

---

# EEC 581 Computer Architecture

## Lec 6 – Instruction Level Parallelism (2.4 & 2.5 Dynamic Scheduling)

Chansu Yu  
Electrical and Computer Engineering  
Cleveland State University

---

### Acknowledgement ...

- **Part of class notes are from**
  - David Patterson
  - Electrical Engineering and Computer Sciences
  - University of California, Berkeley
  - <http://www.eecs.berkeley.edu/~pattsrn>
  - <http://www-inst.eecs.berkeley.edu/~cs252>
  
- **And, from**
  - Mary Jane Irwin
  - Computer Science and Engineering
  - Pennsylvania State University
  - <http://www.cse.psu.edu/~mji>
  - [www.cse.psu.edu/~cg431](http://www.cse.psu.edu/~cg431)

## Overview of Chap. 2 & 3 (again)

---

- Pipelined architecture allows multiple instructions run in parallel (ILP)
  - But, it has data and control hazard problems
  - How can we avoid or alleviate the hazard problems in pipelined architecture?
  - Key idea is to “reorder” the execution of instructions !!!
- 2.3 Branch prediction (branch history table) } 2.6 Speculative execution (“commit”)  
2.4 & 2.5 Multiple issue – dependency Dynamic scheduling (forwarding) }  
2.7 Multiple issue – dependency Static scheduling (“VLIW”)

10/12/2007

3

## Outline

---

- ILP (2.1)
- Compiler techniques to increase ILP (2.1)
- Loop Unrolling (2.2)
- Static Branch Prediction (2.3)
- Dynamic Branch Prediction (2.3)
- Overcoming Data Hazards with Dynamic Scheduling (2.4)
- Tomasulo Algorithm (2.5)
- Conclusion

10/12/2007

4

## Extracting Yet *More* Performance

---

- **Two options:**
  - Increase the depth of the pipeline to increase the clock rate – superpipelining
  
  - Fetch (and execute) more than one instructions at one time (expand every pipeline stage to accommodate multiple instructions) – multiple-issue (VLIW or superscalar)

10/12/2007

5

## Extracting Yet *More* Performance

---

- **Superpipelined:** Increase the depth of the pipeline leading to shorter clock cycles (and more instructions “in flight” at one time)
  - The higher the degree of superpipelining, the more forwarding/hazard hardware needed, the more pipeline latch overhead, and the bigger the clock skew
- **Multiple-issue:** Launching multiple instructions per stage allows the instruction execution rate, CPI, to be less than 1
  - So instead we use **IPC**: instructions per clock cycle
  - E.g., a 6 GHz, four-way multiple-issue processor can execute at a peak rate of 24 billion instructions per second with a best case CPI of 0.25 or a best case IPC of 4

10/12/2007

6

## Multiple-Issue Processor Styles

---

- **Static multiple-issue processors (aka VLIW)**
  - Decisions on which instructions to execute simultaneously are being made statically (at compile time by the compiler)
  - E.g., Intel Itanium and Itanium 2 for the IA-64 ISA – EPIC (Explicit Parallel Instruction Computer)
- **Dynamic multiple-issue processors (aka superscalar)**
  - Decisions on which instructions to execute simultaneously are being made dynamically (at run time by the hardware)
  - E.g., IBM Power 2, Pentium 4, MIPS R10K, HP PA 8500

10/12/2007

7

## Multiple-Issue Datapath Responsibilities

---

- **Must handle, with a combination of hardware and software fixes, the fundamental limitations of**
  - Data hazards
    - » We'll see in more detail
  - Control hazards
    - » Use dynamic branch prediction to help resolve the ILP issue
  - Structural hazards
    - » A SS/VLIW processor has a much larger number of potential resource conflicts
    - » Functional units may have to arbitrate for result buses and register-file write ports
    - » Resource conflicts can be eliminated by duplicating the resource or by pipelining the resource

10/12/2007

8

## Instruction Issue and Completion Policies

---

- **Instruction-issue – initiate execution**
  - **Instruction lookahead capability** – fetch, decode and issue instructions beyond the current instruction
- **Instruction-completion – complete execution**
  - **Processor lookahead capability** – complete issued instructions beyond the current instruction
- **Instruction-commit** – write back results to the RegFile

**In-order issue with in-order completion**

**In-order issue with out-of-order completion**

**Out-of-order issue with out-of-order completion**

**Out-of-order issue with out-of-order completion and in-order commit**

10/12/2007

9

## In-Order Issue with In-Order Completion

---

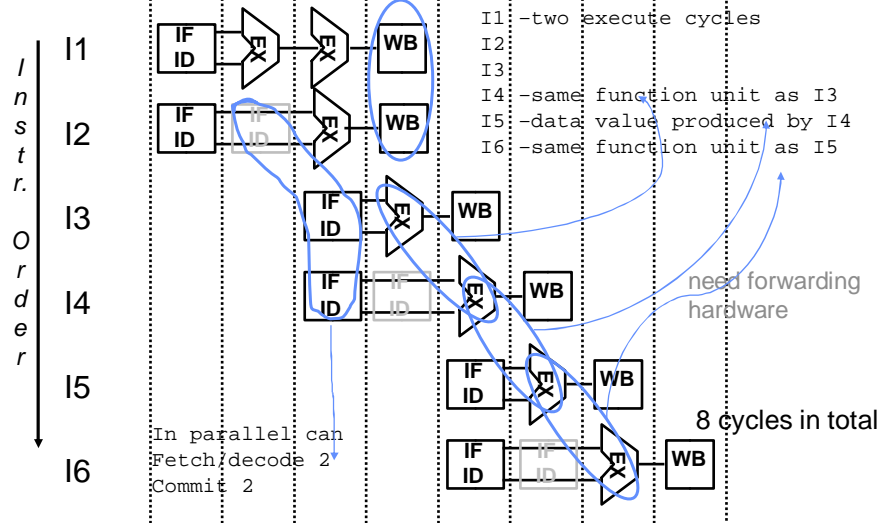
- **Simplest policy is to issue instructions in exact program order and to complete them in the same order they were fetched (i.e., in program order)**
- **Example:**
  - **Assume a pipelined processor**
    - » that can fetch and decode two instructions per cycle,
    - » that has three functional units, and
    - » that can complete (and write back) two results per cycle

```
I1 - needs two execute cycles (a multiply)
I2
I3
I4 - needs the same function unit as I3
I5 - needs data value produced by I4
I6 - needs the same function unit as I5
```

10/12/2007

10

## In-Order Issue, In-Order Completion



10/12/2007

11

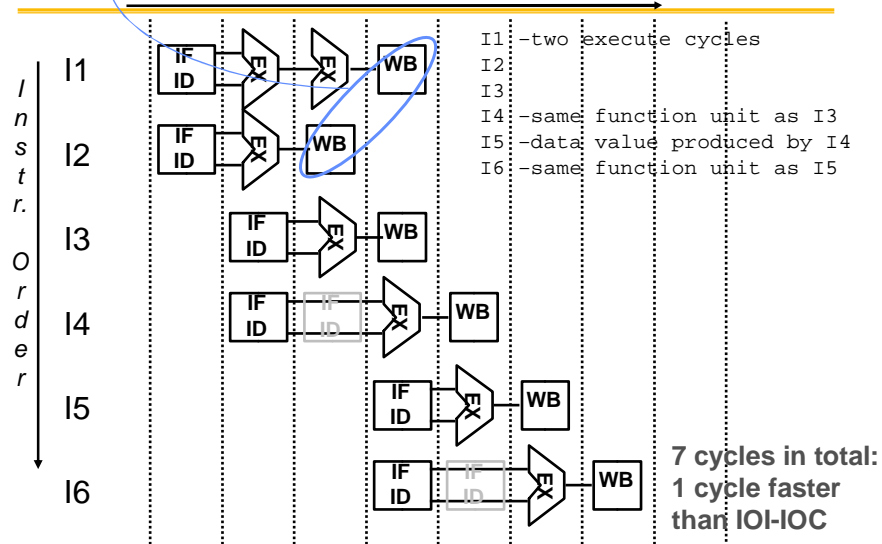
## In-Order Issue with Out-of-Order Completion

- With out-of-order completion, a later instruction may complete **before** a previous instruction
- Instruction issue is **stalled** when there is a resource conflict (e.g., for a functional unit) or a data conflict
- New type of hazards due to
  - Anti-dependency (WAR hazard)
  - Output dependency (WAW hazard)

10/12/2007

12

## IOI-OOC Example



10/12/2007

13

## Data Dependence and Hazards

- $\text{Instr}_j$  is data dependent on  $\text{Instr}_i \Rightarrow$  RAW hazard

$\begin{matrix} \curvearrowright \\ \curvearrowright \end{matrix}$ 
  
 I: add r1,r2,r3  
 J: sub r4,r1,r3

- $\text{Instr}_j$  is name dependent (anti-dependency) on  $\text{Instr}_i \Rightarrow$  WAR hazard

$\begin{matrix} \curvearrowright \\ \curvearrowright \end{matrix}$ 
  
 H: div r1,r2,r3  
 I: add r4,r1,r5  
 J: sub r5,r6,r7

- $\text{Instr}_j$  is output dependent on  $\text{Instr}_i \Rightarrow$  WAW hazard

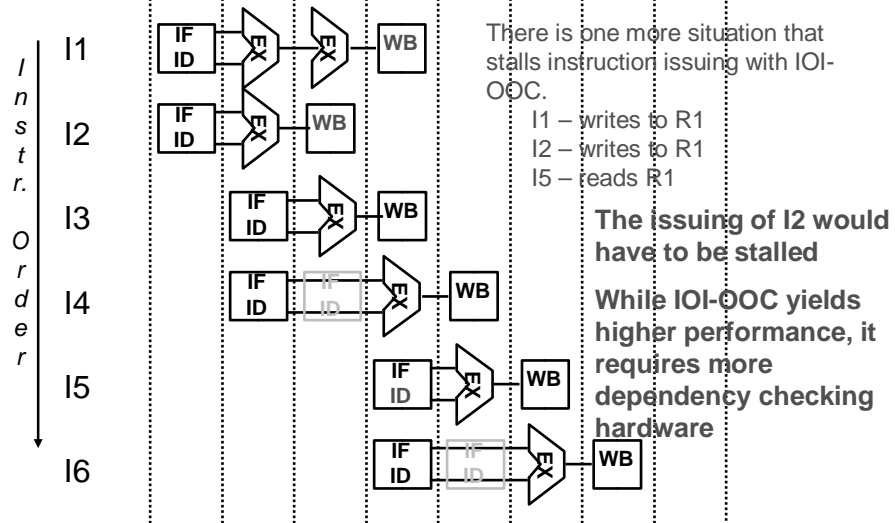
$\begin{matrix} \curvearrowright \\ \curvearrowright \end{matrix}$ 
  
 I: mul r1,r4,r3  
 J: add r1,r2,r3  
 K: sub r6,r1,r7

Not a problem  
 in IOI-IOC  
 processor

10/12/2007

14

## IOI-OOC: Output Dependencies

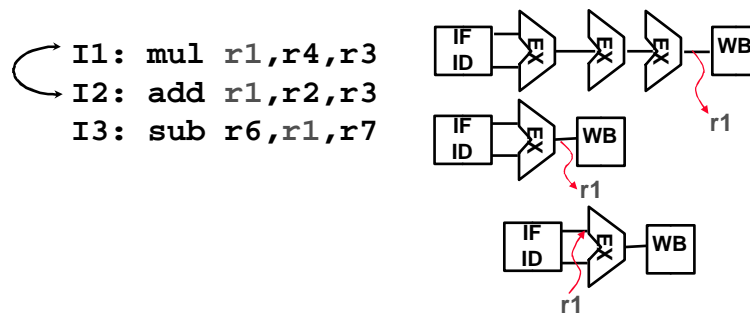


10/12/2007

15

## IOI-OOC: Output Dependencies

- WAW hazard



10/12/2007

16

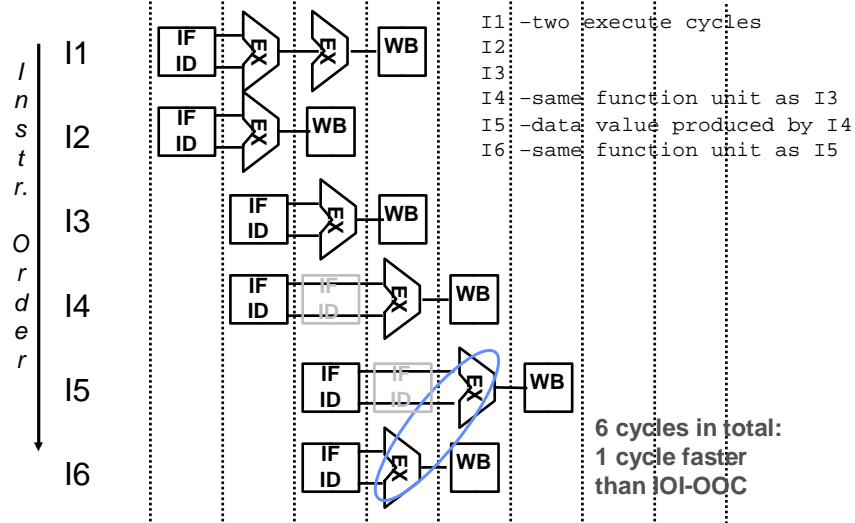
## Out-of-Order Issue with Out-of-Order Completion

- IOI processor stops decoding an instruction whenever it has a resource conflict or a data dependency.
- But, next instructions might have neither resource conflict nor a data dependency
- Fetch and decode instructions *beyond* the conflicted one,
  - store them in an **instruction buffer** (as long as there's room), and
  - flag those instructions in the buffer that don't have resource conflicts or data dependencies
  - Flagged instructions are then issued from the buffer without regard to their program order

10/12/2007

17

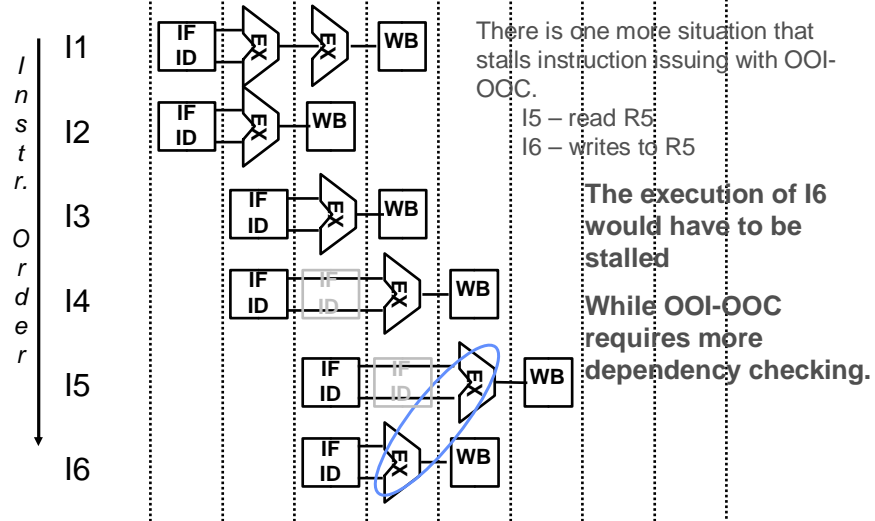
## OOI-OOC Example



10/12/2007

18

## OOI-OOC: Anti-Dependencies

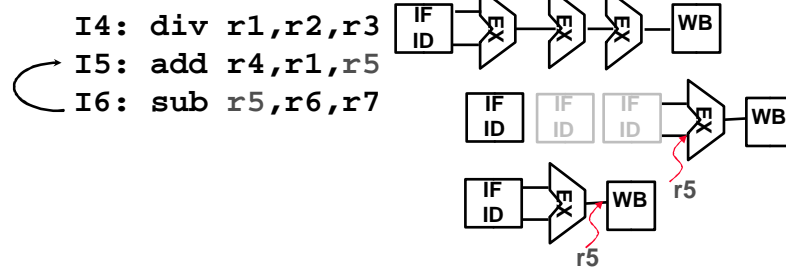


10/12/2007

19

## OOI-OOC: Anti-Dependencies

- WAR hazard



10/12/2007

20

## Dependencies Review

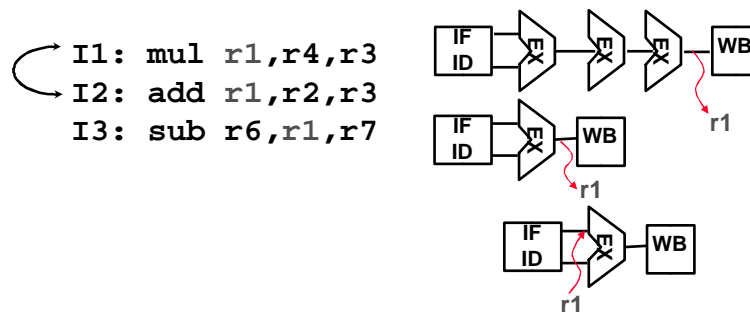
- Each of the three data dependencies
    - True data dependencies (RAW)
    - Anti-dependencies (WAR)
    - Output dependencies (WAW)
- } storage conflicts
- manifests itself through the use of registers (or other storage locations)
- True dependencies represent the flow of data and information through a program
  - Anti- and output dependencies arise because of the limited number of registers; programmers reuse registers for different computations

10/12/2007

21

## Register Renaming

- WAW hazard



- can be avoided by register renaming

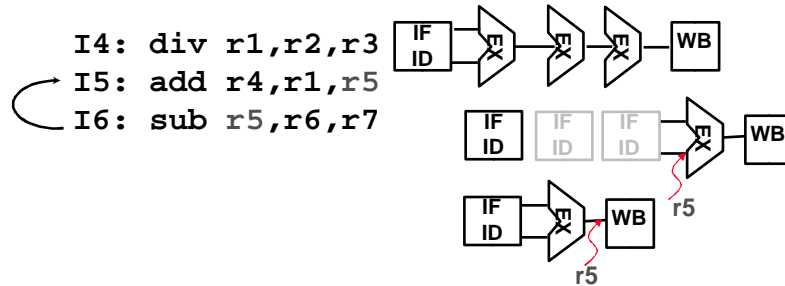
I1: mul r1,r4,r3  
 I2: add r10,r2,r3  
 I3: sub r6,r10,r7

10/12/2007

22

## Register Renaming

- WAR hazard



- Can be avoided by register renaming

I4: `div r1,r2,r3`  
 I5: `add r4,r1,r5`  
 I6: `sub r10,r6,r7`

10/12/2007

23

## Storage Conflicts and Register Renaming

- Storage conflicts can be reduced (or eliminated) by increasing or duplicating the troublesome resource
  - Provide additional registers that are used to reestablish the correspondence between registers and values
- Register renaming – the processor renames the original register identifier in the instruction to a new register (one not in the visible register set)

$R3 := R3 * R5$	$\Rightarrow$	$R3b := R3a * R5a$
$R4 := R3 + 1$		$R4a := R3b + 1$
$R3 := R5 + 1$		$R3c := R5a + 1$

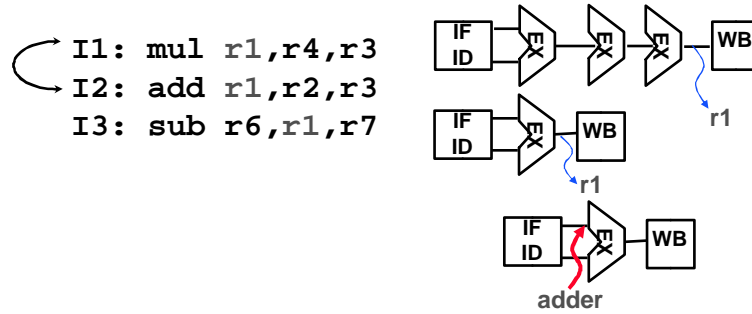
- With a limited number of registers (e.g., IBM 360 in 1966), hardware-based, dynamic scheduling was used.

10/12/2007

24

## Dynamic Scheduling

- WAW hazard



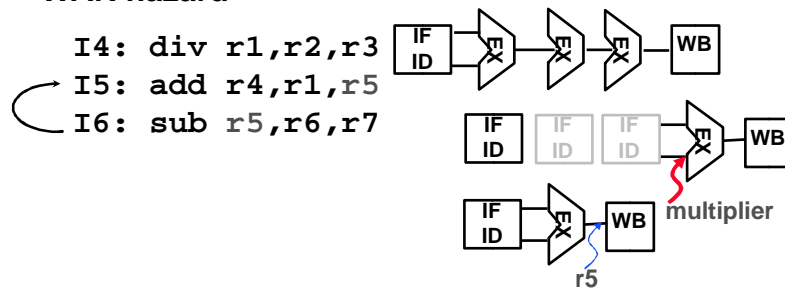
- Specify the functional unit that produces the new value of the register

10/12/2007

25

## Dynamic Scheduling

- WAR hazard



- Specify the functional unit that produces the new value of the register

=> “generalized forwarding”

10/12/2007

26



## HW Schemes: Instruction Parallelism

---

- **Key idea: Allow instructions behind stall to proceed**

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

- **Enables out-of-order execution and allows out-of-order completion (e.g., SUBD)**
  - In a dynamically scheduled pipeline, all instructions still pass through issue stage in order (in-order issue)
- **Will distinguish when an instruction *begins execution* and when it *completes execution*; between 2 times, the instruction is *in execution***
- **Note: Dynamic execution creates WAR and WAW hazards and makes exceptions harder**

10/12/2007

29

## A Dynamic Algorithm: Tomasulo's

---

- **For IBM 360/91 (before caches!)**
  - ⇒ Long memory latency (cache miss delay in modern architecture)
  - ⇒ Long FP delays
- **Architecture**
  - Small number of FP registers (4 in 360) prevented interesting compiler scheduling of operations
  - Pipelined FP functional units (3 cycles for adder, 2 cycles multiplier, 6cycles load, 3 cycles store)

10/12/2007

30

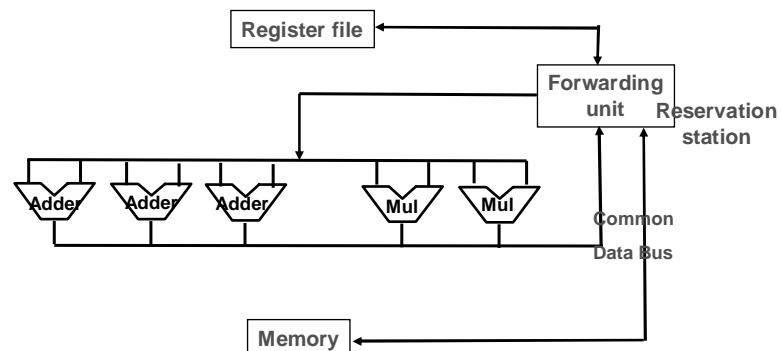
## A Dynamic Algorithm: Tomasulo's

- The smaller number of FP registers and pipelined FP functional units 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, Pentium 4, AMD Opteron, Power 5, ...

10/12/2007

31

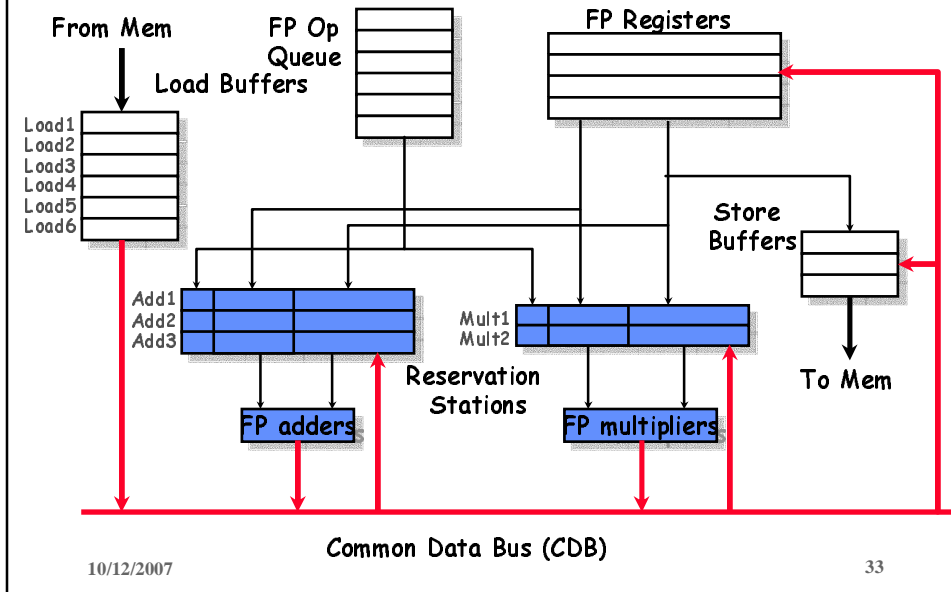
## Tomasulo Organization



10/12/2007

32

## Tomasulo Organization



## 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); called register renaming ;**
  - 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**
  - Avoids RAW hazards by executing an instruction only when its operands are available
- **Load and Stores treated as FUs with RSs as well**
- **Integer instructions can go past branches (predict taken), allowing FP ops beyond basic block in FP queue**

10/12/2007

34

## Reservation Station Components

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

**V<sub>j</sub>, V<sub>k</sub>:** Value of Source operands

- Store buffers has V field, result to be stored

**Q<sub>j</sub>, Q<sub>k</sub>:** Reservation stations producing source registers (value to be written)

- Note: Q<sub>j</sub>, Q<sub>k</sub>=0 => ready
- Store buffers only have Q<sub>i</sub> for RS producing result

Either V<sub>j</sub> or Q<sub>j</sub>

**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.

10/12/2007

35

## 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

10/12/2007

36

## Three Stages of Tomasulo Algorithm

- **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 Fl .pt. +,-; 10 for \* ; 40 clks for /**

10/12/2007

37

## Tomasulo Example

Instruction Stream

Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Exec	Write	Comp	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				

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

3 Load/Buffers

Reservation Stations:

Time	Name	Busy	Op	<i>V<sub>j</sub></i>	<i>V<sub>k</sub></i>	<i>Q<sub>j</sub></i>	<i>Q<sub>k</sub></i>
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

FU count down

3 FP Adder R.S.  
2 FP Mult R.S.

Register result status:

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

Clock cycle counter

10/12/2007

38

## Tomasulo Example Cycle 1

*Instruction status:*

Instruction	j	k	Exec		Write	Load	Busy		Address
			Issue	Comp Result			Op	Address	
LD	F6	34+	R2	1		Load1	Yes	34+R2	
LD	F2	45+	R3			Load2	No		
MULTD	F0	F2	F4			Load3	No		
SUBD	F8	F6	F2						
DIVD	F10	F0	F6						
ADDD	F6	F8	F2						

*Reservation Stations:*

Time	Name	Busy	Op	S1	S2	RS	RS
				Vj	Vk	Qj	Qk
Add1		No					
Add2		No					
Add3		No					
Mult1		No					
Mult2		No					

*Register result status:*

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
1				Load1					

10/12/2007

39

## Tomasulo Example Cycle 2

*Instruction status:*

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

*Reservation Stations:*

Time	Name	Busy	Op	S1	S2	RS	RS
				Vj	Vk	Qj	Qk
Add1		No					
Add2		No					
Add3		No					
Mult1		No					
Mult2		No					

*Register result status:*

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
2		Load2			Load1				

**Note: Can have multiple loads outstanding**

10/12/2007

40

## Tomasulo Example Cycle 3

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

Reservation Stations:				S1	S2	RS	RS
Time	Name	Busy	Op	Vj	Vk	Qj	Qk
Add1		No					
Add2		No					
Add3		No					
Mult1	Yes	MULTD		R(F4)	Load2		
Mult2	No						

Register result status:		F0	F2	F4	F6	F8	F10	F12	...	F30
Clock	3	Multi	Load2		Load1					

- Note: registers names are removed ("renamed") in Reservation Stations; MULT issued
- Load1 completing; what is waiting for Load1?

10/12/2007

41

## Tomasulo Example Cycle 4

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

Reservation Stations:				S1	S2	RS	RS
Time	Name	Busy	Op	Vj	Vk	Qj	Qk
Add1	Yes	SUBD	M(A1)			Load2	
Add2	No						
Add3	No						
Mult1	Yes	MULTD		R(F4)	Load2		
Mult2	No						

Register result status:		F0	F2	F4	F6	F8	F10	F12	...	F30
Clock	4	Multi	Load2	M(A1)	Add1					

- Load2 completing; what is waiting for Load2?

10/12/2007

42

## Tomasulo Example Cycle 5

*Instruction status:*

Instruction	j	k	Issue	Exec Write		Busy	Address
				Comp	Result		
LD	F6	34+	R2	1	3	4	Load1
LD	F2	45+	R3	2	4	5	Load2
MULTD	F0	F2	F4	3			Load3
SUBD	F8	F6	F2	4			
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2				

*Reservation Stations:*

Time	Name	Busy	Op	S1		S2		RS	RS
				Vj	Vk	Qj	Qk		
2	Add1	Yes	SUBD	M(A1)	M(A2)				
	Add2	No							
	Add3	No							
10	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
5									
	FU		Mult1	M(A2)	M(A1)	Add1	Mult2		

- Timer starts down for Add1, Mult1

10/12/2007

43

## Tomasulo Example Cycle 6

*Instruction status:*

Instruction	j	k	Issue	Exec Write		Busy	Address
				Comp	Result		
LD	F6	34+	R2	1	3	4	Load1
LD	F2	45+	R3	2	4	5	Load2
MULTD	F0	F2	F4	3			Load3
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	Yes	SUBD	M(A1)	M(A2)				
	Add2	Yes	ADDD		M(A2)	Add1			
	Add3	No							
9	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
6									
	FU		Mult1	M(A2)	Add2	Add1	Mult2		

- Issue ADDD here despite name dependency on F6?

10/12/2007

44

## Tomasulo Example Cycle 7

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

Reservation Stations:				S1	S2	RS	RS
Time	Name	Busy	Op	Vj	Vk	Qj	Qk
0	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	Yes	ADDD		M(A2)	Add1	
	Add3	No					
8	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:		F0	F2	F4	F6	F8	F10	F12	...	F30	
Clock	7	FU									
		Mult1	M(A2)		Add2	Add1	Mult2				

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

10/12/2007

45

## Tomasulo Example Cycle 8

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

Reservation Stations:				S1	S2	RS	RS
Time	Name	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
2	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
7	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:		F0	F2	F4	F6	F8	F10	F12	...	F30	
Clock	8	FU									
		Mult1	M(A2)		Add2	(M-M)	Mult2				

10/12/2007

46

## Tomasulo Example Cycle 9

*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				

*Reservation Stations:*

Time	Name	Busy	Op	S1	S2	RS	RS
				Vj	Vk	Qj	Qk
	Add1	No					
1	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
6	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
9	Mult1	M(A2)		Add2	(M-M)	Mult2			

10/12/2007

47

## Tomasulo Example Cycle 10

*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			

*Reservation Stations:*

Time	Name	Busy	Op	S1	S2	RS	RS
				Vj	Vk	Qj	Qk
	Add1	No					
0	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
5	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
10	Mult1	M(A2)		Add2	(M-M)	Mult2			

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

10/12/2007

48

## Tomasulo Example Cycle 11

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

Reservation Stations:				S1	S2	RS	RS
Time	Name	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
4	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
11	Mult1	M(A2)		(M-M+M)	(M-M)	Mult2			

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

10/12/2007

49

## Tomasulo Example Cycle 12

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

Reservation Stations:				S1	S2	RS	RS
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	Mult1	M(A2)		(M-M+M)	(M-M)	Mult2			

10/12/2007

50

## Tomasulo Example Cycle 13

Instruction status:				Exec Write				
Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</i>	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:			<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
Time	Name	Busy	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i> <i>Qk</i>
	Add1	No				
	Add2	No				
	Add3	No				
2	Mult1	Yes	MULTD	M(A2)	R(F4)	
	Mult2	Yes	DIVD		M(A1)	Mult1

Register result status:		<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
Clock	13	FU								
		Mult1	M(A2)		(M-M+M(M-M))		Mult2			

10/12/2007

51

## Tomasulo Example Cycle 14

Instruction status:				Exec Write				
Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</i>	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:			<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
Time	Name	Busy	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i> <i>Qk</i>
	Add1	No				
	Add2	No				
	Add3	No				
1	Mult1	Yes	MULTD	M(A2)	R(F4)	
	Mult2	Yes	DIVD		M(A1)	Mult1

Register result status:		<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
Clock	14	FU								
		Mult1	M(A2)		(M-M+M(M-M))		Mult2			

10/12/2007

52

## 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	S1	S2	RS	RS
				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	Mult1	M(A2)		(M-M+M(M-M))	Mult2				

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

10/12/2007

53

## 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	S1	S2	RS	RS
				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	M*F4	M(A2)		(M-M+M(M-M))	Mult2				

- **Just waiting for Mult2 (DIVD) to complete**

10/12/2007

54

## Skip a couple of cycles

10/12/2007

55

## Tomasulo Example Cycle 55

*Instruction status:*

Instruction	j	k	<i>Exec Write</i>			Load1	Load2	Load3	Busy	Address
			Issue	Comp	Result					
LD	F6	34+	R2	1	3	4		No		
LD	F2	45+	R3	2	4	5		No		
MULTD	F0	F2	F4	3	15	16		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					
1	Mult2	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+M(M-M)	Mult2			

10/12/2007

56

## Tomasulo Example Cycle 56

*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	56			
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					
	Mult1	No					
0	Mult2	Yes	DIVD	M*F4	M(A1)		

*Register result status:*

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

- **Mult2 (DIVD) is completing; what is waiting for it?**

10/12/2007

57

## Tomasulo Example Cycle 57

*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	56	57		
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					
	Mult1	No					
	Mult2	Yes	DIVD	M*F4	M(A1)		

*Register result status:*

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

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

10/12/2007

58

## Loop Unrolling with Tomasulo

---

- **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
- **Other perspective: Tomasulo building data flow dependency graph on the fly**

10/12/2007

59

## Loop Unrolling (review)

---

```
Loop: L.D    F0,0(R1)    ;F0=vector element
      MUL.D  F4,F0,F2    ;multiply scalar from F2
      S.D    0(R1),F4    ;store result
      DADDUI R1,R1,-8    ;decrement pointer 8B
      BNEZ   R1,Loop    ;branch R1!=zero
=>
      L.D    F0,0(R1)    ;F0=vector element
      MUL.D  F4,F0,F2    ;multiply scalar from F2
      S.D    0(R1),F4    ;store result
      L.D    F0,0(R1)    ;F0=vector element
      MUL.D  F4,F0,F2    ;multiply scalar from F2
      S.D    0(R1),F4    ;store result
```

10/12/2007

60

## Loop Unrolling with Tomasulo

*Instruction status:*

Instruction	j	k	Exec		Write	Busy	Address	Vk	Qk
			Issue	Comp Result					
LD	F0	0+	R1	√	√	Load1	Yes	R1+0	
MULTD	F4	F0	F2	√		Load2	Yes	R1-8	
SD	F4	0+	R1	√		Load3	No		
LD	F0	0+	R1	√	√	Store1	Yes	R1	Mult1
MULTD	F4	F0	F2	√		Store2	Yes	R1-8	Mult2
SD	F4	0+	R1	√					

*Reservation Stations:*

Time	Name	Busy	Op	S1	S2	RS	RS
				Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MUL		F2	Load1	
	Mult2	Yes	MUL		F2	Load2	

*Register result status:*

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
???	FU		Load2	Mult2					

10/12/2007 61

## Tomasulo's scheme offers 2 major advantages

- 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
- Elimination of stalls for WAW and WAR hazards

## 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! (Section 2.6)**

10/12/2007

63

## Conclusions ...

---

- **Reservations stations: *renaming* to larger set of registers + buffering source operands**
  - Prevents registers as bottleneck
  - Avoids WAR, WAW hazards
  - Allows loop unrolling in HW
- **Not limited to basic blocks (integer units gets ahead, beyond branches)**
- **Helps cache misses as well**
- **Lasting Contributions**
  - Dynamic scheduling
  - Register renaming
  - Load/store disambiguation
- **360/91 descendants are Intel Pentium 4, IBM Power 5, AMD Athlon/Opteron, ...**

10/12/2007

64