Appunti su Approfondimento

1) Cosa si intende per pipeline? Quali sono le sue caratteristiche?

Pipeline è una tecnica dove più istruzioni vengono sovrapposte durante l'esecuzione, con le seguenti caratteristiche:

→ viene suddiviso in fasi e queste fasi sono collegate tra loro per formare una struttura simile a un tubo;

→ le istruzioni entrano da un'estremità ed escono da un'altra estremità;

→ aumenta il throughput complessivo delle istruzioni.

2) Quali sono le 5 fasi di pipeline?

Le 5 fasi di pipeline sono:

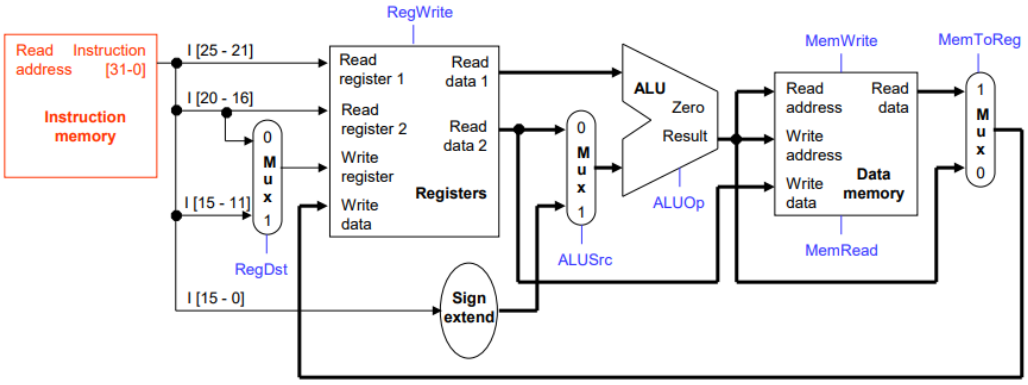
Step	Nome	Descrizione
Instruction Fetch	IF	Legge l'istruzione dalla memoria
Instruction Decode	ID	Legge il registro sorgente e genera il segnale di controllo
Execute	EX	Esegue istruzione un'istruzione R type o branch
Memory	MEM	Legge da o scrive in data memory
Writeback	WB	Memorizza il risultato nel registro di destinazione

Esempio:

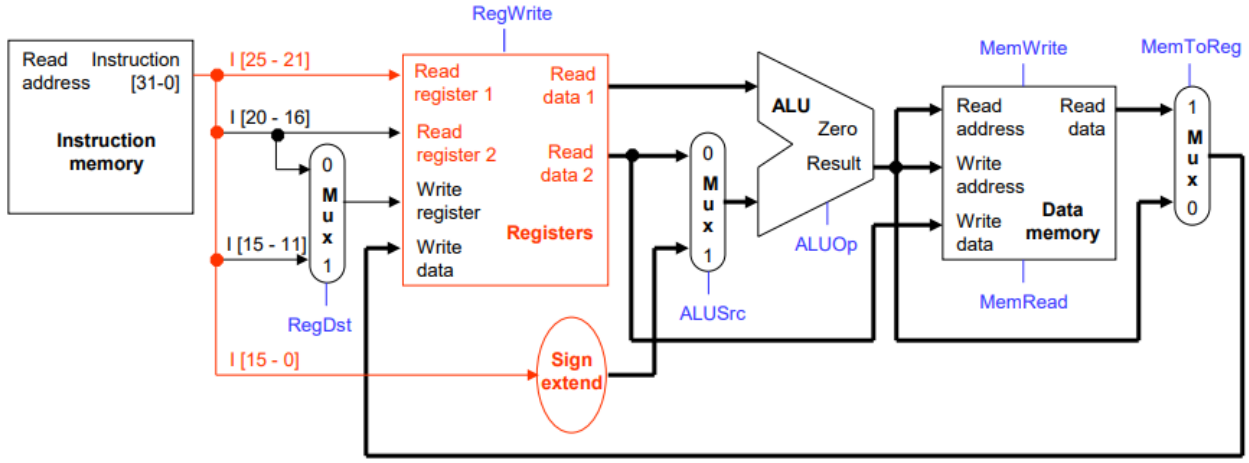
Istruzione	Step
Branch on Equal	IF ID EX
R type	IF ID EX WB
Store word	IF ID EX MEM
Load word	IF ID EX MEM WB

Visualizzazione delle fasi con Datapath

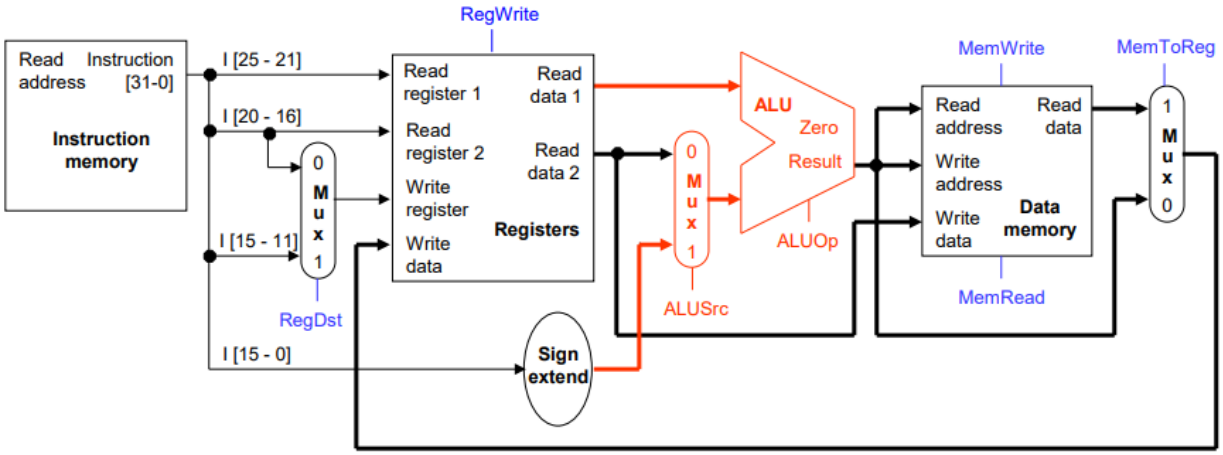
1) Instruction Fetch (IF): legge l'istruzione dalla memoria



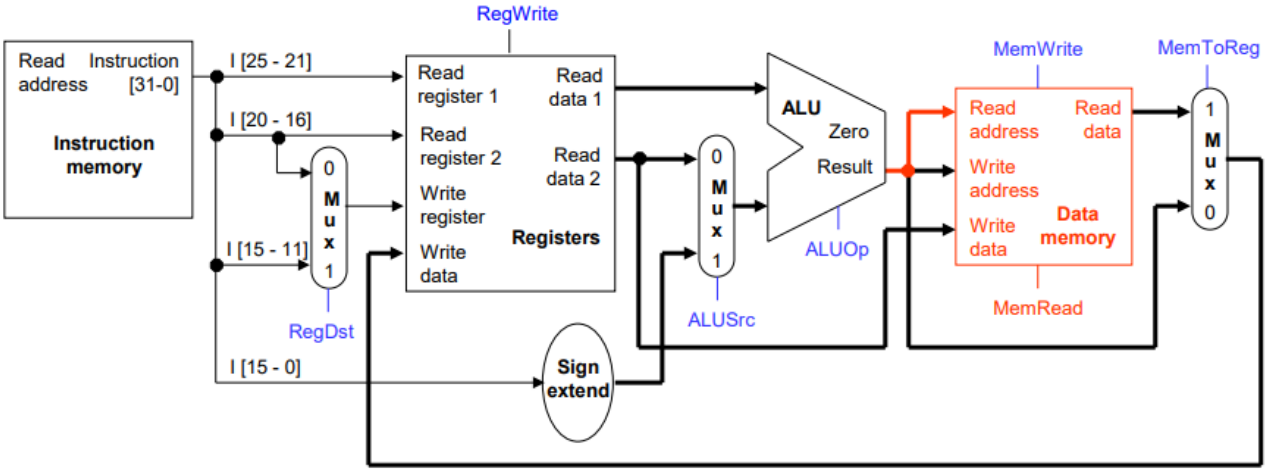
2) Instruction Decode (ID): legge il registro sorgente e genera il segnale di controllo



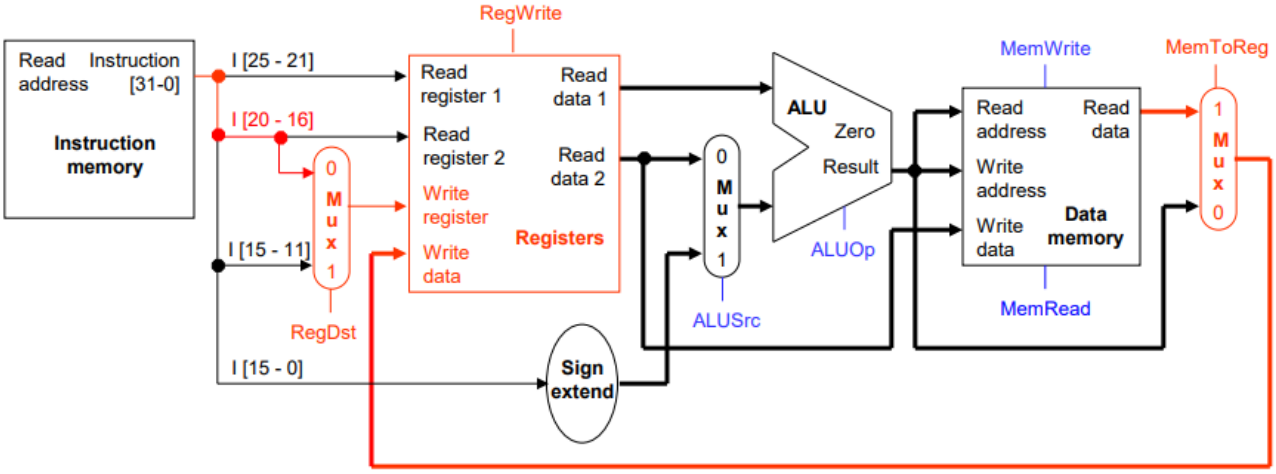
3) Execute: esegue istruzione un'istruzione R type o branch.



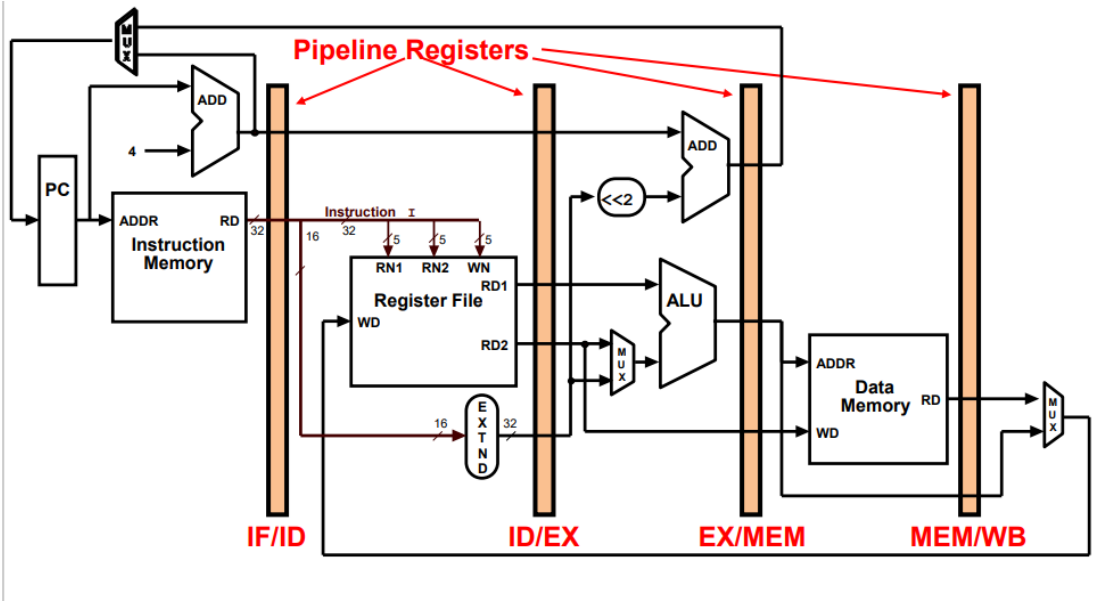
4) Memory: legge da o scrive in data memory



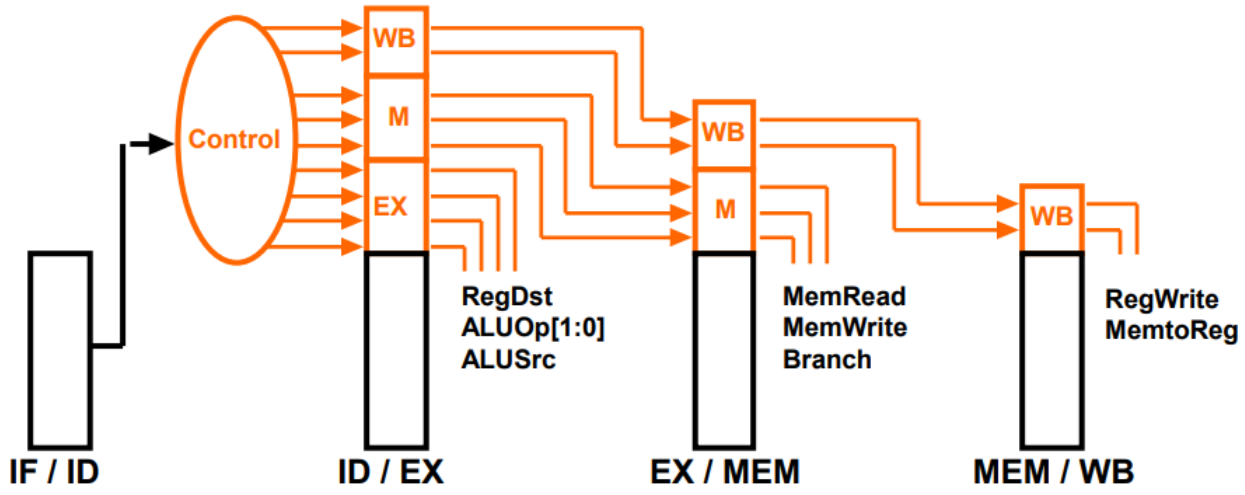
5) Writeback: memorizza il risultato nel registro di destinazione



Visualizzazione Pipeline Data Processor



Segnali di controllo



3) Quali sono i vantaggi e svantaggi di pipeline?

I vantaggi di pipeline sono:

→ più istruzioni vengono elaborate contemporaneamente;

→ funziona perché le fasi sono isolate dai registri;

Gli svantaggi sono:

→ le istruzioni interferiscono con altre istruzioni (Hazard), che si suddivide in:

1 – Structure Hazard: istruzioni diverse potrebbero richiedere lo stesso componente hardware (ad es. memoria) nello stesso ciclo di clock;

2 – Control Hazard: si tende ad effettuare una decisione prima della valutazione della condizione;

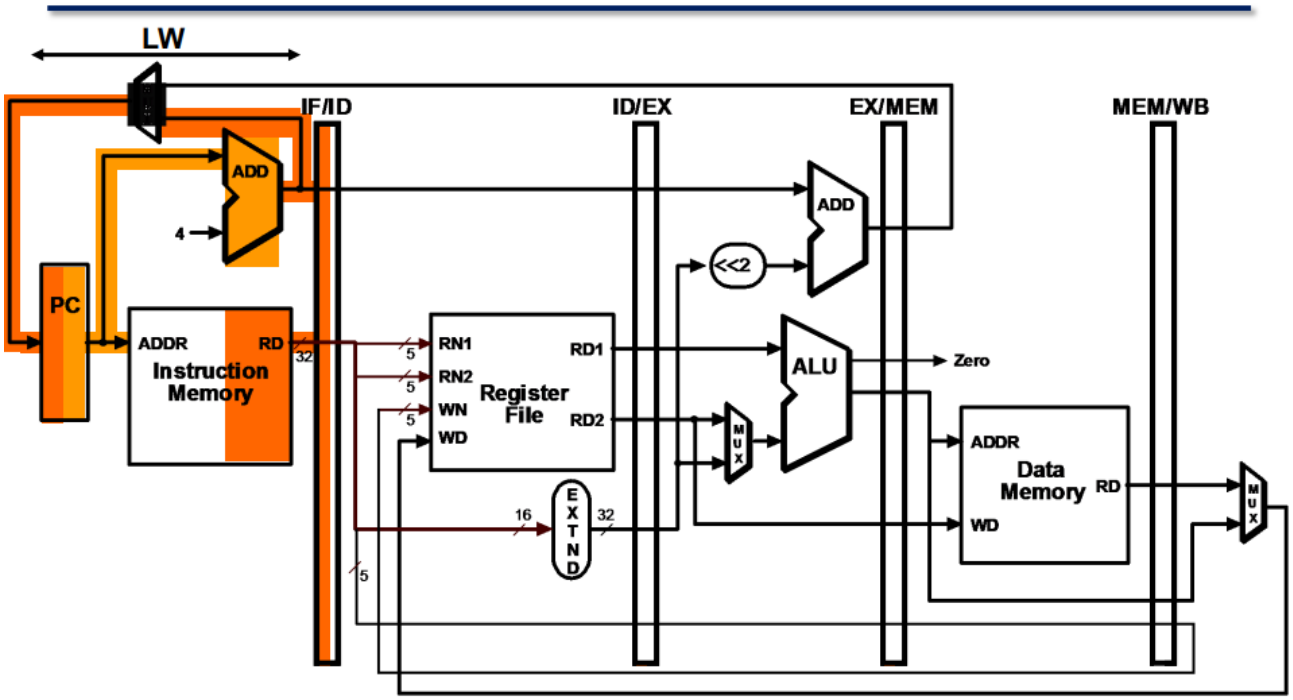
3 – Data Hazard: l'istruzione può richiedere un risultato prodotto da un'istruzione precedente che non è ancora stato completato;

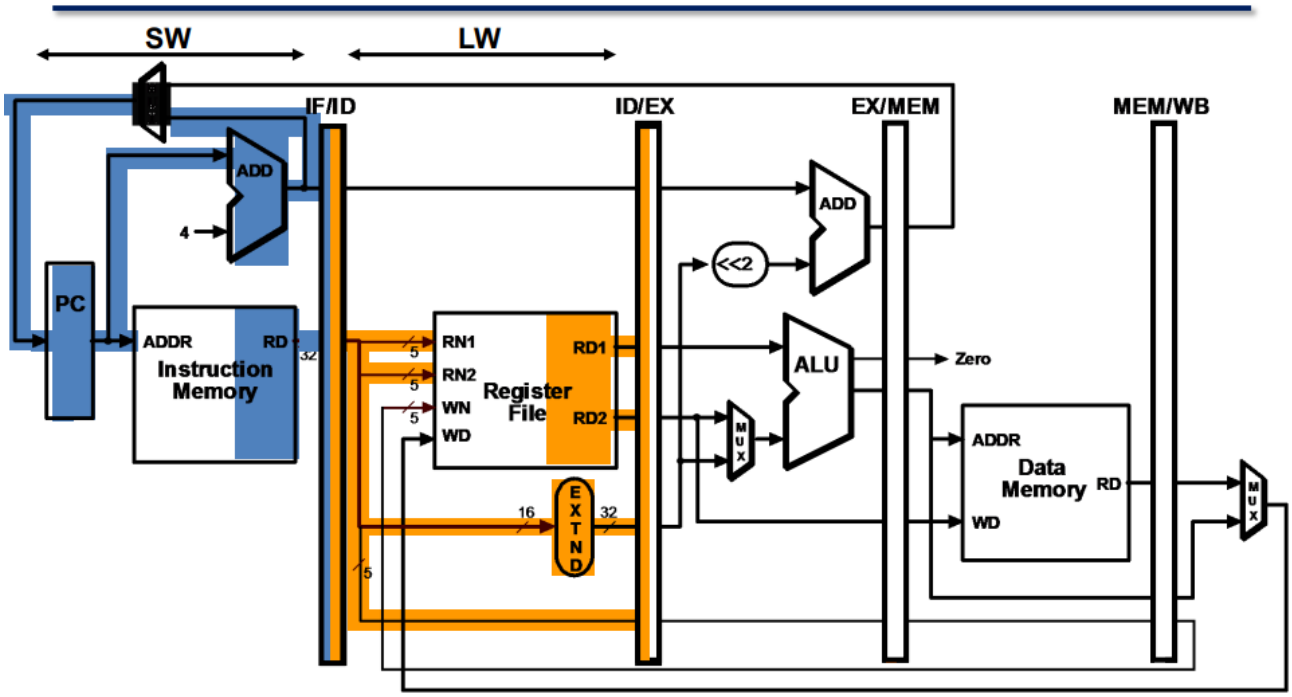
→ stallo di CPU.

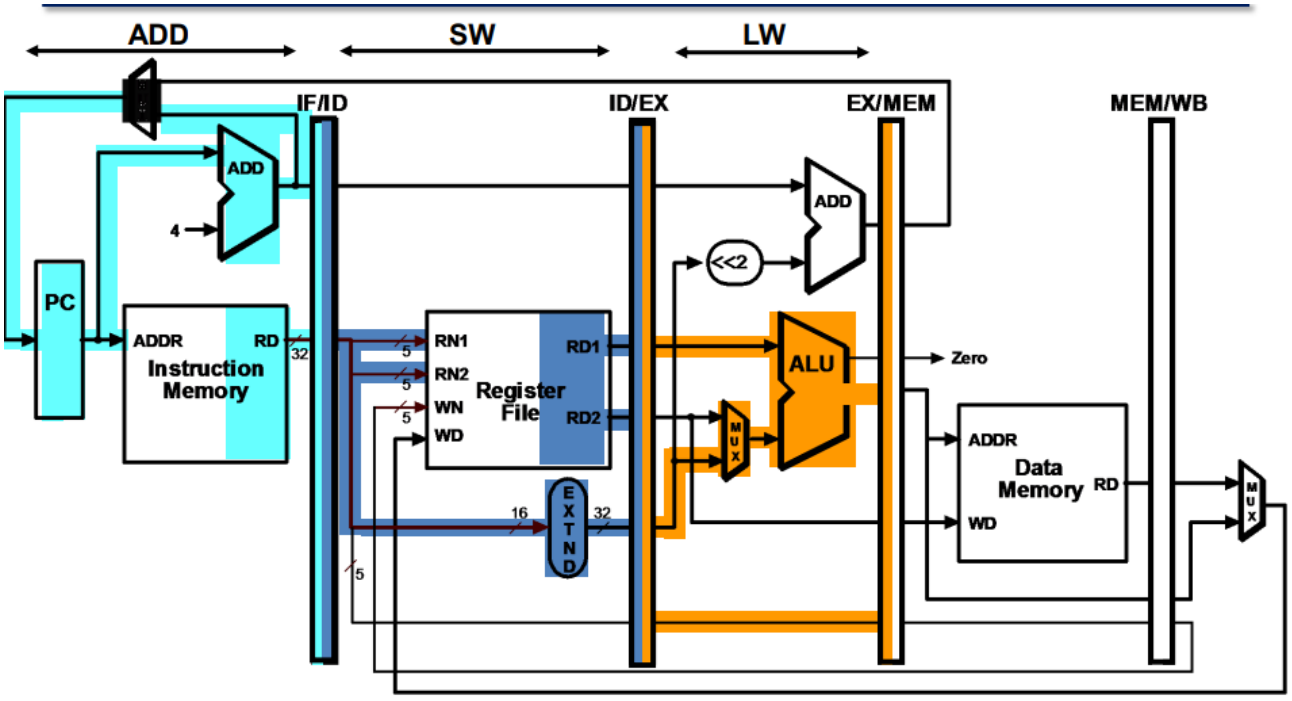
Ad esempio date queste istruzioni:

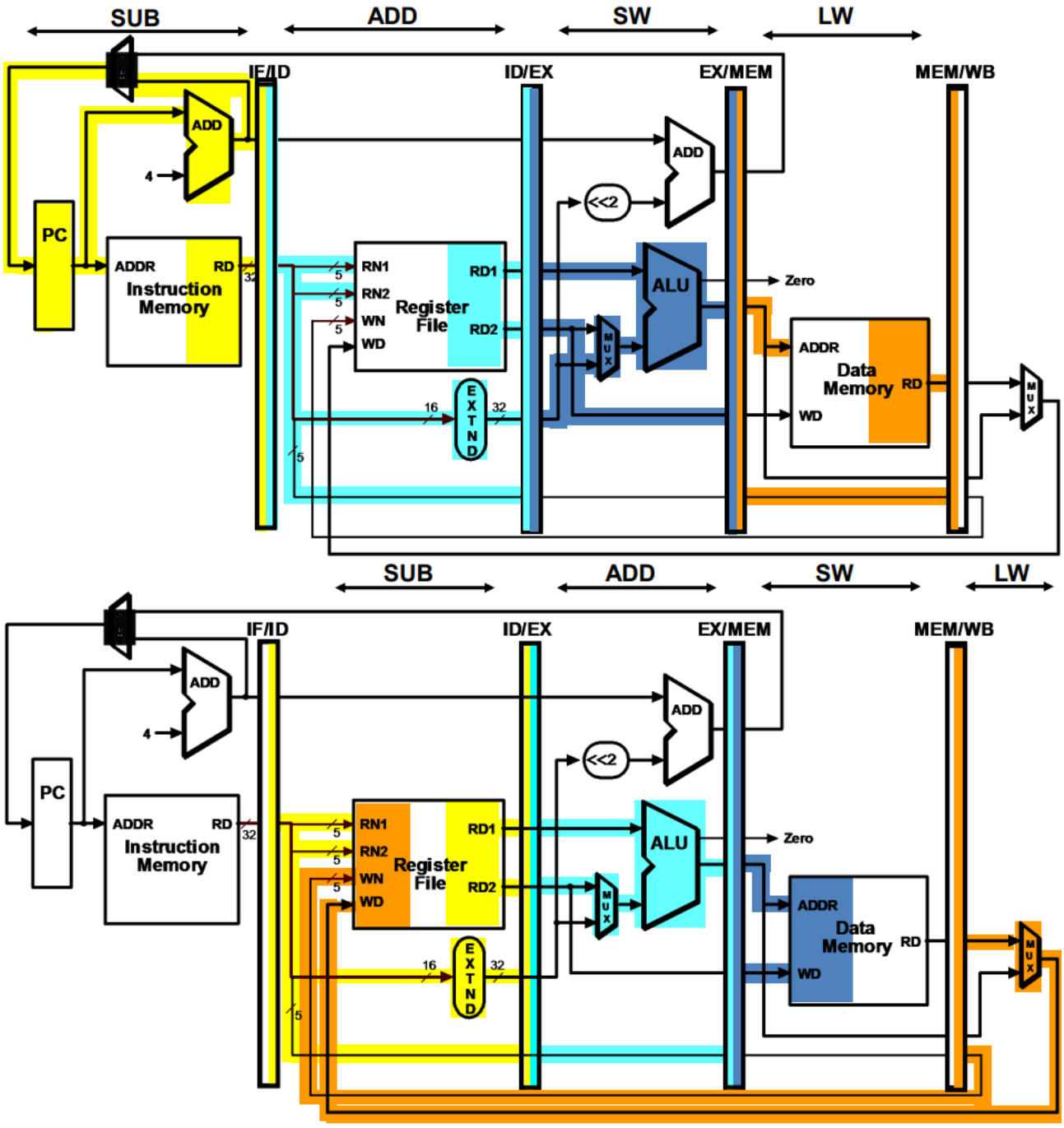
```
lw $r0, 10($r1)
sw $r3, 20($r4)
add $r5, $r6, $r7
sub $r8, $r9, $r10
```

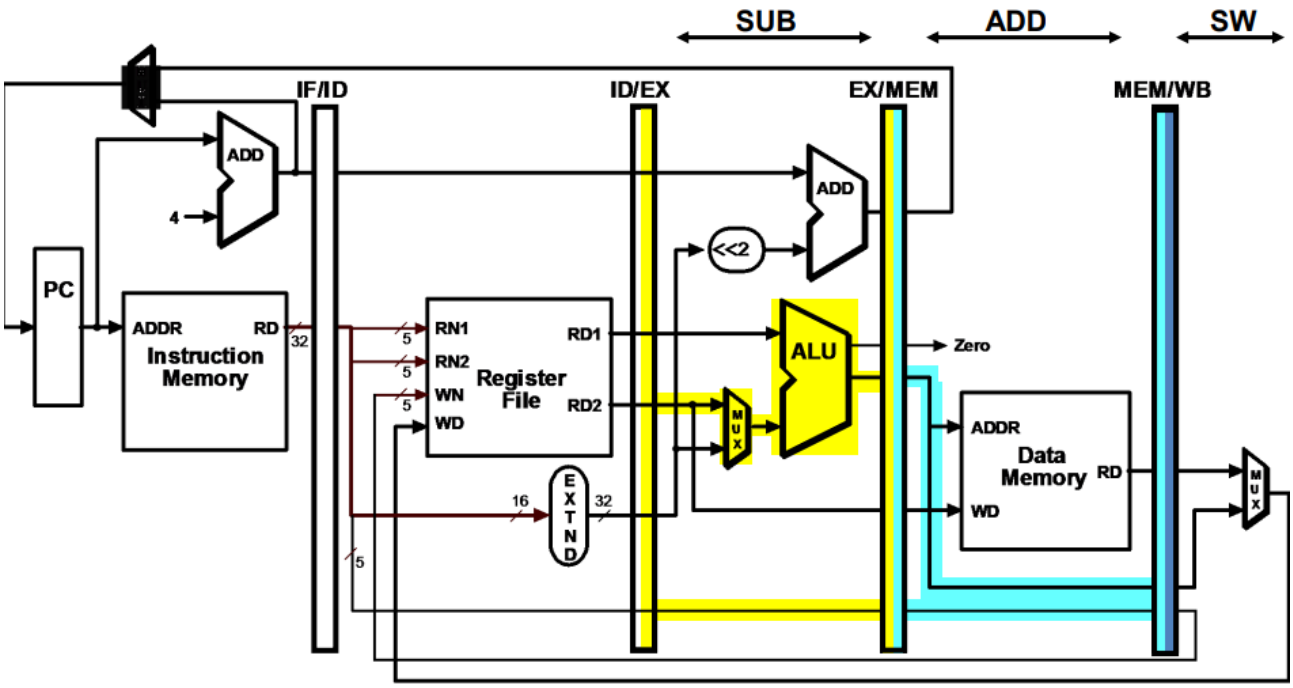
Si hanno le seguenti fasi:

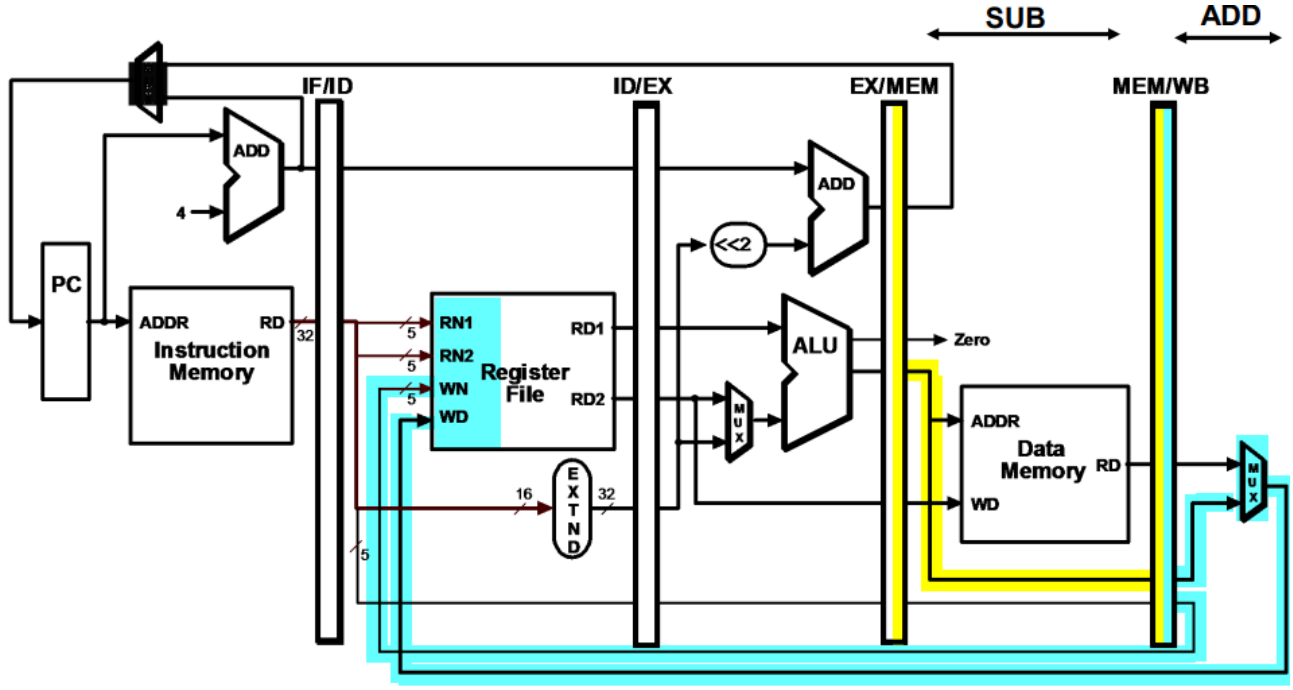


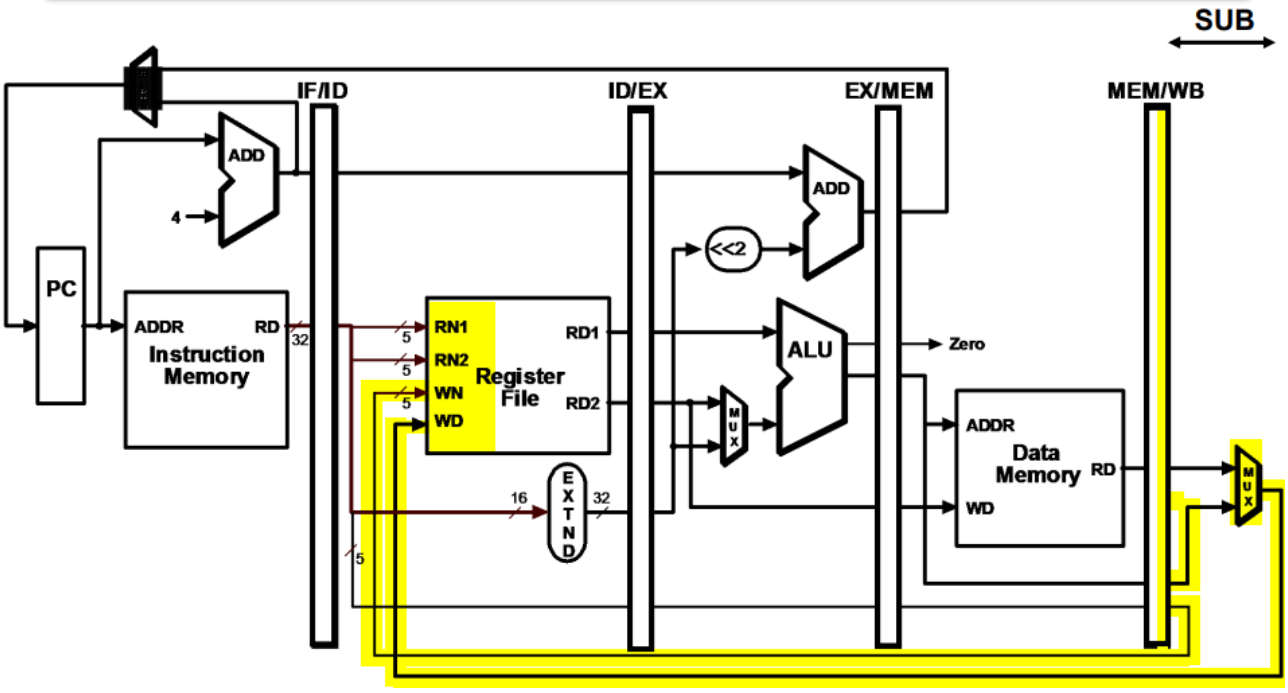

SW
LW
IF/ID
ID/EX
EX/MEM
MEM/WB
ADD
4
PC
ADDR
RD
Instruction Memory
32
RN1
RN2
WN
WD
Register File
RD1
RD2
5
16
EXTND
32
<<2
ADD
ALU
Zero
MUX
ADDR
Data Memory
RD
WD
MUX

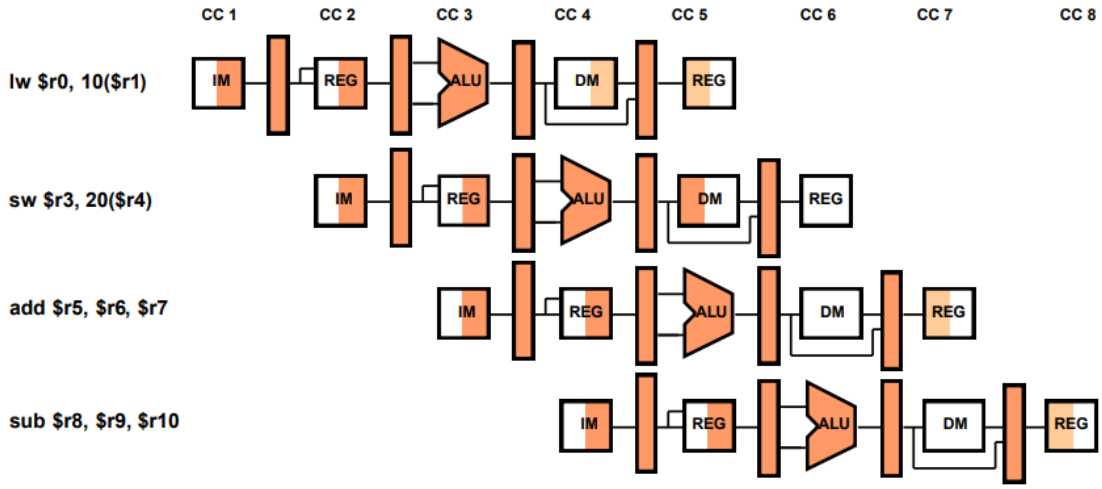

ADD
SW
LW
IF/ID
ID/EX
EX/MEM
MEM/WB
ADD
4
PC
ADDR
RD
Instruction Memory
32
RN1
RN2
WN
WD
Register File
RD1
RD2
5
16
EXTND
32
<<2
ADD
ALU
Zero
MUX
ADDR
Data Memory
RD
WD
MUX


SUB
ADD
SW
LW
IF/ID
ID/EX
EX/MEM
MEM/WB
ADD
4
PC
ADDR
RD
Instruction Memory
32
RN1
RN2
WN
WD
RD1
RD2
Register File
5
16
EXTND
32
<<2
ALU
Zero
MUX
ADDR
Data Memory
RD
WD


SUB
ADD
SW
IF/ID
ID/EX
EX/MEM
MEM/WB
ADD
4
PC
ADDR
RD
Instruction Memory
32
RN1
RN2
WN
WD
Register File
RD1
RD2
5
<<2
ALU
Zero
MUX
EXTND
16
32
ADDR
Data Memory
RD
WD

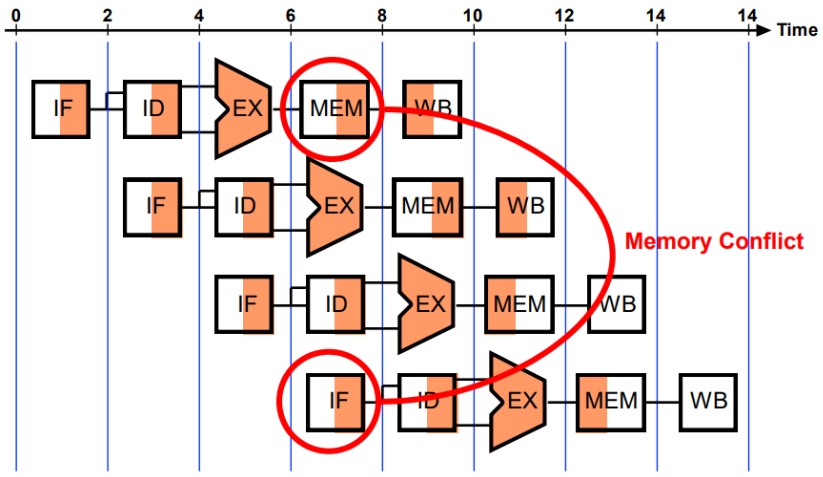

SUB
ADD
IF/ID
ID/EX
EX/MEM
MEM/WB
ADD
4
PC
ADDR
RD
Instruction Memory
32
RN1
RN2
WN
WD
Register File
RD1
RD2
5
<<2
ALU
Zero
MUX
EXTND
16
32
ADDR
Data Memory
RD
WD





4) Cosa si intende per Structural Hazard?

È una tipologia di hazard in cui viene utilizzata la stessa risorsa due volte contemporaneamente.



5) Quali sono le tipologie di pipeline hazard? Come può essere risolto?

Le pipeline hazards possono essere di tre tiplogie:

→ Structural hazards: in cui viene utilizzata la stessa risorsa in due diverse istruzioni contemporaneamente;

→ Data hazards: vengono utilizzati i dati prima che siano pronti e si ha che:

1 - gli operandi sorgente dell'istruzione vengono prodotti da un'istruzione precedente in pipeline;

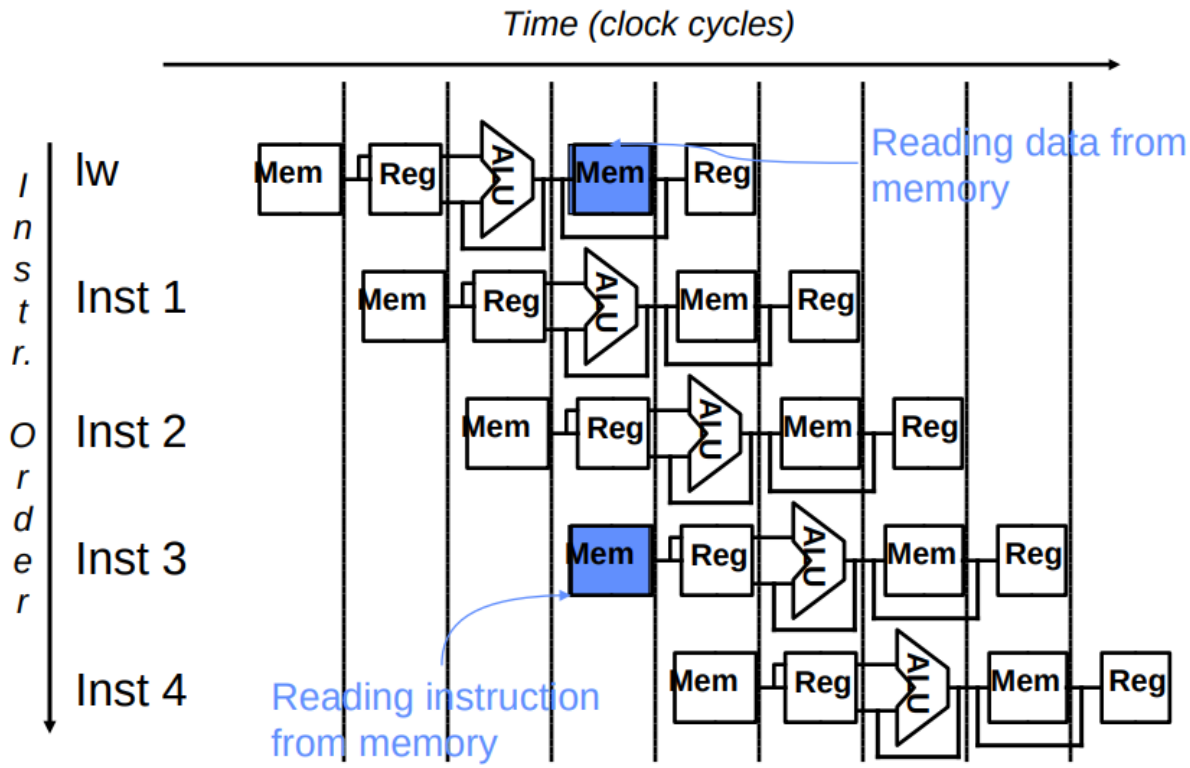
2 – istruzione di caricamento seguita immediatamente da un'istruzione ALU che utilizza l'operando di caricamento come valore di origine;

→ Control hazards: viene presa una decisione prima della valutazione della condizione (istruzioni di branch).

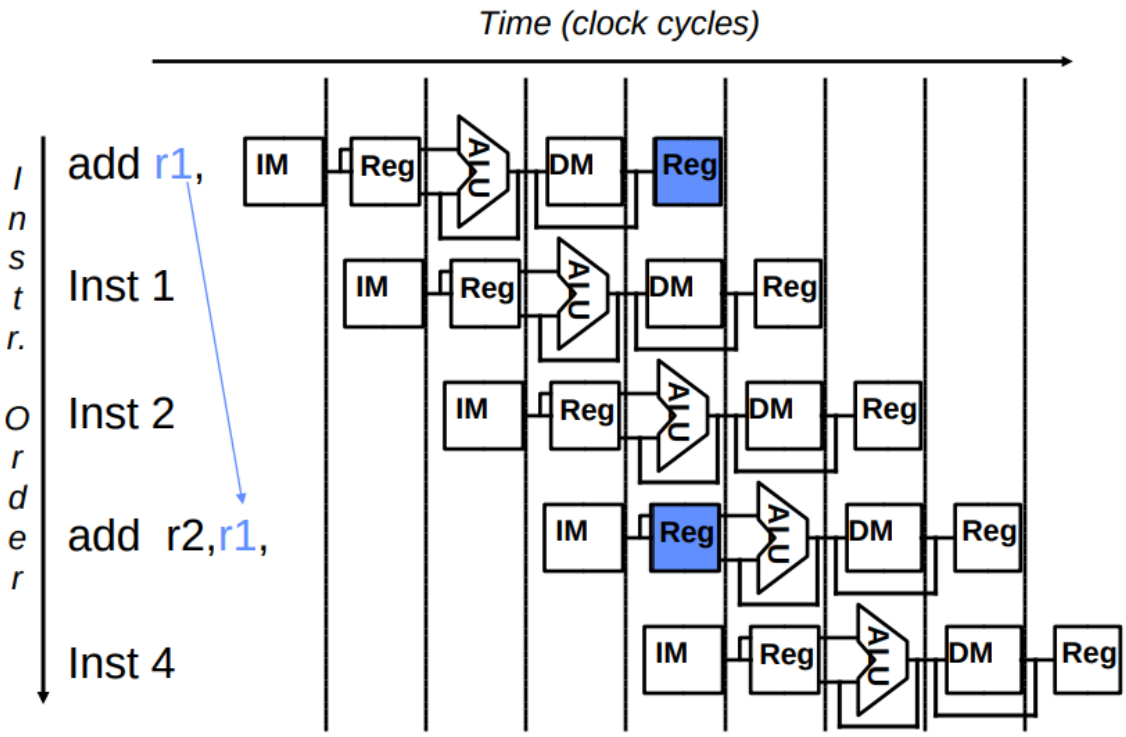
Gli hazards possono essere risolti mediante le attese (waiting) in cui il pipeline control deve ispezionare l'hazard e agire su come risolvere l'hazard.

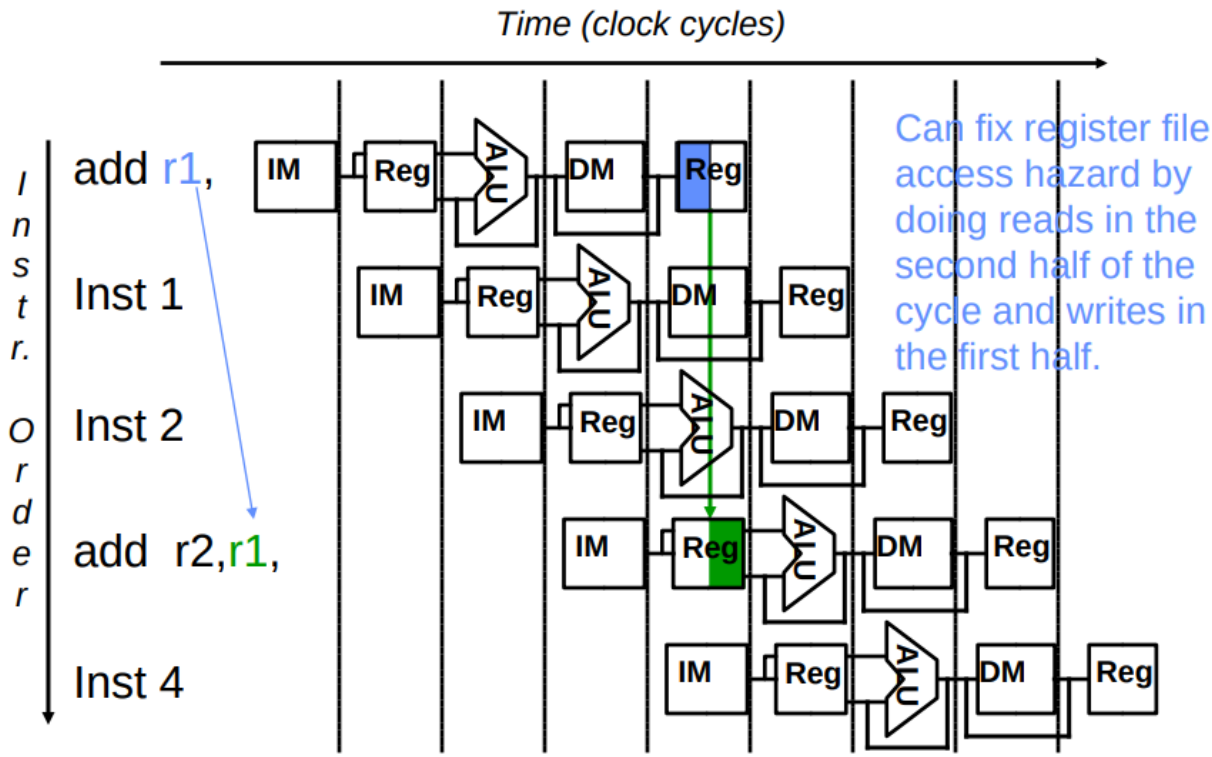
Esempi:

1 – La lettura in memoria potrebbe essere una Structural Hazard

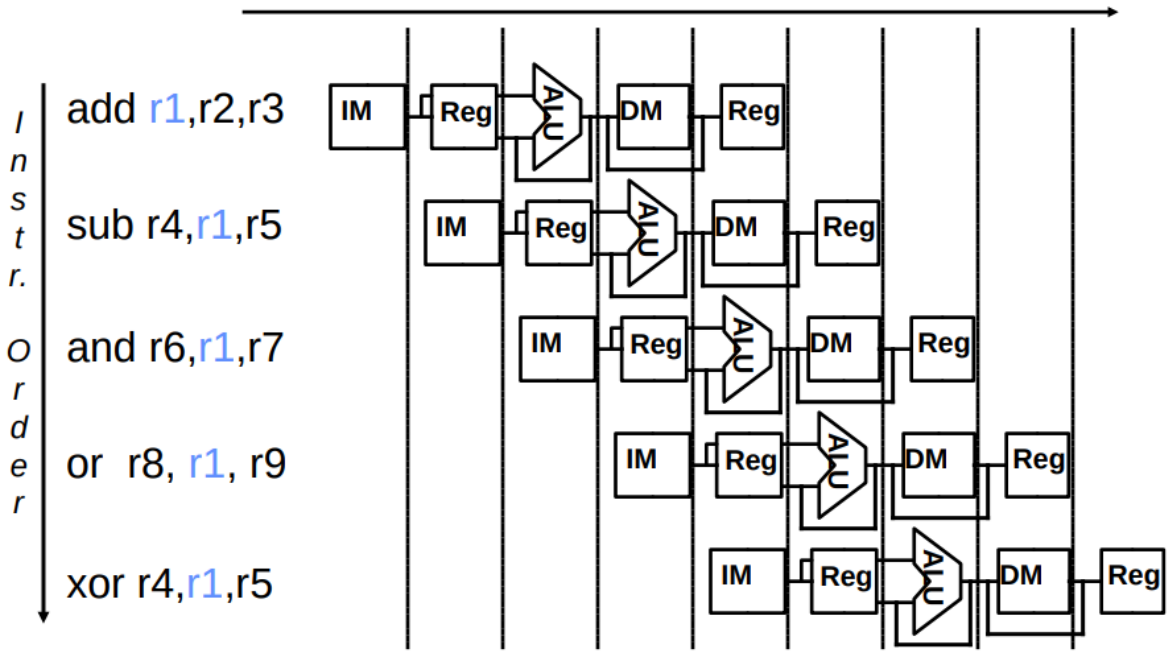


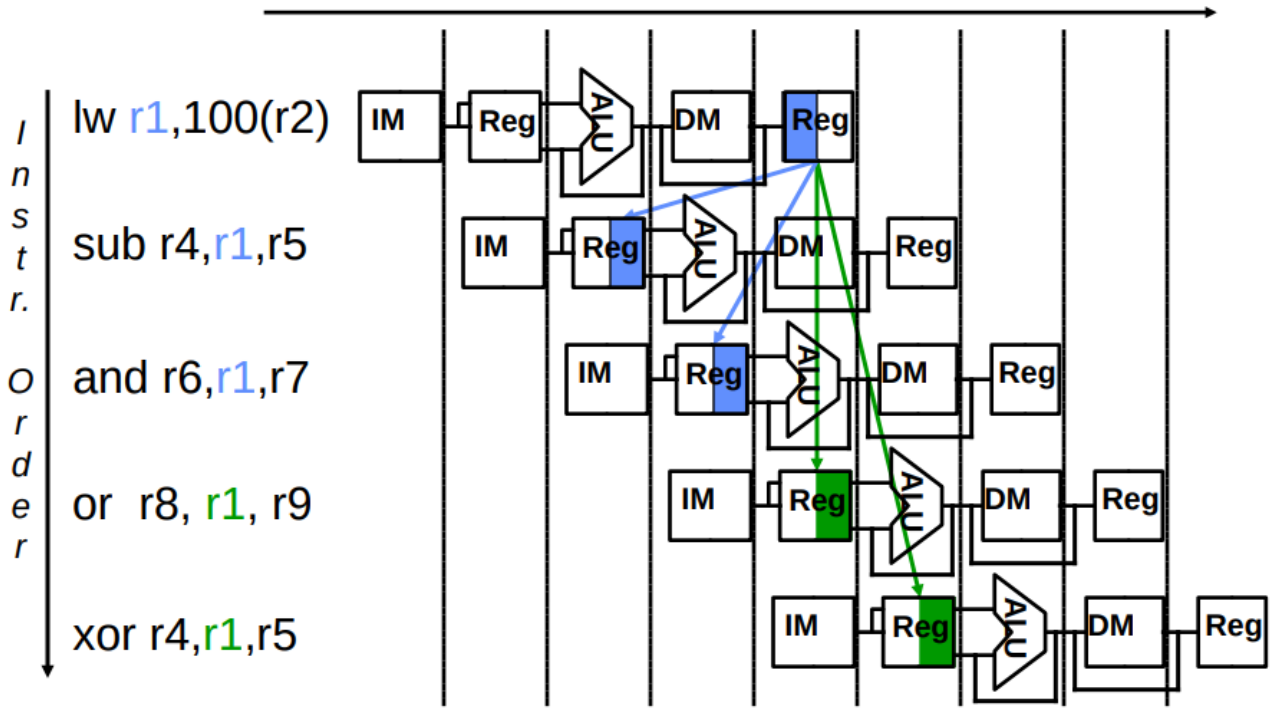
2 – Accesso al Register File potrebbe essere una Structural Hazard

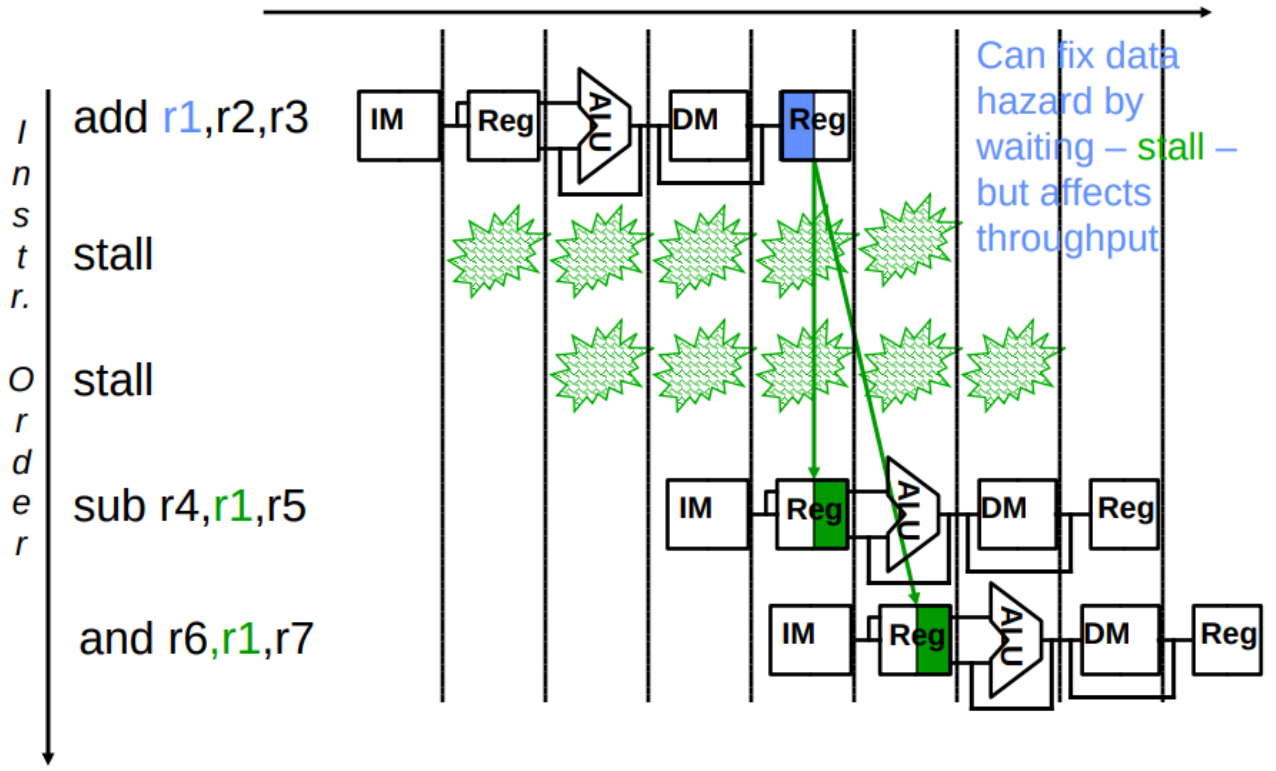


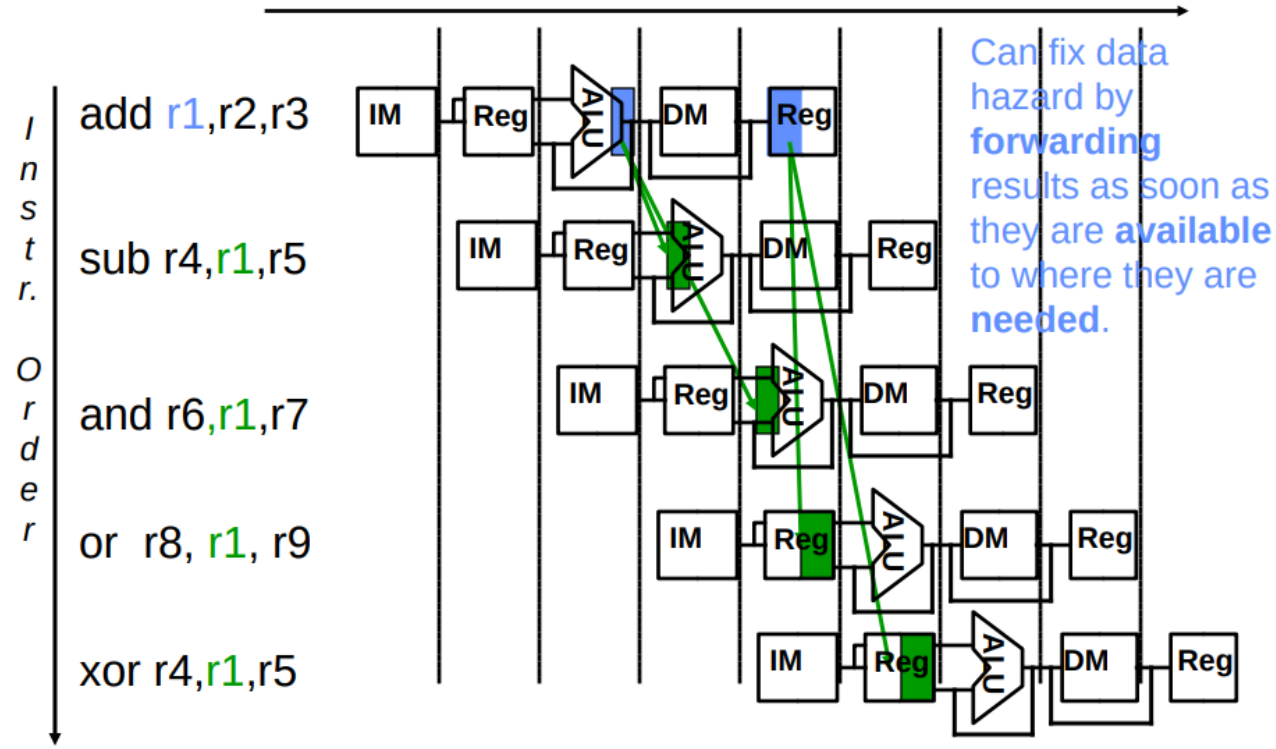


L'utilizzo dei registri e i caricamenti possono causare data hazard


Instr. Order
add r1,r2,r3
sub r4,r1,r5
and r6,r1,r7
or r8, r1, r9
xor r4,r1,r5
IM
Reg
ALU
DM
Reg


Instr. Order
lw r1,100(r2)
sub r4,r1,r5
and r6,r1,r7
or r8, r1, r9
xor r4,r1,r5
IM
Reg
ALU
DM
Reg



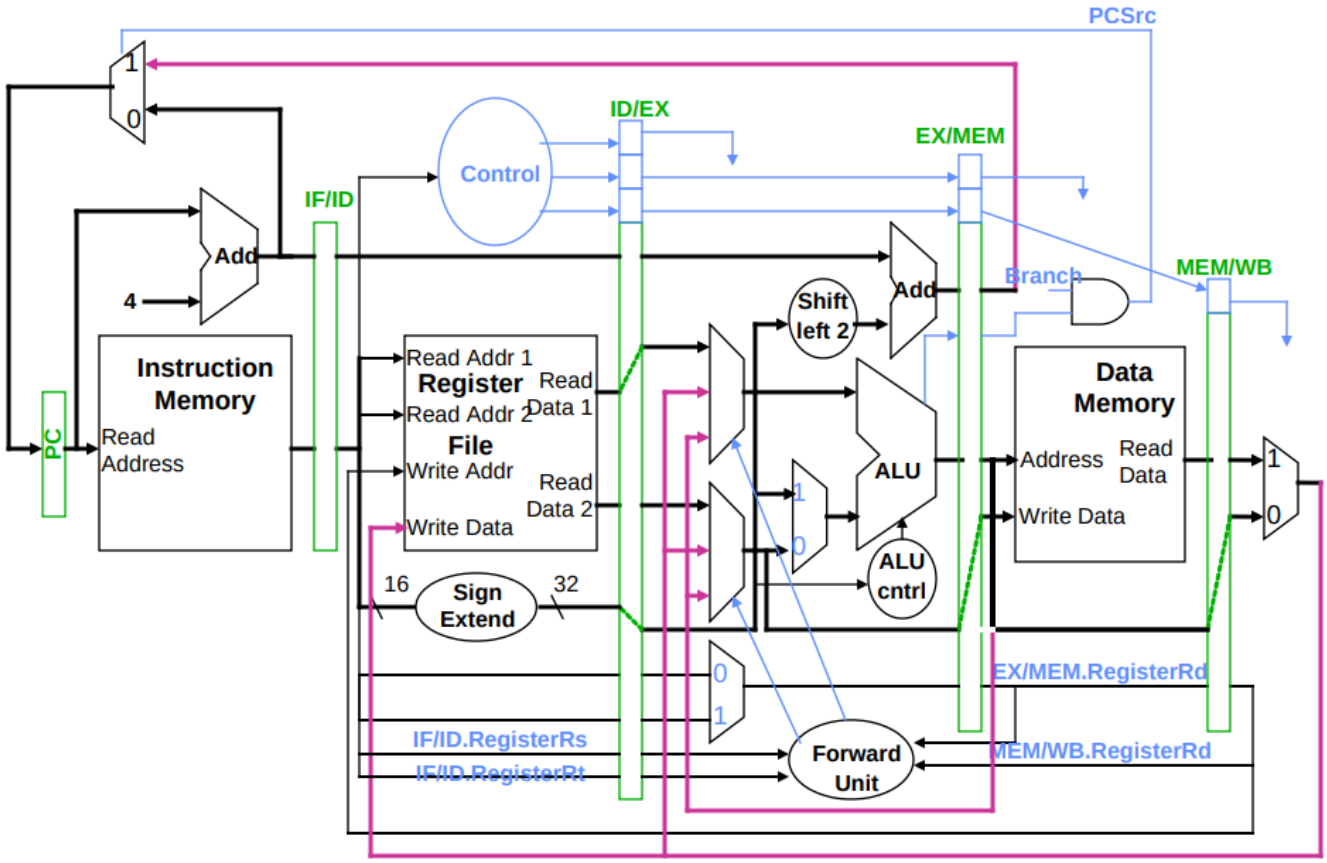


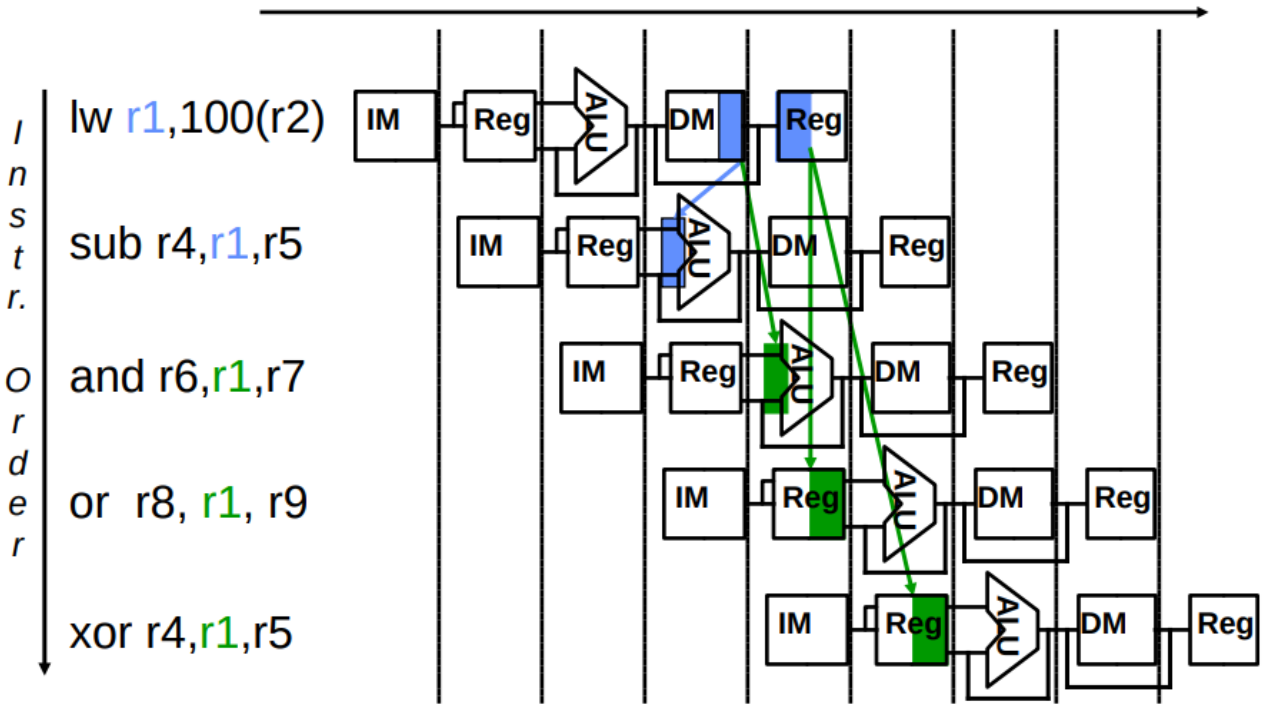
6) Data forwarding: definizione

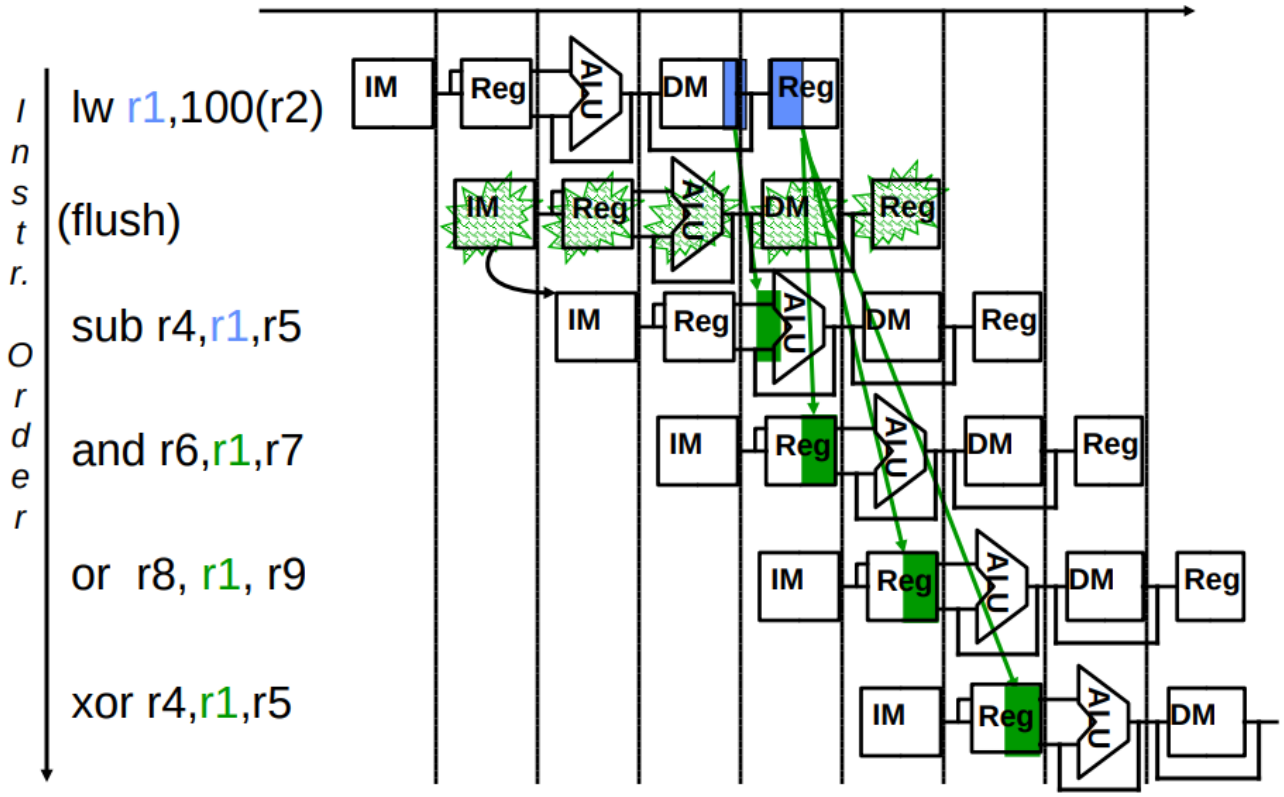
Il data forwarding consente al processore di eseguire alcune istruzioni in parallelo inviando i risultati di un'istruzione immediatamente a quella che dovrà continuarne l'elaborazione senza attendere che quest'ultima sia già arrivata a uno stadio (fase) di pipeline in cui deve essere eseguita. Con l'inoltro può

funzionare a pieno regime anche in presenza di dipendenze dei dati.

Hardware Data Forwarding







7) Control Hazard: quali sono le possibili soluzioni di hazard?

Le possibili soluzioni di control hazard sono:

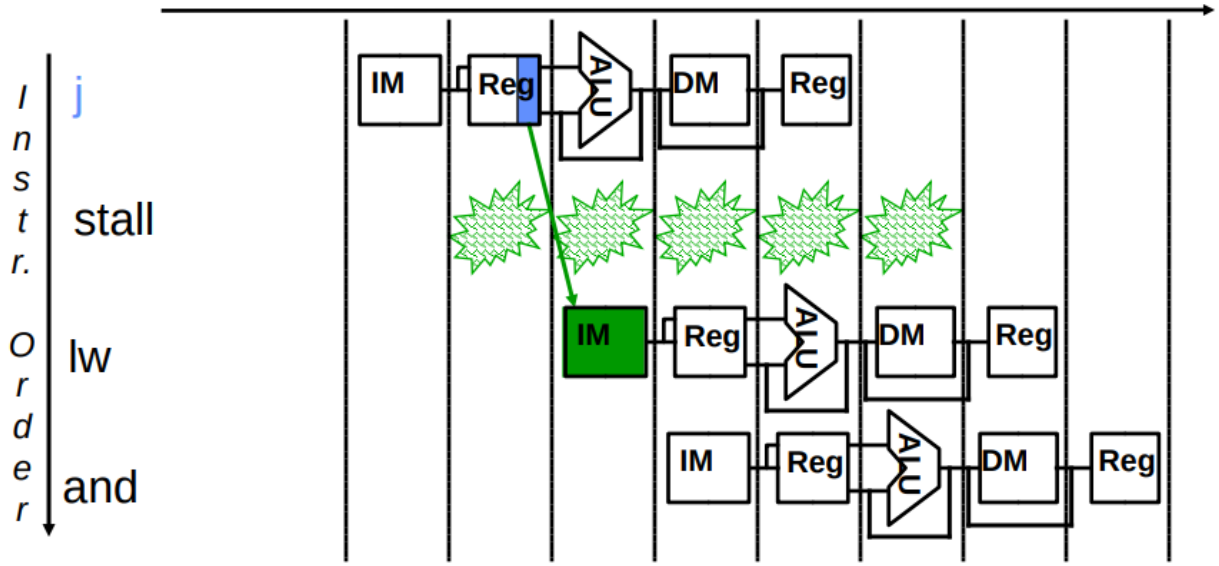
→ stallo;

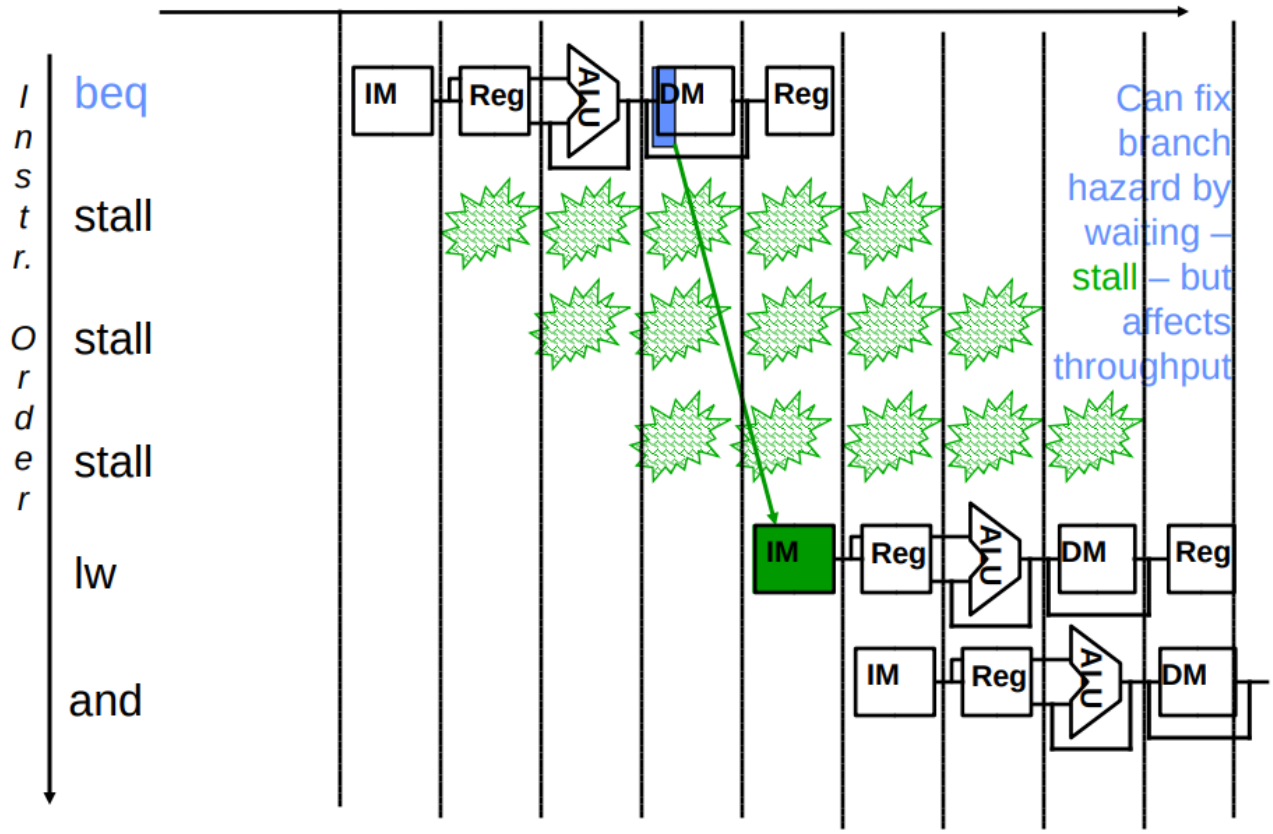
→ spostamento del punto di decisione all'inizio della pipeline;

→ previsione;

→ decisione di ritardo.

Istruzioni di jump e branch





8) Si commenti la correttezza della seguente affermazione: "Il pipeling consente ad un maggiore numero di istruzioni di raggiungere il completamento, per unità di tempo, ma non diminuisce il tempo perché una istruzione venga completata"

È una tecnologia utilizzata nell'architettura hardware per incrementare il numero di istruzioni eseguite nello stesso momento. Ad ogni ciclo di clock viene eseguita una operazione, ma parallelizzando la computazione di più operazioni si ottiene un throughput maggiore, quindi il tempo di esecuzione della singola istruzione non cambia.

9) Si chiarisca se una lunga sequenza di istruzioni, priva di salti condizionati, dia luogo ad una migliore o peggiore performance in una CPU con pipelining

La mancanza di salti condizionati ha una performance migliore e permette un throughput maggiore, perchè ogni volta che in un sistema con pipeline si verifica un salto la pipeline va svuotata per permettere l'acquisizione dei nuovi dati contenuti nel blocco a cui si è saltato. Questo spreca dei cicli di clock e diminuisce il throughput dei dati.

10) Si descriva cosa si intende per pipeline stall e perché sia comunemente chiamata "bubble"

Lo stallo avviene quando non è possibile in un'architettura con pipeline eseguire

l'istruzione immediatamente successiva. Vengono inserite delle "bolle" nella pipeline, ovvero si blocca il flusso di istruzioni finché il conflitto non viene risolto.

11) Si descrivano degli esempi di "data hazards" e si accenni a come questi si potrebbero affrontare, potendo intervenire sull'hardware della CPU

La dipendenza dai dati (data hazards) è una situazione che si verifica quando un'istruzione richiede per essere eseguita dei risultati di operazioni che non sono ancora terminate. Degli esempi sono: RAW - Read After Write, WAR - Write After Read, WAW - Write After Write. Una possibile soluzione a queste dipendenze è di assegnarle al compilatore che può inserire un numero adeguato di istruzioni NOP per risolvere le dipendenze.

12) Si descriva cosa sia la branch prediction e in quale modo possa effettivamente aumentare le performance?

La predizione delle diramazioni (branch prediction) è il compito della BPU (Branch Prediction Unit), una componente della CPU che cerca di prevedere l'esito di un'operazione su cui si basa l'accettazione di una istruzione di salto condizionato, evitando rallentamenti che possono essere molto evidenti in una architettura con pipeline. L'importanza di questa operazione è evidente soprattutto per i microprocessori moderni con lunghe pipeline, che per ogni errore di previsione devono sprecare molti cicli di clock di lavoro prezioso.