CS530 – ILP & Pipelining

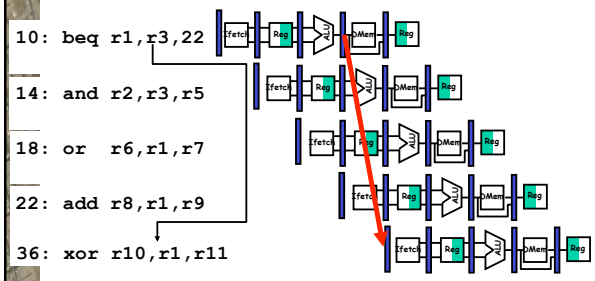
Gregory D. Peterson

ILP & Pipelining
10/14/14

Control Hazard on Branches

Three Stage Stall

```
10: beq r1,r3,22
14: and r2,r3,r5
18: or  r6,r1,r7
22: add r8,r1,r9
36: xor r10,r1,r11
```



What do you do with the 3 instructions in between?
How do you do it?
Where is the "commit"?

Branch Stall Impact

- If CPI = 1, 30% branch,
  Stall 3 cycles => new CPI = 1.9!
- Two part solution:
  - Determine branch taken or not sooner, AND
  - Compute taken branch address earlier
- MIPS branch tests if register = 0 or ≠ 0
- MIPS Solution:
  - Move Zero test to ID/RF stage
  - Adder to calculate new PC in ID/RF stage
  - 1 clock cycle penalty for branch versus 3

Four Branch Hazard Alternatives

#1: Stall until branch direction is clear

#2: Predict Branch Not Taken
- Execute successor instructions in sequence
- "Squash" instructions in pipeline if branch actually taken
- Advantage of late pipeline state update
- 47% MIPS branches not taken on average
- PC+4 already calculated, so use it to get next instruction

#3: Predict Branch Taken
- 53% MIPS branches taken on average
- But haven't calculated branch target address in MIPS
  - MIPS still incurs 1 cycle branch penalty
  - Other machines: branch target known before outcome

Four Branch Hazard Alternatives

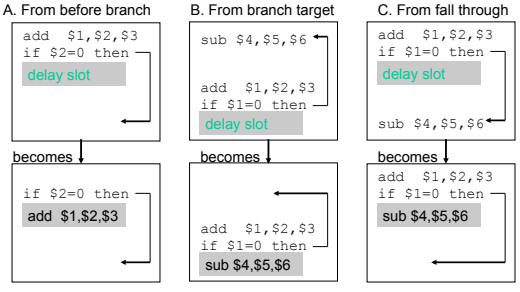
#4: Delayed Branch
- Define branch to take place AFTER a following instruction

```
branch instruction
 sequential successor1
 sequential successor2
 ........
 sequential successorn
branch target if taken
```

Branch delay of length *n*

- 1 slot delay allows proper decision and branch target address in 5 stage pipeline
- MIPS uses this

Scheduling Branch Delay Slots



- A is the best choice, fills delay slot & reduces instruction count (IC)
- In B, the `sub` instruction may need to be copied, increasing IC
- In B and C, must be okay to execute `sub` when branch fails

Delayed Branch

- Compiler effectiveness for single branch delay slot:
  - Fills about 60% of branch delay slots
  - About 80% of instructions executed in branch delay slots useful in computation
  - About 50% (60% x 80%) of slots usefully filled
- Delayed Branch downside: As processor go to deeper pipelines and multiple issue, the branch delay grows and need more than one delay slot
  - Delayed branching has lost popularity compared to more expensive but more flexible dynamic approaches
  - Growth in available transistors has made dynamic approaches relatively cheaper

Evaluating Branch Alternatives

$$\text{Pipeline speedup} = \frac{\text{Pipeline depth}}{1 + \text{Branch frequency} \times \text{Branch penalty}}$$

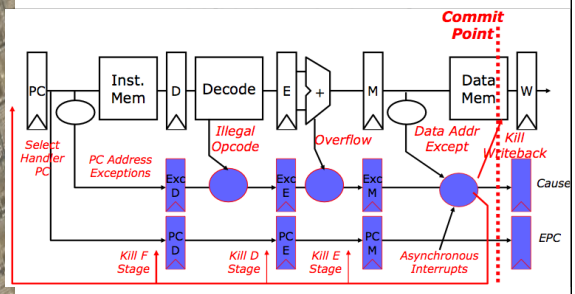
Assume 4% unconditional branch, 6% conditional branch-untaken, 10% conditional branch-taken

Scheduling scheme	*Branch penalty*	*CPI*	*speedup v. unpipelined*	*speedup v. stall*
Stall pipeline	3	1.60	3.1	1.0
Predict taken	1	1.20	4.2	1.33
Predict not taken	1	1.14	4.4	1.40
Delayed branch	0.5	1.10	4.5	1.45

Problems with Pipelining

- Exception: An unusual event happens to an instruction during its execution
  - Examples: divide by zero, undefined opcode
- Interrupt: Hardware signal to switch the processor to a new instruction stream
  - Example: a sound card interrupts when it needs more audio output samples (an audio "click" happens if it is left waiting)
- Problem: It must appear that the exception or interrupt must appear between 2 instructions (I_i and I_{i+1})
  - The effect of all instructions up to and including I_i is totalling complete
  - No effect of any instruction after I_i can take place
- The interrupt (exception) handler either aborts program or restarts at instruction I_{i+1}

Precise Exceptions in Static Pipelines



Key observation: architected state only changes in memory and register write stages.

What Makes Pipelining Hard?

Interrupts cause great havoc!

Examples of interrupts:
- Power failing,
- Arithmetic overflow,
- I/O device request,
- OS call,
- Page fault

Interrupts (also known as: faults, exceptions, traps) often require
- surprise jump (to vectored address)
- linking return address
- saving of PSW (including CCs)
- state change (e.g., to kernel mode)

There are 5 instructions executing in 5 stage pipeline when an interrupt occurs:
- **How to stop the pipeline?**
- **How to restart the pipeline?**
- **Who caused the interrupt?**

What Makes Pipelining Hard?

Interrupts cause great havoc!

What happens on interrupt while in delay slot ?
- **Next instruction is not sequential**

solution #1: save multiple PCs
- **Save current and next PC**
- **Special return sequence, more complex hardware**

solution #2: single PC plus
- **Branch delay bit**
- **PC points to branch instruction**

Stage	*Problem that causes the interrupt*
IF	Page fault on instruction fetch; misaligned memory access; memory-protection violation
ID	Undefined or illegal opcode
EX	Arithmetic interrupt
MEM	Page fault on data fetch; misaligned memory access; memory-protection violation

Multi-Cycle Operations

Floating Point

Floating point gives long execution time.

This causes a stall of the pipeline.

It's possible to pipeline the FP execution unit so it can initiate new instructions without waiting full latency. Can also have multiple FP units.

FP Instruction	*Latency*	*Initiation Interval*
Add, Subtract	3	1
Multiply	7	1
Divide	35	36
Square root	111	112
Negate	1	1
Absolute value	1	1
FP compare	2	2

13

Multi-Cycle Operations

Floating Point

Divide, Square Root take -10X to -30X longer than Add

- Interrupts?
- Adds WAR and WAW hazards since pipelines are no longer same length

	1	2	3	4	5	6	7	8	9	10	11
i	IF	ID	EX	MEM	WB						
I + 1		IF	ID	EX	EX	EX	EX	MEM	WB		
I + 2			IF	ID	EX	MEM	WB				
I + 3				IF	ID	EX	EX	EX	EX	MEM	WB
I + 4					IF	ID	EX	MEM	WB		
I + 5						IF	ID	--	--	EX	EX
I + 6							IF	--	--	ID	EX

Notes:
I + 2: no WAW, but this complicates an interrupt
I + 4: no WB conflict
I + 5: stall forced by structural hazard
I + 6: stall forced by in-order issue

14

Dynamic Hardware Prediction

- **Dynamic Branch Prediction is the ability of the hardware to make an educated guess about which way a branch will go - will the branch be taken or not.**
- **The hardware can look for clues based on the instructions, or it can use past history - we will discuss both of these directions.**
- **Key Concept: A Branch History Table contains information about what a branch did the last time it was executed.**

15

Dynamic Hardware Prediction

Basic Branch Prediction: Branch Prediction Buffers

Dynamic Branch Prediction

- Performance = *f*(accuracy, cost of misprediction)
- Branch History – The Lower bits of PC do an address of the index table of 1-bit values
  - Says whether or not branch taken last time
- Problem: in a loop, 1-bit BHT will cause two mis-predictions:
  - End of loop case, when it exits instead of looping as before
  - First time through loop on *next* time through code, when it predicts exit instead of looping

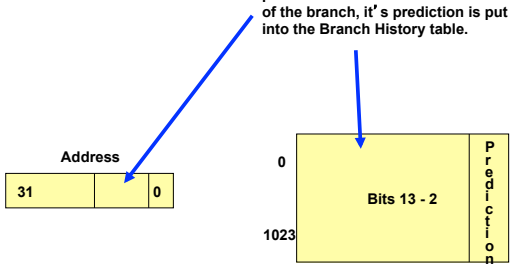
16

Dynamic Hardware Prediction

Basic Branch Prediction: Branch Prediction Buffers

- How Does It Work?

Bits 2 – 13 define 1024 different possibilities. Based on the address of the branch, it's prediction is put into the Branch History table.



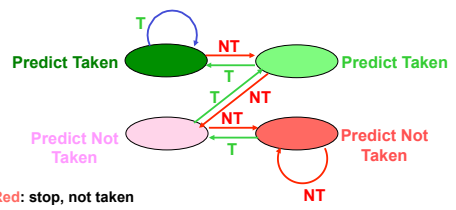
17

Dynamic Hardware Prediction

Basic Branch Prediction: Branch Prediction Buffers

Dynamic Branch Prediction

- Solution: 2-bit scheme where change prediction only if get misprediction *twice*



- Red: stop, not taken
- Green: go, taken
- Adds *hysteresis* to decision making process

18

Dynamic Hardware Prediction

Basic Branch Prediction: Branch Prediction Buffers

Branch History Table Accuracy

- Mispredict because either:
  - Wrong guess for that branch
  - Got branch history of wrong branch when index the table
- 4096 entry table programs vary from 1% misprediction (nasa7, tomcatv) to 18% (eqntott), with spice at 9% and gcc at 12%
- 4096 about as good as infinite table, but 4096 is a lot of HW

Dynamic Hardware Prediction

Basic Branch Prediction: Branch Prediction Buffers

Correlating Branches

What if we have the code:

```
If ( aa == 2)   aa = 0;
If ( bb == 2 )  bb = 0;
If ( aa != bb ) { …
```

Then the third "if" can be somewhat predicted based on the 1st two "ifs"

Generated MIPS Code:

```
        DSUBUI  R3, R1, #2
        BNEZ    R3, L1
        DADD    R1, R0, R0
L1:
        DSUBUI  R3, R2, #2
        BNEZ    R3, L2
        DADD    R2, R0, R0
L2:
        DSUBU   R3, R1, R2
        BEQZ    R3, L3
```

This branch is based on the Outcome of the previous 2 branches.

Advantages of Dynamic Scheduling

- **Handles cases when dependences unknown at compile time**
  - (e.g., because they may involve a memory reference)
- **It simplifies the compiler**
- **Allows code that compiled for one pipeline to run efficiently on a different pipeline**
- **Hardware speculation, a technique with significant performance advantages, that builds on dynamic scheduling**

Dynamic Scheduling

Logistics

- Chapter 3 of the text uses, as an example of Dynamic Scheduling, an algorithm due to Tomasulo.
- We first use another scoreboarding technique which is discussed in Appendix C

Dynamic Scheduling

The idea:

HW Schemes: Instruction Parallelism

- Why is this in Hardware at run time?
  - Works when can't know real dependence at compile time
  - Compiler simpler
  - Code for one machine runs well on another
- Key Idea: Allow instructions behind stall to proceed.
- Key Idea: Instructions executing in parallel. There are multiple execution units, so use them.

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

Even though ADDD stalls, the SUBD has no dependencies and can run.

  - Enables out-of-order execution => out-of-order completion

Dynamic Scheduling

The idea:

HW Schemes: Instruction Parallelism

- Out-of-order execution divides ID stage:
  1. Issue—decode instructions, check for structural hazards
  2. Read operands—wait until no data hazards, then read operands
- Scoreboards allow instruction to execute whenever 1 & 2 hold, not waiting for prior instructions.
- A scoreboard is a "data structure" that provides the information necessary for all pieces of the processor to work together.
- We will use *In order issue*, out of order execution, out of order commit (also called completion)
- First used in CDC6600. Our example modified here for MIPS.
- CDC had 4 FP units, 5 memory reference units, 7 integer units.
- MIPS has 2 FP multiply, 1 FP adder, 1 FP divider, 1 integer.

Dynamic Scheduling

Using A Scoreboard

Scoreboard Implications

- Out-of-order completion => WAR, WAW hazards?
- Solutions for WAR
  - **Queue both the operation and copies of its operands**
  - **Read registers only during Read Operands stage**
- For WAW, must detect hazard: stall until other completes
- Need to have multiple instructions in execution phase => multiple execution units or pipelined execution units
- Scoreboard keeps track of dependencies, state or operations
- Scoreboard replaces ID, EX, WB with 4 stages

Dynamic Scheduling

Using A Scoreboard

Four Stages of Scoreboard Control

1. Issue —decode instructions & check for structural hazards (ID1)

   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 WAW hazard exists, then the instruction issue stalls, and no further instructions will issue until these hazards are cleared.

Dynamic Scheduling

Using A Scoreboard

Four Stages of Scoreboard Control

2. Read operands —wait until no data hazards, then read operands (ID2)

   A source operand is available if no earlier issued active instruction is going to write it, or if the register containing the operand is being written by a currently active functional unit.

   When the source operands are available, the scoreboard tells the functional unit to proceed to read the operands from the registers and begin execution. The scoreboard resolves RAW hazards dynamically in this step, and instructions may be sent into execution out of order.

Dynamic Scheduling

Using A Scoreboard

Four Stages of Scoreboard Control

3. Execution —operate on operands (EX)

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

4. Write result —finish execution (WB)

   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.

   Example:

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

   Scoreboard would stall SUBD until ADDD reads operands

Dynamic Scheduling

Using A Scoreboard

Three Parts of the Scoreboard

1. Instruction status—which of 4 steps the instruction is in

2. Functional unit status—Indicates the state of the functional unit (FU). 9 fields for each functional unit
   - Busy—Indicates whether the unit is busy or not
   - Op—Operation to perform in the unit (e.g., + or –)
   - Fi—Destination register
   - Fj, Fk—Source-register numbers
   - Qj, Qk—Functional units producing source registers Fj, Fk
   - Rj, Rk—Flags indicating when Fj, Fk are ready

3. Register result status—Indicates which functional unit will write each register, if one exists. Blank when no pending instructions will write that register

Dynamic Scheduling

Using A Scoreboard

Detailed Scoreboard 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

Dynamic Scheduling: Examples

Now we look at an example of how Dynamic Scheduling actually works.

It's all about accounting!

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example

This is the sample code we'll be working with in the example:

```
LD      F6, 34(R2)
LD      F2, 45(R3)
MULT    F0, F2, F4
SUBD    F8, F6, F2
DIVD    F10, F0, F6
ADDD    F6, F8, F2
```

What are the hazards in this code?

Latencies (clock cycles):	
LD	1
MULT	10
SUBD	2
DIVD	40
ADDD	2

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example

Instruction status

Instruction	j	k	Issue	Read operan	Executi comple	Write Result
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	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								
	Divide	No								

Register result status

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

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 1

Instruction status

Instruction	j	k	Issue	Read operan	Executi comple	Write Result
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				

Issue LD #1

Shows in which cycle the operation occurred.

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

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 2

Instruction status

Instruction	j	k	Issue	Read operan	Executi comple	Write Result
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				

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

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 3

Instruction status

Instruction	j	k	Issue	Read operan	Executi comple	Write Result
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	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	F0	F2	F4	F6	F8	F10	F12	...	F30
3 FU				Integer					

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 4

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
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	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	F0	F2	F4	F6	F8	F10	F12	...	F30
4 FU				Integer					

37

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 5

Issue LD #2 since integer unit is now free.

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
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	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							

38

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 6

Issue MULT.

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
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	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	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							

39

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 7

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

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
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	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	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				

40

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 8a

DIVD issues. MULT and SUBD both waiting for F2.

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
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	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	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			

41

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 8b

LD #2 writes F2.

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
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	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			

42

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 9

Now MULT and SUBD can both read F2.
How can both instructions do this at the same time??

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
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	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			Yes	Yes
	Mult2	No								
2	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
9	FU Mult1				Add	Divide			

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 11

ADDD can't start because add unit is busy.

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
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	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			Yes	Yes
	Mult2	No								
0	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
11	FU Mult1				Add	Divide			

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 12

SUBD finishes.
DIVD waiting for F0.

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
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	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
7	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	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			

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 13

ADDD issues.

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
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	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			Yes	Yes
	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			

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 14

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
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	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? 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	Mult1		No	Yes

Register result status

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

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 15

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
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	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? 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	Mult1		No	Yes

Register result status

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

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 16

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
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	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			Yes	Yes
	Mult2	No								
0	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
16	FU	Mult1				Add		Divide	

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 17

ADDD can't write because of DIVD. RAW!

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
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	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? 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	Mult1		No	Yes

Register result status

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

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 18

Nothing Happens!!

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
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	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? 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	Mult1		No	Yes

Register result status

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

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 19

MULT completes execution.

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
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	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? 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	Mult1		No	Yes

Register result status

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

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 20

MULT writes.

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
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	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	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status

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

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 21

DIVD loads operands

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
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	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	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status

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

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 22

Now ADDD can write since WAR removed.

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
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	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								
40	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status

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

55

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 61

DIVD completes execution

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
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	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	Yes	Div	F10	F0	F6			Yes	Yes

Register result status

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

56

Dynamic Scheduling

Using A Scoreboard

Scoreboard Example Cycle 62

DONE!!

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
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	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								

57