Announcements

- CW1 due Monday, Feb 20 @ 4.00pm
  - No need to hand anything in to ITO (use 'submit')

Dynamic Scheduling

- Pipelining: Issue instructions in every cycle (CPI → 1)
- Compiler scheduling (static scheduling) reduces impact of dependences
  - Increased compiler complexity, especially when attempting global scheduling (across BB's)
  - Limited information at compile time
    - Dynamically-linked libraries and OS functionality
    - Microarchitectural uncertainty: branch outcomes, memory addresses, cache misses
  - Not portable to different pipeline implementations
- Hardware scheduling so far: in-order instruction execution
  - Instructions after a stalled instruction must wait even if independent

Dynamic Scheduling: Main Idea

- Example:

```
DIV.D F0,F2,F4    ;F0=F2/F4
ADD.D F10,F0,F8   ;F10=F0+F8
SUB.D F12,F8,F14  ;F12=F8-F14
```

  - DIV.D is a long latency operation
  - ADD.D depends on DIV.D but SUB.D does not
- Solution: out-of-order execution
  - Detect dependence of ADD.D and block it
  - Detect that SUB.D is not dependent and execute it
  - Now SUB.D executes before ADD.D even though it comes after it in program order
  - Hardware must be able to look ahead of blocked instructions
  - Multiple functional units can be effectively used

Terminology

- Instruction fetch: fetch instruction from memory
- Instruction issue: decode instruction, check for structural hazards, and send to execution units
- Instruction execution: execute instruction after registers are read once dependences are cleared
- Instruction completion (or retire or commit): finish instruction and update processor state
- Some combinations are possible:
  - In-order issue, execution and completion
  - In-order issue and out-of-order execution and in-order completion
  - Out-of-order issue, execution and completion

Data dependences

- Read after Write - RAW (Flow, True)
  - MUL R3, R1,R2
  - DADD R5, R3, R4
- Write after Read – WAR (Anti, Name)
  - MUL R3, R1, R2
  - DADD R1, R5, R6
- Write after Write – WAW (Output, Name)
  - MUL R3, R1, R2
  - DADD R3, R4, R5

Dynamic Scheduling 1: Scoreboarding

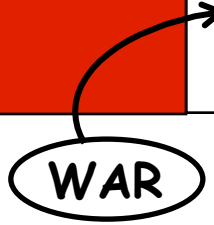
- Handles all RAW, WAR, and WAW with proper stalls, but allows independent instructions to proceed
- Step 1: Issue (part of original ID stage)
  - Issue instruction to functional unit iff functional unit is free and no earlier instruction writes to the same destination register (WAW)
- Step 2: Read operands (part of original ID stage)
  - Wait until source registers become available from earlier instructions through register file (RAW)
- Step 3: Execute (original EXE stage)
  - Execute instruction and notify scoreboard when done
- Step 4: Write result (original WB stage)
  - Wait until earlier instructions read operands before writing to register file (WAR)

Scoreboard Organization

- Instruction status: either one of the four steps of the instruction operation (Issue, Read Op, Execute, Write)
- Functional unit status:
  - Busy – functional unit is being used
  - Op – type of operation to be performed (e.g., add, sub, etc.)
  - F_i – destination register
  - F_j, F_k – source registers
  - Q_j, Q_k – functional units producing F_j and F_k
  - R_j, R_k – ready flag, indicates if F_j and F_k are ready but not yet read. Set to "no" after operands are read.
- Register result status: indicates which functional unit will write the register next (one per register), and also reserves register thereby detecting WAW hazards.

Scoreboarding Pipeline Control

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



Scoreboard Example

- Instruction sequence:

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

- Latencies:
  - Integer → 1 cycle
  - FP add → 2 cycles
  - FP multiply → 10 cycles
  - FP divide → 40 cycles
- Functional units: 1 integer (also for ld/st), 1 FP adder, 2 FP multipliers, 1 FP divider

Scoreboard example – cycle 1

Instruction status

Instruction	j	k	Issue	Read operand	Executi complet	Write Result
LD F6	34+	R2	1			
LD F2	45+	R3				
MULTI F0	F2	F4				
SUBD F8	F6	F2				
DIVD F10	F0	F6				
ADDD F6	F8	F2				

Issue LD #1

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

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

Scoreboard example – cycle 2

Instruction status

Instruction		j	k	Issue	Read operand	Execution 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				

LD #2 can't issue since integer unit is busy.
MULT can't issue because we require in-order issue.

Functional unit status

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

Scoreboard example – cycle 4

Instruction status

Instruction		j	k	Issue	Read operand	Execution complete	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3				
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 for Qj	FU for 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
4	FU									

Scoreboard example – cycle 5

Instruction status

Instruction		j	k	Issue	Read operand	Execution complete	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5			
MULTI	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F2		R3				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

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

Issue LD #2 since integer unit is now free.

Scoreboard example – cycle 6

Instruction status

Instruction		j	k	Issue	Read operand	Execution complete	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6		
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Issue MULT.

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F2		R3				No
	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	Mult	Integer							

Scoreboard example – cycle 7

Instruction status

Instruction		j	k	Issue	Read operand	Execution complete	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2	7			
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

MULT can't read its operands (F2) because LD #2 hasn't finished.

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F2		R3				No
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
	Mult2	No								
	Add	Yes	Subd	F8	F6	F2		Integer	Yes	No
	Divide	No								

Register result status

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

Scoreboard example – cycle 8

DIVD issues.
MULT and SUBD both waiting for F2.
LD #2 writes F2.

Instruction status

Instruction		j	k	Issue	Read operand	Execution complete	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTI	F0	F2	F4	6			
SUBD	F8	F6	F2	7			
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Functional unit status

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

Register result status

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

Scoreboard example – cycle 9

Now MULT and SUBD can both read F2.

Instruction status

Instruction		j	k	Issue	Read Op	EX compl	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 for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
10	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
2	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
9	FU	Mult1				Add	Divide			

Scoreboard example – cycle 11

Instruction status

Instructio	j	k	Issue	Read operan	Execu compl	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9	11	
DIVI F10	F0	F6	8			
ADD F6	F8	F2				

ADDD can't start because add unit is busy.

Functional unit status

Tim	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
8	Mult1	Yes	Mult	F0	F2	F4			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 operan	Execu compl	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				

SUBD finishes.
DIVD waiting for F0.

Functional unit status

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

Scoreboard example – cycle 13

Instruction status

Instruction		j	k	Issue	Read operan	Execu compl	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			

ADDD issues.

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
6	Mult1	Yes	Mult	F0	F2	F4			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 operan	Execu compl	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

Tim	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU fo Qj	FU for 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			No	No
	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 16

Instruction status

Instruction		j	k	Issue	Read operan	Execu compl	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

Tim	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
3	Mult1	Yes	Mult	F0	F2	F4			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

This is fixable! (anti-dependance)

ADDD can't write because of DIVD WAR!

Instruction status

Instruction		j	k	Issue	Read operan	Execu compl	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

Tim	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU fo Qj	FU for 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			

Scoreboard example – cycle 19

Instruction status

Instruction		j	k	Issue	Read operand	Execution complete	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	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	

MULT completes execution.

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for 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		j	k	Issue	operat	compl	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	

MULT writes.

Functional unit status

Tim	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU fo	FU for Fj?	Fk?	
	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		j	k	Issue	operan	compl	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	

DIVD loads operands

Functional unit status

Tim	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU fo	FU for 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			No	No

Register result status

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

Scoreboard example – cycle 22

Instruction		j	k	Issue	operat	compl	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

Now ADDD can write since WAR removed.

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	Yes	Div	F10	F0	F6			No	No

Register result status

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

Scoreboard example – cycle 61

Instruction		j	k	Issue	operat	compl	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

DIVD completes execution

Functional unit status

Tim	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU fo	FU for Fj? Qj	Qk	Fk? Rj	Rk
	Integer	No									
	Mult1	No									
	Mult2	No									
	Add	No									
	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 operand	Executi comple	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTI	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8	21	61	62
ADDD	F6	F8	F2	13	14	16	22

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
0	Divide	No								

Register result status

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

Scoreboard Summary

- Dynamically schedules instructions
- Forces instructions to wait on RAW, WAR, WAW dependences and structural hazards
- First used in the CDC 6600 in 1964 and yielded performance improvements of 1.7x to 2.5x
- Hardware cost (size) of scoreboard equivalent to one of the functional units

Scoreboard Limitations

- No forwarding – read from register
  - Mitigating by a shorter "back end" of the pipeline (i.e., few stages after issue) and out-of-order execution
- Structural hazards – stall at issue
- WAW hazard – stall at issue
- WAR hazard – stall at write

Next lecture: Dynamic scheduling using Tomasulo's algorithm

- Avoids WAW & WAR via register renaming
- Supports forwarding using a centralized result bus