

Advanced Pipelining and Instruction-Level Parallelism (2)

Riferimenti bibliografici

“Computer architecture, a quantitative approach”, Hennessy & Patterson: (Morgan Kaufmann eds.)

Tomasulo’s Algorithm



- For IBM 360/91 about 3 years after CDC 6600 (1966)
- Goal: High Performance without special compilers
- Differences between IBM 360 & CDC 6600 ISA
 - IBM has only 2 register specifiers/instr vs. 3 in CDC 6600
 - IBM has 4 FP registers vs. 8 in CDC 6600
 - IBM has memory-register ops
- Why Study? lead to Alpha 21264, HP 8000, MIPS 10000, Pentium II, PowerPC 604, ...

Tomasulo's algorithm

- Dynamic scheduling implies:
 - Out-of-order execution
 - Out-of-order completion
- Creates the possibility for WAR and WAW hazards
- Tomasulo's Approach
 - Tracks when operands are available
 - Introduces register renaming in hardware
 - Minimizes WAW and WAR hazards

Register Renaming

- Example:

```
DIV.D F0,F2,F4
ADD.D F6,F0,F8
S.D F6,0(R1)
SUB.D F8,F10,F14
MUL.D F6,F10,F8
```

antidependence

antidependence

+ name dependence with F6

Register Renaming

□ Example:

```
DIV.D F0,F2,F4
ADD.D S,F0,F8
S.D S,0(R1)
SUB.D T,F10,F14
MUL.D F6,F10,T
```

- Now only RAW hazards remain, which can be strictly ordered

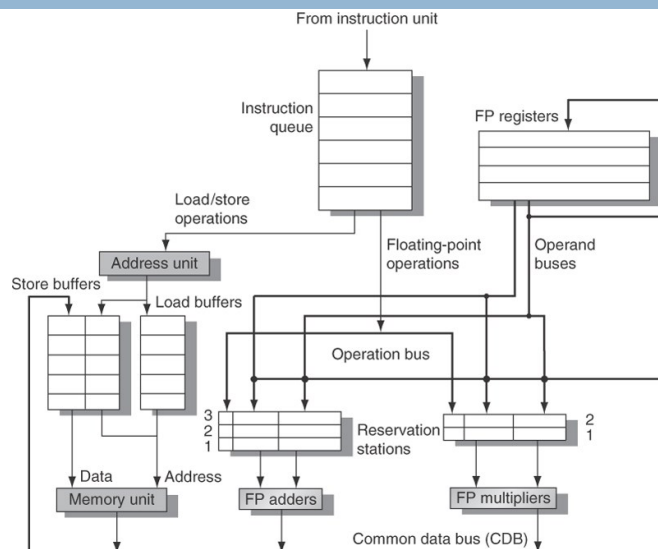
Tomasulo's approach

- Control & buffers **distributed** with Function Units (FUs) vs. centralized in scoreboard
 - FU buffers called **Reservation Stations** (RS) have pending operands
- Register renaming is provided by reservation stations (RS) which contains:
 - The instruction
 - Buffered operand values (when available)
 - Reservation station number of instruction providing the operand values
- RS fetches and buffers an operand as soon as it becomes available (not necessarily involving register file)
- Pending instructions designate the RS to which they will send their output

Tomasulo's approach

- As instructions are issued, the register specifiers are renamed with the reservation station
- May be more reservation stations than registers
- Load and Stores treated as FUs with RSs as well
 - ▣ Load and store buffers hold data or addresses from or to memory
- FP registers are connected by buses to functional unit and store buffers
- Results from FU and memory are sent on a Common Data Bus to everywhere except load buffer
- Only the last output updates the register file

Tomasulo's Algorithm



Three steps of Tomasulo's Algorithm

□ Issue

- Get next instruction from FIFO queue
- If available RS, issue the instruction to the RS with operand values if available
- If operand values not available, stall the instruction

□ Execute

- When operand becomes available, store it in any reservation stations waiting for it
- When all operands are ready, execute the instruction
- Loads and store maintained in program order through effective address
- No instruction allowed to initiate execution until all branches that proceed it in program order have completed

Three steps of Tomasulo's Algorithm

□ Write result

- Write result on CDB into reservation stations and store buffers
 - (Stores must wait until address and value are received)

Tomasulo vs. Scoreboard (IBM 360/91 vs. CDC 6600)

Pipelined Functional Units (6 load, 3 store, 3 +, 2 x/÷)	Multiple Functional Units (1 load/store, 1 +, 2 x, 1 ÷)
window size: ≤ 14 instructions	≤ 5 instructions
No issue on structural hazard	same
WAR: renaming avoids	stall completion
WAW: renaming avoids	stall issue
Broadcast results from FU	Write/read registers
Control: reservation stations	central scoreboard

Review: Dynamic HW Techniques for out-of-order execution

- HW exploitation of ILP
 - Works when can't know dependence at compile time
 - Code for one machine runs well on another
- Scoreboard (CDC 6600 in 1963)
 - Centralized control structure
 - No register renaming, no forwarding
 - Pipeline stalls for WAR and WAW hazards
- Reservation stations (IBM 360/91 in 1966)
 - Distributed control structures
 - **Implicit** renaming of registers (dispatched pointers)
 - WAR and WAW hazards eliminated by register renaming
 - Results broadcast to all reservation stations for RAW

Reservation Station Components

- **Op**: Operation to perform in the unit (e.g., + or -)
- **Vj, Vk**: **Value** of Source operands
 - Store buffers has only one V field, result to be stored
- **Qj, Qk**: Reservation stations producing source registers (value to be written)
 - Note: No ready flags as in Scoreboard; 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.

Steps in Tomasulo algorithm

Instruction state	Wait until	Action or bookkeeping
Issue FP operation	Station r empty	<pre> if (RegisterStat[rs].Qi != 0) {RS[r].Qj ← RegisterStat[rs].Qi} else {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0}; if (RegisterStat[rt].Qi != 0) {RS[r].Qk ← RegisterStat[rt].Qi} else {RS[r].Vk ← Regs[rt]; RS[r].Qk ← 0}; RS[r].Busy ← yes; RegisterStat[rd].Q ← r; </pre>
Load or store	Buffer r empty	<pre> if (RegisterStat[rs].Qi != 0) {RS[r].Qj ← RegisterStat[rs].Qi} else {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0}; RS[r].A ← imm; RS[r].Busy ← yes; </pre>
Load only		<pre> RegisterStat[rt].Qi ← r; </pre>
Store only		<pre> if (RegisterStat[rt].Qi != 0) {RS[r].Qk ← RegisterStat[rs].Qi} else {RS[r].Vk ← Regs[rt]; RS[r].Qk ← 0}; </pre>

Tomasulo Example Cycle 1

Instruction status:

Instruction	j	k	Exec Write		Busy Address	
			Issue	Comp Result	Load	Address
LD	F6	34+	R2	1	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	

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					

Tomasulo Example Cycle 2

Instruction status:

Instruction	j	k	Exec Write		Busy Address	
			Issue	Comp Result	Load	Address
LD	F6	34+	R2	1	Yes	34+R2
LD	F2	45+	R3	2	Yes	45+R3
MULTD	F0	F2	F4		No	
SUBD	F8	F6	F2		No	
DIVD	F10	F0	F6		No	
ADDD	F6	F8	F2		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: Unlike 6600, can have multiple loads outstanding

Tomasulo Example Cycle 3

Instruction status:

Instruction	j	k	Exec Write			Busy	Address
			Issue	Comp	Result		
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:

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									
	FU									
3	Mult1	Load2			Load1					

- Note: registers names are removed ("renamed") in Reservation Stations; MULT issued vs. scoreboard

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	S2	RS	RS
				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:

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

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	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	Mult1	M(A2)		M(A1)	Add1	Mult2			

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	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	Mult1	M(A2)		Add2	Add1	Mult2			

- Issue ADDD here vs. scoreboard?

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	S2	RS	RS
				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:

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

- Add1 completing; what is waiting for it?

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	S1	S2	RS	RS
				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	F0	F2	F4	F6	F8	F10	F12	...	F30
8	FU	Mult1	M(A2)	Add2	(M-M)	Mult2			

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	FU	Mult1	M(A2)	Add2	(M-M)	Mult2			

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	FU	Mult1	M(A2)	Add2	(M-M)	Mult2			

- Add2 completing; what is waiting for it?

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	S1	S2	RS	RS
				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			

- All quick instructions complete in this cycle!

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

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

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

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	FU	Mult1	M(A2)	(M-M+M)	(M-M)	Mult2			

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	FU	M*F4	M(A2)	(M-M+M)	(M-M)	Mult2			

Faster than light computation (skip a couple of cycles)

Tomasulo Example Cycle 55

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
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	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
1	Mult2	Yes	DIVD	M*F4	M(A1)		

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
55	FU	M*F4	M(A2)	(M-M+V	(M-M)	Mult2			

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	FU	M*F4	M(A2)		(M-M+N)	(M-M)	Mult2		

- Mult2 is completing; what is waiting for it?

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	FU	M*F4	M(A2)		(M-M+N)	(M-M)	Result		

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

Tomasulo Example Cycle 62

Instruction status:

Instruction	j	k	Read Exec Write				Exec Write			
			Issue	Oper	Comp	Result	Issue	Comp	Result	
LD	F6	34+	R2	1	2	3	4	1	3	4
LD	F2	45+	R3	5	6	7	8	2	4	5
MULTD	F0	F2	F4	6	9	19	20	3	15	16
SUBD	F8	F6	F2	7	9	11	12	4	7	8
DIVD	F10	F0	F6	8	21	61	62	5	56	57
ADDD	F6	F8	F2	13	14	16	22	6	10	11

- Why take longer on scoreboard/6600?
 - Structural Hazards
 - Lack of forwarding

Hardware-based speculation

- It hard to exploit more ILP, maintaining control dependences
- Branch prediction reduces stalls due to branches, but it is not sufficient to generate the desirable amount of ILP
- A multiple-issue processor can execute a branch every clock cycle
- Overcoming control dependence by speculating on the branch outcome and executing the program as the guess was correct

Hardware-Based Speculation

- We need mechanisms to handle incorrect speculations
 - Execute instructions along predicted execution paths but only commit the results if prediction was correct
- Instruction commit: allowing an instruction to update the register file when instruction is no longer speculative (we know branch outcome)
- Need an additional piece of hardware to prevent any irrevocable action until an instruction commits
 - I.e. updating state or taking an execution

Hardware based-speculation

- It combines three key ideas:
 - Branch prediction, to choose the next instruction
 - Speculation, to allow execution before resolution of control dependences and to undo of incorrectly speculated sequence
 - Dynamic scheduling

Implementing speculation

- Separate the bypassing of results among instructions, from the completion of an instruction (updating registers and memory)
- We need to separate the completing of execution from instruction commit
- The key idea: out-of-order execution, commit in order to prevent any irrevocable action

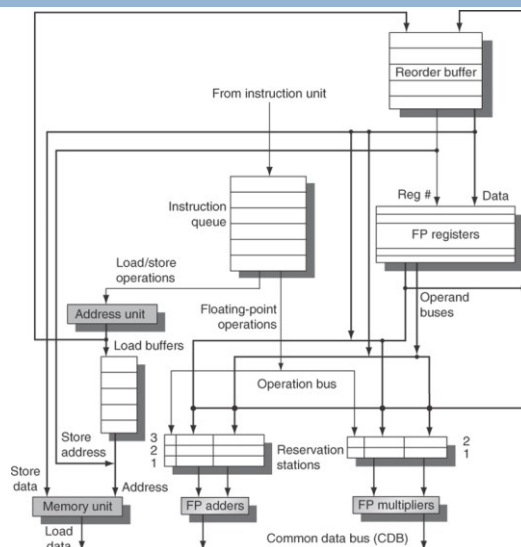
Reorder Buffer

- Adding commit phase requires an additional set of buffers (Reorder buffers) that holds the result of instruction that have finished execution but have not committed
- Four fields:
 - Instruction type: branch/store/register
 - Destination field: register number/memory address
 - Value field: output value
 - Ready field: completed execution?
- Modify reservation stations:
 - Operand source is now reorder buffer instead of functional unit

Reorder Buffer

- Register values and memory values are not written until an instruction commits
- On misprediction:
 - ▣ Speculated entries in ROB are cleared
- Exceptions:
 - ▣ Not recognized until it is ready to commit

Tomasulo's algorithm with speculation



Four steps of Tomasulo's Algorithm with reorder buffer

□ Issue

- Get next instruction from FIFO queue
- If available RS and available slot in ROB, issue the instruction to the RS with operand values if available
- Send operands to RS if available in register/ROB
- If operand values not available, stall the instruction
- The ROB allocate for results is sent to RS

□ Execute

- When operand becomes available, store it in any reservation stations waiting for it
- When all operands are ready, execute the instruction

Four steps of Tomasulo's Algorithm with reorder buffer

□ Write result

- Write result on CDB (with ROB tag) into ROB and reservation stations

□ Commit

- Branch with incorrect prediction
 - ROB is flushed and execution restart at the correct successor
- Branch with correct prediction
 - Remove instruction from ROB
- Other instructions
 - Update register file/memory and remove instruction from ROB

Multiple Issue and Static Scheduling

- To achieve $CPI < 1$, need to complete multiple instructions per clock

- Solutions:
 - ▣ Statically scheduled superscalar processors
 - ▣ VLIW (very long instruction word) processors
 - ▣ dynamically scheduled superscalar processors

Multiple Issue

Common name	Issue structure	Hazard detection	Scheduling	Distinguishing characteristic	Examples
Superscalar (static)	Dynamic	Hardware	Static	In-order execution	Mostly in the embedded space: MIPS and ARM, including the ARM Coretex A8
Superscalar (dynamic)	Dynamic	Hardware	Dynamic	Some out-of-order execution, but no speculation	None at the present
Superscalar (speculative)	Dynamic	Hardware	Dynamic with speculation	Out-of-order execution with speculation	Intel Core i3, i5, i7; AMD Phenom; IBM Power 7
VLIW/LIW	Static	Primarily software	Static	All hazards determined and indicated by compiler (often implicitly)	Most examples are in signal processing, such as the TI C6x
EPIC	Primarily static	Primarily software	Mostly static	All hazards determined and indicated explicitly by the compiler	Itanium

VLIW Processors

- Package multiple operations into one instruction
- Example VLIW processor:
 - One integer instruction (or branch)
 - Two independent floating-point operations
 - Two independent memory references
- Must be enough parallelism in code to fill the available slots

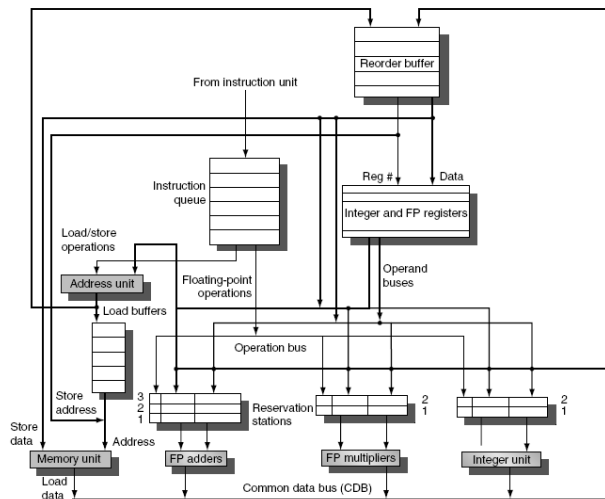
VLIW Processors

- Disadvantages:
 - Statically finding parallelism
 - Code size
 - No hazard detection hardware
 - Binary code compatibility

Dynamic Scheduling, Multiple Issue, and Speculation

- Modern microarchitectures:
 - Dynamic scheduling + multiple issue + speculation
- Two approaches:
 - Assign reservation stations and update pipeline control table in half clock cycles
 - Only supports 2 instructions/clock
 - Design logic to handle any possible dependencies between the instructions
 - Hybrid approaches
- Issue logic can become bottleneck

Overview of Design



Multiple Issue

- Limit the number of instructions of a given class that can be issued in a “bundle”
 - I.e. on FP, one integer, one load, one store

- Examine all the dependencies among the instructions in the bundle

- If dependencies exist in bundle, encode them in reservation stations

- Also need multiple completion/commit

Example

```
Loop: LD R2,0(R1)      ;R2=array element
      DADDIU R2,R2,#1  ;increment R2
      SD R2,0(R1)     ;store result
      DADDIU R1,R1,#8  ;increment pointer
      BNE R2,R3,LOOP  ;branch if not last element
```

Example (No Speculation)

Iteration number	Instructions	Issues at clock cycle number	Executes at clock cycle number	Memory access at clock cycle number	Write CDB at clock cycle number	Comment
1	LD R2,0(R1)	1	2	3	4	First issue
1	DADDIU R2,R2,#1	1	5		6	Wait for LW
1	SD R2,0(R1)	2	3	7		Wait for DADDIU
1	DADDIU R1,R1,#8	2	3		4	Execute directly
1	BNE R2,R3,LOOP	3	7			Wait for DADDIU
2	LD R2,0(R1)	4	8	9	10	Wait for BNE
2	DADDIU R2,R2,#1	4	11		12	Wait for LW
2	SD R2,0(R1)	5	9	13		Wait for DADDIU
2	DADDIU R1,R1,#8	5	8		9	Wait for BNE
2	BNE R2,R3,LOOP	6	13			Wait for DADDIU
3	LD R2,0(R1)	7	14	15	16	Wait for BNE
3	DADDIU R2,R2,#1	7	17		18	Wait for LW
3	SD R2,0(R1)	8	15	19		Wait for DADDIU
3	DADDIU R1,R1,#8	8	14		15	Wait for BNE
3	BNE R2,R3,LOOP	9	19			Wait for DADDIU

Example

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5		6	7	Wait for LW
1	SD R2,0(R1)	2	3		7	7	Wait for DADDIU
1	DADDIU R1,R1,#8	2	3		4	8	Commit in order
1	BNE R2,R3,LOOP	3	7			8	Wait for DADDIU
2	LD R2,0(R1)	4	5	6	7	9	No execute delay
2	DADDIU R2,R2,#1	4	8		9	10	Wait for LW
2	SD R2,0(R1)	5	6			10	Wait for DADDIU
2	DADDIU R1,R1,#8	5	6		7	11	Commit in order
2	BNE R2,R3,LOOP	6	10			11	Wait for DADDIU
3	LD R2,0(R1)	7	8	9	10	12	Earliest possible
3	DADDIU R2,R2,#1	7	11		12	13	Wait for LW
3	SD R2,0(R1)	8	9			13	Wait for DADDIU
3	DADDIU R1,R1,#8	8	9		10	14	Executes earlier
3	BNE R2,R3,LOOP	9	13			14	Wait for DADDIU