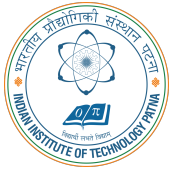
CS321: Computer Architecture

Additional topics



Arijit Mondal

Dept. of Computer Science & Engineering

Indian Institute of Technology Patna

arijit@iitp.ac.in

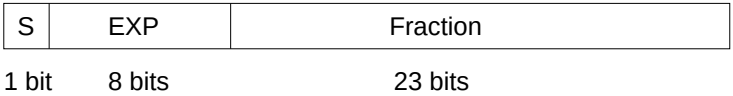
Floating point numbers

Floating point numbers: IEEE 754-1985

- Single precision numbers

Floating point numbers: IEEE 754-1985

- **Single precision numbers**



Floating point numbers: IEEE 754-1985

- **Single precision numbers**

- **Double precision numbers**

Floating point numbers: IEEE 754-1985

- **Single precision numbers**

- **Double precision numbers**

Floating point numbers: IEEE 754-1985

- **Single precision numbers**

- **Double precision numbers**

- **Number =** $(-1)^S \times (1 + F) \times 2^{EXP-Bias}$
  - **Bias for single precision numbers is 127**
  - **Bias for double precision numbers is 1023**

Single precision range

- $0000\,0000$ **and** $1111\,1111$ **are reserved.**
- **Smallest value**

Single precision range

- $0000\,0000$ **and** $1111\,1111$ **are reserved.**
- **Smallest value**
  - **Exponent: 0000 0001** $\Rightarrow 1 - 127 = -126$
  - **Fraction: 000…0** $\Rightarrow$ **significand = 1.0**
  - **Value:** $\pm 1.0 \times 2^{-126} = \pm 1.2 \times 10^{-38}$

Single precision range

- $0000\,0000$ **and** $1111\,1111$ **are reserved.**
- **Smallest value**
  - **Exponent: 0000 0001** $\Rightarrow 1 - 127 = -126$
  - **Fraction: 000...0** $\Rightarrow$ **significand = 1.0**
  - **Value:** $\pm 1.0 \times 2^{-126} = \pm 1.2 \times 10^{-38}$
- **Largest value**

Single precision range

- $0000\,0000$ **and** $1111\,1111$ **are reserved.**
- **Smallest value**
  - **Exponent: 0000 0001** $\Rightarrow 1 - 127 = -126$
  - **Fraction: 000…0 $\Rightarrow$ significand = 1.0**
  - **Value:** $\pm 1.0 \times 2^{-126} = \pm 1.2 \times 10^{-38}$
- **Largest value**
  - **Exponent: 1111 1110** $\Rightarrow 254 - 127 = 126$
  - **Fraction: 111…1 $\Rightarrow$ significand $\approx$ 2.0**
  - **Value:** $\pm 2.0 \times 2^{126} = \pm 3.4 \times 10^{38}$

Floating point example

- **Represent** -0.75

Floating point example

- **Represent** -0.75
  - $-0.75 = (-1)^1 \times (1.1) \times 2^{-1}$

Floating point example

- **Represent** -0.75
  - $-0.75 = (-1)^1 \times (1.1) \times 2^{-1}$
  - **S=1**

Floating point example

- **Represent** -0.75
  - $-0.75 = (-1)^1 \times (1.1) \times 2^{-1}$
  - **S=1**
  - **F=1000...0**

Floating point example

- **Represent** -0.75
  - $-0.75 = (-1)^1 \times (1.1) \times 2^{-1}$
  - **S=1**
  - **F=1000...0**
  - **EXP=**-1**+Bias=**$126_{10} = 01111110_2$

Floating point example

- **Represent** -0.75
  - $-0.75 = (-1)^1 \times (1.1) \times 2^{-1}$
  - **S=1**
  - **F=1000...0**
  - **EXP=**-1**+Bias=**$126_{10} = 01111110_2$
- $1\ 10000001\ 0100\ldots00 = ?$

Floating point example

- **Represent** -0.75
  - $-0.75 = (-1)^1 \times (1.1) \times 2^{-1}$
  - **S=1**
  - **F=1000...0**
  - **EXP=**-1**+Bias=**$126_{10} = 01111110_2$
- $1\ 10000001\ 0100\ldots00 = ?$
  - **S=**$(-1)^1 = -1$

Floating point example

- **Represent** -0.75
  - $-0.75 = (-1)^1 \times (1.1) \times 2^{-1}$
  - **S=1**
  - **F=1000...0**
  - **EXP=**-1**+Bias=**$126_{10} = 01111110_2$
- $1\ 10000001\ 0100\ldots00 = ?$
  - **S=**$(-1)^1 = -1$
  - **F=**$1.01_2 = 1.25_{10}$

Floating point example

- **Represent** -0.75
  - $-0.75 = (-1)^1 \times (1.1) \times 2^{-1}$
  - **S=1**
  - **F=1000...0**
  - **EXP=**-1**+Bias=**$126_{10} = 01111110_2$
- $1\ 10000001\ 0100\ldots00 = ?$
  - **S=**$(-1)^1 = -1$
  - **F=**$1.01_2 = 1.25_{10}$
  - **EXP=**$10000001_2 - 127_{10} = 129 - 127 = 2$

Floating point example

- **Represent** -0.75
  - $-0.75 = (-1)^1 \times (1.1) \times 2^{-1}$
  - **S=1**
  - **F=1000...0**
  - **EXP=**-1**+Bias=**$126_{10} = 01111110_2$
- $1\ 10000001\ 0100\ldots00 = ?$
  - **S=**$(-1)^1 = -1$
  - **F=**$1.01_2 = 1.25_{10}$
  - **EXP=**$10000001_2 - 127_{10} = 129 - 127 = 2$
  - **Number=**$-1 \times 1.25 \times 2^2 = -5$

Floating point encoding

Exponent	Fraction	Comments
0	0	0
0	Non-zero	Denormalized number
$1 - 254$	Anything	Floating point number
255	0	$\pm$ infinity
255	Non-zero	NaN

Scoreboard

Types of Data Hazards

- **Consider executing a sequence of** `Rk ← Ri op Rj`
- **Data-Dependence**
  - `R3 ← R1 op R2`
  - `R5 ← R3 op R4`
- **Anti-Dependence**
  - `R3 ← R1 op R2`
  - `R1 ← R4 op R5`
- **Out-Dependence**
  - `R3 ← R1 op R2`
  - `R3 ← R6 op R7`

Types of Data Hazards

- **Consider executing a sequence of** `Rk ← Ri op Rj`
- **Data-Dependence**
  - `R3 ← R1 op R2`
  - `R5 ← R3 op R4`

  Read-after-write (RAW) Hazard
- **Anti-Dependence**
  - `R3 ← R1 op R2`
  - `R1 ← R4 op R5`
- **Out-Dependence**
  - `R3 ← R1 op R2`
  - `R3 ← R6 op R7`

Types of Data Hazards

- **Consider executing a sequence of** `Rk ← Ri op Rj`
- **Data-Dependence**
  - `R3 ← R1 op R2`
  - `R5 ← R3 op R4`

  Read-after-write (RAW) Hazard
- **Anti-Dependence**
  - `R3 ← R1 op R2`
  - `R1 ← R4 op R5`

  Write-after-read (WAR) Hazard
- **Out-Dependence**
  - `R3 ← R1 op R2`
  - `R3 ← R6 op R7`

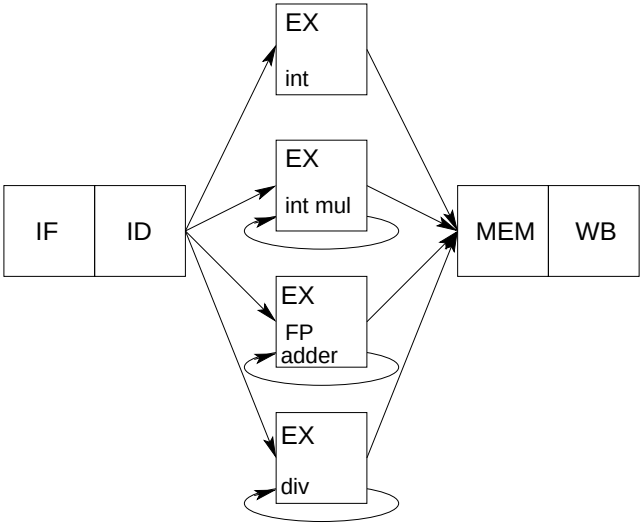
Types of Data Hazards

- **Consider executing a sequence of** `Rk ← Ri op Rj`
- **Data-Dependence**
  - `R3 ← R1 op R2` **Read-after-write (RAW) Hazard**
  - `R5 ← R3 op R4`
- **Anti-Dependence**
  - `R3 ← R1 op R2` **Write-after-read (WAR) Hazard**
  - `R1 ← R4 op R5`
- **Out-Dependence**
  - `R3 ← R1 op R2` **Write-after-write (WAW) Hazard**
  - `R3 ← R6 op R7`

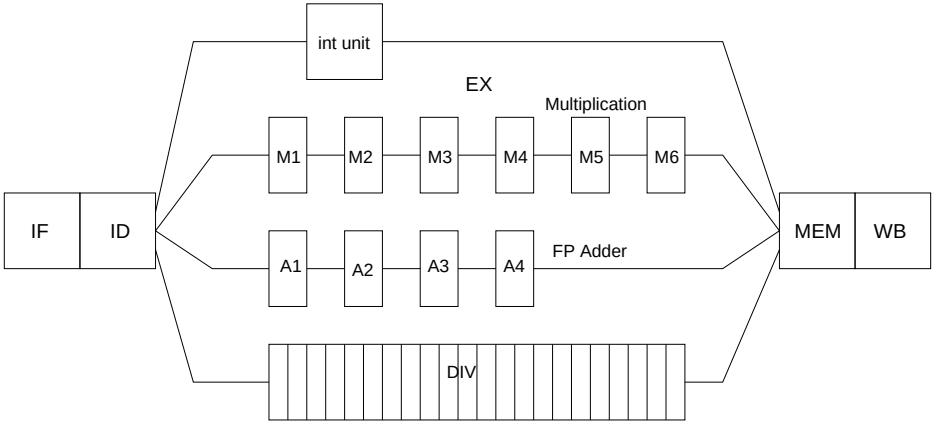
FP operations in MIPS

- Completion of floating point operation in one cycle is impractical
  - A longer CPU clock cycle will be required
  - Otherwise, huge amount of hardware will be required
- Floating point pipeline with longer latency
- Floating point operations will have the same pipeline stage. Differences are
  - EX cycle may be repeated required number of times
  - There may be more than one FP functional units
  - Stalls occurs if the instruction to be issued causes a structural or data hazards
- Latency of functional unit - Number of intervening cycles between instr producing the result and instr that uses the result
- Initiation interval - Number of cycles that must be elapsed between issuing an instruction of given type

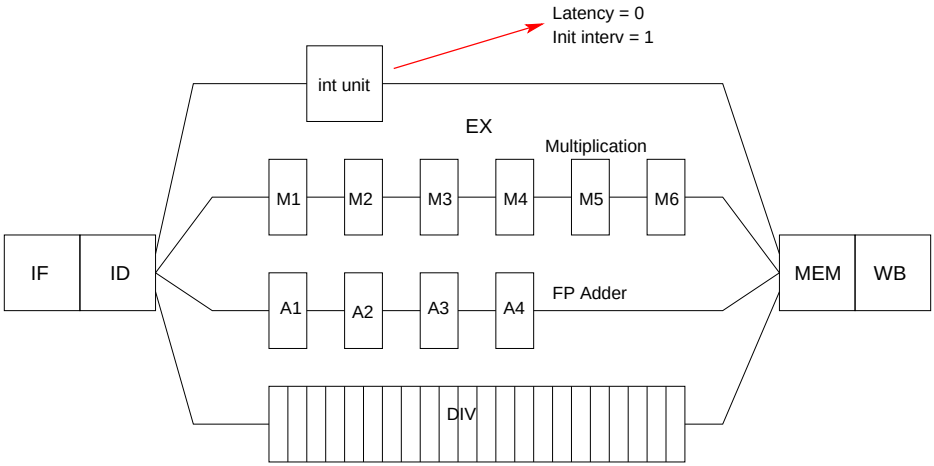
FP operations: Non-pipelined



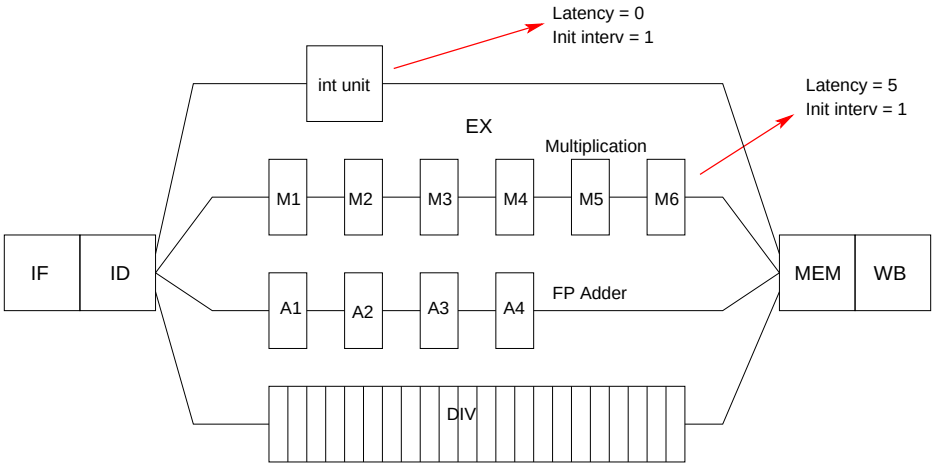
FP operations: Pipeline



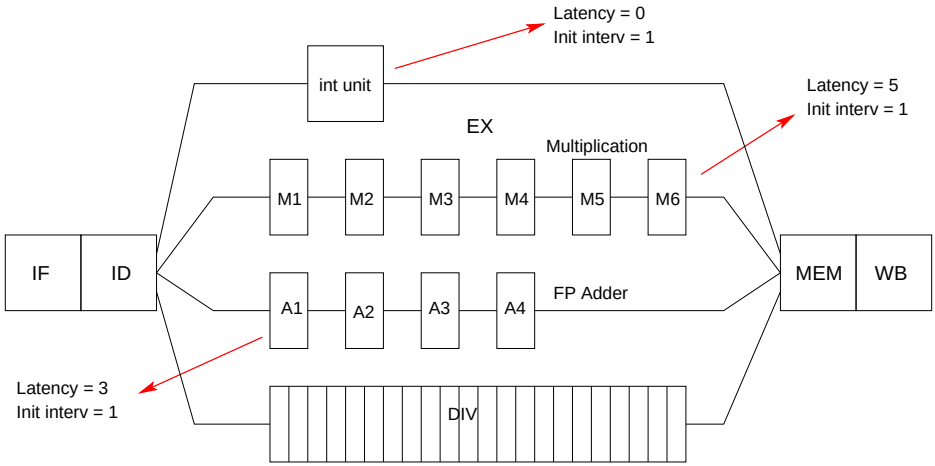
FP operations: Pipeline



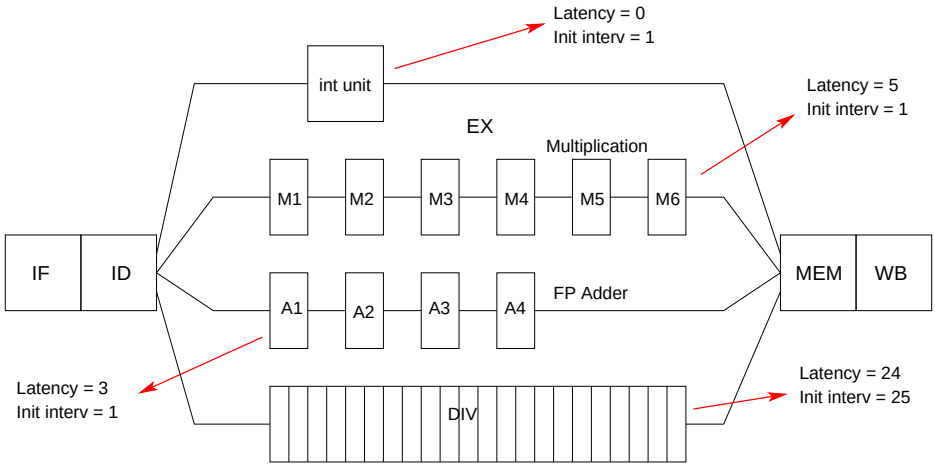
FP operations: Pipeline



FP operations: Pipeline



FP operations: Pipeline



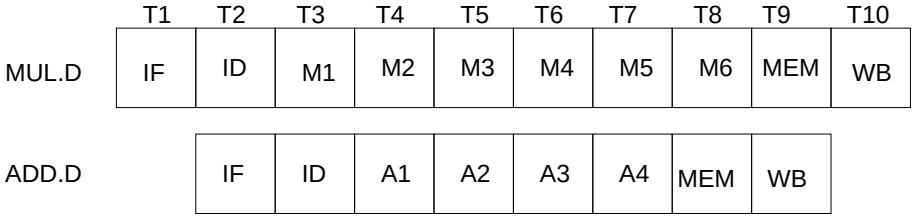
Pipeline characteristics with FP

- Instructions are processed in-order in IF, ID, EX
- Longer RAW hazard stalls likely due to long FP latencies
- Structural hazards possible
  - FP unit may not be available
  - MEM, WB reached by several instructions simultaneously
- WAR hazards is impossible
- WAW hazards can occur since it is possible for instructions to reach WB out-of-order.
- Instructions are allowed to complete out-of-order requiring special measures to handle exceptions

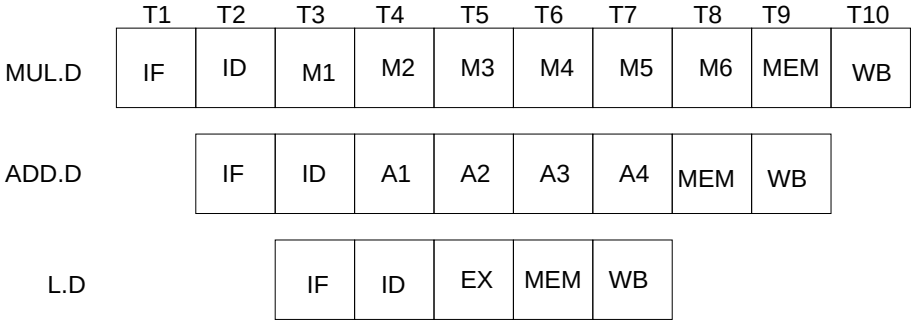
Pipeline Timing

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
MUL.D	IF	ID	M1	M2	M3	M4	M5	M6	MEM	WB

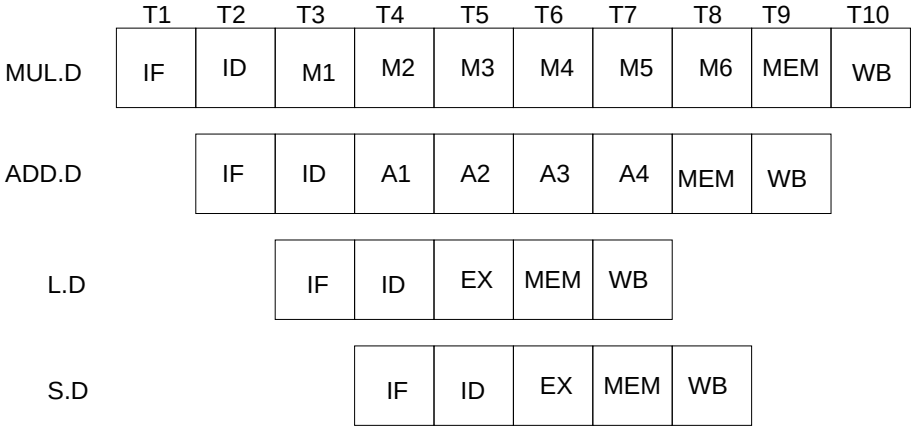
Pipeline Timing



Pipeline Timing



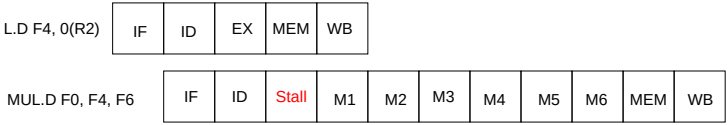
Pipeline Timing



RAW Hazard

MUL.D F0, F4, F6

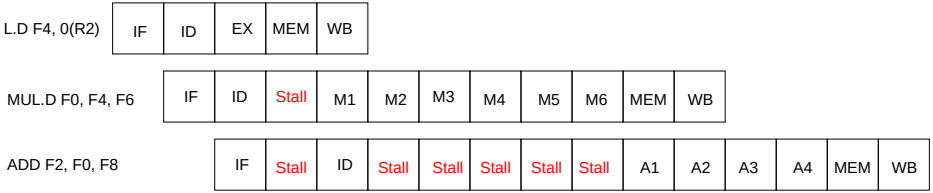
RAW Hazard


L.D F4, 0(R2)
IF
ID
EX
MEM
WB
MUL.D F0, F4, F6
IF
ID
Stall
M1
M2
M3
M4
M5
M6
MEM
WB

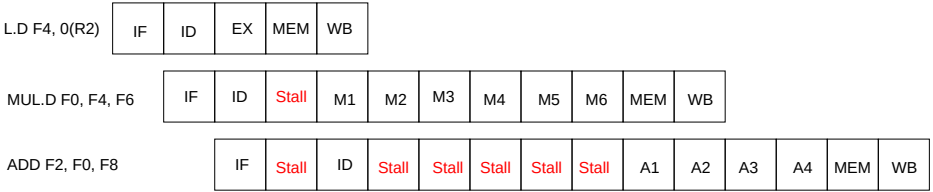
RAW Hazard


L.D F4, 0(R2)
IF
ID
EX
MEM
WB
MUL.D F0, F4, F6
IF
ID
Stall
M1
M2
M3
M4
M5
M6
MEM
WB
ADD F2, F0, F8

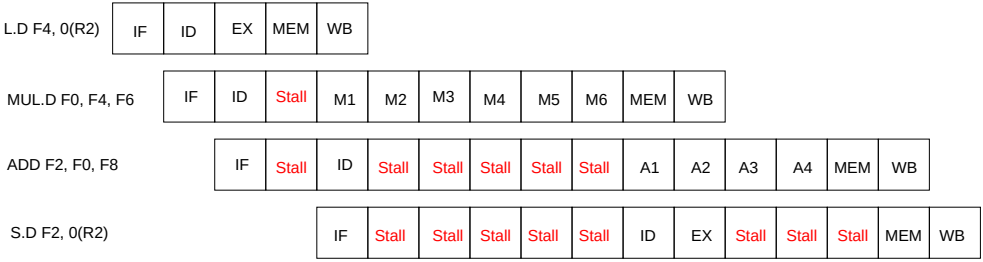
RAW Hazard


L.D F4, 0(R2)
IF
ID
EX
MEM
WB
MUL.D F0, F4, F6
IF
ID
Stall
M1
M2
M3
M4
M5
M6
MEM
WB
ADD F2, F0, F8
IF
Stall
ID
Stall
Stall
Stall
Stall
Stall
A1
A2
A3
A4
MEM
WB

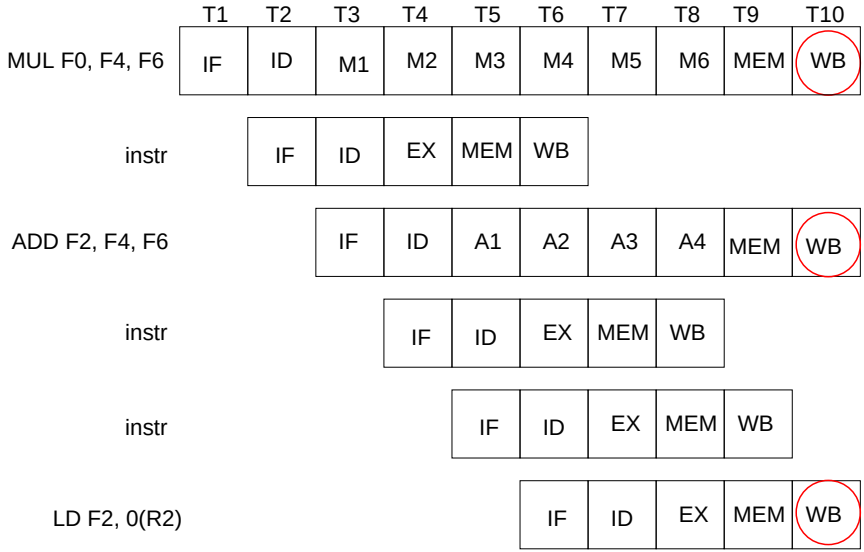
RAW Hazard


L.D F4, 0(R2)
IF ID EX MEM WB
MUL.D F0, F4, F6
IF ID Stall M1 M2 M3 M4 M5 M6 MEM WB
ADD F2, F0, F8
IF Stall ID Stall Stall Stall Stall Stall A1 A2 A3 A4 MEM WB
S.D F2, 0(R2)

RAW Hazard


L.D F4, 0(R2)
IF ID EX MEM WB
MUL.D F0, F4, F6
IF ID Stall M1 M2 M3 M4 M5 M6 MEM WB
ADD F2, F0, F8
IF Stall ID Stall Stall Stall Stall Stall A1 A2 A3 A4 MEM WB
S.D F2, 0(R2)
IF Stall Stall Stall Stall Stall ID EX Stall Stall Stall MEM WB

Structural Hazard



Exceptions in multicycle pipeline

```
DIV.D F0, F2, F4
ADD.D F10, F10, F8
SUB.D F12, F12, F14
```

- The ADD.D, SUB.D instructions can complete before DIV.D is completed causing out-of-order execution completion
- If SUB.D causes a floating-point arithmetic exception it may prevent DIV.D from completing and draining the floating-point may not be possible causing an imprecise exception.
- Handling such situation
  - Ignore the problem and settle for imprecise exception.
  - Buffer the results of the operation until all the operations issues earlier are done.
  - A history file keeps track of the original values of registers
  - A Future file keeps the newer value of a register; when all earlier instructions have completed the main register file is updated from the future file. On an exception the main register file has the precise values for the interrupted state.

Dynamic scheduling

- Data hazards in pipeline are handled by
  - Result forwarding and bypassing to reduce latency and hide or reduce the effect of true data dependence
  - Hazard detection hardware to stall the pipeline starting with the instruction that uses the result.
  - Compiler-based static pipeline scheduling to separate the dependent instructions minimizing actual hazards and stalls in scheduled code.
- Dynamic scheduling
  - Uses a hardware-based mechanism to rearrange instruction execution order to reduce stalls at runtime.
  - Enables handling some cases where dependencies are unknown at compile time.
  - Similar to the other pipeline optimizations above, a dynamically scheduled processor cannot remove true data dependencies, but tries to avoid or reduce stalling.

Idea of dynamic pipeline scheduling

- Dynamic pipeline scheduling overcomes the limitations of in-order execution by allowing out-of-order instruction execution
- Instruction are allowed to start executing out-of-order as soon as their operands are available
- Example

  DIVD F0, F2, F4
  ADDD F10, F0, F8
  SUBD F12, F8, F14

- This implies allowing out-of-order instruction commit (completion)
- May lead to imprecise exceptions if an instruction issued earlier raises an exception. (Similar to pipelines with multi-cycle floating point units)

Dynamic Pipeline Scheduling

- Dividing the Instruction Decode ID stage into two stages:
  - Issue: Decode instructions, check for structural hazards
  - Read operands: Wait until data hazard conditions, if any, are resolved, then read operands when available
  - All instructions pass through the issue stage in order but can be stalled or pass each other in the read operands stage
- In the instruction fetch stage IF, fetch an additional instruction every cycle into a latch or several instructions into an instruction queue
- Increase the number of functional units to meet the demands of the additional instructions in their EX stage

Dynamic Scheduling with Scoreboard

- The score board is a hardware mechanism that maintains an execution rate of one instruction per cycle by executing an instruction as soon as its operands are available and no hazard conditions prevent it
- It replaces ID, EX, WB with four stages: ID1, ID2, EX, WB
- Every instruction goes through the scoreboard where a record of data dependencies is constructed (corresponds to instruction issue)
- A system with a scoreboard is assumed to have several functional units with their status information reported to the scoreboard
- If the scoreboard determines that an instruction cannot execute immediately it executes another waiting instruction and keeps monitoring hardware units status and decide when the instruction can proceed to execute
- The scoreboard also decides when an instruction can write its results to registers (hazard detection and resolution is centralized in the scoreboard)

Stages in Scoreboard

- **Issue (ID1): If a functional unit for the instruction is available, the scoreboard issues the instruction to the functional unit and updates its internal data structure; structural and WAW hazards are resolved here. (this replaces part of ID stage in the conventional MIPS pipeline).**
- **Read operands (ID2): The scoreboard monitors the availability of the source operands. A source operand is available when no earlier active instruction will write it. When all source operands are available the scoreboard tells the functional unit to read all operands from the registers (no forwarding supported) and start execution (RAW hazards resolved here dynamically). This completes ID.**
- **Execution (EX): The functional unit starts execution upon receiving operands. When the results are ready it notifies the scoreboard (replaces EX, MEM in MIPS).**
- **Write result (WB): Once the scoreboard senses that a functional unit completed execution, it checks for WAR hazards and stalls the completing instruction if needed otherwise the write back is completed.**

Scoreboard

- Instruction status: Which of 4 steps the instruction is in
- Functional unit status: Indicates the state of the functional unit (FU). Nine fields for each functional unit:
  - Busy – Indicates whether the unit is busy or not
  - Op - Operation to perform in the unit (e.g., + or —)
  - 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
- Register result status: Indicates which functional unit will write to each register, if one exists. Blank when no pending instructions will write that register.

Scoreboard Example

Functional unit	# FU	EX latency
Integer	1	0
FP Multiply	2	10
FP Add	1	2
FP Division	1	40

```
L.D   F6, 34(R2)
L.D   F2, 45(R3)
MUL.D F0, F2, F4
SUB.D F8, F6, F2
DIV.D F10, F0, F6
ADD.D F6, F8, F2
```

Scoreboard example

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2				
L.D F2	45	R3				
MUL F0	F2	F4				
SUB F8	F6	F2				
DIV F10	F0	F6				
ADD F6	F8	F2				

Register result status

Clock	F0	F2	F4	F6	F8	F10
FU						

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer									
	Mult1									
	Mult2									
	Add									
	Divide									

Scoreboard example: Cycle 1

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1			
L.D F2	45	R3				
MUL F0	F2	F4				
SUB F8	F6	F2				
DIV F10	F0	F6				
ADD F6	F8	F2				

Register result status

Clock		F0	F2	F4	F6	F8	F10
1	FU				int		

Issue L.D #1

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	LD	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Scoreboard example: Cycle 2

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2		
L.D F2	45	R3				
MUL F0	F2	F4				
SUB F8	F6	F2				
DIV F10	F0	F6				
ADD F6	F8	F2				

Register result status

Clock		F0	F2	F4	F6	F8	F10
2	FU				int		

Cannot issue L.D #2
Integer unit is busy

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	LD	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Scoreboard example: Cycle 3

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	
L.D F2	45	R3				
MUL F0	F2	F4				
SUB F8	F6	F2				
DIV F10	F0	F6				
ADD F6	F8	F2				

Register result status

Clock		F0	F2	F4	F6	F8	F10
3	FU				int		

Cannot issue L.D #2
Integer unit is busy

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	LD	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Scoreboard example: Cycle 4

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3				
MUL F0	F2	F4				
SUB F8	F6	F2				
DIV F10	F0	F6				
ADD F6	F8	F2				

Register result status

Clock		F0	F2	F4	F6	F8	F10
4	FU				int		

Cannot issue L.D #2
Integer unit is busy

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	LD	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Scoreboard example: Cycle 5

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5			
MUL F0	F2	F4				
SUB F8	F6	F2				
DIV F10	F0	F6				
ADD F6	F8	F2				

Register result status

Clock: 5

	F0	F2	F4	F6	F8	F10
FU		int				

Issue L.D #2
Integer unit is free

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	LD	F2		R3				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Scoreboard example: Cycle 6

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6		
MUL F0	F2	F4	6			
SUB F8	F6	F2				
DIV F10	F0	F6				
ADD F6	F8	F2				

Register result status

Clock 6

	F0	F2	F4	F6	F8	F10
FU	Mult1	int				

Issue MUL

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	LD	F2		R3				Yes
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
	Mult2	No								
	Add	No								
	Divide	No								

Scoreboard example: Cycle 7

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	
MUL F0	F2	F4	6			
SUB F8	F6	F2	7			
DIV F10	F0	F6				
ADD F6	F8	F2				

Register result status

Clock 7

	F0	F2	F4	F6	F8	F10
FU	Mult1	int			Add	

MULT cannot read its operand F2 as LD #2 has not finished

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	LD	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								

Scoreboard example: Cycle 8a

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	
MUL F0	F2	F4	6			
SUB F8	F6	F2	7			
DIV F10	F0	F6	8			
ADD F6	F8	F2				

Register result status

Clock 8

	F0	F2	F4	F6	F8	F10
FU	Mult1	int			Add	Divide

Issue DIV. MUL and SUB wait for F2

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	LD	F2		R3				Yes
	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

Scoreboard example: Cycle 8b

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	8
MUL F0	F2	F4	6			
SUB F8	F6	F2	7			
DIV F10	F0	F6	8			
ADD F6	F8	F2				

Register result status

Clock 8

FU	F0	F2	F4	F6	F8	F10
	Mult1				Add	Divide

LD #2 writes F2

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	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

Scoreboard example: Cycle 9

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9		
DIV F10	F0	F6	8			
ADD F6	F8	F2				

Register result status

Clock		F0	F2	F4	F6	F8	F10
9	FU	Mult1				Add	Divide

MUL and SUB both read F2

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	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

Scoreboard example: Cycle 11

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9	11	
DIV F10	F0	F6	8			
ADD F6	F8	F2				

Register result status

Clock		F0	F2	F4	F6	F8	F10
11	FU	Mult1				Add	Divide

ADD cannot start as add unit is busy

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	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

Scoreboard example: Cycle 12

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9	11	12
DIV F10	F0	F6	8			
ADD F6	F8	F2				

Register result status

Clock = 12

FU	F0	F2	F4	F6	F8	F10
	Mult1					Divide

SUB finishes
DIV waiting for F0

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	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

Scoreboard example: Cycle 13

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9	11	12
DIV F10	F0	F6	8			
ADD F6	F8	F2	13			

Register result status

Clock 13

	F0	F2	F4	F6	F8	F10
FU	Mult1			Add		Divide

Issue ADD

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	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

Scoreboard example: Cycle 14

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9	11	12
DIV F10	F0	F6	8			
ADD F6	F8	F2	13	14		

Register result status

Clock		F0	F2	F4	F6	F8	F10
14	FU	Mult1			Add		Divide

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	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

Scoreboard example: Cycle 15

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9	11	12
DIV F10	F0	F6	8			
ADD F6	F8	F2	13	14		

Register result status

Clock = 15

	F0	F2	F4	F6	F8	F10
FU	Mult1			Add		Divide

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	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

Scoreboard example: Cycle 16

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9	11	12
DIV F10	F0	F6	8			
ADD F6	F8	F2	13	14	16	

Register result status

Clock		F0	F2	F4	F6	F8	F10
16	FU	Mult1			Add		Divide

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	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

Scoreboard example: Cycle 17

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9	11	12
DIV F10	F0	F6	8			
ADD F6	F8	F2	13	14	16	

Register result status

Clock		F0	F2	F4	F6	F8	F10
17	FU	Mult1			Add		Divide

ADD cannot write because of DIV. RAW

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	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

Scoreboard example: Cycle 18

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9	11	12
DIV F10	F0	F6	8			
ADD F6	F8	F2	13	14	16	

Register result status

Clock		F0	F2	F4	F6	F8	F10
18	FU	Mult1			Add		Divide

Nothing happens

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	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

Scoreboard example: Cycle 19

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	8
MUL F0	F2	F4	6	9	19	
SUB F8	F6	F2	7	9	11	12
DIV F10	F0	F6	8			
ADD F6	F8	F2	13	14	16	

Register result status

Clock		F0	F2	F4	F6	F8	F10
19	FU	Mult1			Add		Divide

MUL completes execution

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	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

Scoreboard example: Cycle 20

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	8
MUL F0	F2	F4	6	9	19	20
SUB F8	F6	F2	7	9	11	12
DIV F10	F0	F6	8			
ADD F6	F8	F2	13	14	16	

Register result status

Clock		F0	F2	F4	F6	F8	F10
20	FU				Add		Divide

MUL writes

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		Yes	Yes

Scoreboard example: Cycle 21

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	8
MUL F0	F2	F4	6	9	19	20
SUB F8	F6	F2	7	9	11	12
DIV F10	F0	F6	8	21		
ADD F6	F8	F2	13	14	16	

Register result status

Clock		F0	F2	F4	F6	F8	F10
21	FU				Add		Divide

DIV loads operands

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		Yes	Yes

Scoreboard example: Cycle 22

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	8
MUL F0	F2	F4	6	9	19	20
SUB F8	F6	F2	7	9	11	12
DIV F10	F0	F6	8	21		
ADD F6	F8	F2	13	14	16	22

Register result status

Clock		F0	F2	F4	F6	F8	F10
22	FU						Divide

ADD can write now

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
40	Divide	Yes	Div	F10	F0	F6	Mult1		Yes	Yes

Scoreboard example: Cycle 61

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	8
MUL F0	F2	F4	6	9	19	20
SUB F8	F6	F2	7	9	11	12
DIV F10	F0	F6	8	21	61	
ADD F6	F8	F2	13	14	16	22

Register result status

Clock		F0	F2	F4	F6	F8	F10
61	FU						Divide

DIV completes

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
0	Divide	Yes	Div	F10	F0	F6	Mult1		Yes	Yes

Scoreboard example: Cycle 62

Instruction status

Instruction	j	k	Issue	RD	EC	WR
L.D F6	34	R2	1	2	3	4
L.D F2	45	R3	5	6	7	8
MUL F0	F2	F4	6	9	19	20
SUB F8	F6	F2	7	9	11	12
DIV F10	F0	F6	8	21	61	62
ADD F6	F8	F2	13	14	16	22

Register result status

Clock 62 FU

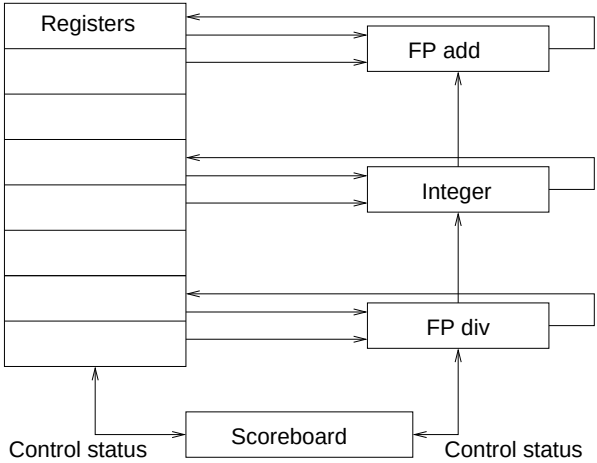
F0	F2	F4	F6	F8	F10

DONE !!

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Scoreboard


Registers
FP add
Integer
FP div
Scoreboard
Control status
Control status