul	t1	No								
			Mul	t2	No								
			Add	l	No								
			Divi	ide	No								
Register result status:													
	Clock				<i>F0</i>	<i>F</i> 2	<i>F4</i>	<i>F6</i>	F8	F10	<i>F12</i>	• • •	<i>F30</i>

Integer

FU

4



Integer: 1cc Multi: 10cc

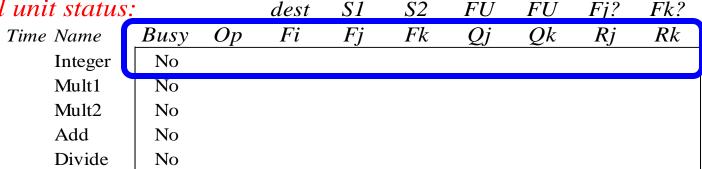
Add: 2cc

FU

Divide: 40cc

Fk?

#### Functional unit status:



SI

*S*2

FU

dest

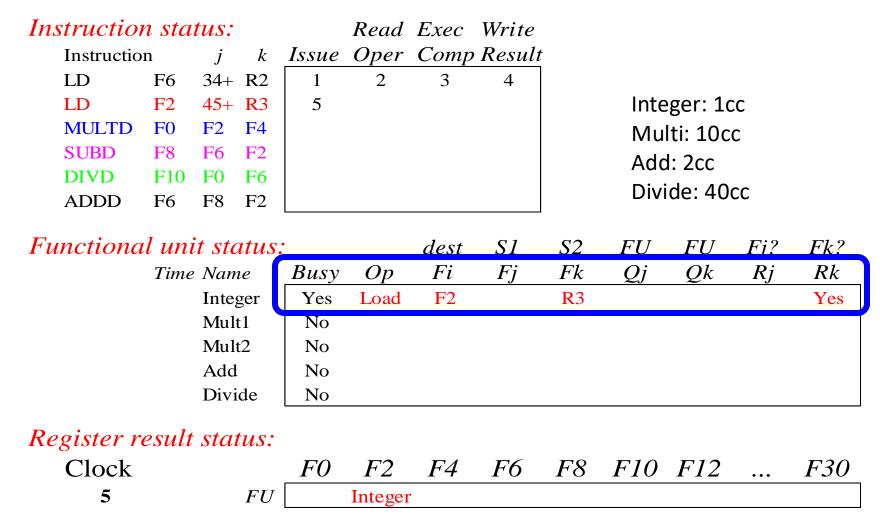
### Register result status:

Clock F0*F2* F4 *F*6 F8 F10 F12 F30 FU4 Integer

### Issue stalls

Write F6

Instruction		Read	Exec	Write								
Instruction		$\dot{j}$	$\boldsymbol{k}$	Issue	Oper	Comp	Result					
LD	F6	34+	R2	1	2	3	4					
LD	F2	45+	<b>R3</b>	5				Integer: 1cc				
MULTD	F0	F2	F4					Multi: 10cc				
SUBD	F8	F6	F2					Add: 2cc				
DIVD	F10	F0	<b>F6</b>									
ADDD	F6	F8	F2					Divide: 40cc				
Functional unit status:						dest	<i>S1</i>	<i>S</i> 2	FU	FU	Fj?	Fk?
	Time Name			Busy	Op	Fi	Fj	Fk	Qj	Qk	Řj	Rk
	Integer			Yes	Load	F2		<b>R</b> 3				Yes
	Mult1			No								
	Mult2		No									
	Add			No								
		Divi	de	No								
Register result status:												
Clock				FO	F2	F4	<i>F6</i>	F8	F10	<i>F12</i>	•••	F30
5			FU		Integer	•						



The 2<sup>nd</sup> load is issued

Fk?

# Scoreboard Example: Cycle 6

Instruction	n sta	tus:			Read	Exec	Write
Instructio	Instruction			Issue	Oper	Comp	Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	<b>R</b> 3	5	6		
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Integer: 1cc Multi: 10cc Add: 2cc

Divide: 40cc

FU

Fj?

FU

### Functional unit status:

Time Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Řj	Rk
Integer	Yes	Load	F2		R3				Yes
Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
Mult2	No								
Add	No								
Divide	No								

SI

*S*2

### Register result status:

dest

Instruction status:				Read	Exec	Write							
Instructio	n	j	$\boldsymbol{k}$	Issue	Oper	Comp	Result						
LD	F6	34+	R2	1	2	3	4						
LD	F2	45+	R3	5	6				Integ	er: 1cc			
MULTD	F0	F2	F4	6					•				
SUBD	F8	F6	F2						Multi				
DIVD	F10	F0	F6						Add: 2				
ADDD	F6	F8	F2						Divide	e: 40cc			
Functional unit status:  Time Name B			Busy	Ор	dest Fi	S1 Fj	S2 Fk	FU Qj	$FU \ Qk$	Fj? Rj	Fk? Rk		
		Inte	ger	Yes	Load	F2	, i	R3	0			Yes	
		Mul	t1	Yes	Mult	F0	F2	F4	Integer		No	Yes	
		Mul	t2	No									
		Add		No									
		Divi	de	No									
Register r	esult	sta	tus:										
Clock				F0	<i>F</i> 2	F4	<i>F6</i>	F8	F10	<i>F12</i>	•••	<i>F30</i>	

MULT is issued but has to wait for F2 from LOAD (RAW)

Mult1 Integer

6

II-2

F30

# Scoreboard Example: Cycle 7

1	nctri	uction	status:
	usiri	iciioni	siains.

Instructio	n	k	
LD	F6	34+	R2
LD	F2	45+	R3
MULTD	F0	F2	F4
SUBD	F8	F6	F2
DIVD	F10	F0	F6
ADDD	F6	F8	F2

Read Ex	cec Write
---------	-----------

Issue	Oper	Comp	Result
1	2	3	4
5	6	7	
6			
7			

Integer: 1cc

Multi: 10cc

Add: 2cc

Divide: 40cc

TII

### Functional unit status:

Time	Name
	Integer
	Mult1
	Mult2
	Add
	Divide

•			aest	SI	32	FU I	TU FJ	f(K)
	Busy	Op	Fi	Fj	Fk	Qj	Qk R	j $Rk$
	Yes	Load	F2		R3			No
	Yes	Mult	F0	F2	F4	Integer	N	o Yes
	No							
	Yes	Sub	F8	F6	F2	In	teger Ye	es No
	No							

### Register result status:

Clock

F0F2Mult1 Integer

F4

F6

F8 F10 F12

Add

Instruction	n sta	tus:		Read	Exec	Write	
Instruction	n	j	k	Issue	Oper	Comp	Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	<b>R</b> 3	5	6	7	
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2	7			
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Integer: 1cc

Multi: 10cc

Add: 2cc

FII

Divide: 40cc

FII

Fi?

Fk?

### Functional unit status:

e direct Secretis.			acsi	$\mathcal{D}_{\mathbf{I}}$	52	1 0	1 0	$\boldsymbol{I}$ $\boldsymbol{J}$ .	1 10.
Time Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	Load	F2		R3				No
Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
Mult2	No								
Add	Yes	Sub	F8	F6	F2		Integer	Yes	No
Divide	NO								

SI

*S*2

### Register result status:

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

dest

Now SUBD can be issued but has to wait for operands Read multiple operands?

	ทรา	ruci	tion	stati	110.
_	IUSU			Sicil	vis.

Instruction LD F6 34 + R2LD F2 45 + R3MULTD F0 F2 F4 **SUBD** F8 F6 F2 **DIVD** F10 F0 F6 F8 F2 ADDD F6

		Keaa	Exec	write
	Issue	Oper	Comp	Result
,	1	2	3	4
	5	6	7	
	6			
	7			
	8			

Doad Exac White

dost

51

Integer: 1cc

Multi: 10cc

Add: 2cc

FII

Divide: 40cc

FII

Fi?

Fk2

### Functional unit status:

Time	Name
	Integer
	Mult1
	Mult2
	Add
	Divide

•			uesi	$\mathcal{O}I$	52	$I^*U$	I	IJ.	I'A.
	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Yes	Load	F2		R3				No
	Yes	Mult	F0	F2	F4	Integer		No	Yes
	No								
	Yes	Sub	F8	F6	F2		Integer	Yes	No
	Yes	Div	F10	F0	F6	Mult1		No	Yes

52

### Register result status:

Clock

FU M

F0	F2	<i>F4</i>	F6	F8	F10
Tult1	Integer			Add	Divide

F10 F12 ... F30

Instruction	on	j	k	Issue	Oper	Comp	Result					
LD	F6	34+	R2	1	2	3	4					
LD	F2	45+	<b>R</b> 3	5	6	7				_		
MULTD	F0	F2	F4	6					Integ	er: 1cc		
SUBD	F8	F6	F2	7					Multi	: 10cc		
DIVD	F10	F0	F6	8					Add:	2cc		
ADDD	F6	F8	F2						Divid	e: 40cc		
Function	Functional unit status:				dest	S1	<i>S2</i>	FU	FU	Fj?	Fk?	
	Time	Nam	ie	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
		Integ	ger	Yes	Load	F2		R3				No
		Mul	t1	Yes	Mult	F0	F2	F4	Integer		No	Yes
		Mul	t2	No								
		Add		Yes	Sub	F8	F6	F2		Integer	Yes	No

Read Exec Write

Register result status:

Divide

Yes

Instruction status:

F10

F0

**F6** 

Mult1

No

Yes

DIVD is issued but there is another RAW hazard (F0) from MULTD then DIVD has to wait for F0

Div

In	struction	n sta	tus:			Read	Exec	Write
	Instruction	$\dot{j}$	k	Issue	Oper	Comp	Result	
	LD	F6	34+	R2	1	2	3	4
	LD	F2	45+	R3	5	6	7	8
	<b>MULTD</b>	F0	F2	F4	6			
	SUBD	F8	F6	F2	7			
	DIVD	F10	F0	F6	8			
	ADDD	F6	F8	F2				

Integer: 1cc Multi: 10cc

Add: 2cc

Divide: 40cc

Fj?

Rj

Yes

Yes

No

Fk?

Rk

Yes

Yes

Yes

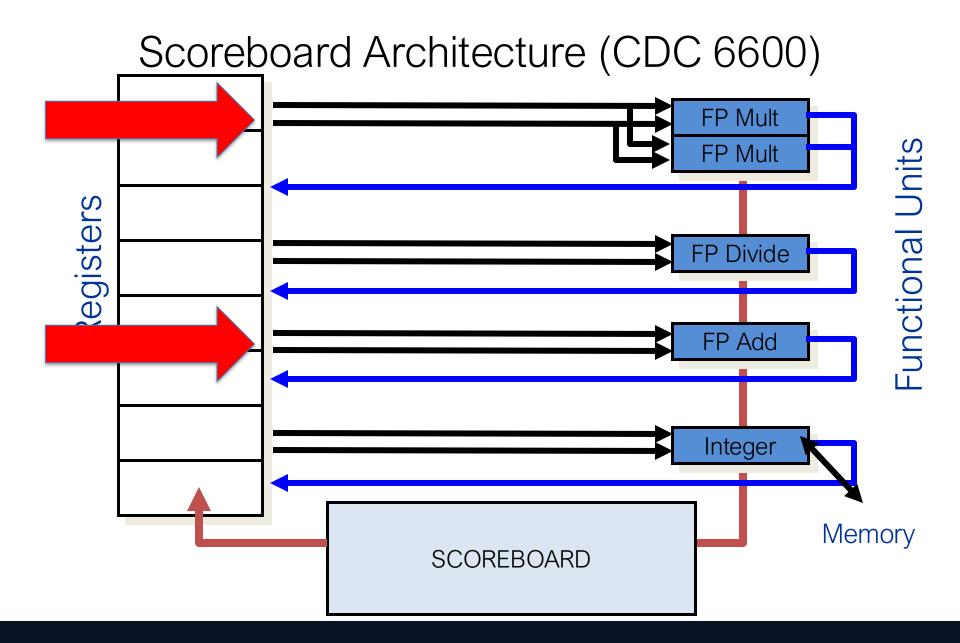
### Functional unit status:

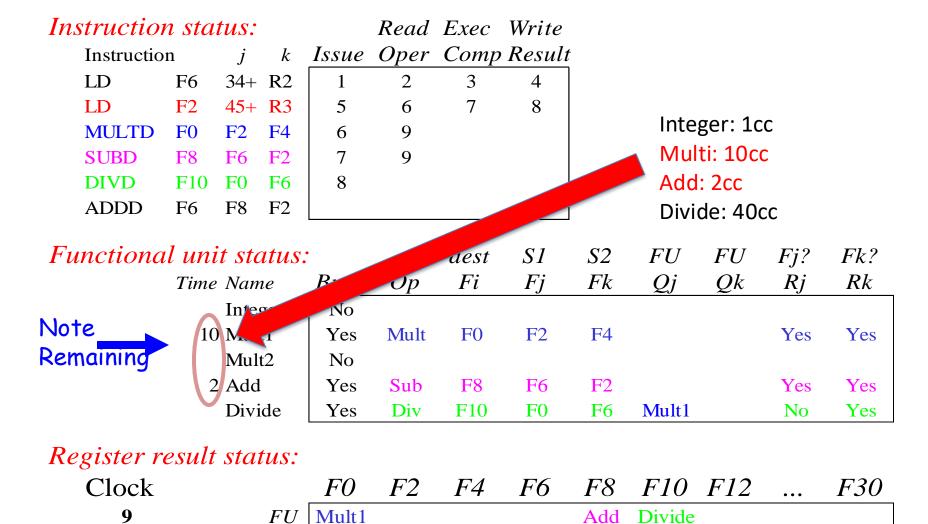
l unit status:	•		dest	SI	<i>S</i> 2	FU	FU
Time Name	Busy	Op	Fi	Fj	Fk	Qj	Qk
Integer	No						
Mult1	Yes	Mult	FO	F2	F4		
Mult2	No						
Add	Yes	Sub	F8	F6	F2		
Divide	Yes	Div	F10	FO	F6	Mult1	

### Register result status:

Clock F2F4 *F6* F8 F10 F12 F30 FO8 FUMult1 Add Divide

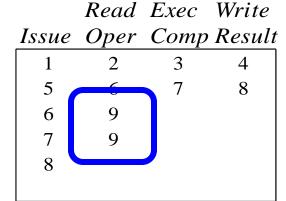
Load completes, and operands for MULT an SUBD are ready





	nst	ru	cti	io	n	st	at	us	7.
_				٠.	•	~ •			•

Instruction kLD F6 34 + R2LD F2 45 + R3**MULTD** F0 F2 F4 **SUBD** F8 F6 F2 **DIVD** F10 F<sub>0</sub> F6 F8 **ADDD** F6 F2



Integer: 1cc

Multi: 10cc

Add: 2cc

**LII** 

Divide: 40cc

CII

E;2

E1. 2

### Functional unit status:

Note
Remaining

Time Name
Integer
Mult1
Mult2
Add
Divide

•			aesi	$\mathcal{S}I$	32	$\Gamma U$	$\Gamma U$	$\Gamma J$ :	$\Gamma K$ ?	
	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk	
	No									
	Yes	Mult	F0	F2	F4			Yes	Yes	
	No									
	Yes	Sub	F8	F6	F2			Yes	Yes	
	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

dost

C1

Read operands for MULTD & SUBD

MULTD and SUBD are sent in execution in parallel

Issue ADDD? No for structural hazard on ADD Functional Unit

7	<b>r</b>	4	, ·		4	
/	nsi	tru	C11	OH	stat	<i>'SM</i>

Instruction LD F6 34 + R2LD F2 45 + R3**MULTD** F0 F2 F4 **SUBD** F8 F6 F2 **DIVD** F10 F0 F6 F8 F2 **ADDD** F6

	Read	Exec	Write
Issue	Oper	Comp	Result
1	2	3	4
5	6	7	8
6	9		
7	9		
8			

Integer: 1cc

Multi: 10cc

Add: 2cc

TII

Divide: 40cc

TII

E:2

L1-9

### Functional unit status:

Time	Name
	Integer
9	Mult1
	Mult2
1	Add
	Divide

•		aesi	$\mathcal{S}I$	32	FU	FU	$\Gamma J$ :	FK?
Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
No								
Yes	Mult	F0	F2	F4			No	No
No								
Yes	Sub	F8	F6	F2			No	No
Yes	Div	F10	F0	F6	Mult1		No	Yes

C1

### Register result status:

Clock 10

 $\mathbf{F}$ <sub>2</sub> 2

# Scoreboard Example: Cycle 11

7	7				•	4	4
	n	CT	rı ı	CT	100	stai	<i>†11 C</i> •
	. 1 U L	ועכ	ıvı	$\cup \iota \iota$	UUU	Siui	ws.

Instructio	n	j	$\boldsymbol{k}$
LD	F6	34+	R2
LD	F2	45+	R3
MULTD	F0	F2	F4
SUBD	F8	F6	F2
DIVD	F10	F0	F6
ADDD	F6	F8	F2

	Read	Exec	Write
Issue	Oper	Comp	Result
1	2	3	4
5	6	7	8
6	9		
7	9	11	
8			

Integer: 1cc Multi: 10cc

Add: 2cc

 $\mathbf{E}II$ 

Divide: 40cc

 $\mathbf{E}II$ 

### Functional unit status:

Time	Name
	Integer
8	Mult1
	Mult2
0	Add
	Divide

		aesi	$\mathcal{S}I$	32	FU	$\Gamma U$	$\Gamma J$ :	$\Gamma K$ :
Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
No								
Yes	Mult	F0	F2	F4			No	No
No								
Yes	Sub	F8	F6	F2			No	No
Yes	Div	F10	F0	F6	Mult1		No	Yes

 $C_{2}$ 

C1

### Register result status:

Clock 11

	F0	<i>F</i> 2	F4	<i>F6</i>	F8	F10	<i>F12</i>	•••	F30
FU	Mult1				Add	Divide			

SUBD ends

Tristructio	Tristruction status.				Reau	LACC	VVI LLE					
Instruction	on	j	$\boldsymbol{k}$	Issue	Oper	Comp	Result					
LD	F6	34+	R2	1	2	3	4					
LD	F2	45+	R3	5	6	7	8		Intege	er: 1cc		
MULTD	F0	F2	F4	6	9				Multi:	10cc		
SUBD	F8	F6	F2	7	9	11	12		Add: 2	) (		
DIVD	F10	F0	F6	8						:: 40cc		
ADDD	F6	F8	F2						Divide	. 4000		
ADDD	10	10										
Function						dest	S1	<i>S2</i>	FU	FU	Fj?	Fk?
	al uni		atus:	Busy	Ор	dest Fi	S1 Fj	S2 Fk	FU Qj	$FU \ Qk$	Fj? Rj	Fk? Rk
	al uni	it stc	atus: 1e		Ор						v	
	al uni Time	it ste Nam	atus: 1e ger	Busy	<i>Op</i> Mult						v	
	al uni Time	it sto Nam Integ	atus: 1e ger t1	Busy No	*	Fi	Fj	Fk			Rj	Rk
	al uni Time	it Sto Nam Integ Mult	atus: ne ger t1 t2	Busy No Yes	*	Fi	Fj	Fk			Rj	Rk

Read Exec Write

### Register result status:

Instruction status:

Clock	F0	F2	F4	<i>F6</i>	F8	F10 F12	•••	<i>F30</i>
12	FU Mult1					Divide		

Read operands for DIVD?

nstruction	r sta	tus:			Read	Exec	Write
Instructio	j	k	Issue	Oper	Comp	Result	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	<b>R3</b>	5	6	7	8
<b>MULTD</b>	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	FO	F6	8			
ADDD	F6	F8	F2	13			

Integer: 1cc Multi: 10cc Add: 2cc

Divide: 40cc

 $\mathbf{L}III$ 

Fi2

F1-2

 $\mathbf{\Gamma}II$ 

### Functional unit status:

Time Name	Busy	$O_I$
Integer	No	
6 Mult1	Yes	Mu
Mult2	No	
Add	Yes	Ad
Divide	Yes	Di

<b>5</b> .			aesi	$\mathcal{S}I$	32	$\Gamma U$	F U	$\mathbf{\Gamma} f$ :	$\mathbf{\Gamma} \mathbf{K}$ :
	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	No								
	Yes	Mult	F0	F2	F4			No	No
	No								
			F6					Yes	Yes
	Yes	Div	F10	FO	F6	Mult1		No	Yes

C1

### Register result status:

Clock 13

SUBD writes results in CC12 and ADDD can be issued in CC13

Ins	struction	ı sta	tus:			Read	Exec	Write
	Instruction	1	$\dot{J}$	k	Issue	Oper	Comp	Result
	LD	F6	34+	R2	1	2	3	4
	LD	F2	45+	R3	5	6	7	8
	MULTD	F0	F2	F4	6	9		
	SUBD	F8	F6	F2	7	9	11	12
	DIVD	F10	F0	F6	8			
	ADDD	F6	F8	F2	13	14		

Integer: 1cc Multi: 10cc Add: 2cc

Divide: 40cc

FU

Fi?

Fk?

FU

### Functional unit status:

Time Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No								
5 Mult1	Yes	Mult	F0	F2	F4			No	No
Mult2	No								
2 Add	Yes	Add	F6	F8	F2			Yes	Yes
Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

SI

S2

### Register result status:

Clock		FO	F2	F4	<i>F6</i>	F8	F10 F12	•••	F30
14	FU [	Mult1			Add		Divide		

dest

ADDD reads operands

(out-of-order read operands: ADDD reads operands before DIVD)

- 7	<b>T</b>	4	, •	4 4
- 1	nc	TV1	เกรากท	status:
		uu	$\iota \cup \iota \iota \cup \iota \iota$	siaius.

Instruction	j	k	
LD	F6	34+	R2
LD	F2	45+	<b>R3</b>
MULTD	F0	F2	F4
SUBD	F8	F6	F2
DIVD	F10	F0	F6
ADDD	F6	F8	F2

Read	Exec	Write
------	------	-------

Issue	Oper	Comp	Result
1	2	3	4
5	6	7	8
6	9		
7	9	11	12
8			
13	14		

Integer: 1cc

Multi: 10cc

Add: 2cc

Divide: 40cc

### Functional unit status:

1 iiie	rume
	Integer
4	Mult1
	Mult2
1	Add

Divide

Time Name

•	•		dest	SI	<i>S2</i>	FU	FU	Fj?	Fk?	
	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk	
	No									
	Yes	Mult	F0	F2	F4			No	No	
	No									
	Yes	Add	F6	F8	F2			No	No	
	Yes	Div	F10	FO	F6	Mult1		No	Yes	

### Register result status:

Divide

*F30* 

	nstri	iction	ı status:	•
-	IUSUIU	$\iota \cup \iota \iota \cup \iota \iota$	i bicilib.	

truction	n sta	tus:			Read	Exec	Write
Instruction	n	j	$\boldsymbol{k}$	Issue	Oper	Comp	Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	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	

Integer: 1cc

Multi: 10cc

Add: 2cc

Divide: 40cc

### Functional unit status.

Time	Name
	Integer
3	Mult1
	Mult2
0	Add
	Divide

FU

•	•		dest	SI	<i>S</i> 2	FU	FU	Fj?	Fk?	
	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk	
	No									
	Yes	Mult	F0	F2	F4			No	No	
	No									
	Yes	Add	F6	F8	F2			No	No	
	Yes	Div	F10	F0	F6	Mult1		No	Yes	

### Register result status:

Clock 16

F10 F12 *F30* F0F2*F4 F6* F8 Add Mult1 Divide

- 7	<b>T</b>	4	, •	4 4
- 1	nc	TV1	เกรากท	status:
		u	$\iota \cup \iota \iota \cup \iota \iota$	siaius.

Instruction k LD F6 34 + R2LD F2 45 + R3**MULTD** F0 F2. F4 F8 F2 SUBD **DIVD** F10 F0 F6

Exec	Write
	Exec

Issue	Oper	Comp	Result
1	2	3	4
5	6	7	8
6	9		
7	9	11	12
8			
13	14	16	

Integer: 1cc

Multi: 10cc

Add: 2cc

Divide: 40cc

TITI

T: 2

T71-9

### Functional unit status:

F6

Time Name
Integer
2 Mult1
Mult2
Add

Divide

F8

F2

		aest	SI	32	FU	FU	FJ?	FK?	
Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk	
No									
Yes	Mult	F0	F2	F4			No	No	
No									
Yes	Add	F6	F8	F2			No	No	
Yes	Div	F10	FO	F6	Mult1		No	Yes	

### Register result status:

Clock

**ADDD** 

F0 F2 Mult1

*F4 F6* Add

F8 F10

F10 F12

... F30

Why not write result of ADD???

Instruction status:				Read	Exec	Write						
Instructio	n	$\dot{J}$	$\boldsymbol{k}$	Issue	Oper	Comp	Result					
LD	F6	34+	R2	1	2	3	4					
LD	F2	45+	R3	5	6	7	8					
MULTD	F0	F2	F4	6	9							
SUBD	F8	F6	T-2	7	9	11	12	M	/AD	Haz	andl	
DIVD	F10	F0	F6	8				V	VAK	Muzi	uru!	
ADDD	F6	F8	F2	13	14	16						
Functiona	l un	it sto	atus:	•		dest	SI	<i>S</i> 2	FU	FU	Fj?	Fk?
	Time	e Nam	<i>ie</i>	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
		Integ	ger	No								
	4	2 Mult	t1	Yes	Mult	F0	F2	F4			No	No
		Mult	t2	No								
		Add		Yes	Add	F6	F8	F2			No	No
		Divi	de	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

Why not write result of ADD???

DIVD must first read F6 but cannot read until MULTD writes F0

7	T , ,	•	4 4
	nstruct	$1 \cap n$	Ctatuc.
	<i>listi u</i> ct	$\iota \mathcal{O} \iota \iota$	siains.

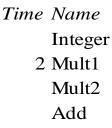
Instructio	j	$\boldsymbol{k}$	
LD	F6	34+	R2
LD	F2	45+	R3
MULTD	F0	F2	F4
SUBD	F8	F6	F2
DIVD	F10	F0	F6
ADDD	F6	F8	F2

Read Ex	xec Write
---------	-----------

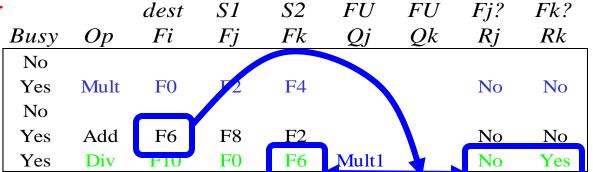
Issue	Oper	Comp	Result
1	2	3	4
5	6	7	8
6	9		
7	9	11	12
8			
13	14	16	

### WAR Hazard!

### Functional unit status:



Divide



### Register result status:

Clock 17

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

Why not write result of ADD???

DIVD must first read F6 but cannot read until MULTD writes F0

- 7	<b>T</b>	, •	4 4
_	nctr	uction	status:
	IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	ucucon	siains.

Instructio	n	$\dot{J}$	k	Iss		
LD	F6	34+	R2			
LD	F2	45+	R3	:		
MULTD	F0	F2	F4			
SUBD	F8	F6	F2	,		
DIVD	F10	F0	<b>F6</b>			
ADDD	F6	F8	F2	1		

Read	Exec	Wri	ite
_		_	_

Issue	Oper	Comp	Result
1	2	3	4
5	6	7	8
6	9		
7	9	11	12
8			
13	14	16	

Integer: 1cc

Multi: 10cc

Add: 2cc

Divide: 40cc

### Functional unit status:

Гіте	Name
	Integer
1	Mult1
	Mult2
	Add
	Divide

		dest	SI	<i>S2</i>	FU	FU	FJ!	FK!
Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
No								
Yes	Mult	F0	F2	F4			No	No
No								
Yes	Add	F6	F8	F2			No	No
Yes	Div	F10	FO	F6	Mult1		No	Ves

### Register result status:

Clock 18

FU

F0F2 *F4* 

F8 F10 F12

*F30* 

Mult1 Add Divide

*F6* 

Read Exec Write

- 1		4	•	4	4
	1/1 CT	アリノクサ	$1 \cap n$	CTA	#11 C •
		ruct	LUIL	$\Delta L U L$	
_				~	

	v Sici	vvo.			recuu	DACC	******
Instruction	n	$\dot{j}$	$\boldsymbol{k}$	Issue	Oper	Comp	Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	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	

Integer: 1cc

Multi: 10cc

Add: 2cc

Divide: 40cc

### Functional unit status:

Time	Name
	Integer
0	Mult1
	Mult2
	Add
	Divide

•		dest	51	<i>S2</i>	FU	FU	Fj?	FK?	
Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk	
No									
Yes	Mult	F0	F2	F4			No	No	
No									
Yes	Add	F6	F8	F2			No	No	
Yes	Div	F10	F0	F6	Mult1		No	Yes	

### Register result status:

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

Instructi	on sta	tus:			Read	Exec	Write
Instruct	ion	$\dot{j}$	k	Issue	Oper	Comp	Result
LD	F6	34+	R2	1	2	3	4
T.D.		4 ~	<b>D</b> 0		_	_	0

 LD
 F2
 45+
 R3

 MULTD
 F0
 F2
 F4

 SUBD
 F8
 F6
 F2

 DIVD
 F10
 F0
 F6

ADDD F6 F8 F2

	Keaa	Exec	write
Issue	Oper	Comp	Result
1	2	3	4
5	6	7	8
6	9	19	20
7	9	11	12
8			
13	14	16	

Integer: 1cc

Multi: 10cc

Add: 2cc

Divide: 40cc

FII

Fi?

Fk?

### Functional unit status:

Time Name

Integer

Mult1

Mult2

Add

Divide

		acsi	$\mathcal{D}I$	02	10	10	IJ.	1 1.
Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
No								
No								
No								
Yes	Add	F6	F8	F2			No	No
Yes	Div	F10	F0	F6			Yes	Yes

*S*2

FII

### Register result status:

Clock 20

FU

*F0 F2* 

*F4* 

dest

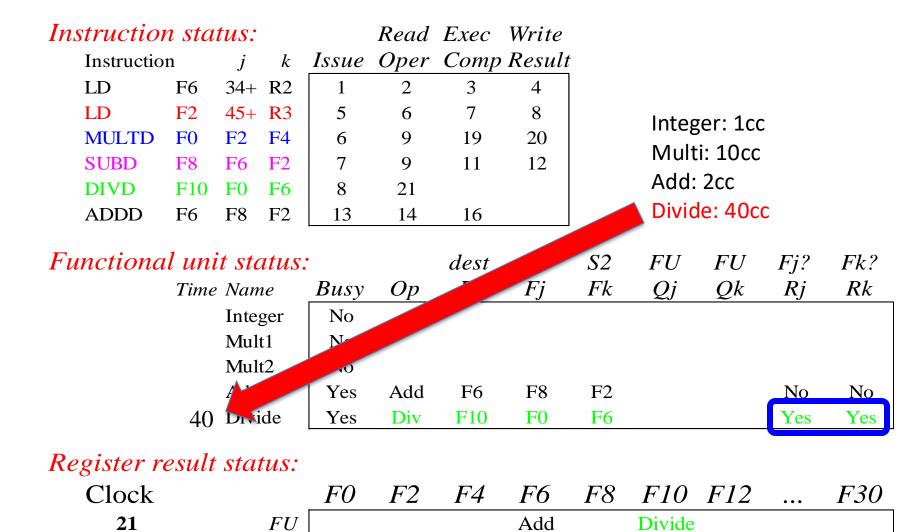
 $\frac{F6}{\text{Add}}$ 

51

F8 F10 Divide

*F10 F12* 

... F30



WAR Hazard is now gone...

Instruction	n sta	tus:			Read	Exec	Write					
Instructio	n	$\dot{j}$	k	Issue	Oper	Comp	Result					
LD	F6	34+	R2	1	2	3	4					
LD	F2	45+	<b>R</b> 3	5	6	7	8		Integ	ger: 1cc		
MULTD	F0	F2	F4	6	9	19	20		Mult	i: 10cc		
SUBD	F8	F6	F2	7	9	11	12		Add:			
DIVD	F10	F0	<b>F6</b>	8	21						_	
ADDD	F6	F8	F2	13	14	16	22		DIVIC	de: 40c	C	
Functiona	ıl uni	it sto	atus:	•		dest	S1	<i>S2</i>	FU	FU	Fj?	Fk?
	Time	Man	10	Busy	Op	Fi	Fj	Fk	$O_i$	Qk	Řj	Rk
	1 ime	r warr	ie	Dusy	Op	$I^{+}\iota$	I'J	I'K	Qj	$\mathcal{L}^{\kappa}$	$\boldsymbol{n}_{j}$	IΛK
	1 ime	Inte		No	<i>Op</i>	<u> </u>	<u>I'J</u>	I'K	<u> </u>	<u>Q</u> K	NJ	IK
	Time		ger		<u>Op</u>	1.1	TJ	TK	<u> </u>	<u>Q</u> r.	<u> </u>	NK
	1 ime	Inte	ger t1	No	<u> </u>	1.1	<u> </u>	I'K	<u> </u>	<u> </u>	<u> N</u>	- KK
	1 tme	Integ Mul	ger t1 t2	No No	Ор	11	<u> </u>	1'K	<u> </u>	<u> </u>	- NJ	- KK
		Integ Mul Mul	ger t1 t2	No No No	Div	F10	F0	F6	<u> </u>	<u>Q</u> r	No	No
Register r	39	Integ Mul Mul Add Divi	ger t1 t2 de	No No No No	-				<u> </u>	<u>Q</u> r		
<i>Register r</i> Clock	39	Integ Mul Mul Add Divi	ger t1 t2 de	No No No No	-				F10	F12		

Now DIVD has read its operands, ADDD can write the result in F6

# Faster than light computation (skip a couple of cycles)



Instruction status:					Read	Exec	Write					
Instructio	n	$\dot{j}$	k	Issue	Oper	Comp	Result	_				
LD	F6	34+	R2	1	2	3	4					
LD	F2	45+	R3	5	6	7	8		Int	ogor: 1	CC	
MULTD	F0	F2	F4	6	9	19	20			eger: 1		
SUBD	F8	F6	F2	7	9	_11	12			ılti: 10c	CC	
DIVD	F10	F0	<b>F6</b>	8	21	61			Add	d: 2cc		
ADDD	F6	F8	F2	13	14	16	22		Div	ide: 40	cc	
Functional unit status:				•		dest	<i>S1</i>	<i>S2</i>	FU	FU	Fj?	Fk?
	Time Name				Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
		Integ	ger	No								
		Mul	t1	No								
		Mul	t2	No								
	Add											
	(	) Divi	de	Yes	Div	F10	F0	F6			No	No
	_											
Register r	esult	tstai	tus:									
Clock				_F0	<i>F</i> 2	<i>F4</i>	<i>F6</i>	F8	F10	<i>F12</i>	•••	F30
61			FU						Divide			

Instruction		Read	Exec	Write									
Instructio	n	j	k	Issue	Oper	Comp	Result	t					
LD	F6	34+	R2	1	2	3	4						
LD	F2	45+	R3	5	6	7	8	Integer: 1cc Multi: 10cc Add: 2cc					
MULTD	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	Divide: 40cc					
Functiona	al un	it sta	atus	:		dest	S1	<i>S</i> 2	FU	FU	Fj?	Fk?	
	Time	e Nan	1e	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk	
		Inte	ger	No									
		Mul	t1	No									
		Mul	t2	No									
		Add	[	No									
		Divi	ide	No									
Register r	esuli	t sta	tus:										
Clock	F0	F2	<i>F4</i>	<i>F6</i>	F8	<i>F10</i>	<i>F12</i>	•••	<i>F30</i>				

DIVD writes in F10

FU

**62** 

Instruction	tus:			Read	Exec	Write						
Instructio	n	j	k	Issue	Oper	Comp	Result					
LD	F6	34+	R2	1	2	3	4					
LD	F2	45+	R3	5	6	7	8			4		
MULTD	F0	F2	F4	6	9	19	20		Integ	ger: 1cc		
SUBD	F8	F6	F2	7	9	11	12		Mult	i: 10cc		
DIVD	F10	F0	F6	8	21	61	62		Add:	2cc		
ADDD	F6	F8	F2	13	14	16	22		Divid	le: 40c	С	
Functiona		0	dest	S1	S2	FU	FU	Fj?	Fk?			
	Time	? Nan	<i>ie</i>	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
		Integ	_	No								
		Mul	t1	No								
		Mul	t2	No								
	Add											
		Divi	de	No								
Register r	esult	stai	tus:									
Clock				F0	<i>F</i> 2	F4	<i>F6</i>	F8	F10	<i>F12</i>	• • •	F30

**62** 

FU

Instructio	on sta	itus:			Read	Exec	Write					
Instructi	on	$\dot{j}$	k	Issue	Oper	Comp	Result					
LD	F6	34+	R2	1	2	3	4					
LD	F2	45+	R3	5	6	7	8					
MULTD	F0	F2	F4	6	9	19	20		Integ	ger: 1cc	,	
SUBD	F8	F6	F2	7	9	11	12		Mult	i: 10cc		
DIVD	F10	F0	<b>F6</b>	8	21	61	62		Add:	2cc		
ADDD	F6	F8	F2	13	14	16	22		Divid	le: 40c	C	
	_											
Functional unit status:						dest	<i>S1</i>	<i>S</i> 2	FU	FU	Fj?	Fk?
	Time	e Nan	ne	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
		Inte	ger	No								
		Mul	t1	No								
		Mul	t2	No								
		Add	l	No								
		Divi	ide	No								
<b>.</b>	1											
Register	resuli	t sta	tus:									
Clock	<i>F0</i>	<i>F</i> 2	<i>F4</i>	<i>F6</i>	F8	F10	<i>F12</i>	• • •	F30			

**62** 

FU

Instruction	n sta	tus:			Read	Exec	Write					
Instructio	n	j	$\boldsymbol{k}$	Issue	Oper	Comp	Result	<u>L</u>				
LD	F6	34+	R2	1	2	3	4					
LD	F2	45+	R3	5	6	7	8			4		
MULTD	F0	F2	F4	6	9	19	20		Integ	ger: 1cc		
SUBD	F8	F6	F2	7	9	11	12		Mult	i: 10cc		
DIVD	F10	F0	F6	8	21	61	62		Add:	2cc		
ADDD	F6	F8	F2	13	14	16	22		Divid	le: 40c		
Functiona	atus.			dest	S1	<i>S</i> 2	FU	FU	Fj?	Fk?		
	Time Name Bus					Fi	Fj	Fk	Qj	Qk	Rj	Rk
		Inte	ger	No								
		Mul	t1	No								
		Mul	t2	No								
		Add		No								
		Divi	ide	No								
Register r	esuli	t sta	tus:									
Clock				F0	<i>F</i> 2	<i>F4</i>	F6	F8	<i>F10</i>	<i>F12</i>	•••	F30

**62** 

FU

Instruction	Read I	Exec	Write											
Instructio	n	$\dot{J}$	k	Issue	Oper (	Comp	Resul	$\frac{d}{dt}$						
LD	F6	34+	R2	1	2	3	4							
LD	F2	45+	R3	5	6	7	8			4				
MULTD	F0	F2	F4	6	9	19	20		Integ	ger: 1cc				
SUBD	F8	F6	F2	7	9	11	12		Multi: 10cc					
DIVD	F10	FO	F6	8	21	61	62		Add: 2cc					
ADDD	F6	F8	F2	13	14	16	22	Divide: 40cc						
			•					_						
Functiona	l un	it sto	atus:			dest	<i>S1</i>	<i>S</i> 2	FU	FU	Fj?	Fk?		
TO THE FR Qj Qk Rj Rk										Rk				
		Mul		No		3	} [5]		BU					
Regis C		S	FU [								•••	F30		

## CDC 6600 Scoreboard

- Key idea of Scoreboard: Allow instructions behind stall to proceed (Decode ⇒ Issue Instruction & Read Operands)
- Speedup of 2.5 w.r.t. no dynamic scheduling
- Speedup 1.7 by reorganizing instructions from compiler
- BUT slow memory (no cache) limits benefit
- Limitations of 6600 scoreboard:
  - No forwarding hardware
  - Limited to instructions in basic block (small window)
  - Small number of functional units (structural hazards), especially integer/load store units
  - Do not issue on structural hazards
  - Wait for WAR hazards
  - Prevent WAW hazards

# Advanced Computer Architectures

(High Performance Processors and Systems)

# Dynamic Scheduling: Scoreboard

Politecnico di Milano v1