

Register Renaming

register renaming (in hardware)

- change register names to eliminate WAR/WAW hazards
- one of the most elegant concepts in computer architecture

key: think of architectural registers as names, not locations

- can have more locations than names
- dynamically map names to locations
- **map table** holds the current mappings (name→location)
 - write: allocate new location and record it in map table
 - read: find location of most recent write by name lookup in map table
 - minor detail: must de-allocate locations appropriately

Register Renaming Example

- names: **r1, r2, r3**, locations: **11, 12, 13, 14, 15, 16, 17**
- original mapping: **r1**→**11**, **r2**→**12**, **r3**→**13** (14–17 “free”)

raw instructions	map table	free locations	renamed instructions																		
	<table><tr><th>r1</th><th>r2</th><th>r3</th></tr><tr><td>11</td><td>12</td><td>13</td></tr><tr><td>14</td><td>12</td><td>13</td></tr><tr><td>14</td><td>12</td><td>15</td></tr><tr><td>16</td><td>12</td><td>15</td></tr><tr><td>16</td><td>17</td><td>15</td></tr></table>	r1	r2	r3	11	12	13	14	12	13	14	12	15	16	12	15	16	17	15	14, 15, 16, 17	
r1	r2	r3																			
11	12	13																			
14	12	13																			
14	12	15																			
16	12	15																			
16	17	15																			
add r1, r2, r3		15, 16, 17	add 14, 12, 13																		
sub r3, r2, r1		16, 17	sub 15, 12, 14																		
mul r1, r2, r3		17	mul 16, 12, 15																		
div r2, r1, r3			div 17, 16, 15																		

- renaming removes **WAW/WAR**, leaves **RAW** intact!!
 - Tomasulo’s algorithm (next) implements this concept “in principle”

DS Method #2: Tomasulo's Algorithm

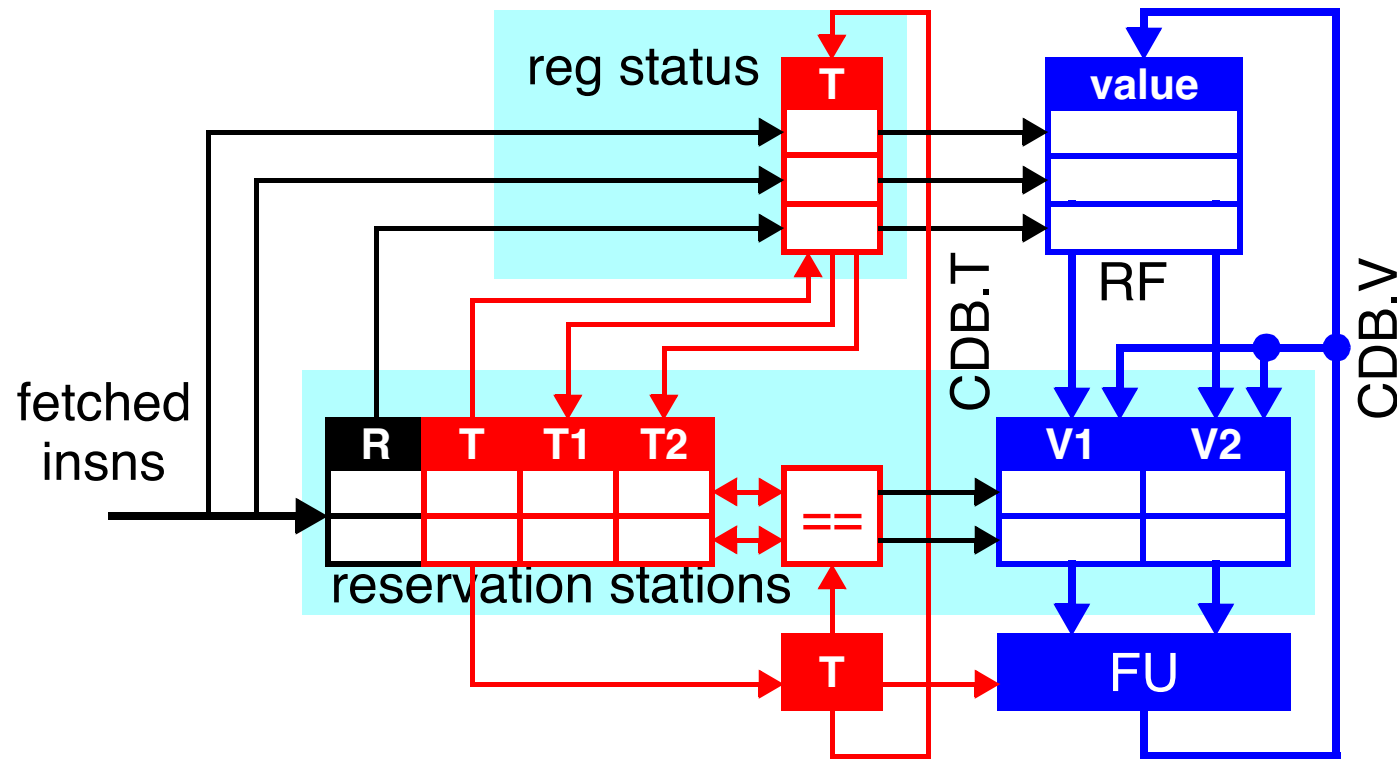
instruction buffer \Rightarrow reservation stations (RS)

- *distributed* control scheme (Scoreboard was centralized)
 - uses data bypassing
 - *common data bus (CDB)* broadcasts results to RS
 - *register renaming* eliminates WAR/WAW hazards
- first implementation: IBM 360/91 [1967]
 - dynamic scheduling for *FP units only*
- our example: Simple Tomasulo
 - dynamic scheduling for *everything*
 - load/store buffers replaced by reservation stations
 - no bypassing (for comparison with Scoreboard)
 - 5 RS: 1 ALU, 1 load, 1 store, 2 FP (3 cycle, pipelined)

Tomasulo Data Structures

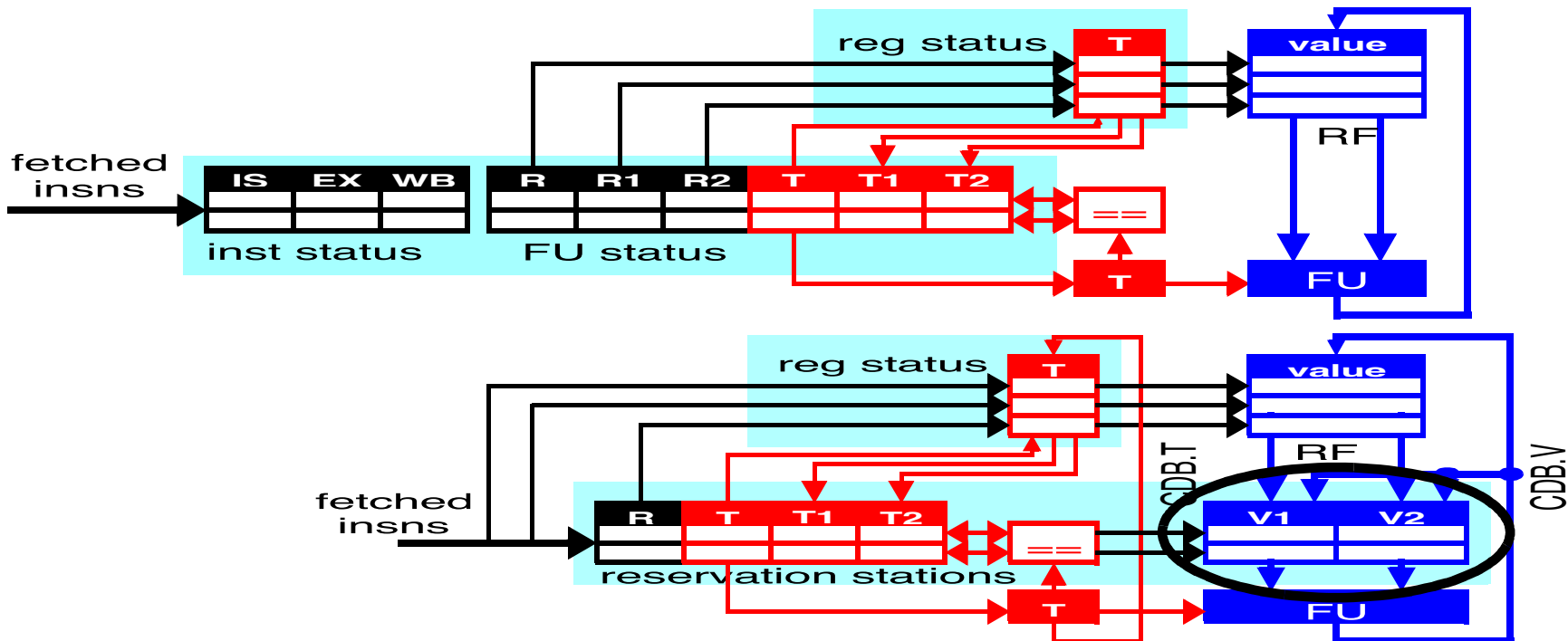
- reservation stations
 - **busy**, **FU**, **op**, **R**: destination register name
 - **T1**, **T2**: source register tag (RS# that will produce the value)
 - **T**: destination register tag (RS# of this RS)
 - **V1**, **V2**: source register value
- register table
 - **T**: tag (RS# that will write register)
- CDB: common data bus
 - broadcasts <value, tag> of completed instructions
- tags interpreted as (more sophisticated) ready bits
 - tag == 0? value is ready (somewhere)
 - tag != 0? value is not ready, wait until CDB broadcasts this tag

Simple Tomasulo



- instruction fields and status bits
- tags
- values

Scoreboard vs. Tomasulo



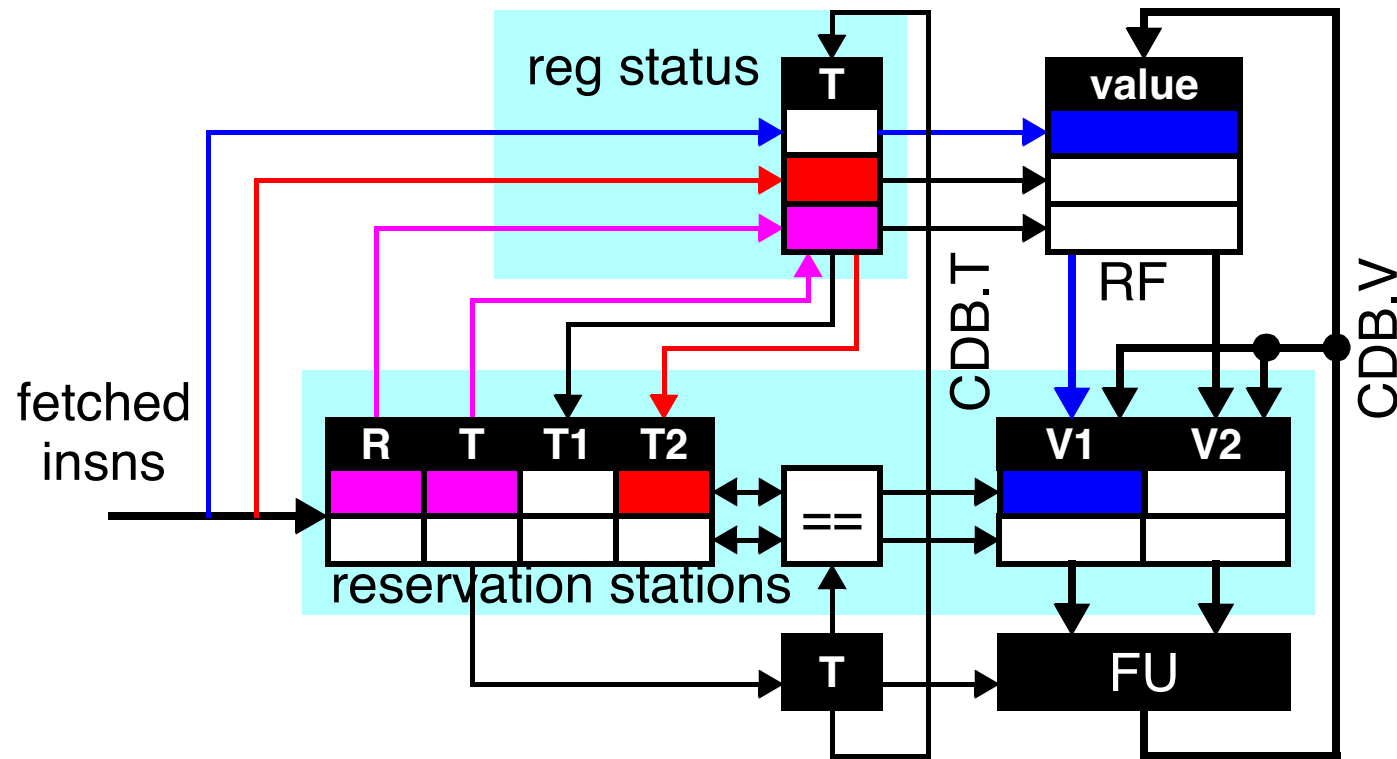
- what about Tomasulo implements register renaming?
 - value copies in reservation stations (RS)
 - instruction holds correct input values in its own RS
 - future instructions can overwrite RF master copy, won't matter!

Tomasulo Pipeline

new pipeline structure: IF, *DS*, *IS*, EX, *WB*

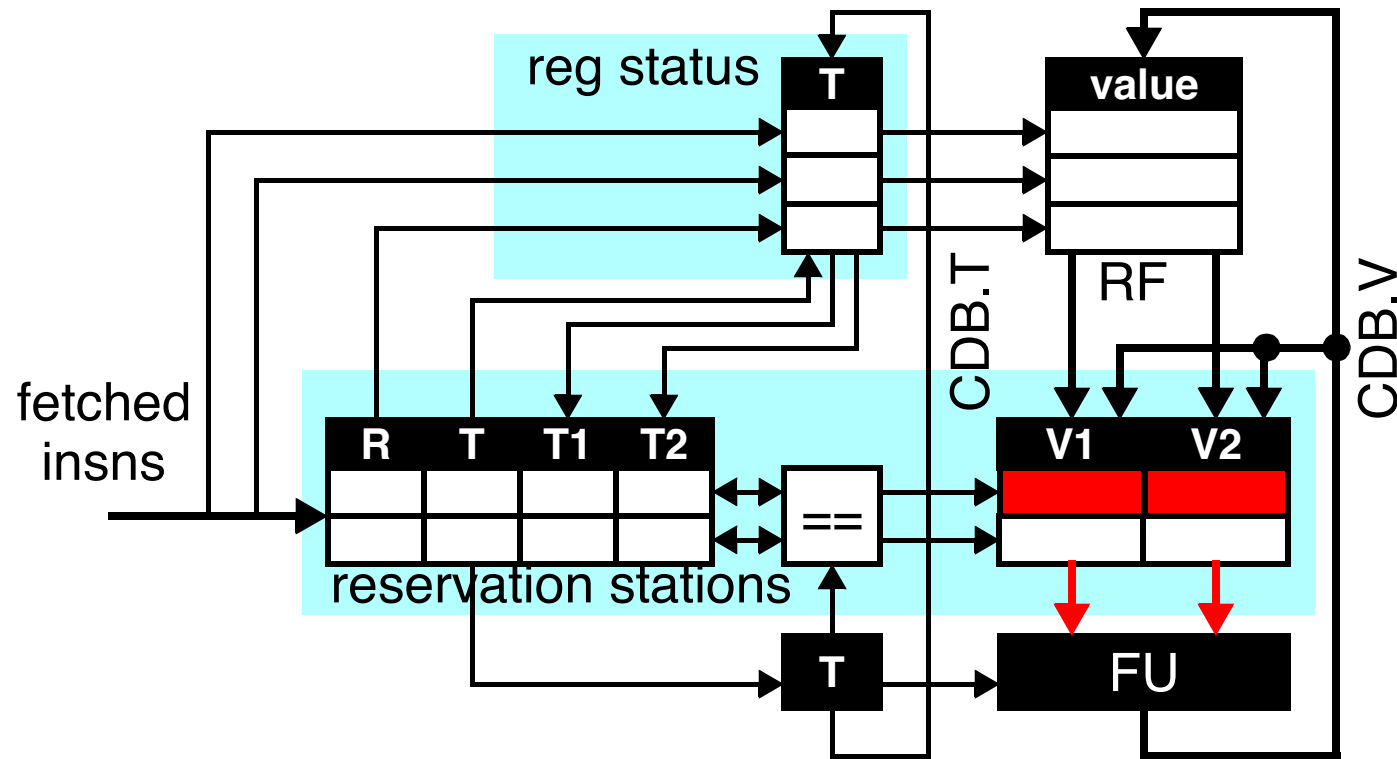
- *DS (dispatch)*
 - (available RS) ?
(allocate RS, *copy ready values*, non-ready tags to RS) : (stall)
- *IS (issue)*
 - (operands ready) ? (execute) : (wait, monitor CDB)
- *WB (writeback)*
 - (CDB available) ? (broadcast result, write reg, free RS) : (wait)
- assume
 - WB and RAW dependent IS can go in same cycle
 - WB and structural dependent DS can go in same cycle

Tomasulo: Dispatch (DS)



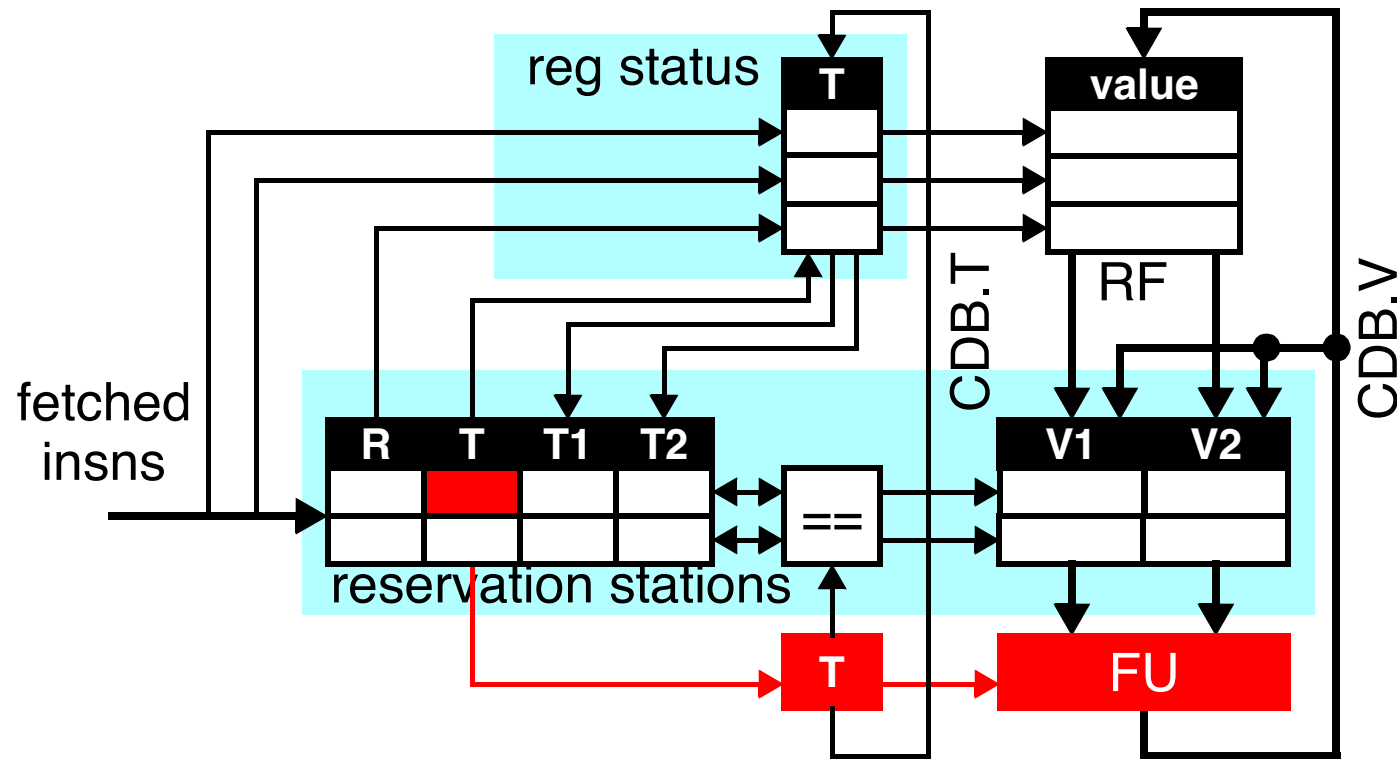
- stall for structural hazards
 - read ready register values into RS
 - read tags for non-ready register values into RS
 - rename (set status for) new result

Tomasulo: Issue (IS)

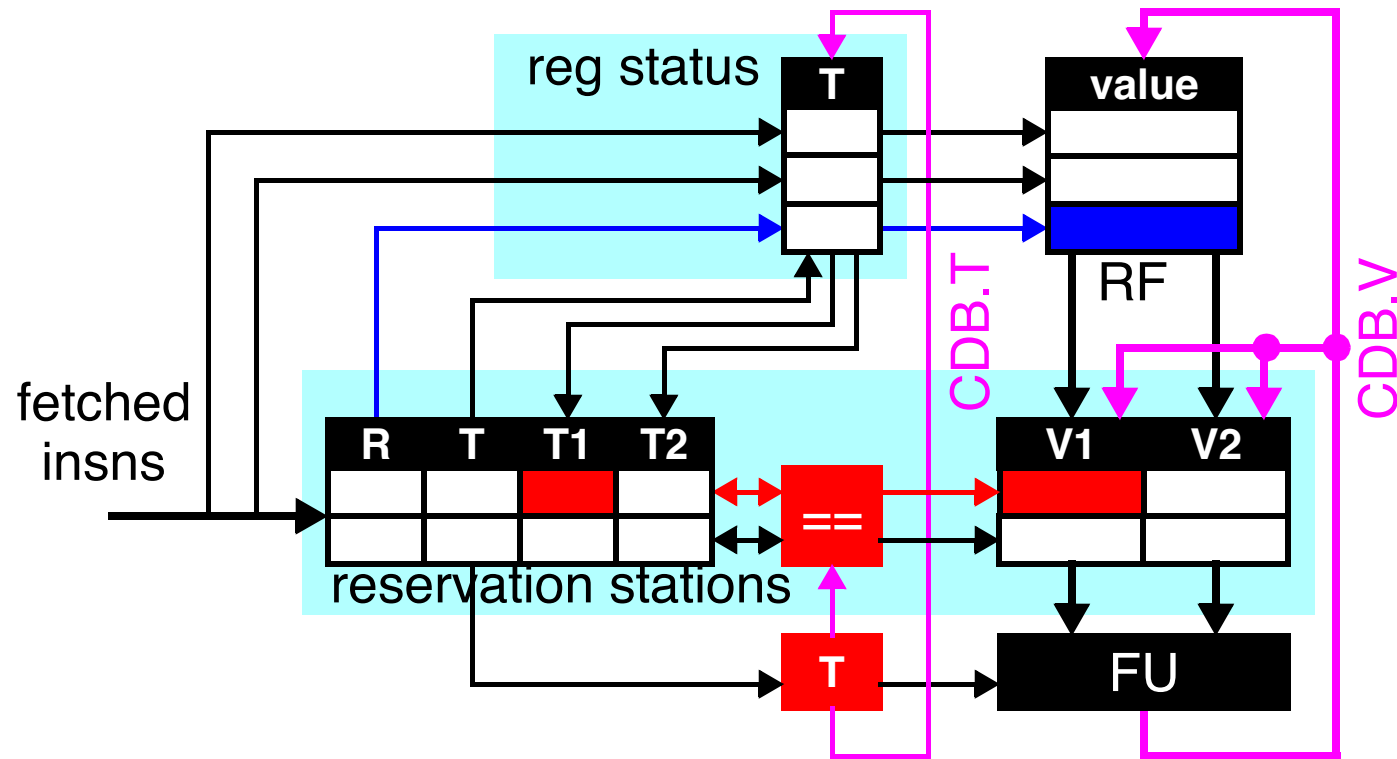


- wait for RAW hazards
 - read register values from reservation stations

Tomasulo: Execute (EX)



Tomasulo: Writeback (WB)



- wait for free CDB
 - broadcast result (value+tag) on CDB
 - write result to register + clear reg status (if tag matches)
 - compare with RS input tags (match? clear tag + copy value)

Tomasulo Data Structures

Instruction Status (illustration only)				
instruction	DS	IS	EX	WB
ldf f0,X(r1)				
mul f4,f0,f2				
stf f4,Z(r1)				
add r1,r1,#4				
ldf f0,X(r1)				
mul f4,f0,f2				
stf f4,Z(r1)				

Reg. Status	
reg	T
f0	
f2	
f4	
r1	

CDB	
reg	T
r1	

Reservation Stations & Load/Store Buffers								
T	FU	busy	op	V1	V2	T1	T2	addr
1	ALU	No						
2	load	No						
3	store	No						
4	FP1	No						
5	FP2	No						

Tomasulo-Style Register Renaming

names: architectural registers

locations: registers in register file AND reservation stations (RS)

- values can (and do) exist in both!
 - copies eliminate WAR hazards
- called “value-based” or “copy-based” renaming

locations referred to internally by **tags**

- register table translates names to tags
 - tag == 0 means “in register file”
 - tag != 0 means “in RS#tag”
- CDB broadcasts values with tags attached
 - so instructions know what value they are looking at