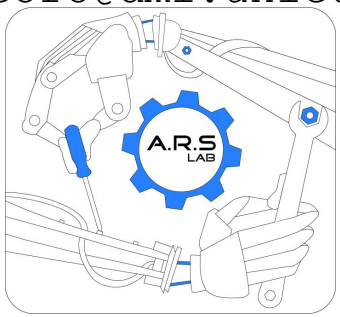
Pipelining

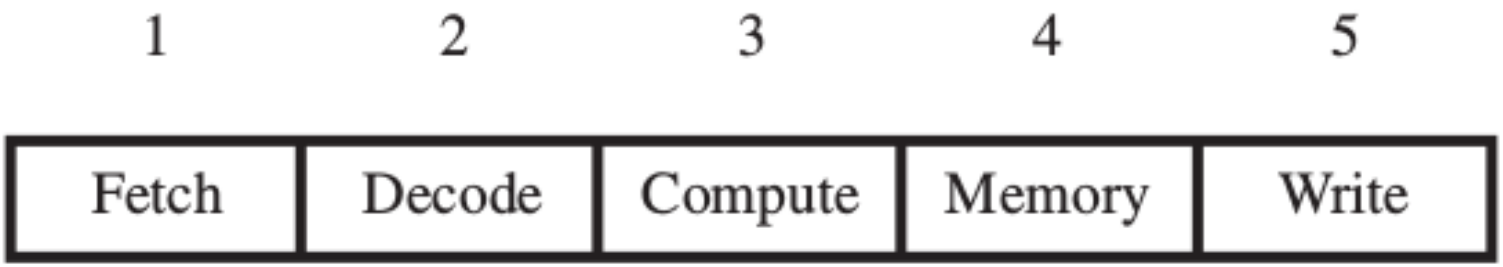
Corrado Santoro

Dipartimento di Matematica e Informatica
santoro@dmi.unict.it



Corso di Architettura degli Elaboratori

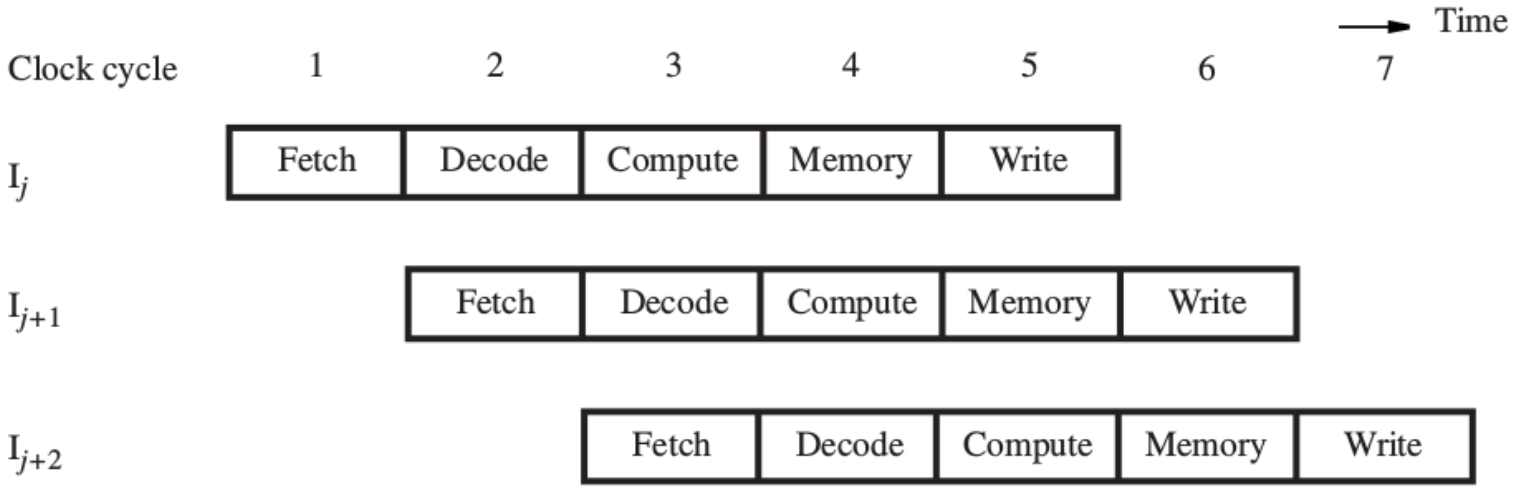
Datapath e Stages



Stages

- **Fetch** prelevamento istruzione
- **Decode** selezione registri sorgente
- **Compute** ALU
- **Memory** accesso alla memoria
- **Writing** scrittura registro di destinazione

Datapath, Stages e Pipelining



Pipelining

- Mentre lo stadio k sta eseguendo l'istruzione I_j, lo stadio precedente, $k-1$, è libero
- Lo stadio $k-1$ può pertanto eseguire l'istruzine I_{j+1}
- **Senza Pipelining** Ogni istruzione richiede **almeno** 5 cicli di clock
- **Con Pipelining** Ogni istruzione richiede (almeno) **un solo** ciclo di clock

E' sempre possibile il Pipelining?

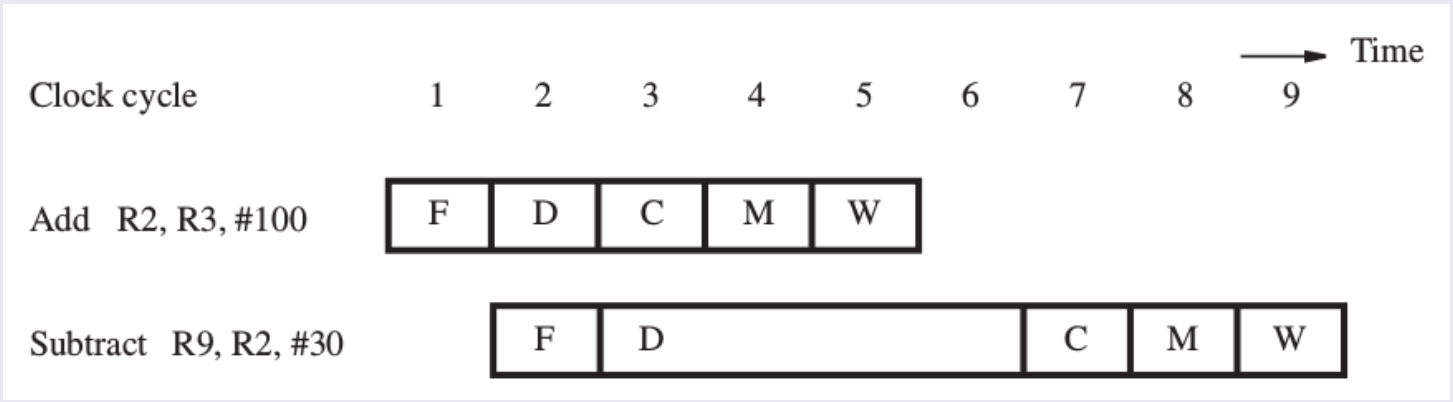
Pipelining Stall

- NO!!
- Esistono casi in cui due istruzioni I_j e I_{j+1} non possono essere eseguite in pipeline
- Essi riguardano tutte le situazioni in cui vi è una *data dependency* tra le due istruzioni
- In questi casi occorre **bloccare** (mettere in **stallo**) la pipeline
- L'istruzione I_{j+1} deve attendere dunque il completamento fino allo stage 5 dell'istruzione I_j
- Le condizioni che provocano uno stallo della pipeline sono dette **hazard**

Data Dependency Hazard

Esempio

```
ADD R2, R3 #100
SUB R9, R2 #30
```

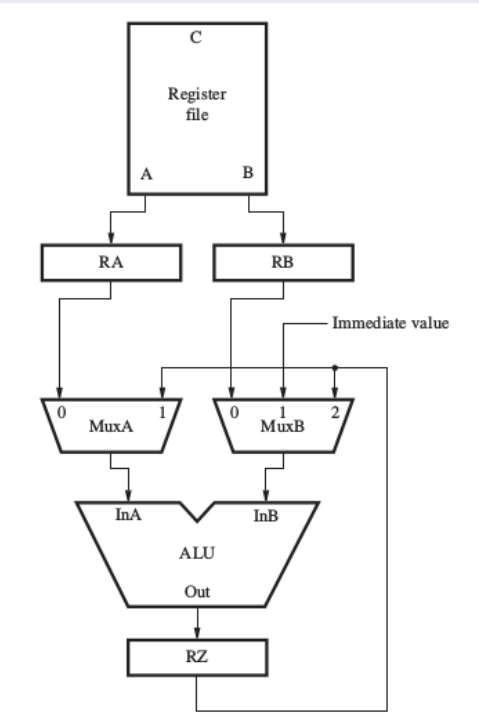


La **SUB** deve attendere il completamento della **ADD**

Soluzione: Operand Forwarding

```
ADD R2, R3 #100
SUB R9, R2 #30
```

Il datapath viene modoficato in modo da far sì che l'output dello stage 3 possa essere (anche) inviato all'input della ALU

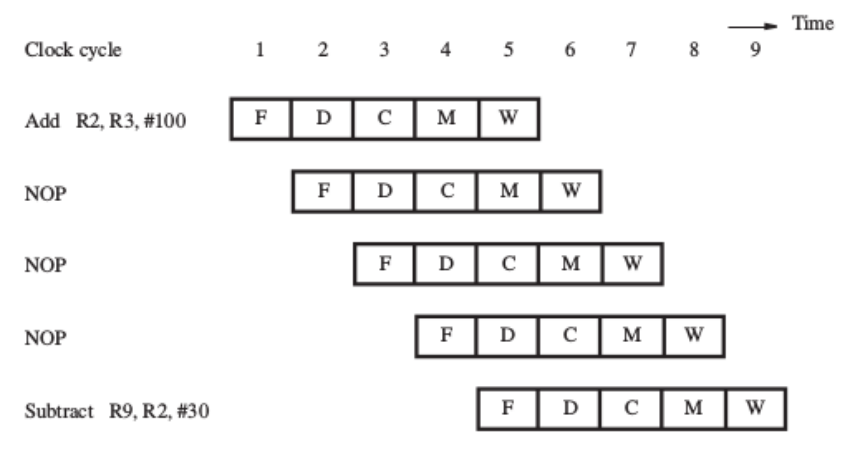


Data Dependency Hazard

Soluzione: NOP Insertion

```
ADD R2, R3 #100
NOP
NOP
NOP
SUB R9, R2 #30
```

Le **NOP** permettono alla **ADD** di completare il percorso nel datapath

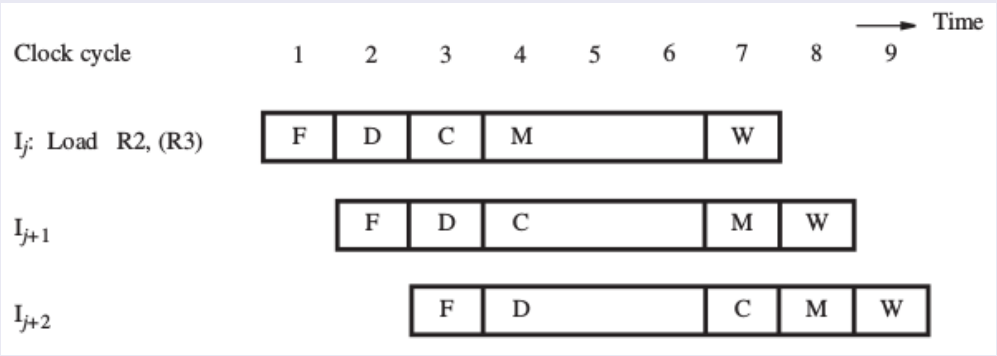


Memory Delay Hazard

Esempio

```
LDR R2, [R3]
SUB R9, R2 #30
```

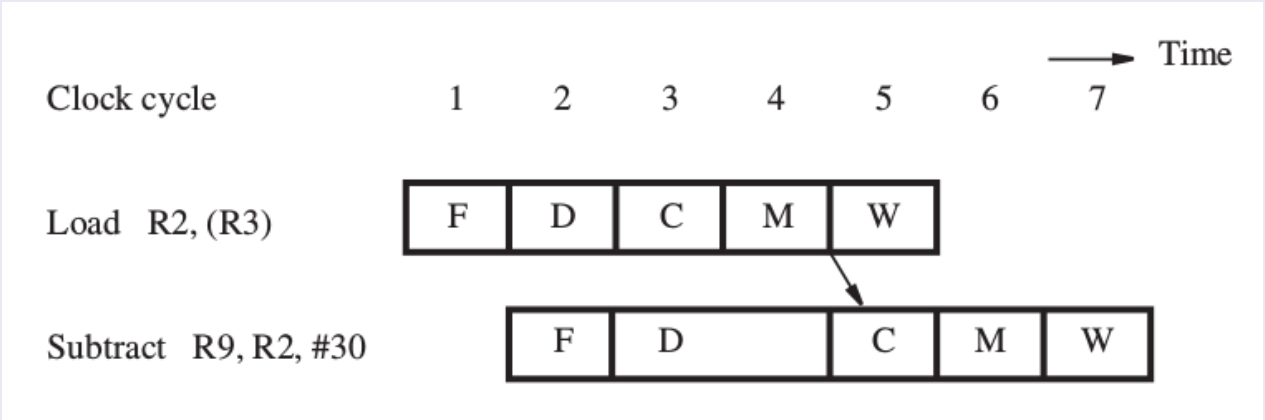
- L'accesso alla memoria richiede (in genere) vari cicli clock
- Se la dipendenza include una Load o Store, occorre bloccare la pipeline per tutta la durata dell'accesso alla memoria



Soluzione: Operand Forwarding

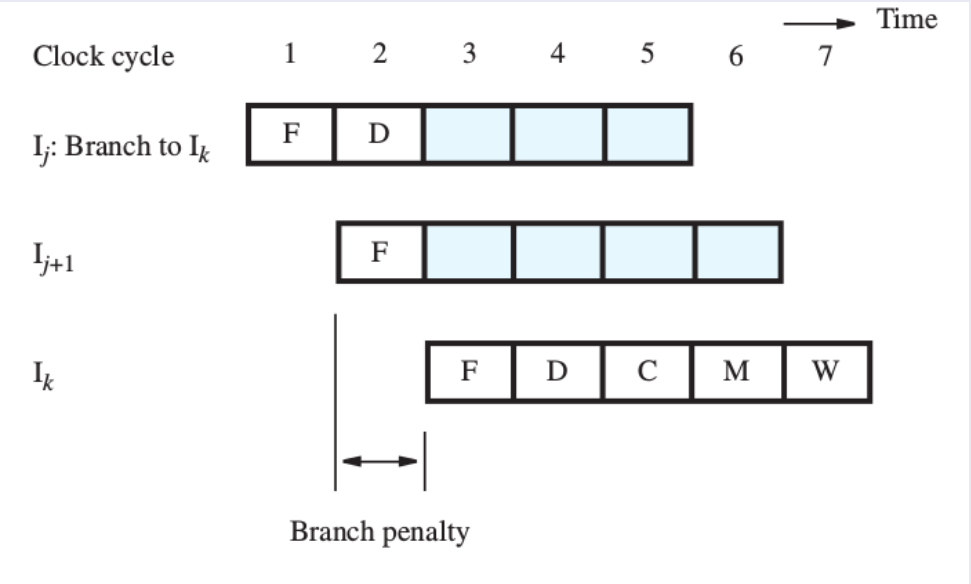
```
LDR R2, [R3]
SUB R9, R2 #30
```

- L'uscita dello stage 4 viene inviata come input aggiuntivo dello stage 3



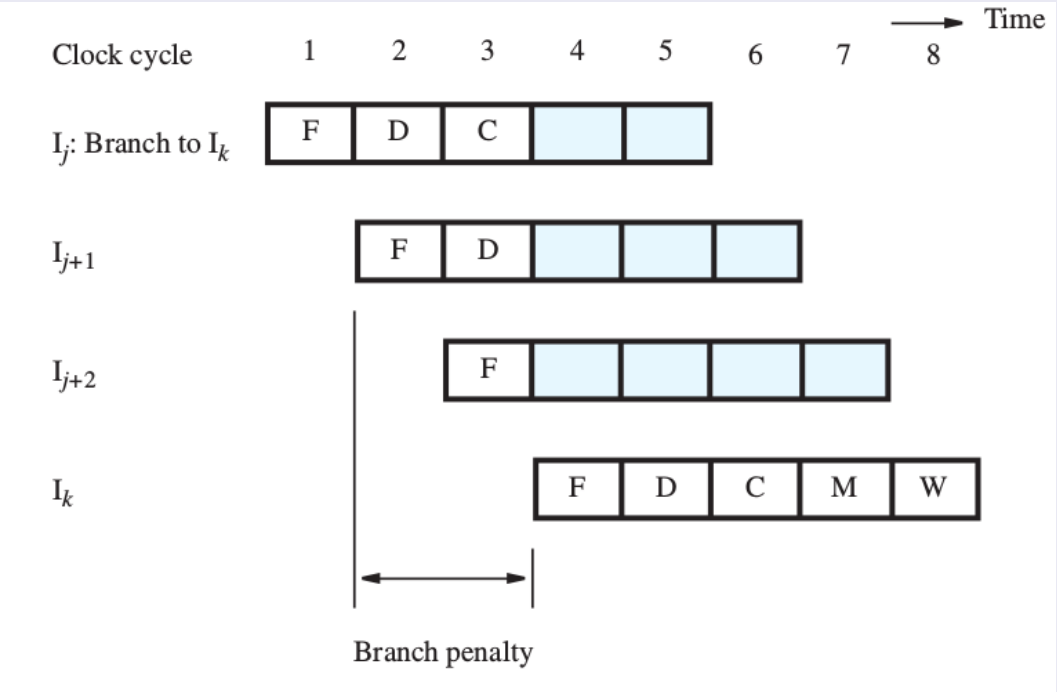
Salti incondizionati assoluti

- L'istruzione di **branch** implica sempre una "penalità" di efficienza
- Il branch con destinazione **assoluta** comporta un ritardo di esecuzione già a partire dalla fase di fetch



Salti incondizionati relativi

- L'istruzione di **branch** implica sempre una "penalità" di efficienza
- Il branch con destinazione **relativa** (`B offset`) comporta un ritardo di esecuzione a partire dalla fase di compute



Salti condizionati

- L'istruzione di **branch condizionato** implica la valutazione della condizione
- In funzione della condizione occorre eseguire il fetch o dell'istruzione **successiva** o dell'istruzione all'indirizzo **target**
- Quale istruzione prelevare? I_{j+1} o I_t?

```
        I_j       BEQ target
        I_j+1     ...
                  ...
                  ...
target  I_t       ...
```

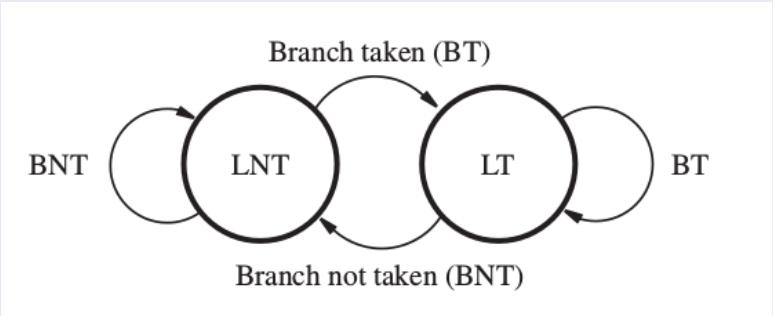
Static Branch Prediction

- Si decide **a priori** (design-time) se effettuare il fetch di I_{j+1} o I_t
- Ogni volta che la predizione è errata si paga la **"branch penalty"**

```
        I_j       BEQ target
        I_j+1     ...
              ...
              ...
target  I_t       ...
```

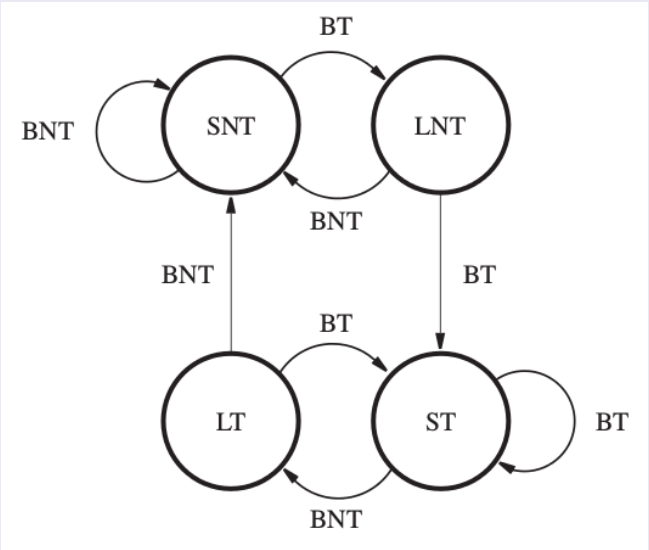
Dynamic Branch Prediction

- Macchina a stati finiti con due stati:
  - **LT** = Branch likely to be taken (salto probabilmente effettuato, l_t)
  - **LNT** = Branch likely not to be taken (salto probabilmente non effettuato, l_{j+1})
- Si decide lo **stato iniziale**
- Ogni volta che la predizione è errata si paga la **"branch penalty"** e **si cambia stato**



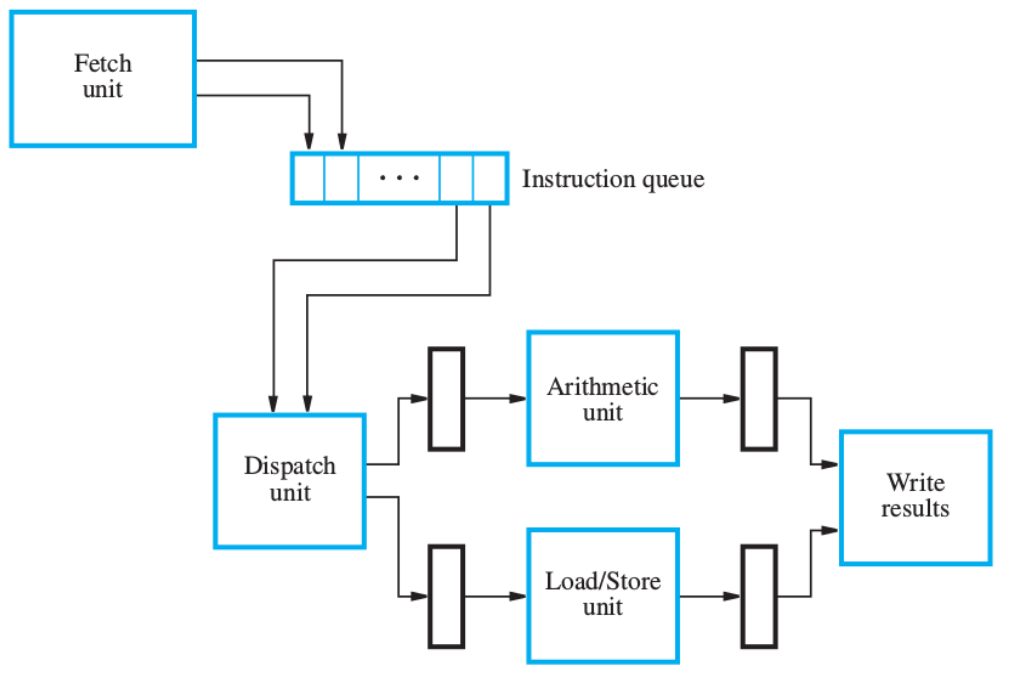
More Accurate Dynamic Branch Prediction

- Macchina a stati finiti con **quattro** stati:
  - **ST** = Strongly likely to be taken (salto molto probabilmente effettuato, I_t)
  - **LT** = Likely to be taken (salto probabilmente effettuato, I_t)
  - **LNT** = Likely not to be taken (salto probabilmente non effettuato, I_{j+1})
  - **SNT** = Strongly likely not to be taken (salto molto probabilmente non effettuato, I_{j+1})



Processori Superscalari

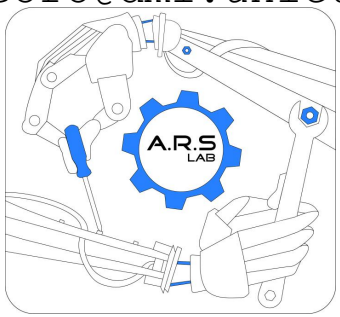
- Più pipeline **specializzate**
- Esempio:
  - *pipeline 1* $\rightarrow$ ALU Operations
  - *pipeline 2* $\rightarrow$ Load/Store Operations



Pipelining

Corrado Santoro

Dipartimento di Matematica e Informatica
santoro@dmi.unict.it



Corso di Architettura degli Elaboratori