

Lecture 15: Instruction Level Parallelism and Dynamic Execution

March 11, 2002
Prof. David E. Culler
Computer Science 252
Spring 2002

3/12/02

CS252/Culler
Lec. 15.1

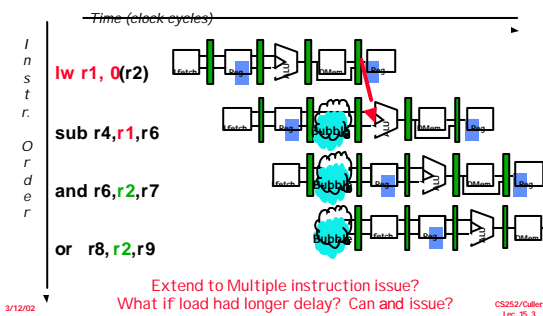
Recall from Pipelining Review

- Pipeline CPI = Ideal pipeline CPI + Structural Stalls + Data Hazard Stalls + Control Stalls
 - **Ideal pipeline CPI**: measure of the maximum performance attainable by the implementation
 - **Structural hazards**: HW cannot support this combination of instructions
 - **Data hazards**: Instruction depends on result of prior instruction still in the pipeline
 - **Control hazards**: Caused by delay between the fetching of instructions and decisions about changes in control flow (branches and jumps)

3/12/02

CS252/Culler
Lec. 15.2

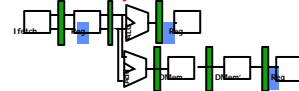
Recall Data Hazard Resolution: In-order issue, in-order completion



3/12/02

CS252/Culler
Lec. 15.3

In-Order Issue, Out-of-order Completion



- Which hazards are present? RAW? WAR? WAW?
- load $r3 \leftarrow r1, r2$
- add $r1 \leftarrow r5, r2$
- sub $r3 \leftarrow r3, r1$ or $r3 \leftarrow r2, r1$
- Register Reservations
 - when issue mark destination register busy till complete
 - check all register reservations before issue

3/12/02

CS252/Culler
Lec. 15.4

Ideas to Reduce Stalls

Technique	Reduces
Dynamic scheduling	Data hazard stalls
Dynamic branch prediction	Control stalls
Issuing multiple instructions per cycle	Ideal CPI
Speculation	Data and control stalls
Dynamic memory disambiguation	Data hazard stalls involving memory
Loop unrolling	Control hazard stalls
Basic compiler pipeline scheduling	Data hazard stalls
Compiler dependence analysis	Ideal CPI and data hazard stalls
Software pipelining and trace scheduling	Ideal CPI and data hazard stalls
Compiler speculation	Ideal CPI, data and control stalls

Chapter 3 (red arrow pointing to Dynamic scheduling, Dynamic branch prediction, Issuing multiple instructions per cycle)

Chapter 4 (blue arrow pointing to Speculation, Dynamic memory disambiguation, Loop unrolling, Basic compiler pipeline scheduling, Compiler dependence analysis, Software pipelining and trace scheduling, Compiler speculation)

3/12/02

CS252/Culler
Lec. 15.5

Instruction-Level Parallelism (ILP)

- Basic Block (BB) ILP is quite small
 - BB: a straight-line code sequence with no branches in except to the entry and no branches out except at the exit
 - average dynamic branch frequency 15% to 25%
=> 4 to 7 instructions execute between a pair of branches
 - Plus instructions in BB likely to depend on each other
- To obtain substantial performance enhancements, we must exploit ILP across multiple basic blocks
- Simplest: loop-level parallelism to exploit parallelism among iterations of a loop
 - Vector is one way
 - If not vector, then either dynamic via branch prediction or static via loop unrolling by compiler

3/12/02

CS252/Culler
Lec. 15.6

Data Dependence and Hazards

- Instr_j is **data dependent** on Instr_i , Instr_j tries to read operand before Instr_i writes it

```

I: add r1,r2,r3
  J: sub r4,r1,r3
  
```

- or Instr_j is data dependent on Instr_k which is dependent on Instr_i
- Caused by a “**True Dependence**” (compiler term)
- If true dependence caused a hazard in the pipeline, called a **Read After Write (RAW) hazard**

3/12/02

CS252/Caller
Lec. 15.7

Data Dependence and Hazards

- Dependences are a property of **programs**
- Presence of dependence indicates **potential** for a hazard, but actual hazard and length of any stall is a property of the **pipeline**
- Importance of the data dependencies
 - 1) indicates the possibility of a hazard
 - 2) determines order in which results must be calculated
 - 3) sets an upper bound on how much parallelism can possibly be exploited
- Today looking at HW schemes to avoid hazard

3/12/02

CS252/Caller
Lec. 15.8

Name Dependence #1: Anti-dependence

- **Name dependence**: when 2 instructions use same register or memory location, called a **name**, but no flow of data between the instructions associated with that name; 2 versions of name dependence

- Instr_j writes operand **before** Instr_i reads it

```

  I: sub r4,r1,r3
  J: add r1,r2,r3
  K: mul r6,r1,r7
  
```

Called an “**anti-dependence**” in compiler work. This results from reuse of the name “**r1**”

- If anti-dependence caused a hazard in the pipeline, called a **Write After Read (WAR) hazard**

3/12/02

CS252/Caller
Lec. 15.9

Name Dependence #2: Output dependence

- Instr_j writes operand **before** Instr_i writes it.

```

  I: sub r1,r4,r3
  J: add r1,r2,r3
  K: mul r6,r1,r7
  
```

- Called an “**output dependence**” by compiler writers. This also results from the reuse of name “**r1**”
- If anti-dependence caused a hazard in the pipeline, called a **Write After Write (WAW) hazard**

3/12/02

CS252/Caller
Lec. 15.10

ILP and Data Hazards

- **program order**: order instructions would execute in if executed sequentially 1 at a time as determined by original source program
- HW/SW goal: exploit parallelism by preserving appearance of program order
 - modify order in manner than cannot be observed by program
 - must not affect the outcome of the program
- Ex: Instructions involved in a name dependence can execute simultaneously **if name used in instructions is changed** so instructions do not conflict
 - Register renaming resolves name dependence for regs
 - Either by compiler or by HW
 - add r1, r2, r3
 - sub r2, r4, r5
 - and r3, r2, 1

3/12/02

CS252/Caller
Lec. 15.11

Control Dependencies

- Every instruction is control dependent on some set of branches, and, in general, these control dependencies must be preserved to preserve program order

```

if p1 {
  S1;
};
if p2 {
  S2;
}
  
```

- S1 is control dependent on p1, and S2 is control dependent on p2 but not on p1.

3/12/02

CS252/Caller
Lec. 15.12

Control Dependence Ignored

- Control dependence need not always be preserved
 - willing to execute instructions that should not have been executed, thereby violating the control dependences, if can do so without affecting correctness of the program
- Instead, 2 properties critical to program correctness are **exception behavior** and **data flow**

3/12/02

CS252/Oaller
Lec. 15.13

Exception Behavior

- Preserving exception behavior => any changes in instruction execution order must not change how exceptions are raised in program (=> no new exceptions)
- Example:

```
DADDU    R2, R3, R4
BEQZ     R2, L1
LW       R1, 0(R2)
L1:
```
- Problem with moving LW before BEQZ?

3/12/02

CS252/Oaller
Lec. 15.14

Data Flow

- Data flow**: actual flow of data values among instructions that produce results and those that consume them
 - branches make flow dynamic, determine which instruction is supplier of data
- Example:

```
DADDU    R1, R2, R3
BEQZ     R4, L
DSUBU    R1, R5, R6
L:  ...
OR       R7, R1, R8
```
- OR depends on DADDU or DSUBU?
Must preserve data flow on execution

3/12/02

CS252/Oaller
Lec. 15.15

CS 252 Administrivia

- Final Project Proposals due 3/17
 - send URL to page containing
 - title & participants
 - problem statement
 - annotated bibliography
 - we'll monitor progress through the pages
- Assignment 3 out, due in 3/19
- Quiz 3/21

3/12/02

CS252/Oaller
Lec. 15.16

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

3/12/02

CS252/Oaller
Lec. 15.17

HW Schemes: Instruction Parallelism

- Key idea: Allow instructions behind stall to proceed

```
DIVD    F0, F2, F4
ADD     F10, F0, F8
SUBD   F12, F8, F14
```
- Enables **out-of-order execution** and allows **out-of-order completion**
- Will distinguish when an instruction **begins execution** and when it **completes execution**; between 2 times, the instruction is **in execution**
- In a dynamically scheduled pipeline, all instructions pass through issue stage in order (**in-order issue**)

3/12/02

CS252/Oaller
Lec. 15.18

Dynamic Scheduling Step 1

- Simple pipeline had 1 stage to check both structural and data hazards: Instruction Decode (ID), also called Instruction Issue
- Split the ID pipe stage of simple 5-stage pipeline into 2 stages:
 - **Issue**—Decode instructions, check for structural hazards
 - **Read operands**—Wait until no data hazards, then read operands

3/12/02

CS252/Osher
Lec. 15.19

A Dynamic Algorithm: Tomasulo's Algorithm

- For IBM 360/91 (before caches!)
- Goal: High Performance without special compilers
- Small number of floating point registers (4 in 360) prevented interesting compiler scheduling of operations
 - This led Tomasulo to try to figure out how to get more effective registers — **renaming in hardware!**
- Why Study 1966 Computer?
- The descendants of this have flourished!
 - Alpha 21264, HP 8000, MIPS 10000, Pentium III, PowerPC 604, ...

3/12/02

CS252/Osher
Lec. 15.20

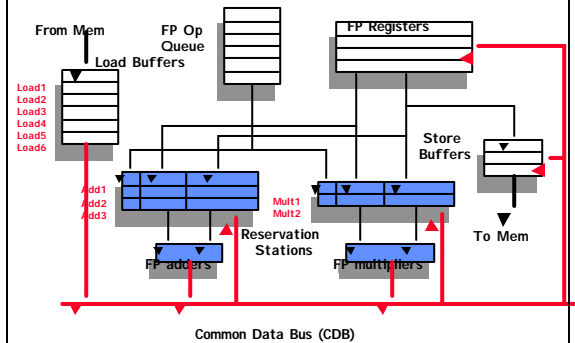
Tomasulo Algorithm

- Control & buffers **distributed** with Function Units (FU)
 - FU buffers called "**reservation stations**"; have pending operands
- Registers in instructions replaced by values or pointers to reservation stations(RS);
 - form of **register renaming** ;
 - avoids WAR, WAW hazards
 - More reservation stations than registers, so can do optimizations compilers can't
- Results to FU from RS, **not through registers**, over **Common Data Bus** that broadcasts results to all FUs
- Load and Stores treated as FUs with RSs as well
- Integer instructions can go past branches, allowing FP ops beyond basic block in FP queue

3/12/02

CS252/Osher
Lec. 15.21

Tomasulo Organization



3/12/02

CS252/Osher
Lec. 15.22

Reservation Station Components

Op: Operation to perform in the unit (e.g., + or -)

Vj, Vk: **Value** of Source operands

- Store buffers has V field, result to be stored

Qj, Qk: Reservation stations producing source registers (value to be written)

- Note: Qj, Qk=0 => ready
- Store buffers only have Qi for RS producing result

Busy: Indicates reservation station or FU is busy

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

3/12/02

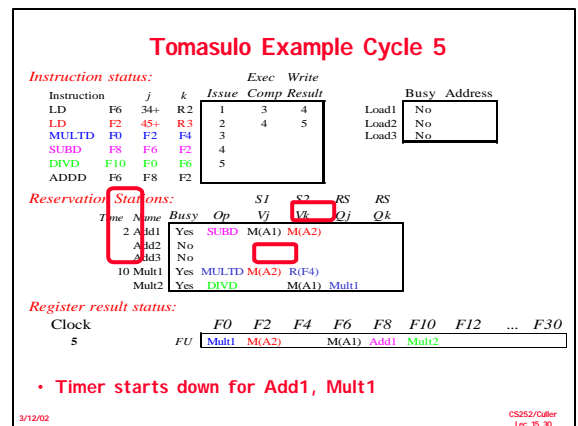
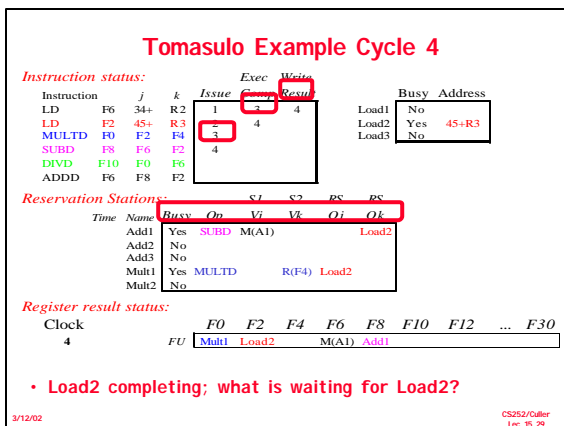
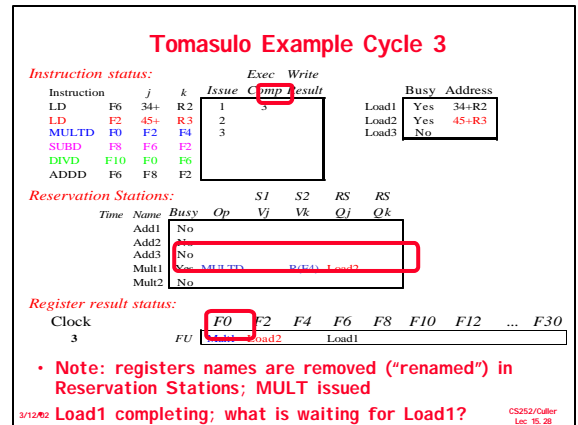
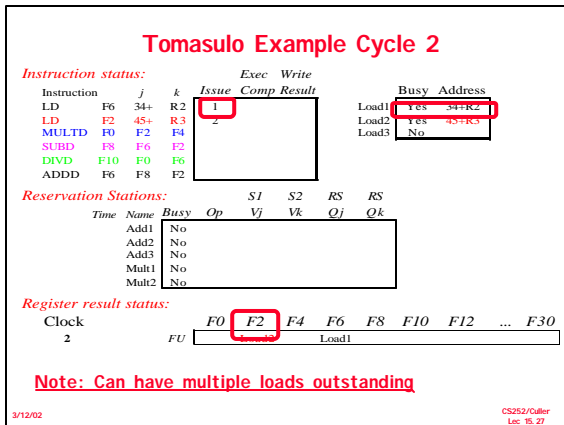
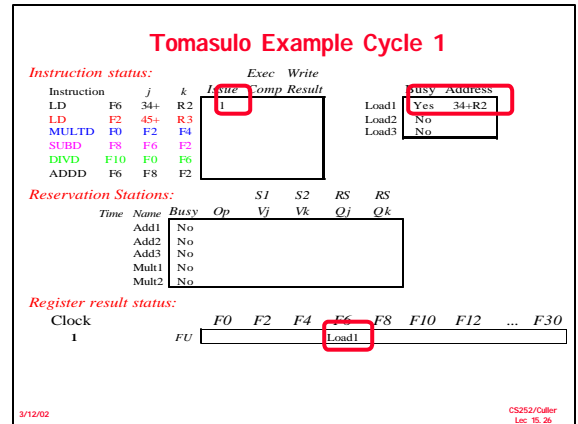
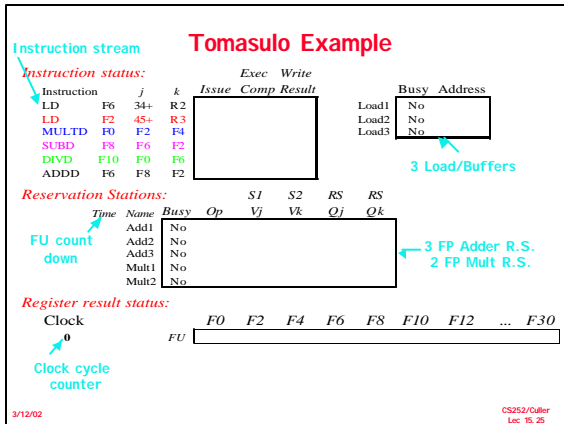
CS252/Osher
Lec. 15.23

Three Stages of Tomasulo Algorithm

1. **Issue**—get instruction from FP Op Queue
 - If reservation station free (no structural hazard), control issues instr & sends operands (renames registers).
 2. **Execute**—operate on operands (EX)
 - When both operands ready then execute;
 - if not ready, watch Common Data Bus for result
 3. **Write result**—finish execution (WB)
 - Write on Common Data Bus to all awaiting units;
 - mark reservation station available
- **Normal data bus**: data + destination ("go to" bus)
 - **Common data bus**: data + source, ("come from" bus)
 - 64 bits of data + 4 bits of Functional Unit **source** address
 - Write if matches expected Functional Unit (produces result)
 - Does the broadcast
 - **Example speed**:
3 clocks for FI .pt. +, -; 10 for * ; 40 clks for /

3/12/02

CS252/Osher
Lec. 15.24



Tomasulo Example Cycle 6

Instruction status:

Instruction	j	k	Issue	Exec	Write	Comp	Result	Busy	Address
LD	F6	34+	R2	1	3	4		Load1	No
LD	F2	45+	R3	2	4	5		Load2	No
MULTD	F0	F2	F4	3				Load3	No
SUBD	F8	F6	F2	4					
DIVD	F10	F0	F6	5					
ADDD	F6	F8	F2	6					

Reservation Stations:

Time	Name	Busy	Op	S1	S2	RS	RS
				Vj	Vk	Qj	Qk
1	Add1	No	ADD	M(A1)	M(A2)	Add1	
2	Add2	Yes	ADDD	M(A2)	M(A2)	Add1	
3	Add3	No					
9	Multi1	Yes	MULTD	M(A2)	R(F4)		
	Multi2	Yes	DIVD	M(A1)	Multi1		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
6	FU	Multi1	M(A2)	Add2	Add1	Multi2			

• Issue ADDD here despite name dependency on F6?

3/12/02

CS252/Osher
Lec. 15.31

Tomasulo Example Cycle 7

Instruction status:

Instruction	j	k	Issue	Exec	Write	Comp	Result	Busy	Address
LD	F6	34+	R2	1	3	4		Load1	No
LD	F2	45+	R3	2	4	5		Load2	No
MULTD	F0	F2	F4	3				Load3	No
SUBD	F8	F6	F2	4					
DIVD	F10	F0	F6	5					
ADDD	F6	F8	F2	6					

Reservation Stations:

Time	Name	Busy	Op	S1	S2	RS	RS
				Vj	Vk	Qj	Qk
0	Add1	Yes	SUBD	M(A1)	M(A2)		
1	Add2	Yes	ADDD	M(A2)	M(A2)	Add1	
2	Add3	No					
8	Multi1	Yes	MULTD	M(A2)	R(F4)		
	Multi2	Yes	DIVD	M(A1)	Multi1		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
7	FU	Multi1	M(A2)	Add2	Add1	Multi2			

• Add1 (SUBD) completing; what is waiting for it?

3/12/02

CS252/Osher
Lec. 15.32

Tomasulo Example Cycle 8

Instruction status:

Instruction	j	k	Issue	Exec	Write	Comp	Result	Busy	Address
LD	F6	34+	R2	1	3	4		Load1	No
LD	F2	45+	R3	2	4	5		Load2	No
MULTD	F0	F2	F4	3				Load3	No
SUBD	F8	F6	F2	4	7	8			
DIVD	F10	F0	F6	5					
ADDD	F6	F8	F2	6					

Reservation Stations:

Time	Name	Busy	Op	S1	S2	RS	RS
				Vj	Vk	Qj	Qk
	Add1	No					
2	Add2	Yes	ADDD	M(A1)	M(A2)		
3	Add3	No					
7	Multi1	Yes	MULTD	M(A2)	R(F4)		
	Multi2	Yes	DIVD	M(A1)	Multi1		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
8	FU	Multi1	M(A2)	Add2	Add1	Multi2			

3/12/02

CS252/Osher
Lec. 15.33

Tomasulo Example Cycle 9

Instruction status:

Instruction	j	k	Issue	Exec	Write	Comp	Result	Busy	Address
LD	F6	34+	R2	1	3	4		Load1	No
LD	F2	45+	R3	2	4	5		Load2	No
MULTD	F0	F2	F4	3				Load3	No
SUBD	F8	F6	F2	4	7	8			
DIVD	F10	F0	F6	5					
ADDD	F6	F8	F2	6					

Reservation Stations:

Time	Name	Busy	Op	S1	S2	RS	RS
				Vj	Vk	Qj	Qk
	Add1	No					
1	Add2	Yes	ADDD	(M-M)	M(A2)		
3	Add3	No					
6	Multi1	Yes	MULTD	M(A2)	R(F4)		
	Multi2	Yes	DIVD	M(A1)	Multi1		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
9	FU	Multi1	M(A2)	Add2	(M-M)	Multi2			

3/12/02

CS252/Osher
Lec. 15.34

Tomasulo Example Cycle 10

Instruction status:

Instruction	j	k	Issue	Exec	Write	Comp	Result	Busy	Address
LD	F6	34+	R2	1	3	4		Load1	No
LD	F2	45+	R3	2	4	5		Load2	No
MULTD	F0	F2	F4	3				Load3	No
SUBD	F8	F6	F2	4	7	8			
DIVD	F10	F0	F6	5					
ADDD	F6	F8	F2	6	10				

Reservation Stations:

Time	Name	Busy	Op	S1	S2	RS	RS
				Vj	Vk	Qj	Qk
	Add1	No					
0	Add2	Yes	ADDD	(M-M)	M(A2)		
3	Add3	No					
5	Multi1	Yes	MULTD	M(A2)	R(F4)		
	Multi2	Yes	DIVD	M(A1)	Multi1		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
10	FU	Multi1	M(A2)	Add2	(M-M)	Multi2			

• Add2 (ADDD) completing; what is waiting for it?

3/12/02

CS252/Osher
Lec. 15.35

Tomasulo Example Cycle 11

Instruction status:

Instruction	j	k	Issue	Exec	Write	Comp	Result	Busy	Address
LD	F6	34+	R2	1	3	4		Load1	No
LD	F2	45+	R3	2	4	5		Load2	No
MULTD	F0	F2	F4	3				Load3	No
SUBD	F8	F6	F2	4	7	8			
DIVD	F10	F0	F6	5					
ADDD	F6	F8	F2	6	10	11			

Reservation Stations:

Time	Name	Busy	Op	S1	S2	RS	RS
				Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
4	Multi1	Yes	MULTD	M(A2)	R(F4)		
	Multi2	Yes	DIVD	M(A1)	Multi1		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
11	FU	Multi1	M(A2)	(M-M)	(M-M)	Multi2			

• Write result of ADDD here?
• All quick instructions complete in this cycle!

3/12/02

CS252/Osher
Lec. 15.36

Tomasulo Example Cycle 12

Instruction status:

Instruction	j	k	Exec			Write	Busy	Address
			Issue	Comp	Result			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
	3 Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD	M(A1)	Mult1		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
12	FU	Mult1	M(A2)	(M-M)+N	(M-M)	Mult2			

3/12/02

CS252/Oller
Lec. 15.27

Tomasulo Example Cycle 13

Instruction status:

Instruction	j	k	Exec			Write	Busy	Address
			Issue	Comp	Result			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
	2 Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD	M(A1)	Mult1		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
13	FU	Mult1	M(A2)	(M-M)+N	(M-M)	Mult2			

3/12/02

CS252/Oller
Lec. 15.28

Tomasulo Example Cycle 14

Instruction status:

Instruction	j	k	Exec			Write	Busy	Address
			Issue	Comp	Result			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
	1 Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD	M(A1)	Mult1		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
14	FU	Mult1	M(A2)	(M-M)+N	(M-M)	Mult2			

3/12/02

CS252/Oller
Lec. 15.29

Tomasulo Example Cycle 15

Instruction status:

Instruction	j	k	Exec			Write	Busy	Address
			Issue	Comp	Result			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3	15		Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
	0 Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD	M(A1)	Mult1		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
15	FU	Mult1	M(A2)	(M-M)+N	(M-M)	Mult2			

• Mult1 (MULTD) completing; what is waiting for it?

3/12/02

CS252/Oller
Lec. 15.40

Tomasulo Example Cycle 16

Instruction status:

Instruction	j	k	Exec			Write	Busy	Address
			Issue	Comp	Result			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3	15	16	Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	40 Mult2	Yes	DIVD	M*F4	M(A1)		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
16	FU	M*F4	M(A2)	(M-M)+N	(M-M)	Mult2			

• Just waiting for Mult2 (DIVD) to complete

3/12/02

CS252/Oller
Lec. 15.41

Faster than light computation
(skip a couple of cycles)

3/12/02

CS252/Oller
Lec. 15.42

Tomasulo Example Cycle 55

Instruction status:

Instruction	j	k	Exec Write			Load1	Load2	Load3	Busy	Address
			Issue	Comp	Result					
LD	F6	34+	R2	1	3	4	No	No	No	
LD	F2	45+	R3	2	4	5	No	No	No	
MULTD	F0	F2	F4	3	15	16	No	No	No	
SUBD	F8	F6	F2	4	7	8	No	No	No	
DIVD	F10	F0	F6	5	56	57	No	No	No	
ADDD	F6	F8	F2	6	10	11	No	No	No	

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk
Add1	No						
Add2	No						
Add3	No						
Multi1	No						
Multi2	Yes	DIVD	M*F4	M(A1)			

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
55	FU	M*F4	M(A2)	(M-M)+N	(M-M)	Multi2			

3/12/02

CS252/Osher Lec. 15.43

Tomasulo Example Cycle 56

Instruction status:

Instruction	j	k	Exec Write			Load1	Load2	Load3	Busy	Address
			Issue	Comp	Result					
LD	F6	34+	R2	1	3	4	No	No	No	
LD	F2	45+	R3	2	4	5	No	No	No	
MULTD	F0	F2	F4	3	15	16	No	No	No	
SUBD	F8	F6	F2	4	7	8	No	No	No	
DIVD	F10	F0	F6	5	56	57	No	No	No	
ADDD	F6	F8	F2	6	10	11	No	No	No	

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk
Add1	No						
Add2	No						
Add3	No						
Multi1	No						
Multi2	Yes	DIVD	M*F4	M(A1)			

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
56	FU	M*F4	M(A2)	(M-M)+N	(M-M)	Multi2			

3/12/02

CS252/Osher Lec. 15.44

- **Multi2 (DIVD) is completing; what is waiting for it?**

Tomasulo Example Cycle 57

Instruction status:

Instruction	j	k	Exec Write			Load1	Load2	Load3	Busy	Address
			Issue	Comp	Result					
LD	F6	34+	R2	1	3	4	No	No	No	
LD	F2	45+	R3	2	4	5	No	No	No	
MULTD	F0	F2	F4	3	15	16	No	No	No	
SUBD	F8	F6	F2	4	7	8	No	No	No	
DIVD	F10	F0	F6	5	56	57	No	No	No	
ADDD	F6	F8	F2	6	10	11	No	No	No	

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk
Add1	No						
Add2	No						
Add3	No						
Multi1	No						
Multi2	Yes	DIVD	M*F4	M(A1)			

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
56	FU	M*F4	M(A2)	(M-M)+N	(M-M)	Result			

3/12/02

CS252/Osher Lec. 15.45

- **Once again: In-order issue, out-of-order execution and out-of-order completion.**

Tomasulo Drawbacks

- **Complexity**
 - delays of 360/91, MIPS 10000, Alpha 21264, IBM PPC 620 in CA: AQA 2/e, but not in silicon!
- **Many associative stores (CDB) at high speed**
- **Performance limited by Common Data Bus**
 - Each CDB must go to multiple functional units
 - ▷ high capacitance, high wiring density
 - Number of functional units that can complete per cycle limited to one!
 - » Multiple CDBs ▷ more FU logic for parallel assoc stores
- **Non-precise interrupts!**
 - We will address this later

3/12/02

CS252/Osher Lec. 15.46

Tomasulo Loop Example

```

Loop: LD      F0  0  R1
      MULTD   F4  F0  F2
      SD      F4  0  R1
      SUBI    R1  R1  #8
      BNEZ   R1  Loop
    
```

- This time assume Multiply takes 4 clocks
- Assume 1st load takes 8 clocks (L1 cache miss), 2nd load takes 1 clock (hit)
- To be clear, will show clocks for SUBI, BNEZ
 - Reality: integer instructions ahead of FI. Pt. Instructions
- Show 2 iterations

3/12/02

CS252/Osher Lec. 15.47

Loop Example

Instruction status:

ITER	Instruction	j	k	Exec Write			Load1	Load2	Load3	Busy	Addr	Fu
				Issue	Comp	Result						
1	LD	F0	0	R1			No	No	No			
1	MULTD	F4	F0	F2			No	No	No			
1	SD	F4	0	R1			No	No	No			
2	LD	F0	0	R1			No	No	No			
2	MULTD	F4	F0	F2			No	No	No			
2	SD	F4	0	R1			No	No	No			

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
Add1	No							LD
Add2	No							MULTD
Add3	No							SD
Multi1	No							SUBI
Multi2	No							BNEZ

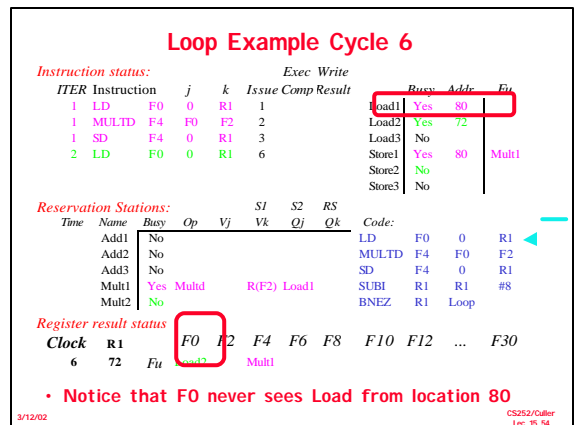
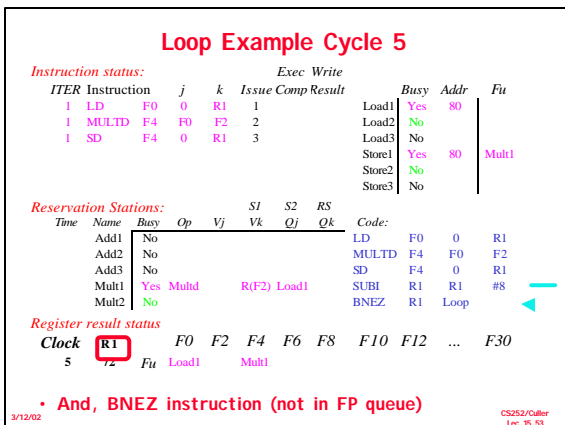
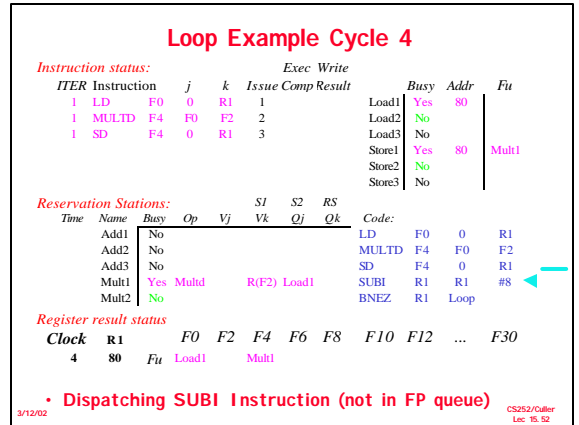
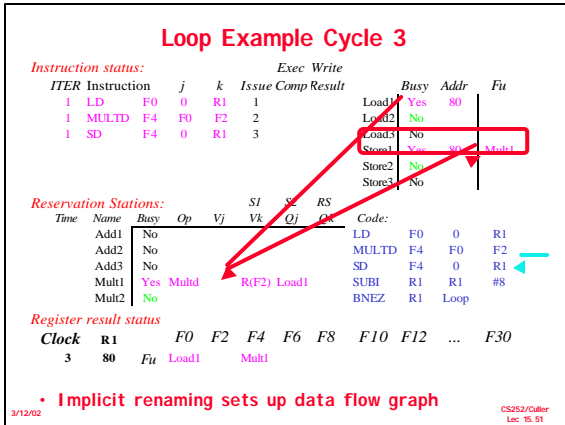
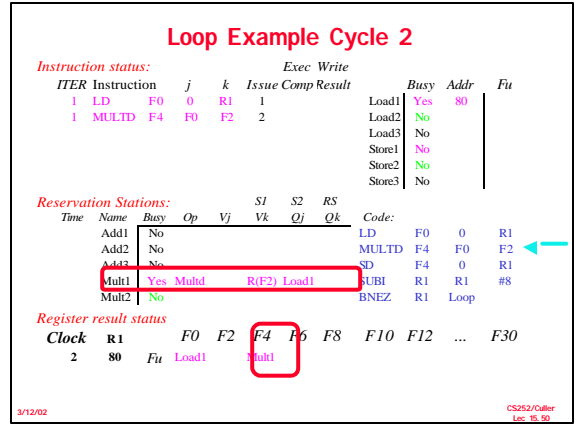
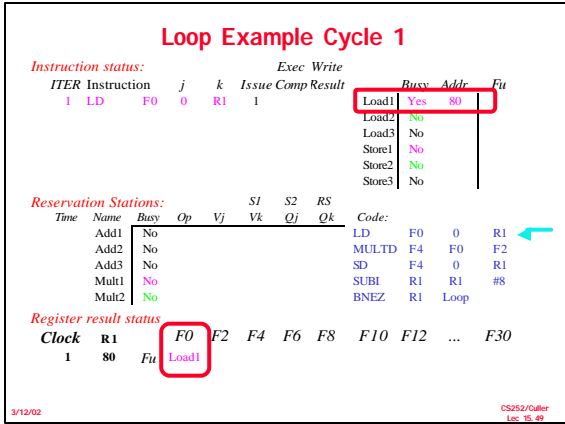
Register result status:

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
0	80									

3/12/02

CS252/Osher Lec. 15.48

Value of Register used for address, iteration control



Loop Example Cycle 7

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1	Load1	Yes	80	
1	MULTD	F4	F0	F2	2	Load2	Yes	72	
1	SD	F4	0	R1	3	Load3	No		
2	LD	F0	0	R1	6	Store1	Yes	80	Mult1
2	MULTD	F4	F0	F2	7	Store2	No		
					8	Store3	No		

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:	LD	F0	F2	F4	F6	F8	F10	F12	...	F30
	Add1	No							LD	F0	0	R1						
	Add2	No							MULTD	F4	F0	F2						
	Add3	No							SD	F4	0	R1						
	Mult1	Yes	Multd			R(F2)	Load1		SUBI	R1	R1	#8						
	Mult2	Yes	Multd			R(F2)	Load2		BNEZ	R1	Loop							

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
7	72	Fu	Load2	Mult2						

- Register file completely detached from computation
- First and Second iteration completely overlapped

Loop Example Cycle 8

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1	Load1	Yes	80	
1	MULTD	F4	F0	F2	2	Load2	Yes	72	
1	SD	F4	0	R1	3	Load3	No		
2	LD	F0	0	R1	6	Store1	Yes	80	Mult1
2	MULTD	F4	F0	F2	7	Store2	Yes	72	Mult2
2	SD	F4	0	R1	8	Store3	No		

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:	LD	F0	F2	F4	F6	F8	F10	F12	...	F30
	Add1	No							LD	F0	0	R1						
	Add2	No							MULTD	F4	F0	F2						
	Add3	No							SD	F4	0	R1						
	Mult1	Yes	Multd			R(F2)	Load1		SUBI	R1	R1	#8						
	Mult2	Yes	Multd			R(F2)	Load2		BNEZ	R1	Loop							

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
8	72	Fu	Load2	Mult2						

- Register file completely detached from computation
- First and Second iteration completely overlapped

Loop Example Cycle 9

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1	Load1	Yes	80	
1	MULTD	F4	F0	F2	2	Load2	Yes	72	
1	SD	F4	0	R1	3	Load3	No		
2	LD	F0	0	R1	6	Store1	Yes	80	Mult1
2	MULTD	F4	F0	F2	7	Store2	Yes	72	Mult2
2	SD	F4	0	R1	8	Store3	No		

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:	LD	F0	F2	F4	F6	F8	F10	F12	...	F30
	Add1	No							LD	F0	0	R1						
	Add2	No							MULTD	F4	F0	F2						
	Add3	No							SD	F4	0	R1						
	Mult1	Yes	Multd			R(F2)	Load1		SUBI	R1	R1	#8						
	Mult2	Yes	Multd			R(F2)	Load2		BNEZ	R1	Loop							

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
9	72	Fu	Load2	Mult2						

- Load1 completing: who is waiting?
- Note: Dispatching SUBI

Loop Example Cycle 10

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1	Load1	Yes	80	
1	MULTD	F4	F0	F2	2	Load2	Yes	72	
1	SD	F4	0	R1	3	Load3	No		
2	LD	F0	0	R1	6	Store1	Yes	80	Mult1
2	MULTD	F4	F0	F2	7	Store2	Yes	72	Mult2
2	SD	F4	0	R1	8	Store3	No		

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:	LD	F0	F2	F4	F6	F8	F10	F12	...	F30
	Add1	No							LD	F0	0	R1						
	Add2	No							MULTD	F4	F0	F2						
	Add3	No							SD	F4	0	R1						
4	Mult1	Yes	Multd			R(F2)	Load1		SUBI	R1	R1	#8						
	Mult2	Yes	Multd			R(F2)	Load2		BNEZ	R1	Loop							

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
10	64	Fu	Load2	Mult2						

- Load2 completing: who is waiting?
- Note: Dispatching BNEZ

Loop Example Cycle 11

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1	Load1	Yes	80	
1	MULTD	F4	F0	F2	2	Load2	No		
1	SD	F4	0	R1	3	Load3	Yes	64	
2	LD	F0	0	R1	6	Store1	Yes	80	Mult1
2	MULTD	F4	F0	F2	7	Store2	Yes	72	Mult2
2	SD	F4	0	R1	8	Store3	No		

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:	LD	F0	F2	F4	F6	F8	F10	F12	...	F30
	Add1	No							LD	F0	0	R1						
	Add2	No							MULTD	F4	F0	F2						
	Add3	No							SD	F4	0	R1						
3	Mult1	Yes	Multd			M[80]	R(F2)		SUBI	R1	R1	#8						
4	Mult2	Yes	Multd			M[72]	R(F2)		BNEZ	R1	Loop							

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
11	64	Fu	Load3	Mult2						

- Next load in sequence

Loop Example Cycle 12

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1	Load1	No		
1	MULTD	F4	F0	F2	2	Load2	No		
1	SD	F4	0	R1	3	Load3	Yes	64	
2	LD	F0	0	R1	6	Store1	Yes	80	Mult1
2	MULTD	F4	F0	F2	7	Store2	Yes	72	Mult2
2	SD	F4	0	R1	8	Store3	No		

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:	LD	F0	F2	F4	F6	F8	F10	F12	...	F30
	Add1	No							LD	F0	0	R1						
	Add2	No							MULTD	F4	F0	F2						
	Add3	No							SD	F4	0	R1						
2	Mult1	Yes	Multd			M[80]	R(F2)		SUBI	R1	R1	#8						
3	Mult2	Yes	Multd			M[72]	R(F2)		BNEZ	R1	Loop							

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
12	64	Fu	Load3	Mult2						

- Why not issue third multiply?

Loop Example Cycle 13

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1	9	10	Load1	No
1	MULTD	F4	F0	F2	2	14	15	Load2	No
1	SD	F4	0	R1	3			Load3	Yes 64
2	LD	F0	0	R1	6	10	11	Store1	Yes 80 Mult1
2	MULTD	F4	F0	F2	7			Store2	Yes 72 Mult2
2	SD	F4	0	R1	8			Store3	No

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:	LD	F0	0	R1
Add1	No							LD	F0	0	R1	←
Add2	No							MULTD	F4	F0	F2	←
Add3	No							SD	F4	0	R1	
1	Mult1	Yes	Multd	M[80]	R(F2)			SUBI	R1	R1	#8	
2	Mult2	Yes	Multd	M[72]	R(F2)			BNEZ	R1	Loop		

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
13	64	Fu	Load3	Mult2						

• Why not issue third store?

3/12/02

CS252/Osher
Lec. 15.43

Loop Example Cycle 14

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1	9	10	Load1	No
1	MULTD	F4	F0	F2	2	14	15	Load2	No
1	SD	F4	0	R1	3			Load3	Yes 64
2	LD	F0	0	R1	6	10	11	Store1	Yes 80 Mult1
2	MULTD	F4	F0	F2	7			Store2	Yes 72 Mult2
2	SD	F4	0	R1	8			Store3	No

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:	LD	F0	0	R1
Add1	No							LD	F0	0	R1	←
Add2	No							MULTD	F4	F0	F2	←
Add3	No							SD	F4	0	R1	
0	Mult1	Yes	Multd	M[80]	R(F2)			SUBI	R1	R1	#8	
1	Mult2	Yes	Multd	M[72]	R(F2)			BNEZ	R1	Loop		

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
14	64	Fu	Load3	Mult2						

• Mult1 completing. Who is waiting?

3/12/02

CS252/Osher
Lec. 15.42

Loop Example Cycle 15

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1	9	10	Load1	No
1	MULTD	F4	F0	F2	2	14	15	Load2	No
1	SD	F4	0	R1	3			Load3	Yes 64
2	LD	F0	0	R1	6	10	11	Store1	Yes 80 Mult1
2	MULTD	F4	F0	F2	7			Store2	Yes 72 Mult2
2	SD	F4	0	R1	8			Store3	No

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:	LD	F0	0	R1
Add1	No							LD	F0	0	R1	←
Add2	No							MULTD	F4	F0	F2	←
Add3	No							SD	F4	0	R1	
0	Mult1	No						SUBI	R1	R1	#8	
4	Mult2	Yes	Multd	M[72]	R(F2)			BNEZ	R1	Loop		

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
15	64	Fu	Load3	Mult2						

• Mult2 completing. Who is waiting?

3/12/02

CS252/Osher
Lec. 15.43

Loop Example Cycle 16

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1	9	10	Load1	No
1	MULTD	F4	F0	F2	2	14	15	Load2	No
1	SD	F4	0	R1	3			Load3	Yes 64
2	LD	F0	0	R1	6	10	11	Store1	Yes 80 Mult1
2	MULTD	F4	F0	F2	7	15	16	Store2	Yes 72 Mult2
2	SD	F4	0	R1	8			Store3	No

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:	LD	F0	0	R1
Add1	No							LD	F0	0	R1	←
Add2	No							MULTD	F4	F0	F2	←
Add3	No							SD	F4	0	R1	
4	Mult1	Yes	Multd	M[80]	R(F2)	Load3		SUBI	R1	R1	#8	
	Mult2	No						BNEZ	R1	Loop		

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
16	64	Fu	Load3	Mult1						

3/12/02

CS252/Osher
Lec. 15.44

Loop Example Cycle 17

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1	9	10	Load1	No
1	MULTD	F4	F0	F2	2	14	15	Load2	No
1	SD	F4	0	R1	3			Load3	Yes 64
2	LD	F0	0	R1	6	10	11	Store1	Yes 80 Mult1
2	MULTD	F4	F0	F2	7	15	16	Store2	Yes 72 Mult2
2	SD	F4	0	R1	8			Store3	Yes 64 Mult1

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:	LD	F0	0	R1
Add1	No							LD	F0	0	R1	←
Add2	No							MULTD	F4	F0	F2	←
Add3	No							SD	F4	0	R1	
Mult1	Yes	Multd			R(F2)	Load3		SUBI	R1	R1	#8	←
Mult2	No							BNEZ	R1	Loop		

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
17	64	Fu	Load3	Mult1						

3/12/02

CS252/Osher
Lec. 15.45

Loop Example Cycle 18

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1	9	10	Load1	No
1	MULTD	F4	F0	F2	2	14	15	Load2	No
1	SD	F4	0	R1	3			Load3	Yes 64
2	LD	F0	0	R1	6	10	11	Store1	Yes 80 Mult1
2	MULTD	F4	F0	F2	7	15	16	Store2	Yes 72 Mult2
2	SD	F4	0	R1	8			Store3	Yes 64 Mult1

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:	LD	F0	0	R1
Add1	No							LD	F0	0	R1	←
Add2	No							MULTD	F4	F0	F2	←
Add3	No							SD	F4	0	R1	
Mult1	Yes	Multd			R(F2)	Load3		SUBI	R1	R1	#8	←
Mult2	No							BNEZ	R1	Loop		

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
18	64	Fu	Load3	Mult1						

3/12/02

CS252/Osher
Lec. 15.46

Loop Example Cycle 19

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1	9	10	Load1	No
1	MULTD	F4	F0	F2	2	14	15	Load2	No
1	SD	F4	0	R1	3	18	19	Load3	Yes
2	LD	F0	0	R1	6	10	11	Store1	No
2	MULTD	F4	F0	F2	7	15	16	Store2	Yes
2	SD	F4	0	R1	8	19	20	Store3	Yes

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
Add1	No							LD F0 0 R1
Add2	No							MULTD F4 F0 F2
Add3	No							SD F4 0 R1
Mult1	Yes	Multd			R(F2)	Load3		SUBI R1 R1 #8
Mult2	No							BNEZ R1 Loop

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
19	56	Fu	Load3	Mult1						

3/12/02

CS252/Galler
Lec. 15.61

Loop Example Cycle 20

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1	9	10	Load1	Yes
1	MULTD	F4	F0	F2	2	14	15	Load2	No
1	SD	F4	0	R1	3	18	19	Load3	Yes
2	LD	F0	0	R1	6	10	11	Store1	No
2	MULTD	F4	F0	F2	7	15	16	Store2	No
2	SD	F4	0	R1	8	19	20	Store3	Yes

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
Add1	No							LD F0 0 R1
Add2	No							MULTD F4 F0 F2
Add3	No							SD F4 0 R1
Mult1	Yes	Multd			R(F2)	Load3		SUBI R1 R1 #8
Mult2	No							BNEZ R1 Loop

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
20	56	Fu	Load1	Mult1						

• Once again: In-order issue, out-of-order execution and out-of-order completion.

3/12/02

CS252/Galler
Lec. 15.68

Why can Tomasulo overlap iterations of loops?

- **Register renaming**
 - Multiple iterations use different physical destinations for registers (dynamic loop unrolling).
- **Reservation stations**
 - Permit instruction issue to advance past integer control flow operations
 - Also buffer old values of registers - totally avoiding the WAR stall that we saw in the scoreboard.
- **Other perspective: Tomasulo building data flow dependency graph on the fly.**

3/12/02

CS252/Galler
Lec. 15.69

Tomasulo's scheme offers 2 major advantages

- (1) the distribution of the hazard detection logic
 - distributed reservation stations and the CDB
 - If multiple instructions waiting on single result, & each instruction has other operand, then instructions can be released simultaneously by broadcast on CDB
 - If a centralized register file were used, the units would have to read their results from the registers when register buses are available.
- (2) the elimination of stalls for WAW and WAR hazards

3/12/02

CS252/Galler
Lec. 15.70

What about Precise Interrupts?

- State of machine looks as if no instruction beyond faulting instructions has issued
- Tomasulo had:
 - In-order issue, out-of-order execution, and out-of-order completion**
- Need to "fix" the out-of-order completion aspect so that we can find precise breakpoint in instruction stream.

3/12/02

CS252/Galler
Lec. 15.71

Relationship between precise interrupts and speculation:

- **Speculation: guess and check**
- **Important for branch prediction:**
 - Need to "take our best shot" at predicting branch direction.
- **If we speculate and are wrong, need to back up and restart execution to point at which we predicted incorrectly:**
 - This is exactly same as precise exceptions!
- **Technique for both precise interrupts/exceptions and speculation: *in-order completion or commit***

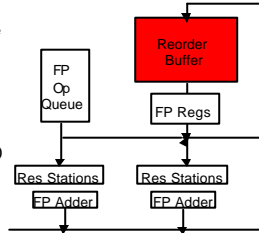
3/12/02

CS252/Galler
Lec. 15.72

HW support for precise interrupts

- Need HW buffer for results of uncommitted instructions: **reorder buffer**

- 3 fields: instr, destination, value
- Use reorder buffer number instead of reservation station when execution completes
- Supplies operands between execution complete & commit
- (Reorder buffer can be operand source => more registers like RS)
- Instructions **commit**
- Once instruction commits, result is put into register
- As a result, easy to undo speculated instructions on mispredicted branches or **exceptions**



3/12/02

CS252/Callier
Lec. 15.73

Four Steps of Speculative Tomasulo Algorithm

1. Issue—get instruction from FP Op Queue

If reservation station and reorder buffer slot free, issue instr & send operands & reorder buffer no. for destination (this stage sometimes called "dispatch")

2. Execution—operate on operands (EX)

When both operands ready then execute; if not ready, watch CDB for result; when both in reservation station, execute; checks RAW (sometimes called "issue")

3. Write result—finish execution (WB)

Write on Common Data Bus to all awaiting FUs & reorder buffer; mark reservation station available.

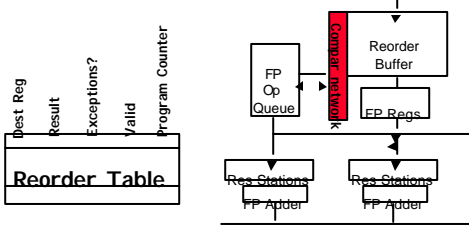
4. Commit—update register with reorder result

When instr. at head of reorder buffer & result present, update register with result (or store to memory) and remove instr from reorder buffer. Mispredicted branch flushes reorder buffer (sometimes called "graduation")

3/12/02

CS252/Callier
Lec. 15.74

What are the hardware complexities with reorder buffer (ROB)?



- How do you find the latest version of a register?
 - (As specified by Smith paper) need associative comparison network
 - Could use future file or just use the register result status buffer to track which specific reorder buffer has received the value
- Need as many ports on ROB as register file

3/12/02

CS252/Callier
Lec. 15.75

Summary

- Reservations stations: *implicit register renaming* to larger set of registers + buffering source operands
 - Prevents registers as bottleneck
 - Avoids WAR, WAW hazards of Scoreboard
 - Allows loop unrolling in HW
- Not limited to basic blocks (integer units gets ahead, beyond branches)
- Today, helps cache misses as well
 - Don't stall for L1 Data cache miss (insufficient ILP for L2 miss?)
- Lasting Contributions
 - Dynamic scheduling
 - Register renaming
 - Load/store disambiguation
- 360/91 descendants are Pentium III; PowerPC 604; MIPS R10000; HP-PA 8000; Alpha 21264

3/12/02

CS252/Callier
Lec. 15.76