Lezione 13/11/2023

Introduzione

La **speculazione hardware** è una tecnica per ridurre le *control dependences*(quelle dei branch) in un processore che implementa il **dynamic scheduling.**
Combina tre idee:

- dynamic branch prediction
- dynamic scheduling
- speculation

Architettura

Si estende l'architettura di **Tomasulo** in modo da supportare la speculazione.
Nella fase di esecuzione ci sono due fasi:

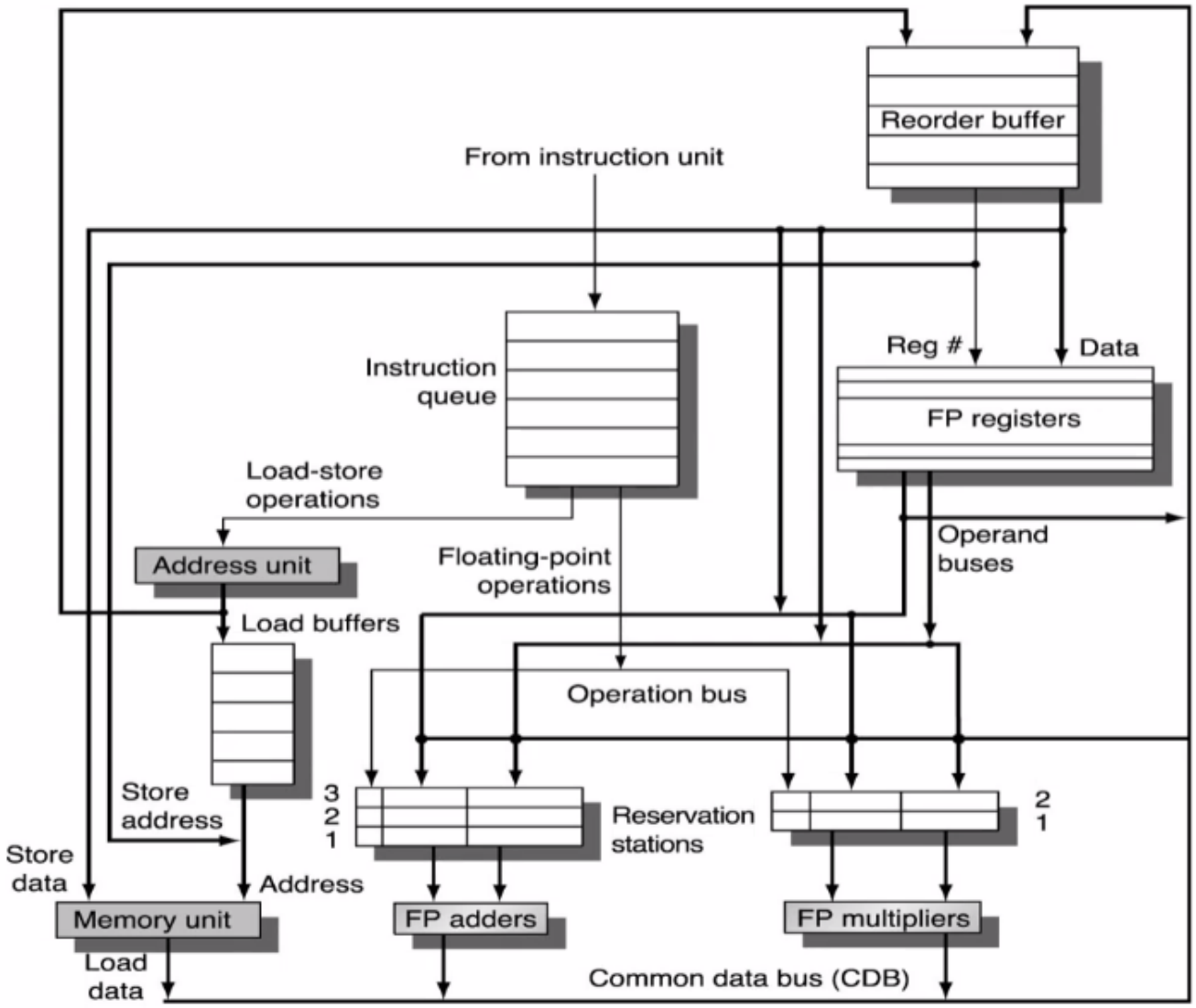
- computazione di risultati e il suo inoltro alle altre istruzioni
- *update* del register file e memory che viene eseguita soltanto quando non è più speculativo(**instruction commit**).
  La speculazione dunque avviene nella prima fase dell'esecuzione, e se va a buon fine viene *completata* nella fase di update

ReOrder Buffer(ROB)

Mantiene le informazioni sull'ordine di esecuzione del programma originale.
E' simile ad una *reservation station*
Contiene ulteriori registri virtuali e integra lo **store buffer** originale dell'architettura di Tomasulo.
Le istruzioni che non sono ancora state *committate* vengono mandate qua.
Viene implementato come un **buffer circolare**
Dal **ROB**,se le istruzioni non sono ancora state committate, è possibile leggere i dati prodotti(come se fosse un cammino di *forwarding*).
Ogni entry ha:

- **Instruction type:** branch/store/register
- **Destination:** indirizzo di memoria o registro
- **Value:** valore quando l'istruzione è completata ma non ancora committata
- **Ready:**
  - **2:** committata
  - **1:** completata

- **0:** non completata



Le istruzioni, quando vengono *dispatchate* alla reservation stations, vengono anche inserite nel **Reorder Buffer**

Instruction Execution Steps(di nuovo)

Issue(o Dispatch)

L'istruzione viene estratta dalla *queue* se c'è una reservation station vuota **e** uno slot vuoto nel **ROB**.
I suoi operandi vengono inviati alle reservation station se sono nel register file o nel **ROB**. Gli viene inviato anche il numero della entry del **ROB** come tag

Execution

Appena tutti gli operandi sono disponibile l'istruzione viene eseguita. Questo fa sì che non capitino RAW hazard. Gli operandi possono essere presi anche dal CDB

Write Result

Appena il risultato è disponibile, il risultato viene messo sul **CDB** e mandato nel **ROB**(nel field *value*). Le reservation stations leggono dal CDB in caso stiano aspettando degli operandi. L'entry della reservation station viene marchiata come libera quando entra in questo stadio

Commit(o Completion)

Il **ROB** viene ordinato secondo l'ordine originale.
Appena un'istruzione raggiunge la *testa* del buffer:

- in caso di misprediction branch, il buffer viene flushato
- altrimenti il risultato viene scritto nel register file(o memoria in caso di store).
- in entrambi i casi, la entry del **ROB** viene marchiata come libera

Hazard handling

WAW e WAR hazards

Non possono capitare grazie al dynamic renaming e l'update della memoria accade in ordine

RAW hazards

Non capitano grazie all'enforcing dell'ordine del programma mentre si calcolano gli effective address di tutte le istruzioni prima e anche grazie anche al fatto che una load non può andare nel suo secondo step se esiste una entry **store** nella ROB con la stessa destinazione della **load**

Exception Handling

Le eccezioni non vengono eseguite subito ma salvate nel **ROB**. Quando l'istruzione viene committata, l'eccezione viene eseguita e le entry del **ROB** subito dopo flushate

Speculating expensive events

Quando capita un'operazione costosa a livello di tempo capita speculativamente, alcuni processori aspettano che non sia più speculativo ovvero: invece di rischiare di eseguire una operazione sbagliata, il processore si accerta che non sia speculazione.