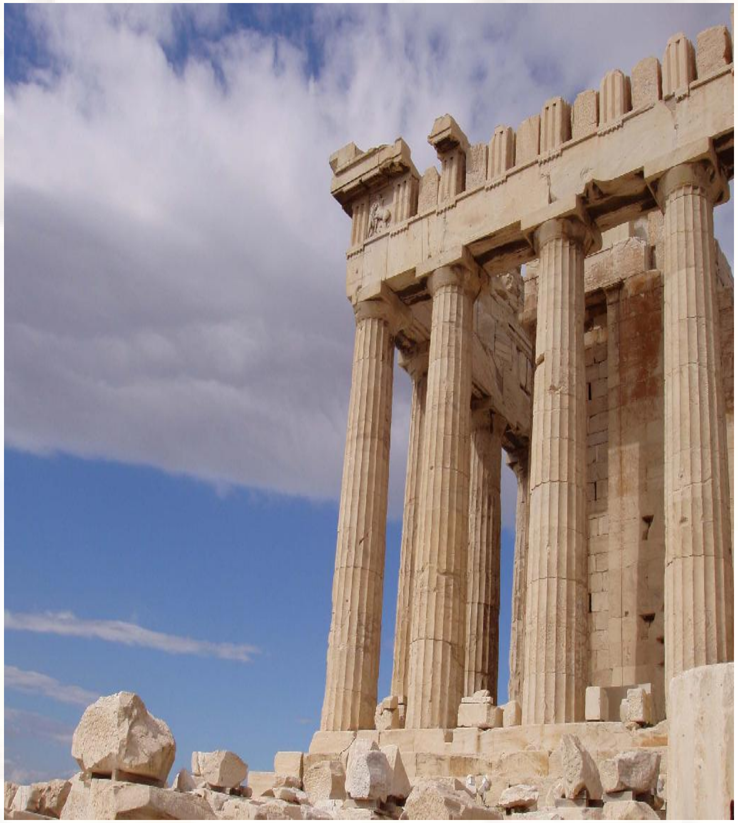
Ch3-- ILP & its exploration



Ch3-0 Extending 5-stage pipeline to support multicycle operations

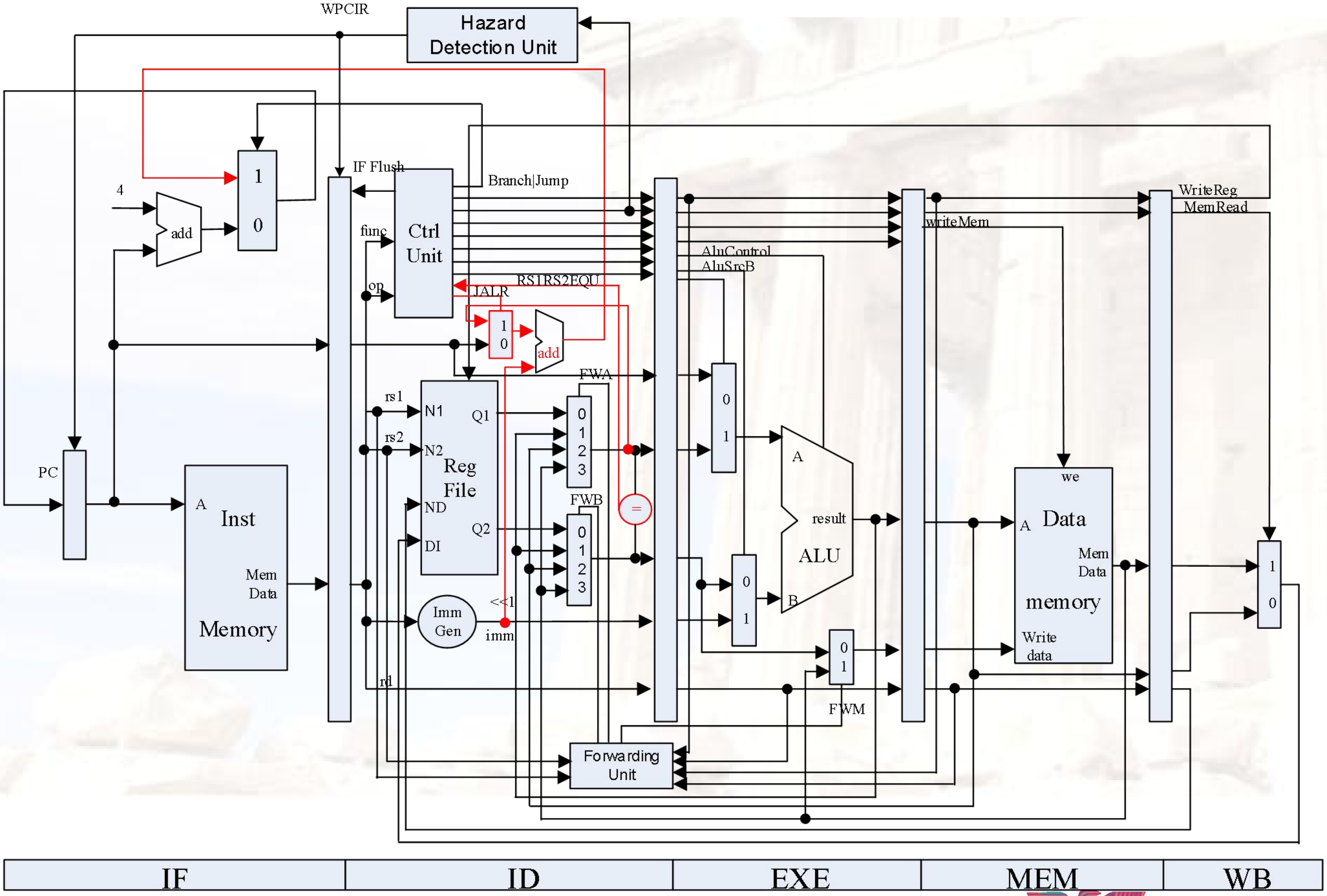
App C5-C6

2024-10-14

Review of Pipeline Hazards in CO

- **Structural hazards**
  - These are conflicts over hardware resources.
  - add extra hardware resources; full pipelined the functional units; otherwise still have to stall
  - Allow machine with Structural hazard, since it happens not so often
- **Data hazards**
  - Instruction depends on result of prior computation which is not ready (computed or stored) yet
  - Stall; double bump; forwarding path; compiler scheduling
- **Control hazards**
  - branch condition and the branch PC are not available in time to fetch an instruction on the next clock
  - Flushing; predict taken, predict-not-taken, delayed branch
  - Moving target address calculation and condition comparison forward as earlier as possible.

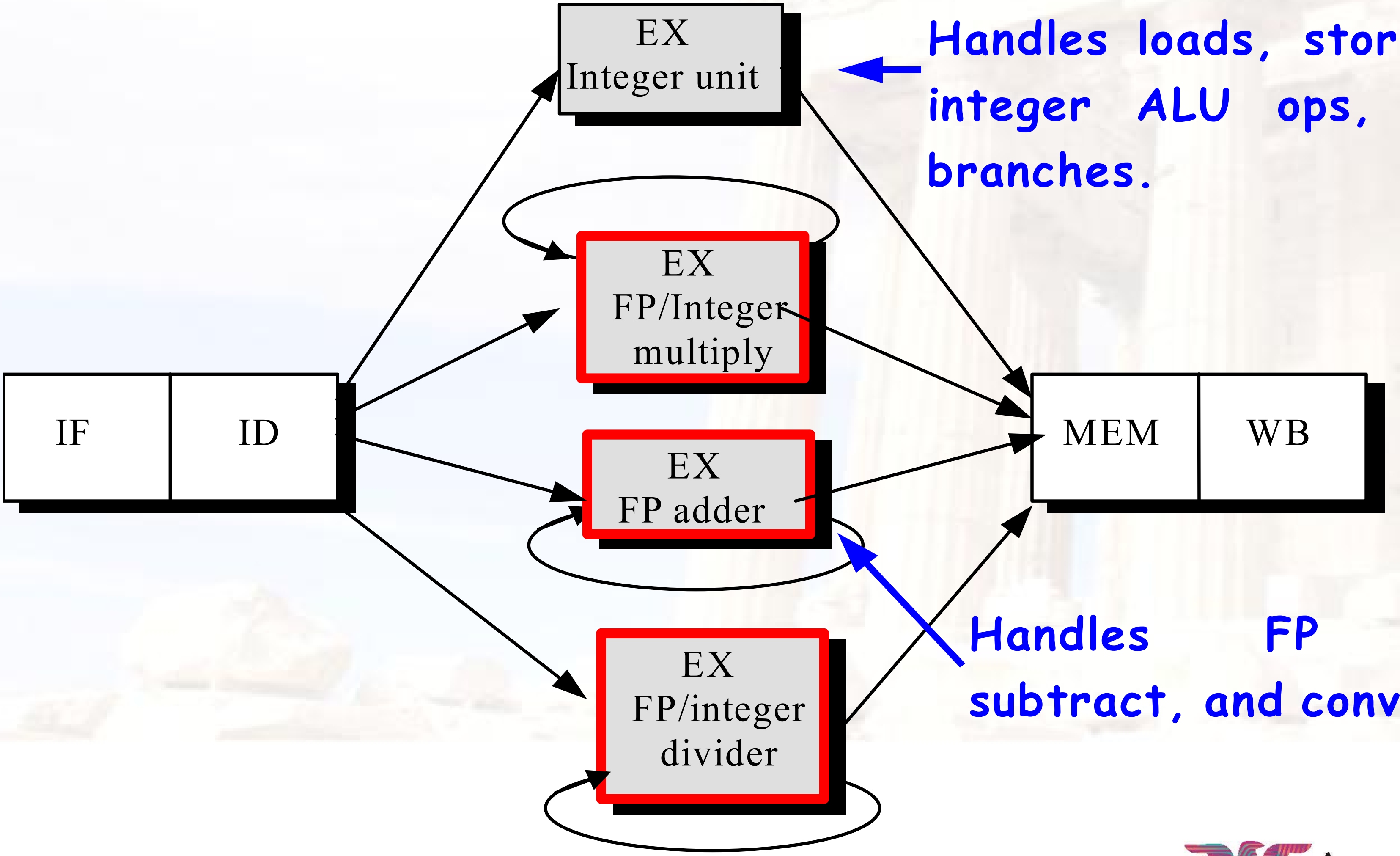
Pipelined CPU supporting RISC V



Extending the MIPS pipeline to handle MultiCycle Operations

- ❑ Alternative resolutions to handle floating-point operations (or complex operations)
  - ➢ Complete operation in 1 or 2 clock cycles X
    - ○ Which means using a slow clock,
    - ○ or/and using enormous amounts of logic in FP units.
  - ➢ Allow for a longer latency for operations
    - ○ The EX cycle may be repeated as many times as needed to complete the operation
    - ○ There may be multiple FP units

5-stage pipeline with FP units



Handles loads, stores , integer ALU ops, and branches.

Handles FP add, subtract, and conversion

Pipelining some of the FP units

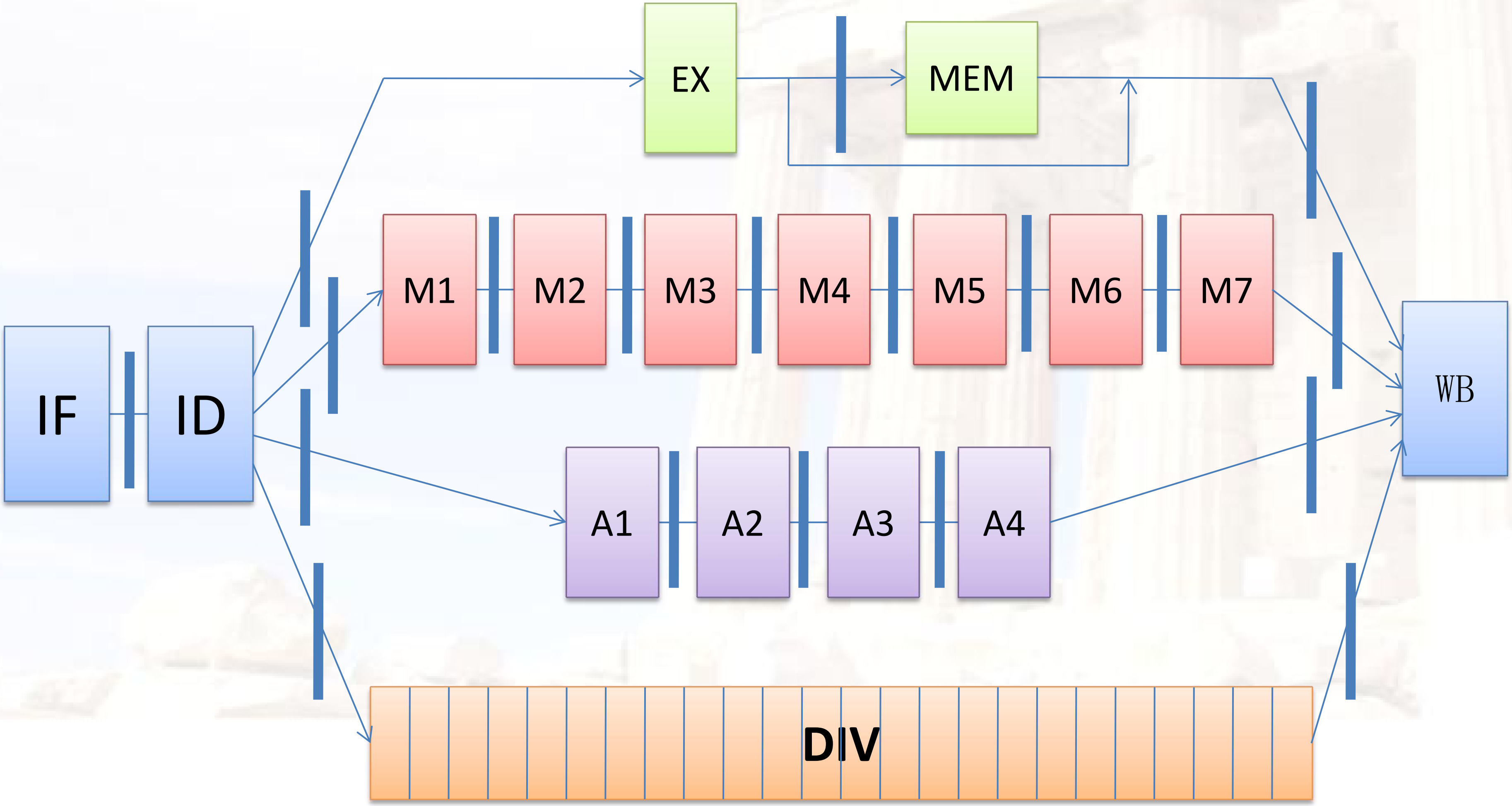
- ❑ Two terminologies
  - Latency----the number of intervening cycles between an instruction that produces a result and an instruction that uses the result.
  - Initiation interval----the number of cycles that must elapse between instructions issue to the same unit.
    - ○ For full pipelined units, initiation interval is 1
    - ○ For unpipelined units, initiation interval is always the latency plus 1.

Latencies and initiation intervals for functional units

Functional unit	Latency	Initiation interval
Integer ALU	0	1
Data memory(integer and FP loads)	1	1
FP add	3	1
FP multiply (also integer multiply)	6	1
FP divide (also integer divide)	24	25

Note: latency = function unit time -1 clock cycle

Pipeline supports multiple outstanding FP operations



Specifications

- Memory bandwidth: **double words/one cycle**
- New pipeline latches are required:
  - M1/M2, M2/M3, M3/M4, M4/M5, M5/M6, M6/M7
  - A1/A2, A2/A3, A3/A4
- New connection registers are required:
  - ID/EX, ID/M1, ID/A1, ID/DIV
  - EX/MEM, M7/MEM, A4/MEM, DIV/MEM
- Because the divider unit is unpipelined, **structural hazards** can occur.
- Because the instructions have varying running times, the number of register writes required in a cycle can be larger than 1
- **New data hazards: WAW** is possible due to disorder WBs
- Due to longer latency of operations, **stalls for RAW hazards will be more frequent.**
- Problems with **exceptions** resulting from disorder completion

Issuing in order and completion out of order

Instruction	1	2	3	4	5	6	7	8	9	10	11	12
MUL.D	IF	ID	M1	M2	M3	M4	M5	M6	M7	WB		
ADD.D		IF	ID	A1	A2	A3	A4	WB				
MUL.D			IF	ID	M1	M2	M3	M4	M5	M6	M7	WB
LD.D				IF	ID	EX	MEM	WB				
SD.D					IF	ID	EX	MEM	WB			

1、multiple writes on one clock cycyle

2、out of order writes : different /same Destination Register

Structural Hazards for the FP register write port

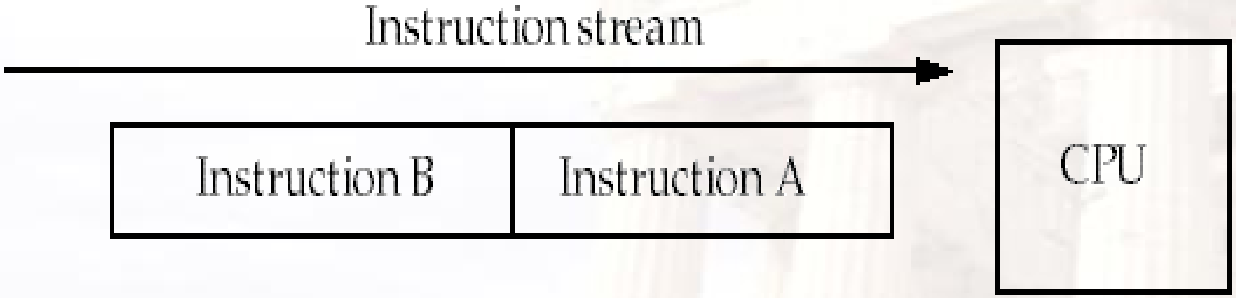
Instruction	1	2	3	4	5	6	7	8	9	10
MUL.D F0,F4, F6	IF	ID	M1	M2	M3	M4	M5	M6	M7	WB
……		IF	ID	EX	MEM	WB				
……			IF	ID	EX	MEM	WB			
ADD.D F2, F4, F6				IF	ID	A1	A2	A3	A4	WB
……					IF	ID	EX	MEM	WB	
LD.D F8, 0(R2)						IF	ID	EX	MEM	WB

How to solve the write port conflict ?

- Increase the number of write portsX
  - **Unattractive** at all !
  - No worthy since steady state usage is close to 1.
- Detect and insert stalls by serializing the writes
  - **Track the use of the write port in the ID stage and to stall an instruction before it issues**
    - Additional Hardware: **a shift register+ write conflict logic**
    - The shift register tracks when already-issued instructions will use the register file, and right shift 1 bit each clock.
    - The stalls might ***aggravate*** the data hazards
    - All interlock detection and stall insertion occurs in ID stage
  - **To stall a conflicting instruction when it tries to enter the MEM or WB stage.**
    - Easy to detect the conflict at this point
    - Complicates pipeline control since stalls can now occur in two places.

Types of data hazards

- Consider two instructions, A and B. A occurs before B.



- **RAW(Read after write) true dependence**
  - Instruction A writes Rx，instruction B reads Rx

ADD x5, x8,x10
MUL x6, x5, x9

- **WAW(Write after write) output dependence**
  - Instruction A writes Rx，instruction B writes Rx

ADD x5, x8,x10
MUL x5, x6, x9

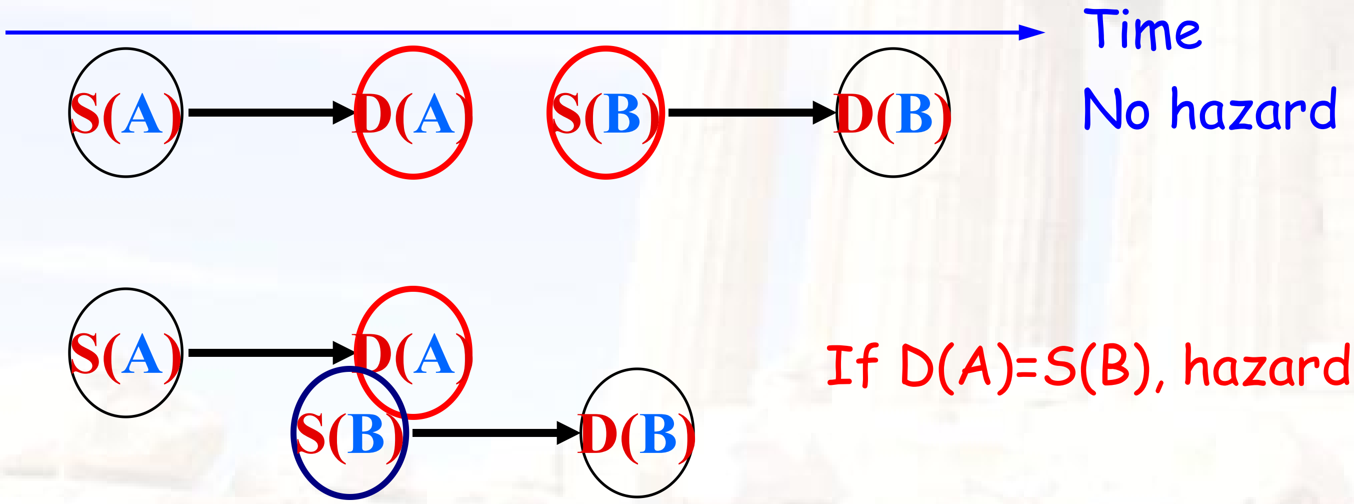
- **WAR(Write after read) anti-denpendence**
  - Instruction A reads Rx，instruction B writes Rx

ADD x5, x8,x9
MUL x8, x6, x10

- Hazards are named according to the ordering **that** **MUST** **be preserved by the pipeline**

RAW dependence:Ture data dependence

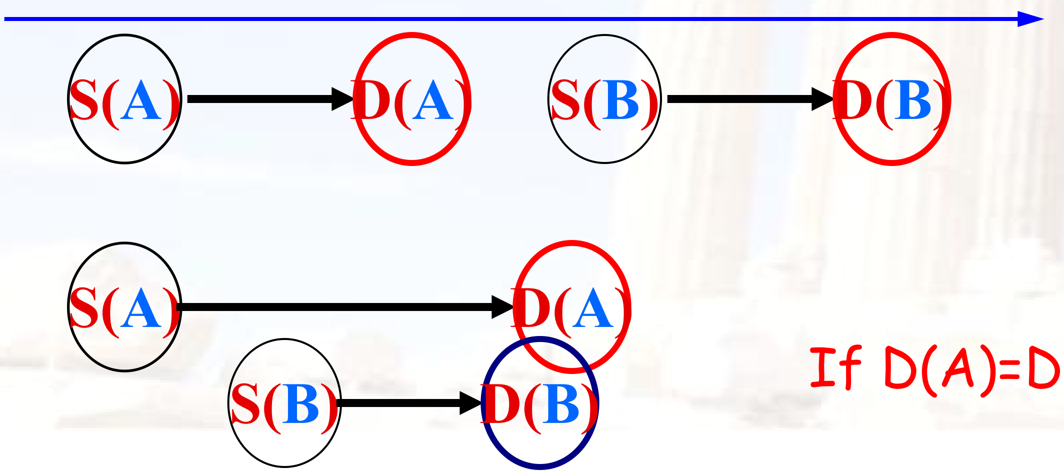
- B tries to read a register before A has written it and gets the old value.
- This is common, and forwarding helps to solve it.



If D(A)=S(B), hazard occur.

WAW dependence- naming dependence1

- ❑B tries to write an operand before A has written it.
- ❑After instruction B has executed, the value of the register should be B's result, but A's result is stored instead.
- ❑This can only happen with pipelines that write values in more than one stage, or in variable-length pipelines (i.e. FP pipelines).



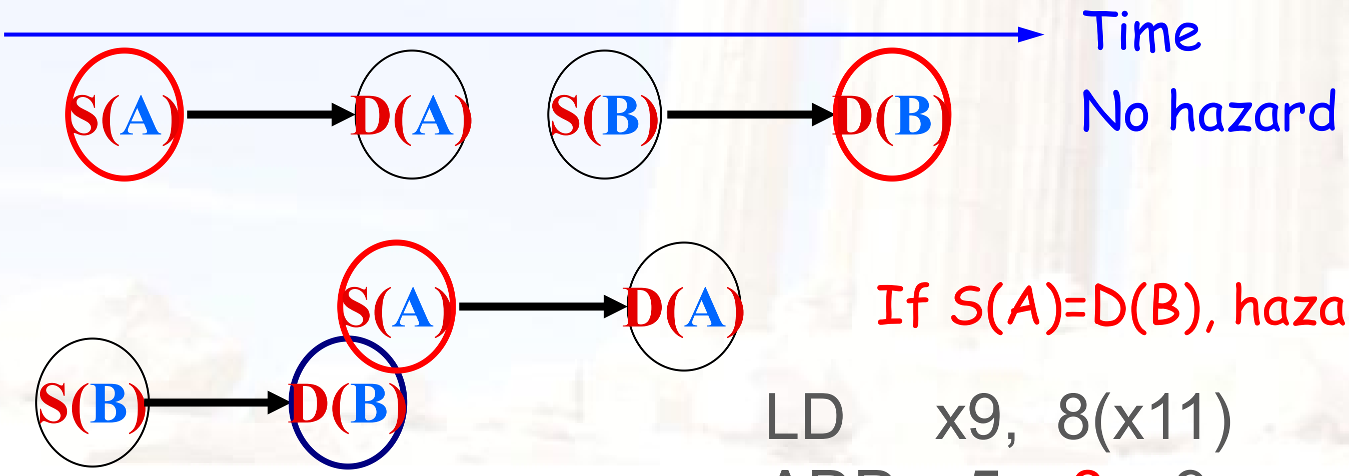
Time

No hazard

If D(A)=D(B), hazard occur.

WAR dependence –naming dependence2

- B tries to write a register before A has read it.
- In this case, A uses the new (incorrect) value.
- This type of hazard is rare because most pipelines read values early and write results late.
- However, it might happen for a CPU that had complex addressing modes. i.e. autoincrement.



If S(A)=D(B), hazard occur.

```
LD   x9,  8(x11)
ADD  x5, x8, x9
MUL  x8, x6, x10
```

Stalls arising from RAW hazards

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LD.D **F4**, 0(R2)	IF	ID	EX	MEM	WB											
MUL.D F0, **F4**, F6		IF	ID	stall	M1	M2	M3	M4	M5	M6	M7	WB				
ADD.D **F2**, F0, F8			IF	stall	ID	stall	stall	stall	stall	stall	stall	A1	A2	A3	A4	WB
SD.D **F2**, 0(R2)					IF	stall	stall	stall	stall	stall	stall	ID	EX	stall	stall	MEM

The WAW hazards

Issure in order

Structural Hazard
Register Write Port

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	16
MUL.D **F0**, F4, F6	IF	ID	M1	M2	M3	M4	M5	M6	M7	WB			
……		IF	ID	EX	MEM	WB							
……			IF	ID	EX	MEM	WB						
ADD.D **F2**, F4, F6				IF	ID	stall	A1	A2	A3	A4	WB		
LD.D F2, 0(R2)					IF	stall	ID	stall	stall	EX	DM	WB	
……							IF	stall	stall	ID	EX	DM	WB
LD.D **F8**, 0(R2)										IF	ID	EX	DM

WAW hazard

* Assume 2 instructions after ADD has no write register.

Solving the WAW hazard

- **Stall an instruction** that would "pass" another until after the earlier instruction reaches the MEM phase.
- **Cancel the WB phase of the earlier instruction**
- Both of these can be done in ID, i.e. when LD is about to issue.
- Since pure WAW hazards are not common, either method works.
- Pick the one that simplest to implement.
- The simplest solution for the RISC V pipeline is to hold the instruction in ID if it writes the same register as an instruction already issued.

What other hazards are possible ?

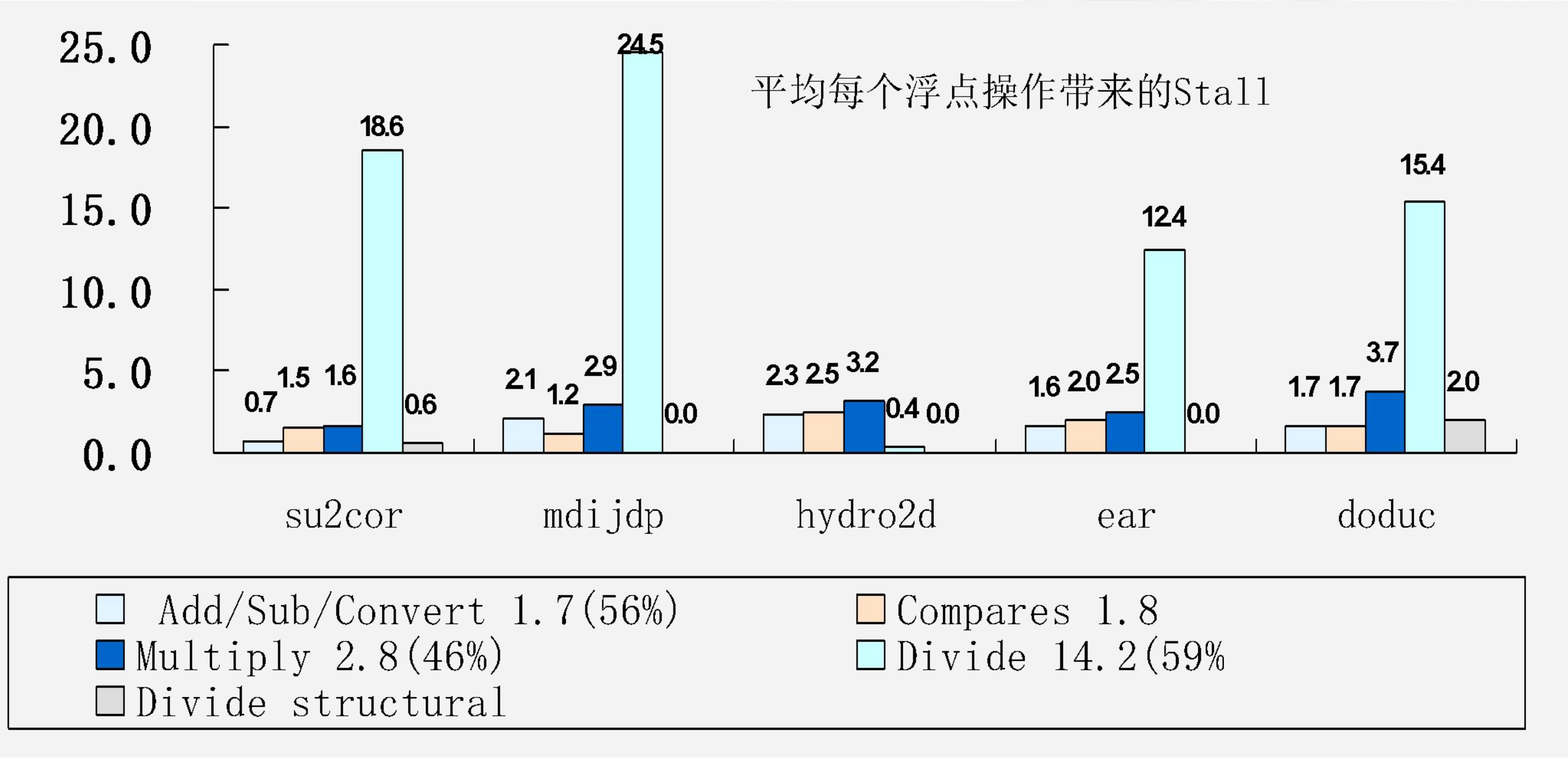
- ❑Hazards among FP instructions.
- ❑Hazards between an FP instruction and an integer instruction.
  - ➢Since two register files exist, only FP loads and stores and FP register moves to integer registers involve hazards.

Checks are required in ID

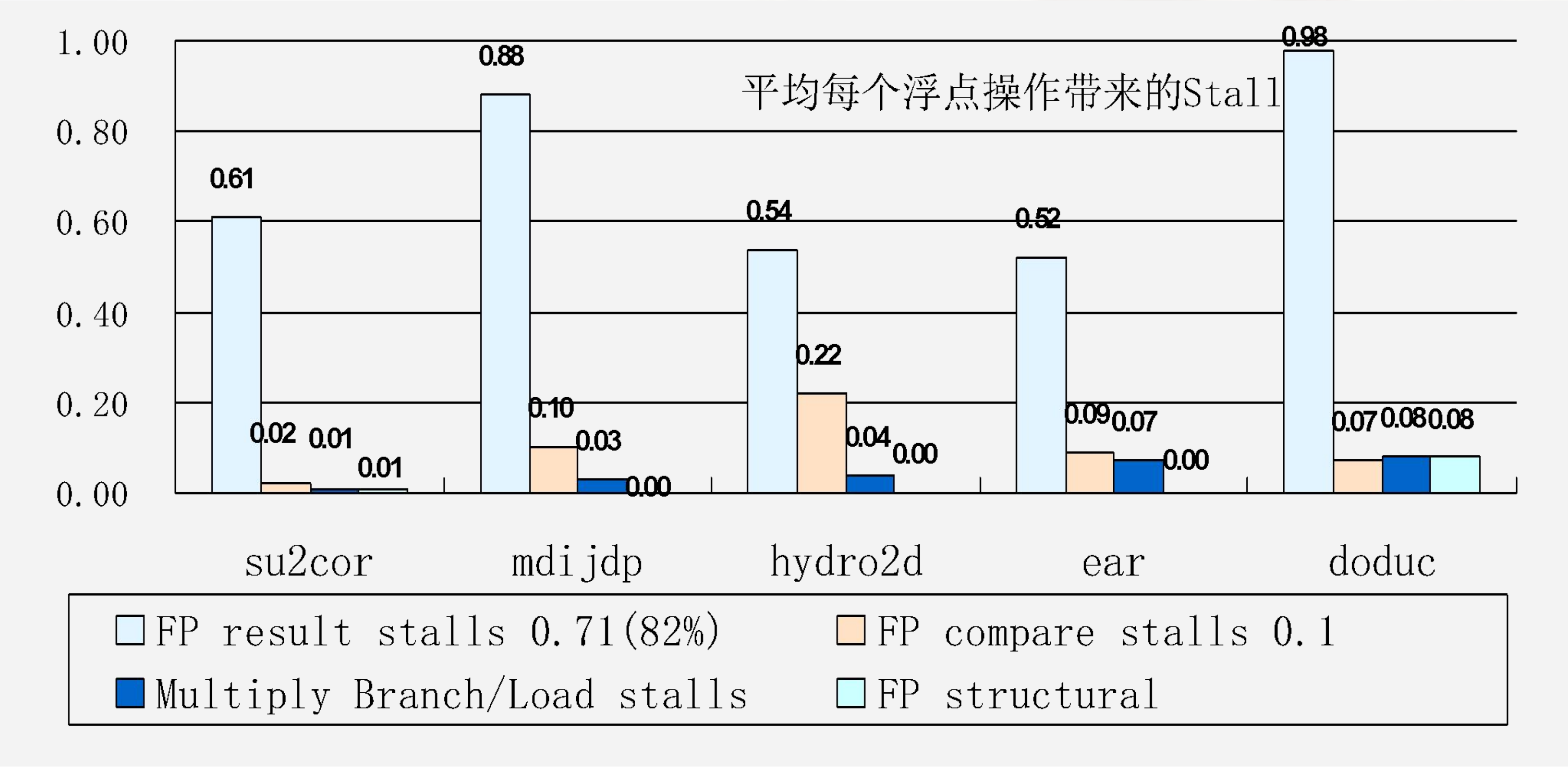
- ❑ Check for structural hazards .
  - ➢ The divider or other not fully pipelined function units
- ❑ Check for RAW hazards
  - ➢ The CPU simply stalls the instruction at ID stage until:
    - ○ Its source registers are no longer listed as destinations in any of the execution pipeline registers (registers between stages of M and A) OR
    - ○ Its source registers are no longer listed as the destination of a load in the EX/MEM register.
- ❑ Check for WAW hazards
  - ➢ Check instructions in A1, ..., A4, Divide, or M1, ...,M7 for the same destination register (check pipeline registers.)
  - ➢ If no WAW hazard, reserve Register write port.
- ❑ Stall instruction in ID if necessary.

Note: reserve the write port
only when instruction in ID can be issued

Performance of FP pipeline



Performance of FP pipeline



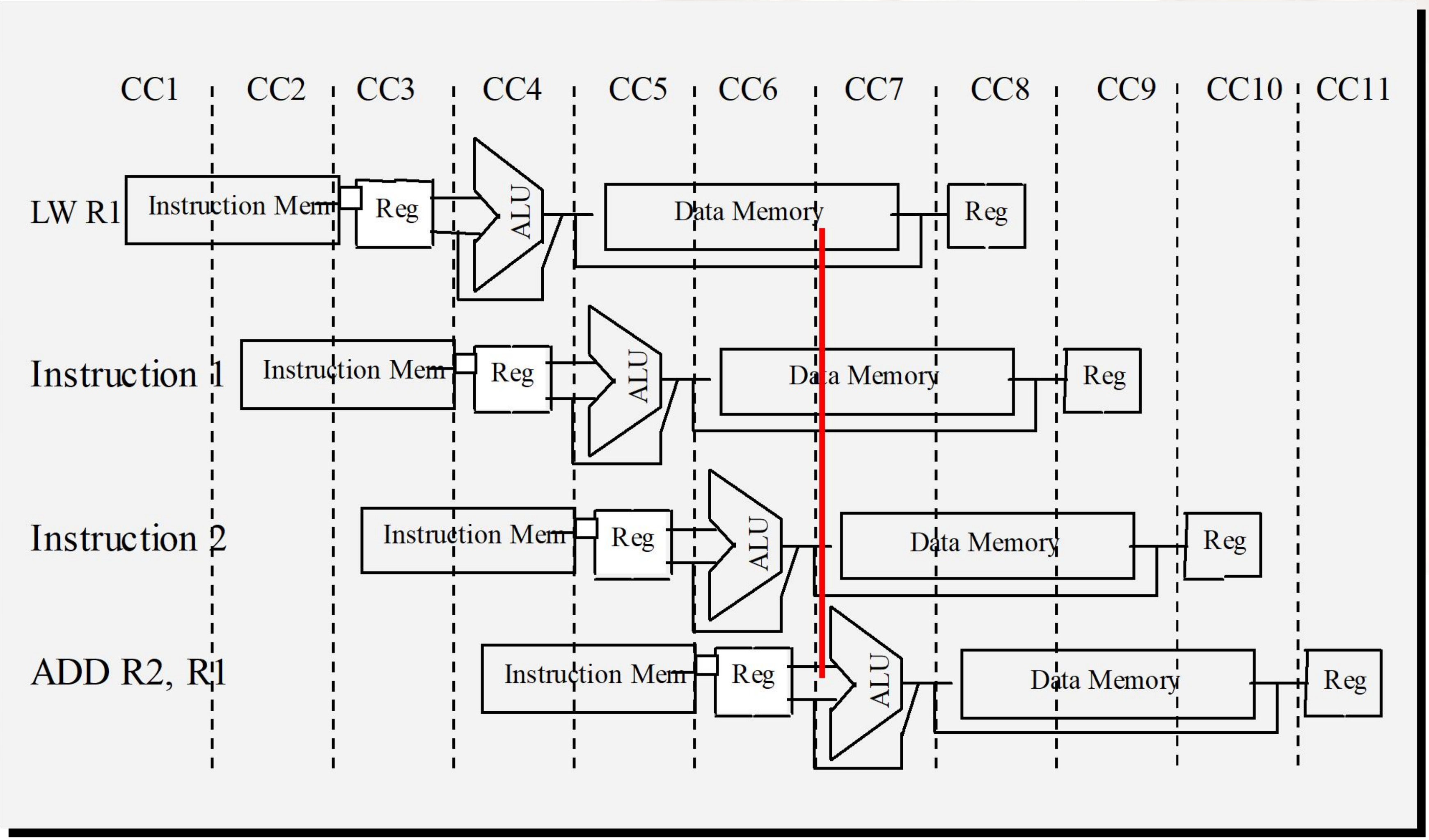
The MIPS R4000 pipeline

- IF—First half of instruction fetch. PC selection occurs. Cache access is initiated.
- IS—Second half of instruction fetch.
  - —This allows the cache access to take two cycles.
- RF—Decode and register fetch, hazard checking, I-cache hit detection.
- EX—Execution: address calculation, ALU Ops, branch target calculation and condition evaluation.
- DF/DS/TC
  - — Data fetched from cache in the first two cycles.
  - — The third cycle involves checking a tag check to determine if the cache access was a hit.
- WB—Write back result for loads and R-R operations.

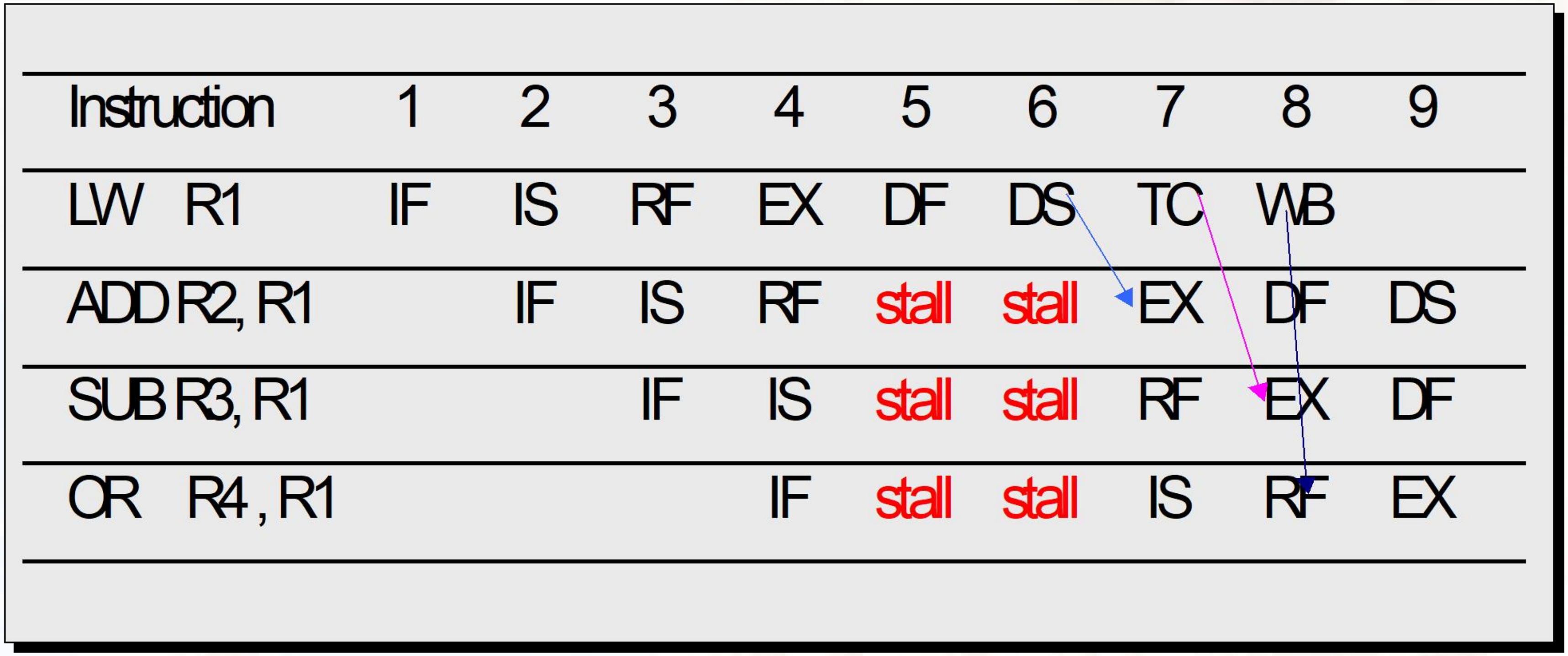
Possible stalls and delays

- ❑Load delay: two cycles
  - ➢The delay might seem to be three cycles, since the tag isn't checked until the end of the TC cycle.
  - ➢However, if TC indicates a miss, the data must be fetched from main memory and the pipeline is backed up to get the real value.

Load stalls – 2 stalls



Example： load stalls

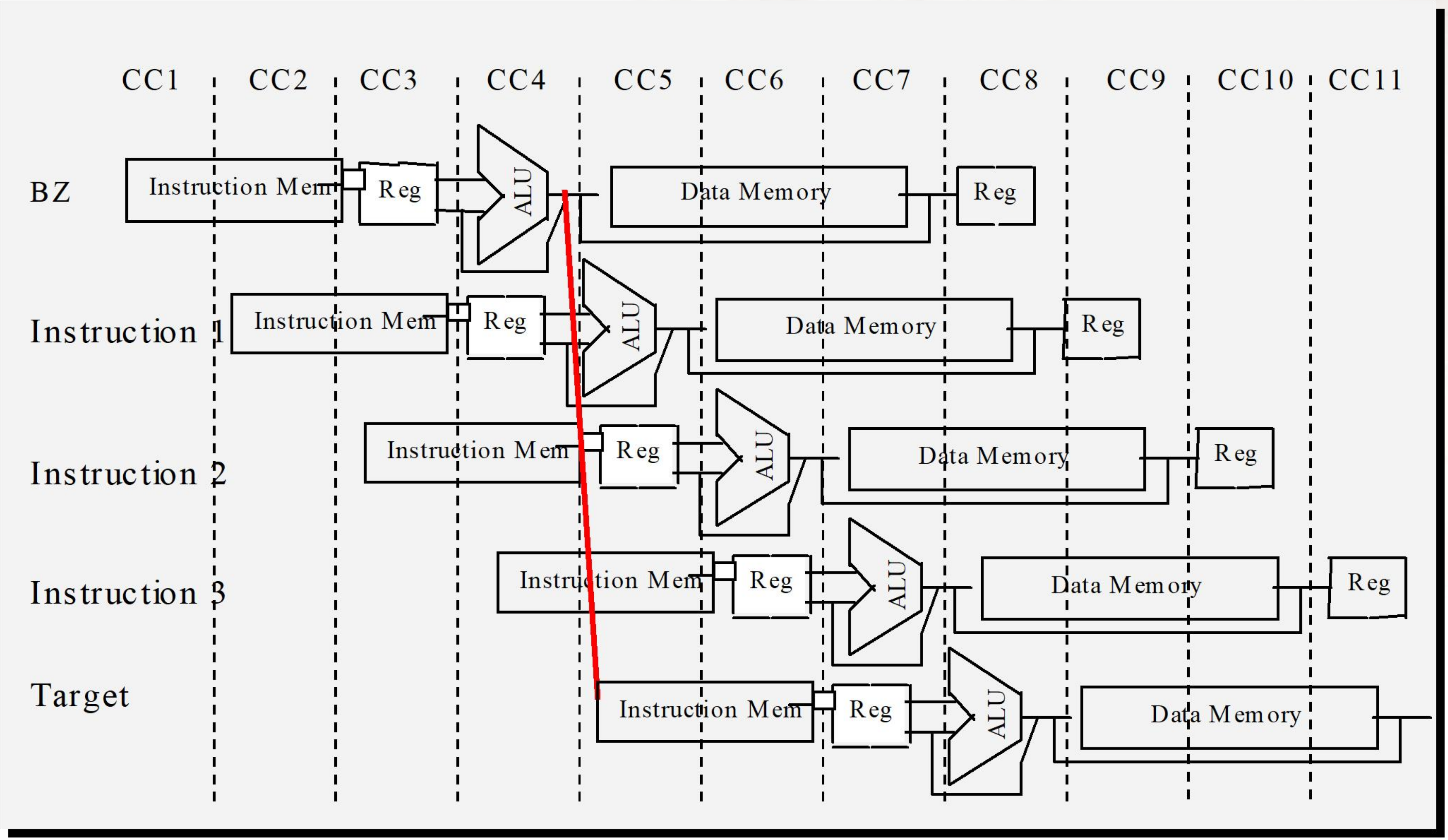


Instruction	1	2	3	4	5	6	7	8	9
LW R1	IF	IS	RF	EX	DF	DS	TC	WB	
ADD R2, R1		IF	IS	RF	stall	stall	EX	DF	DS
SUB R3, R1			IF	IS	stall	stall	RF	EX	DF
OR R4 , R1				IF	stall	stall	IS	RF	EX

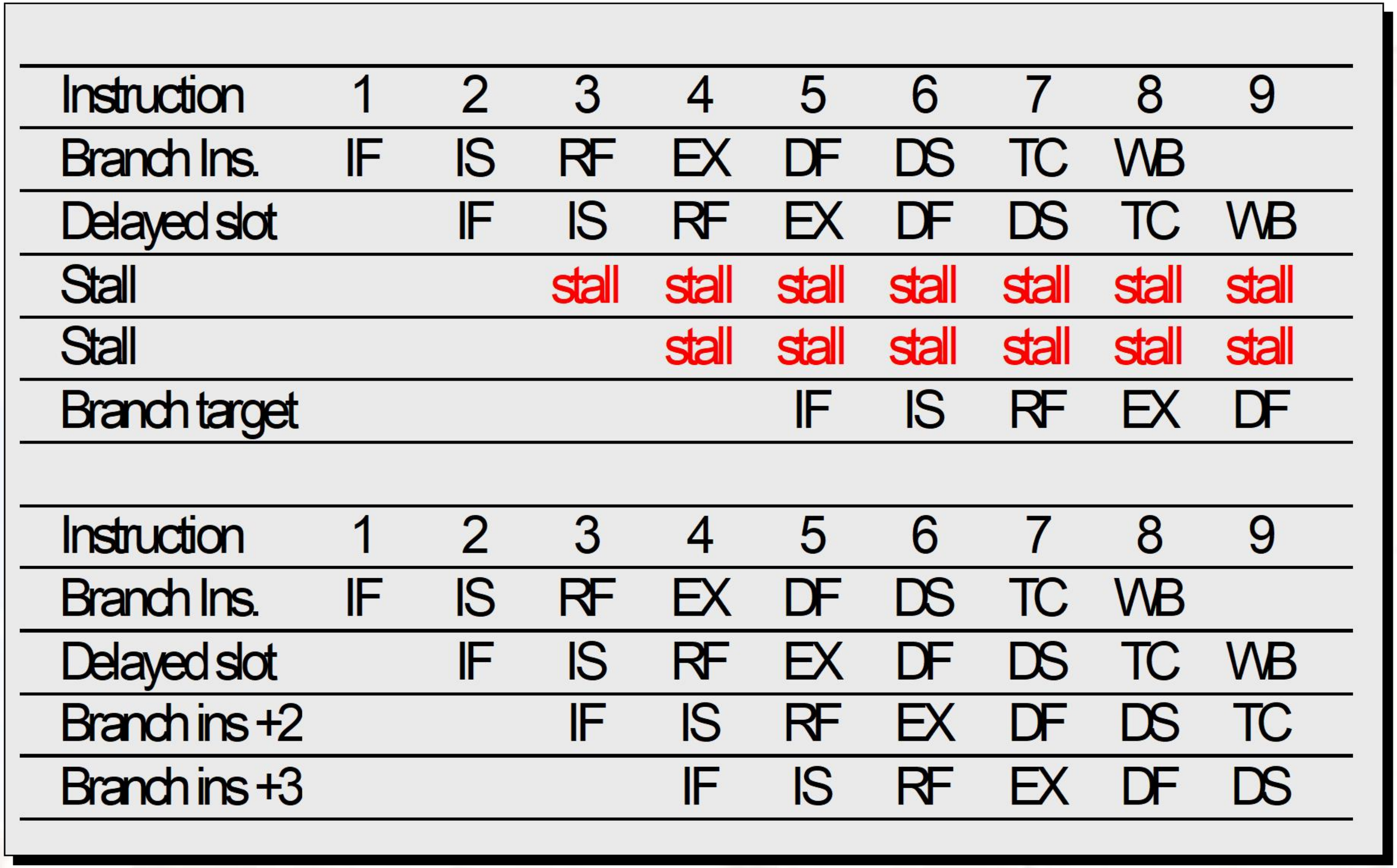
Branch delay: 3 cycles

- ❑Branch delay: three cycles (including one branch delay slot)
  - ➢The branch is resolved during EX, giving a 3 cycle delay.
  - ➢The first cycle may be a regular branch delay slot (instruction always executed) or a branch-likely slot (instruction cancelled if branch not taken).
  - ➢MIPS uses a predict-not-taken method presumably because it requires the least hardware.

Branch Delays： 3 stalls



Pipeline status for branch latency

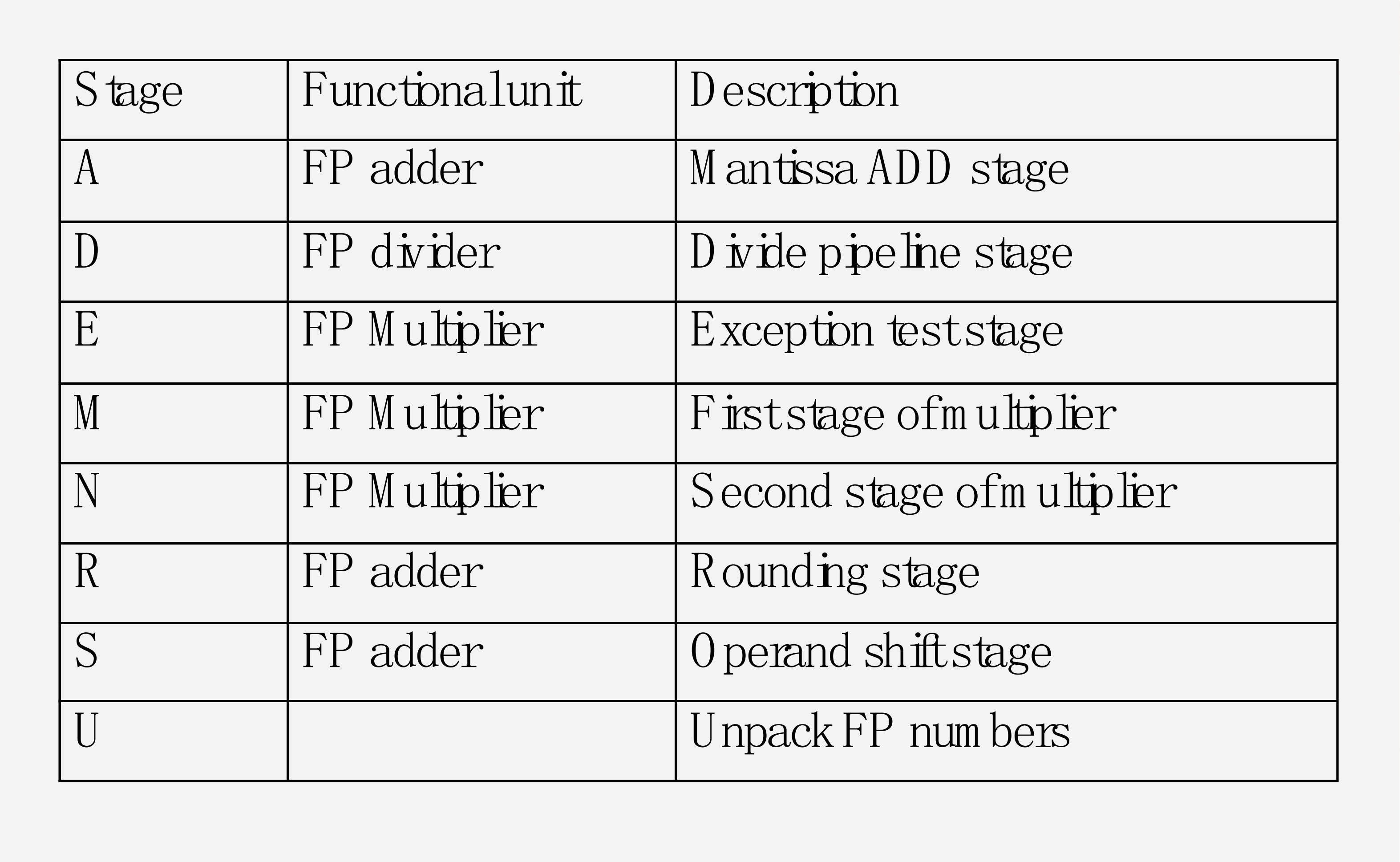


Instruction	1	2	3	4	5	6	7	8	9
Branch Ins.	IF	IS	RF	EX	DF	DS	TC	WB	
Delayed slot		IF	IS	RF	EX	DF	DS	TC	WB
Stall			stall	stall	stall	stall	stall	stall	stall
Stall				stall	stall	stall	stall	stall	stall
Branch target					IF	IS	RF	EX	DF

Instruction	1	2	3	4	5	6	7	8	9
Branch Ins.	IF	IS	RF	EX	DF	DS	TC	WB	
Delayed slot		IF	IS	RF	EX	DF	DS	TC	WB
Branch ins +2			IF	IS	RF	EX	DF	DS	TC
Branch ins +3				IF	IS	RF	EX	DF	DS

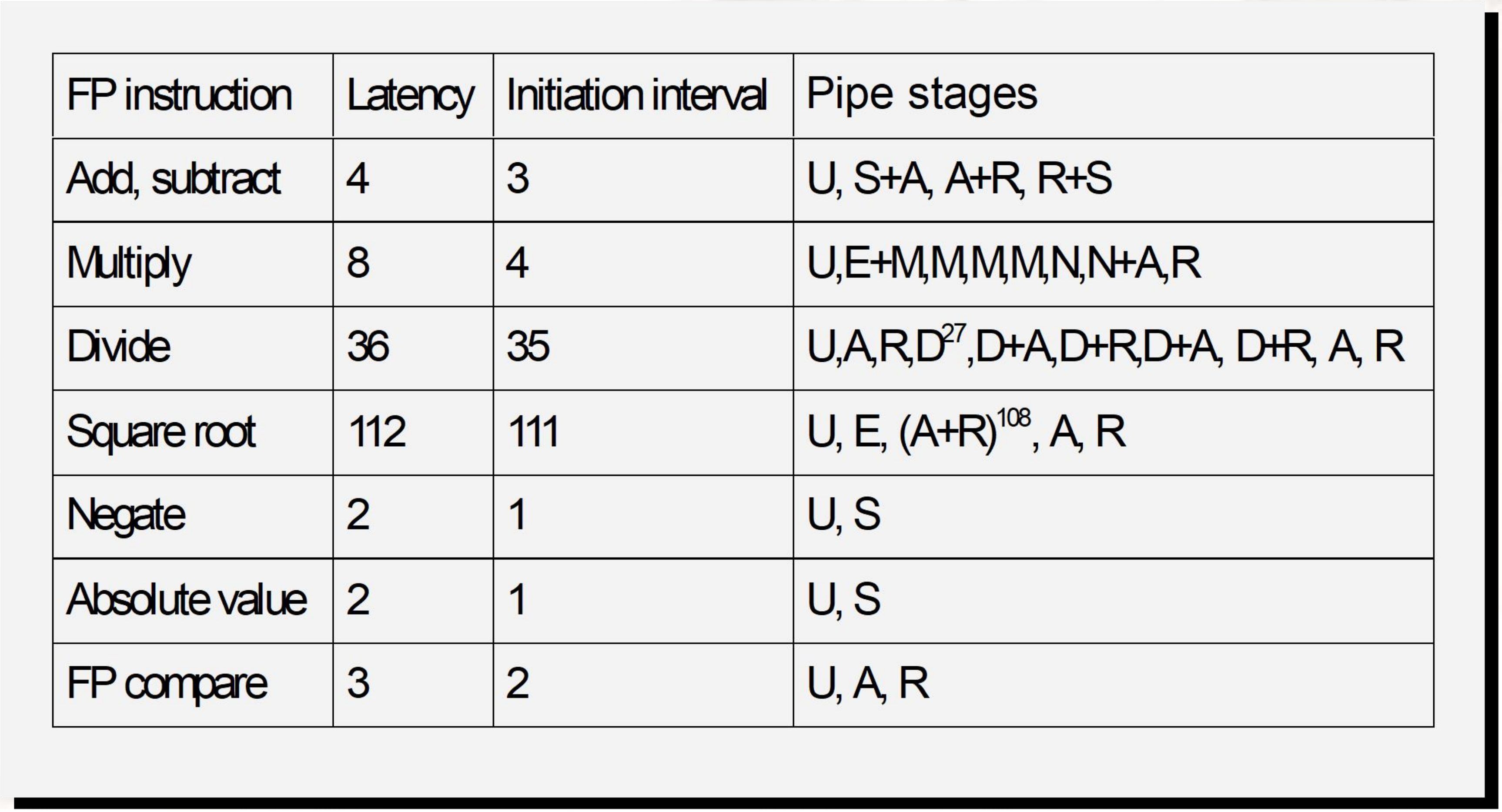
Predict-NOT-taken + Delayed Branch

The FP 8-stage operational pipeline



Stage	Functional unit	Description
A	FP adder	Mantissa ADD stage
D	FP divider	Divide pipeline stage
E	FP Multiplier	Exception test stage
M	FP Multiplier	First stage of multiplier
N	FP Multiplier	Second stage of multiplier
R	FP adder	Rounding stage
S	FP adder	Operand shift stage
U		Unpack FP numbers

Latency and initiation intervals

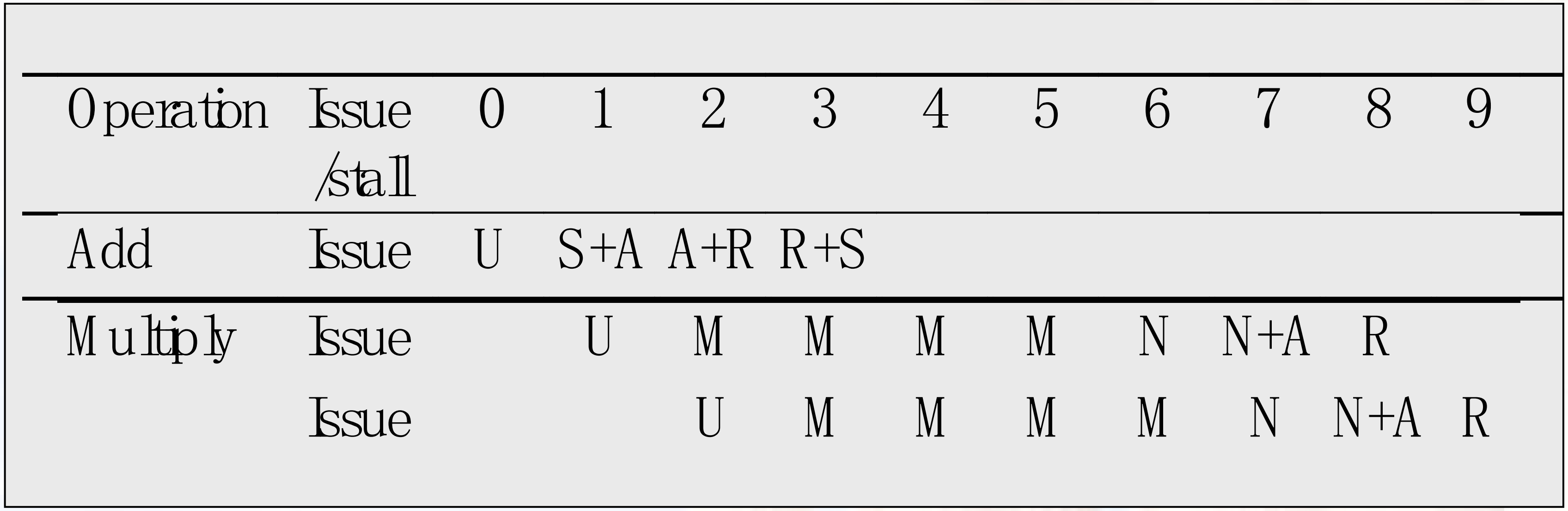


FP instruction	Latency	Initiation interval	Pipe stages
Add, subtract	4	3	U, S+A, A+R, R+S
Multiply	8	4	U,E+M,M,M,M,N,N+A,R
Divide	36	35	U,A,R,D^{27},D+A,D+R,D+A, D+R, A, R
Square root	112	111	U, E, (A+R)^{108}, A, R
Negate	2	1	U, S
Absolute value	2	1	U, S
FP compare	3	2	U, A, R

Structural hazards-1

Operation	Issue /stall	0	1	2	3	4	5	6	7	8	9
Multiply	Issue	U	M	M	M	M	N	N+A	R		
Add	Issue		U	S+A	A+R	R+S					
	Issue			U	S+A	A+R	R+S				
	Issue				U	S+A	A+R	R+S			
	Stall					U	S+A	A+R	R+S		
	Stall						U	S+A	A+R	R+S	
	Issue							U	S+A	A+R	R+S
	Issue								U	S+A	A+R

Structural hazards-2



Operation	Issue /stall	0	1	2	3	4	5	6	7	8	9
Add	Issue	U	S+A	A+R	R+S						
Multiply	Issue		U	M	M	M	M	N	N+A	R	
	Issue			U	M	M	M	M	N	N+A	R

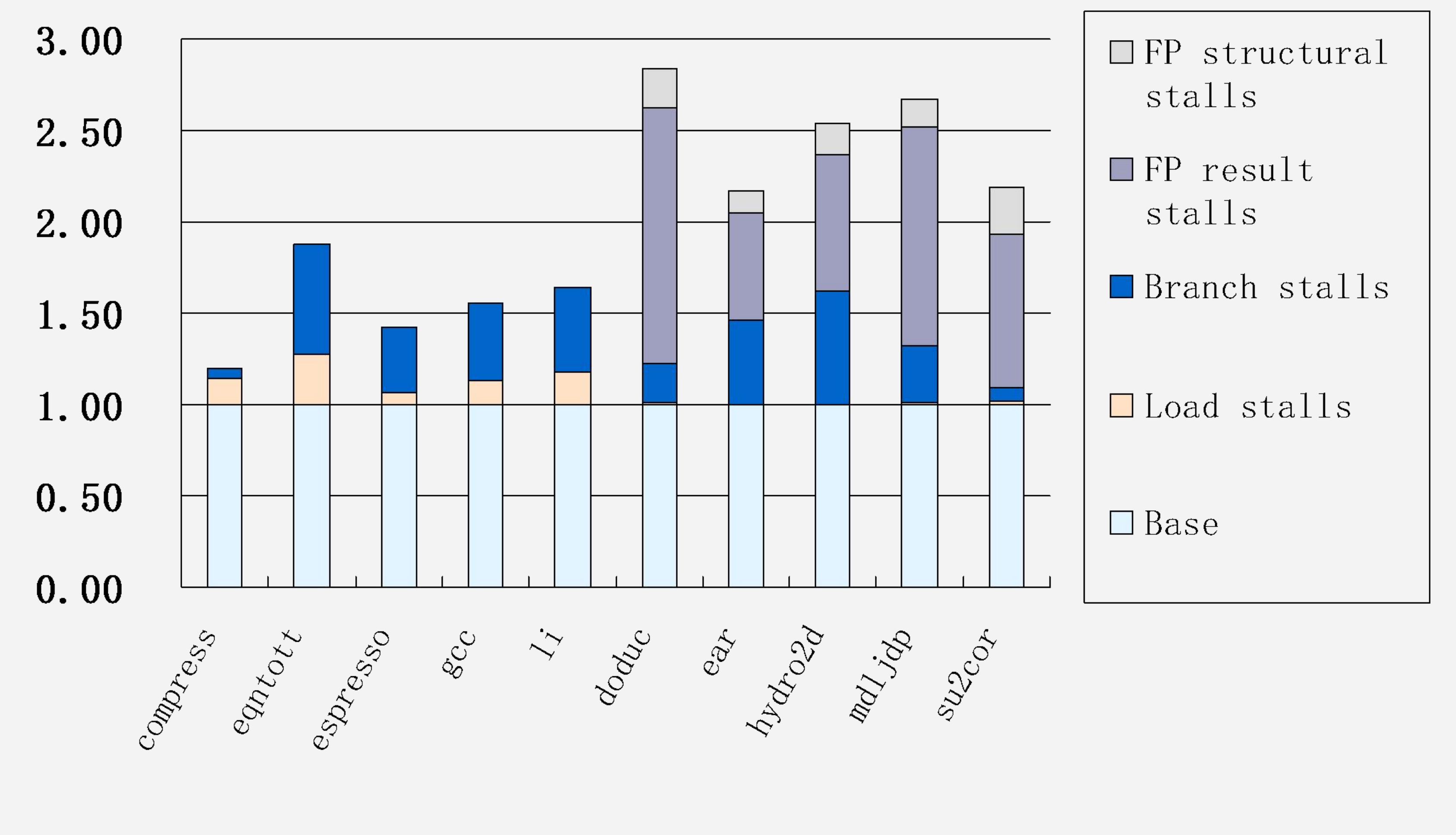
Structural hazards-3

Operation	Issue/stall	25	26	27	28	29	30	31	32	33	34	35
Divide	Issue in cycle 0	D	D	D	D	D	D+A	D+R	D+A	D+R	A	R
Add	Issue		U	S+A	A+R	R+S						
	Issue			U	S+A	A+R	R+S					
	Stall				U	S+A	A+R	R+S				
	Stall					U	S+A	A+R	R+S			
	Stall						U	S+A	A+R	R+S		
	Stall							U	S+A	A+R	R+S	
	Stall								U	S+A	A+R	R+S
	Stall									U	S+A	A+R
	Issue										U	S+A
	Issue											U

Structural hazards-4

Operation	Issue	0	1	2	3	4	5	6	7	8	9
Add	stall	U	S+A	A+R	R+S						
Divide	Issue		U	A	R	D	D	D	D	D	D
	Issue			U	A	R	D	D	D	D	D

Performance loss measurements


compress
eqntott
espresso
gcc
li
doduc
ear
hydro2d
mdljdp
su2cor
FP structural stalls
FP result stalls
Branch stalls
Load stalls
Base