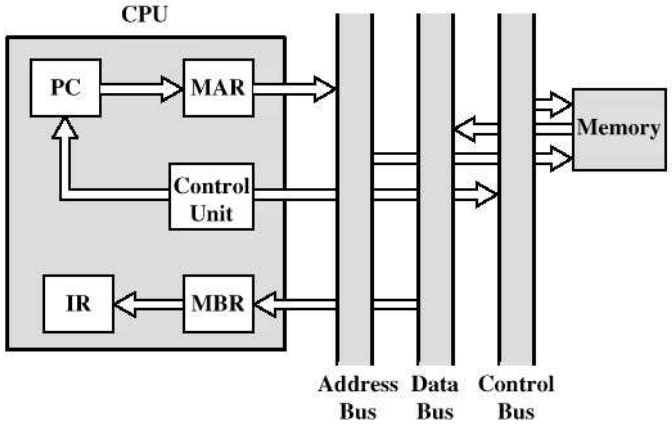
Flusso dei dati (Instruction Fetch)

Dipende dalla architettura della CPU, in generale:

- Fetch
  - PC contiene l'indirizzo della istruzione successiva
  - Tale indirizzo viene spostato in MAR
  - L'indirizzo viene emesso sul bus degli indirizzi
  - La unità di controllo richiede una lettura in memoria principale
  - Il risultato della lettura in memoria principale viene inviato nel bus dati, copiato in MBR, ed infine in IR
  - Contemporaneamente il PC viene incrementato

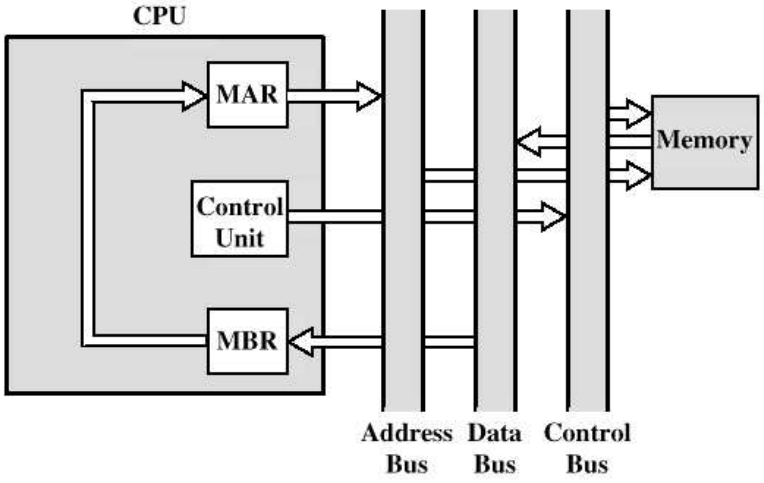
Flusso dei dati (Diagramma di Fetch)



Flusso dei dati (Data Fetch)

- IR è esaminato
- Se il codice operativo della istruzione richiede un indirizzamento indiretto, si esegue il ciclo di indirettezza
  - gli N bit più a destra di MBR vengono trasferiti nel MAR
  - L'unità di controllo richiede la lettura dalla memoria principale
  - Il risultato della lettura (indirizzo dell'operando) viene trasferito in MBR

Flusso dei dati (Diagramma di Indirettezza)



Flusso dei dati (Execute)

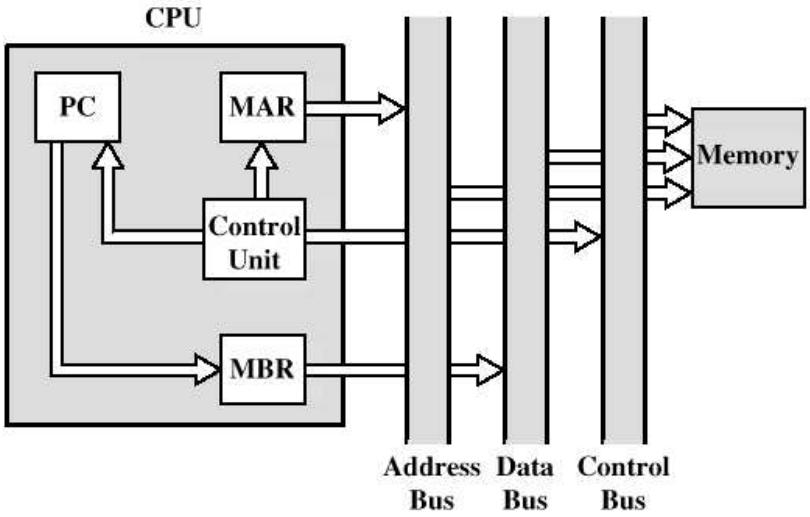
- Può assumere molte forme
- Dipende dalla istruzione da eseguire
- Può includere
  - lettura/scrittura della memoria
  - Input/Output
  - Trasferimento di dati fra registri e/o in registri
  - Operazioni della ALU

Flusso dei dati (Interrupt)

Semplice e prevedibile:

- Contenuto corrente del PC deve essere salvato per permettere il ripristino della esecuzione dopo la gestione dell'interruzione
  - Contenuto PC copiato in MBR
  - Indirizzo di locazione di memoria speciale (es. stack pointer) caricato in MAR
  - Contenuto di MBR scritto in memoria
- PC caricato con l'indirizzo della prima istruzione della routine di gestione della interruzione
- Fetch della istruzione puntata da PC

Flusso dei dati (Diagram. di Interrupt)



Prefetch

- La fase di prelievo della istruzione accede alla memoria principale
- La fase di esecuzione di solito non accede alla memoria principale
- Si può prelevare l'istruzione successiva durante l'esecuzione della istruzione corrente
- Questa operazione si chiama "instruction prefetch"

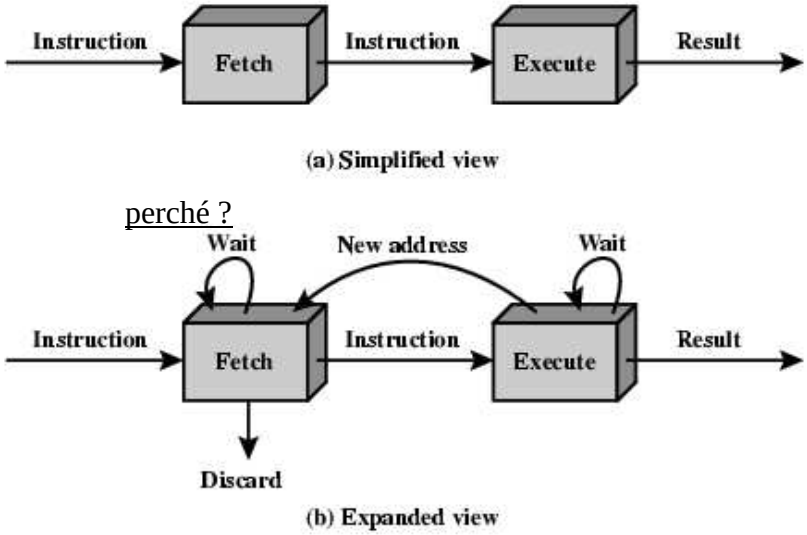
Prefetch

(a) Simplified view

Miglioramento delle prestazioni

- Il prefetch non raddoppia le prestazioni:
  - L'esecuzione di istruzioni jump o branch possono rendere vano il prefetch (perché ?)
  - La fase di prelievo è tipicamente più breve della fase di esecuzione
    - Prefetch di più istruzioni ?
- Aggiungere più fasi per migliorare le prestazioni

Prefetch



Evoluzione delle architetture

Evoluzione strutturale

- **Parallelismo**
  - Se un lavoro non può essere svolto più velocemente da una sola persona (unità), allora conviene **decomporlo** in parti che possano essere eseguite da più persone (unità) **contemporaneamente**
    - **Catena di montaggio**

Pipeline
Generalità 1

- Ipotizziamo che per svolgere un dato lavoro **L** si debbano eseguire tre fasi distinte e sequenziali

  L $\Rightarrow$ [*fase1*] [*fase2*] [*fase3*]
- Se ogni fase richiede **T** unità di tempo, un unico esecutore svolge un lavoro **L** ogni **3T** unità di tempo
- Per ridurre i tempi di produzione si possono utilizzare **più esecutori**

Pipeline
Generalità 2

- Soluzione (ideale) a parallelismo totale

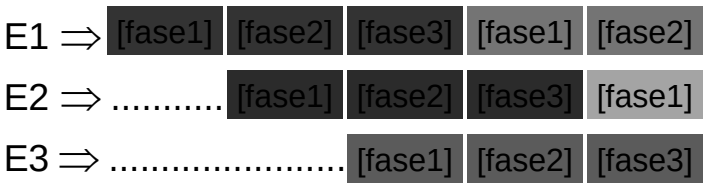
  E1 $\Rightarrow$ [fase1.**A**] [fase2.**A**] [fase3.**A**] | [fase1.**D**] …

  E2 $\Rightarrow$ [fase1.**B**] [fase2.**B**] [fase3.**B**] | [fase1.**E**] …

  E3 $\Rightarrow$ [fase1.**C**] [fase2.**C**] [fase3.**C**] | [fase1.**F**] …
- **N** esecutori svolgono un lavoro ogni **3T/N** unità di tempo
- Il problema è **come** preservare la **dipendenza funzionale** nell'esecuzione (di fasi) dei 'lavori' **A**, **B**, **C**, **D**, **E**, **F**, …

Pipeline
Generalità 3

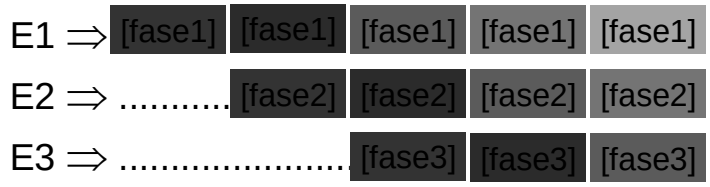
- Soluzione **pipeline** ad **esecutori generici**



- Ogni esecutore esegue un ciclo di lavoro **completo** (*sistema totalmente replicato*)
- A regime, **N** esecutori svolgono un lavoro **L** ogni **3T/N** unità di tempo rispettandone la sequenza

Pipeline
Generalità 4

- Soluzione **pipeline** ad **esecutori specializzati**



- Ogni esecutore svolge sempre e solo la **stessa** fase di lavoro
- Soluzione più efficace in termini di **uso di risorse** (**3T/N** lavori con **N/3** risorse)

Prefetch come pipeline a due stadi