Computer Science 146
Computer Architecture

Fall 2019

Harvard University

Instructor: Prof. David Brooks
dbrooks@eecs.harvard.edu

Lecture 5: Exceptions, Multicycle Ops,
Dynamic Scheduling

Lecture Outline

- Homework 1 Questions?
- Finish up on control hazards…
- Multicycle Operations
  - Hazards
  - Precise Exceptions
- Dynamic Scheduling with Scoreboards
- Next Time:
  - Dynamic Scheduling with Tomasulo's Algorithm

Control Hazards

Cycle	1	2	3	4	5	6	7	8
Branch Instr.	IF	ID	EX	MEM	WB			
Instr +1		IF	stall	stall	IF	ID	EX	MEM
Instr +2			stall	stall	stall	IF	ID	EX

- In base pipeline, branch outcome not known until MEM
- Simple solution – stall until outcome is known
- Length of control hazard is branch delay
  - In this simple case, it is 3 cycles (assume 10% cond. branches)
  - $CPI_{Real} = CPI_{Ideal} + CPI_{Stall} = 1.0 + 3 \text{ cycles} * .1 = 1.3$

Control Hazards: Solutions
Fast Branch Resolution

- Performance penalty could be more than 30%
  - Deeper pipelines, some code is very branch heavy
- Fast Branch Resolution
  - Adder in ID for PC + immediate targets
  - Only works for simple conditions (compare to 0)
  - Comparing two register values could be too slow

Cycle	1	2	3	4	5	6	7	8
Branch Instr.	IF	ID	EX	MEM	WB			
Instr +1		stall	IF	ID	EX	MEM	WB	
Instr +2			stall	IF	ID	EX	MEM	WB

Control Hazards: Branch Characteristics

- Integer Benchmarks: 14-16% instructions are conditional branches
- FP: 3-12%
- On Average:
  - 67% of conditional branches are "taken"
  - 60% of forward branches are taken
  - 85% of backward branches are taken
  - Why?

Control Hazards: Solutions

1. Stall Pipeline
   - Simple, No backing up, No Problems with Exceptions
2. Assume not taken
   - Speculation requires back-out logic:
     - What about exceptions, auto-increment, etc
   - Bets the "wrong way"
3. Assume taken
   - Doesn't help in simple pipeline! (don't know target)
4. Delay Branches
   - Can help a bit… we'll see pro's and con's soon

Control Hazards: Assume Not Taken

Cycle	1	2	3	4	5	6	7	8
Untaken Branch	IF	ID	EX	MEM	WB			
Instr +1		IF	ID	EX	MEM	WB		
Instr +2			IF	ID	EX	MEM	WB	

Looks good if we're right!

Cycle	1	2	3	4	5	6	7	8
Taken Branch	IF	ID	EX	MEM	WB			
Instr +1		IF	flush	flush	flush	flush		
Branch Target			IF	ID	EX	MEM	WB	
Branch Target +1				IF	ID	EX	MEM	WB

Control Hazards: Branch Delay Slots

- Find one instruction that will be executed no matter which way the branch goes
- Now we don't care which way the branch goes!
  - Harder than it sounds to find instructions
- What to put in the slot (80% of the time)
  - Instruction from before the branch (indep. of branch)
  - Instruction from taken or not-taken path
    - Always safe to execute? May need clean-up code (or nullifying branches)
    - Helps if you go the right way
- Slots don't help much with today's machines
  - Interrupts are more difficult (why? We'll see soon)

Now for the hard stuff!

- Precise Interrupts
  - What are interrupts?
  - Why do they have to be precise?

- Must have well-defined state at interrupt
  - All older instructions are complete
  - All younger instructions have not started
  - All interrupts are taken in program order

Interrupt Taxonomy

- Synchronous vs. Asychronous (HW error, I/O)
- User Request (exception?) vs. Coerced
- User maskable vs. Nonmaskable (Ignorable)
- Within vs. Between Instructions
- Resume vs. Terminate

The difficult exceptions are *resumable* interrupts *within* instructions

- Save the state, correct the cause, restore the state, continue execution

Interrupt Taxonomy

Exception Type	Sync vs. Async	User Request Vs. Coerced	User mask vs. nommask	Within vs. Between Insn	Resume vs. terminate
I/O Device Req.	Async	Coerced	Nonmask	Between	Resume
Invoke O/S	Sync	User	Nonmask	Between	Resume
Tracing Instructions	Sync	User	Maskable	Between	Resume
Breakpoint	Sync	User	Maskable	Between	Resume
Arithmetic Overflow	Sync	Coerced	Maskable	Within	Resume
Page Fault (not in main m)	Sync	Coerced	Nonmask	Within	Resume
Misaligned Memory	Sync	Coerced	Maskable	Within	Resume
Mem. Protection Violation	Sync	Coerced	Nonmask	Within	Resume
Using Undefined Insns	Sync	Coerced	Nonmask	Within	Terminate
Hardware/Power Failure	Async	Coerced	Nonmask	Within	Terminate

Interrupts on Instruction Phases

Exception Type	IF	ID	EXE	MEM	WB
Arithmetic Overflow			X		
Page Fault (not in main memory)	X			X	
Misaligned Memory	X			X	
Mem. Protection Violation	X			X	

- Exceptions can occur on many different phases
- However, exceptions are only handled in WB
- Why?

load IF ID EX MEM WB
add IF ID EX MEM WB

How to take an exception?

1. Force a trap instruction on the next IF
2. Squash younger instructions (Turn off all writes (register/memory) for faulting instruction and all instructions that follow it)
3. Save all processor state after trap begins
   - PC-chain, PSW, Condition Codes, trap condition
   - PC-chain is length of the branch delay plus 1
4. Perform the trap/exception code then restart where we left off

Summary of Exceptions

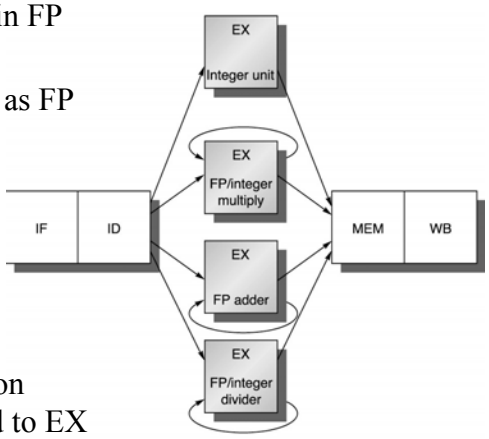
- Precise interrupts are a headache!
- All architected state must be precise
- Delayed branches
- Preview: Out-of-Order completion
  - What if something writes-back earlier than the exception?
- Some machines punt on the problem
  - Precise exceptions only for integer pipe
  - Special “precise mode” used for debugging (10x slower)

Multicycle Operations

- Basic RISC pipeline
  - All operations take 1 cycle
- Unfortunately, not the case in real processors
  - FP add, Integer/FP Multiply can be 2-6 cycles
  - 20-50 cycles for integer/FP divide, square root
  - Cache misses can be hundreds of cycles
- Difficulties
  - Hard to pipeline
  - Differ in number of clock cycles
  - Number of operands varies

Multicycle Operations

- For example, longer latency in FP unit
- EX may continue for as long as FP takes to finish
- Assume four separate ALUs
  - Integer unit
  - FP/Integer Multiplier
  - FP Adder
  - FP/Integer Divider
- Instruction stalls all instruction behind it if it can not proceed to EX

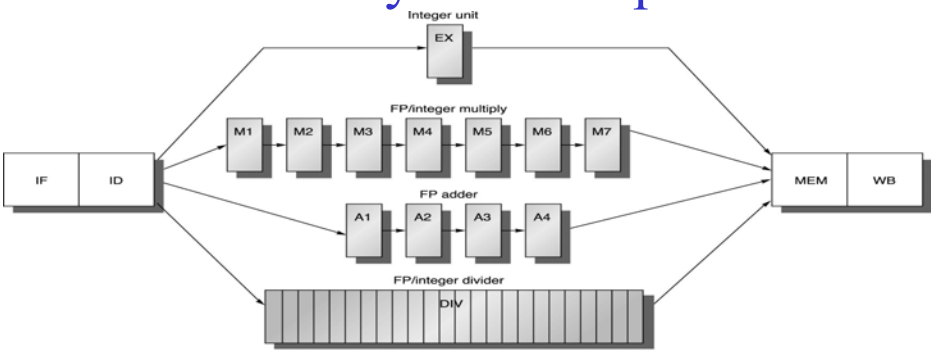


Multicycle Metrics

- Initiation Interval
  - Number of cycles that must elapse between issuing 2 operations of a given type
- Latency
  - Number of cycles between an instruction that *produces* a result and an instruction that *uses* the result

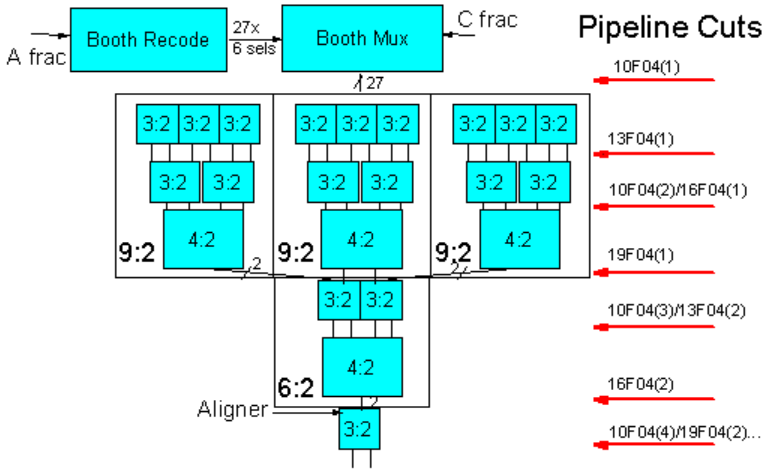
Functional Unit	Latency	Initiation Interval
Integer ALU	0	1
Data Memory	1	1
FP Add	3	1
FP Multiply	6	1
FP Divide	24	24

Multicycle Example

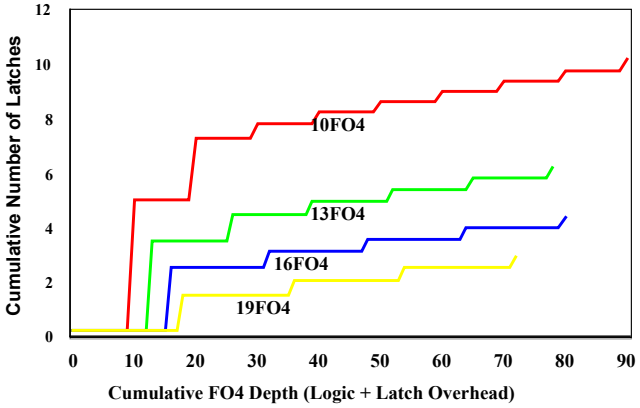


ADD R1, R2, R3	IF	ID	EX	MEM	WB					
DIVD F2, F2, F3		IF	ID	E1	E2	E3	…	E25	MEM	WB
ADDD F10, F2, F8			IF	ID	------Stall-------				E1	E2

Pipelining a FP Multiplier



Pipelining a FP Multiplier: Limits to Pipelining



Multicycle: Hazards/Exceptions

- New Issues
  - Structural Hazards on non-pipelined units
  - Register writes per cycle can > 1 (what is the max?)
  - WAW hazards are possible – are WAR?
  - Instruction complete out of order (what is the problem?)
  - Longer latency ops (what is the problem?)

Structural Hazards

- FP Divide not pipelined
  - Too much hardware needed
- Register Write port contention
  - Can fix through replicating hardware (multiported register file)
  - Can fix through stalls in ID
    - Track WB usage in ID with WB reservation bits (shift register)
    - Simplest scheme (all hardware is in ID)
  - Can fix through stalls when entering MEM or WB
    - Less hardware, but multiple stall points

WAW Hazards

- Why weren't they a problem before?

MULD F0, F4, F6	IF	ID	M1	M2	M3	M4	M5	M6	M7	MEM	WB
…		IF	ID	EX	MEM	WB					
…			IF	ID	EX	MEM	WB				
ADDD F0, F4, F6				IF	ID	A1	A2	A3	A4	MEM	WB

- Are they a problem?
  - Why generate 2 writes without an intervening read?
    - Branch Delay slots, Instruction that trap conflict with trap handler
  - Could happen, so we must check

WAW Hazard Logic

- Solutions:
  - Stall younger instruction writeback
    - Intuitive solution, fairly simple implementation
  - Squash older instruction writeback
    - Why not? The younger will overwrite it anyway…
    - No stalling/performance loss
    - What about precise exceptions?

Multicycle: Summary of Hazards

- Three additional checks must be performed in ID
  - Check for structural hazards
    - Make sure functional unit (FU) is not busy
    - Make sure Reg Write port is available
  - Check for RAW data hazard
    - Wait until sources are not a pending destination in any pipeline registers not available before instruction needs result
  - Check for WAW data hazards
    - Determine if any instruction in A1…A4, or M1…M7, or D has the same register destination as the instruction. If so stall!
- Concepts are the same, logic is more complicated

Multicycle: Out of Order Completion

- What could go wrong here?

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

- Solutions
  1. Ignore the problem (1960s, early 70s)
  2. Buffer the results (with forwarding) until all earlier ops complete
     - History files, Future Files (Can be combined with out-of-order issue)
  3. Imprecise exception with enough info to allow trap-handlers to clean up
  4. Hybrid: Allow issue to continue only if all older instructions have cleared their exception points

Instruction Level Parallelism (ILP)

- Quest for ILP drives much architecture research
  - Dynamic instruction scheduling
    - Scoreboarding
    - Tomasulo's Algorithm
    - Register Renaming (removing artificial dependencies WAR/WAWs)
  - Dynamic Branch Prediction
  - Superscalar/Multiple instruction Issue
  - Hardware Speculation
- All of these things fit well together

Dynamic Scheduling

- Scheduling tries to re-arrange instructions to improve performance
- Previously:
  - We assume when ID detects a hazard that cannot be hidden by bypassing/forwarding pipeline stalls
  - AND, we assume the compiler tries to reduce this
- Now:
  - Re-arrange instructions at runtime to reduce stalls
  - Why hardware and not compiler?

Goals of Scheduling

- Goal of Static Scheduling
  - Compiler tries to avoids/reduce dependencies
- Goal of Dynamic Scheduling
  - Hardware tries to avoid stalling when present
- Why hardware and not compiler?
  - Code Portability
  - More information available dynamically (run-time)
  - ISA can limit registers ID space
  - Speculation sometimes needs hardware to work well
  - Not everyone uses "gcc –O3"

Dynamic Scheduling: Basic Idea

- Example

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

- Hazard detection during decode stalls whole pipeline
- No reason to stall in these cases: Out-of-Order Execution
  - Reduces stalls, improved FU utilization, more parallel execution
  - To give the appearance of sequential execution: precise interrupts
  - First, we will study without this, then see how to add them

How to implement?

- Previously:
  - Instruction Decode and Operation Fetch are a single cycle
- Now:
  - Split ID into two phases
    - Issue – decode instructions, check for *structural* hazards
    - Read Operands – Wait until no *data* hazards, then read ops
  - Dividing hazard checks into a two-step process
- Out-of-order execution => WAR/WAW hazards

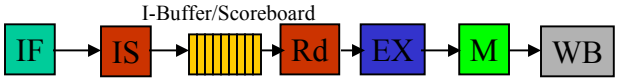
```
DIVD   F0, F2, F4
ADDD   F10, F0, F8
SUBD   F8, F8, F14
```
WAR

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

Scoreboarding Basics

- Previously:
  - Instruction Decode and Operation Fetch are a single cycle
- Now:
  - To support multiple instructions in ID stage, we need two things
    - Buffered storage (Instruction Buffer/Window/Queue)
    - Split ID into two phases

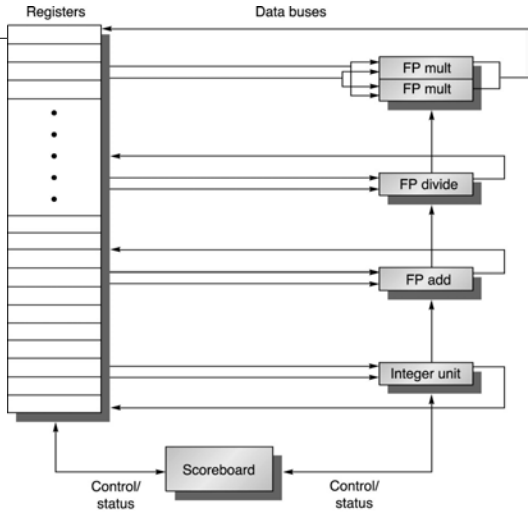
Standard Pipeline



New Pipeline

Scoreboarding

- Centralized scheme
  - No bypassing
  - WAR/WAW hazards are a problem
- Originally proposed in CDC6600 (S. Cray, 1964)



Scoreboarding Stages – Issue (Or Dispatch)

- Fetch – Same as before
- Issue (Check Structural Hazards)
  - If FU is free an no other active instruction has same destination register (WAW), then issue instruction
  - Do not issue until structural hazards cleared
  - Stalled instruction stay in I-Buffer
  - Size of buffer is also a structural Hazard
    - May have to stall Fetch if buffer fills
  - Note: Issue is In-Order, stalls stops younger instructions

Scoreboarding Stages – Read Operands (Or Issue!)

- Read Operands (Check Data Hazards)
  - Check scoreboard for whether source operands are available
  - Available?
    - No earlier issued active instructions will write register
    - No currently active FU is going to write it
  - Dynamically avoids RAW hazards

Scoreboarding Stages – Execution/Write Result

- Execution
  - Execute/Update scoreboard
- Write Result
  - Scoreboard checks for WAR stalls and stalls *completing* instruction, if necessary
  - Before, stalls only occur at the beginning of instructions, now it can be at the end as well
  - Can happen if:
    - Completing instruction *destination* register matches an older instruction that has not yet read its *source* operands

Scoreboarding Control Hardware

- Three main parts
  - Instruction Status Bits
    - Indicate which of the four stages instruction is in
  - Functional Unit Status Bits
    - Busy (In Use or not), Operation being Performed
    - Fi -- Destination Register, Fj, Fk, -- Source Registers
    - Qj, Qk – FU producing source regs Fj, Fk
    - Rj, Rk – Flags indicating when Fj, Fk are ready but not yet read
  - Register Result Status
    - Which FU will write each register

Scoreboard Example

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2				
LD	F2	45+	R3				
MULTD	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 Qj*	*FU 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									

Example courtesy of Prof. Broderson, CS152, UCB, *Copyright (C) 2001 UCB*

Scoreboard Example: Cycle 1

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write Result
LD	F6	34+	R2	1			
LD	F2	45+	R3				
MULTD	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 Qj	FU 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 Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2		
LD	F2	45+	R3				
MULTD	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 Qj	FU 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					

- **Issue 2nd LD?**

Scoreboard Example: Cycle 3

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write 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				

Functional unit status:

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F6		R2				No
	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 Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3				
MULTD	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 Qj	FU 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
4	FU				Integer					

Scoreboard Example: Cycle 5

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5			
MULTD	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 Qj	FU 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

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6		
MULTD	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 Qj	FU 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 Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	
MULTD	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 Qj	FU Qk	Fj? Rj	Fk? 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								

Register result status:

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

- **Read multiply operands?**

Scoreboard Example: Cycle 8a (First half of clock cycle)

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	
MULTD	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 Qj	FU Qk	Fj? Rj	Fk? 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	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 (Second half of clock cycle)

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	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 Qj	FU 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 Oper	Exec Comp	Write 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		
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Functional unit status:

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU 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

Note Remaining →

Register result status:

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

- Read operands for MULT & SUB? Issue ADDD?

Scoreboard Example: Cycle 10

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write 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		
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Functional unit status:

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

Register result status:

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

Scoreboard Example: Cycle 11

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write 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	
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Functional unit status:

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
	Integer	No								
8	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
0	Add	Yes	Sub	F8	F6	F2			No	No
	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 Oper	Exec Comp	Write 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				

Functional unit status:

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
	Integer	No								
7	Mult1	Yes	Mult	F0	F2	F4			No	No
	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:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write 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			

Functional unit status:

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
	Integer	No								
6	Mult1	Yes	Mult	F0	F2	F4			No	No
	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 Oper	Exec Comp	Write 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		

Functional unit status:

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? 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

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 Oper	Exec Comp	Write 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		

Functional unit status:

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
	Integer	No								
4	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
1	Add	Yes	Add	F6	F8	F2			No	No
	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 Oper	Exec Comp	Write 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	

Functional unit status:

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

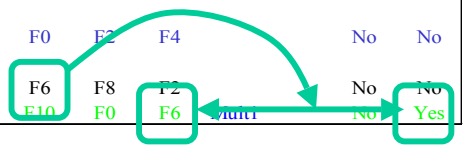
Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write 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	

WAR Hazard!

Functional unit status:

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
	Integer	No								
2	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	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			

- **Why not write result of ADD???**

Scoreboard Example: Cycle 18

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write 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	

Functional unit status:

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
	Integer	No								
1	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	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:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write 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	

Functional unit status:

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
	Integer	No								
0	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	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 Oper	Exec Comp	Write 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
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 Qj	FU Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	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:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write 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
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 Qj	FU Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status:

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

- WAR Hazard is now gone...

Scoreboard Example: Cycle 22

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write 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
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 Qj	FU Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
39	Divide	Yes	Div	F10	F0	F6			No	No

Register result status:

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

Faster than light computation (skip a couple of cycles)

Scoreboard Example: Cycle 61

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write 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
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 Qj	FU Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
0	Divide	Yes	Div	F10	F0	F6			No	No

Register result status:

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

Scoreboard Example: Cycle 62

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write 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
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 Qj	FU 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
62	*FU*									

Review: Scoreboard Example: Cycle 62

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write 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
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 Qj	FU 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
62	*FU*									

- **In-order issue; out-of-order execute & commit**

Scoreboarding Limitations

- Number and type of functional units
- Number of instruction buffer entries (scoreboard size)
- Amount of application ILP (RAW hazards)
- Presence of antidependencies (WAR) and output dependencies (WAW)

For next time

- Don’t forget HW1
- Dynamic Scheduling, Tomasulo Style
- Compare the two methods