FONDAMENTI DI INFORMATICA

Prof. PIER LUCA MONTESSORO

Facoltà di Ingegneria
Università degli Studi di Udine

Architetture CISC e RISC

Nota di Copyright

Architetture CISC e RISC

- Misura della potenza di calcolo
- Unità di controllo microprogrammata
- Architetture RISC e pipeline

Misura della potenza di calcolo

Cos'è la potenza di calcolo

- È la capacità del sistema di eseguire elaborazioni più o meno complesse in tempi più o meno lunghi

Prestazioni della CPU

- Il tempo T_{CPU} per eseguire un programma può essere stimato in:

$$T_{CPU} = N_{istr} \cdot C_{PI} \cdot T$$

dove:

- N_{istr} è il numero di istruzioni del programma
- C_{PI} è il numero medio di cicli di clock per istruzione macchina
- $T=1/f_{ck}$ è il periodo del clock

Osservazioni

- N_{istr} dipende dal repertorio di istruzioni e dall'ottimizzazione del compilatore

  Una CPU CISC permette di ridurre N_{istr}
- C_{PI} dipende dall'architettura. Un repertorio di istruzioni semplice permette di eseguire un'istruzioni in pochi cicli di clock e di sfruttare un elevato pipeline (RISC), riducendo C_{PI}
- $T=1/f_{ck}$ dipende dalla tecnologia elettronica e, in parte, dall'architettura. Istruzioni complesse richiedono frequenze di clock più basse

Prestazioni complessive

- Oltre alle prestazioni della CPU vanno considerati anche:
  - numero delle CPU (per sistemi multiprocessore)
  - architettura complessiva del sistema
  - velocità di accesso alla memoria RAM
  - quantità di memoria RAM
  - velocità del disco di paginazione
  - velocità dei dispositivi periferici

Indici di prestazioni

Per la misura delle prestazioni di un'architettura si utilizzano dei programmi di riferimento (*benchmarks*)

- Problema: le prestazioni di un elaboratore dipendono dal tipo di programma che viene eseguito. L'insieme dei benchmark utilizzato può favorire una macchina e penalizzarne un'altra

Indici di prestazioni

- I principali indici utilizzati sono:
  - MIPS
  - MFLOPS
  - SPECMARKS

MIPS

- Millions of Instructions Per Second
- Misura alquanto arbitraria:
  - Istruzione media di un applicativo medio
  - Legata principalmente al calcolo intero (puntatori, stringhe, database, ecc.)
- Riferimento: il VAX 11/780 della Digital era una macchina da 1 MIPS
- In seguito alla diffusione degli elaboratori Digital è stata adottata anche come unità di misura da altre case

MFLOPS

- Millions of FLOating-point instructions Per Second
- Misura arbitraria tanto quanto il MIPS:
  - Legata principalmente al calcolo numerico (es. inversione di matrici di numeri in singola o doppia precisione)
- È abbastanza scorrelata dal numero di MIPS dello stesso calcolatore perché è determinante la presenza o meno di un coprocessore matematico

SPECMARKS

- Misura adottata da numerosi costruttori e basata su un set standard di benchmark
- I benchmarck utilizzati sono programmi *public domain*, e quindi permettono di effettuare confronti abbastanza significativi
- L'impostazione dei benchmark è tuttavia alquanto sbilanciata verso il calcolo in floating-point

Unità di controllo microprogrammata

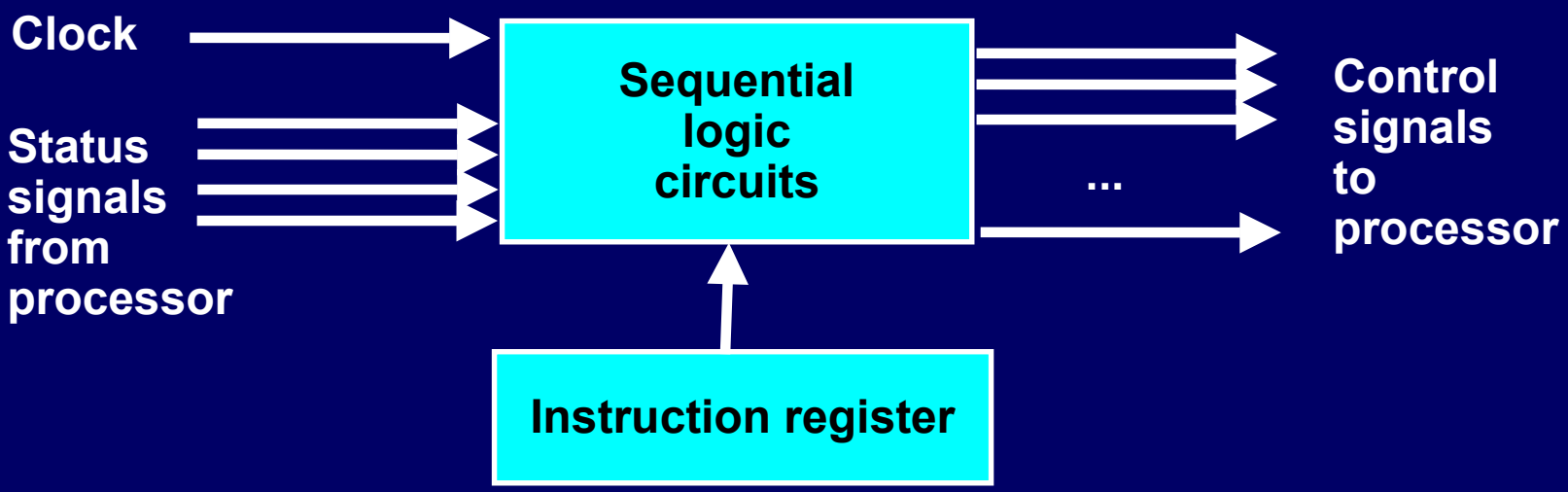
CISC
Complex Instruction Set Computer

- L'insieme di istruzioni macchina è molto vasto, e riduce il "gap" tra linguaggio ad alto livello e linguaggio macchina
- In alcuni casi (es. VAX) si possono avere istruzioni addirittura più potenti di quelle esprimibili in linguaggi ad alto livello. Questo aumenta la complessità dei compilatori
- Spesso le CPU CISC sono microprogrammate

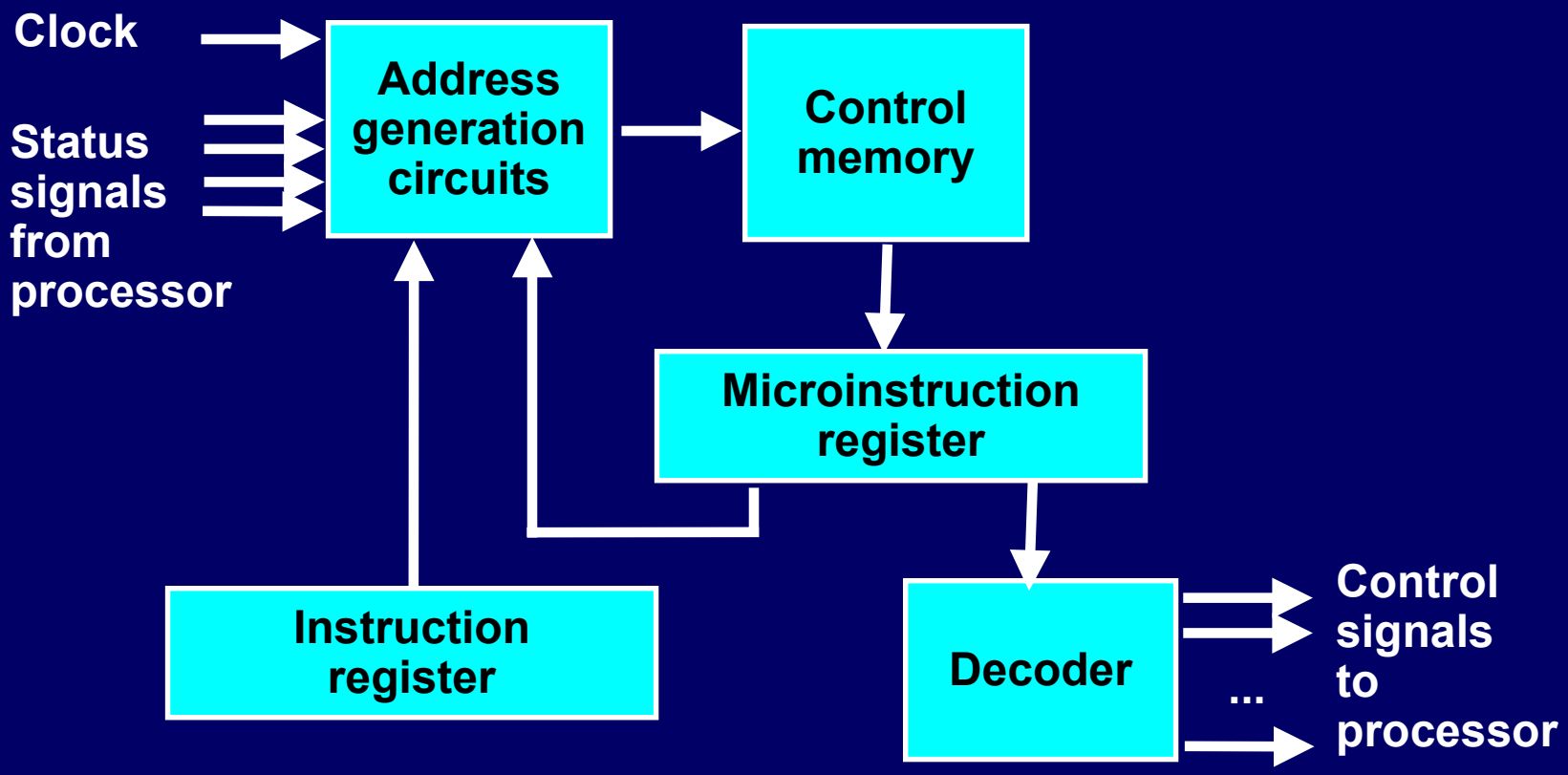
Microprogrammazione

- La fase di decodifica dell'istruzione (in questo contesto detta macroistruzione) nella CPU avviene attraverso un programma residente nella CPU stessa (microcodice, cioè sequenza di microistruzioni)
- Vantaggi:
  - macroistruzioni molto potenti
  - facile gestione di macroistruzioni a formato variabile (es. numero di operandi)
  - possibilità di upgrade del microcodice

Unità di controllo non microprogrammata



Unità di controllo microprogrammata



Architetture RISC e pipeline

Tipo di CPU

- Due tipi di CPU:
  - CISC - Complex Instruction Set Computer
  - RISC - Reduced Instruction Set Computer

QUANTO È COMPLESSA UN' "ISTRUZIONE"?

RISC

- Progetto dell'università di Berkeley (CA) iniziato nel 1980

- Principio di base:
  1) valutare lo sfruttamento dell'area di silicio nei processori single-chip
  2) individuarne gli svantaggi
  3) definire un'architettura più efficiente

Vantaggi ottenibili dall'aumento della complessità dei circuiti digitali

- Maggior parallelismo: un sommatore da 32 bit è più veloce di uno da 16 usato 2 volte per ogni operazione
- Possibilità di overlap tra esecuzione di un'istruzione ed il fetch della successiva (per programmi con un numero non eccessivo di salti - alcuni processori eseguono anche prefetch su più rami)
- Più registri nella CPU, sfruttabili dal compilatore per ottimizzare il codice

Svantaggi dovuti all'aumento della complessità dei circuiti digitali

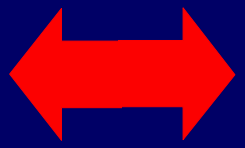
- Dimensioni maggiori implicano maggiori ritardi di propagazione dei segnali
- Un maggior numero di porte logiche implica una minor quantità di potenza elettrica disponibile per ciascuna, e quindi minor capacità di pilotare altre porte
- Necessità di più elementi: più numerosi o più grandi multiplexer, maggior fanout di uscita, più circuiti sui bus - tutto questo riduce la velocità

Esiste un limite tecnologico alla dimensione dell'area di silicio

È necessario operare una scelta:

- Elevata complessità dell'unità di controllo (e quindi più istruzioni)
- Pochi registri

A parità di area di silicio



- Ridotta complessità dell'unità di controllo (e quindi meno istruzioni)
- Register file

Prima fase del progetto RISC

- Individuare le operazioni (e, di conseguenza, le istruzioni) indispensabili e d'uso più frequente nei programmi
- Analizzare gli elementi funzionali necessari per la loro esecuzione e la tempistica
- Includere nell'insieme di istruzioni così definito altre istruzioni ancora abbastanza frequenti e che possono sfruttare tali elementi funzionali (cioè che non richiedono modifiche dell'architettura)

E le altre istruzioni?

- Vengono realizzate tramite sequenze di istruzioni più semplici. Tuttavia questo non è svantaggioso, in quanto:
  - è raramente necessario, anche con programmi in linguaggi ad alto livello (un programma medio su un RISC richiede il 67% di istruzioni in più rispetto allo stesso programma eseguito su un VAX-11)
  - il compilatore è più semplice, più efficiente, e può trarre vantaggio dall'architettura (es. maggior numero di registri)

Seconda fase

- Definizione dell'architettura

  tutte le istruzioni in una sola parola, con formato fisso

  → semplicità ed efficienza per il fetch e la decodifica

  molti registri nella CPU (globali e in overlapping windows)

  → veloce accesso agli operandi (tipicamente in un computer vengono effettuati il 70% di accessi ad operandi e il 30% di operazioni)

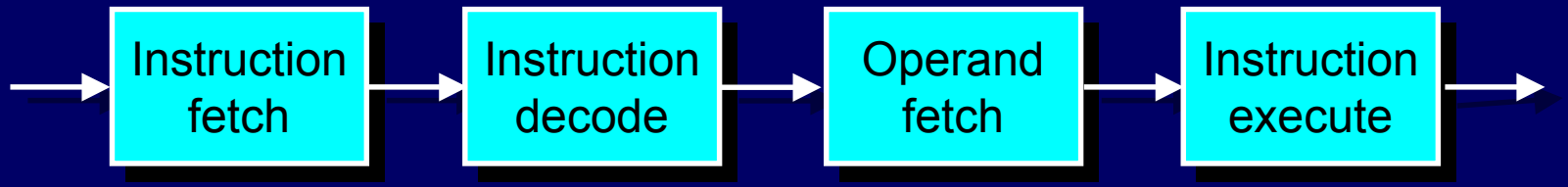
  operazioni solo su registri, load e store accedono alla memoria

  → semplicità ed efficienza

  – pipeline e delayed control transfer

Pipeline

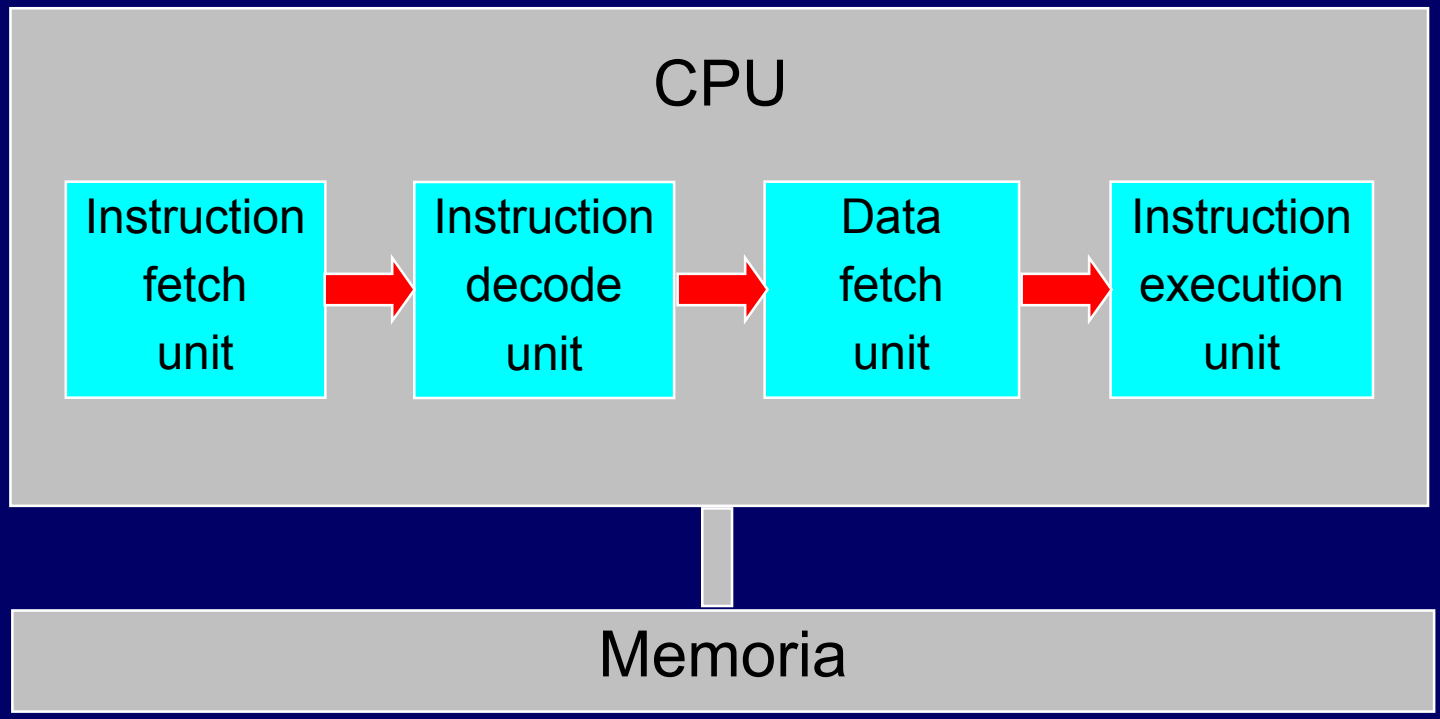
- Normalmente l'esecuzione di un'istruzione segue lo schema:



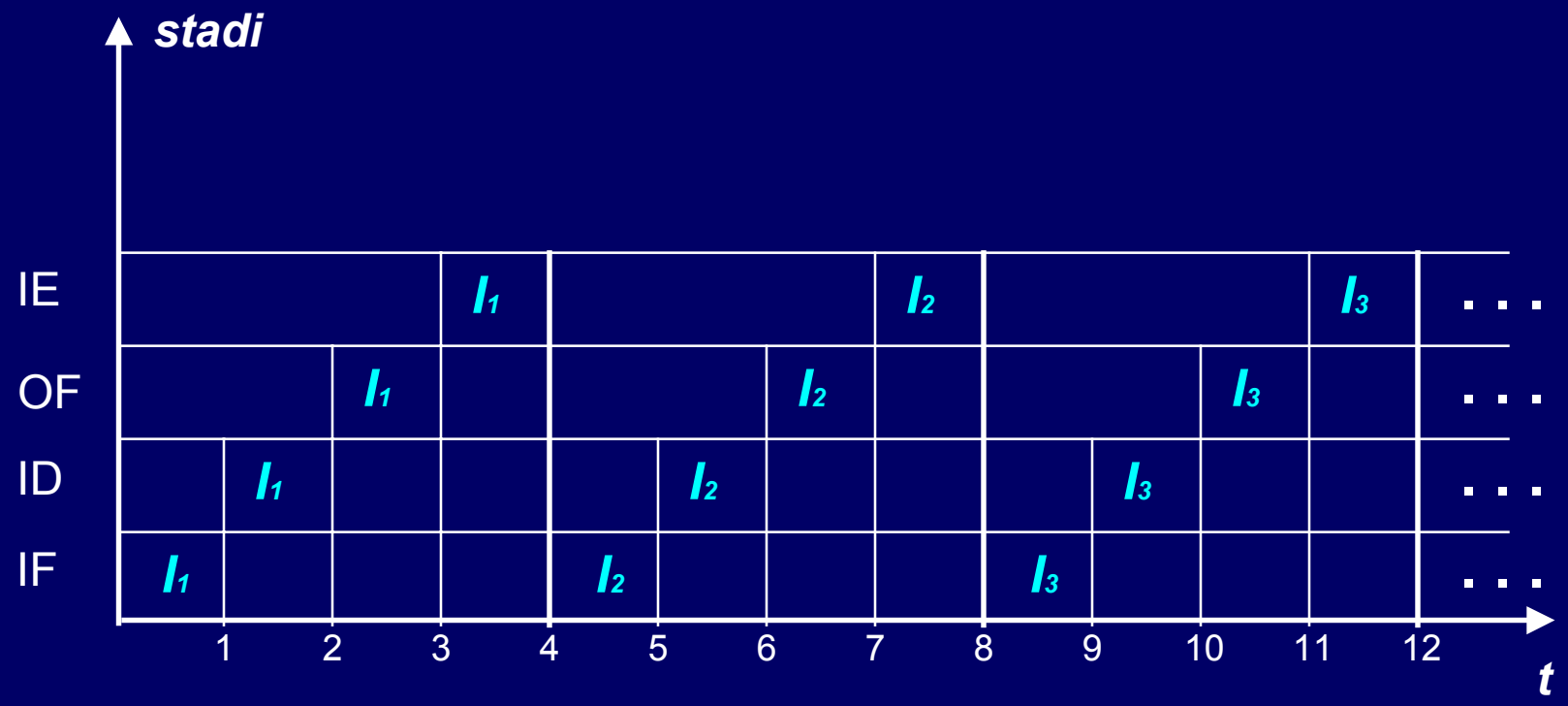
- Nei calcolatori senza pipeline ogni fase può iniziare soltanto dopo che la precedente è terminata
- Con il pipeline, invece . . .

Pipeline

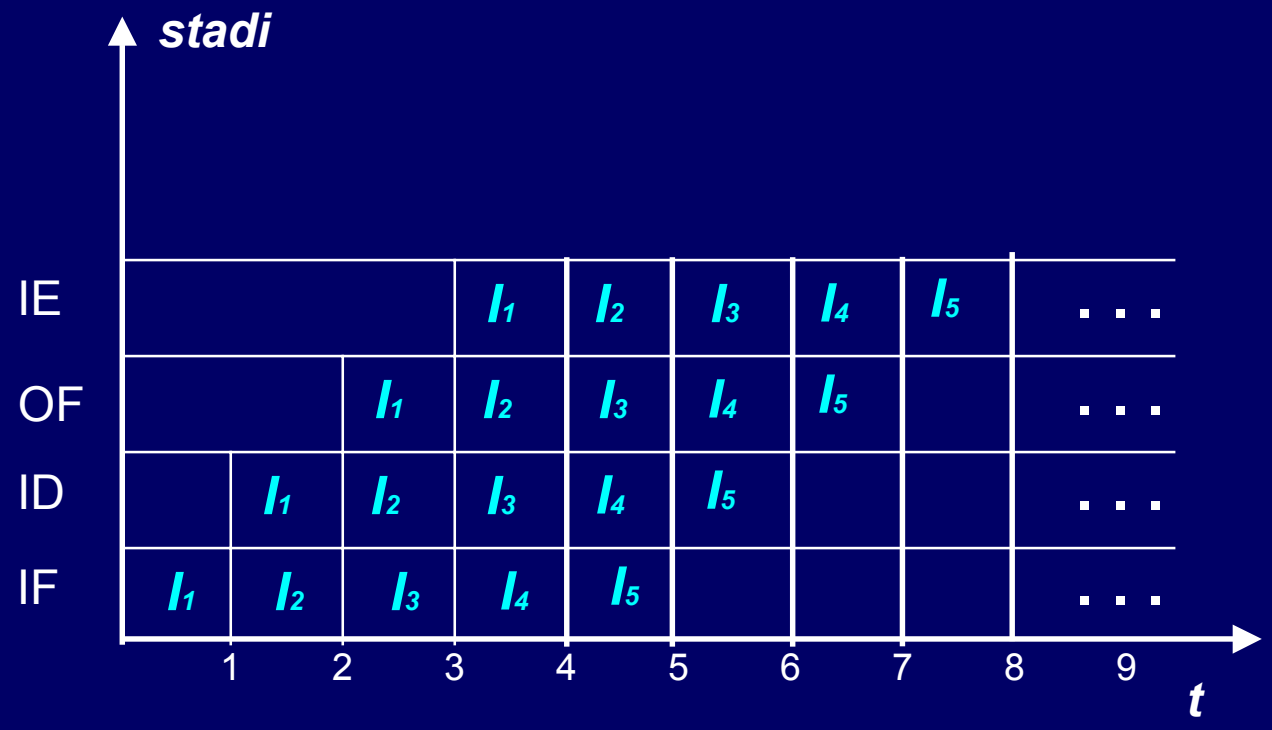
- Unità funzionali indipendenti



Esecuzione senza pipeline



Esecuzione con pipeline



Delayed control transfer

```
X++;
if (y == 5)
{
    z = 0;
    ...
}
else
{
    z = 1;
    ...
}
```

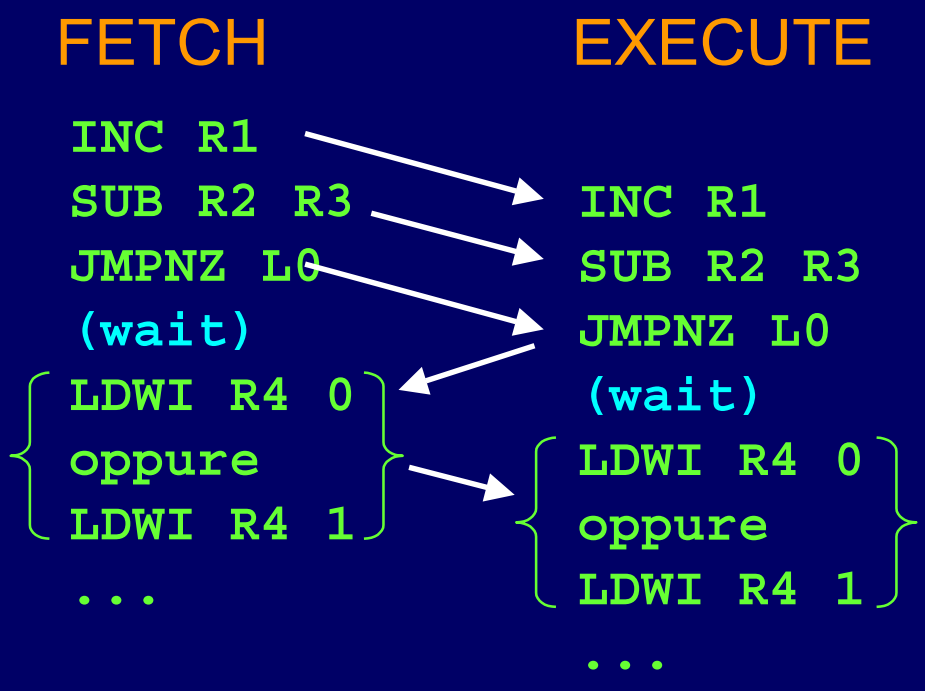
il costrutto *if ... then ... else* impedisce il prefetch dell'istruzione successiva: sarà *z = 0* o *z = 1*?

Codice generato dal compilatore

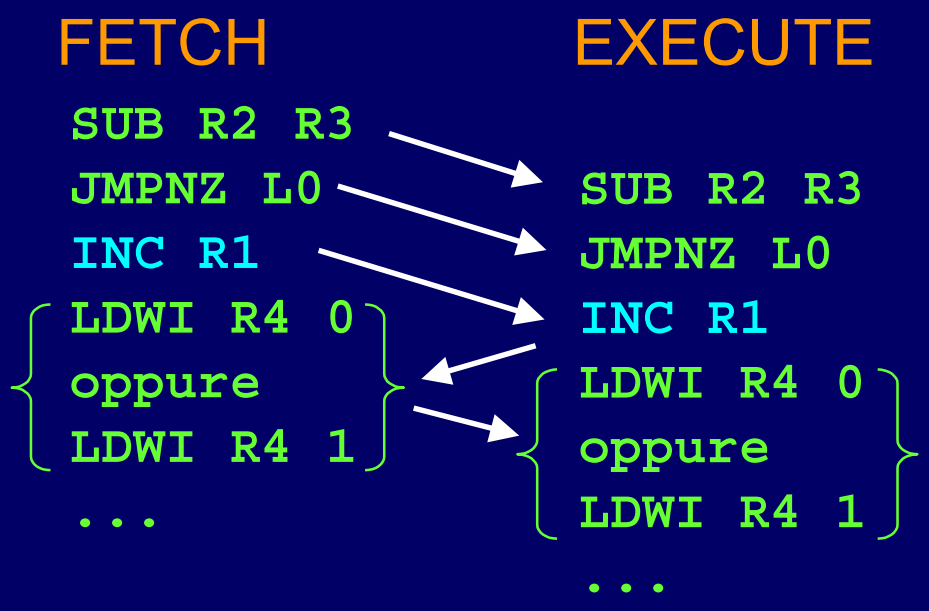
```
X++;
if (y == 5)
{
    z = 0;
    ...
}
else
{
    z = 1;
    ...
}
```

```
    INC R1     ; (R1 contiene x)
    SUB R2 R3  ; (R2 = y, R3 = 5)
    JMPNZ L0
    LDWI R4 0  ; (R4 = z)
    ...
    JMP L1
L0: LDWI R4 1
    ...
L1:
```

Esecuzione



Codice ottimizzato



I compilatori e l'ottimizzazione del codice

- La funzione più complessa di un compilatore moderno consiste nell'ottimizzazione del codice
- La complessità di tale compito si scontra con la necessità che il compilatore sia veloce
- I compilatori per architetture RISC riescono in genere ad ottimizzare il codice meglio ed in meno tempo rispetto ai compilatori per CISC, i quali devono lavorare in un contesto molto più complicato (elevato numero di istruzioni)

RISC

- Riduzione del set di istruzioni per poter rendere più semplice la parte di controllo nella CPU
- Le istruzioni scartate sono quelle statisticamente meno usate e vengono emulate via software
- Minor spazio sul chip occupato dall'unità di controllo, e quindi possibilità di avere più registri, maggiore pipeline, ecc.

Evoluzione delle architetture RISC: i processori superscalari

- Possono avviare l'esecuzione di più di un'istruzione per colpo di clock
- Dispongono di unità aritmetiche indipendenti (es. per calcolo intero, per calcolo floating point, per operazioni logiche, ecc.) che possono eseguire istruzioni in parallelo
- I processori Pentium Intel si basano su un'architettura superscalare