

Lecture 21: Instruction-Level Parallelism

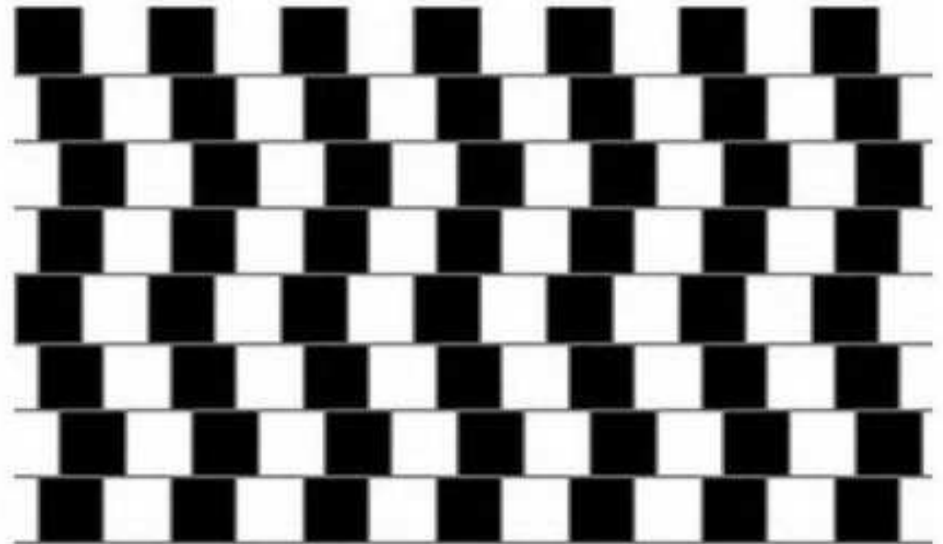
COS 471a, COS 471b / ELE 375

Computer Architecture and Organization

Princeton University
Fall 2004

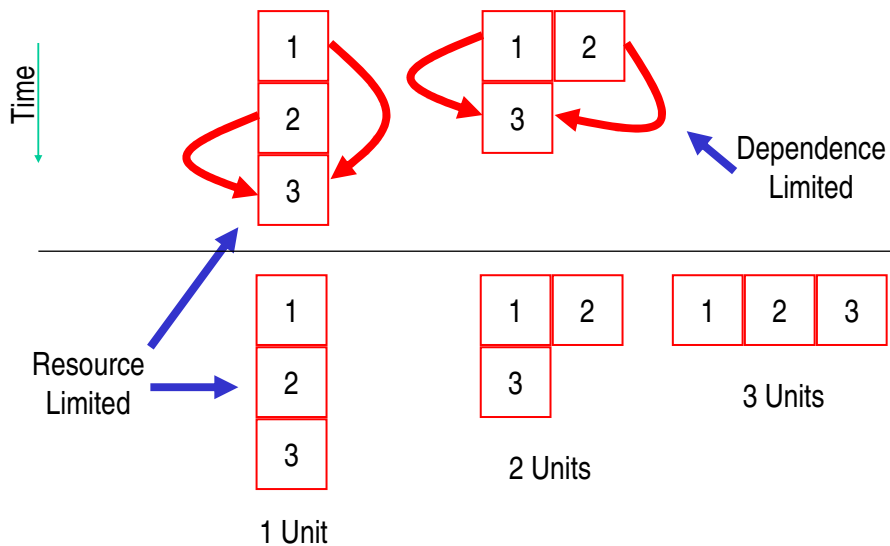
Prof. David August

1



Parallelism

Independent units of work can execute concurrently if sufficient resources exist



3

Where to Find Parallelism?

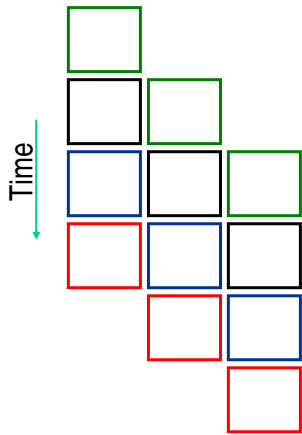
Parallelism can be found/exists at different granularities

- Instruction Level
 - Ex: add instruction executes with multiply instruction
 - Compiler/HW good at finding this
- Thread Level
 - Ex: screen redraw function executes with recalculate in spreadsheet
 - Programmers good at finding this
- Process Level
 - Ex: Simulation job runs on same machines as spreadsheet
 - Users good at creating this

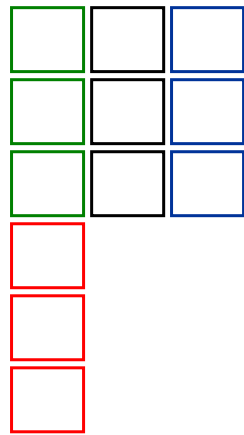
4

Organization of Parallelism

Pipelined (3 Stages)



Concurrent (3 Pipes)



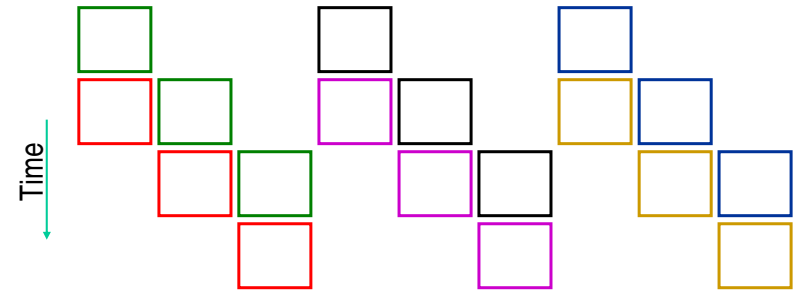
Resource →

Speedup?

5

Organization of Parallelism

Pipelined & Concurrent (3 Stages/3 Pipes)

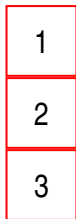


Resource →

Speedup?

6

Speedup with Parallelism

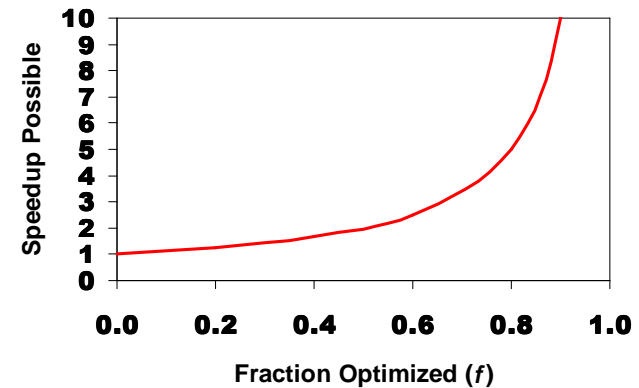


- 3 resources à 3x speedup
- N resources à N-times speedup
- But, in practice...

7

Don't Forget Amdahl's Law

Speed Enhancement is limited by fraction optimized:



$$\lim_{s \rightarrow \infty} \frac{1}{1 - f + \frac{f}{s}} = \frac{1}{1 - f}$$

where f is fraction optimized,
s is speedup of that fraction

8

Response Time Measurement

$$CPU\ time = \frac{Instructions}{Program} \times \frac{Cycles}{Instruction} \times \frac{Seconds}{Cycle}$$

- Instruction-Level Parallelism
 - CPI - Cycles per Instruction
 - IPC - Instructions per Cycle

9

Types of ILP Processors Compiler or Processor Exposed

- Explicitly Parallel Machines
 - ISA Support, simple hardware
 - VLIW, EPIC
 - Compiler expresses parallelism in the program
- Out-of-Order Machines
 - ISA independent, complex hardware
 - RISC, CISC
 - Hardware computes parallelism in the program
 - Compiler still helpful

10

ILP: Resources and Dependences

Dependences:

- Same as in discussion of pipelining

Resources:

- Usually execution units are limiting factor (Types: ALU, Float, Memory)
- Also: bypass, data busses, etc.

11

ILP: Dependences: Registers

Register Dependences

- True Dependences (RAW): Respect
- False Register Dependences (WAR, WAW): Rename

$$r1 = r2 + r3$$

$$r2 = r9 - 7$$

12

ILP: Dependences: Memory

Memory Dependences

- RAW, WAW, WAR: same as register deps except:
- Renaming difficult or impossible
- Discovery difficult in compiler, late in HW

```
M[r1] = r2 + r3
  ↘ ?
r4 = M[r9]
```

13

ILP: Dependences: Memory

Memory Dependences

- Compiler analysis quite sophisticated
- Understanding of stack, head, globals
- Example:

```
r1 = malloc(5)
r2 = malloc(5)
M[r1] = r3
r4 = M[r2]
```

14

Exposing ILP

```
r2 = 0
Loop:
r1 = M[r2]    ;; Assume array starts at address 0
r1 = r1 + 5
M[r2] = r1
r2 = r2 + 4
branch r2 < 1000, Loop
```

Any parallelism here?

15

Exposing ILP

Optimization Step 1: Unrolling

```
r2 = 0
Loop:
r1 = M[r2]
r1 = r1 + 5
M[r2] = r1
r2 = r2 + 4
branch r2 >= 1000, Exit ;; delete after proving even iteration count
r1 = M[r2]
r1 = r1 + 5
M[r2] = r1
r2 = r2 + 4
branch r2 < 1000, Loop
Exit:
```

Now any parallelism?

16

Exposing ILP

Optimization Step 2: Branch Elimination

```
r2 = 0
Loop:
  r1 = M[r2]
  r1 = r1 + 5
  M[r2] = r1
  r2 = r2 + 4
  r1 = M[r2]
  r1 = r1 + 5
  M[r2] = r1
  r2 = r2 + 4
  branch r2 < 1000, Loop
Exit:
```

Now any parallelism?

17

Exposing ILP

Optimization Step 3: Induction Variable Opti

```
r2 = 0
Loop:
  r1 = M[r2]
  r1 = r1 + 5
  M[r2] = r1
  r1 = M[r2 + 4]
  r1 = r1 + 5
  M[r2 + 4] = r1
  r2 = r2 + 8
  branch r2 < 1000, Loop
Exit:
```

Now any parallelism?

18

Exposing ILP

Optimization Step 4: Register Renaming

```
r2 = 0
Loop:
  r1 = M[r2]
  r1 = r1 + 5
  M[r2] = r1
  r11 = M[r2 + 4]
  r11 = r11 + 5
  M[r2 + 4] = r11
  r2 = r2 + 8
  branch r2 < 1000, Loop
Exit:
```

Now any parallelism?

19

Expressing ILP: Scheduling

```
r2 = 0
Loop:
  r1 = M[r2]           r11 = M[r2 + 4]
  r1 = r1 + 5         r11 = r11 + 5
  M[r2] = r1         M[r2 + 4] = r11
  r2 = r2 + 8
  branch r2 < 1000, Loop
Exit:
```

Can we do better?
If no, why?
If yes, how much better?

20

Exposing ILP in hardware

Which optimizations could hardware perform?

- Unrolling
- Branch elimination
- Induction variable optimization
- Register renaming

21

ILP: Resources and Dependences

Dependences:

- Same as in discussion of pipelining

Resources:

- Usually execution units are limiting factor (Types: ALU, Float, Memory)
- Also: bypass, data busses, etc.

22

ILP: Resources

Basic Principle:

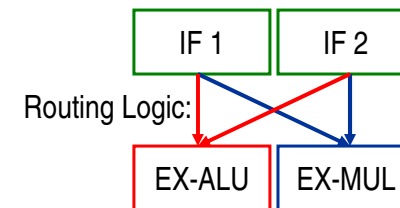
- Given N resources, N times speedup
- Assumes full resource utilization
- Implies: Goal is full resource utilization

Scheduling:

- minimize schedule height given resources/dependences
- maximize average resource utilization

23

ILP: Resources: Resource Maps



$$r1 = r2 + r3$$

$$r2 = r9 * 7$$

Add (alternative1):

Cycle	IF1	IF2	EX-ALU	EX-MUL
1				
2				

Add (alternative2):

Cycle	IF1	IF2	EX-ALU	EX-MUL
1				
2				

Mul (alternative1):

Cycle	IF1	IF2	EX-ALU	EX-MUL
1				
2				

Mul (alternative2):

Cycle	IF1	IF2	EX-ALU	EX-MUL
1				
2				

24

ILP:Resources

Resource Maps

$$r1 = r2 + r3$$

$$r2 = r9 * 7$$

Machine Allocation:

Cycle	IF1	IF2	EX-ALU	EX-MUL
1				
2				
3				

Cycle 1: $r1 = r2 + r3$ $r2 = r9 * 7$

25

ILP:Resources

Resource Maps

$$r1 = r2 + r3$$

$$r2 = r9 - 7$$

Machine Allocation:

Cycle	IF1	IF2	EX-ALU	EX-MUL
1				
2				
3				

Cycle 1: $r1 = r2 + r3$

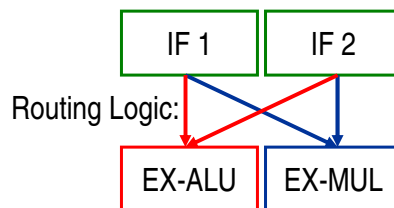
Cycle 2: $r2 = r9 - 7$

26

Resource Management in Hardware

Arbitration:

- Fixed Priority (IF1, then IF2)
- Round Robin (Alternate)
- Other



Describe the control logic for arbitration

27

A Dynamic Scheduling/OOO 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 2-4, PowerPC 604, ...

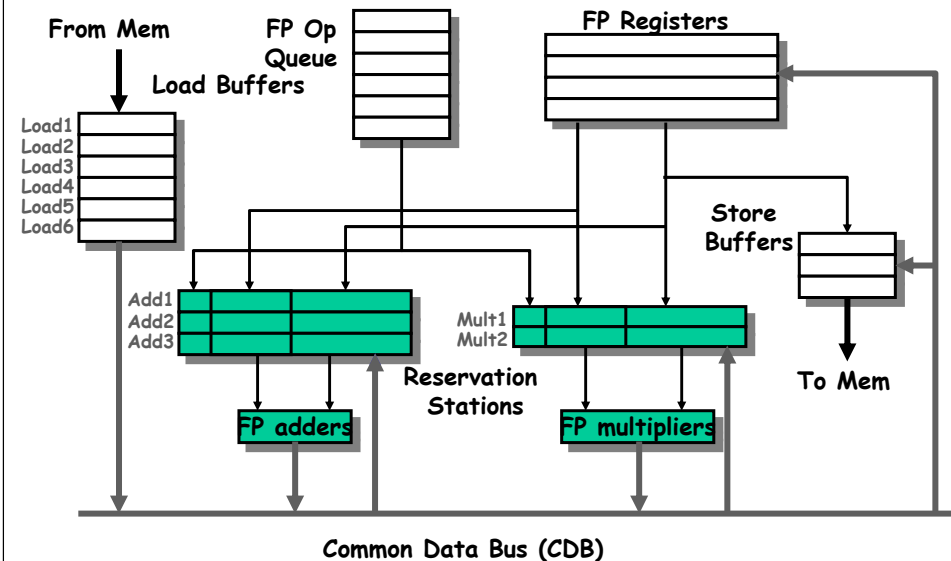
28

Tomasulo's 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 ;
 - 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

29

Tomasulo Machine Organization



30

Reservation Station Components

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

Vj, Vk: Value of source operands

- Store buffer has V field, result to be stored

Qj, Qk: Source of value still to be computed

- Note: $Q_j, Q_k = 0 \Rightarrow$ ready
- Store buffer only has 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.

31

Three Stages of Tomasulo's Algorithm

- Issue—get instruction from FP Op Queue
 - If reservation station free (no structural hazard), control issues instr & sends operands (renames registers).
 - Execute—operate on operands (EX)
 - When both operands ready then execute; if not ready, watch Common Data Bus for result
 - 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 latencies:
 - 3 clocks for Floating point add, sub; 10 for mult ; 40 for div

32

Tomasulo Example

Instruction stream

Instruction status:

Instruction	j	k	Exec Write		Busy	Address
			Issue	Comp Result		
LD	F6	34+	R2		No	
LD	F2	45+	R3		No	
MULTD	F0	F2	F4		No	
SUBD	F8	F6	F2		No	
DIVD	F10	F0	F6		No	
ADDD	F6	F8	F2		No	

Load	Busy	Address
Load1	No	
Load2	No	
Load3	No	

3 Load/Buffers

Reservation Stations:

Time	Name	Busy	Op	S1	S2	RS	RS
				Vj	Vk	Qj	Qk
	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

33

Tomasulo Example Cycle 1

Instruction status:

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

Load	Busy	Address
Load1	Yes	34+R2
Load2	No	
Load3	No	

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					

34

Tomasulo Example Cycle 2

Instruction status:

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

Load	Busy	Address
Load1	Yes	34+R2
Load2	Yes	45+R3
Load3	No	

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

35

Tomasulo Example Cycle 3

Instruction status:

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

Load	Busy	Address
Load1	Yes	34+R2
Load2	Yes	45+R3
Load3	No	

Reservation Stations:

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

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
3			Mult1	Load2			Load1		

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

36

Tomasulo Example Cycle 4

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		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:

Time	Name	Busy	Op	S1 Vi	S2 Vk	RS Qi	RS Qk
Add1	Yes	SUBD	M(A1)				Load2
Add2	No						
Add3	No						
Mult1	Yes	MULTD			R(F4)		Load2
Mult2	No						

Register result status:

Clock	FU	F0	F2	F4	F6	F8	F10	F12	...	F30
4		Mult1	Load2		M(A1)	Add1				

- Load2 completing; what is waiting for Load2?

37

Tomasulo Example Cycle 5

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				
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2					

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS 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	FU	F0	F2	F4	F6	F8	F10	F12	...	F30
5		Mult1	M(A2)		M(A1)	Add1	Mult2			

- Timer starts down for Add1, Mult1

38

Tomasulo Example Cycle 6

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				
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6				

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS 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	FU	F0	F2	F4	F6	F8	F10	F12	...	F30
6		Mult1	M(A2)		Add2	Add1	Mult2			

- Issue ADDD here despite name dependency on F6?

39

Tomasulo Example Cycle 7

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			
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6				

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS 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:

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

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

40

Tomasulo Example Cycle 8

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	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:

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

41

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	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	FU	F0	F2	F4	F6	F8	F10	F12	...	F30
9		Mult1	M(A2)		Add2	(M-M)	Mult2			

42

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	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	FU	F0	F2	F4	F6	F8	F10	F12	...	F30
10		Mult1	M(A2)		Add2	(M-M)	Mult2			

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

43

Tomasulo Example Cycle 11

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					
4	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock	FU	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!

44

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	S1	S2	RS	RS
				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	FU	F0	F2	F4	F6	F8	F10	F12	...	F30
12		Mult1	M(A2)		(M-M+N)	(M-M)	Mult2			

45

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	S1	S2	RS	RS
				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	FU	F0	F2	F4	F6	F8	F10	F12	...	F30
13		Mult1	M(A2)		(M-M+N)	(M-M)	Mult2			

46

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	S1	S2	RS	RS
				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	FU	F0	F2	F4	F6	F8	F10	F12	...	F30
14		Mult1	M(A2)		(M-M+N)	(M-M)	Mult2			

47

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

48

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

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

- Just waiting for Mult2 (DIVD) to complete

49

Tomasulo Example Cycle 55

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							
1	Mult2	Yes	DIVD	M*F4	M(A1)				

Register result status:

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

50

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

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

51

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

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

52

Tomasulo Drawbacks

- Complexity
- 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

53

Tomasulo Loop Example

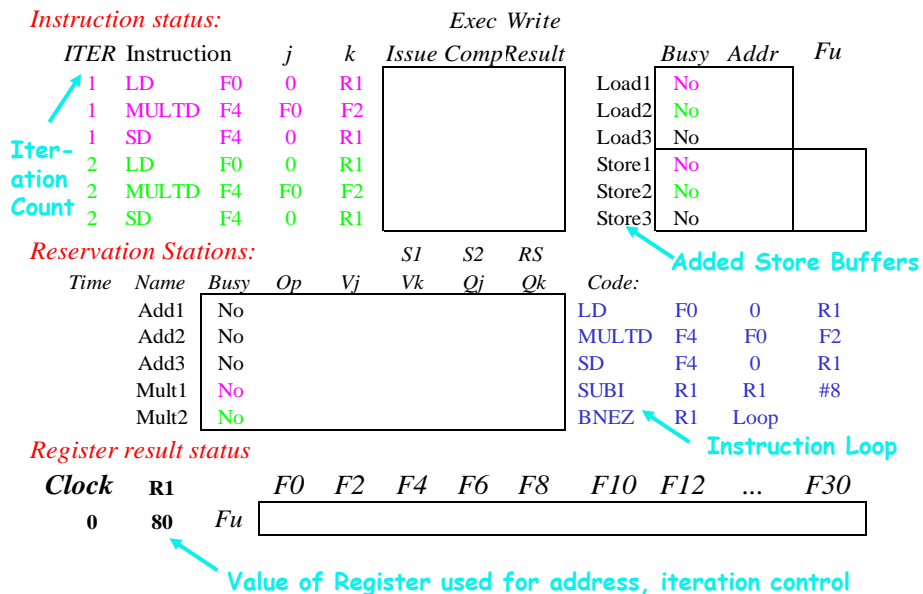
```

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)
- Let's see 2 iterations

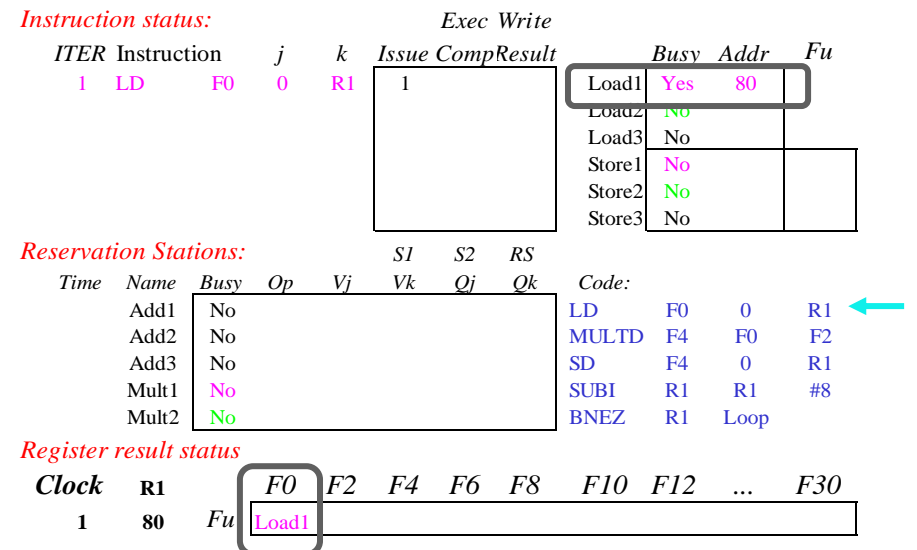
54

Loop Example



55

Loop Example Cycle 1



56

Loop Example Cycle 2

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1		Yes	80	
1	MULTD	F4	F0	F2	2		No		
							No		
							No		
							No		
							No		

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) Load1	SUBI R1 R1 #8
Mult2	No							BNEZ R1 Loop

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
2	80	Fu	Load1	Mult1						

57

Loop Example Cycle 3

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1		Yes	80	
1	MULTD	F4	F0	F2	2		No		
1	SD	F4	0	R1	3		No		
							Yes	80	Mult1
							No		
							No		

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) Load1	SUBI R1 R1 #8
Mult2	No							BNEZ R1 Loop

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
3	80	Fu	Load1	Mult1						

- Implicit renaming sets up data flow graph

58

Loop Example Cycle 4

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1		Yes	80	
1	MULTD	F4	F0	F2	2		No		
1	SD	F4	0	R1	3		No		
							Yes	80	Mult1
							No		
							No		

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) Load1	SUBI R1 R1 #8
Mult2	No							BNEZ R1 Loop

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
4	80	Fu	Load1	Mult1						

- Dispatching SUBI Instruction (not in FP queue)

59

Loop Example Cycle 5

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1		Yes	80	
1	MULTD	F4	F0	F2	2		No		
1	SD	F4	0	R1	3		No		
							Yes	80	Mult1
							No		
							No		

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) Load1	SUBI R1 R1 #8
Mult2	No							BNEZ R1 Loop

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
5	72	Fu	Load1	Mult1						

- And, BNEZ instruction (not in FP queue)

60

Loop Example Cycle 6

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
							Store2	No	
							Store3	No	

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)	Load1	SUBI R1 R1 #8
Mult2	No							BNEZ R1 Loop

Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
6	72	Fu	Load2	Mult1						

- Notice that F0 never sees Load from location 80

61

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
2	MULTD	F4	F0	F2	7		Store2	No	
							Store3	No	

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)	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 (looks like unrolling!)

62

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
2	MULTD	F4	F0	F2	7		Store2	Yes	72
2	SD	F4	0	R1	8		Store3	No	

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)	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						

63

Loop Example Cycle 9

Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu
1	LD	F0	0	R1	1	9	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
2	MULTD	F4	F0	F2	7		Store2	Yes	72
2	SD	F4	0	R1	8		Store3	No	

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)	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

64

Loop Example Cycle 10

Instruction status:

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

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
4	Mult1	Yes	Multd	M[80]	R(F2)			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

65

Loop Example Cycle 11

Instruction status:

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

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
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

66

Loop Example Cycle 12

Instruction status:

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

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
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?

67

Loop Example Cycle 13

Instruction status:

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

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
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?

68

Loop Example Cycle 14

Instruction status:

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

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
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?

69

Loop Example Cycle 15

Instruction status:

ITER	Instruction	j	k	Exec Write			Fu
				Issue	Comp	Result	
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
2	MULTD F4 F0 F2	7			Store2	Yes	72
2	SD F4 0 R1	8			Store3	No	

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
0	Mult1	No						SUBI R1 R1 #8
0	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?

70

Loop Example Cycle 16

Instruction status:

ITER	Instruction	j	k	Exec Write			Fu
				Issue	Comp	Result	
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
2	MULTD F4 F0 F2	7	15	16	Store2	Yes	72
2	SD F4 0 R1	8			Store3	No	

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
4	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
16	64	Fu	Load3	Mult1						

71

Loop Example Cycle 17

Instruction status:

ITER	Instruction	j	k	Exec Write			Fu
				Issue	Comp	Result	
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
2	MULTD F4 F0 F2	7	15	16	Store2	Yes	72
2	SD F4 0 R1	8			Store3	Yes	64

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
17	64	Fu	Load3	Mult1						

72

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	18		Load3	Yes 64
2	LD	F0	0	R1	6	10	11	Store1	Yes 80 [80]*R2
2	MULTD	F4	F0	F2	7	15	16	Store2	Yes 72 [72]*R2
2	SD	F4	0	R1	8			Store3	Yes 64 Mult1

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
18	64	Fu	Load3	Mult1						

73

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 64
2	LD	F0	0	R1	6	10	11	Store1	No
2	MULTD	F4	F0	F2	7	15	16	Store2	Yes 72 [72]*R2
2	SD	F4	0	R1	8	19		Store3	Yes 64 Mult1

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						

74

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 56
1	MULTD	F4	F0	F2	2	14	15	Load2	No
1	SD	F4	0	R1	3	18	19	Load3	Yes 64
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 64 Mult1

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.

75

Why can Tomasulo overlap iterations of loops?

- Register renaming
 - Multiple iterations use different physical destinations for registers (dynamic loop unrolling).
- Reservation stations
 - Also buffer old values of registers - totally avoiding the WAR stall that we saw in the scoreboard.
 - Multiple dynamic instructions for each static instruction
- Other perspective: Tomasulo building data flow dependency graph on the fly.

76

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

77

What about Precise Interrupts?

- 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.

78

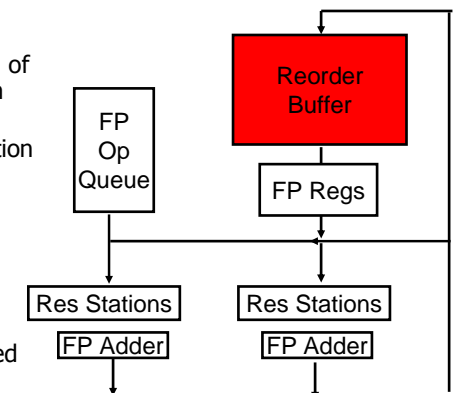
Relationship between precise interrupts and speculation:

- Speculation is a form of guessing.
- 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

79

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



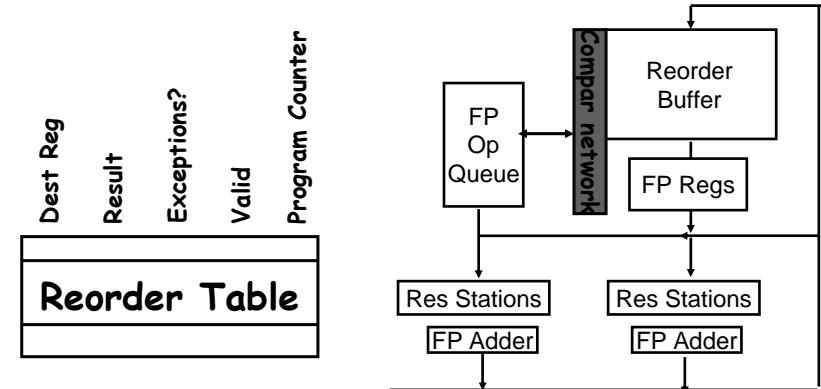
80

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.

81

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



- How do you find the latest version of a register?
 - Need associative comparison network
- Need as many ports on ROB as register file

82

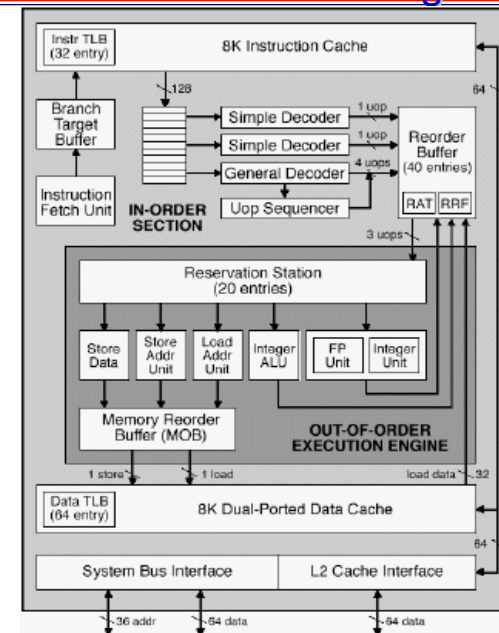
Dynamic Scheduling PowerPC 604 and Pentium2-4

Parameter	PowerPC	Pentium
Max. instructions issued/clock	4	3
Max. instr. complete exec./clock	6	6
Max. instr. committed/clock	6	3
Window (Instrs in reorder buffer)	16	40
Number of reservations stations	12	20
Number of rename registers	8int/12FP	40
No. integer functional units (FUs)	2	2
No. floating point FUs	1	1
No. branch FUs	1	1
No. complex integer FUs	1	0
No. memory FUs	1	1 load +1 store

Q: How to pipeline 1 to 17 byte x86 instructions?

83

Pentium 2-4 Block Diagram



Microprocessor Report

84

Dynamic Scheduling in Pentium2-4

- P2-4 don't pipeline 80x86 instructions
- P2-4 decode unit translates the Intel instructions into 72-bit micro-operations (- DLX)
- Sends micro-operations to reorder buffer & reservation stations
- Takes 1 clock cycle to determine length of 80x86 instructions + 2 more to create the micro-operations
- 12-14 clocks in total pipeline (- 3 state machines)
- Many instructions translate to 1 to 4 micro-operations
- Complex 80x86 instructions are executed by a conventional microprogram (8K x 72 bits) that issues long sequences of micro-operations

85

Tomasulo 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

86

Parallelism Summary

- Parallelism
 - Pipeline
 - Concurrent
- Limitations
 - Dependence
 - Resource
- Sources
 - Instruction-Level
 - Thread-Level
 - Process-Level
- ILP Exposed in
 - Compiler
 - Hardware

- Next time: Multiple Processors

87