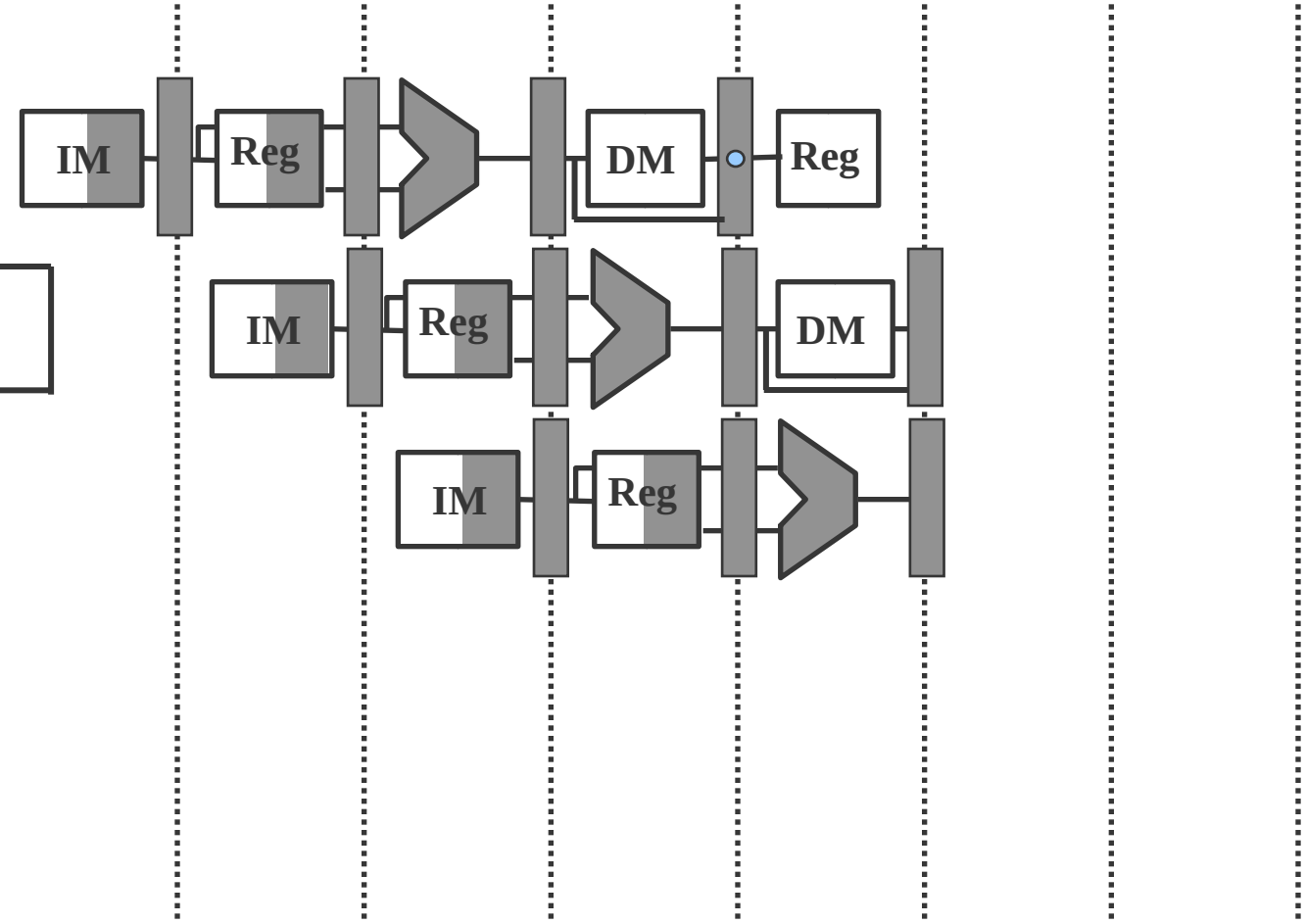
Criticità del Controllo

Criticità di salto

40 beq $1,$3,28

44 and $12,$2,$5

72 lw $4,50($7)



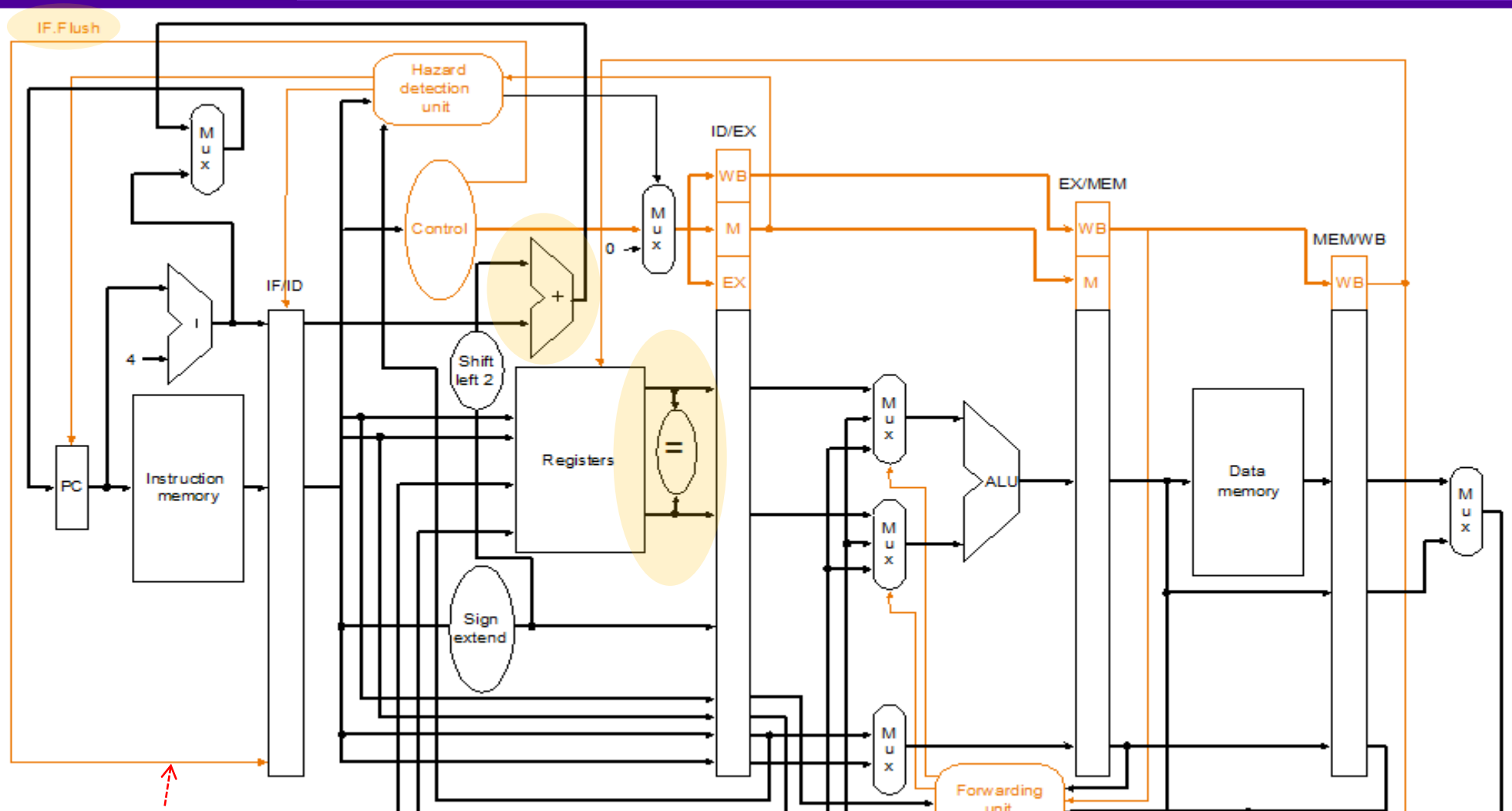
Soluzioni

- Inserire dopo il salto un'istruzione che viene comunque eseguita
  - istruzione da eseguire sia che il salto avvenga che in caso contrario
  - istruzione che quando viene eseguita comunque non "fa danni"
  - approccio sempre meno usato
- Assumere che il salto non avvenga
  - possibile ottimizzazione
- Predire l'esito del salto in modo dinamico

Assumere che salto non avvenga

- Se al contrario il salto avviene, le istruzioni prelevate e decodificate devono essere scartate
  - trasformare le istruzioni in IF, ID, ed EX in nop
- Ottimizzazione
  - anticipare la decisione del salto nella fase ID
    - usare XOR e NOR per decidere se i due registri sono uguali
    - anticipo il calcolo dell'indirizzo di destinazione del salto
  - solo istruzione in IF deve essere resa nop (un solo stallo)
    - Azzero la parte di IF/ID che contiene l'istruzione (segnale IF.Flush)

Cammino dei dati e controllo (parziali)



Azzera la parte del registro che contiene l'istruzione → si trasforma l'istruzione appena caricata in una *nop*;

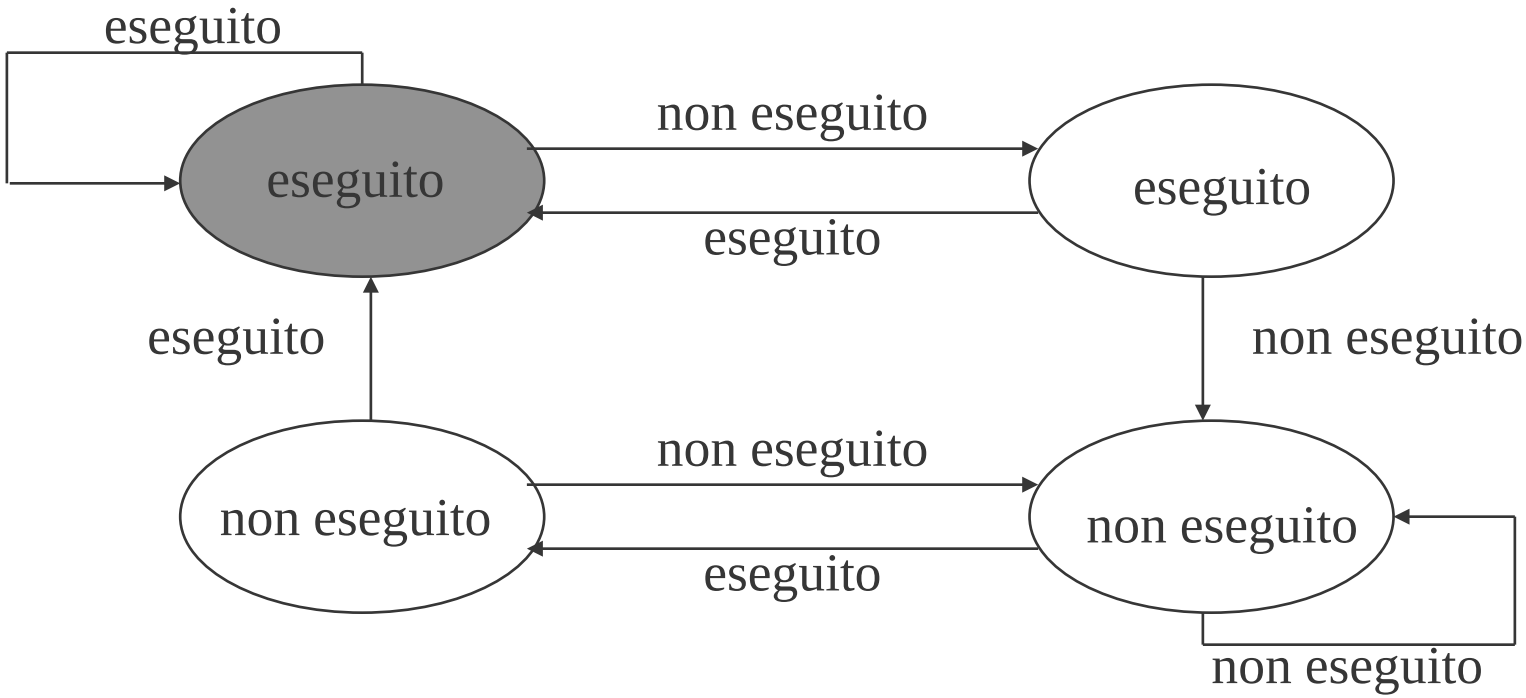
Predizione dinamica

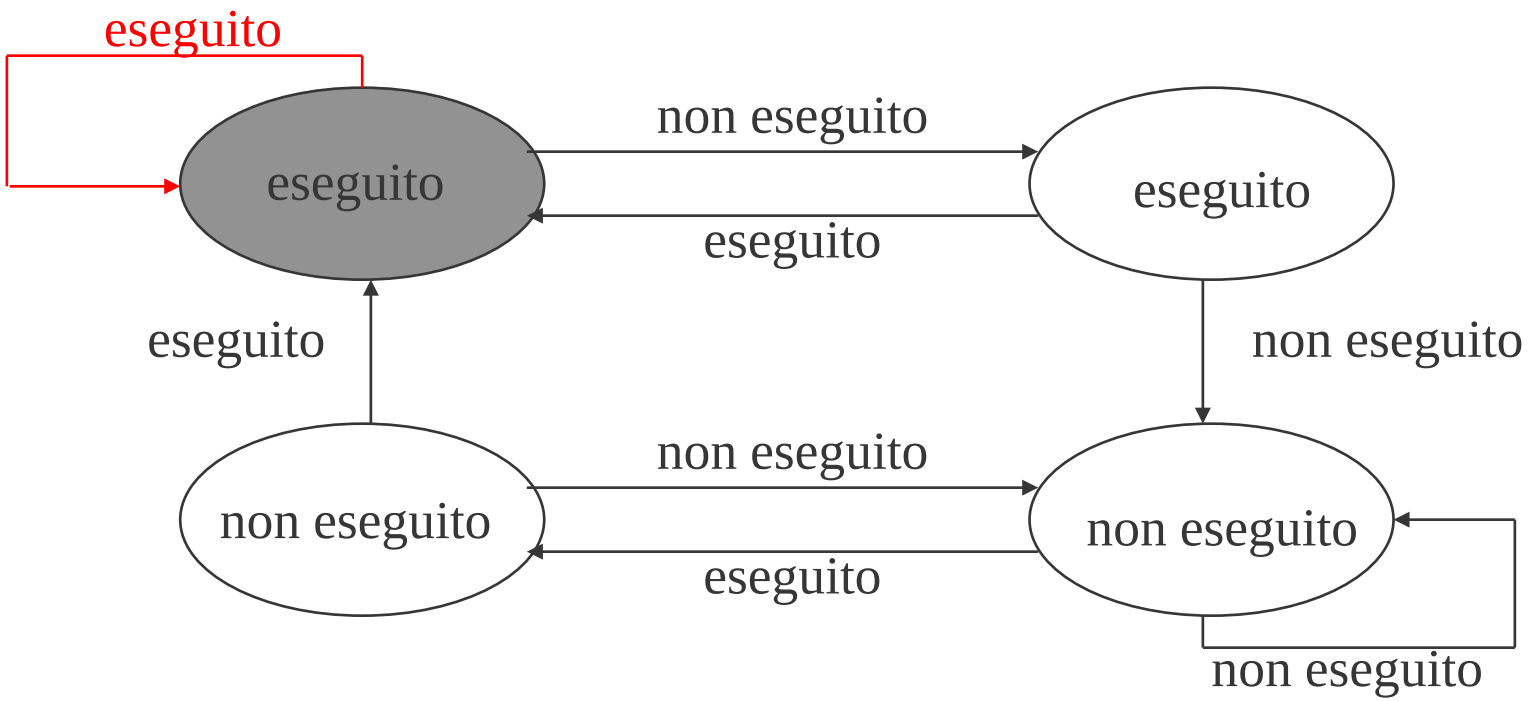
- Analizzare l'indirizzo dell'istruzione di salto per determinare se l'ultima volta che era stata eseguita il salto era avvenuto
- Tabella di storia del salto
  - piccola memoria indirizzata da porzione di bit meno significativi dell'indirizzo dell'istruzione di salto
  - ogni elemento contiene un bit
    - 1: salto eseguito in precedenza
    - 0: salto non eseguito in precedenza

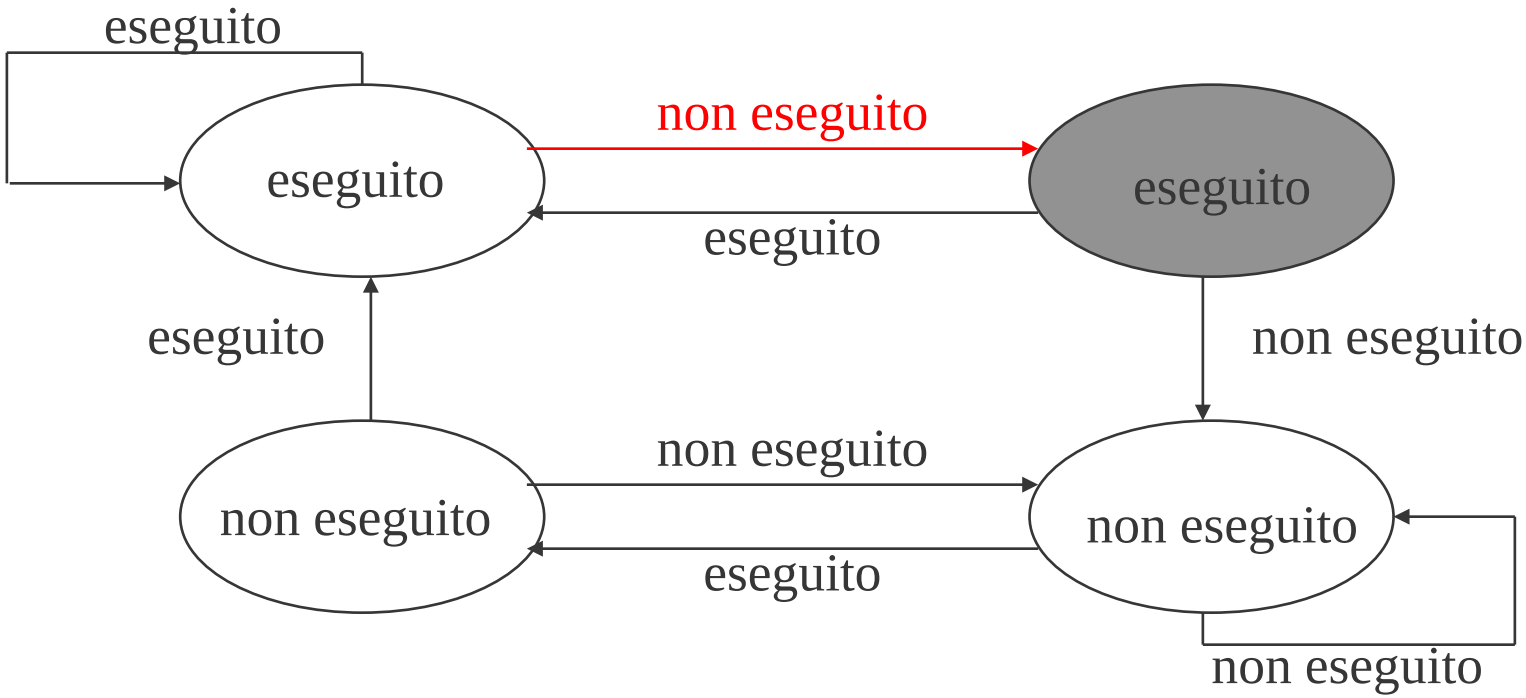
  Informazione non necessariamente corretta...
    - potrebbe provenire da un'altra istruzione di salto il cui indirizzo ha gli stessi bit meno significativi

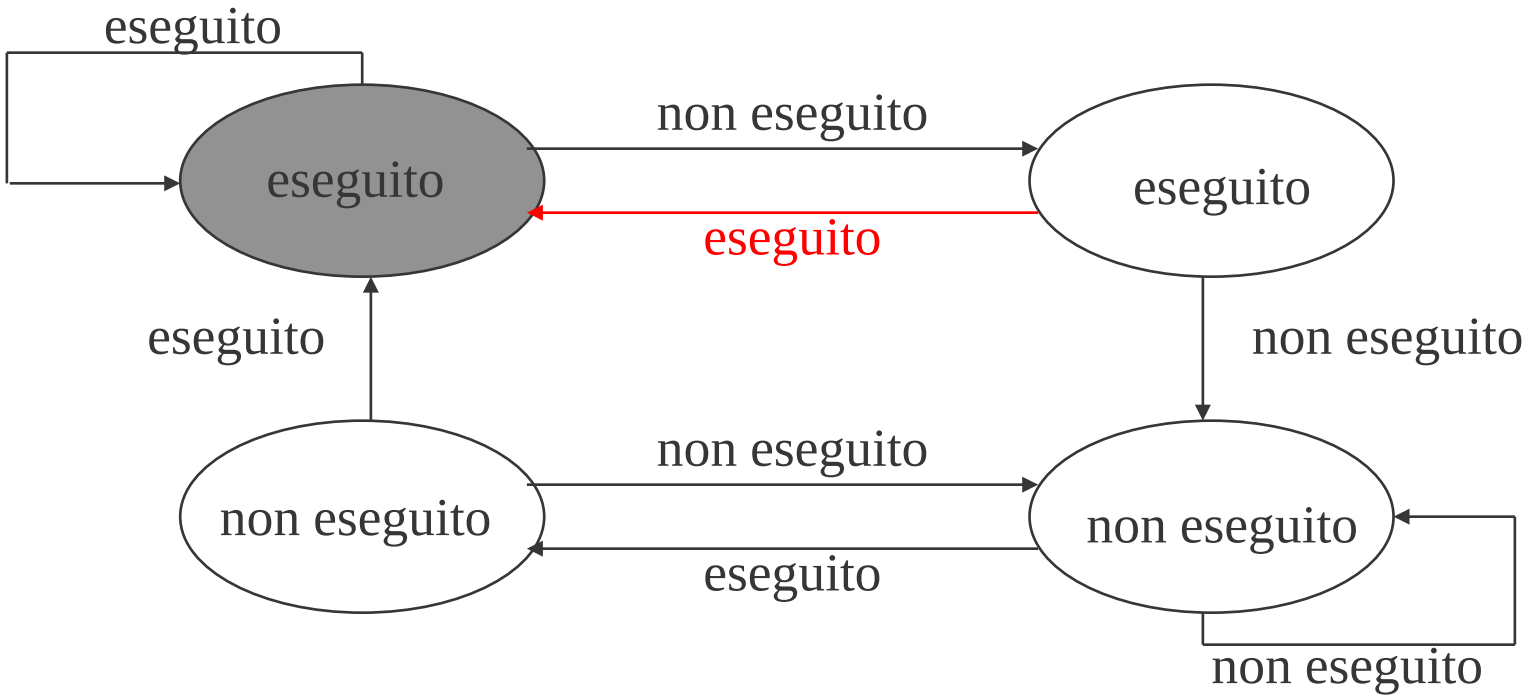
Problema

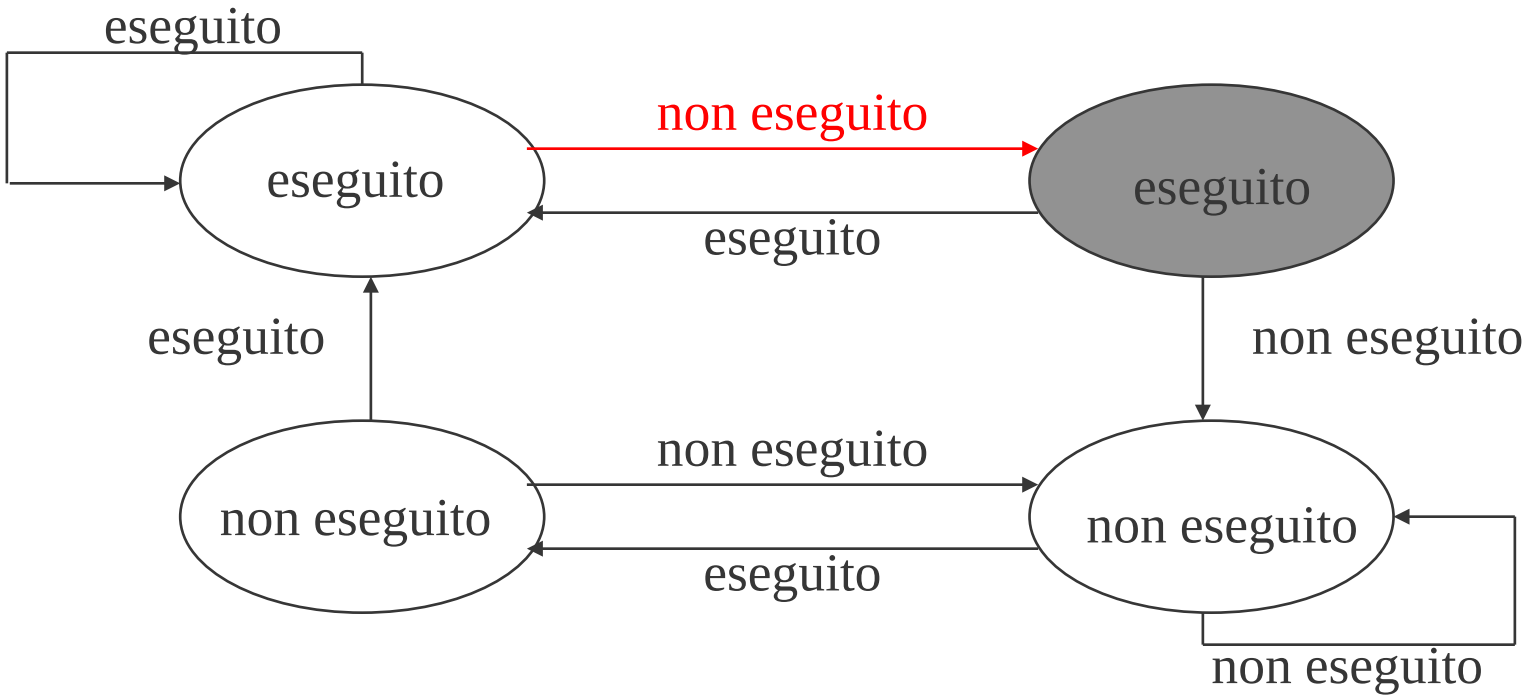
- Salti "regolari" (ovvero eseguiti o non eseguiti molto spesso) vengono mal previsti due volte anzichè una volta
- Esempio:
  - ciclo di dieci salti di cui nove eseguiti ed uno no
  - mal previsto il primo salto
    - la volta precedente non eseguito
  - mal previsto il decimo salto
    - la volta precedente eseguito
  - Accuratezza di previsione pari all'80%
- Soluzione:
  - usare due bit invece di uno per rappresentare quattro stati
    - previsione deve sbagliare due volte prima di cambiare

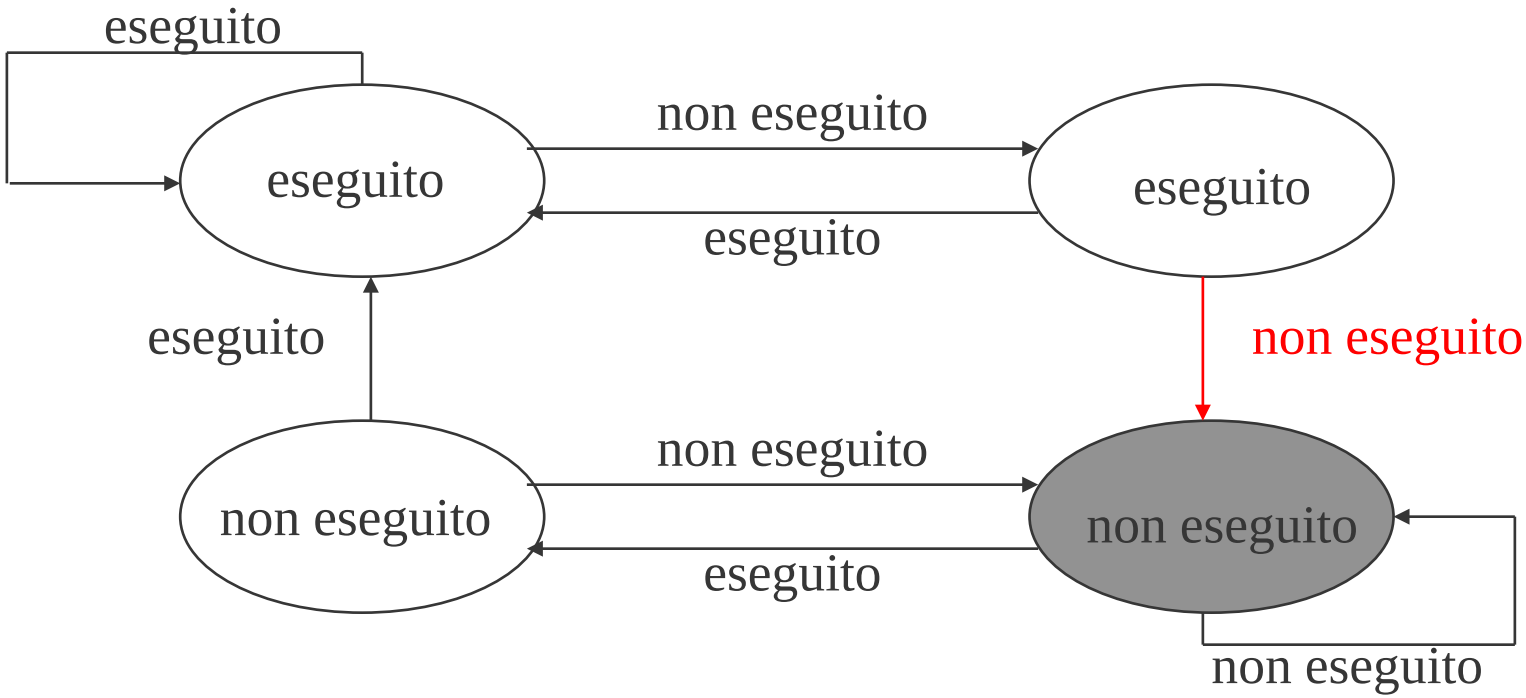

eseguito
eseguito
non eseguito
eseguito
eseguito
non eseguito
non eseguito
non eseguito
eseguito
non eseguito
non eseguito


eseguito
eseguito
non eseguito
eseguito
eseguito
non eseguito
non eseguito
non eseguito
eseguito
non eseguito
non eseguito


eseguito
eseguito
non eseguito
eseguito
eseguito
eseguito
non eseguito
non eseguito
non eseguito
eseguito
non eseguito
non eseguito


eseguito
eseguito
non eseguito
eseguito
eseguito
eseguito
non eseguito
non eseguito
non eseguito
eseguito
non eseguito
non eseguito


eseguito
eseguito
non eseguito
eseguito
eseguito
eseguito
non eseguito
non eseguito
non eseguito
eseguito
non eseguito
non eseguito


eseguito
eseguito
non eseguito
eseguito
eseguito
eseguito
non eseguito
non eseguito
non eseguito
eseguito
non eseguito
non eseguito

Eccezioni

- Eccezioni: evento (interno al processore) non previsto che interrompe l'esecuzione di un programma
- Vengono trattate come se si trattasse di un hazard sul controllo.
- Esempio: overflow aritmetico in una add
  - In fase di EX rilevo l'overflow
  - Elimino dalla pipeline tutte le istruzioni che seguono la add
  - Prelevo le istruzioni dal nuovo indirizzo, che è quello di risposta all'eccezione

Eccezioni - 1

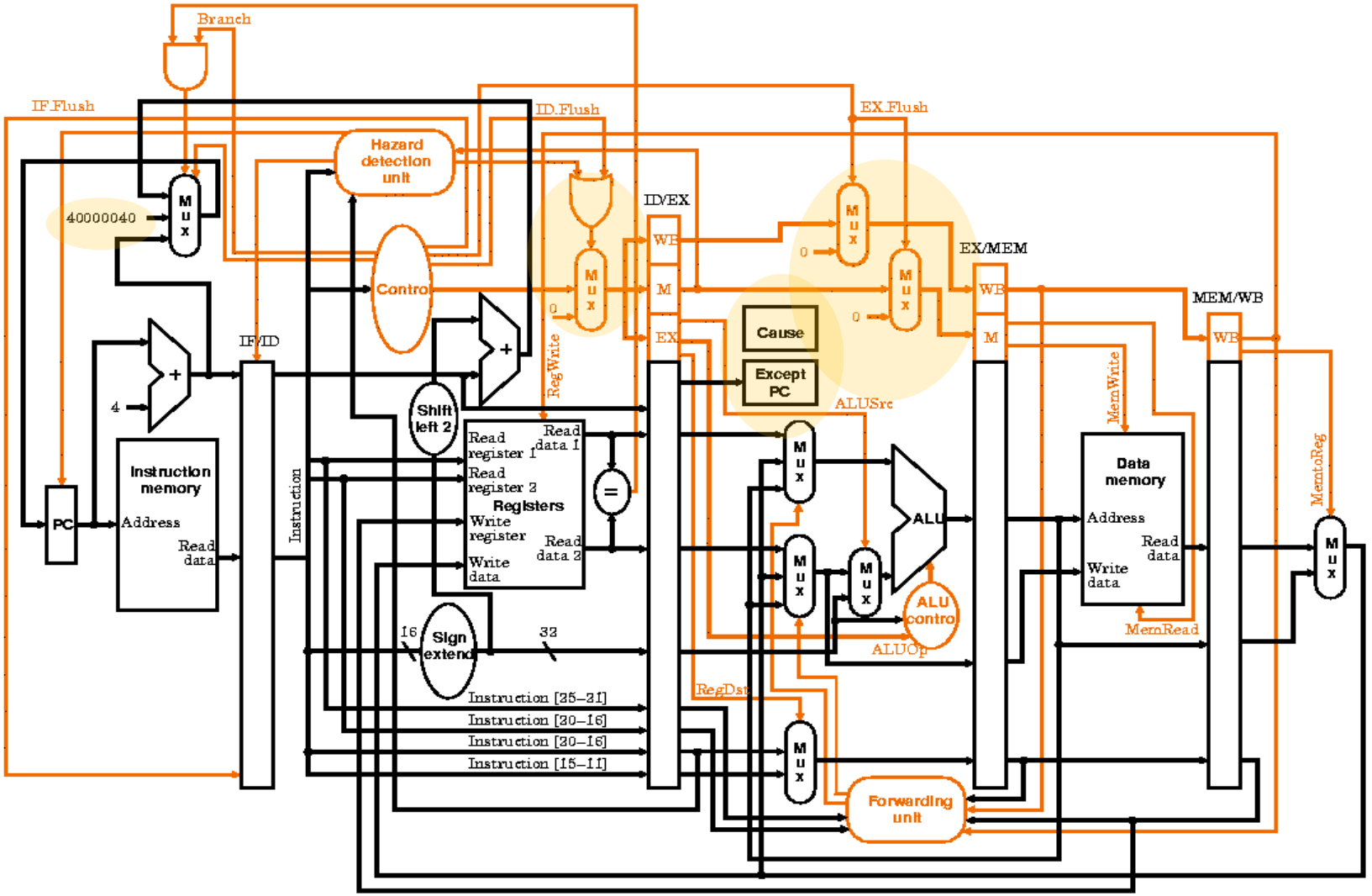
Realizzazione:

- Rendere nop le istruzioni in IF, ID ed EX
  - Istruzione in IF
    - come per il salto (azzero la parte del registro IF/ID che contiene l'istruzione – IF.Flush)
  - Istruzione in ID
    - metto in OR il segnale proveniente da URC e quello proveniente dall'unità di rilevazione dell'eccezione (ID.Flush)
  - Istruzione in EX
    - nuovo segnale (EX.Flush) per gestire un multiplexer con input 0

Eccezioni - 2

- Aggiungere nuovo input al multiplexer che governa l'input di PC con indirizzo la procedura di gestione delle eccezioni (es, 0x4000 0040)
- Disabilitare fase WB dell'istruzione che ha generato eccezione
  - in tal modo, gli operandi sorgente sono ancora disponibili
- Salvare la causa dell'eccezione nel registro "Causa" e l'indirizzo di istruzione che ha generato eccezione in EPC (in realtà, PC+4)

Cammino dei dati + ctrl. finale



Migliorare la pipeline

- Superpipeline
  - Pipeline più lunga
- Pipeline con parallelizzazione dell'esecuzione
  - Replicare le componenti interne del calcolatore in modo da caricare più istruzioni nello stesso stadio della pipeline

Parallelizzazione statica

- Le decisioni vengono prese staticamente (durante la compilazione)

Parallelizzazione dinamica

- Le decisioni vengono prese dinamicamente (durante l'esecuzione)

Superpipeline

- Si aumenta la profondità di pipeline per aumentare la velocità di clock dei singoli stadi e ridurne quindi la latenza.
  - Ad esempio, se lo stadio EX impiega tre volte il tempo che sarebbe richiesto per l'esecuzione di ogni altro stadio, potremmo suddividere lo stadio EX in tre stadi separati, diminuendo quindi il periodo di clock e riducendo quindi l'ammontare di tempo "sprecato" negli altri stadi più veloci.
- Il problema è quello di capire come è possibile suddividere i vari stadi, e quello di progettare una unità di controllo più complessa per gestire la superpipeline e prevenire tutti i possibili azzardi.
- Alcuni microprocessori hanno 20 o più stadi di pipeline (es: Intel Pentium IV)
  - Intel (Core i7): pipeline a 14 stadi

Parallelizzazione statica (esempio del MIPS)

- Eseguire due istruzioni per stadio della pipeline
  - Due cammini di esecuzione (vie), indipendenti fra loro, che quindi vanno in parallelo:
    - Può eseguire istruzioni di **Tipo R** oppure di **salto condizionato**
    - Può eseguire istruzioni di **accesso a memoria** (load o store)
  - → Caricare e decodificare 64 bit di istruzione (istruzioni eseguite sempre in coppia, eventualmente con *nop*)

Tipo R o salto	IF	ID	EX	MEM	WB		
Load o store	IF	ID	EX	MEM	WB		
Tipo R o salto		IF	ID	EX	MEM	WB	
Load o store		IF	ID	EX	MEM	WB	
Tipo R o salto			IF	ID	EX	MEM	WB
Load o store			IF	ID	EX	MEM	WB

Requisiti hardware

- Quattro porte addizionali di input al banco dei registri
  - due nuovi indirizzi di lettura
  - un nuovo indirizzo di scrittura
  - un nuovo dato di scrittura
- Due porte addizionali di uscita dal banco dei registri
- Un sommatore addizionale
  - Per calcolare l'indirizzo di memoria (la ALU esegue l'operazione aritmetico-logica)

Esempio

```
Loop:   lw    $t0,  0($s1)
        addu  $t0,  $t0,  $s2
        sw    $t0,  0($s1)
        addi  $s1,  $s1,  -4
        bne   $s1,  $zero, Loop
```

- Possibile soluzione (CPI = 0.8):

Loop:	*nop*	lw $t0, 0($s1)
	addi $s1, $s1, -4	*nop*
	addu $t0, $t0, $s2	*nop*
	bne $s1, $zero, Loop	sw $t0, 0($s1)

Pipeline con parallelizzazione dinamica (processori superscalari)

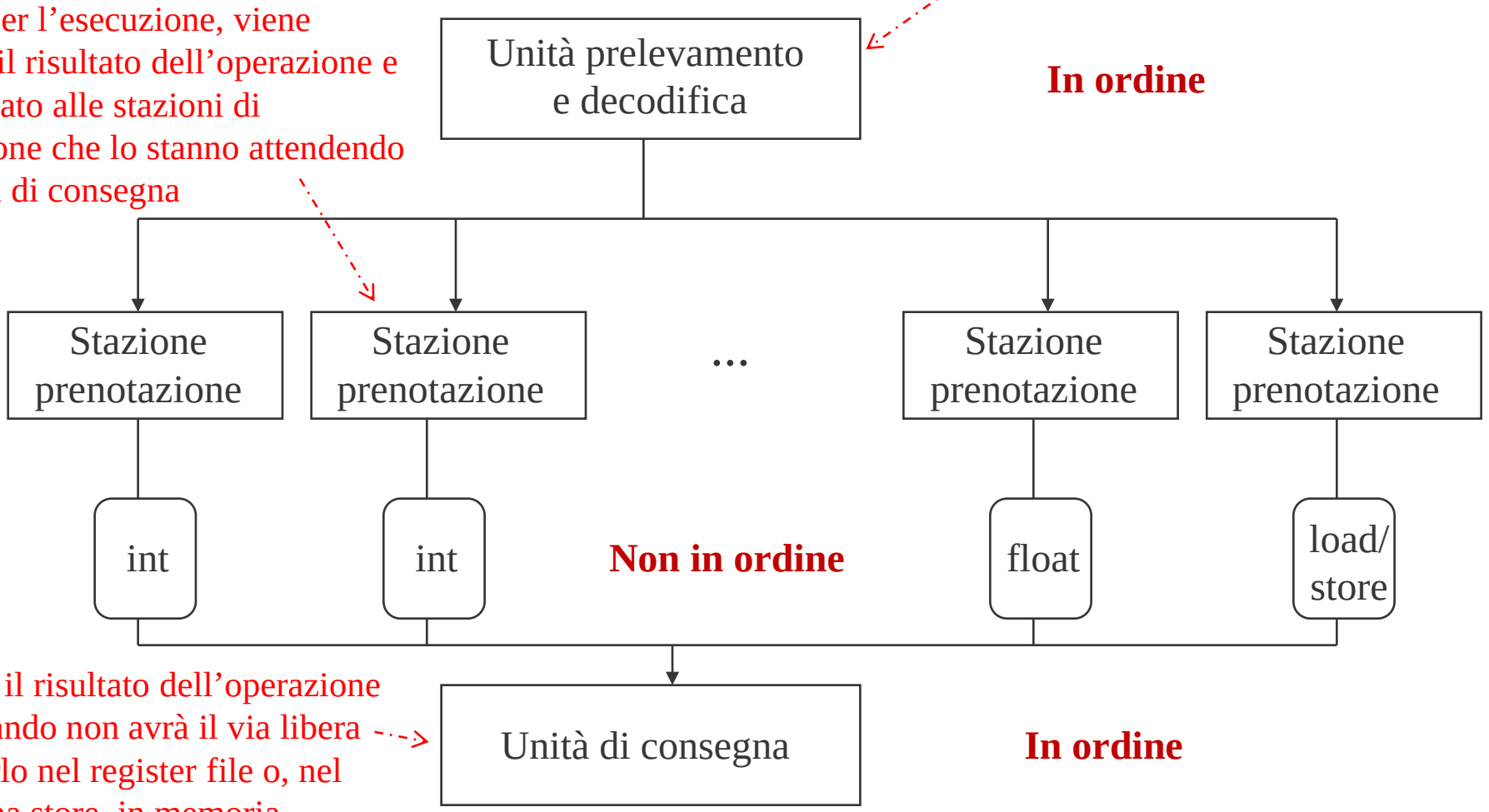
- Si selezionano le istruzioni da eseguire in parallelo durante l'esecuzione stessa.
- Versione più semplice:
  - le istruzioni sono eseguite in ordine, e
  - il processore decide se nessuna, una o più istruzioni possono essere avviate all'esecuzione nello stesso ciclo.
- Versione estesa:
  - si include un riordinamento dinamico delle istruzioni, cercando anche di evitare hazard e stalli.

Riordinamento dinamico del codice

- Tre componenti principali della pipeline:
  - Unità di prelevamento e decodifica istruzioni
  - Più unità funzionali di esecuzione (10 o più)
    - dotate di un buffer, detto stazione di prenotazione, che memorizza operandi ed operazione
  - Unità di consegna risultati
- Preserva l'ordine delle istruzioni

Non appena la stazione di prenotazione contiene tutti gli operandi richiesti e l'unità funzionale è pronta per l'esecuzione, viene calcolato il risultato dell'operazione e viene inviato alle stazioni di prenotazione che lo stanno attendendo e all'unità di consegna

Preleva, decodifica e invia ciascuna istruzione alla corrispondente unità funzionale di esecuzione



Conserva il risultato dell'operazione fino a quando non avrà il via libera per salvarlo nel register file o, nel caso di una store, in memoria