CS 223 Computer Organization & Architecture

Lecture 31 [29.04.2020]

Dynamic Scheduling with Tomasulo’s Algorithm

John Jose

Assistant Professor

Department of Computer Science & Engineering

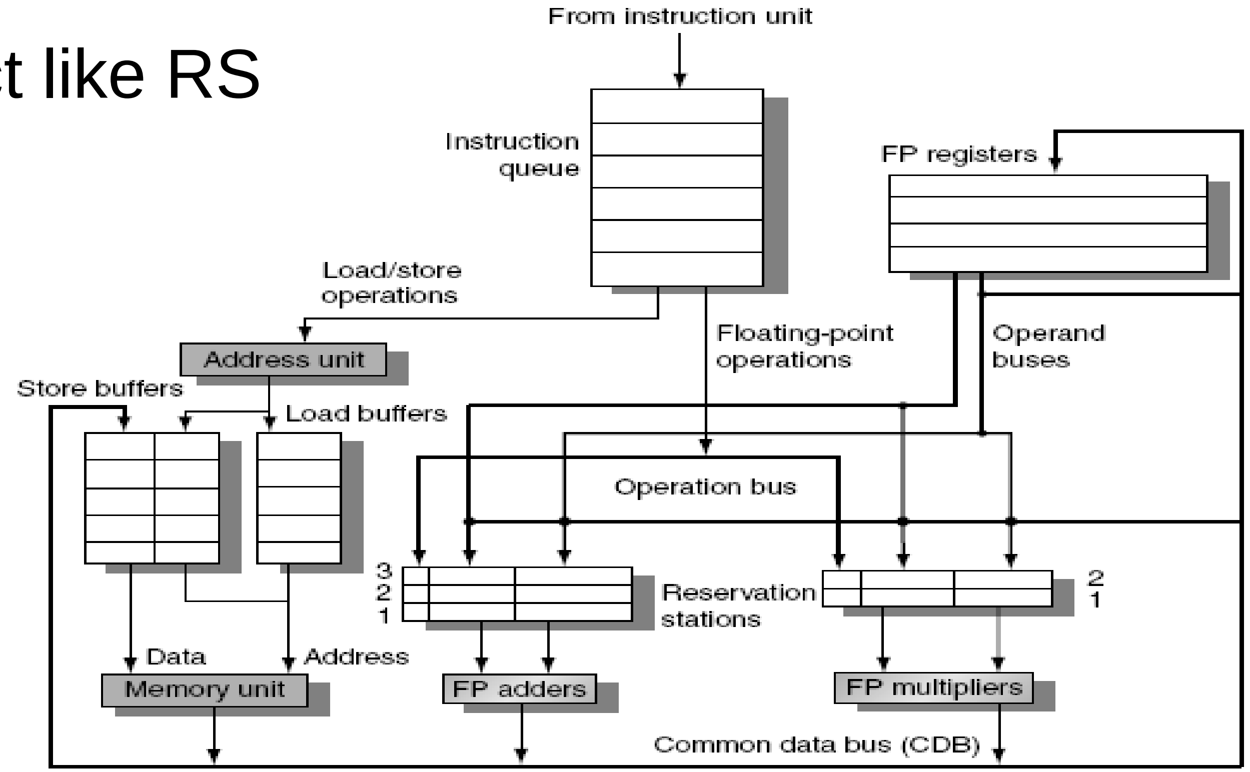
Indian Institute of Technology Guwahati, Assam.

How dynamic scheduling works ?

- To allow out-of-order execution, **split the ID stage into two**
  - **Issue**—Decode instructions, check for structural hazards.
  - **Read operands**—Wait until no data hazards, then read operands.
- In a dynamically scheduled pipeline, all instructions pass through the **issue stage in order** (in-order issue); however, they can be stalled or bypass each other in the second stage (read operands) and thus enter execution out of order.
- Done by - **score boarding technique**
- Approach used - **Tomasulo's algorithm**

Tomasulo’s Algorithm

- Load and store buffers contain data and addresses.
- They also act like RS



Dynamic Scheduling - Tomasulo

- **Issue**
  - Get next instruction from FIFO queue
  - If RS available, issue the instruction to the RS with operand values if available
  - If operand values not available, stall the instruction

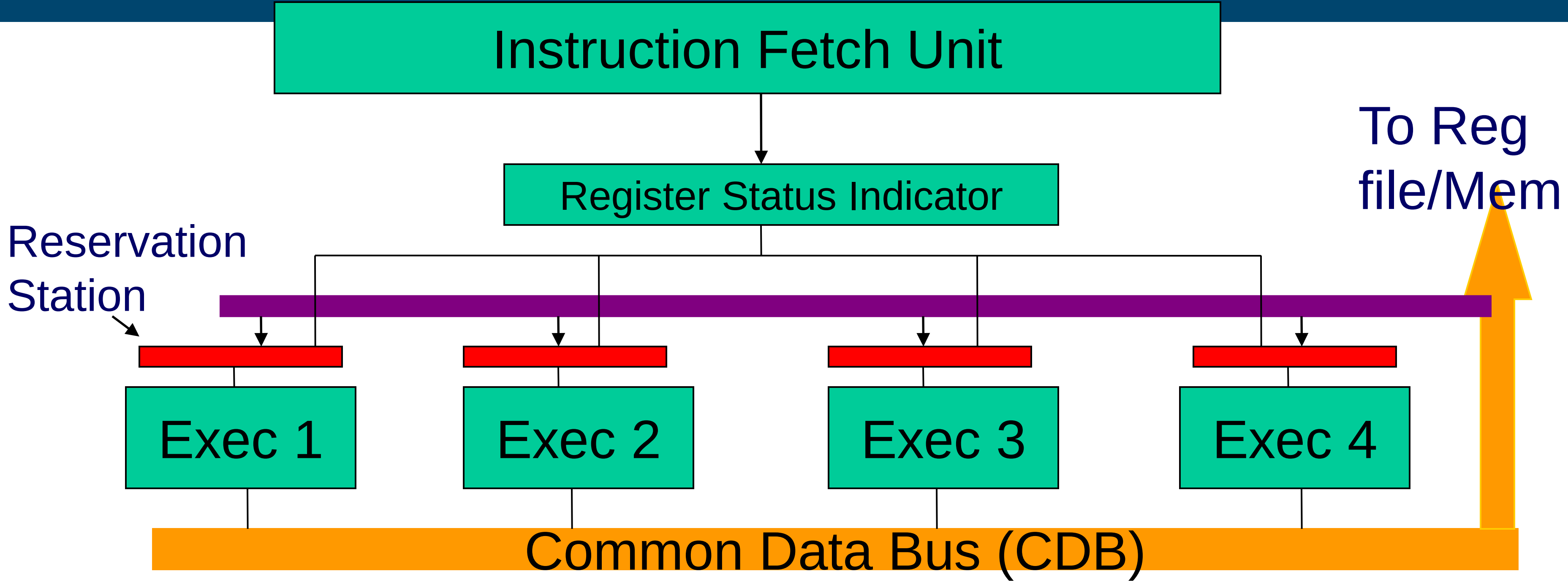
Dynamic Scheduling - Tomasulo

- **Execute**
  - When operand becomes available, store it in any reservation stations waiting for it
  - When all operands are ready, execute the instruction
  - Loads and store uses buffers
  - No instruction will initiate execution until all branches that precede it in program order have completed

Dynamic Scheduling - Tomasulo

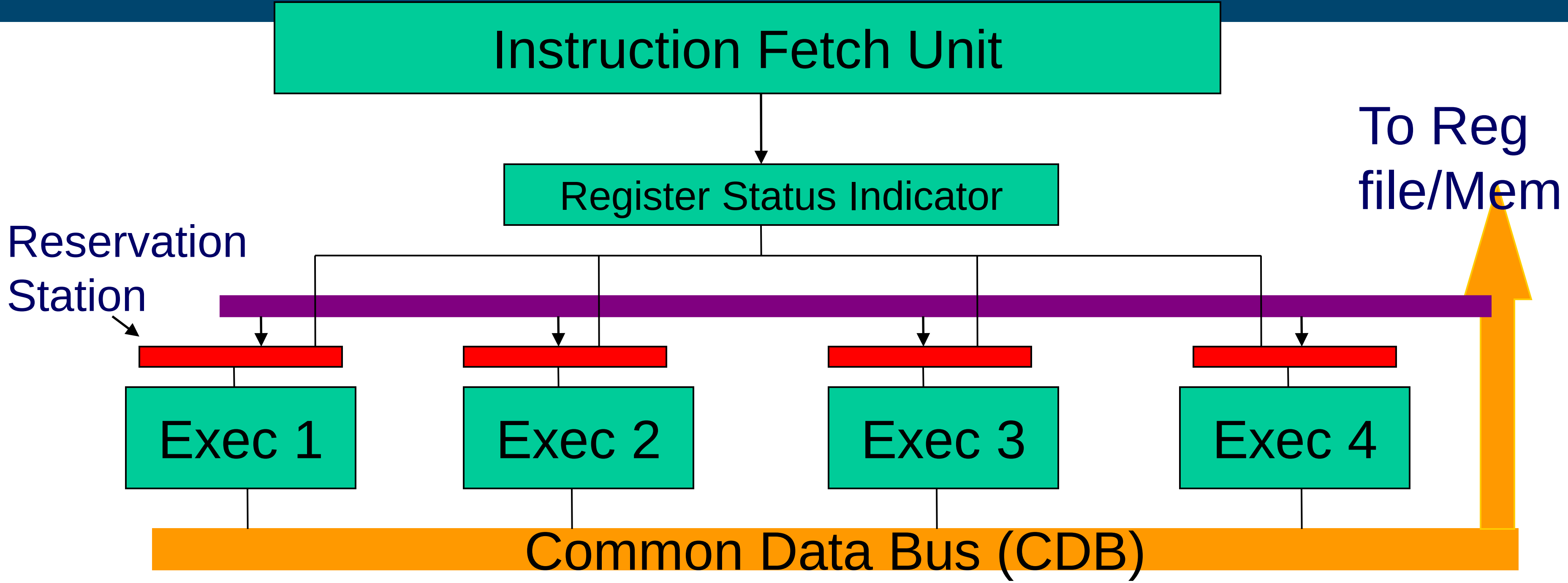
- **Write result**
  - Write result into CDB (there by it reaches the reservation station, store buffer and registers file) with name of execution unit that generated the result.
  - Stores must wait until address and value are received

Dynamic Scheduling - Tomasulo



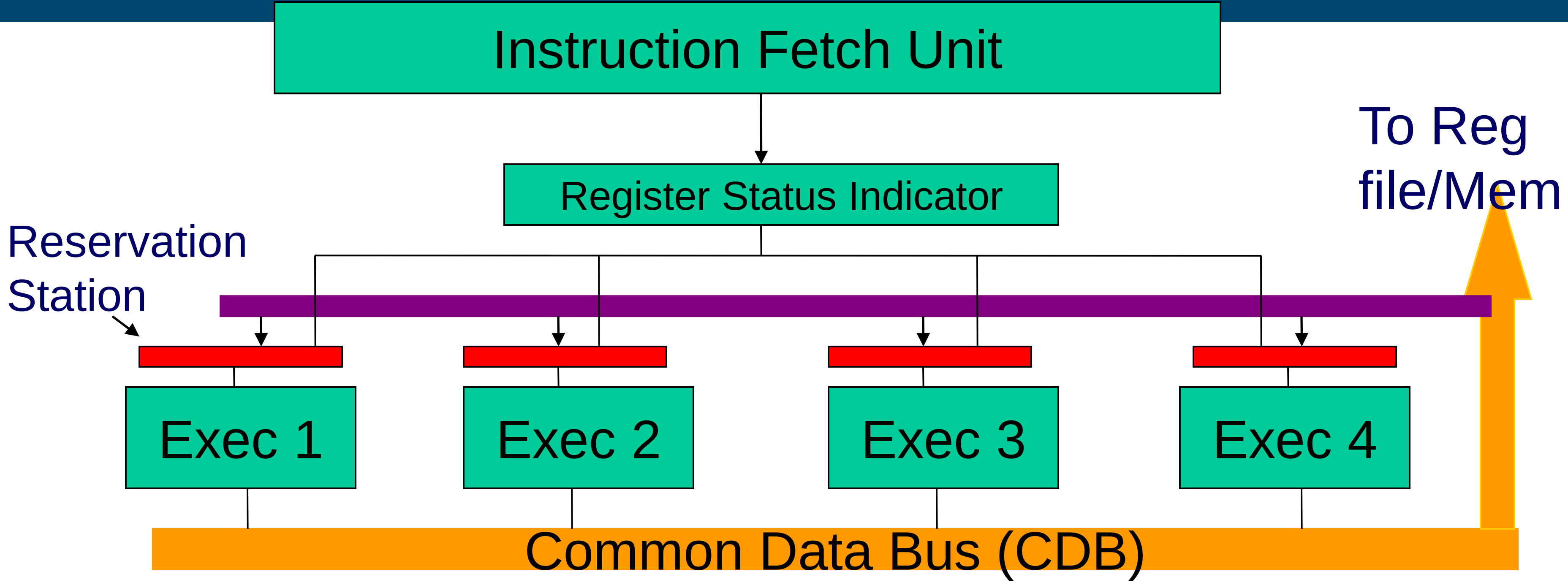
Instructions are fetched one by one and decoded to find the type of operation and the source of operands

Dynamic Scheduling - Tomasulo



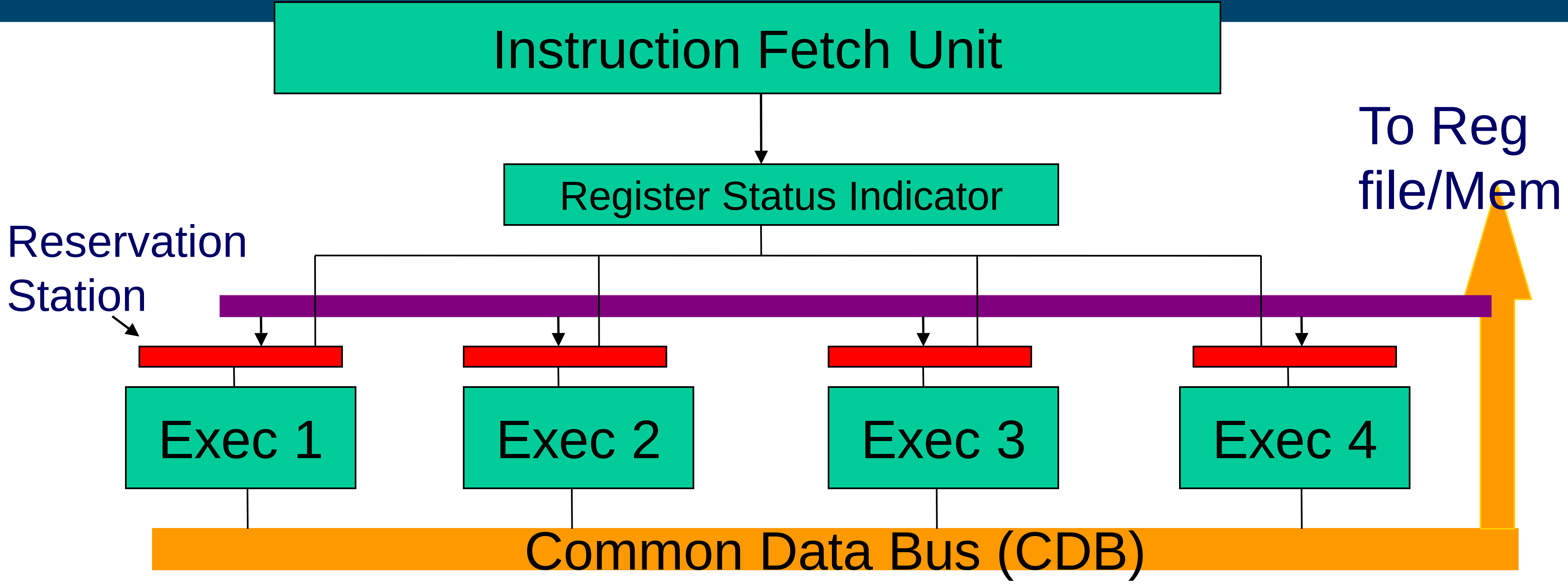
Register Status Indicator indicates whether the latest value of the register is in the reg file or currently being computed by some execution unit and if so, it states the execution unit number

Dynamic Scheduling - Tomasulo



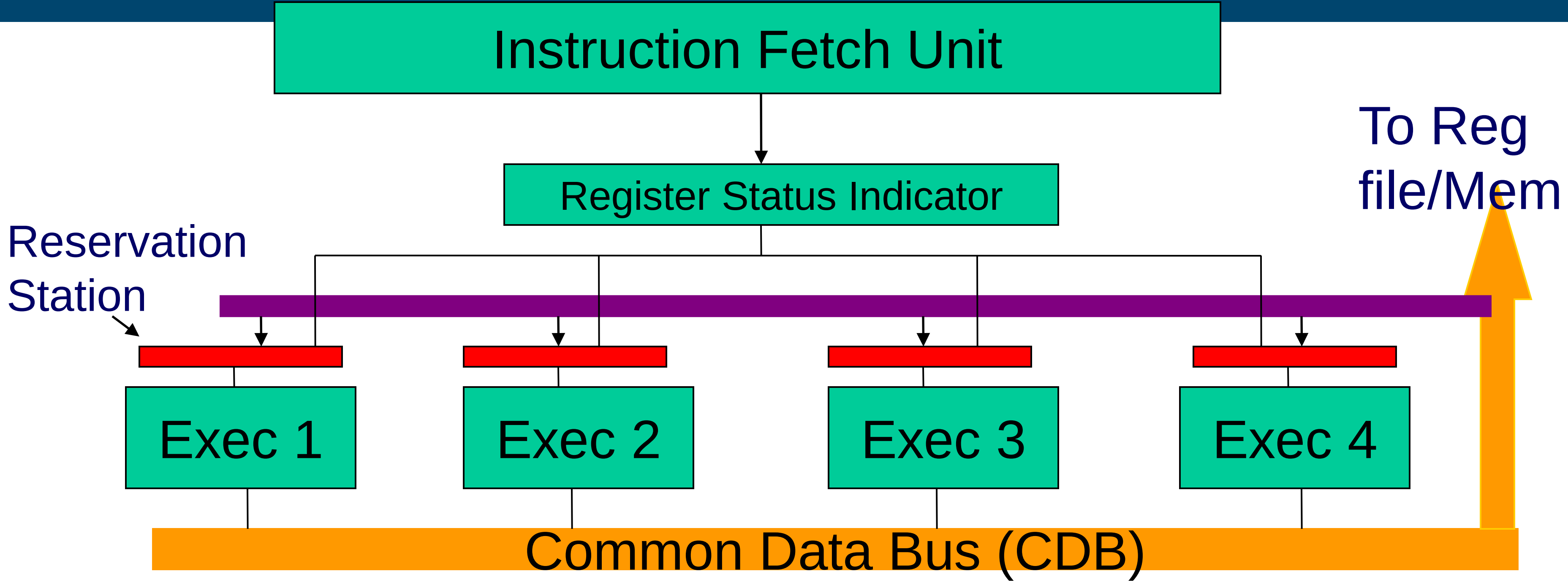
If all operands available then operation proceeds in the allotted execution unit, else, it waits in the reservation station of the allotted execution unit pinging the CDB

Dynamic Scheduling - Tomasulo



Every Execution unit writes the result along with the unit number on to the CDB which is forwarded to all reservation stations, Reg-file and Memory

Dynamic Scheduling - Tomasulo



Dynamic Scheduling - Tomasulo

An Example:

1. ADD R1, R2, R3
2. ST R1, [R4+50]
3. ADD R1, R5, R6
4. SUB R7,R1,R8
5. ST R1, [R4 + 54]
6. ADD R1, R9, R10

Instruction Fetch

Register Status Indicator

Reg Number	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
Status	0	0	0	0	0	0	0	0	0	0

Empty	Empty	Empty	Empty	Empty	Empty

Dynamic Scheduling - Tomasulo

An Example:

1. --
2. ST R1, [R4+50]
3. ADD R1, R5, R6
4. SUB R7,R1,R8
5. ST R1, [R4 + 54]
6. ADD R1, R9, R10

Instruction Fetch

ADD R1, R2, R3

Register Status Indicator

Reg Number	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
Status	1	0	0	0	0	0	0	0	0	0

Ins 1	Empty	Empty	Empty	Empty	Empty

Dynamic Scheduling - Tomasulo

An Example:

Instruction Fetch

ST R1, [R4+50]

1. ---
2. ---
3. ADD R1, R5, R6
4. SUB R7,R1,R8
5. ST R1, [R4 + 54]
6. ADD R1, R9, R10

Register Status Indicator

Reg Number	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
Status	1	0	0	0	0	0	0	0	0	0

I 1, E	I 2, W 1	Empty	Empty	Empty	Empty

Dynamic Scheduling - Tomasulo

An Example:

Instruction Fetch

ADD R1, R5, R6

1. ---
2. ---
3. ---
4. SUB R7,R1,R8
5. ST R1, [R4 + 54]
6. ADD R1, R9, R10

Register Status Indicator

Reg Number	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
Status	3	0	0	0	0	0	0	0	0	0

I 1, E	I 2, W 1	I 3, E	Empty	Empty	Empty

Note: Reservation Station stores the number of the execution unit that shall yield the latest value of a register.

Dynamic Scheduling - Tomasulo

An Example:

1. ---
2. ---
3. ---
4. ---
5. ST R1, [R4 + 54]
6. ADD R1, R9, R10

Instruction Fetch

SUB R7,R1,R8

Register Status Indicator

Reg Number	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
Status	3	0	0	0	0	0	4	0	0	0

I 1, E	I 2, W 1	I 3, E	I 4, W 3	Empty	Empty

Dynamic Scheduling - Tomasulo

An Example:

Instruction Fetch

ST R1, [R4 + 54]

1. ---
2. ---
3. ---
4. ---
5. ---
6. ADD R1, R9, R10

Register Status Indicator

Reg Number	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
Status	3	0	0	0	0	0	4	0	0	0

I 1, E	I 2, W 1	I 3, E	I 4, W 3	I 5, W 3	Empty

Dynamic Scheduling - Tomasulo

An Example:

1. ADD R1, R2, R3
2. ST U1, [R4+50]
3. ADD R1, R5, R6
4. SUB R7, U3, R8
5. ST U3, [R4 + 54]
6. ADD R1, R9, R10

Instruction Fetch

ADD R1, R9, R10

Register Status Indicator

Reg Number	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
Status	6	0	0	0	0	0	4	0	0	0

I 1, E	I 2, W 1	I 3, E	I 4, W 3	I 5, W 3	I 6, E

Effectively three Instructions are executing and others waiting for the appropriate results. The whole program is converted as shown above.

Dynamic Scheduling - Tomasulo

An Example:

1. ADD R1, R2, R3
2. ST U1, [R4+50]
3. ADD R1, R5, R6
4. SUB R7, U3, R8
5. ST U3, [R4 + 54]
6. ADD R1, R9, R10

Instruction Fetch

ADD R1, R9, R10

Register Status Indicator

Reg Number	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
Status	6	0	0	0	0	0	4	0	0	0

I 1, E	I 2, W 1	I 3, E	I 4, W 3	I 5, W 3	I 6, E

Operand Forwarding and Register Renaming is done automatically

Dynamic Scheduling - Tomasulo

An Example:

1. ADD R1, R2, R3
2. ST U1, [R4+50]
3. ADD R1, R5, R6
4. SUB R7, U3, R8
5. ST U3, [R4 + 54]
6. ADD R1, R9, R10

Instruction Fetch

ADD R1, R9, R10

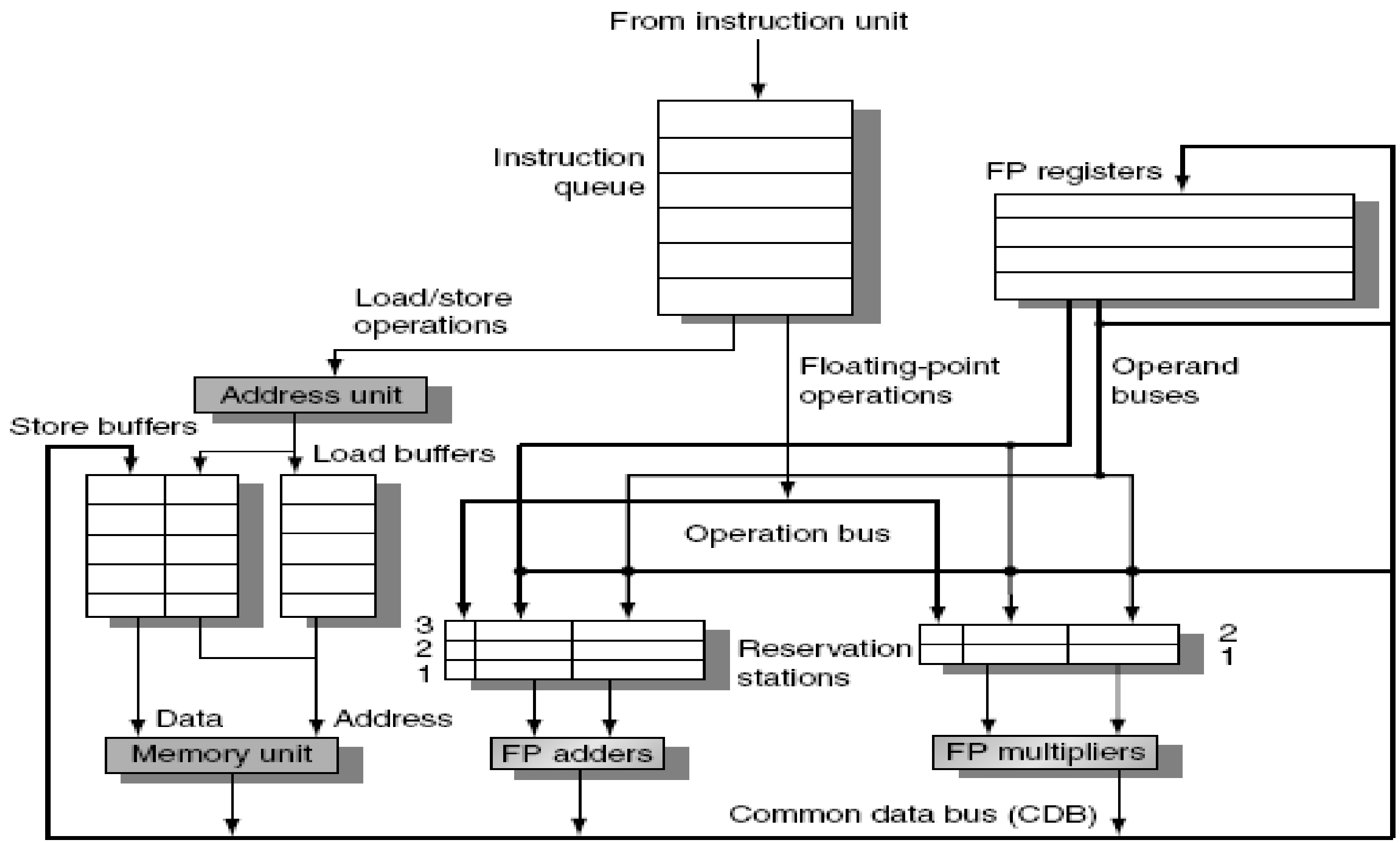
Register Status Indicator

Reg Number	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
Status	6	0	0	0	0	0	4	0	0	0

I 1, E	I 2, W 1	I 3, E	I 4, W 3	I 5, W 3	I 6, E

Execution unit 6, on completion will make R1 entry in Register Status Indicator 0. Similarly unit 4 will make R7 entry 0.

Tomasulo's Algorithm



Reservation Stations

- **Each reservation station has seven fields.**
- **{Op, Qj, Qk, Vj, Vk, A, Busy}**

1. **Op** —The operation to perform on source operands S1 and S2.

2,3. **Qj, Qk** —The reservation stations that will produce the corresponding source operand; a value of zero indicates that the source operand is already available in Vj or Vk.

4,5. **Vj, Vk** —The value of the source operands.

Only one of the **V field** or the **Q field** is valid for each operand. For loads, the Vk field is used to hold the offset field.

Reservation Stations

- ❖ **Each reservation station has seven fields.**
- ❖ **{Op, Qj, Qk, Vj, Vk, A, Busy}**

6. **A** —Used to hold information for the memory address calculation for a load or store. Initially, the immediate field of the instruction is stored here; after the address calculation, the effective address is stored here.

7. **Busy** —Indicates that this reservation station and its accompanying functional unit are occupied.

Register Status Indicator

- **The register file has a field, Qi: (RSI)**
- Qi—The number of the reservation station that contains the operation whose result should be stored into this register.
- If the value of Qi = 0 no currently active instruction is computing a result destined for this register, meaning that the value is simply the register contents.

Tomasulo’s Algorithm-Illustration

Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load-1	Yes	Load					32+Reg[R2]
Load-2	No						
Add-1	No						
Add-2	No						
Add-3	No						
Mult-1	No						
Mult-2	No						

```
L.D        F6,32(R2)
L.D        F2,44(R3)
MUL.D      F0,F2,F4
SUB.D      F8,F2,F6
DIV.D      F10,F0,F6
ADD.D      F6,F8,F2
```

F0	F2	F4	F6	F8	F10
0	0	0	Load-1	0	0

Tomasulo’s Algorithm-Illustration

Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load-1	Yes	Load					32+Reg[R2]
Load-2	**Yes**	**Load**					**44+Reg[R3]**
Add-1	No						
Add-2	No						
Add-3	No						
Mult-1	No						
Mult-2	No						

```
L.D       F6,32(R2)
L.D       F2,44(R3)
MUL.D     F0,F2,F4
SUB.D     F8,F2,F6
DIV.D     F10,F0,F6
ADD.D     F6,F8,F2
```

F0	F2	F4	F6	F8	F10
0	**Load-2**	0	Load-1	0	0

Tomasulo’s Algorithm-Illustration

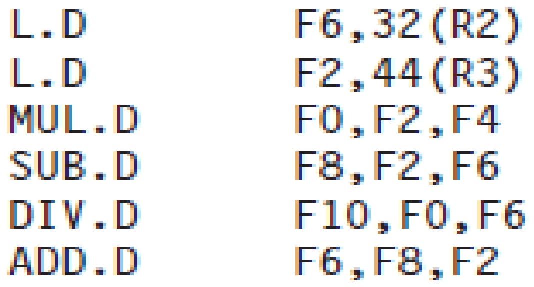
Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load-1	Yes	Load					32+Reg[R2]
Load-2	Yes	Load					44+Reg[R3]
Add-1	No						
Add-2	No						
Add-3	No						
Mult-1	**Yes**	**MUL**		**Reg[F4]**	**Load-2**		
Mult-2	No						

```
L.D        F6,32(R2)
L.D        F2,44(R3)
MUL.D      F0,F2,F4
SUB.D      F8,F2,F6
DIV.D      F10,F0,F6
ADD.D      F6,F8,F2
```

F0	F2	F4	F6	F8	F10
Mult-1	Load-2	0	Load-1	0	0

Tomasulo’s Algorithm-Illustration

Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load-1	Yes	Load					32+Reg[R2]
Load-2	Yes	Load					44+Reg[R3]
Add-1	**Yes**	**SUB**			**Load-1**	**Load-2**	
Add-2	No						
Add-3	No						
Mult-1	Yes	MUL		Reg[F4]	Load-2		
Mult-2	No						



```
L.D        F6,32(R2)
L.D        F2,44(R3)
MUL.D      F0,F2,F4
SUB.D      F8,F2,F6
DIV.D      F10,F0,F6
ADD.D      F6,F8,F2
```

F0	F2	F4	F6	F8	F10
Mult-1	Load-2	0	Load-1	**Add-1**	0

Tomasulo’s Algorithm-Illustration

Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load-1	Yes	Load					32+Reg[R2]
Load-2	Yes	Load					44+Reg[R3]
Add-1	Yes	SUB			Load-1	Load-2	
Add-2	No						
Add-3	No						
Mult-1	Yes	MUL		Reg[F4]	Load-2		
Mult-2	**Yes**	**DIV**			**Mult-1**	**Load-1**	

```
L.D       F6,32(R2)
L.D       F2,44(R3)
MUL.D     F0,F2,F4
SUB.D     F8,F2,F6
DIV.D     F10,F0,F6
ADD.D     F6,F8,F2
```

F0	F2	F4	F6	F8	F10
Mult-1	Load-2	0	Load-1	Add-1	**Mult-2**

Tomasulo's Algorithm-Illustration

Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load-1	Yes	Load					32+Reg[R2]
Load-2	Yes	Load					44+Reg[R3]
Add-1	Yes	SUB			Load-1	Load-2	
Add-2	**Yes**	**ADD**			**Add-1**	**Load-2**	
Add-3	No						
Mult-1	Yes	MUL		Reg[F4]	Load-2		
Mult-2	Yes	DIV			Mult-1	Load-1	

```
L.D         F6,32(R2)
L.D         F2,44(R3)
MUL.D       F0,F2,F4
SUB.D       F8,F2,F6
DIV.D       F10,F0,F6
ADD.D       F6,F8,F2
```

F0	F2	F4	F6	F8	F10
Mult-1	Load-2	0	**Add-2**	Add-1	Mult-2

johnjose@iitg.ac.in

http://www.iitg.ac.in/johnjose/