Course on: “Advanced Computer Architectures”

# Scoreboard Dynamic Scheduling Algorithm


Prof. Cristina Silvano
Politecnico di Milano
email: cristina.silvano@polimi.it

# Recap on Dynamic Scheduling

- **Simple scalar pipeline:** Hazards due to true data dependences that cannot be solved by forwarding cause the stall of the pipeline: no new instructions can be fetched nor issued even if there are not data dependences!

- **Solution: Allow data independent instructions behind a stall to proceed**
  - HW rearranges dynamically the instruction execution to reduce stalls

**=> This enables out-of-order execution and out-of-order commit.**

- First implemented in CDC 6600 (1963).

# Example of Dynamic Scheduling

```
DIVD  F0,F2,F4     # takes many cycles
ADDD  F10,F0,F8    # RAW F0
SUBD  F12,F8,F14
```

- RAW hazard: **ADDD** stalls for RAW hazards on **F0** (stall many cycles for **DIVD** commit).
- **SUBD** would stall even if not data dependent on anything in the pipeline.

- **BASIC IDEA: to enable SUBD to proceed**
  **=> this generates *out-of-order execution***

# Scoreboard Basic Assumptions

- We consider a ***single-issue*** processor.
- Instruction Fetch stage fetches and issues instructions in program order ***(in-order issue)***.
- Instruction execution begins ***as soon as operands are ready*** whenever not dependent on previous instructions *(no **RAW** hazards)*.
- There are ***multiple*** pipelined Functional Units with ***variable latencies.***
- Execution stage might require ***multiple cycles,*** depending on the operation type and latency.
- Memory stage might require ***multiple cycles*** access time due to data cache misses.

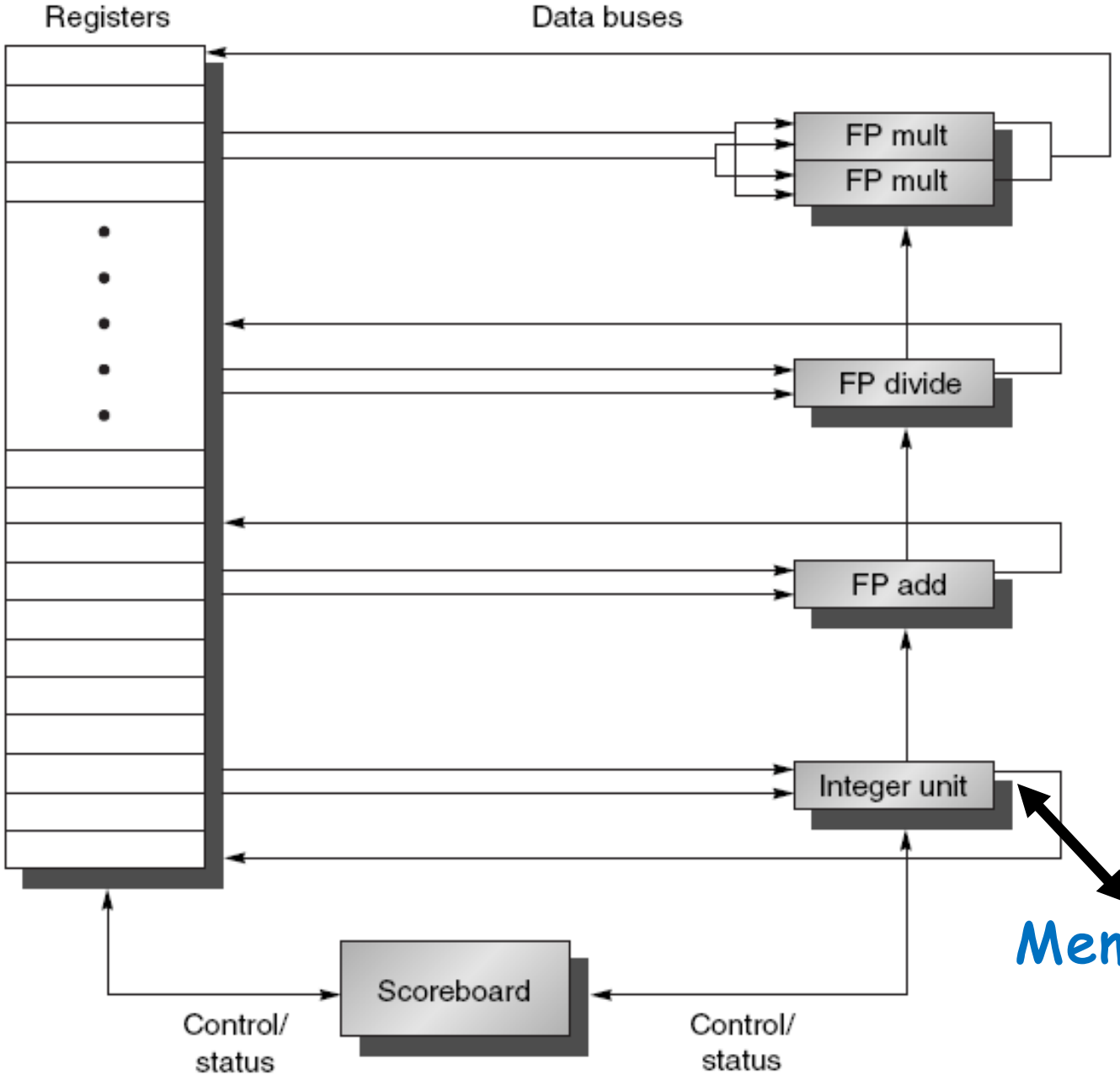
⇒ ***Out-of-order execution & out-of-order commit***

*(this introduces the possibility of **WAR** & **WAW** hazards)*.

# Scoreboard basic scheme

- Scoreboard allows ***data independent instructions behind a stall to proceed,*** not waiting for prior instructions.
- We distinguish when an instruction begins execution and it completes execution: between the two times, the instruction is ***in execution.***
- Scoreboard pipeline allows ***multiple instructions in execution at the same time*** $\Rightarrow$ that requires multiple pipelined functional units.
- ***In-order issue, out-of-order execution, out-of-order completion (commit)***
  - ***No forwarding!***
  - ***Imprecise interrupt/exception model for now!***

# Scoreboard basic architecture




Functional Units

Memory Unit

# Scoreboard Pipeline Stages

- Scoreboard divides the **ID** stage in **two stages:**
  1. **Issue**—Decode instructions and check for structural hazards
  2. **Read operands (RR)**—Wait until not dependent on previous instructions and no data hazards, then read operands

# Scoreboard Pipeline Stages

- Scoreboard divides the ID stage in two stages:
  1. Issue—Decode instructions and check for structural hazards
  2. Read operands (RR)—Wait until not dependent on previous instructions and no data hazards, then read operands

- Scoreboard allows instructions to execute whenever 1 & 2 hold, not waiting for prior instructions to complete.
- Scoreboard keeps track of **dependencies** and state of **parallel ongoing operations.**
- Instructions pass through the issue stage ***in-order,*** but they can be stalled or bypass each other in the read operand stage ***(out-of-order read operands).***
- Then instructions enter execution ***out-of-order*** and have different latencies, which implies ***out-of-order completion (commit).***
- **Summary: In-order *issue* but out-of-order *read-operands* ⇒ *out-of-order execution & commit.***

# Scoreboard Implications

- There are multiple instructions in execution phase
  ➔ Multiple execution units or pipelined execution units
- No register renaming (compile time technique).
- Out-of-order commit ⇒ **WAR and WAW hazards can occur**

- **Solutions for WAR:**
  - Read registers only during Read Operands stage.
  - Stall write back until previous registers have been read.

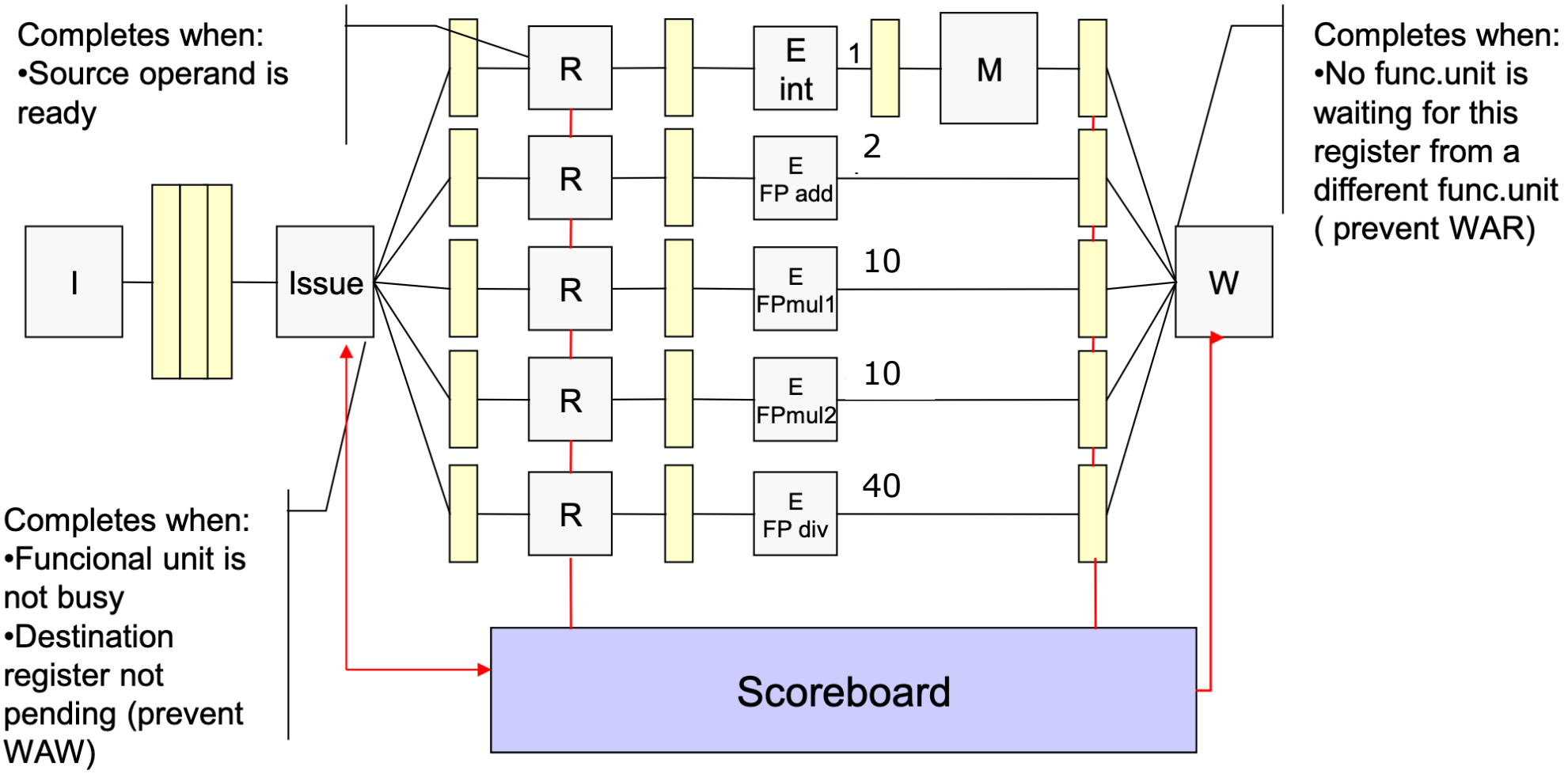
- **Solution for WAW:**
  - Detect WAW hazard and stall issue of new instruction until previous instruction causing WAW completes.

# Scoreboard Scheme

- **Any hazard detection and resolution is centralized in the Scoreboard:**
  - Every instruction goes through the Scoreboard, where a record of data dependences is constructed
  - The Scoreboard then determines **when** the instruction can read its operand and begin execution **(check for RAW)**
  - If the Scoreboard decides the instruction cannot execute immediately, it monitors every change and decides **when** the instruction can execute.
  - The scoreboard controls **when** the instruction can write its result into destination register **(check for WAR & WAW)**

# Scoreboard Architecture

- The idea of a scoreboard is to keep track of the status of instructions, functional units and registers



# Four Stages of Scoreboard Control

1. **Issue**
   **Decode instruction and check for structural hazards & WAW hazards**

   **Instructions issued in program order (for hazard checking)**

   - If a functional unit for the instruction is available **(no structural hazard)** and no other active instruction has the same destination register **(no WAW hazard)** => the Scoreboard issues the instruction to the FU and updates its data structure.

   - If either a **structural hazard** or a **WAW hazard** exists => the instruction issue stalls, and no further instructions will issue until these hazards are solved.

# Four Stages of Scoreboard Control

2. **Read Operands**

**Wait until no RAW hazards => then read operands.**
**Check for structural hazards in reading ports of RF.**

- A source operand is available if:
  - No earlier issued active instruction will write it or
  - A functional unit is writing its value in a register
- When the source operands are available, the Scoreboard tells the FU to proceed to read the operands from the RF and begin execution.
- RAW hazards are solved dynamically in this step
  => **out-of-order** reading of operands
  => instructions are sent into execution **out-of-order.**
- **No data forwarding**

# Four Stages of Scoreboard Control

3. **Execution**

   **The FU begins execution upon receiving operands.**
   **When the result is ready, it notifies the Scoreboard that execution has been completed.**

- FUs are characterized by **variable latency** to complete execution.
- **Load/Store latency depends on data cache HIT/MISS times.**

**=> Out-of-order execution**

# Four Stages of Scoreboard Control

4. **Write result**

**Check for WAR hazards on destination.**

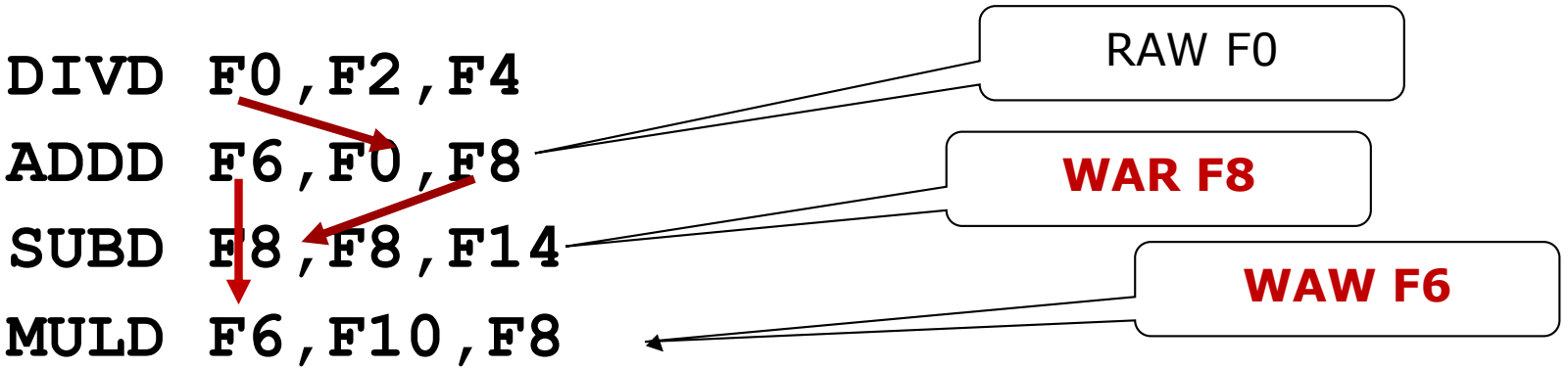
**Check for structural hazards in writing RF and finish execution.**

Once the Scoreboard is aware that the FU has completed execution, **the Scoreboard checks for WAR hazards.**

- If none, it writes results.
- If there is a **WAR** => the Scoreboard stalls the completing instruction.

**=> Out-of-order commit**

# RAW/WAR/WAW Example




- To avoid the **WAR** hazard on F8, the Scoreboard would:
  - Stall **SUBD** in the WB stage, waiting for **ADDD** reads F0 and F8;
- To avoid the **WAW** hazard on F6, the Scoreboard would:
  - Stall **MULD** in the ISSUE stage until **ADDD** writes F6.

- Note: Any WAR/WAW hazard could have been solved through register renaming at compile time.

# Recap: SCOREBOARD BASIC SCHEME

- IN-ORDER ISSUE
- OUT-OF-ORDER READ OPERANDS
- OUT-OF-ORDER EXECUTION
- OUT-OF-ORDER COMPLETION
- NO FORWARDING
- Control is centralized into the Scoreboard

# Recap: SCOREBOARD STAGES

- **ISSUE (IN-ORDER):**
  - Check for structural hazards
  - Check for WAW hazards on destination operand **(*)**
- **READ OPERANDS (OUT-OF-ORDER)**
  - Check for RAW hazards
  - Check for structural hazards in reading RF
- **EXECUTION (OUT-OF-ORDER)**
  - Execution completion depends on latency of FUs
  - Execution completion of LD/ST depends on cache hit/miss latencies
- **WRITE RESULTS (OUT-OF-ORDER)**
  - Check for WAR hazards on destionation operand
  - Check for structural hazards in writing RF

# Recap: SCOREBOARD optimizations

## (*) Optimizations:

1. Check for **WAW** **postponed** from **ISSUE** stage to **WRITE** stage
2. Data forwarding

# Scoreboard Structure

1. **Instruction status**
2. **Functional Unit status**

   Indicates the state of the functional unit (FU):

   | | |
   |---|---|
   | **Busy** | Indicates whether the unit is busy or not |
   | **Op** | The operation to perform in the unit (+,-, etc.) |
   | **Fi** | Destination register |
   | **Fj, Fk** | Source register numbers |
   | **Qj, Qk** | Functional units producing source registers Fj, Fk |
   | **Rj, Rk** | Flags indicating when Fj, Fk are ready.<br>Flags are set to NO after operands are read. |

3. **Register result status**

   Indicates which functional unit will write each register. Blank if no pending instructions will write that register.

# Scoreboard Example: Analysis of dependences and hazards

```
LD   F6, 34(R2)
LD   F2, 45(R3)
MULTD F0, F2, F4    # RAW F2
SUBD  F8, F6, F2    # RAW F2, RAW F6
DIVD  F10, F0, F6   # RAW F0, RAW F6
ADDD  F6, F8, F2    # WAW F6, WAR F6,
                    # RAW F8, RAW F2
```

# Scoreboard Example

*Instruction status:*

| Instruction | | *j* | *k* | *Issue* | *Read Oper* | *Exec Comp* | *Write Result* |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | | | | |
| LD | F2 | 45+ | R3 | | | | |
| MULTD | F0 | F2 | F4 | | | | |
| SUBD | F8 | F6 | F2 | | | | |
| **DIVD** | **F10** | **F0** | **F6** | | | | |
| ADDD | F6 | F8 | F2 | | | | |

*Functional unit status:*

| *Time* | *Name* | *Busy* | *Op* | *dest Fi* | *S1 Fj* | *S2 Fk* | *FU Qj* | *FU Qk* | *Fj? Rj* | *Fk? Rk* |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| | Mult1 | No | | | | | | | | |
| | Mult2 | No | | | | | | | | |
| | Add | No | | | | | | | | |
| | Divide | No | | | | | | | | |

*Register result status:*

| Clock | *F0* | *F2* | *F4* | *F6* | *F8* | *F10* | *F12* | ... | *F30* |
|---|---|---|---|---|---|---|---|---|---|
| *FU* | | | | | | | | | |

# Scoreboard Example: Cycle 1

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | | | |
| LD | F2 | 45+ | R3 | | | | |
| MULTD | F0 | F2 | F4 | | | | |
| SUBD | F8 | F6 | F2 | | | | |
| **DIVD** | **F10** | **F0** | **F6** | | | | |
| ADDD | F6 | F8 | F2 | | | | |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | Yes | Load | F6 | | R2 | | | | Yes |
| | Mult1 | No | | | | | | | | |
| | Mult2 | No | | | | | | | | |
| | Add | No | | | | | | | | |
| | Divide | No | | | | | | | | |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | *FU* | | | | Integer | | | | | |

# Scoreboard Example Cycle 2

Instruction status

| Instruction | j | k | Issue | Read operand | Executi comple | Write Result |
|---|---|---|---|---|---|---|
| LD F6 | 34+ | R2 | 1 | 2 | | |
| LD F2 | 45+ | R3 | | | | |
| MULTI F0 | F2 | F4 | | | | |
| SUBD F8 | F6 | F2 | | | | |
| DIVD F10 | F0 | F6 | | | | |
| ADDD F6 | F8 | F2 | | | | |

Functional unit status

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU for j Qj | FU for k Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | Yes | Load | F6 | | R2 | | | | Yes |
| | Mult1 | No | | | | | | | | |
| | Mult2 | No | | | | | | | | |
| | Add | No | | | | | | | | |
| | Divide | No | | | | | | | | |

Register result status

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | FU | | | | Integer | | | | | |

***Issue 2nd load?*** **No: Integer Unit busy – Cannot issue 2nd Load due to structural hazard on Integer Unit => *Issue stalls***

***Issue multiply?***

# Scoreboard Example Cycle 3

Instruction status

| Instruction | | j | k | Issue | Read operand | Execution complete | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | |
| LD | F2 | 45+ | R3 | | | | |
| MULTI | F0 | F2 | F4 | | | | |
| SUBD | F8 | F6 | F2 | | | | |
| DIVD | F10 | F0 | F6 | | | | |
| ADDD | F6 | F8 | F2 | | | | |

Functional unit status

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU for j Qj | FU for k Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | Yes | Load | F6 | | R2 | | | | Yes |
| | Mult1 | No | | | | | | | | |
| | Mult2 | No | | | | | | | | |
| | Add | No | | | | | | | | |
| | Divide | No | | | | | | | | |

Register result status

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| 3 | FU | | | | Integer | | | | | |

- ***Issue stalls***
- **Load execution complete in one clock cycle (ideal data cache hit)**

# Scoreboard Example: Cycle 4

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | | | | |
| MULTD | F0 | F2 | F4 | | | | |
| SUBD | F8 | F6 | F2 | | | | |
| DIVD | F10 | F0 | F6 | | | | |
| ADDD | F6 | F8 | F2 | | | | |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| | Mult1 | No | | | | | | | | |
| | Mult2 | No | | | | | | | | |
| | Add | No | | | | | | | | |
| | Divide | No | | | | | | | | |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| 4 | FU | | | | Integer | | | | | |

- ***Issue stalls***
- **Write F6 & Integer Unit no more busy**

# Scoreboard Example: Cycle 5

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | | | |
| MULTD | F0 | F2 | F4 | | | | |
| SUBD | F8 | F6 | F2 | | | | |
| DIVD | F10 | F0 | F6 | | | | |
| ADDD | F6 | F8 | F2 | | | | |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | Yes | Load | F2 | | R3 | | | | Yes |
| | Mult1 | No | | | | | | | | |
| | Mult2 | No | | | | | | | | |
| | Add | No | | | | | | | | |
| | Divide | No | | | | | | | | |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| 5 | FU | | Integer | | | | | | | |

- **The second load is issued**

# Scoreboard Example: Cycle 6

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | | |
| MULTD | F0 | F2 | F4 | 6 | | | |
| SUBD | F8 | F6 | F2 | | | | |
| DIVD | F10 | F0 | F6 | | | | |
| ADDD | F6 | F8 | F2 | | | | |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | Yes | Load | F2 | | R3 | | | | Yes |
| | Mult1 | Yes | Mult | F0 | F2 | F4 | Integer | | No | Yes |
| | Mult2 | No | | | | | | | | |
| | Add | No | | | | | | | | |
| | Divide | No | | | | | | | | |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| 6 | FU | Mult1 | Integer | | | | | | | |

- **MULT is issued, but it has to wait for F2 from 2nd LOAD (RAW Hazard on F2)**

# Scoreboard Example: Cycle 7

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | |
| MULTD | F0 | F2 | F4 | 6 | | | |
| SUBD | F8 | F6 | F2 | 7 | | | |
| DIVD | F10 | F0 | F6 | | | | |
| ADDD | F6 | F8 | F2 | | | | |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | Yes | Load | F2 | | R3 | | | | Yes |
| | Mult1 | Yes | Mult | F0 | F2 | F4 | Integer | | No | Yes |
| | Mult2 | No | | | | | | | | |
| | Add | Yes | Sub | F8 | F6 | F2 | | Integer | Yes | No |
| | Divide | No | | | | | | | | |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| 7 | FU | Mult1 | Integer | | | Add | | | | |

- **Load execution completed in one clock cycle (data cache hit)**
- **Read multiply operands? *Not yet***
- **SUBD can be issued to ADD Functional Unit (then SUBD has to wait for RAW F2 from load)**

# Scoreboard Example: Cycle 8

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | |
| MULTD | F0 | F2 | F4 | 6 | | | |
| SUBD | F8 | F6 | F2 | 7 | | | |
| **DIVD** | **F10** | **F0** | **F6** | 8 | | | |
| ADDD | F6 | F8 | F2 | | | | |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | Yes | Load | F2 | | R3 | | | | No |
| | Mult1 | Yes | Mult | F0 | F2 | F4 | Integer | | No | Yes |
| | Mult2 | No | | | | | | | | |
| | Add | Yes | Sub | F8 | F6 | F2 | | Integer | Yes | No |
| | Divide | Yes | **Div** | **F10** | **F0** | **F6** | Mult1 | | **No** | **Yes** |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **8** | *FU* | Mult1 | Integer | | | Add | **Divide** | | | |

- **DIVD is issued but there is another RAW hazard (F0) from MULTD**
  **-> DIVD has to wait for reading F0**

# Scoreboard Example: Cycle 8 cont'd

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | | | |
| SUBD | F8 | F6 | F2 | 7 | | | |
| **DIVD** | **F10** | **F0** | **F6** | 8 | | | |
| ADDD | F6 | F8 | F2 | | | | |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| | Mult1 | Yes | Mult | F0 | F2 | F4 | | | Yes | Yes |
| | Mult2 | No | | | | | | | | |
| | Add | Yes | Sub | F8 | F6 | F2 | | | Yes | Yes |
| | Divide | Yes | **Div** | **F10** | **F0** | **F6** | Mult1 | | **No** | **Yes** |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **8** | *FU* | Mult1 | | | | Add | **Divide** | | | |

- **Load completes (Writes F2), and F2 operands for MULT and SUBD are ready**

# Scoreboard Example: Cycle 9

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | | |
| SUBD | F8 | F6 | F2 | 7 | 9 | | |
| **DIVD** | **F10** | **F0** | **F6** | 8 | | | |
| ADDD | F6 | F8 | F2 | | | | |

*Functional unit status:*

Note Remaining

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| 10 | Mult1 | Yes | Mult | F0 | F2 | F4 | | | Yes | Yes |
| | Mult2 | No | | | | | | | | |
| 2 | Add | Yes | Sub | F8 | F6 | F2 | | | Yes | Yes |
| | Divide | Yes | **Div** | **F10** | **F0** | **F6** | Mult1 | | **No** | **Yes** |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **9** | *FU* | Mult1 | | | | Add | **Divide** | | | |

- **Read operands for MULTD & SUBD by multiple-port Register File (4 read ports)**
- **Issue ADDD? WAW F6 is gone but there is a structural hazard on ADD Functional Unit**
- **MULTD & SUBD are sent in execution in parallel with latency 10 cycles for MULTD & 2 cycles for SUBD**

# Scoreboard Example: Cycle 10

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | | |
| SUBD | F8 | F6 | F2 | 7 | 9 | | |
| **DIVD** | **F10** | **F0** | **F6** | 8 | | | |
| ADDD | F6 | F8 | F2 | | | | |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| 9 | Mult1 | Yes | Mult | F0 | F2 | F4 | | | Yes | Yes |
| | Mult2 | No | | | | | | | | |
| 1 | Add | Yes | Sub | F8 | F6 | F2 | | | Yes | Yes |
| | Divide | Yes | Div | F10 | F0 | F6 | Mult1 | | No | Yes |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **10** | *FU* | Mult1 | | | | Add | Divide | | | |

# Scoreboard Example: Cycle 11

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | | |
| SUBD | F8 | F6 | F2 | 7 | 9 | 11 | |
| **DIVD** | **F10** | **F0** | **F6** | 8 | | | |
| ADDD | F6 | F8 | F2 | | | | |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| 8 | Mult1 | Yes | Mult | F0 | F2 | F4 | | | Yes | Yes |
| | Mult2 | No | | | | | | | | |
| 0 | Add | Yes | Sub | F8 | F6 | F2 | | | Yes | Yes |
| | Divide | Yes | Div | F10 | F0 | F6 | Mult1 | | No | Yes |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **11** | *FU* | Mult1 | | | | Add | Divide | | | |

- **SUBD ends execution**

# Scoreboard Example: Cycle 12

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | | |
| SUBD | F8 | F6 | F2 | 7 | 9 | 11 | 12 |
| **DIVD** | **F10** | **F0** | **F6** | 8 | | | |
| ADDD | F6 | F8 | F2 | | | | |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| 7 | Mult1 | Yes | Mult | F0 | F2 | F4 | | | Yes | Yes |
| | Mult2 | No | | | | | | | | |
| | Add | No | | | | | | | | |
| | Divide | Yes | Div | F10 | F0 | F6 | Mult1 | | No | Yes |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **12** | *FU* | Mult1 | | | | | Divide | | | |

- **SUBD writes result in F8**

# Scoreboard Example: Cycle 13

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | | |
| SUBD | F8 | F6 | F2 | 7 | 9 | 11 | 12 |
| **DIVD** | **F10** | **F0** | **F6** | 8 | | | |
| ADDD | F6 | F8 | F2 | 13 | | | |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| 6 | Mult1 | Yes | Mult | F0 | F2 | F4 | | | Yes | Yes |
| | Mult2 | No | | | | | | | | |
| | Add | Yes | Add | F6 | F8 | F2 | | | Yes | Yes |
| | Divide | Yes | Div | F10 | F0 | F6 | Mult1 | | No | Yes |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | … | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **13** | *FU* | Mult1 | | | Add | | Divide | | | |

- **ADDD can be issued (WAW F6 was gone and ADD unit is available)**
- **DIVD still waits for operand F0 from MULTD**

# Scoreboard Example: Cycle 14

*Instruction status:*

| Instruction | | j | k | *Issue* | *Read Oper* | *Exec Comp* | *Write Result* |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | | |
| SUBD | F8 | F6 | F2 | 7 | 9 | 11 | 12 |
| **DIVD** | **F10** | **F0** | **F6** | 8 | | | |
| ADDD | F6 | F8 | F2 | 13 | 14 | | |

*Functional unit status:*

| *Time* | *Name* | *Busy* | *Op* | *dest Fi* | *S1 Fj* | *S2 Fk* | *FU Qj* | *FU Qk* | *Fj? Rj* | *Fk? Rk* |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| 5 | Mult1 | Yes | Mult | F0 | F2 | F4 | | | Yes | Yes |
| | Mult2 | No | | | | | | | | |
| 2 | Add | Yes | Add | F6 | F8 | F2 | | | Yes | Yes |
| | Divide | Yes | Div | F10 | F0 | F6 | Mult1 | | No | Yes |

*Register result status:*

| Clock | | *F0* | *F2* | *F4* | *F6* | *F8* | *F10* | *F12* | ... | *F30* |
|---|---|---|---|---|---|---|---|---|---|---|
| **14** | *FU* | Mult1 | | | Add | | Divide | | | |

- **ADDD reads operands (out-of-order read operands: ADDD reads operands before DIVD)**

# Scoreboard Example: Cycle 15

*Instruction status:*

| Instruction | | *j* | *k* | *Issue* | *Read Oper* | *Exec Comp* | *Write Result* |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | | |
| SUBD | F8 | F6 | F2 | 7 | 9 | 11 | 12 |
| **DIVD** | **F10** | **F0** | **F6** | 8 | | | |
| ADDD | F6 | F8 | F2 | 13 | 14 | | |

*Functional unit status:*

| *Time* | *Name* | *Busy* | *Op* | *dest Fi* | *S1 Fj* | *S2 Fk* | *FU Qj* | *FU Qk* | *Fj? Rj* | *Fk? Rk* |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| 4 | Mult1 | Yes | Mult | F0 | F2 | F4 | | | Yes | Yes |
| | Mult2 | No | | | | | | | | |
| 1 | Add | Yes | Add | F6 | F8 | F2 | | | Yes | Yes |
| | Divide | Yes | Div | F10 | F0 | F6 | Mult1 | | No | Yes |

*Register result status:*

| Clock | | *F0* | *F2* | *F4* | *F6* | *F8* | *F10* | *F12* | ... | *F30* |
|---|---|---|---|---|---|---|---|---|---|---|
| **15** | *FU* | Mult1 | | | Add | | Divide | | | |

- **ADDD starts execution**

# Scoreboard Example: Cycle 16

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | | |
| SUBD | F8 | F6 | F2 | 7 | 9 | 11 | 12 |
| **DIVD** | **F10** | **F0** | **F6** | 8 | | | |
| ADDD | F6 | F8 | F2 | 13 | 14 | 16 | |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| 3 | Mult1 | Yes | Mult | F0 | F2 | F4 | | | Yes | Yes |
| | Mult2 | No | | | | | | | | |
| 0 | Add | Yes | Add | F6 | F8 | F2 | | | Yes | Yes |
| | Divide | Yes | Div | F10 | F0 | F6 | Mult1 | | No | Yes |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **16** | *FU* | Mult1 | | | Add | | Divide | | | |

- **ADDD ends execution, but WAR F6 must be detected before writing the result in RF**

# Scoreboard Example: Cycle 17

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | | |
| SUBD | F8 | F6 | F2 | 7 | 9 | 11 | 12 |
| **DIVD** | **F10** | **F0** | **F6** | 8 | | | |
| ADDD | F6 | F8 | F2 | 13 | 14 | 16 | |

**WAR F6 Hazard!**

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| 2 | Mult1 | Yes | Mult | F0 | F2 | F4 | | | Yes | Yes |
| | Mult2 | No | | | | | | | | |
| | Add | Yes | Add | F6 | F8 | F2 | | | Yes | Yes |
| | Divide | Yes | **Div** | **F10** | **F0** | **F6** | Mult1 | | **No** | **Yes** |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **17** | *FU* | Mult1 | | | Add | | **Divide** | | | |

- ***Why not write result of ADDD???*** **WAR F6 must be detected before writing for result of ADDD in F6**
- **DIVD must first read F6 (before ADDD write F6), but DIVD cannot read operands until MULTD writes F0 (RAW on F0)**

# Scoreboard Example: Cycle 18

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | | |
| SUBD | F8 | F6 | F2 | 7 | 9 | 11 | 12 |
| **DIVD** | **F10** | **F0** | **F6** | 8 | | | |
| ADDD | F6 | F8 | F2 | 13 | 14 | 16 | |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| 1 | Mult1 | Yes | Mult | F0 | F2 | F4 | | | Yes | Yes |
| | Mult2 | No | | | | | | | | |
| | Add | Yes | Add | F6 | F8 | F2 | | | Yes | Yes |
| | Divide | Yes | Div | F10 | F0 | F6 | Mult1 | | No | Yes |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **18** | *FU* | Mult1 | | | Add | | Divide | | | |

# Scoreboard Example: Cycle 19

*Instruction status:*

| Instruction | | *j* | *k* | *Issue* | *Read Oper* | *Exec Comp* | *Write Result* |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | 19 | |
| SUBD | F8 | F6 | F2 | 7 | 9 | 11 | 12 |
| **DIVD** | **F10** | **F0** | **F6** | 8 | | | |
| ADDD | F6 | F8 | F2 | 13 | 14 | 16 | |

*Functional unit status:*

| *Time* | *Name* | *Busy* | *Op* | *dest Fi* | *S1 Fj* | *S2 Fk* | *FU Qj* | *FU Qk* | *Fj? Rj* | *Fk? Rk* |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| 0 | Mult1 | Yes | Mult | F0 | F2 | F4 | | | Yes | Yes |
| | Mult2 | No | | | | | | | | |
| | Add | Yes | Add | F6 | F8 | F2 | | | Yes | Yes |
| | Divide | Yes | Div | F10 | F0 | F6 | Mult1 | | No | Yes |

*Register result status:*

| Clock | | *F0* | *F2* | *F4* | *F6* | *F8* | *F10* | *F12* | *…* | *F30* |
|---|---|---|---|---|---|---|---|---|---|---|
| **19** | *FU* | Mult1 | | | Add | | Divide | | | |

- **MULTD ends execution**

# Scoreboard Example: Cycle 20

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | 19 | 20 |
| SUBD | F8 | F6 | F2 | 7 | 9 | 11 | 12 |
| **DIVD** | **F10** | **F0** | **F6** | 8 | | | |
| ADDD | F6 | F8 | F2 | 13 | 14 | 16 | |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| | Mult1 | No | | | | | | | | |
| | Mult2 | No | | | | | | | | |
| | Add | Yes | Add | F6 | F8 | F2 | | | Yes | Yes |
| | Divide | Yes | **Div** | **F10** | **F0** | **F6** | | | **Yes** | **Yes** |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **20** | *FU* | | | | Add | | **Divide** | | | |

- **MULTD writes in F0**

# Scoreboard Example: Cycle 21

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | 19 | 20 |
| SUBD | F8 | F6 | F2 | 7 | 9 | 11 | 12 |
| **DIVD** | **F10** | **F0** | **F6** | 8 | 21 | | |
| ADDD | F6 | F8 | F2 | 13 | 14 | 16 | |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| | Mult1 | No | | | | | | | | |
| | Mult2 | No | | | | | | | | |
| | Add | Yes | Add | F6 | F8 | F2 | | | Yes | Yes |
| 40 | Divide | Yes | **Div** | **F10** | **F0** | **F6** | | | **Yes** | **Yes** |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **21** | *FU* | | | | Add | | **Divide** | | | |

- **DIVD can read operands**
- **WAR F6 hazard is now gone...**

# Scoreboard Example: Cycle 22

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | 19 | 20 |
| SUBD | F8 | F6 | F2 | 7 | 9 | 11 | 12 |
| **DIVD** | **F10** | **F0** | **F6** | 8 | 21 | | |
| ADDD | F6 | F8 | F2 | 13 | 14 | 16 | 22 |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| | Mult1 | No | | | | | | | | |
| | Mult2 | No | | | | | | | | |
| | Add | No | | | | | | | | |
| 39 | Divide | Yes | **Div** | **F10** | **F0** | **F6** | | | **Yes** | **Yes** |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **22** | *FU* | | | | | | **Divide** | | | |

- **DIVD has read its operands in previous cycle, so WAR F6 is gone**
- **ADDD can now write the result in F6**

*skipping some cycles…*

# Scoreboard Example: Cycle 61

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | 19 | 20 |
| SUBD | F8 | F6 | F2 | 7 | 9 | 11 | 12 |
| **DIVD** | **F10** | **F0** | **F6** | 8 | 21 | 61 | |
| ADDD | F6 | F8 | F2 | 13 | 14 | 16 | 22 |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| | Mult1 | No | | | | | | | | |
| | Mult2 | No | | | | | | | | |
| | Add | No | | | | | | | | |
| 0 | Divide | Yes | **Div** | **F10** | **F0** | **F6** | | | **Yes** | **Yes** |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **61** | *FU* | | | | | | **Divide** | | | |

- **DIVD ends execution**

# Scoreboard Example: Cycle 62

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | 19 | 20 |
| SUBD | F8 | F6 | F2 | 7 | 9 | 11 | 12 |
| DIVD | F10 | F0 | F6 | 8 | 21 | 61 | 62 |
| ADDD | F6 | F8 | F2 | 13 | 14 | 16 | 22 |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| | Mult1 | No | | | | | | | | |
| | Mult2 | No | | | | | | | | |
| | Add | No | | | | | | | | |
| | Divide | No | | | | | | | | |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **62** | *FU* | | | | | | | | | |

- **DIVD writes in F10**

# Recap: Scoreboard Example: Cycle 62

*Instruction status:*

| Instruction | | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34+ | R2 | 1 | 2 | 3 | 4 |
| LD | F2 | 45+ | R3 | 5 | 6 | 7 | 8 |
| MULTD | F0 | F2 | F4 | 6 | 9 | 19 | 20 |
| SUBD | F8 | F6 | F2 | 7 | 9 | 11 | 12 |
| **DIVD** | **F10** | **F0** | **F6** | 8 | 21 | 61 | 62 |
| ADDD | F6 | F8 | F2 | 13 | 14 | 16 | 22 |

*Functional unit status:*

| Time | Name | Busy | Op | dest Fi | S1 Fj | S2 Fk | FU Qj | FU Qk | Fj? Rj | Fk? Rk |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | No | | | | | | | | |
| | Mult1 | No | | | | | | | | |
| | Mult2 | No | | | | | | | | |
| | Add | No | | | | | | | | |
| | Divide | No | | | | | | | | |

*Register result status:*

| Clock | | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **62** | *FU* | | | | | | | | | |

- *In-order issue*
- ***Out-of-order reading operands & execute & commit***

# Reference

**Appendix A** of the text book: J. Hennessey, D. Patterson,
*"Computer Architecture: a quantitative approach"*
*4th* Edition, Morgan-Kaufmann Publishers.