Scoreboard information (3 tables)

- **Instruction status**:
  - issued, read operands and started execution (dispatched), completed execution or wrote result,
- **Functional unit status** (assuming non-pipelined units)
  - busy/not busy
  - operation (if unit can perform more than one operation)
  - destination register - F_i
  - source registers (containing source operands) - F_j and F_k
  - the unit producing the source operands (if stall to avoid RAW hazards) - Q_j and Q_k
  - flags to indicate that source operands are ready -- R_j and R_k
- **Register result status**:
  - indicates the functional unit that contains an instruction which will write into each register (if any)

(24)

Four stages of scoreboard control

- **Issue only if no structural, WAR or WAW hazards.**
  - Issue (and reserve the functional unit) if the functional unit is free and
    - » No issued or dispatched instruction (in state “issued” or “dispatched”) will write to the destination register (to avoid WAW)
    - » No issued instruction (in state “issued”) will read from the destination register (to avoid WAR)
  - otherwise, stall, and block subsequent instructions
  - the fetch unit stalls when the queue between the fetch and the issue stages is full (may be only one buffer).
- **Read operands only if no RAW hazard.**
  - If a RAW hazard is detected, wait until the operands are ready,
  - When the operands are ready, read the registers and move to the execution stage,
  - Note that instructions may proceed to the EX stage out-of-order.
- **Execution.**
  - When execution terminates, notify the score board.
- **Write result to register file**

(25)

Example:

Instruction status

Instruction		Issue	Read op.	Exec. Completed	Write result	
fld	F6, 34(R2)	X	X	X	X	done
fld	F2, 45(R3)	X	X	X		
fmul.d	F0, F2, F4	X				
fsub,d	F8, F6, F2	X				
fdiv.d	F10, F0, F12	X				
fadd.d	F6, F8, F2					Not in

Func. unit status

Unit	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	Load	F2	R3				Yes	
Mult1	Yes	Mult	F0	F2	F4	Int.		No	Yes
Mult2	No								
Add	Yes	Sub	F8	F6	F2		Int.	Yes	No
divide	Yes	Div	F10	F0	F12	Mult1		No	Yes

Register status

	F0	F2	F4	F6	F8	F10	F12	…	F30
Func. U	Mult1	Int.			Add	Div			

Redundant information

(26)

Example: when “fld F2, 45(R3)” is writing

Instruction status

Instruction		Issue	Read op.	Exec. Completed	Write result	
fld	F6, 34(R2)	X	X	X	X	done
fld	F2, 45(R3)	X	X	X	X	
fmul.d	F0, F2, F4	X				
fsub,d	F8, F6, F2	X				
fdiv.d	F10, F0, F12	X				
fld	F6, 34(R2)					Not in

Func. unit status

Unit	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	Load	F2	R3				Yes	
Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
Mult2	No								
Add	Yes	Sub	F8	F6	F2			Yes	Yes
divide	Yes	Div	F10	F0	F12	Mult1		No	Yes

Register status

	F0	F2	F4	F6	F8	F10	F12	…	F30
Func. U	Mult1				Add	Div			

(27)

Example: 3 cycles after “fsub.d” finished writing

Instruction status

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

Func. unit status

Unit	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
Mult2	No								
Add	Yes	add	F4	F8	F2			Yes	Yes
divide	Yes	Div	F10	F0	F12	Mult1		No	Yes

Register status

	F0	F2	F4	F6	F8	F10	F12	…	F30
FU	Mult1		Add			Div			

(28)

Limitations of the scoreboard approach

- No forwarding
- Structural hazards are cleared before instruction “issue”
- WAW and WAR hazards are cleared before instruction “issue”
- Did not discuss control hazards
- Execution units are not pipelined

Possible enhancement

- If we can have “k” write-backs to registers per cycle and “k” parallel buses between registers and pipeline units, them
  - k functional units may be released per cycle
  - k instructions may be dispatched per cycles.
  - k instructions may be issued per cycle.

Need to extend the scoreboard to the case where the execution units are pipelined?

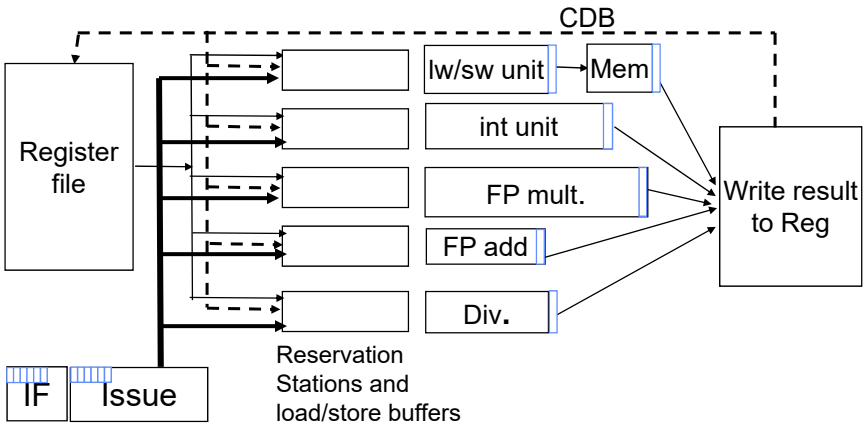
(29)

The Tomasulo approach

- Introduced for IBM 360/91 in 1966
- Main improvements over the scoreboard approach:
  - Uses forwarding on a Common Data Bus (CDB) -- more efficient dealing with RAW hazards,
  - avoids WAR hazards by reading the operands in the instruction-issue order, instead of stalling at issue. To accomplish this an instruction reads an available operand before waiting for the other.
  - Avoids WAW hazards by renaming the registers (using the *id* of a reservation station rather than the register *id*)
  - The control information and logic are distributed to the functional unit and not centralized.

(30)

The architecture for the Tomasulo scheme



- Each reservation station has an *id* and is used by one instruction during the lifetime of this instruction.
- Each unit has one or more reservation stations
- Reservation stations play the role of temporary registers (renaming)

(31)