Out-of-Order Execution: Tomasulo's Algorithm

Computer Architecture

Wei Wang

Text Book Chapters

- “Computer Architecture” Hennessy and Patterson, Chapter 3.4 and 3.5 about “Dynamic Scheduling”.

Road Map

Scoreboarding is Limited By

- Number and type of functional units.
- Number of instruction buffer entries (scoreboard size)
- Amount of application ILP (RAW hazards)
- Presence of anti-dependencies (WAR) and output dependencies (WAW)
  - In-order issue for WAW/Structural Hazards limits scheduler.
  - WAR stalls are critical for loops (WAR prevents hardware unrolling in Scoreboarding).

The WAR Limitation of Scoreboarding

- WAR stalls are critical for loops (WAR prevents hardware unrolling in Scoreboarding).
  - For example, for the following two iterations of a loop, instruction 5 cannot be finished due to WAR hazard between instruction 3 and instruction 5, preventing the instructions following it from execution,

```
1  mov R3, [R12+R14]
2  mul R3, R3, 11
3  add R5, R3, 10
4  add R14, R14, 4
5  mov R3, [R12+R14]
6  mul R3, R3, 11
7  add R5, R3, 10
8  add R14, R14, 4
```

- One way to address this WAR limitation is to rename registers.

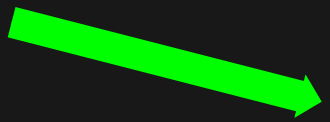
Register Renaming

- Dynamically change register names to eliminate “false dependencies” (WAR/WAW hazards)
- Architectural registers are *Names* not Locations
  - Many more locations (e.g., “reservation stations” or “physical registers”) than names (“logical or architectural registers”)
  - Dynamically map names to locations (e.g., mapping EAX to a inter-stage buffer).

Register Renaming Example

- For example, the previous code can be changed by renaming R3 to R6 to allow two iterations of the loop to executed in parallel (assuming there are enough functional units)

```
1   mov R3, [R12+R14]
2   mul R3, R3, 11
3   add R5, R3, 10
4   add R14, R14, 4
5   mov R3, [R12+R14]
6   mul R3, R3, 11
7   add R5, R3, 10
8   add R14, R14, 4
```



```
1   mov R3, [R12+R14]
2   mul R3, R3, 11
3   add R5, R3, 10
4   add R14, R14, 4
5   mov R6, [R12+R14]
6   mul R6, R6, 11
7   add R6, R6, 10
8   add R14, R14, 4
```

Tomasulo's Approach

- Used in IBM 360/91 Machines (1967)
- Similar to scoreboarding, but added register renaming
- Key concept: Reservation Stations
- Very important in Architecture, since most processors use this algorithm.

Reservation Stations (RS)

- Distributed (rather than centralized) control scheme.
  - Bypassing is allowed via Common Data Bus (CDB) to RS
  - Register Renaming eliminates WAR/WAW hazards
- Scoreboard/Instruction Buffer => Reservation Stations
  - Fetch and buffer operands as soon as available
    - Eliminates the need to always get values from registers
  - Pending instructions designate reservation stations that will provide their inputs
  - Successive writes to a register cause only the last one to update the register

Register Renaming with Tomasulo

- At instruction issue:
  - Registers for source operands are renamed to the names of the registers in reservation stations
  - Values can exist in the registers of the reservation stations or the register file
    - To eliminate WARs, register values are copied to reservation stations at issue
    - Other methods example use pointer-based renaming (map-table)

Reservation Station Components

- *Op*: Operation to perform in the unit
- Q_j, Q_k: Ids of the reservation station entries that produce source values
  - No ready flags needed as in Scoreboard;
  - If index is 0, then the values (V_j or V_k) are ready
  - Store buffers only have Q_i for RS producing result, since stores only need one register.
- V_j, V_k: Values of Source operands
- *Busy*: Indicates reservation station or FU are occupied
- *Register Result Status*: Indicates which functional unit will write each register, if one exists. Blank when no pending instructions that will write that register.

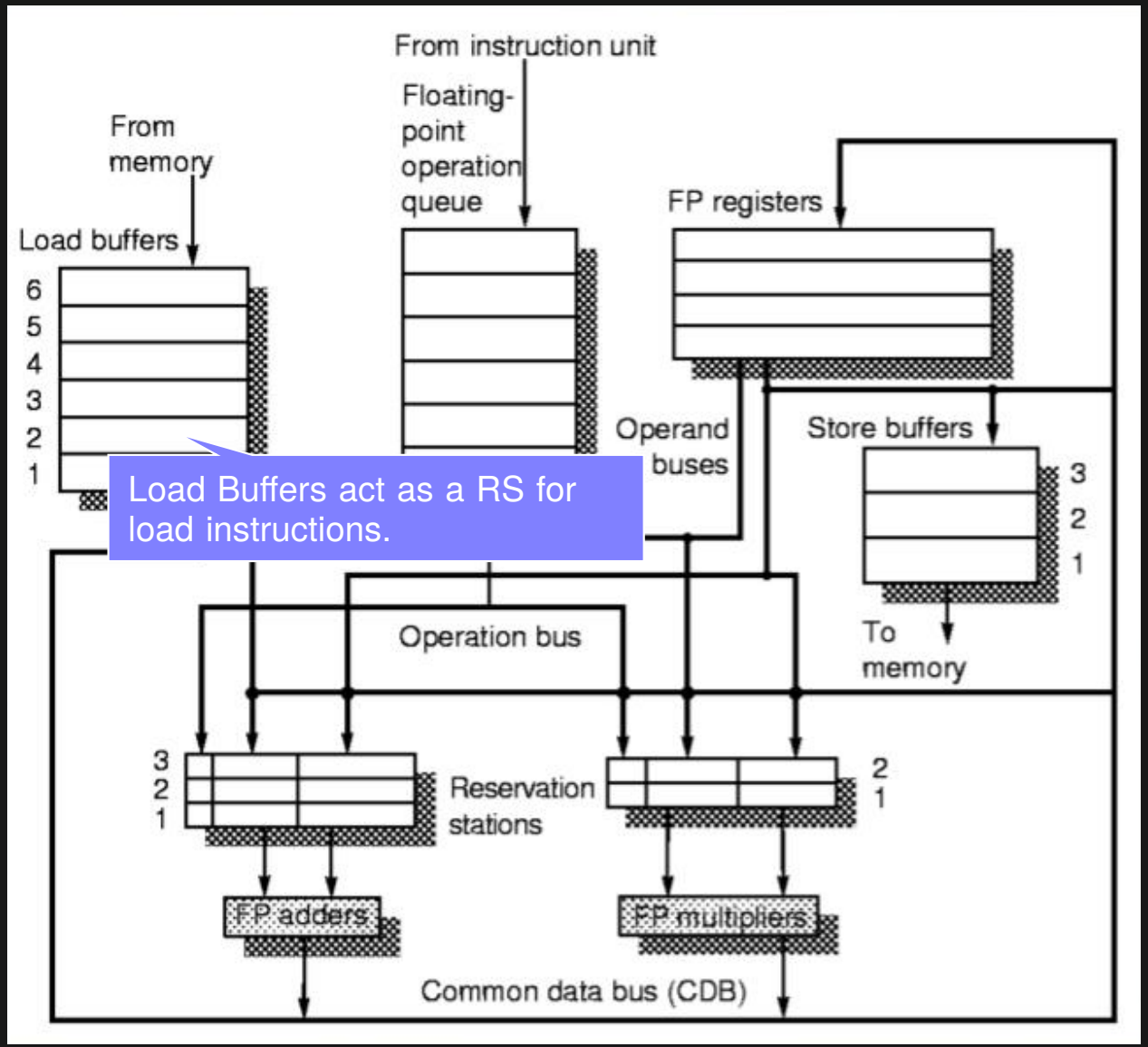
Three Stages of Tomasulo Algorithm

1. Issue (IS): get instruction from instruction queue
   - Issue if there are slots in the reservation station (no structural hazard)
   - Control Unit issues instruction and send operands (rename registers) to reservation stations.
2. Execution (EX): operate on operands
3. Write back (WB): finish execution and write back to registers
   - Send result value to Common Data Bus (CDB)
   - Registers in register file and reservation stations pick up values from CDB based on needs.

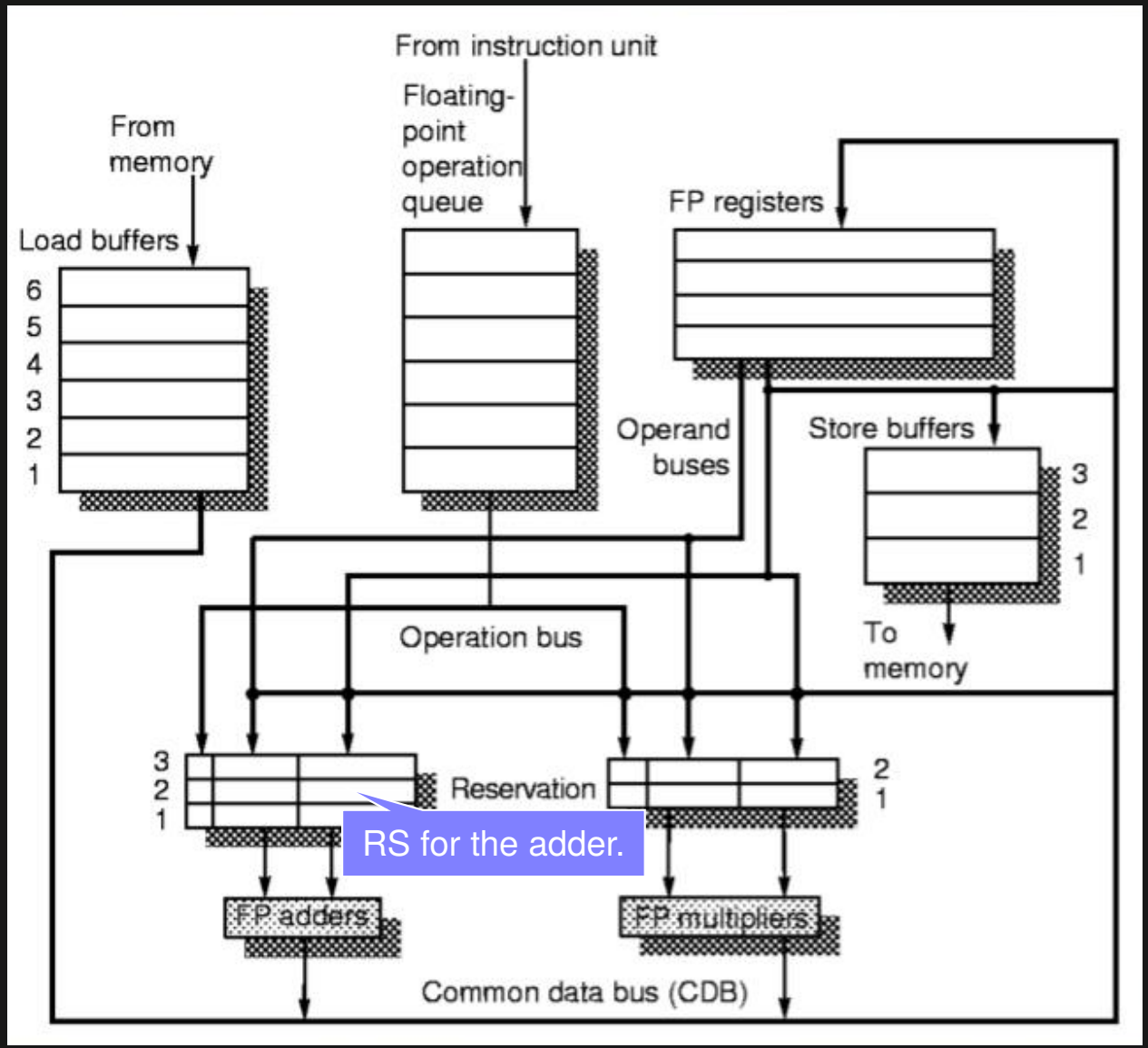
Common Data Bus

- Normal data bus: data + destination (“go to” bus)
- Common data bus: data + source (“come from” bus)
  - 64 bits of data + 4 bits id of the source reservation station entry that generates the result/data
  - Registers in RS and register file monitors the bus and pick up the value based on the source id.
  - A broadcasting bus

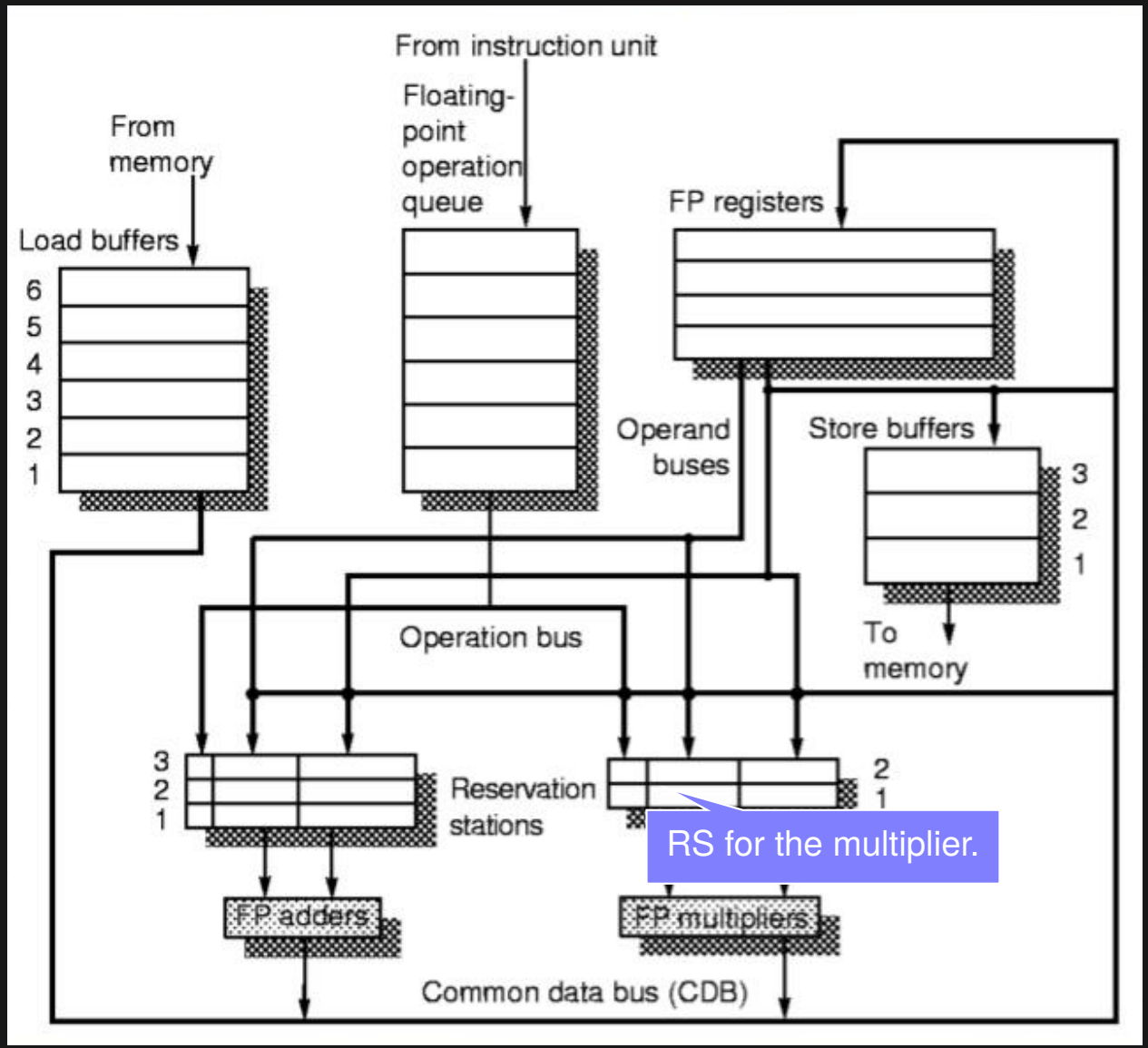
Tomasulo Organization



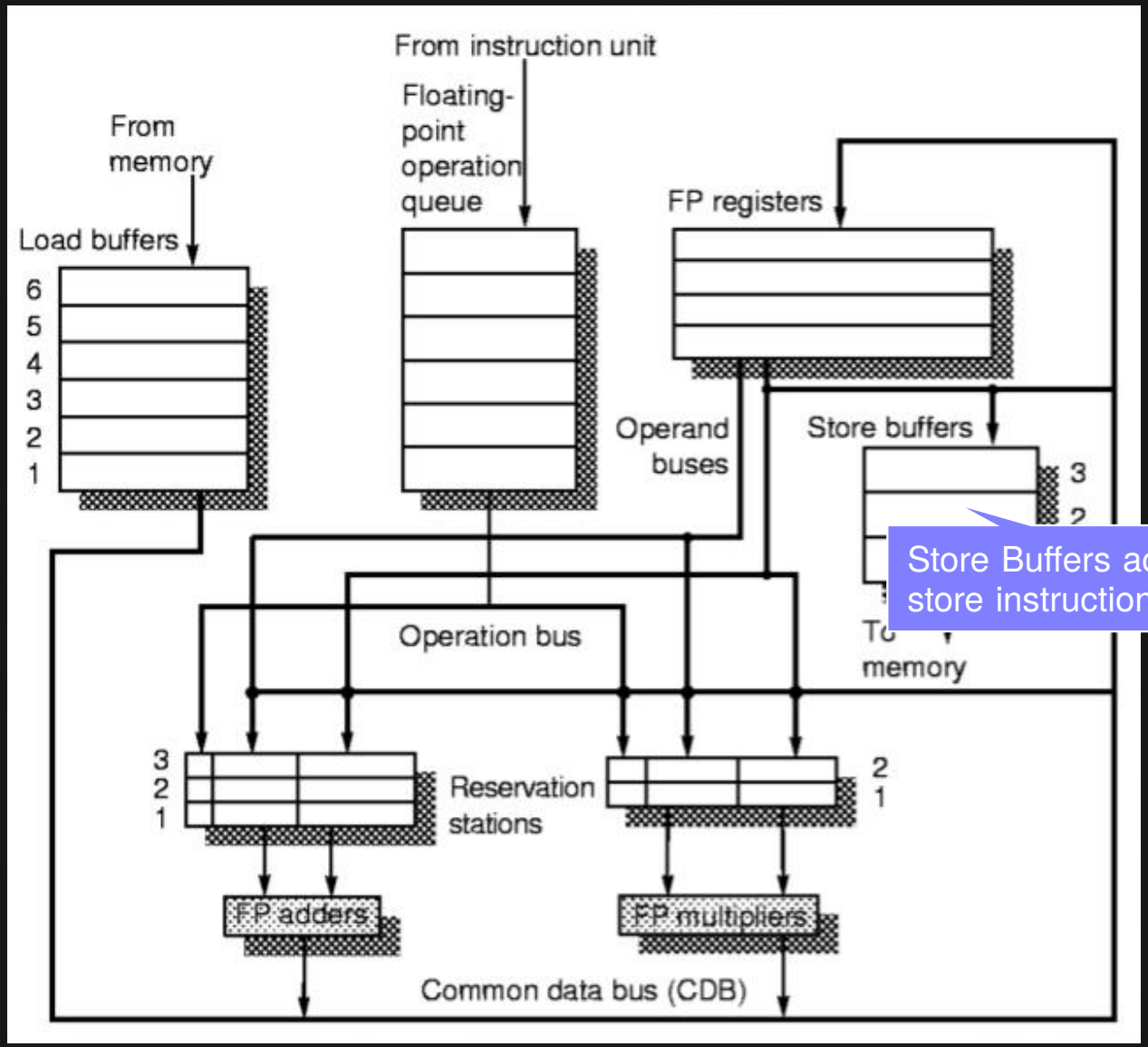
Tomasulo Organization



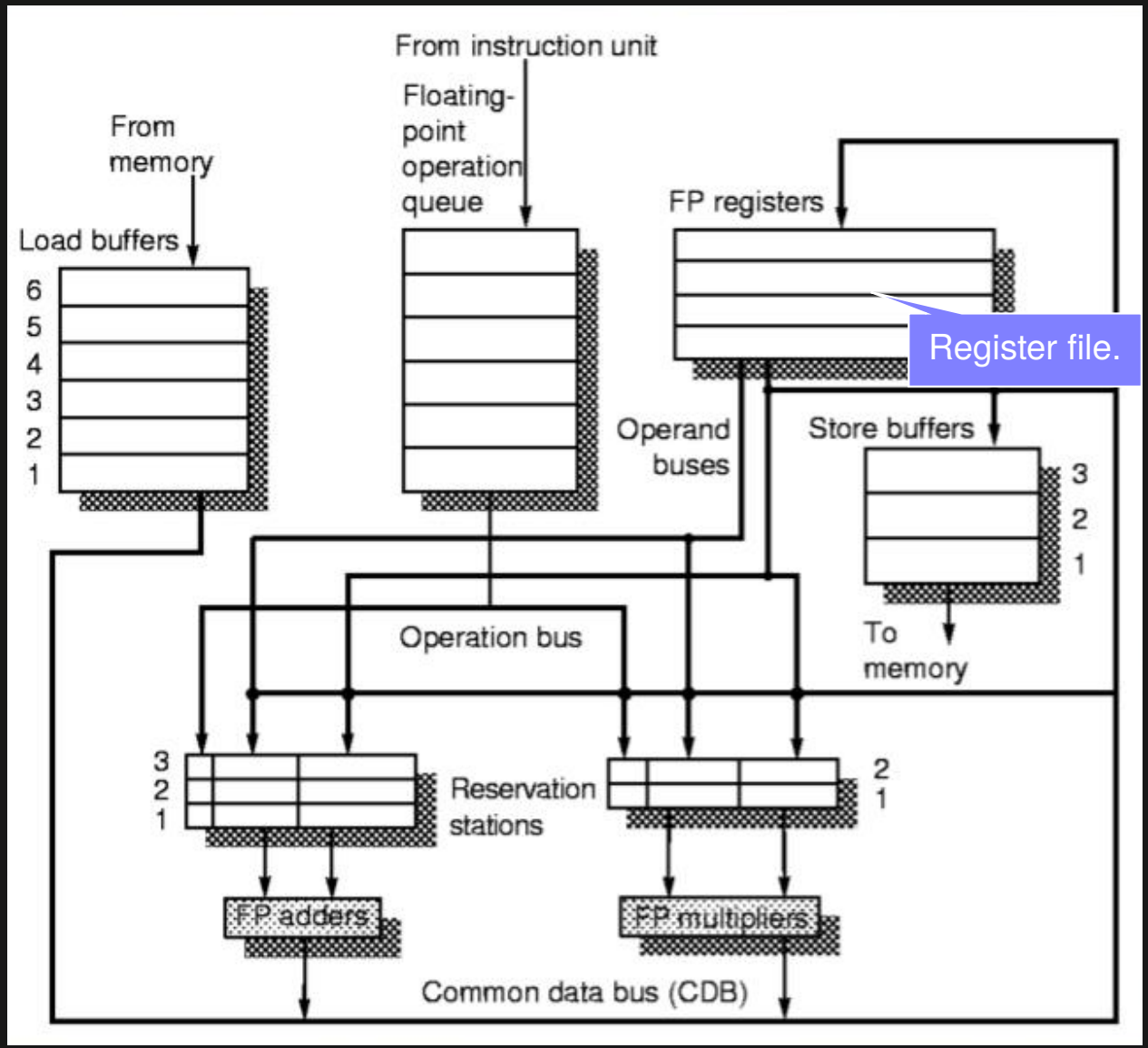
Tomasulo Organization



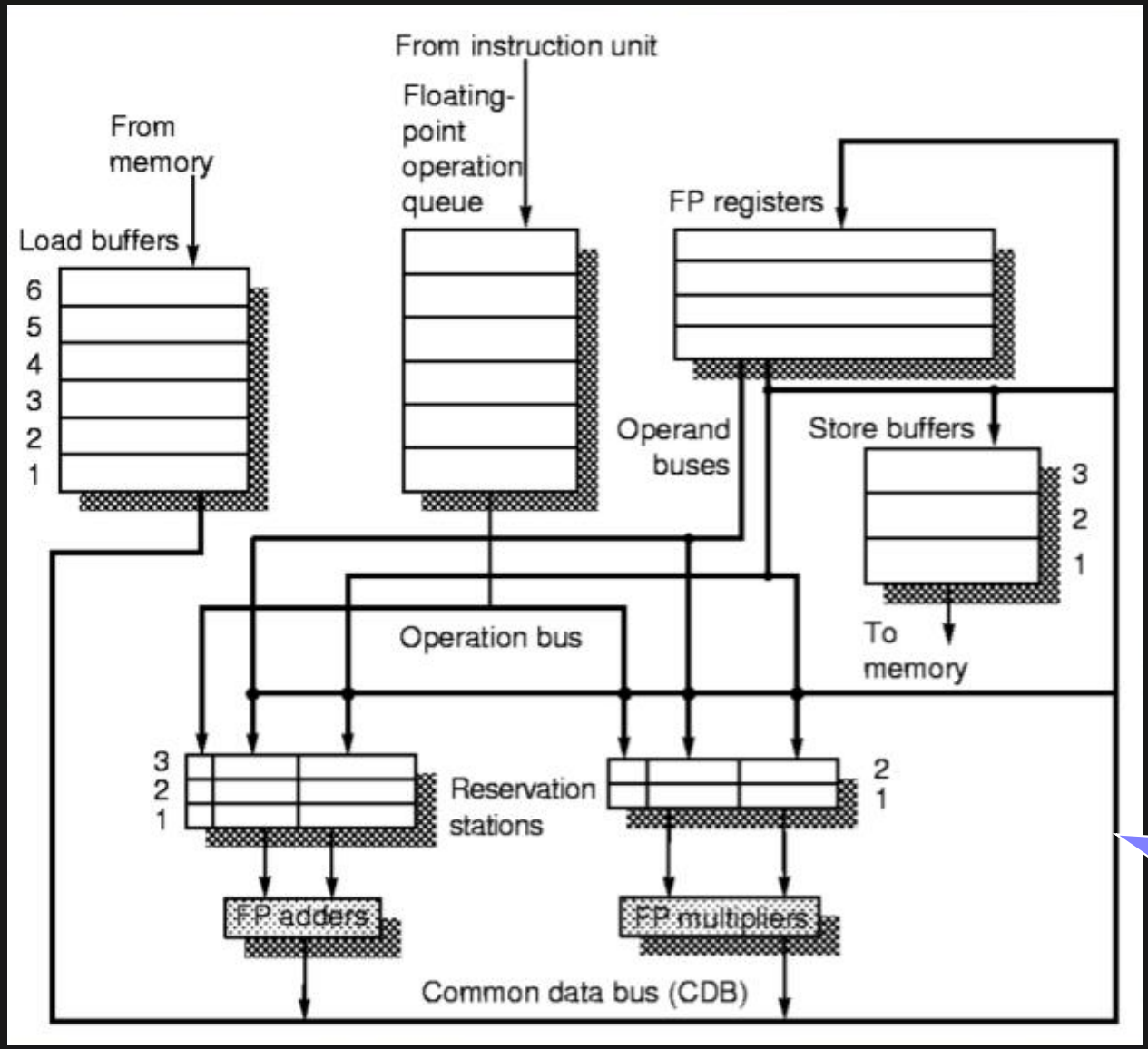
Tomasulo Organization



Tomasulo Organization



Tomasulo Organization



Load buffers, adder and multiplier send values and sources to CDB. CDB broadcasts the values and sources to RS, register file and store buffers.

Road Map

Overview of Tomasulo's Algorithm

An Example of Tomasulo's Algorithm

Another Example of Tomasulo's Algorithm

Tomasulo Wrapup

Acknowledgment

Tomasulo Example

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11			
LD	R2	45+	R13			
MUL	R0	R2	R4			
SUB	R8	R6	R2			
DIV	R10	R0	R6			
ADD	R6	R8	R2			

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU												

Clock: cycle 0

Tomasulo Example: Cycle 1

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1		
LD	R2	45+	R13			
MUL	R0	R2	R4			
SUB	R8	R6	R2			
DIV	R10	R0	R6			
ADD	R6	R8	R2			

Load Buffers:

	Busy	Address
Load1	Yes	34+R11
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Load1											

Clock: cycle 1

Tomasulo Example: Cycle 1

Instruction Status:

	Instructions			Stages		
Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
LD	R6	34+	R11	1		
LD	R2	45+	R13			
MUL	R0	R2	R4			
SUB	R8	R6	R2			
DIV	R10	R0	R6			
ADD	R6	R8	R2			

Issue the first instruction at cycle 1.

Load Buffers:

	Busy	Address
Load1	Yes	34+R11
Load2	N	
Load3		

Occupied the first load buffer entry with instruction 1.

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU							Load1					

Register R6 will be filled with the result from the Load1.

Clock: cycle 1

Tomasulo Example: Cycle 2

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2	
LD	R2	45+	R13	2		
MUL	R0	R2	R4			
SUB	R8	R6	R2			
DIV	R10	R0	R6			
ADD	R6	R8	R2			

Load Buffers:

	Busy	Address
Load1	Yes	34+R11
Load2	Yes	45+R13
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU			Load2				Load1					

Clock: cycle 2

Tomasulo Example: Cycle 2

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2	
LD	R2	45+	R13	2		
MUL	R0	R2	R4			
SUB	R8	R6	R6			
DIV	R10	R0	R0			
ADD	R6	R8				

Issue the 2nd instruction at cycle 2, since load buffers have empty entries. Unlike Scoreboarding, as long as there are entries in the load buffers, load instructions can be issued.

Load Buffers:

	Busy	Address
Load1	Yes	34+R11
Load2	Yes	45+R13
Load3	No	

Occupied the 2nd load buffer entry with instruction 2.

Reservation Stations for th

Time	Name	Busy	Op	V_j	V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU			Load2				Load1					

Clock: cycle

Register R2 will be filled with the result from Load2.

Tomasulo Example: Cycle 2

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
LD	R6	34+	R11	1	2	
LD	R2	45+	R13	2		
MUL	R0	R2	R4			
SUB	R8	R6	R2			
DIV	R10	R0	R6			
ADD	R6	R8	R2			

Instruction 1 starts execution. Takes 2 cycles to complete.

Load Buffers:

	Busy	Address
Load1	Yes	34+R11
Load2	Yes	45+R13
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU			Load2				Load1					

Clock: cycle 2

Tomasulo Example: Cycle 3

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	
LD	R2	45+	R13	2	3	
MUL	R0	R2	R4	3		
SUB	R8	R6	R2			
DIV	R10	R0	R6			
ADD	R6	R8	R2			

Load Buffers:

	Busy	Address
Load1	Yes	34+R11
Load2	Yes	45+R13
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL		Value(R4)	Load2	
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		Load2				Load1					

Clock: cycle 3

Tomasulo Example: Cycle 3

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2-3	
LD	R2	45+	R13	2	3	
MUL	R0	R2	R4	3		
SUB	R8	R6	R2			
DIV	R10	R0	R6			
ADD	R6	R8	R2			

Issue the 3rd instruction at cycle 3.

Load Buffers:

	Busy	Address
Load1	Yes	34+R11
Load2	Yes	45+R13
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL		Value(R4)	Load2	
	Mul2	No					

Occupy Mul1 with the 3rd instruction. Copy the value of register $R4$ in to V_k (src2 value). Mark Q_j to "Load2", indicating src1 comes from "Load2".

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6
FU	Mul1		Load2				Load1

Register R0 will be filled with the result from Mul1.

Tomasulo Example: Cycle 3

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2-3	
LD	R2	45+	R13	2	3	
MUL	R0	R2	R4	3		
SUB	R8	R6	R2			
DIV	R10	R0	R6			
ADD	R6	R8	R2			

Instruction 1 finishes execution at the end of cycle 3.

Load Buffers:

	Busy	Address
Load1	Yes	34+R11
Load2	Yes	45+R13
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL		Value(R4)	Load2	
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		Load2				Load1					

Clock: cycle 3

Tomasulo Example: Cycle 3

Instruction Status:

	Instructions			Stages		
Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
LD	R6	34+	R11	1	2-3	
LD	R2	45+	R13	2	3	
MUL	R0	R2	R4	3		
SUB	R8	R6	R2			
DIV	R10	R0	R6			
ADD	R6	R8	R2			

Instruction 2 starts at cycle 3, assuming multiple memory loads can happened at the same time. Takes 2 cycles to execute.

Load Buffers:

	Busy	Address
Load1	Yes	34+R11
Load2	Yes	45+R13
Load3	No	

Reservation Stations for the Adder and Mul

Time	Name	Busy	Op	Src V_j	V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL		Value(R4)	Load2	
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		Load2				Load1					

Clock: cycle 3

Tomasulo Example: Cycle 4

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	
MUL	R0	R2	R4	3		
SUB	R8	R6	R2	4		
DIV	R10	R0	R6			
ADD	R6	R8	R2			

Load Buffers:

	Busy	Address
Load1	No	
Load2	Yes	45+R13
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	SUB	M_1			Load2
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL		Value(R4)	Load2	
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		Load2				M_1		Add1			

Clock: cycle 4

Tomasulo Example: Cycle 4

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	
MUL	R0	R2	R4	3		
SUB	R8	R6	R2	4		
DIV	R10	R0	R6			
ADD	R6	R8	R2			

Issue the 4th instruction at cycle 4.

Load Buffers:

	Busy	Address
Load1	No	
Load2	Yes	45+R13
Load3	No	

Reservation Stations for the Adder and Mu

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	SUB	M_1			Load2
	Add2	No					
	Add3						
	Mul1				R4)	Load2	
	Mul2						

Occupy Add1 with the 4th instruction. Copy the value (M_1) of from "Load1" from CDB to V_j (src1 value). Mark Q_k to "Load2", indicating src1 comes from "Load2".

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		Load2				M_1		Add1			

Register R8 will be filled with the result from Add1.

Clock: cycle 4

Tomasulo Example: Cycle 4

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	
MUL	R0	R2	R4	3		
SUB	R8	R6	R2	4		
DIV	R10	R0	R6			
ADD	R6	R8	R2			

Load Buffers:

	Busy	Address
Load1	No	
Load2	Yes	45+R13
Load3		

Load1 is released.

Instruction 1 writes back. Load buffers sends result value M_1 and id "Load1" on to CDB.

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	SUB	M_1			Load2
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL		Value(R4)	Load2	
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		Load2				M_1		Add1			

Register R6 sees id "Load1" on CDB matches its source, picks up the value M_1.

Clock: cycle 4

Tomasulo Example: Cycle 4

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	
MUL	R0	R2	R4	3		
SUB	R8	R6	R2	4		
DIV	R10	R0	R6			
ADD	R6	R8	R2			

Instruction 2 finishes at cycle 4.

Load Buffers:

	Busy	Address
Load1	No	
Load2	Yes	45+R13
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	SUB	M_1			Load2
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL		Value(R4)	Load2	
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		Load2				M_1		Add1			

Clock: cycle 4

Tomasulo Example: Cycle 4

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	
MUL	R0	R2	R4	3		
SUB	R8	R6	R2	4		
DIV	R10	R0	R6			
ADD	R6	R8	R2			

Instruction 3 stalled due to missing src1.

Load Buffers:

	Busy	Address
Load1	No	
Load2	Yes	45+R13
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	SUB	M_1			Load2
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL		Value(R4)	Load2	
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		Load2				M_1		Add1			

Clock: cycle 4

Tomasulo Example: Cycle 5

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3		
SUB	R8	R6	R2	4		
DIV	R10	R0	R6	5		
ADD	R6	R8	R2			

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	SUB	M_1	M_2		
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				M_1		Add1		Mul2	

Clock: cycle 5

Tomasulo Example: Cycle 5

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3		
SUB	R8	R6	R2	4		
DIV	R10	R0	R6	5		
ADD	R6	R8	R2			

Instruction 2 writes back. Load buffers sends result value M_2 and id "Load2" on to CDB.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Load2 is re-leased.

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	SUB	M_1	M_2		
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				M_1		Add1		Mul2	

Clock: cycle

Register R2 sees id "Load2" on CDB matches its source, picks up the value M_2.

Tomasulo Example: Cycle 5

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3		
SUB	R8	R6	R2	4		
DIV	R10	R0	R6	5		
ADD	R6	R8	R2			

Issue the 5th instruction at cycle 5.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Mu

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	SUB	M_1	M_2		
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Occupy Mul2 with the 5th instruction. Copy the value (M_1) of from R0 to V_k (src2 value). Mark Q_j to "Mul1", indicating src1 comes from "Mul1".

Register Result Status:

	R0	R1	R	R8	R9	R10	... ...
FU	Mul1		M	Add1		Mul2	

Register R10 will be filled with the result from Mul2.

Clock: cycle 5

Tomasulo Example: Cycle 5

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3		
SUB	R8	R6	R2	4		
DIV	R10	R0	R6	5		
ADD	R6	R8	R2			

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	SUB	M_1	M_2		
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Mul1 sees id “Load2” on CDB matches its source, picks up the value M_2 and set it to V_j.

Register Result Status:

	R0	R1	R2	R3	9	R10	... ...
FU	Mul1		M_2		M_1 Add1	Mul2	

Clock: cycle 5

Tomasulo Example: Cycle 5

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3		
SUB	R8	R6	R2	4		
DIV	R10	R0	R6	5		
ADD	R6	R8	R2			

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	SUB	M_1	M_2		
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL	M			
	Mul2	Yes	DIV				

Add1 sees id "Load2" on CDB matches its source, picks up the value M_2 and set it to V_k.

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				M_1		Add1		Mul2	

Clock: cycle 5

Tomasulo Example: Cycle 6

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6	
SUB	R8	R6	R2	4	6	
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
1	Add1	Yes	SUB	M_1	M_2		
	Add2	Yes	ADD		M_2	Add1	
	Add3	No					
9	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Add2		Add1		Mul2	

Clock: cycle 6

Tomasulo Example: Cycle 6

Instruction Status:

	Instructions			Stages		
Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6	
SUB	R8	R6	R2	4	6	
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Issue the 6th instruction at cycle 6, since there are empty slots in the RS for adders.

Reservation Stations for the Ad

Time	Name	Busy				Src1 RS Id Q_j	Src2 RS Id Q_k
1	Add1	Yes	SUB	M_1	M_2		
	Add2	Yes	ADD		M_2	Add1	
	Add3	No					
9	Mul1				R4)		
	Mul2					Mul1	

Occupy Add2 with the 6th instruction. Copy the value (M_2) of from R2 to V_k (src2 value). Mark Q_j to "Add1", indicating src1 comes from "Add1".

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Add2		Add1		Mul2	

Register R6 will be filled with the result from Add2.

Clock: cycle 6

Tomasulo Example: Cycle 6

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6	
SUB	R8	R6	R2	4	6	
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6		

All source operands are ready, instruction 3 starts execution.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
1	Add1	Yes	SUB	M_1	M_2		
	Add2	Yes	ADD		M_2	Add1	
	Add3	No					
9	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Add2		Add1		Mul2	

Clock: cycle 6

Tomasulo Example: Cycle 6

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6	
SUB	R8	R6	R2	4	6	
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

All source operands are ready, instruction 4 starts execution.

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Valu V_j	V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
1	Add1	Yes	SUB	M_1	M_2		
	Add2	Yes	ADD		M_2	Add1	
	Add3	No					
9	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Instruction 3 and 4 still needs 1 and 9 cycles to finish.

Re

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Add2		Add1		Mul2	

Clock: cycle 6

Tomasulo Example: Cycle 7

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6	
SUB	R8	R6	R2	4	6-7	
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
0	Add1	Yes	SUB	M_1	M_2		
	Add2	Yes	ADD		M_2	Add1	
	Add3	No					
8	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Add2		Add1		Mul2	

Clock: cycle 7

Tomasulo Example: Cycle 7

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6	
SUB	R8	R6	R2	4	6-7	
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6		

Instruction 4 finishes.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
0	Add1	Yes	SUB	M_1	M_2		
	Add2	Yes	ADD		M_2	Add1	
	Add3	No					
8	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Add2		Add1		Mul2	

Clock: cycle 7

Tomasulo Example: Cycle 8

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6	
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	Yes	ADD	Val_1	M_2		
	Add3	No					
7	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Add2		Val_1		Mul2	

Clock: cycle 8

Tomasulo Example: Cycle 8

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6	
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6		

Instruction 4 writes back.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	Yes	ADD	Val_1	M_2		
	Add3						
7	Mul1				R4)		
	Mul2					Mul1	

Add1 sends result value Val_1 and id "Add1" on to CDB. Instruction 4 releases Add1.

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Add2		Val_1		Mul2	

Clock: cycle 8

Tomasulo Example: Cycle 8

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6	
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	Yes	ADD	Val_1	M_2		
	Add3	No					
7	Mul1	Yes	MU				
	Mul2	Yes	DIV			ul1	

Add2 reads id "Add1" from CDB, which matches its V_j. THerefore, Add2 reads value Val_1 from CDB to Val_j

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Add2		Val_1		Mul2	

Register R8 sees id "Add1" on CDB, which matches its source, so it picks up the value Val_1.

Clock: cycle 8

Tomasulo Example: Cycle 9

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6	
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6	9	

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
1	Add2	Yes	ADD	Val_1	M_2		
	Add3	No					
6	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Add2		Val_1		Mul2	

Clock: cycle 9

Tomasulo Example: Cycle 9

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6	
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6	9	

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multipl

Time	Name	Busy	Op	Src1 V*j*	alue	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
1	Add2	Yes	ADD	Val_1	M_2		
	Add3	No					
				M_2	Value(R4)		
					M_1	Mul1	

Instruction 6 starts execution.

Add takes 2 cycles. 1 cycle remains.

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Add2		Val_1		Mul2	

Clock: cycle 9

Tomasulo Example: Cycle 10

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6	
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6	9-10	

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
0	Add2	Yes	ADD	Val_1	M_2		
	Add3	No					
5	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Add2		Val_1		Mul2	

Clock: cycle 10

Tomasulo Example: Cycle 10

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6	
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6	9-10	

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Instruction 6 finishes execution.

Reservation Stations for the Adder and Multiplie

Time	Name	Busy	Op	Src1 V$_j$	ue	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
0	Add2	Yes	ADD	Val_1	M_2		
	Add3	No					
5	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Add2		Val_1		Mul2	

Clock: cycle 10

Tomasulo Example: Cycle 11

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6	
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6	9-10	11

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
4	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Val_2		Val_1		Mul2	

Clock: cycle 11

Tomasulo Example: Cycle 11

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6	
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6	9-10	11

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Instruction 6 writes back.

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	S V_k	c1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
4	Mul1				R4)		
	Mul2					Mul1	

Add2 sends result value Val_2 and id "Add2" on to CDB. Instruction 6 releases Add2.

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Val_2		Val_1		Mul2	

Clock: cycle 11

Tomasulo Example: Cycle 11

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6	
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6	9-10	11

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
4	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Val_2		Val_1		Mul2	

Clock

Register R6 sees id “Add2” on CDB, which matches its source, so it picks up the value Val_2. Unlike Scoreboarding, WAW does not prevent write back as registers are already renamed: instruction 5 (DIV) waits on “Mul1” instead of R6 in the RS.

Tomasulo Example: Cycle 15

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6-15	
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6	9-10	11

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
0	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Val_2		Val_1		Mul2	

Clock: cycle 15

Tomasulo Example: Cycle 15

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6-15	
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6	9	

Instruction 3 finishes.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
0	Mul1	Yes	MUL	M_2	Value(R4)		
	Mul2	Yes	DIV		M_1	Mul1	

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Mul1		M_2				Val_2		Val_1		Mul2	

Clock: cycle 15

Tomasulo Example: Cycle 16

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2/3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6-15	16
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6	9-10	11

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
	Mul2	Yes	DIV	Val_3	M_1		

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Val_3		M_2				Val_2		Val_1		Mul2	

Clock: cycle 16

Tomasulo Example: Cycle 16

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2/3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6-15	16
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6	9-10	

Instruction 3 writes back.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
	Mul2	Yes	DIV	Val_3	M_1		

Mul1 sends result value Val_3 and id "Mul1" on to CDB. Instruction 3 releases Mul1.

Register Result Status

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Val_3		M_2				Val_2		Val_1		Mul2	

Clock: cycle 16

Tomasulo Example: Cycle 16

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2/3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6-15	16
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5		
ADD	R6	R8	R2	6	9-10	11

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
	Mul2	Yes	DIV	Val_3	M_1		

Register Result Status:

	R0	R1	R2	R3	R9	R10	... ...
FU	Val_3		M_2			Mul2	

Mul2 reads id "Mul1" from CDB, which matches its V_j. THerefore, Mul2 reads value Val_2 from CDB to Val_j

Register R0 sees id "Mul1" on CDB, which matches its source, so it picks up the value Val_3.

Tomasulo Example: Cycle 17

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6-15	16
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5	17	
ADD	R6	R8	R2	6	9-10	11

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
39	Mul2	Yes	DIV	Val_3	M_1		

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Val_3		M_2				Val_2		Val_1		Mul2	

Clock: cycle 17

Tomasulo Example: Cycle 17

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6-15	16
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5	17	
ADD	R6	R8	R2	6	9-10	11

Instruction 5 starts execution.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
39	Mul2	Yes	DIV	Val_3	M_1		

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Val_3		M_2				Val_2		Val_1		Mul2	

Clock: cycle 17

Tomasulo Example: Cycle 56

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6-15	16
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5	17-56	
ADD	R6	R8	R2	6	9-10	11

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
0	Mul2	Yes	DIV	Val_3	M_1		

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Val_3		M_2				Val_2		Val_1		Mul2	

Clock: cycle 56

Tomasulo Example: Cycle 56

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6-15	16
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5	17-56	
ADD	R6	R8	R2	6	9-10	11

Instruction 5 finishes execution.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
0	Mul2	Yes	DIV	Val_3	M_1		

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Val_3		M_2				Val_2		Val_1		Mul2	

Clock: cycle 56

Tomasulo Example: Cycle 57

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6-15	16
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5	17-56	57
ADD	R6	R8	R2	6	9-10	11

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
0	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Val_3		M_2				Val_2		Val_1		Val_4	

Clock: cycle 57

Tomasulo Example: Cycle 57

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6-15	16
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5	17-56	57
ADD	R6	R8	R2	6	9-10	11

Instruction 5 writes back.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
0	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Val_3		M_2				Val_2		Val_1		Val_4	

Clock: cycle 57

Tomasulo Example: Cycle 57

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
LD	R6	34+	R11	1	2-3	4
LD	R2	45+	R13	2	3-4	5
MUL	R0	R2	R4	3	6-15	16
SUB	R8	R6	R2	4	6-7	8
DIV	R10	R0	R6	5	17-56	57
ADD	R6	R8	R2	6	9-10	11

Instructions | Stages

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Out-of-order completion.

In-order issue.

Out-of-order execution.

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Valu V_j	V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
0	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU	Val_3		M_2				Val_2		Val_1		Val_4	

Clock: cycle 57

Comparing to Scoreboard

- Comparing to Scoreboard's 62 cycles, Tomasulo's Algorithm takes only 57 cycles to execute the same code.
- Why Tomasulo's Algorithm is faster?
  - Fewer structure hazards (Scoreboarding cannot issue if functional units are busy).
  - CDB acts as a data forwarding mechanism: write back also forwards the value to RS, eliminating a separate register read.

Road Map

Overview of Tomasulo's Algorithm

An Example of Tomasulo's Algorithm

Another Example of Tomasulo's Algorithm

Tomasulo Wrapup

Acknowledgment

Tomasulo Example 2

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11			
MUL	R1	16	R1			
ADD	R5	R5	R1			
LD	R1	4	R1			
MUL	R1	16	R1			
ADD	R5	R5	R1			

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU												

Clock: cycle 0

Tomasulo Example 2: Cycle 1

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R1	0+	R11	1		
MUL	R1	16	R1			
ADD	R5	R5	R1			
LD	R1	4	R11			
MUL	R1	16	R1			
ADD	R5	R5	R1			

Load Buffers:

	Busy	Address
Load1	Yes	0 + R11
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Load1										

Clock: cycle 1

Tomasulo Example 2: Cycle 1

Instruction Status:

	Instructions			Stages		
Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
LD	R1	0+	R11	1		
MUL	R1	16	R1			
ADD	R5	R5	R1			
LD	R1	4	R11			
MUL	R1	16	R1			
ADD	R5	R5	R1			

Issue the first instruction at cycle 1.

Load Buffers:

	Busy	Address
Load1	Yes	0 + R11
Load2	No	
Load3		

Occupied the first load buffer entry with instruction 1.

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	No					
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Load1										
Clock												

Register R1 will be filled with the result from the Load1.

Tomasulo Example 2: Cycle 2

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2	
MUL	R1	16	R1	2		
ADD	R5	R5	R1			
LD	R1	4	R11			
MUL	R1	16	R1			
ADD	R5	R5	R1			

Load Buffers:

	Busy	Address
Load1	Yes	0 + R11
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL	16			Load1
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul1										

Clock: cycle 2

Tomasulo Example 2: Cycle 2

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R1	0+	R11	1	2	
MUL	R1	16	R1	2		
ADD	R5	R5	R1			
LD	R1	4	R11			
MUL	R1	16	R1			
ADD	R5	R5	R1			

Issue the 2nd instruction at cycle 2

Load Buffers:

	Busy	Address
Load1	Yes	0 + R11
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL	16			Load1
	Mul2	No					

Occupy Mul1 with the 3rd instruction. Copy the immd value 16 in to V_k (src1 value). Mark Q_k to "Load1", indicating src2 comes from "Load1".

Register Result Status:

	R0	R1	R2	R3	R9	R10	... ...
FU		Mul1					

Clock

Register R1 will be filled with the result from Mul1. Note that, value from Load1 will not be copied into R1 anymore. Instead, Load1 will be forwarded to Mul1 directly.

Tomasulo Example 2: Cycle 2

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
LD	R1	0+	R11	1	2	
MUL	R1	16	R1	2		
ADD	R5	R5	R1			
LD	R1	4	R11			
MUL	R1	16	R1			
ADD	R5	R5	R1			

Instruction 1 starts executing. Takes 2 cycles to complete.

Load Buffers:

	Busy	Address
Load1	Yes	0 + R11
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL	16			Load1
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul1										

Clock: cycle 2

Tomasulo Example 2: Cycle 3

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R1	0+	R11	1	2-3	
MUL	R1	16	R1	2		
ADD	R5	R5	R1	3		
LD	R1	4	R11			
MUL	R1	16	R1			
ADD	R5	R5	R1			

Load Buffers:

	Busy	Address
Load1	Yes	0 + R11
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL	16			Load1
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul1				Add1						

Clock: cycle 3

Tomasulo Example 2: Cycle 3

Instruction Status:

	Instructions			Stages		
Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
LD	R1	0+	R11	1	2-3	
MUL	R1	16	R1	2		
ADD	R5	R5	R1	3		
LD	R1	4	R11			
MUL	R1	16	R1			
ADD	R5	R5	R1			

Issue the 3rd instruction at cycle 3.

Load Buffers:

	Busy	Address
Load1	Yes	0 + R11
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	No					
	Add3	No					
	Mul1	Yes	M				Load1
	Mul2	No					

Occupy Add1 with the 3rd instruction. Copy the value of register $R5$ in to V_j (src1 value). Mark Q_k to "Mul1", indicating src2 comes from "Mul1".

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul1				Add1						

Register R5 will be filled with the result from Add1.

Clock: cycle 3

Tomasulo Example 2: Cycle 3

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	
MUL	R1	16	R1	2		
ADD	R5	R5	R1	3		
LD	R1	4	R11			
MUL	R1	16	R1			
ADD	R5	R5	R1			

Instruction 1 finishes execution at the end of cycle 3.

Load Buffers:

	Busy	Address
Load1	Yes	0 + R11
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL	16			Load1
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul1				Add1						

Clock: cycle 3

Tomasulo Example 2: Cycle 4

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2		
ADD	R5	R5	R1	3		
LD	R1	4	R11	4		
MUL	R1	16	R1			
ADD	R5	R5	R1			

Load Buffers:

	Busy	Address
Load1	No	
Load2	Yes	4 + R11
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL	16	M_1		
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Load2				Add1						

Clock: cycle 4

Tomasulo Example 2: Cycle 4

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2		
ADD	R5	R5	R1	3		
LD	R1	4	R11	4		
MUL	R1	16	R1			
ADD	R5	R5	R1			

Issue the 4th instruction at cycle 4.

Load Buffers:

	Busy	Address
Load1	No	
Load2	Yes	4 + R11
Load3	No	

Occupied the 2nd load buffer entry with instruction 2.

Reservation Stations for the Adder and Mu

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL	16	M_1		
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Load2				Add1						
Clock												

Register R1 will be filled with the result from Load2. Note that, value from Mul1 will not be copied into R1 anymore. Instead, Mul1 will be forwarded to Add1 directly.

Tomasulo Example 2: Cycle 4

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2		
ADD	R5	R5	R1	3		
LD	R1	4	R11	4		
MUL	R1	16	R1			
ADD	R5	R5	R1			

Load Buffers:

	Busy	Address
Load1	No	
Load2	Yes	4 + R11
Load3		

Load1 is released.

Instruction 1 writes back. Load buffers sends result value M_1 and id "Load1" on to CDB.

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	No					
	Add3	No					
	Mul1	Yes	MUL	16	M_1		
	Mul2	No					

Mul1 sees id "Load1" on CDB matches its source, picks up the value M_1 and set it to V_k.

Register Result Status:

	R0	R1	R2	R3	...	9	R10	... ...
FU		Load2				Add1		

Clock: cycle 4

Tomasulo Example 2: Cycle 5

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5	
ADD	R5	R5	R1	3		
LD	R1	4	R11	4	5	
MUL	R1	16	R1	5		
ADD	R5	R5	R1			

Load Buffers:

	Busy	Address
Load1	No	
Load2	Yes	4 + R11
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	No					
	Add3	No					
9	Mul1	Yes	MUL	16	M_1		
	Mul2	Yes	MUL	16			Load2

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add1						

Clock: cycle 5

Tomasulo Example 2: Cycle 5

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5	
ADD	R5	R5	R1	3		
LD	R1	4	R11	4	5	
MUL	R1	16	R1	5		
ADD	R5	R5	R1			

Issue the 5th instruction at cycle 5.

Load Buffers:

	Busy	Address
Load1	No	
Load2	Yes	4 + R11
Load3	No	

Reservation Stations for the Adder and Mu

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	No					
	Add3	No					
9	Mul1	Yes	MUL	16	M_1		
	Mul2	Yes	MUL	16			Load2

Occupy Mul2 with the 5th instruction. Copy the immd value 16 in to V_k (src1 value). Mark Q_k to "Load2", indicating src2 comes from "Load2".

Register Result Status:

	R0	R1	R2	R3	... ...
FU		Mul2		Add1	

Clock

Register R1 will be filled with the result from Mul2. Note that, value from Load2 will not be copied into R1 anymore. Instead, Mul1 will be forwarded to Mul2 directly.

Tomasulo Example 2: Cycle 5

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5	
ADD	R5	R5	R1	3		
LD	R1	4	R11	4	5	
MUL	R1	16	R1	5		
ADD	R5	R5	R1			

Instruction 2 starts executing at Mul1

Load Buffers:

	Busy	Address
Load1	No	
Load2	Yes	4 + R11
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	No					
	Add3	No					
9	Mul1	Yes	MUL	16	M_1		
	Mul2	Yes	MUL	16			Load2

Instruction 2 still needs 9 cycles to finish.

Re

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add1						

Clock: cycle 5

Tomasulo Example 2: Cycle 5

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5	
ADD	R5	R5	R1	3		
LD	R1	4	R11	4	5	
MUL	R1	16	R1	5		
ADD	R5	R5	R1			

Instruction 4 starts executing at Load2

Load Buffers:

	Busy	Address
Load1	No	
Load2	Yes	4 + R11
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	No					
	Add3	No					
9	Mul1	Yes	MUL	16	M_1		
	Mul2	Yes	MUL	16			Load2

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add1						

Clock: cycle 5

Tomasulo Example 2: Cycle 6

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5	
ADD	R5	R5	R1	3		
LD	R1	4	R11	4	5-6	
MUL	R1	16	R1	5		
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	Yes	4 + R11
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
8	Mul1	Yes	MUL	16	M_1		
	Mul2	Yes	MUL	16			Load2

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add2						

Clock: cycle 6

Tomasulo Example 2: Cycle 6

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5	
ADD	R5	R5	R1	3		
LD	R1	4	R11	4	5-6	
MUL	R1	16	R1	5		
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	Yes	4 + R11
Load3	No	

Issue the 6th instruction at cycle 6.

Reservation Stations for the Adder and M

Time	Name	Busy	Op	S	c2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
8	Mul1						
	Mul2						Load2

Occupy Add2 with the 6th instruction. Both operands are not ready, and are from “Add1” and “Mul2” each.

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add2						

Clock: cycle 6

Register R5 will be filled with the result from Add2. Note that, value from Add1 will not be copied into R5 anymore. Instead, Add1 will be forwarded to Add2 directly.

Tomasulo Example 2: Cycle 6

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5	
ADD	R5	R5	R1	3		
LD	R1	4	R11	4	5-6	
MUL	R1	16	R1	5		
ADD	R5	R5	R1	6		

Instruction 4 finishes execution at the end of cycle 6.

Load Buffers:

	Busy	Address
Load1	No	
Load2	Yes	4 + R11
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Val V_j	Src2 Val V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
8	Mul1	Yes	MUL	16	M_1		
	Mul2	Yes	MUL	16			Load2

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add2						

Clock: cycle 6

Tomasulo Example 2: Cycle 7

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5	
ADD	R5	R5	R1	3		
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5		
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
7	Mul1	Yes	MUL	16	M_1		
	Mul2	Yes	MUL	16	M_2		

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add2						

Clock: cycle 7

Tomasulo Example 2: Cycle 7

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5	
ADD	R5	R5	R1	3		
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5		
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Load2 is released.

Instruction 4 writes back. Load buffers sends result value M_2 and id "Load2" on to CDB.

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
7	Mul1	Yes	MUL	16	M_1		
	Mul2	Yes	MUL	16	M_2		

Register Result Status:

	R0	R1	R2	R3	...	R9	R10	... ...
FU		Mul2						

Mul2 sees id "Load2" on CDB matches its source, picks up the value M_2 and set it to V_k.

Clock: cycle 7

Tomasulo Example 2: Cycle 8

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5	
ADD	R5	R5	R1	3		
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8	
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
6	Mul1	Yes	MUL	16	M_1		
9	Mul2	Yes	MUL	16	M_2		

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add2						

Clock: cycle 8

Tomasulo Example 2: Cycle 8

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5	
ADD	R5	R5	R1	3		
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8	
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Instruction 5 starts executing at Mul2, assuming we have two adders. We now have two iterations of the loop running at the same time.

Reservation Stations for the Adder and

Time	Name	Busy	Op			RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD				Mul1
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
6	Mul1	Yes	MUL	16	M_1		
9	Mul2	Yes	MUL	16	M_2		

Instruction 2 and 5 still need 6 and 9 cycles to finish.

	R5	R6	R7	R8	R9	R10	... ...
FU	Mul2	Add2					

Clock: cycle 8

Tomasulo Example 2: Cycle 14

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	
ADD	R5	R5	R1	3		
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8	
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
0	Mul1	Yes	MUL	16	M_1		
3	Mul2	Yes	MUL	16	M_2		

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add2						

Clock: cycle 14

Tomasulo Example 2: Cycle 14

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	
ADD	R5	R5	R1	3		
LD	R1	4	R11	4	5-	
MUL	R1	16	R1	5	8	
ADD	R5	R5	R1	6		

Instruction 2 finishes executing.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)			Mul1
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
0	Mul1	Yes	MUL	16	M_1		
3	Mul2	Yes	MUL	16	M_2		

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add2						

Clock: cycle 14

Tomasulo Example 2: Cycle 15

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3		
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8	
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)	Val_1		
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
	Mul1	No					
2	Mul2	Yes	MUL	16	M_2		

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add2						

Clock: cycle 15

Tomasulo Example 2: Cycle 15

Instruction Status:

	Instructions			Stages		
Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3		
LD	R1	4	R11	4	5-6	
MUL	R1	16	R1	5	8	
ADD	R5	R5	R1	6		

Instruction 2 writes back.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)	Val_1		
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
	Mul1	No					
2	Mul2	Yes	MUL	16	M_2		

Mul1 sends result value Val_1 and id "Mul1" on to CDB. Instruction 2 releases Mul1.

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add2						

Clock: cycle 15

Tomasulo Example 2: Cycle 15

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	Stages IS	EX	WB
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3		
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8	
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	Yes	ADD	Val(R5)	Val_1		
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
	Mul1	No					
2	Mul2	Yes	MU				

Add1 sees id "Mul1" on CDB matches its source, picks up the value Val_1 and set it to V_k.

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add2						

Clock: cycle 15

Tomasulo Example 2: Cycle 16

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16	
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8	
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
1	Add1	Yes	ADD	Val(R5)	Val_1		
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
	Mul1	No					
1	Mul2	Yes	MUL	16	M_2		

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add2						

Clock: cycle 16

Tomasulo Example 2: Cycle 16

Instruction Status:

	Instructions			Stages		
Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16	
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8	
ADD	R5	R5	R1	6		

Instruction 3 starts executing at Add1.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
1	Add1	Yes	ADD	Val(R5)	Val_1		
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
	Mul1	No					
1	Mul2	Yes	MUL	16	M_2		

Instruction 3 and 5 still each needs 1 cycle to finish.

	R5	R6	R7	R8	R9	R10	... ...
FU	Mul2		Add2				

Clock: cycle 16

Tomasulo Example 2: Cycle 17

Instruction Status:

	Instructions			Stages		
Op	Dest i	Src1 j	Src2 k	IS	EX	WB
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-17	
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
0	Add1	Yes	ADD	Val(R5)	Val_1		
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
	Mul1	No					
0	Mul2	Yes	MUL	16	M_2		

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add2						

Clock: cycle 17

Tomasulo Example 2: Cycle 17

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-1	
ADD	R5	R5	R1	6		

Instruction 3 finishes executing.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
0	Add1	Yes	ADD	Val(R5)	Val_1		
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
	Mul1	No					
0	Mul2	Yes	MUL	16	M_2		

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add2						

Clock: cycle 17

Tomasulo Example 2: Cycle 17

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-17	
ADD	R5	R5	R1	6		

Instruction 5 finishes executing.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
0	Add1	Yes	ADD	Val(R5)	Val_1		
	Add2	Yes	ADD			Add1	Mul2
	Add3	No					
	Mul1	No					
0	Mul2	Yes	MUL	16	M_2		

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Mul2				Add2						

Clock: cycle 17

Tomasulo Example 2: Cycle 18

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	18
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-17	
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	Yes	ADD	Val_2			Mul2
	Add3	No					
	Mul1	No					
0	Mul2	Yes	MUL	16	M_2		

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Val_3				Add2						

Clock: cycle 18

Tomasulo Example 2: Cycle 18

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	18
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-17	
ADD	R5	R5	R1	6		

Instruction 3 writes back.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	Yes	ADD	Val_2			Mul2
	Add3						
	Mul1						
0	Mul2						

Add1 sends result value Val_2 and id "Add1" on to CDB. Instruction 3 releases Add1.

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Val_3				Add2						

Clock: cycle 18

Tomasulo Example 2: Cycle 18

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	18
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-17	
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	Yes	ADD	Val_2			Mul2
	Add3	No					
	Mul1	No					
0	Mul2	Yes	M				

Add2 sees id "Add1" on CDB matches its source, picks up the value Val_2 and set it to V_j.

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Val_3				Add2						

Clock: cycle 18

Tomasulo Example 2: Cycle 18

Instruction Status:

	Instructions			Stages		
Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	18
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-17	
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Instruction 5 cannot write back due to CDB is busy (structural hazard).

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	S	1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	Yes	ADD	Val_2			Mul2
	Add3	No					
	Mul1	No					
0	Mul2	Yes	MUL	16	M_2		

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Val_3				Add2						

Clock: cycle 18

Tomasulo Example 2: Cycle 19

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	18
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-17	19
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	Yes	ADD	Val_2	Val_3		
	Add3	No					
	Mul1	No					
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Val_3				Add2						

Clock: cycle 19

Tomasulo Example 2: Cycle 19

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	18
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-17	19
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Instruction 5 writes back.

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	Yes	ADD	Val_2	Val_3		
	Add3	No					
	Mul1	No					
	Mul2	No					

Mul2 sends result value Val_3 and id "Mul2" on to CDB. Instruction 5 releases Mul2.

Register Result Status:

	R0	R1	R	R8	R9	R10	... ...
FU		Val_3		Add2			

Clock: cycle 19

Tomasulo Example 2: Cycle 19

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	18
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-17	19
ADD	R5	R5	R1	6		

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	Yes	ADD	Val_2	Val_3		
	Add3	No					
	Mul1	No					
	Mul2	No					

Add2 sees id "Mul2" on CDB matches its source, picks up the value Val_3 and set it to V_k.

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Val_3				Add2						

Register R1 sees id "Mul2" on CDB, which matches its source, so it picks up the value Val_3.

Clock: c

Tomasulo Example 2: Cycle 20

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	18
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-17	19
ADD	R5	R5	R1	6	20	

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
1	Add2	Yes	ADD	Val_2	Val_3		
	Add3	No					
	Mul1	No					
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Val_3				Add2						

Clock: cycle 20

Tomasulo Example 2: Cycle 20

Instruction Status:

	Instructions			Stages		
Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	18
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-17	19
ADD	R5	R5	R1	6	20	

Instruction 6 starts executing.

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
1	Add2	Yes	ADD	Val_2	Val_3		
	Add3	No					
	Mul1	No					
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Val_3				Add2						

Clock: cycle 20

Tomasulo Example 2: Cycle 21

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	18
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-17	19
ADD	R5	R5	R1	6	20-21	

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
0	Add2	Yes	ADD	Val_2	Val_3		
	Add3	No					
	Mul1	No					
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Val_3				Add2						

Clock: cycle 21

Tomasulo Example 2: Cycle 21

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	18
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-17	19
ADD	R5	R5	R1	6	20-21	

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Instruction 6 finishes executing.

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
0	Add2	Yes	ADD	Val_2	Val_3		
	Add3	No					
	Mul1	No					
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Val_3				Add2						

Clock: cycle 21

Tomasulo Example 2: Cycle 22

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	18
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-17	19
ADD	R5	R5	R1	6	20-21	22

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	NO					
	Add3	No					
	Mul1	No					
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Val_3				Val_4						

Clock: cycle 22

Tomasulo Example 2: Cycle 22

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	18
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-17	19
ADD	R5	R5	R1	6	20-21	22

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Instruction 6 writes back.

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	NO					
	Add3	No					
	Mul1						
	Mul2						

Add2 sends result value Val_4 and id "Add2" on to CDB. Instruction 6 releases Add2.

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Val_3				Val_4						

Clock: cycle 22

Tomasulo Example 2: Cycle 22

Instruction Status:

Op	Dest *i*	Src1 *j*	Src2 *k*	IS	EX	WB
	Instructions			Stages		
LD	R1	0+	R11	1	2-3	4
MUL	R1	16	R1	2	5-14	15
ADD	R5	R5	R1	3	16-17	18
LD	R1	4	R11	4	5-6	7
MUL	R1	16	R1	5	8-17	19
ADD	R5	R5	R1	6	20-21	22

Load Buffers:

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations for the Adder and Multiplier

Time	Name	Busy	Op	Src1 Value V_j	Src2 Value V_k	Src1 RS Id Q_j	Src2 RS Id Q_k
	Add1	No					
	Add2	NO					
	Add3	No					
	Mul1	No					
	Mul2	No					

Register Result Status:

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	... ...
FU		Val_3				Val_4						

Clock: cycle 22

Register R5 sees id “Add2” on CDB, which matches its source, so it picks up the value Val_4.

Road Map

Tomasulo Review

- Reservation Stations
  - RS provides many more registers than those name-able in the assembly programming.
  - Distribute RAW hazard detection
  - Renaming eliminates WAW hazards
  - Buffering values in Reservation Stations removes WARs
  - Tag match in CDB requires many associative compares
- Common Data Bus
  - Broadcast results to RS and register file.
  - Can be a bottle neck: Multiple writebacks (multiple CDBs) expensive

Load/Store reordering

- In case a store and a load to the same memory address are reordered, store buffer has to forward the value to the load buffer.
  - That is, if *store* $R1$ and *load* $R1$ are reordered to *load* $R1$ and *store* $R1$, *load* $R1$ cannot be issued but wait for store buffer to forward the value.
- To support this forwarding, every load has to check the memory addresses in the store buffers before executing.

Tomasulo vs. Scoreboarding

1. No explicit checking for WAW or WAR hazards
2. CDB broadcasts results rather than waiting on registers
3. Loads/Store are treated like basic FUs
4. Distributed vs. Centralized control

Register Renaming: Pointer-Based

- Registers are renamed (mapped) to an internal register.
- A map table records this renaming/mapping.
- Mapper/Map Table: Hardware to hold these mappings
  - Register Writes: Allocate new location, note mapping in table
  - Register Reads: Look in map table, find location of most recent write
- Deallocate mappings when done
- Used in MIPS R10K, Alpha 21264, Pentium 4 and POWER4

Road Map

Overview of Tomasulo's Algorithm

An Example of Tomasulo's Algorithm

Another Example of Tomasulo's Algorithm

Tomasulo Wrapup

Acknowledgment

Acknowledgment

This lecture is partially based on the slides from Dr. David Brooks. The first Tomasulo's example was based ib Dr. Broderson's example, for CS152 at UCB (Copyright (C) 2001 UCB).