Book keeping in Tomasulo's algorithm

- **Instruction status**:
  - issued, executing or writing result,
- **Reservation stations (functional units) status**:
  - busy/not busy
  - operation (if unit can perform more than one operation)
  - source operands (data values) - V_j and V_k
  - the *reservation stations* producing the source operands (if stall to avoid RAW hazards) - Q_j and Q_k
  - Address field, A, for load/store buffers (store effective address)
- **Register result status**:
  - indicates the *reservation station* that contains an instruction which will write into each register (if any)

Note that the first two tables can be combined

(32)

Three stages of control

- **Issue**
  - If a reservation station is available for the needed functional unit
    - » read ready operands
    - » for operands that are not ready, rename the register to the reservation station that will produce it,
  - Reservation stations for load/store instructions are called load/store buffers.
- **Execution.**
  - Monitor the CDB for the operand that is not ready,
  - When both operands are available, execute.
  - If more than one station per unit, only one unit can start execution.
  - Do not start execution before previous branches have completed.
- **Write result.**
  - Write to CDB (and to registers) -- may be a structural hazards if only one CDB bus.
  - Make the reservation station (the functional unit) available.

(33)

Load and store instructions:

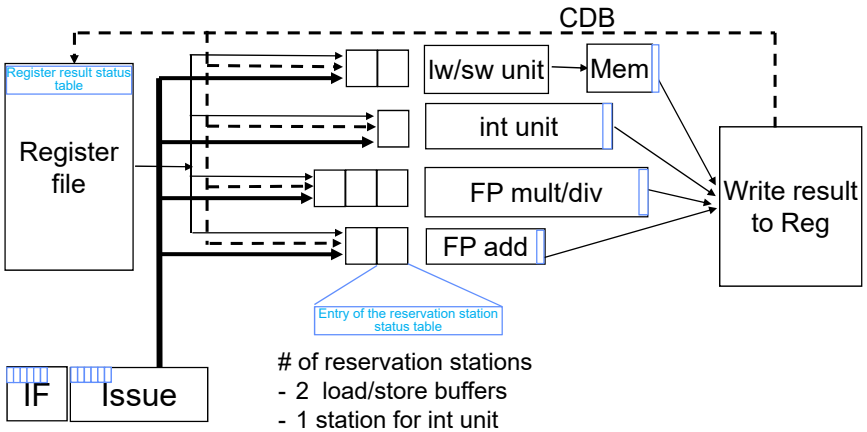
- Uses load/store buffers, and each buffer is like a reservation station.
- Address calculation (put result in buffer), then memory operation
- The result of a load is put on the CDB
- Stores are executed in program order (loads in any order)
- Performs memory disambiguation between store and load buffers,

Remarks:

- May have more reservation stations than registers (a large virtual register space)
- The original Tomasulo algorithm was introduced before caches were incorporated into commercial processors
- If more than one issued instruction writes into a register, only the last one does the actual write (no WAW hazards).

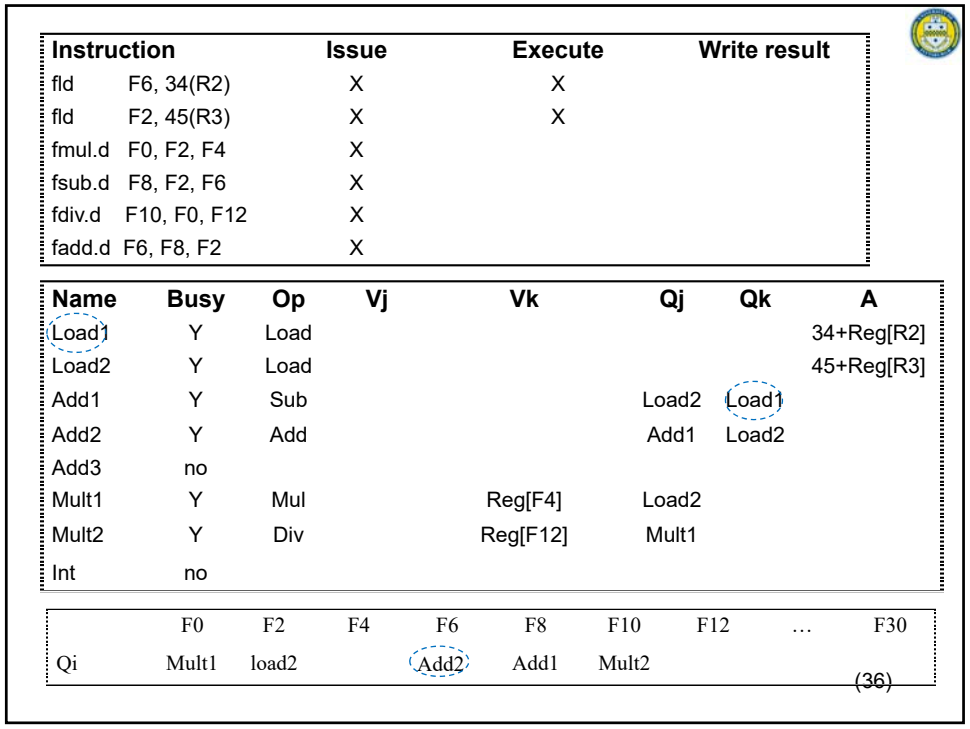
(34)

Example



# of reservation stations
- 2 load/store buffers
- 1 station for int unit
- 3 station for FP mult/div unit
- 2 station for FP add unit

(35)

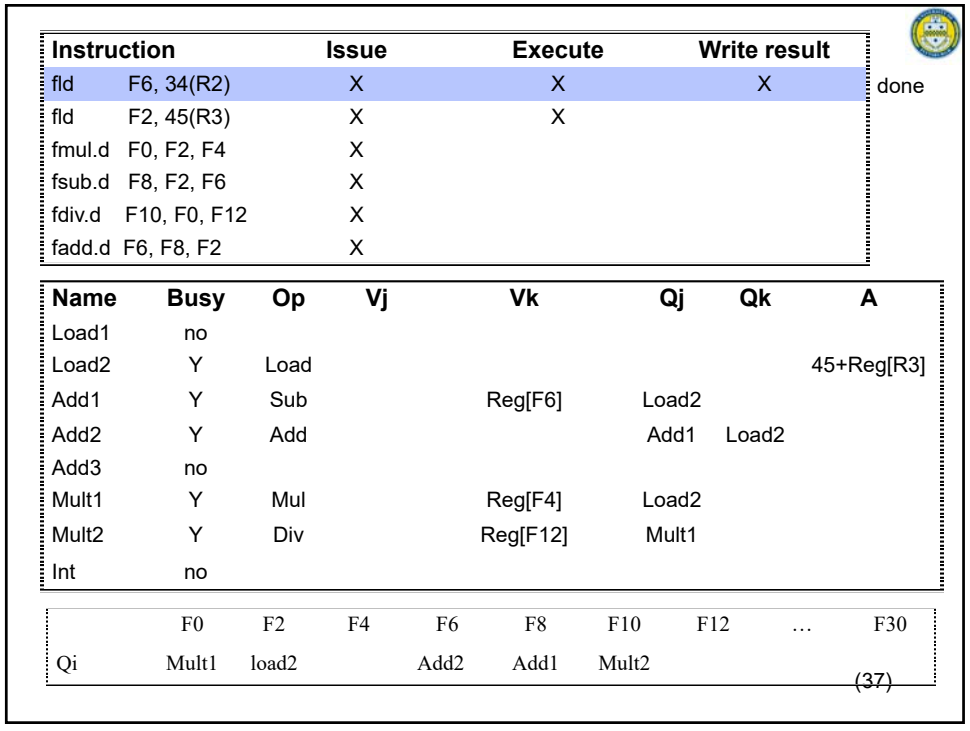


Instruction		Issue	Execute	Write result
fld	F6, 34(R2)	X	X	
fld	F2, 45(R3)	X	X	
fmul.d	F0, F2, F4	X		
fsub.d	F8, F2, F6	X		
fdiv.d	F10, F0, F12	X		
fadd.d	F6, F8, F2	X		

Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	Y	Load					34+Reg[R2]
Load2	Y	Load					45+Reg[R3]
Add1	Y	Sub			Load2	Load1	
Add2	Y	Add			Add1	Load2	
Add3	no						
Mult1	Y	Mul		Reg[F4]	Load2		
Mult2	Y	Div		Reg[F12]	Mult1		
Int	no						

	F0	F2	F4	F6	F8	F10	F12	…	F30
Qi	Mult1	load2		Add2	Add1	Mult2			

(36)

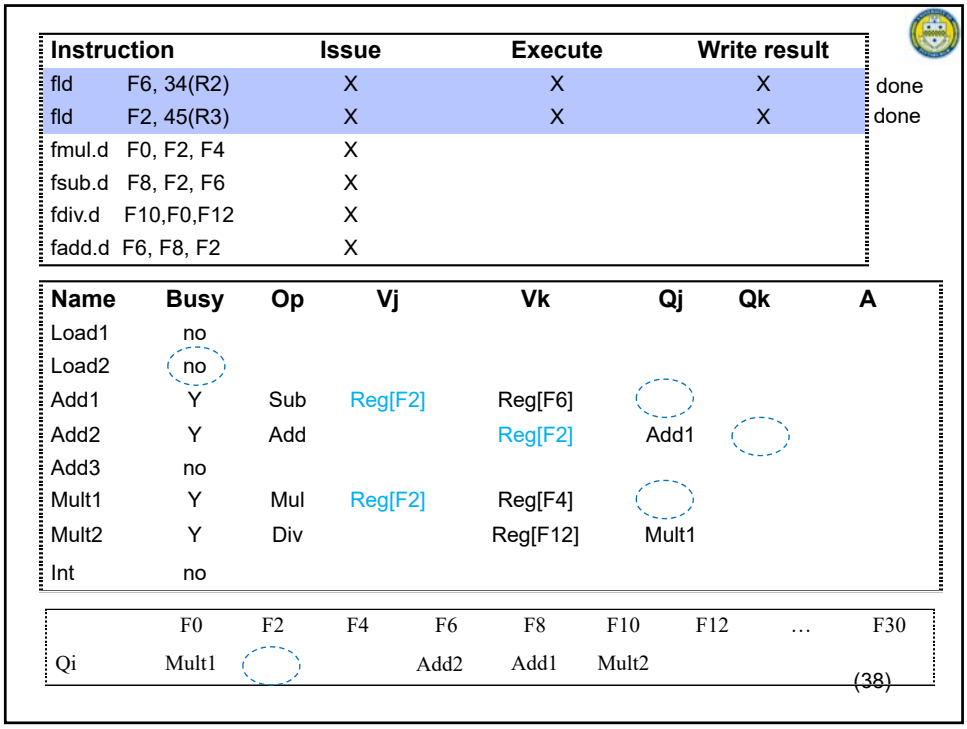


Instruction		Issue	Execute	Write result	
fld	F6, 34(R2)	X	X	X	done
fld	F2, 45(R3)	X	X		
fmul.d	F0, F2, F4	X			
fsub.d	F8, F2, F6	X			
fdiv.d	F10, F0, F12	X			
fadd.d	F6, F8, F2	X			

Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	no						
Load2	Y	Load					45+Reg[R3]
Add1	Y	Sub		Reg[F6]	Load2		
Add2	Y	Add			Add1	Load2	
Add3	no						
Mult1	Y	Mul		Reg[F4]	Load2		
Mult2	Y	Div		Reg[F12]	Mult1		
Int	no						

	F0	F2	F4	F6	F8	F10	F12	…	F30
Qi	Mult1	load2		Add2	Add1	Mult2			

(37)

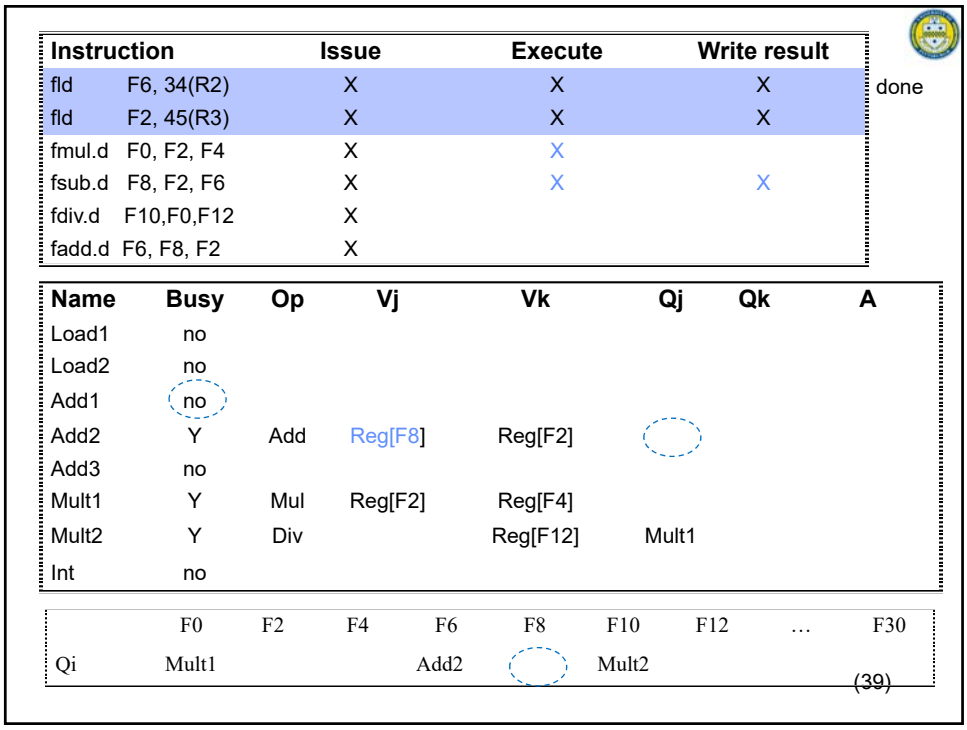


Instruction		Issue	Execute	Write result	
fld	F6, 34(R2)	X	X	X	done
fld	F2, 45(R3)	X	X	X	done
fmul.d	F0, F2, F4	X			
fsub.d	F8, F2, F6	X			
fdiv.d	F10,F0,F12	X			
fadd.d	F6, F8, F2	X			

Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	no						
Load2	no						
Add1	Y	Sub	Reg[F2]	Reg[F6]			
Add2	Y	Add		Reg[F2]	Add1		
Add3	no						
Mult1	Y	Mul	Reg[F2]	Reg[F4]			
Mult2	Y	Div		Reg[F12]	Mult1		
Int	no						

	F0	F2	F4	F6	F8	F10	F12	…	F30
Qi	Mult1			Add2	Add1	Mult2			

(38)



Instruction		Issue	Execute	Write result	
fld	F6, 34(R2)	X	X	X	done
fld	F2, 45(R3)	X	X	X	
fmul.d	F0, F2, F4	X	X		
fsub.d	F8, F2, F6	X	X	X	
fdiv.d	F10,F0,F12	X			
fadd.d	F6, F8, F2	X			

Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	no						
Load2	no						
Add1	no						
Add2	Y	Add	Reg[F8]	Reg[F2]			
Add3	no						
Mult1	Y	Mul	Reg[F2]	Reg[F4]			
Mult2	Y	Div		Reg[F12]	Mult1		
Int	no						

	F0	F2	F4	F6	F8	F10	F12	…	F30
Qi	Mult1			Add2		Mult2			

(39)

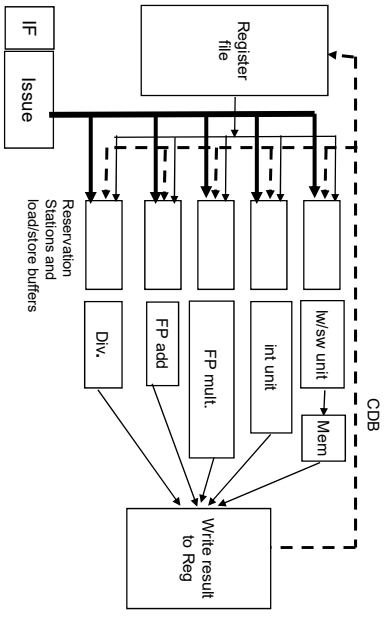
Instruction		Issue	Execute	Write result	
fld	F6, 34(R2)	X	X	X	done
fld	F2, 45(R3)	X	X	X	done
fmul.d	F0, F2, F4	X	X		
fsub.d	F8, F2, F6	X	X	X	done
fdiv.d	F10,F0,F12	X			
fadd.d	F6, F8, F2	X	X	X	

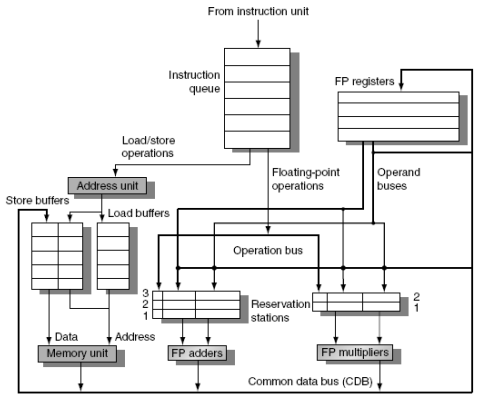
Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	no						
Load2	no						
Add1	no						
Add2	no						
Add3	no						
Mult1	Y	Mul	Reg[F2]	Reg[F4]			
Mult2	Y	Div		Reg[F12]	Mult1		
Int	no						

	F0	F2	F4	F6	F8	F10	F12	…	F30
Qi	Mult1					Mult2			

(40)

Figure 3.10





(41)

Another example

Instruction		Issue	Execute	Write result
fld	F0, 0(R1)	X	X	
fmul.d	F4, F0, F2	X		
fsd	F4, 0(R1)	X		
fld	F0, 8(R1)			
fmul.d	F4, F0, F2			
fsd	F4, 8(R1)			

Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	y	ld					Reg[R1]+0
Load2	no						
store1	y	sd	Reg[R1]			Mult1	
store2	no						
Add	no						
Mult1	y	Mult		Reg[F2]	Load1		
Mult2	no						

	F0	F2	F4	F6	F8	F10	F12	…	F30
Qi	Load1		Mult 1						

Another example

Instruction		Issue	Execute	Write result
fld	F0, 0(R1)	X	X	
fmul.d	F4, F0, F2	X		
fsd	F4, 0(R1)	X		
fld	F0, 8(R1)	X	X	
fmul.d	F4, F0, F2	X		
fsd	F4, 8(R1)	X		

Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	y	ld					Reg[R1]+0
Load2	y	ld					Reg[R1]+8
store1	y	sd				Mult1	Reg[R1]+0
store2	y	sd	Reg[R1]			Mult2	
Add	no						
Mult1	Y	Mul		Reg[F2]	Load1		
Mult2	Y	Mul		Reg[F2]	Load2		

	F0	F2	F4	F6	F8	F10	F12	…	F30
Qi	Load2		Mult 2						

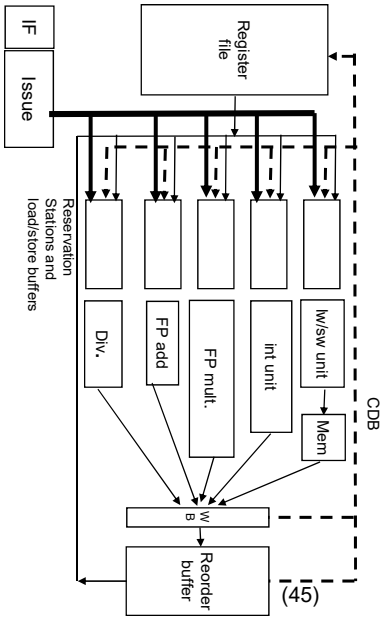
Hardware-based Speculation (§3.6)

- The goal is to allow instructions after a branch to start execution before the outcome of the branch is confirmed.
- There should be *no* consequences (including exceptions) if it is determined that the instruction should not execute.
  - Use dynamic branch prediction and use OOO execution
  - Use a Reorder Buffer (ROB) to reorder instructions after execution
  - Commit results to registers and memory in-order
  - All un-committed results can be flushed if a branch is miss-predicted
  - Service interrupts only when an instruction is ready to commit
- Can free the reservation station when an instruction is in the reorder buffer.
- For each register, R, the register status table keeps the ROB number reserved by the instruction which will write into R (instead of the RES station number).

(44)

Reorder buffers (ROB)

- 3 fields: instruction, destination, value
- When issuing a new instruction, read a register value from the ROB if the status table indicates that the source instruction is in a ROB.
- Hence, ROBs supply operands between execution complete & commit => more virtual registers.
- ROBs form a circular queue ordered in the "issue order".
- Once an instruction reaches the head of the ROB queue, it commits the results into register or memory.
- Hence, it's easy to undo speculated instructions on miss-predicted branches or on exceptions
- Should flush the pipe as soon as you discover a miss-predictions – all earlier instructions should commit (problems??)



(45)

Steps of Speculative Tomasulo Algorithm

Combining branch prediction with dynamic scheduling

- **Issue** (sometimes called Dispatch)
  - If a RES station and a ROB are free, issue the instruction to the RES station after reading ready registers and renaming non-ready registers
- **Execution** (sometimes called issue)
  - When both operands are ready, then execute; if not ready, watch CDB for result; when both in reservation station, execute (checks RAW)
- **Write result** (WB)
  - Write on CDB to all awaiting RES stations & send the instruction to the ROB; mark reservation station available.
- **Commit** (sometimes called graduation)
  - When instruction is at head of ROB, update registers (or memory) with result and free ROB. A miss-predicted branch flushes all non-committed instructions.

(46)

Example:

- Assume that *fmul.d* just finished WB and is entering the ROB
- *fsub.d* and *fadd.d* are already in the ROB
- *fdiv.d* is in RES station waiting for the result of *fmul.d*

Instruction	Issued	Execute	In ROB	committed	
fld F6, 34(R2)	X	X	X	X	done
fld F2, 45(R3)	X	X	X	X	done
fmul.d F0, F2, F4	X	X			
fsub.d F8, F2, F6	X	X	X		
fdiv.d F10, F0, F6	X				
fadd.d F6, F8, F2	X	X	X		

(47)

Reservation stations status

Name	Busy	Op	Vj	Vk	Qj	Qk	Dest.	A
Load1	no							
Load2	no							
Add1	Y	sub	Reg[F2]	Reg[F6]			ROB4	
Add2	Y	add	Reg[F8]	Reg[F2]			ROB6	
Add3	no							
Mult1	Y	Mul	Reg[F2]	Reg[F4]			ROB3	
Mult2	Y	Div		Reg[F0]	ROB6		ROB5	

ROB status

Name	Busy	Instruction		State	Dest.	value
ROB1	no	fld	F6, 34(R2)	Commit	F6	xxx
ROB2	no	fld	F2, 45(R3)	Commit	F2	xxx
ROB3	yes	fmul.d	F0, F2, F4	Wrote result	F0	xxx
ROB4	yes	fsub.d	F8, F2, F6	Wrote result	F8	xxx
ROB5	yes	fdiv.d	F10, F0, F6	Issued/executing	F10	
ROB6	yes	fadd.d	F6, F8, F2	Wrote result	F6	xxx

Register status

	F0	F2	F4	F6	F8	F10	F12	…	F30
Qi	ROB3			ROB6	ROB4	ROB5			

Note: reservation stations Add1 and Add2 may be released after the instructions finish the write back.

(48)