Instruction-Level Parallelism
Dynamic Scheduling

CS448

Dynamic Scheduling

- Static Scheduling – Compiler rearranging instructions to reduce stalls
- Dynamic Scheduling – Hardware rearranges instruction stream to reduce stalls
  - Possible to account for dependencies that are only known at run time - e. g. memory reference
  - Simplifies the compiler
  - Code built for one pipeline in mind could also run efficiently on a different pipeline
  - minimizes the need for speculative slot filling - NP hard
- Once a common tactic with supercomputers and mainframes but was too expensive for single-chip
  - Now fits on a desktop PC's

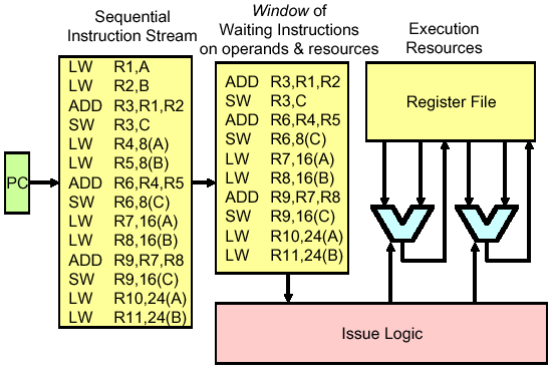
Dynamic Scheduling

- Dependencies that are close together stall the entire pipeline
  - DIVD F0, F2, F4
  - ADDD F10, F0, F8
  - SUBD F12, F8, F14
- The ADD needs the DIV to finish, so there is a stall… which also stalls the SUBD
  - Looong stall for DIV
  - But the SUBD is independent so there is no reason why we shouldn't execute it
  - Or is there?
    - Precise Interrupts - Ignore for now
- Compiler could rearrange instructions, but so could the hardware

3

Dynamic Scheduling

- It would be desirable to decode instructions into the pipe in order but then let them stall individually while waiting for operands before issue to execution units.
- Dynamic Scheduling – Out of Order Issue / Execution
- Scoreboarding



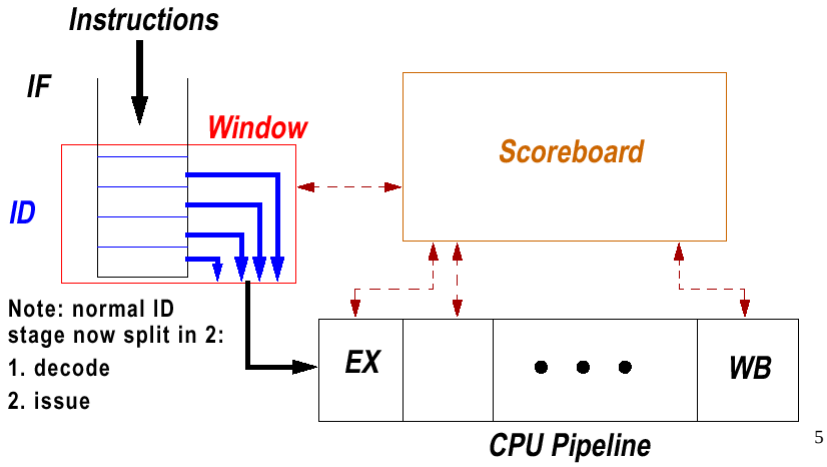
4

Scoreboarding

Split ID stage into two:

Issue (Decode, check for Structural hazards)

Read Operands (Wait until no data hazards, read operands)



5

Scoreboard Overview

- Consider another example:
  - DIVD F0, F2, F4
  - ADDD F10, F0, F8 ; RAW Hazard
  - SUBD F8, F8, F14 ; WAR Hazard
- Note:
  - We may still have to stall in the Scoreboard
  - SUBD wouldn't have to wait if we had a different register other than F8
- MIPS Pipeline changes again
  - Ignore MEM and IF stages for now to focus on the scoreboard

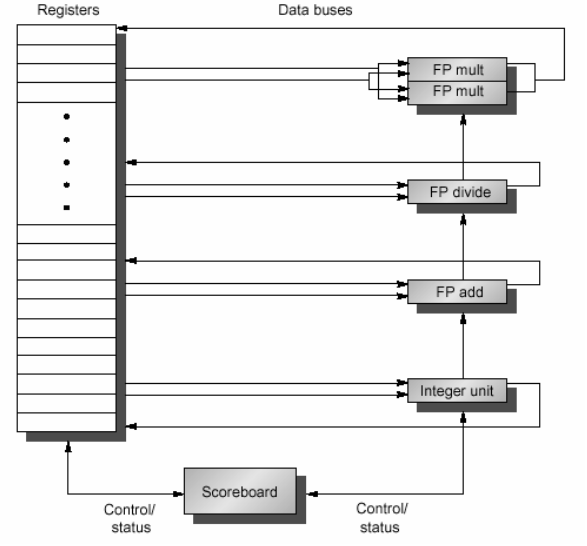
6

Scoreboard and the CDC

- Introduced on CDC 6600
  - 4 Floating Point Units, 5 Memory, 7 Integer Units
  - Load/Store architecture
- Results
  - 70 to 150% improvement
    - Good since this cost over $1 million
  - Recall this is an old machine with no cache, pipelining, primitive compilers
  - Fastest machine of the time
- Back today in IBM, Sun machines, similar concepts in Intel chips

7

MIPS Scoreboard



Two MULT units

Note: Need a lot more buses

8

New Pipeline Steps

IF
- Ignoring IF, and MEM
- Issue
  - If functional unit free and no other active instruction has the same destination register, issue the instruction to the functional unit
  - Stall for WAW hazards or structural hazards
  - Nothing out of order yet
- Read Operands
  - Monitor source operands, if available for an active instruction, then send to the functional unit to execute
  - Instructions get out of order at this step, dynamic RAW resolution

EX
- Execution
  - Starts when operands ready, notifies scoreboard when done

WB
- Write Result
  - Stall for WAR hazards before writing result back to register file

9

Scoreboard Data Structure

- Instruction Status
  - Indicates which of the four steps the instruction is in (Issue, Read Operands, Execute Complete, Write Result)
  - The capacity of the instruction status is the **window size**
- Functional Unit Status
  - Busy : Yes/No indicates if busy or not
  - Op: Operation to perform, e.g. Add or Subtract
  - F_i : Destination Register
  - F_j, F_k : Source Registers
  - Q_j, Q_k : Functional Units producing source register j, k
  - R_j, R_k : Yes/No indicates if source j or k is ready or not, set to NO after operands are read
- Register Result Status
  - For each register, indicates the Functional Unit that will write to it
  - Blank means nothing will write to it
  - Note overlap of this info with Q/F fields, but indexed by register, not instr

10

Example Scoreboard

		Instruction status			
Instruction		**Issue**	**Read operands**	**Execution complete**	**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								
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	Yes	Div	F10	F0	F6	Mult1		No	Yes

	Register result status								
	F0	**F2**	**F4**	**F6**	**F8**	**F10**	**F12**	**...**	**F30**
FU	Mult1	Integer			Sub	Divide			

11

Scoreboard Example

- Assume the following EX latencies
  - FP Add = 2 clock cycles
  - FP Mult = 10 clock cycles
  - FP Div = 40 clock cycles
- Assume entire code fragment (basic block) fits in the window
  - Also assume the code fragment is already loaded
- Issue Rules
  - Issue maximum 1 instruction per cycle
  - Clearing hazards will depend on the pipeline structure

12

MIPS Scoreboarding Rules

- Issue
  - If FU not busy (no structural hazards) and no result pending for destination register (no WAW hazards) then
    - Assign FU entry
    - Set operand source in scoreboard, indicate if ready or not
- Read Operands
  - If source1 and source2 are both "ready" (no RAW hazards) then
    - Set ready=No for both, read registers, and start EX
- Execute
  - Notify scoreboard when complete
- Writeback
  - If none of the functional units needs to use our result register and has yet to read it (no WAR hazards) then
    - Set ready=Yes for any functional unit sources waiting for this FU
    - write the register file, clear the FU status, and clear our destination's register result pending entry.
- Stages must stall for hazards

13

Scoreboard – Cycle 1

IS

Instr	Issue	Read Operands	Exec Complete	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				

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	LD	F6		R2				Yes
Mult1									
Mult2									
Add									
Divide									

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU				Int					

14

Scoreboard – Cycle 2

IS

Instr	Issue	Read Operands	Exec Complete	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				

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	LD	F6		R2				Yes
Mult1									
Mult2									
Add									
Divide									

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU				Int					

Issue 2nd LD?

Scoreboard – Cycle 3

IS

Instr	Issue	Read Operands	Exec Complete	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				

FS

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

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU				Int					

Issue Mult?

Scoreboard – Cycle 4

IS

Instr	Issue	Read Operands	Exec Complete	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				

FS

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

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU									

17

Scoreboard – Cycle 5

IS

Instr	Issue	Read Operands	Exec Complete	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				

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	LD	F2		R3				Yes
Mult1									
Mult2									
Add									
Divide									

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU		Int							

18

Scoreboard – Cycle 6

IS

Instr	Issue	Read Operands	Exec Complete	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				

FS

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									
Add									
Divide									

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU	Mul	Int							

19

Scoreboard – Cycle 7

IS

Instr	Issue	Read Operands	Exec Complete	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				

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	LD	F2		R3				No
Mult1	Yes	Mult	F0	F2	F4	Int		No	Yes
Mult2									
Add	Yes	Sub	F8	F6	F2		Int	Yes	No
Divide									

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU	Mul	Int			Add				

Read Mult operands?

20

Scoreboard – Cycle 8a

IS

Instr	Issue	Read Operands	Exec Complete	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				

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	LD	F2		R3				No
Mult1	Yes	Mult	F0	F2	F4	Int		No	Yes
Mult2									
Add	Yes	Sub	F8	F6	F2		Int	Yes	No
Divide	Yes	Div	F10	F0	F6	Mul1		No	Yes

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU	Mul	Int			Add	Div			

Issue first

21

Scoreboard – Cycle 8b

IS

Instr	Issue	Read Operands	Exec Complete	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				

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	LD	F2		R3				No
Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
Mult2									
Add	Yes	Sub	F8	F6	F2			Yes	Yes
Divide	Yes	Div	F10	F0	F6	Mul1		No	Yes

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU	Mul				Add	Div			

Write result

22

Scoreboard – Cycle 9

IS

Instr	Issue	Read Operands	Exec Complete	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				

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
Mult2									
Add	Yes	Sub	F8	F6	F2			Yes	Yes
Divide	Yes	Div	F10	F0	F6	Mul1		No	Yes

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU	Mul				Add	Div			

Issue ADD?

Scoreboard – Cycle 10

IS

Instr	Issue	Read Operands	Exec Complete	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				

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			No	No
Mult2									
Add	Yes	Sub	F8	F6	F2			No	No
Divide	Yes	Div	F10	F0	F6	Mul1		No	Yes

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU	Mul				Add	Div			

Exec MULT, ADD

Scoreboard – Cycle 11

IS

Instr	Issue	Read Operands	Exec Complete	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				

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			No	No
Mult2									
Add	Yes	Sub	F8	F6	F2			No	No
Divide	Yes	Div	F10	F0	F6	Mul1		No	Yes

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU	Mul				Add	Div			

Scoreboard – Cycle 12

IS

Instr	Issue	Read Operands	Exec Complete	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				

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			No	No
Mult2									
Add	Yes	Sub	F8	F6	F2			No	No
Divide	Yes	Div	F10	F0	F6	Mul1		No	Yes

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU	Mul					Div			

Read opnds DIVD?

Scoreboard – Cycle 13

IS

Instr	Issue	Read Operands	Exec Complete	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			

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			No	No
Mult2									
Add	Yes	Add	F6	F8	F2			Yes	Yes
Divide	Yes	Div	F10	F0	F6	Mul1		No	Yes

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU	Mul			Add		Div			

27

Scoreboard – Cycle 14

IS

Instr	Issue	Read Operands	Exec Complete	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		

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			No	No
Mult2									
Add	Yes	Add	F6	F8	F2			Yes	Yes
Divide	Yes	Div	F10	F0	F6	Mul1		No	Yes

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU	Mul			Add		Div			

28

Scoreboard – Cycle 15

IS

Instr	Issue	Read Operands	Exec Complete	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		

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			No	No
Mult2									
Add	Yes	Add	F6	F8	F2			No	No
Divide	Yes	Div	F10	F0	F6	Mul1		No	Yes

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU	Mul			Add		Div			

Start EX on ADD

29

Scoreboard – Cycle 16

IS

Instr	Issue	Read Operands	Exec Complete	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	

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			No	No
Mult2									
Add	Yes	Add	F6	F8	F2			No	No
Divide	Yes	Div	F10	F0	F6	Mul1		No	Yes

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU	Mul			Add		Div			

30

Scoreboard – Cycle 17

IS

Instr	Issue	Read Operands	Exec Complete	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	

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			No	No
Mult2									
Add	Yes	Add	F6	F8	F2			No	No
Divide	Yes	Div	F10	F0	F6	Mul1		No	Yes

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU	Mul			Add		Div			

Write Result Add?

31

Scoreboard – Cycle 19

IS

Instr	Issue	Read Operands	Exec Complete	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	

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			No	No
Mult2									
Add	Yes	Add	F6	F8	F2			No	No
Divide	Yes	Div	F10	F0	F6	Mul1		No	Yes

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU	Mul			Add		Div			

32

Scoreboard – Cycle 20

IS

Instr	Issue	Read Operands	Exec Complete	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	

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			No	No
Mult2									
Add	Yes	Add	F6	F8	F2			No	No
Divide	Yes	Div	F10	F0	F6	Mul1		Yes	Yes

RRS

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

33

Scoreboard – Cycle 21

IS

Instr	Issue	Read Operands	Exec Complete	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	

FS

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

RRS

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

34

Scoreboard – Cycle 22

IS

Instr	Issue	Read Operands	Exec Complete	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

FS

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

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU						Div			

Safe to write

35

Scoreboard – Cycle 61

IS

Instr	Issue	Read Operands	Exec Complete	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

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No								
Mult1	No								
Mult2									
Add	No								
Divide	Yes	Div	F10	F0	F6			Yes	Yes

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU						Div			

36

Scoreboard – Cycle 62

IS

Instr	Issue	Read Operands	Exec Complete	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

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No								
Mult1	No								
Mult2									
Add	No								
Divide	Yes	Div	F10	F0	F6			Yes	Yes

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU									

37

IS

Instr	Issue	Read Operands	Exec Complete	Write Result
LD F2, 0(R1)				
ADDD F6, F2, F4				
MULTD F8, F6, F0				
LD F10, -8(R1)				
ADDD, F12, F10, F4				
MULTD F14, F12,F0				

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer									
Integer									
Mult									
Add									
Add									
Divide									

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU									

38

Instr	Issue	Read Operands	Exec Complete	Write Result
LD F2, 0(R1)	1	2	3	4
ADDD F6, F2, F4	2	5	7	8
MULTD F8, F6, F0	3	9	19	20
LD F10, -8(R1)	4	5	6	7
ADDD, F12, F10, F4	5	8	10	11
MULTD F14, F12,F0	21	22	32	33

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Y	Load	F2	R1				Yes	
Integer	Y	Load	F10	R1				Yes	
Mult	Y	Mult	F8	F6	F0	Add1		Yes	Yes
Add	Y	Add	F6	F2	F4	Int1		Yes	Yes
Add	Y	Add	F12	F10	F4	Int2		No	Yes
Divide									

	F0	F2	F4	F6	F8	F10	F12	F14	F30
FU					Mult1	Int2	Add2		

39

IS

Instr	Issue	Read Operands	Exec Complete	Write Result
DIVD F10, F8, F2				
ADDD F4, F10, F2				
LD F2, 8(R0)				
ADDD F6, F0, F12				

FS

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Int1									
Int2									
Mult									
Add									
Divide									

RRS

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU									

40

Scoreboarding Comments

- Could improve a bit with forwarding
  - Improve by one cycle for forwarded data
- Whats Missing
  - Control Hazards, i.e. branches
    - Dynamic Branch Prediction
    - Speculative Execution
- Limitations
  - Note we had to wait for a register to be read before writing
    - Could be avoided with Register Renaming
  - Amount of ILP limited by RAW hazards
    - Some estimate the amount of parallelism typically ~2
  - Window Size
  - Number and Type of Functional Units, Lots of Buses
  - Big increase in cost and complexity!

41

Static Scheduling Revisited

- Scoreboarding stalled for WAR and WAW hazards
- Consider the fragment
  - DIVD F1, F2, F3 ; slow
  - SUBD F2, F3, F4 ; must wait for DIV to read, WAR
  - SD 0(R1), F2
  - There is a WAR antidependence on F2 between the DIV and the SUB.
- By "renaming" F2 and subsequent uses of it we avoid the hazard
  - DIVD F1, F2, F3 ; slow
  - SUBD F5, F3, F4 ; Free to start anytime
  - SD 0(R1), F5
- Reduces stalls for some pipelines, but uses additional registers

42

Dynamic Scheduling

- Same idea as before, but in hardware, not software
- Every time you write to a register number, allocate a new physical register from a pool of virtual registers. Results in multiple copies of each register, one for each new value written to it.
- Implementation implications:
  - Need more physical registers than register addresses
  - Keep a pool of unassigned physical registers, and allocate one when a register is written
  - All uses to the original register number must be remapped to the new physical register
  - When a register is written, all previous physical registers allocated for it can be freed as soon as all previous instructions that use it have completed
- Can eliminate stalls for WAR and WAW hazards

43

Register Renaming Idea

- Initial mappings:
  - F1 → V1
  - F2 → V2
  - F3 → V3
  - F4 → V4
- DIVD F1, F2, F3 F1 → V5
- SUBD F2, F3, F4 F2 → V6
- ADDD F1, F3, F2 F1 → V7

Becomes

- DIVD V5, V2, V3
- SUBD V6, V3, V4
- ADDD V7, V3, V6 Note V6 here not V2

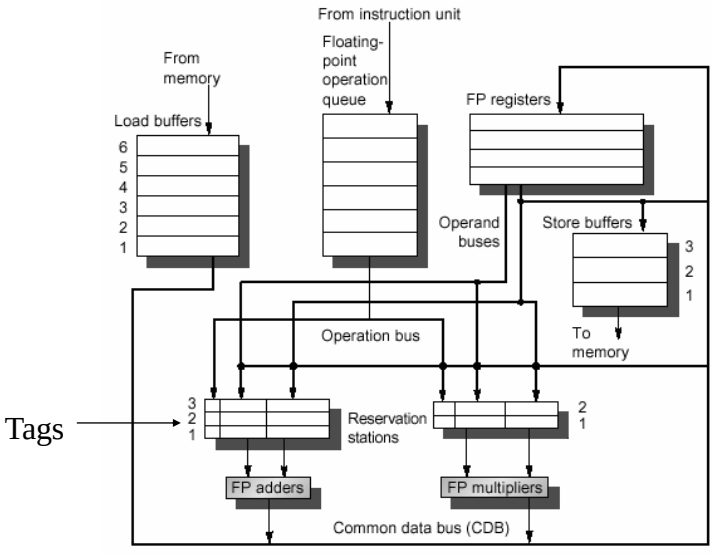
44

Tomasulo's Algorithm

- Uses register renaming
- Idea:
  - When opcode, operands, and functional unit is available, execute the instruction, data-centric
- Replace the Functional Unit Status table with reservation stations for each functional unit
  - The reservation station stores operand values as they become available, **or** the name of a reservation station that will provide the result. This means that the original register number is no longer needed, and is now "renamed" as a reservation station
- Each result is broadcast to any reservation station that needs it
- Instructions can now be issued even if there are structural hazards or WAR/WAW hazards.

45

MIPS FP via Tomasulo



46

Tomasulo vs. Scoreboarding

- Tomasulo's Algorithm
  - Used on IBM 360
  - attempt to deal with long memory and FP latencies
  - 360 permitted register to memory ALU instructions
- vs. CDC scoreboard
  - hazard detection and interlocks are distributed
    - Each functional unit has reservation stations to control execution
    - Reservation stations act as interlocks until data available
  - Results passed directly to functional units rather than through the registers (with renamed registers) multicast on CDB (common data bus)
- Dataflow centric
  - no WAR or WAW hazards exist - rename instead
  - Scoreboard was control centric

47

Tomasulo - Central Idea

- To speed up the 360 we can
  - Add more functional units, but they wouldn't be heavily used
  - Replicating registers is cheaper and increases utilization
  - Helped on 360 which had only 6 FP registers (similar to x86)
- Outcome
  - each execution unit fed by multiple reservation stations
  - Execution units used pipelining
  - Instruction issue to reservation stations is in-order, but reservation stations would execute upon acquiring all source operands → out of order execution
  - reservation stations contain virtual registers that remove WAW and WAR induced stalls

48

Dynamic Scheduling

- Both Scoreboarding and Tomasulo's algorithm were early approaches, but the ideas are still used today
  - Scoreboarding - CDC 6600 (1964)
  - Tomasulo - IBM 360/ 91 (1967)
- Both methods are used today
  - UltraSparc
  - Pentium/Itanium
    - More transistors dedicated to ILP in the original Pentium than in the entire 486