Dynamic Scheduling

- Hardware rearranges instruction execution to reduce stalls of dependent instruction (increases ILP) - permit out-of-order execution.
  - Handles dependences not known at compile-time
  - Allows code compiled for one pipeline to run efficiently on another pipeline
- Can't eliminate true dependences - try to work around them and avoid stalls

Concepts of Dynamic Scheduling

- Consider DLX with in-order issue:

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

Concepts of Dynamic Scheduling

- Consider DLX with in-order issue:

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

True dependence on F0

- SUBD stalls yet isn't dependent on ADDD or DIVD

Concepts of Dynamic Scheduling

- Consider DLX with in-order issue:

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

True dependence on F0

- SUBD stalls yet isn't dependent on ADDD or DIVD
- Suppose we let SUBD "move around" the stall and execute out of order?

Out-of-Order Execution

- In-order, single instruction issue
- Instructions begin execution as soon as their operands and FUs are available
- Exceptions will be a problem....later....
- Instruction issue now has independent checks
  - Check of structural hazards (FU availability)
  - Data hazards (operand availability)
- Split ID (decode) stage into
  - Issue - Decode instruction, check for structural hazards
  - Read Operands - Wait until no data hazards, then read operands

Issue and Read Operands

- In-order issue: Instructions pass through the Issue stage in order to resolve structural hazards
- Bypassing: Instructions may bypass one another in the Read Operand stage, potentially entering the execution pipeline out of order
- How do we check FU and operand availability?
  - *Scoreboarding* - Introduced in the CDC 6600 (circa 1963!!!)

New Hazards Possible

- Out-of-order execution leads to WAR hazards

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

New Hazards Possible

- Out-of-order execution leads to WAR hazards

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

- Of course, WAW hazards are now also possible

Scoreboarding

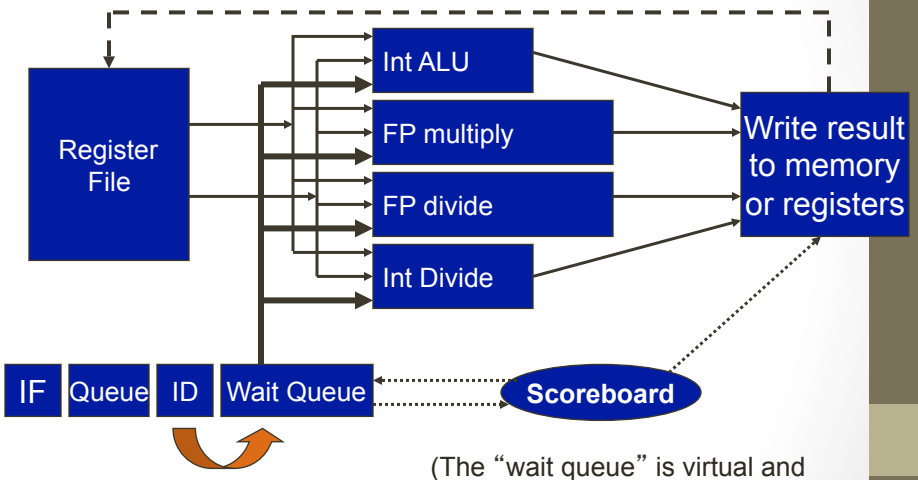
- Keep track of instructions, functional units, and registers to handle hazards
- Goal: CPI=1
  - "Pull out" independent instructions later in the "issue window".
- Wait queue after Issue to hold stalled instructions waiting for operands (it may not really exist)
- Multiple or pipelined functional units (recall: during issue, we stall on a structural hazard)

What does the Scoreboard do???

- Record data dependencies of instructions
- Determines when an instruction can read its operands and begin execution
- Can't execute? Monitor state of operands to decide when instruction can execute
- Determines when an instruction can write its result

⇒ *Scoreboard does all hazard detection and resolution*

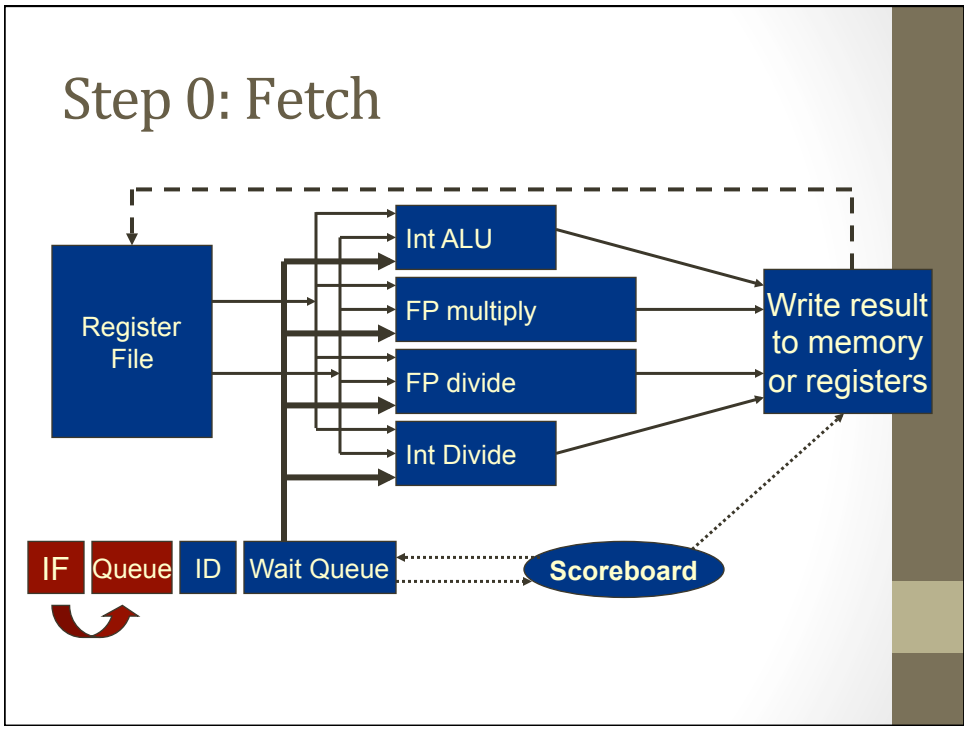
Scoreboard Architecture

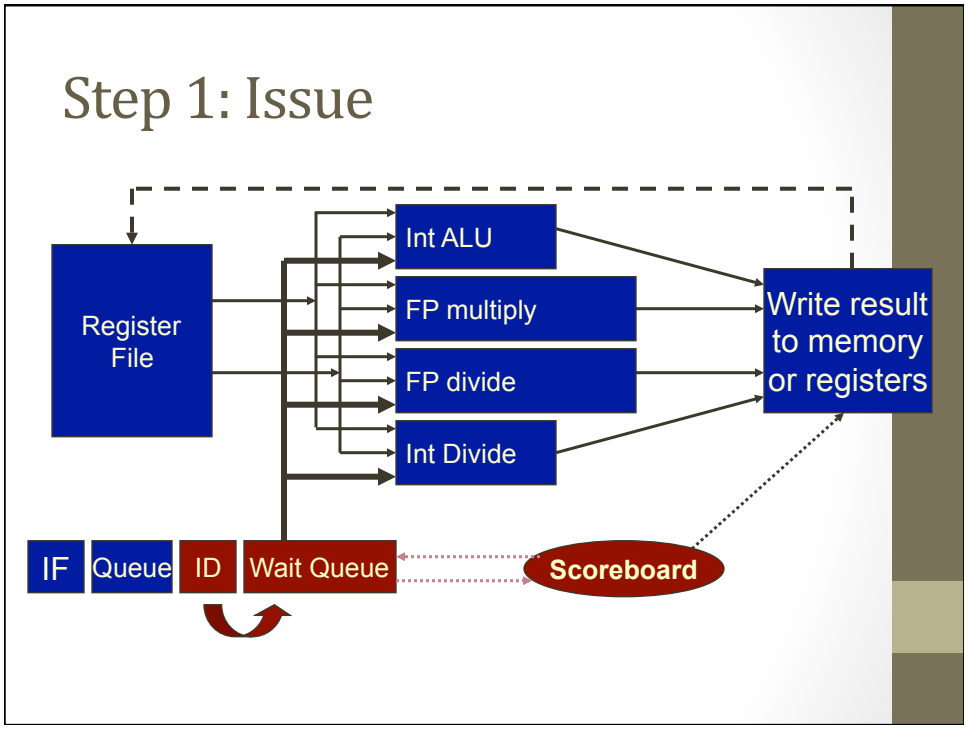


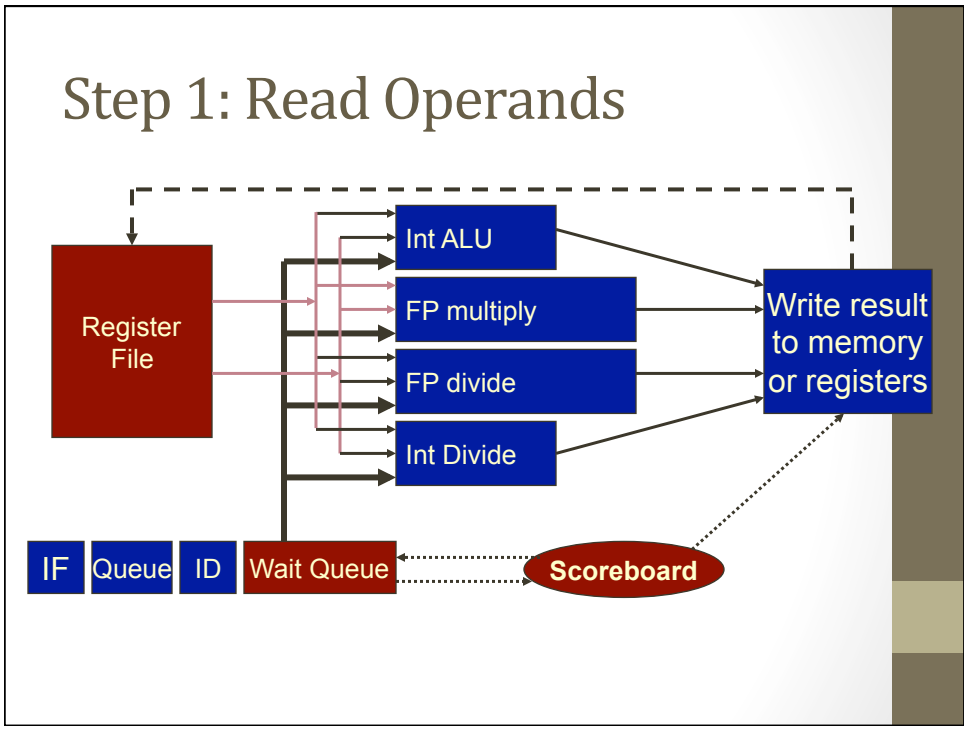
(The "wait queue" is virtual and Doesn't necessarily exist – it's part of the scoreboard.)

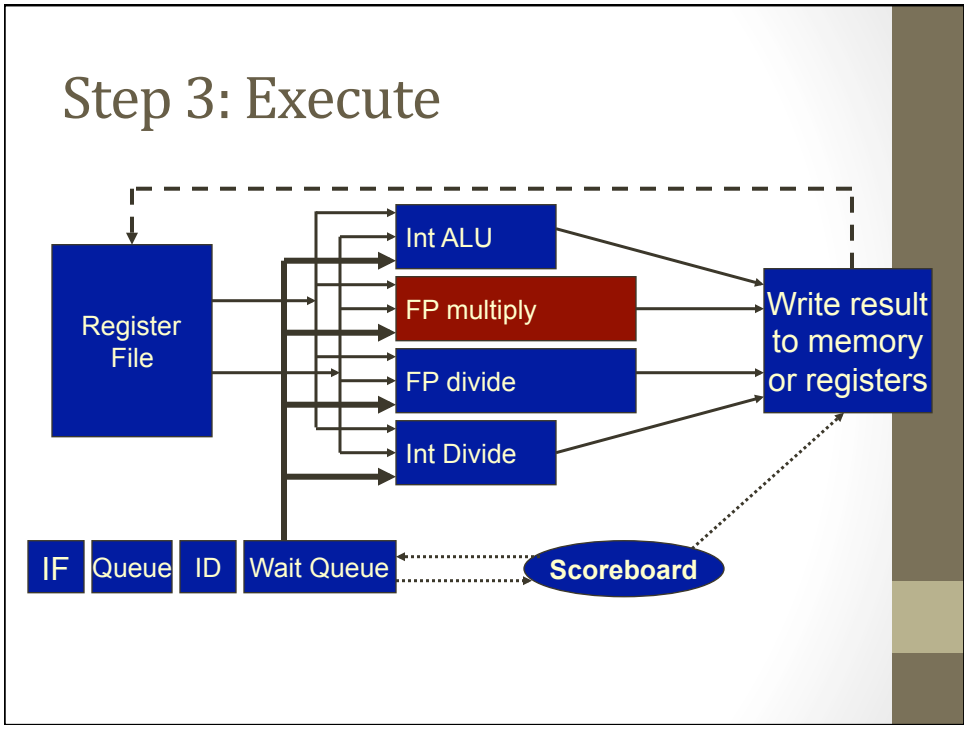
Stage of Scoreboard Control

1. **Issue (I)** - decode instruction, check for structural and WAW hazards, stall when necessary
2. **Read Operands (RO)** - wait until no RAW hazards, then read operands, send operation to FU
3. **Execution (EX)** - FU starts execution upon receiving the operands and notifies scoreboard when it's completed
4. **Write Result (WR)** - Scoreboard checks for WAR hazards and stalls write to register file to avoid them

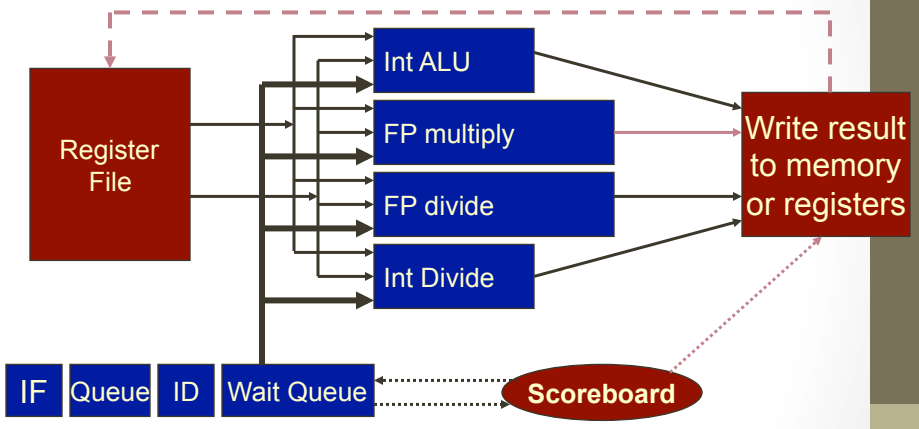

Step 0: Fetch
Register File
Int ALU
FP multiply
FP divide
Int Divide
Write result to memory or registers
IF
Queue
ID
Wait Queue
Scoreboard


Step 1: Issue
Register File
Int ALU
FP multiply
FP divide
Int Divide
Write result to memory or registers
IF
Queue
ID
Wait Queue
Scoreboard


Step 1: Read Operands
Register File
Int ALU
FP multiply
FP divide
Int Divide
Write result to memory or registers
IF
Queue
ID
Wait Queue
Scoreboard


Step 3: Execute
Register File
Int ALU
FP multiply
FP divide
Int Divide
Write result to memory or registers
IF
Queue
ID
Wait Queue
Scoreboard

Step 4: Write Result



Scoreboard Data Structures

- **Instruction status** – current step (of 4) for an
- **Functional unit status**

Field	Description
Busy	Unit busy or not
Op	Operation to perform (e.g., +, -)
Fi	Destination register
Fj,Fk	Source register numbers
Qj,Qk	FUs producing sources Fj,Fk
Rj,Rk	Flags indicating Fj,Fk are ready

- **Register result status** - which FU writes a given register (used to set Qj, Qk when issuing)

Hazard Detection

- **Structural hazards** - in I, ensure that a functional unit is available (makes a "reservation")
- **WAW hazards** - in I, ensure that no previous active instruction has the same destination
- **RAW hazards** - in RO, check that no previous active instruction writes a source register
- **WAR hazards** - in WR, before writing a result, check if any previous instructions that haven't gone past RO need that register as a source

Scoreboard Pipeline Control

Status	Wait until	Bookkeeping
Issue	FU not busy && not result	Busy(FU)←Y; Op(FU)←op; Fi(FU)←D; Fj(FU)←S1; Fk(FU)←S2; Qj←Res(S1); Qk←Res(S2); Rj←!Qj; Rk←!Qk; Res(D)←FU
Read ops	Rj && Rk	Rj←N; Rk←N
Executed	FU done	
Write dest	∀f((Fj(f)≠Fi(FU)|| Rj(f)=N) && (Fk(f)≠Fi(FU) || Rk(f)=N))	∀f(if Qj(f)=FU, Rj(f)←Y); ∀f(if Qk(f)=FU, Rj(f)←Y); Result(Fi(FU))←0; Busy(FU)←N

Write Destination

Wait Until	Bookkeeping
∀f((Fj(f)≠Fi(FU)|| Rj(f)=N) && (Fk(f)≠Fi(FU) || Rk(f)=N))	∀f(if Qj(f)=FU, Rj(f)←Y); ∀f(if Qk(f)=FU, Rj(f)←Y); Result(Fi(FU))←0; Busy(FU)←N

Wait until condition says:

Fj(f)≠Fi(FU)	does this FU write a result used by another FU?
Rj(f)=N	is the other FU waiting for this result?

Bookkeeping says:

if Qj(f)=FU, Rj(f)←Y	set register ready for all consumer FUs
Result(Fi(FU))←0;	clear entry in the register result table

Scoreboard Example

Code Sequence

```
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 Units

1 Integer, 2 FP multipliers, 1 FP adder, 1 FP divider

Pipeline Latencies

LD	1 cycle (uses Integer unit)
MULT	10 cycles (used FP multiplier)
SUBD, ADDD	2 cycles (uses FP adder)
DIVD	40 cycles (uses FP divider)

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

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

CLOCK

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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

				D	S1	S2	F1	F2	R1	R2
Time	*Name*	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
	Integer	Yes	Ld	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU				Int					

CLOCK **1 - fill in FU status, mark FU producing result**

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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

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

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU				Int					

CLOCK 2

Issue 2nd LD?

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
	Integer	Yes	Ld	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU				Int					

CLOCK 3 - single cycle load completes

What if we had a multi-cycle load???

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
	Integer	Yes	Ld	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU				Int					

CLOCK 4 - load completes, write result (WAR?)

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
	Integer	Yes	Ld	F2		R3				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU		Int							

CLOCK 5 - structural hazard for Int unit resolved

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
	Integer	Yes	Ld	F2		R3				Yes
	Mult1	Yes	Mult	F0	F2	F4	Int		No	Yes
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	Int							

CLOCK 6 - LD operands available, issue MULT

Can the SUBD issue on the next cycle?

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
	Integer	Yes	Ld	F2		R3				Yes
	Mult1	Yes	Mult	F0	F2	F4	Int		No	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2		Int	Yes	No
	Divide	No								

Register result status

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

CLOCK 7 - LD finishes, MULT stalls (why?)

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
	Integer	Yes	Ld	F2		R3				Yes
	Mult1	Yes	Mult	F0	F2	F4	Int		No	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2		Int	Yes	No
	Divide	Yes	Div	F10	F0	F6	M1		No	Yes

Register result status

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

CLOCK 8(a) - immediately before LD completes

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*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	M1		No	Yes

Register result status

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

CLOCK 8(b) (the LD completes. M1 and ADD ready.)

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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

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	M1		No	Yes

Register result status

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

CLOCK 9 - MULT and SUBD's operands ready, go!

Can the ADDD issue next cycle? What happens?

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*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	M1		No	Yes

Register result status

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

CLOCK 11 - SUBD finishes before MULT

Can SUBD write its result?

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
	Integer	No								
7	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	No								
	Divide	Yes	Div	F10	F0	F6	M1		No	Yes

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1					Div			

CLOCK 12 - SUBD *completes* before MULT

Can we read operands for DIVD? What about ADDD?

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*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	M1		No	Yes

Register result status

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

CLOCK 13 - ADDD structural hazard cleared

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*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	M1		No	Yes

Register result status

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

CLOCK 14 - ADDD reads operands

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*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	M1		No	Yes

Register result status

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

CLOCK 15

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*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	M1		No	Yes

Register result status

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

CLOCK 16 - ADDD finishes

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*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	M1		No	Yes

Register result status

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

CLOCK 17

Can we write result of ADDD? What about the Add FU?

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*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	M1		No	Yes

Register result status

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

CLOCK 18 - Yikes! Add FU still occupied.

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*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	M1		No	Yes

Register result status

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

CLOCK 19 - MULT finishes after SUBD

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*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

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU				Add		Div			

CLOCK 20 - MULT completes, DIVD ready to go

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*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

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU				Add		Div			

CLOCK 21 - DIVD has operands

Can ADDD now complete on next cycle?

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*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

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU						Div			

CLOCK 22 - DIVD executes, ADDD writes result

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*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

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU						Div			

CLOCK 61 - DIVD finishes

Instruction status

Op	d	j	k	*Issue*	*Read*	*Finish*	*Write*
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*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

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

CLOCK 62 (DIVD writes its result back)

Scoreboarding Complications

- Structural hazards possible on buses
  - Buses to/from register file may be limited
  - Ensure there are not more instructions “in flight” than can successfully fetch registers
- No forwarding
  - Operands are read only when both are available in the register file
  - Fortunately, instructions write into the register file right away

Benefit of Scoreboarding

- Can issue independent instructions "around data hazards"
- On CDC6600
  - Speedup 1.7 for FORTRAN programs
  - Speedup 2.5 for hand-coded assembly
- Of course: There's a cost! Lots o' buses!

Limitations of Scoreboards

- For the CDC 6600 scoreboard (our example)
  - No forwarding
  - Limited to a basic block
  - Does not issue on structural hazards - **stalls pipeline!**
  - Waits for WAR and prevents WAW hazards - **we don't really do anything about these hazards!**
  - A completed instruction on WAR occupies an FU
- Scoreboards in general have limitations too
  - Many parallel buses between registers and pipeline units
  - The number of scoreboard entries – **"instruction window"** size
  - Number of available functional units