HY425 Lecture 04: Dynamic Scheduling with Renaming, Tomasulo's Algorithm

Dimitrios S. Nikolopoulos

University of Crete and FORTH-ICS

October 17, 2011

Scoreboard instruction execution steps

- Issue – decode and check for structural hazards
  - Instructions issued strictly in program order
  - Instruction not issued if structural hazards
  - Instruction not issued if output-dependent on earlier instruction
- Read operands – wait until no hazards (data, structural) read operands
  - True dependences resolved in this stage
  - No value forwarding, result communicated through registers

Scoreboard instruction execution steps

- Execute – functional unit begins execution, notifies scoreboard upon completion
- Write result – stall if WAR hazards are violated (reordering), otherwise write result to register file

Scoreboard limitations

- Parallelism available among instructions in a single basic block
- Instruction window – number of scoreboard entries available to hold instructions and look for ILP
- Number and type of functional units – structural hazards
- Anti- and output-dependences – both stall the scoreboard on write-back
- Relies heavily on compiler to schedule instructions

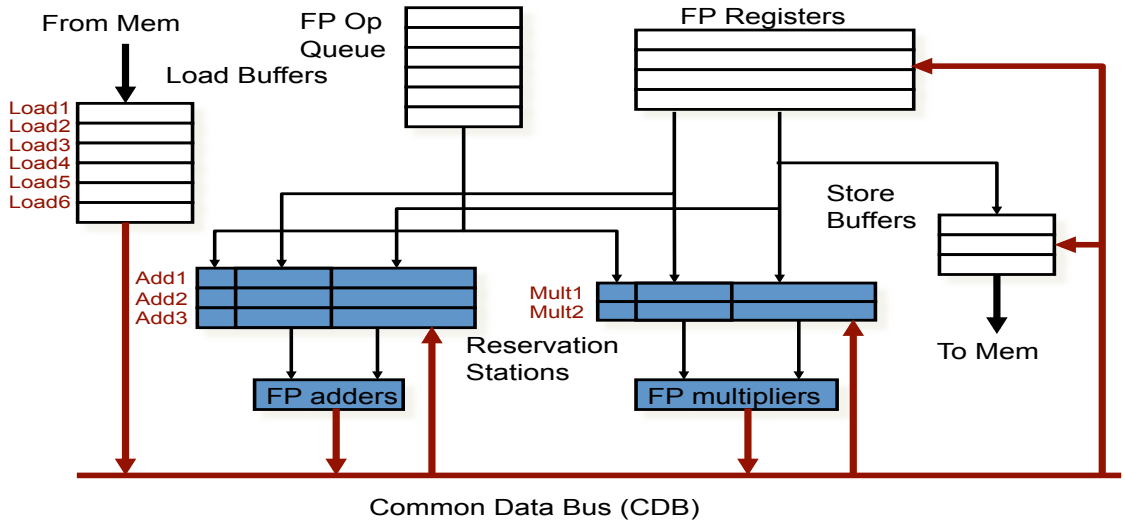
Tomasulo's algorithm key ideas

- Register renaming to avoid WAR and WAW hazards
  - Use alternative destination register instead of stalling at WB
- Operands are communicated to instructions through reservation stations, instead of registers visible to the programmer
  - Reservation stations are operand buffers – can also be seen as additional hidden registers
  - A form of forwarding logic

Tomasulo's algorithm key ideas

- Less dependent on compiler support than scoreboard
  - If more reservation stations than registers, hazards can be eliminated without compiler support
- Hazard detection logic and execution are distributed instead of centralized
  - Reservation stations per FU control execution and receive operands through a common data bus
  - Less contention between waiting instructions at register file

Tomasulo's organization



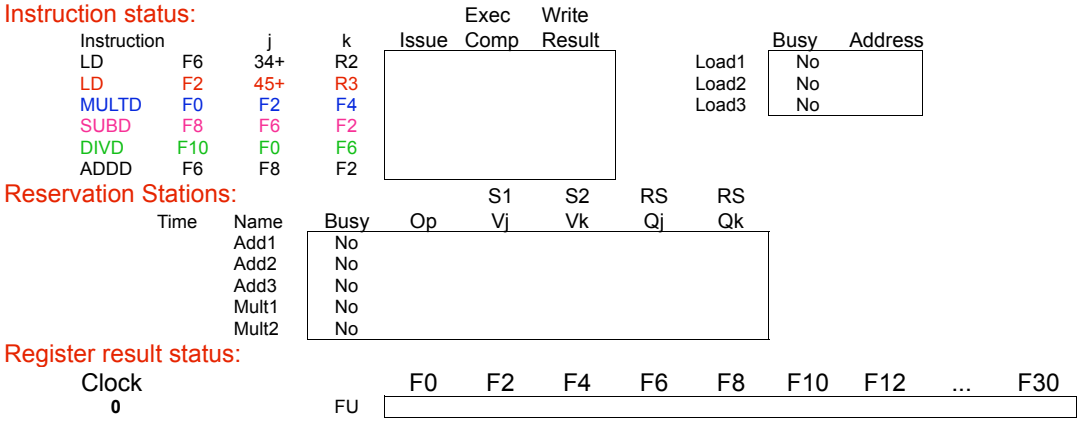
Reservation stations

Op – Operation to perform in the FU
V_j, V_k – Value of source operands
Q_j, Q_k – Reservations stations producing the corresponding source operands. Only one Q or V field is valid for each operand
busy – Indicates that reservation station and accompanying FU are busy

Instruction execution stages in Tomasulo's algorithm

Issue – get instruction from FP operation queue. If reservation station free in corresponding FU (no structural hazard) issue instruction and send operands if in registers. Perform register renaming (assign value if operand available, reservation station ID if operand not available)
Execute – if one or both operands not available monitor common data bus for operand. When all operands available execute instruction. Prevents RAW hazards.
Write result (commit) – Write result on common data bus and from there to registers and any FUs waiting for the result.
Common data bus transmits value plus source reservation station to resolve dependences, instead of value plus destination (typical bus operation)

Tomasulo example



Tomasulo example – cycle 1

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1		
LD	F2	45+	R3			
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	Yes	34+R2
Load2	No	
Load3	No	

Reservation Stations:

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

Register result status:

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

Tomasulo example – cycle 2

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1		
LD	F2	45+	R3	2		
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

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

Reservation Stations:

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

Register result status:

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

Tomasulo example – cycle 3

- Registers renamed, MULTD issued

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	
LD	F2	45+	R3	2		
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

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

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS 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	FU	Mult1	Load2		Load1					

Tomasulo example – cycle 4

- Load2 completes and releases Mult1 and Add1

Instruction status:

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

	Busy	Address
Load1	No	
Load2	Yes	45+R3
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	Yes	SUBD	M(34+R2)			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
4	FU	Mult1	Load2		M(34+R2)	Add1				

Tomasulo example – cycle 5

Instruction status:

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

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
2	Add1	Yes	SUBD	M(34+R2)	M(45+R3)		
	Add2	No					
	Add3	No					
10	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
5	FU	Mult1	M(45+R3)		M(34+R2)	Add1	Mult2			

Tomasulo example – cycle 6

- ADDD issues versus scoreboard stall

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
1	Add1	Yes	SUBD	M(34+R2)	M(45+R3)		
	Add2	Yes	ADDD		M(45+R3)	Add1	
	Add3	No					
9	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
6	FU	Mult1	M(45+R3)		M(34+R2)	Add1	Mult2			

Tomasulo example – cycle 7

- Add1 completes, releases Add2

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
0	Add1	Yes	SUBD	M(34+R2)	M(45+R3)		
	Add2	Yes	ADDD		M(45+R3)	Add1	
	Add3	No					
8	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
7	FU	Mult1	M(45+R3)		M(34+R2)	Add1	Mult2			

Tomasulo example – cycle 8

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
2	Add2	Yes	ADDD	(M-M)	M(45+R3)		
	Add3	No					
7	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
8	FU	Mult1	M(45+R3)		M(34+R2)	(M-M)	Mult2			

Tomasulo example – cycle 9

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
1	Add2	Yes	ADDD	(M-M)	M(45+R3)		
	Add3	No					
6	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
9	FU	Mult1	M(45+R3)		M(34+R2)	(M-M)	Mult2			

Tomasulo example – cycle 10

- Add2 finishes, no waiting RS

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
0	Add2	Yes	ADDD	(M-M)	M(45+R3)		
	Add3	No					
5	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
10	FU	Mult1	M(45+R3)		M(34+R2)	(M-M)	Mult2			

Tomasulo example – cycle 11

- ADDD can proceed and write result, no WAR hazard

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
11	FU	Mult1	M(45+R3)		(M-M+M)	(M-M)	Mult2			

Tomasulo example – cycle 12

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
3	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
12	FU	Mult1	M(45+R3)		(M-M+M)	(M-M)	Mult2			

Tomasulo example – cycle 13

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
2	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
13	FU	Mult1	M(45+R3)		(M-M+M)	(M-M)	Mult2			

Tomasulo example – cycle 14

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
1	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
14	FU	Mult1	M(45+R3)		(M-M+M)	(M-M)	Mult2			

Tomasulo example – cycle 15

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
0	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
15	FU	Mult1	M(45+R3)		(M-M+M)	(M-M)	Mult2			

Tomasulo example – cycle 16

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
40	Mult2	Yes	DIVD	M*F4	M(34+R2)		

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
16	FU	M*F4	M(45+R3)		(M-M+M)	(M-M)	Mult2			

Tomasulo example – cycle 55

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
1	Mult2	Yes	DIVD	M*F4	M(34+R2)		

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
55	FU	M*F4	M(45+R3)		(M-M+M)	(M-M)	Mult2			

Tomasulo example – cycle 56

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5	56	
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
0	Mult2	Yes	DIVD	M*F4	M(34+R2)		

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
56	FU	M*F4	M(45+R3)		(M-M+M)	(M-M)	Mult2			

Tomasulo example – cycle 57

- Out-of-order execution, out-of-order completion, faster than scoreboard

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5	56	57
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

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

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
57	FU	M*F4	M(45+R3)		(M-M+M)	(M-M)	Result			

Memory-based hazards

Dependences due to access in same memory location

- Loads and stores can execute out-of-order if targeting different addresses
- Assume store to M follows load to M in program order
  - Attempt to complete store first causes WAR hazard
- Assume load to M follows store to M in program order
  - Attempt to complete load first causes RAW hazard
- Assume store to M follows store to M in program order
  - Attempt to complete second store first causes WAW hazard

Memory-based hazards

- Solution: execute loads, stores in-order using a load-store queue
- Dynamic memory disambiguation
- Execution of a load-store implies effective address calculation
- Loads wait until they reach head of load-store queue to execute
  - Prior stores have completed
  - Future stores follow them in the queue
  - Execute in two steps: compute effective address, read memory

Memory-based hazards

Resolving memory-based dependencies

- Stores wait until they reach head of load-store queue to execute
  - Prior loads have completed
  - Prior stores have completed
  - Execute in one step: compute effective address

Tomasulo control logic

Instruction state	Wait until	Action or bookkeeping

Issue
FP operation
	Station *r* empty	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].Qi=r;

Load or Store	Buffer r empty	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;
Load only		RegisterStat[rt].Qi=r;
Store only		if (RegisterStat[rt].Qi ≠ 0) {Rs[r].Qk ← RegisterStat[rs].Qi } else { Rs[r].Vk ← Regs[rt]; Rs[r].Qk ← 0 };
Execute FP operation	(Rs[r].Qj = 0) and (Rs[r].Qk=0)	Compute result; operands in Vj and Vk
Load-store step 1	Rs[r].Qj=0 & r is head of load-store queue	Rs[r].A ← Rs[r].Vj + Rs[r].A;
Load step 2	Load step 1 complete	Read from Mem[Rs[r].A]
Write result FP operation or load	Execution complete at r & CDB available	∀x (if (RegisterStat[x].Qi=r) {Regs[x] ← result; RegisterStat[x].Qi ← 0}); ∀x (if (Rs[x].Qj=r) {Rs[x].Vj ← result; Rs[x].Qj ← 0}); ∀x (if (Rs[x].Qk=r) {Rs[x].Vk ← result; Rs[x].Qk ← 0}); Rs[r].Busy ← no;
Store	Execution complete at r & Rs[r].Qk=0	Mem[Rs[r].A] ← Rs[r].Vk; Rs[r].Busy ← no;

Dynamic scheduling strengths and limitations

- Strengths:
  - High performance, if instruction window is sufficiently large
  - Depends less on sophisticated compiler support
  - Good for architectures where it is hard to optimally schedule code, or have few registers
- Limitations:
  - Requires advanced techniques to increase instruction window, most notably branch prediction (e.g. to find instructions across the iterations of a parallel loop)
  - High complexity, therefore high power consumption. Complex control logic for reservation stations, plus need high-speed associative searches.
  - Single common data bus can be a bottleneck

Improving ILP in HW/SW

- Branch prediction (next topic): Uncovering more instructions to overlap by predicting if branches or taken or not taken and reducing branch stall impact
- Multiple issue: Allow processor to issue and hopefully complete more than one instructions per cycle

Improving ILP in HW/SW

- Hardware speculation: Allow processor to execute instructions without knowing the outcome of branches (speculatively)
- Combined techniques: Multiple issue with dynamic scheduling, multiple issue with static scheduling, multiple issue with branch prediction, multiple issue with speculation