Pipeline
Problemi strutturali

Problemi

- Maggiori risorse interne (**severità bassa**): l'evoluzione tecnologica ha spesso permesso di duplicarle
- Colli di bottiglia (**severità alta**): l'accesso alle risorse esterne, p.es.: memoria, è molto costoso e molto frequente (anche 3 accessi per ciclo di clock)

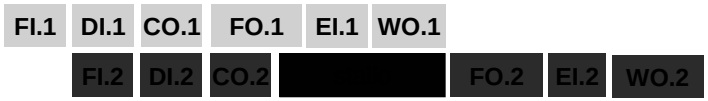
Soluzioni

- Suddividere le memorie (accessi paralleli)
- Introdurre fasi non operative (***nop***)

Pipeline
Dipendenza dai dati 1

- Un dato modificato nella fase **EI** dell'istruzione corrente può dover essere utilizzato dalla fase **FO** dell'istruzione successiva

```
INC [0123]
CMP [0123], AL
```

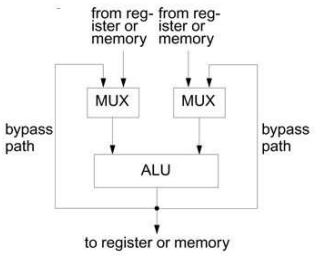


Pipeline
Dipendenza dai dati 2

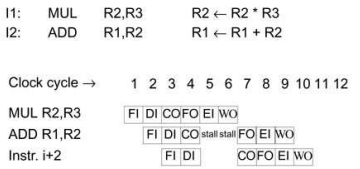
Soluzioni

- Introduzione di fasi non operative (***nop***)
- Individuazione del rischio e prelievo del dato direttamente all'uscita dell'ALU (**data forwarding**)
- Risoluzione a livello di compilatore
- Riordino delle istruzioni (**pipeline scheduling**)

Pipeline
Data forwarding



senza bypass path



con bypass path

Pipeline
Dipendenza dai controlli

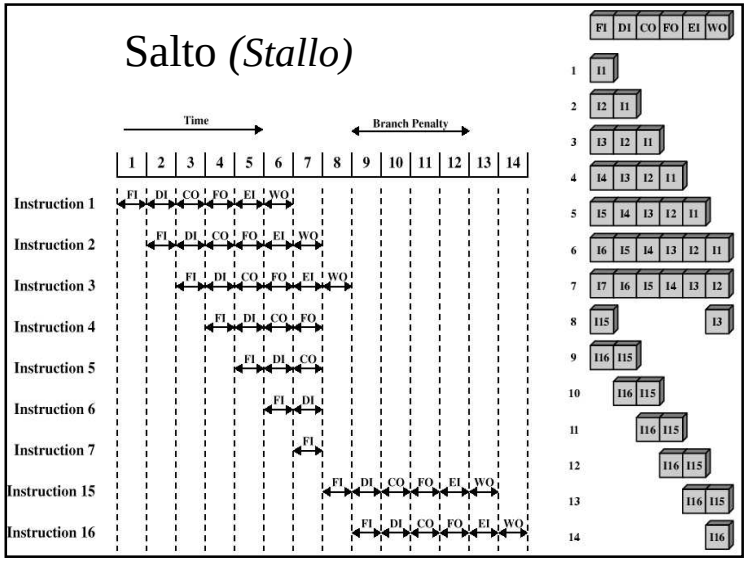
- Tutte le istruzioni che modificano il PC (salti condizionati e non, chiamate a e ritorni da procedure, interruzioni) invalidano il pipeline
- La fase **fetch** successiva carica l'istruzione seguente, che può *non essere* quella giusta
- Tali istruzioni sono circa il 30% del totale medio di un programma

Pipeline
Dipendenza dai controlli

Soluzioni

- Mettere in **stallo** il pipeline fino a quando non si è calcolato l'indirizzo della prossima istruzione
  - Pessima efficienza, massima semplicità
- Individuare le istruzioni critiche per anticiparne l'esecuzione, eventualmente mediante apposita logica di controllo
  - Compilazione complessa, hardware specifico

Salto *(Stallo)*



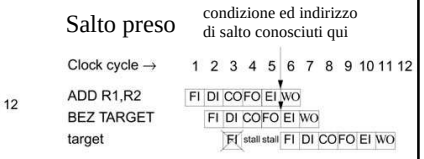
Salto *(Stallo)*

Salto incondizionato

inizialmente caricata istruzione successiva a BR TARGET, poi eliminata quando si riconosce il salto incondizionato alla fine della fase DI

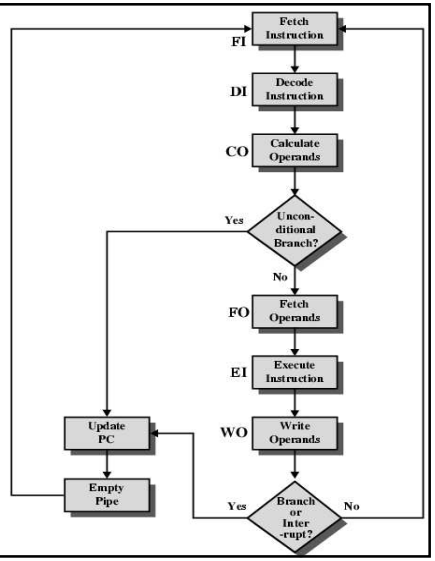
Salto condizionato

Salto preso



Salto non preso

Funzionamento pipeline a 6 stadi con trattamento dei salti ed interrupt tramite svuotamento



Pipeline
Dipendenza dai controlli

Alcune soluzioni per salti condizionati

- flussi multipli (multiple streams)
- prelievo anticipato della destinazione (prefetch branch target)
- buffer circolare (loop buffer)
- predizione del salto (branch prediction)
- salto ritardato (delayed branch)

Pipeline
Dipendenza dai controlli

Flussi multipli: replicare le parti iniziali della pipeline, una che contenga l'istruzione successiva a quella corrente di salto (nel caso il salto non avvenga), e l'altra l'istruzione destinazione (*target*) del salto (nel caso in cui il salto avvenga)

Problemi di questa soluzione:

- conflitti nell'accesso alle risorse (registri, memoria, ALU,…) da parte delle 2 pipeline
- presenza di salti condizionali in sequenza che entrano nelle 2 pipeline prima che si sia risolta la condizione del primo salto condizionale (occorrerebbero 2 pipeline aggiuntive per ogni ulteriore salto condizionale…)

Pipeline
Dipendenza dai controlli

Prelievo anticipato della destinazione: quando si incontra un salto condizionato si effettua il fetch anticipato della istruzione di destinazione del salto in modo da trovarla già caricata nel caso in cui il salto debba avvenire.

Problemi di questa soluzione:

- non evita l'eventuale svuotamento della pipeline con conseguente perdita di prestazioni

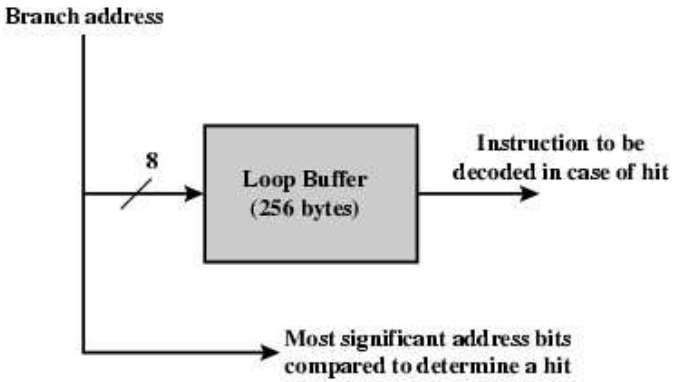
Pipeline
Dipendenza dai controlli

Buffer circolare: si utilizza una memoria piccola e molto veloce (il buffer circolare) dove mantenere le ultime *n* istruzioni prelevate. In caso di salto, si controlla se l'istruzione destinazione è già presente nel buffer, così da evitare il fetch della stessa.

Vantaggi:

- anticipando il fetch, alcune delle istruzioni successive a quella corrente saranno già presenti nel buffer e se non si ha salto non ci sarà bisogno di caricarle dalla memoria
- se si salta in avanti di poche istruzioni (vedi trattamento del costrutto IF THEN ELSE), l'istruzione destinazione sarà già presente nel buffer
- se il salto condizionale realizza un ciclo le cui istruzioni possono essere tutte contenute nel buffer, non c'è bisogno di effettuare fetch ripetuti delle stesse istruzioni

Pipeline
Buffer circolare



Pipeline
Dipendenza dai controlli

Predizione dei salti: si cerca di prevedere se il salto sarà intrapreso oppure no.

Varie possibilità:

- previsione di saltare sempre
- previsone di non saltare mai
- previsione in base al codice operativo

} *approcci statici*

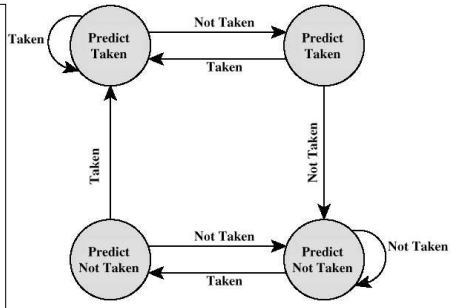
- bit *taken/not taken*
- tabella della storia dei salti

} *approcci dinamici*

Approcci dinamici di predizione: cercano di migliorare la qualità della predizione sul salto memorizzando la storia delle istruzioni di salto condizionato di uno specifico programma.

bit *taken/not taken* :

- ad ogni istruzione di salto condizionato si associano uno o più bit che codificano la storia recente.
- bit memorizzati non in memoria centrale ma in una locazione temporanea ad accesso molto veloce



esempio con 2 bit