



UNIVERSITÀ
di **VERONA**

Dipartimento
di **INFORMATICA**

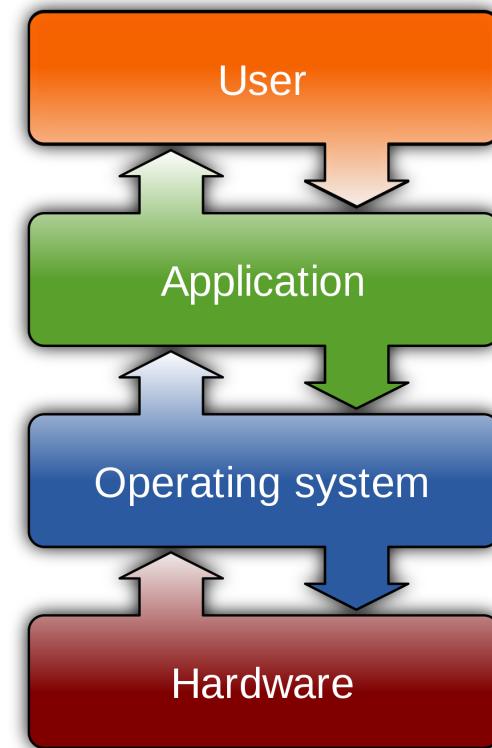
Definizione di sistema operativo

Anno Accademico 2020-2021

Graziano Pravadelli

Che cos'è un sistema operativo?

- Insieme di programmi
- Intermediario tra HW e utente per:
 - facilitare l'uso del computer
 - rendere efficiente l'uso dell'HW
 - evitare conflitti nella allocazione delle risorse HW/SW
- Ambiente per coordinare l'utilizzo dell'HW (CPU, memoria, I/O, ...) da parte dei programmi applicativi



Viste di un sistema operativo

Gestore di risorse

- Risorse HW
 - Dischi, memoria, I/O,...
- Risorse SW
 - File, programmi,...

Programma di controllo

- Controllo dell'esecuzione dei programmi e del corretto utilizzo del sistema

Obiettivi di un sistema operativo

Astrazione

- Semplificazione dell'uso del sistema

Efficienza

- Quanto costa astrarre?

Astrazione ed efficienza
tipicamente in contrasto!

- Windows (molto astratto, meno efficiente)
- Unix (meno astratto, più efficiente)



Storia dei sistemi operativi

The History of the Operating System

Storia dei S.O.

4 generazioni legate
ai calcolatori

Principio ispiratore

Aumento dell'utilizzo
del processore

1a

Generazione (1946-1955)

Enormi calcolatori
a valvole

Non esiste
sistema operativo

Operatore =
Programmatore

Accesso alla
macchina tramite
prenotazione

Esecuzione da
console

- Programma caricato in memoria un'istruzione alla volta agendo su interruttori
- Controllo errori su spie della console

1a

Generazione (Evoluzione)

Diffusione di periferiche (lettore/perforatore di schede, nastri, stampanti)

- Programmi di interazione con periferiche (device driver)

Sviluppo di “software”

- “Librerie” di funzioni comuni
- Compilatori, linker, loader

Scarsa efficienza

- Programmazione facilitata, ma operazioni complesse
- Tempi di setup elevati
 - → basso utilizzo

2a Generazione (1955-1965)

Introduzione dei
transistor nei
calcolatori

Operatore !=
programmatore

Batching

- Eliminazione dello schema a prenotazione
- Operatore elimina parte dei tempi morti

- Batch = lotto
- Raggruppamento di programmi (*job*) simili nell'esecuzione

Batching - Esempio

Senza batching

- Sequenza job: Fortran, Cobol, Fortran, Cobol
 - Comp. Fortran, linker Fortran, comp. Cobol, linker Cobol, Comp. Fortran, ...

Con batching

- Sequenza job: Fortran, Fortran, Cobol, Cobol
 - Comp. Fortran, linker Fortran, comp. Cobol, linker Cobol.

2a Generazione (Evoluzione)

Automatic job sequencing

- Il sistema si occupa di passare da un job all'altro
- S.O. fa il lavoro dell'operatore e rimuove tempi morti

Monitor residente

- Primo vero esempio di S.O.
 - monitor = “gestore”
 - residente → perennemente caricato in memoria
- Componenti del monitor:
 - Driver per dispositivi di I/O
 - Sequenzializzatore dei job
 - Interprete di schede di controllo (lettura ed esecuzione)

Memoria

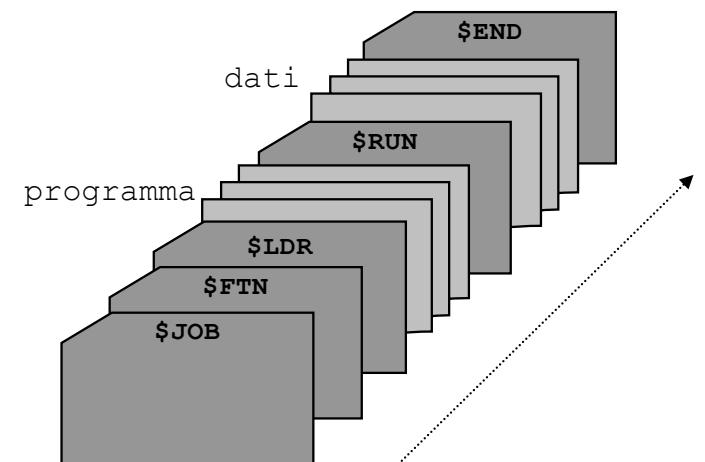
monitor

Driver
Job seq.
Interprete
Programmi utente

2a Generazione (Evoluzione)

- Sequenzializzazione realizzata tramite un linguaggio di controllo (Job Control Language)
 - Schede di controllo
 - Record di controllo (nastri)

- Esempio:
 - inizia job (\$JOB)
 - fine job (\$END)
 - compilatore (es. Fortran) (\$FTN)
 - linker (\$LNK)
 - loader (\$LDR)
 - ...



2a Generazione (Limitazioni)

Problema

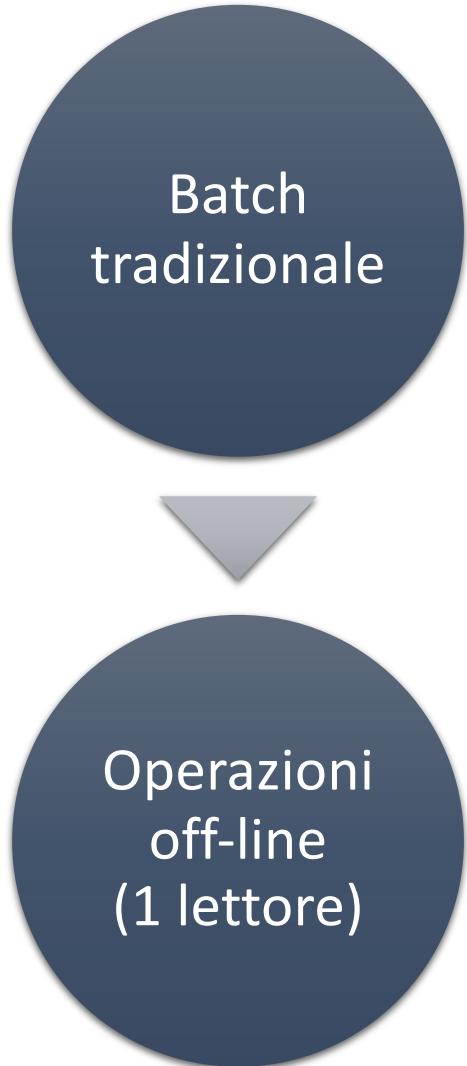
- Utilizzo del sistema ancora basso
- Velocità I/O << Velocità CPU

Esempio

- Compilatore può processare 300 schede/sec
- Velocità lettore di schede: 20 schede/sec
- Lettura di un programma di 1200 schede:
 - 4 sec CPU, 60 sec I/O,
 - Utilizzo CPU = 4/64 → 6.25%

Osservazione

- CPU mai attiva durante I/O
- E' possibile superare questa limitazione?



2a Generazione (Soluzione)

Sovrapposizione delle operazioni di I/O ed elaborazione (CPU)

Elaborazione off-line

- Diffusione dei nastri magnetici, più capienti e più veloci
- Sovrapposizione di I/O e CPU su macchine indipendenti
 - Da scheda a nastro su una macchina
 - Da nastro a CPU su un'altra macchina

CPU ora limitata dalla velocità dei nastri!

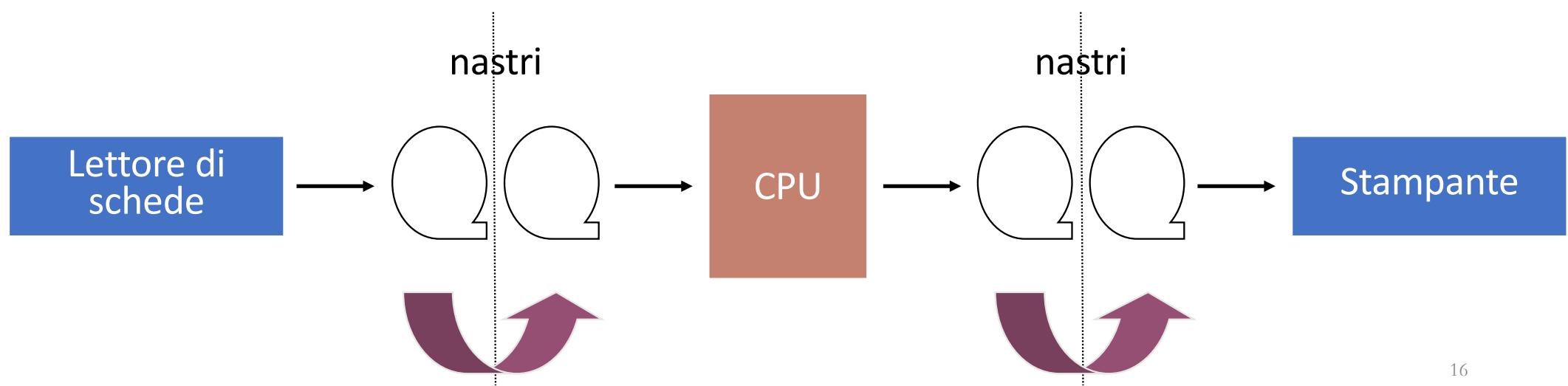
- Vero vantaggio nel caso di più lettori di nastro

Batch vs. operazioni off-line

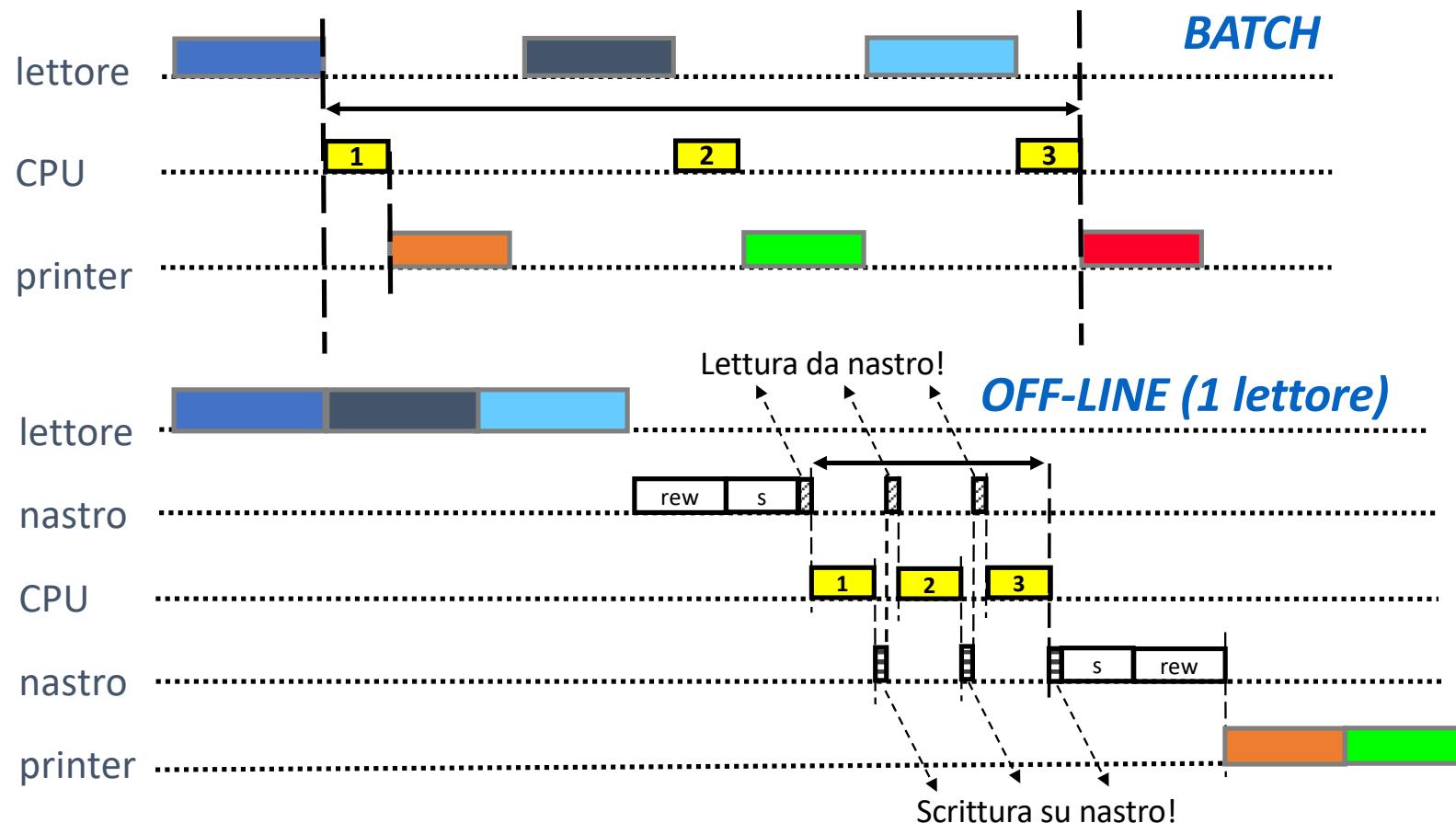
Batch tradizionale



Operazioni off-line (1 lettore)



Batch vs. operazioni offline

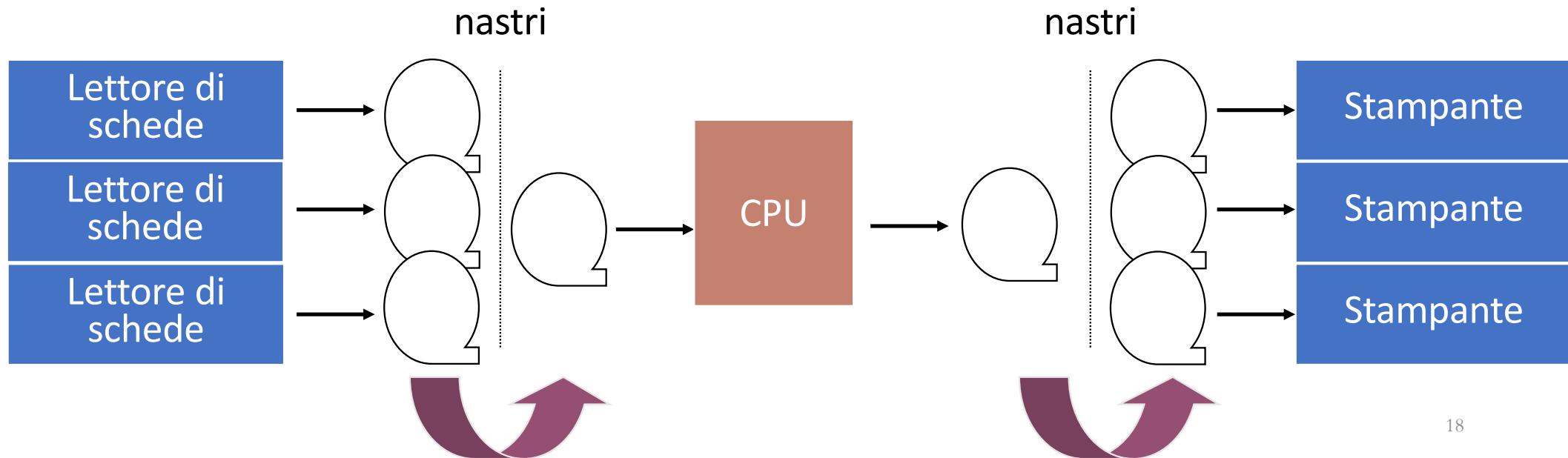


Batch vs. operazioni off-line

Batch tradizionale



Operazioni off-line (n lettori)



Sovrapposizione di CPU e I/O

Operazioni off-line

Sovrapposizione di I/O e CPU su
macchine indipendenti



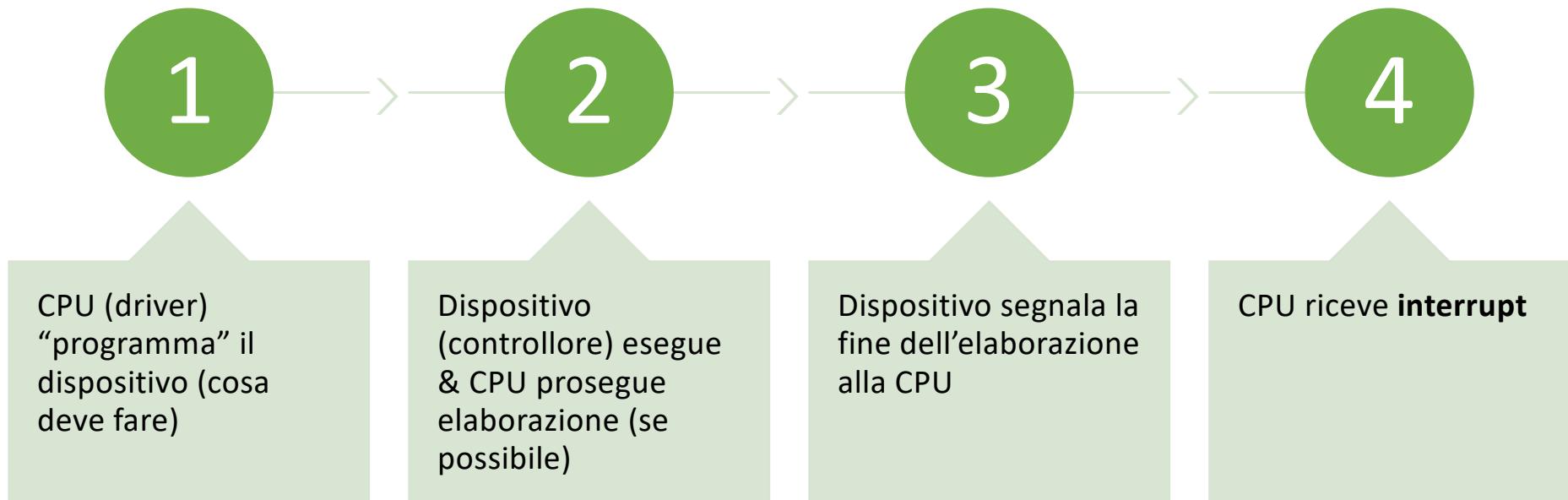
E' possibile sovrapporli
sulla stessa macchina?

Si, ma serve opportuno supporto
architetturale

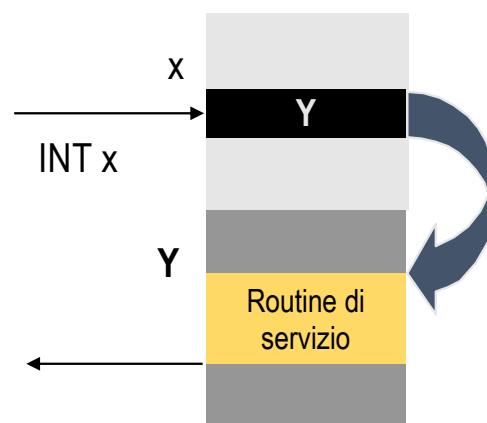
Per sovrapporre CPU e I/O su stessa macchina



Interrupt & I/O



Al ricevimento dell'interrupt



- 1 CPU interrompe l'istruzione corrente salvando lo stato
- 2 Salta a una locazione predefinita X
- 3 "Serve" l'interruzione (trasferimento dati)
- 4 Riprende l'istruzione interrotta

DMA & I/O

Nel caso di dispositivi veloci
(es. dischi):

- Interrupt sono molto frequenti
- Inefficienza

Soluzione:

- DMA (Direct Memory Access)
 - Uno specifico controllore HW (DMA controller) si occupa del trasferimento di blocchi di dati tra I/O e memoria senza interessare la CPU
 - Un solo interrupt per blocco di dati

Buffering

Sovrapposizione di CPU e I/O dello **stesso** job

Dispositivo di I/O
legge/scrive più dati
di quanti richiesti

Utile quando la
velocità dell'I/O e
della CPU sono simili

Nella realtà i
dispositivi di I/O sono
più lenti della CPU
• Miglioramento marginale

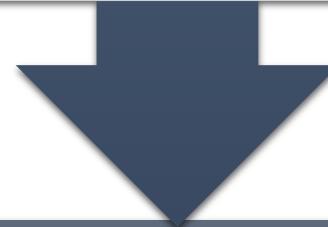
Necessario passare a
Spooling

(Simultaneous Peripheral
Operations On-line)

Spooling

Sovrapposizione di CPU e I/O
di **job** diversi

Nastri magnetici sono sequenziali
(lettore di schede non può scrivere su un'estremità
del nastro mentre la CPU legge dall'altra)



Introduzione dei dischi magnetici ad
accesso casuale

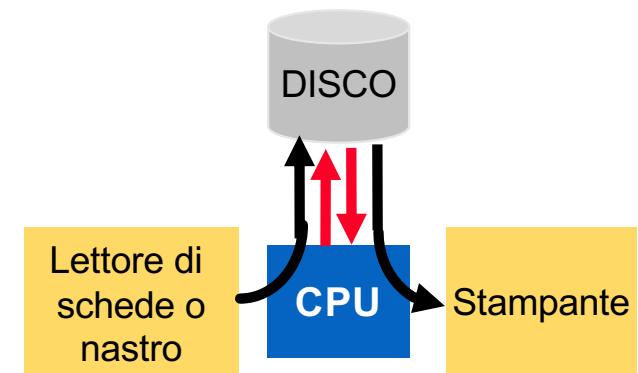
Spooling

Possibile grazie
ad accesso
casuale dei
dischi

- Utilizzo del disco come un grande buffer
- Buffer unico per tutti i job

Nuovo
conceitto “pool
di job”

- Nasce il paradigma “moderno” di programma su disco (che viene caricato in memoria)
- Nasce il concetto di job scheduling
 - Chi deve/può essere caricato sul disco?



3a Generazione (1965-1980)

Un singolo job non potrà mai tener sufficientemente occupata la CPU

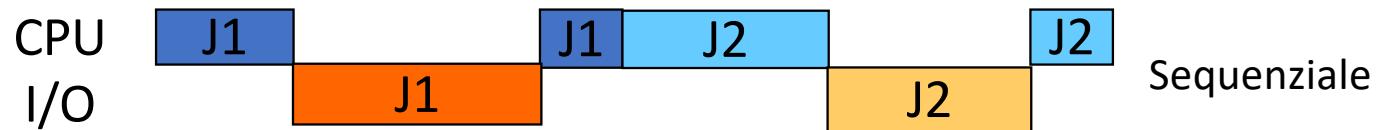
Necessaria la “competizione” di più job

- presenza di più job in memoria
- Sfruttamento delle fasi di attesa (I/O) per l'esecuzione di un nuovo job

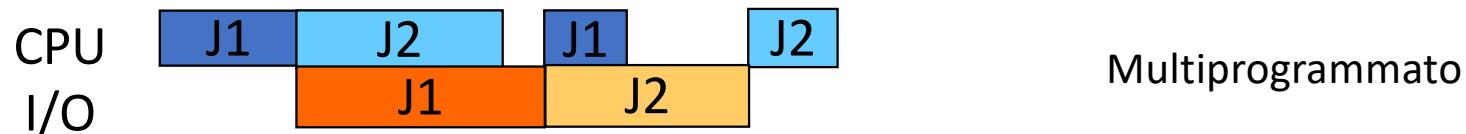


Introduzione della multiprogrammazione e dei circuiti integrati

Multiprogrammazione

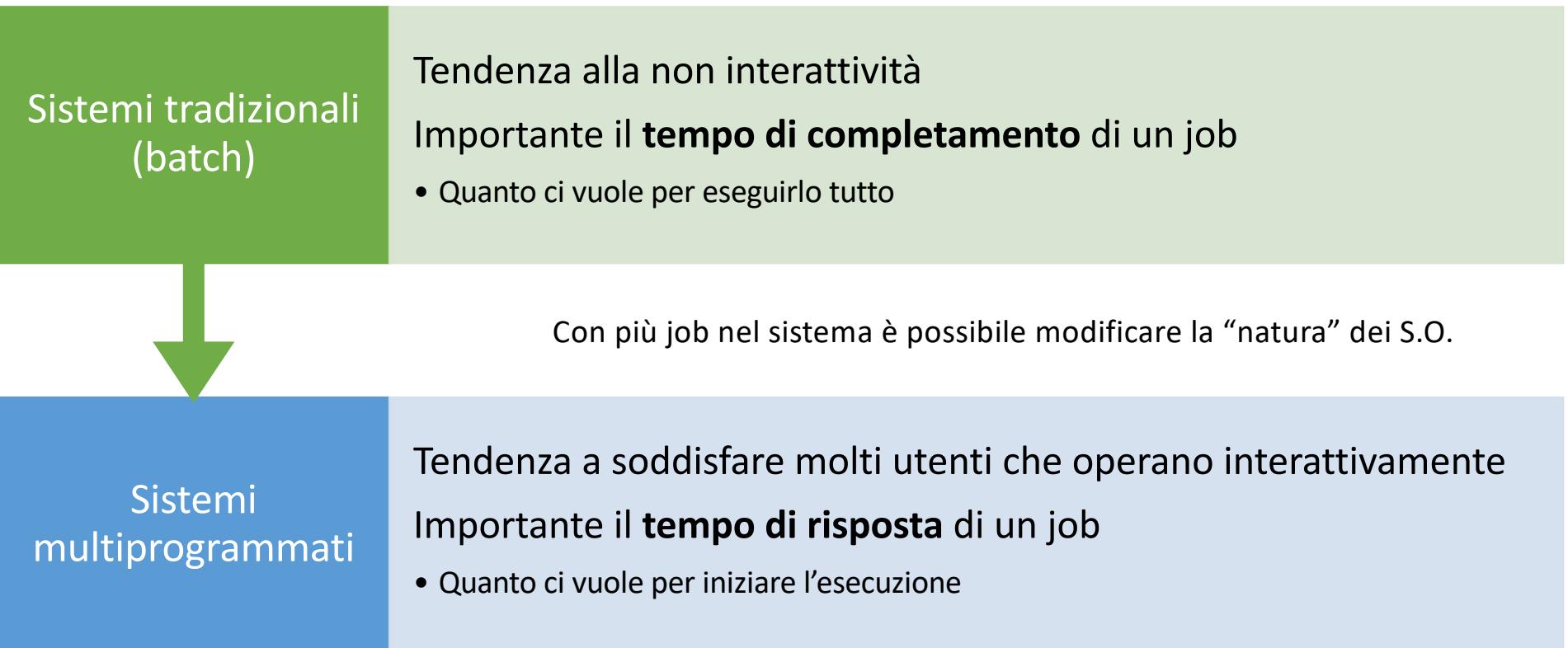


Sequenziale



Multiprogrammato

Batch vs. Multiprogrammazione



Time sharing (multitasking)

Time sharing = condivisione del tempo

- Estensione logica della multiprogrammazione
- L'utente ha l'impressione di avere la macchina solo per sé
- Migliora la interattività (gestione errori, analisi risultati, ...)

Nascita dei sistemi “moderni”

- Tastiera
 - Decisioni sull'evoluzione del sistema basate su comandi utente
 - Comandi brevi vs. job lunghi
- Monitor
 - Output durante l'esecuzione
- File system
 - Astrazione del sistema operativo per accedere a dati e programmi

Protezione

Nei sistemi originari
non si poneva il
problema della
condivisione

Però l'esecuzione di un
programma può
influenzare
l'esecuzione degli altri

Esempio

Ciclo infinito

- Previene l'uso della CPU di altri programmi
- Oppure il programma legge più dati di quanti dovrebbe
- ...

Tre tipi di protezione

I/O

Programmi diversi non devono usare I/O contemporaneamente

Memoria

Programma non può leggere/scrivere in una zona di memoria che non gli “appartiene”

CPU

Prima o poi il controllo della CPU deve tornare al S.O.

Protezione dell'I/O

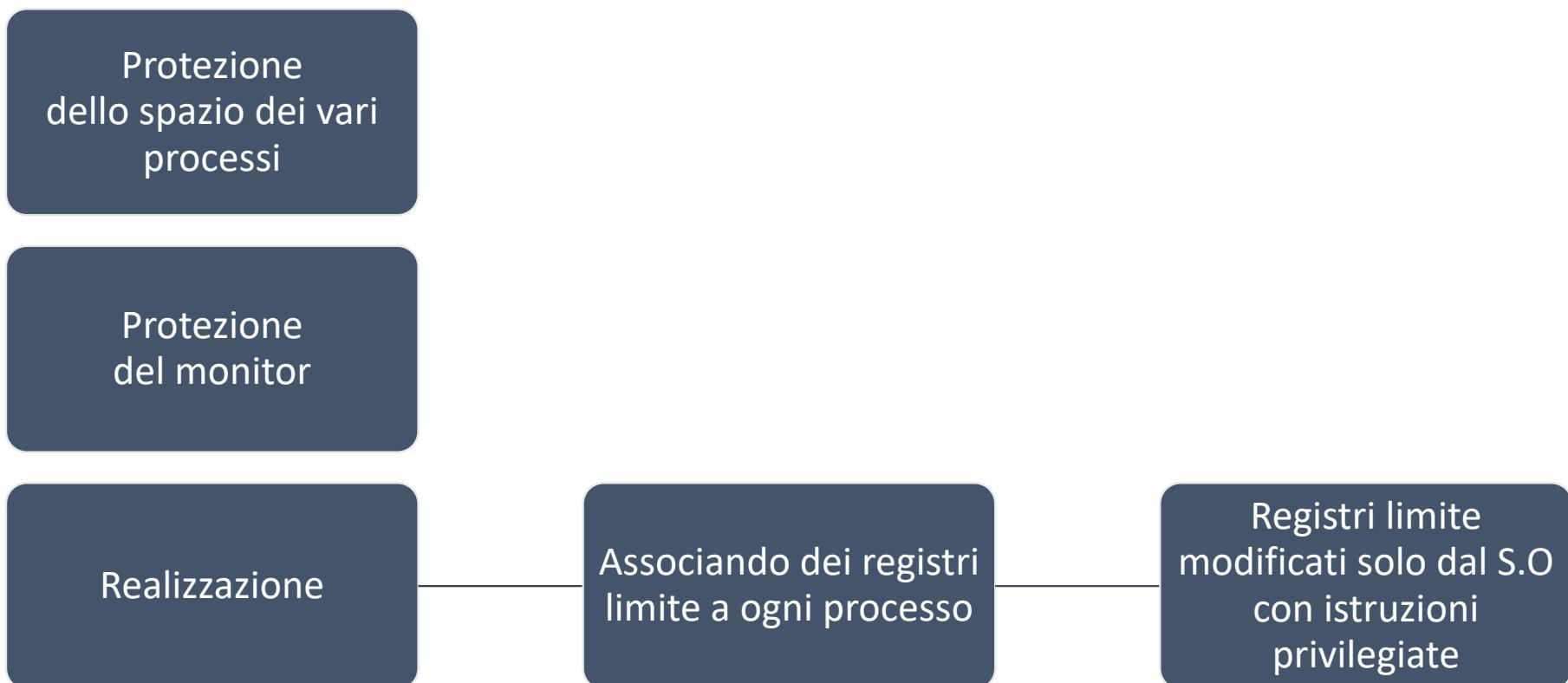
Realizzata tramite il meccanismo del **modo duale** di esecuzione

- **Modo USER**
 - I job non possono accedere direttamente alle risorse di I/O
- **Modo SUPERVISOR (KERNEL)**
 - Il sistema operativo può accedere alle risorse di I/O

Tutte le operazioni di I/O sono privilegiate

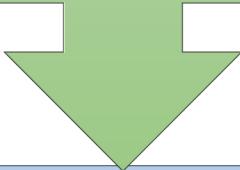
- Sequenza di operazioni (per accesso ad I/O)
 - Istruzioni per accesso ad I/O invocano delle system call
 - System call = interrupt software che cambia la modalità di esecuzione da USER a SUPERVISOR
 - Al termine della system call il S.O. ripristina la modalità USER

Protezione della memoria



Protezione della CPU

Garanzia che il S.O. mantenga il controllo del sistema



Realizzata tramite timer

Associato ad ogni job

Alla scadenza, il controllo ritorna al monitor

4a Generazione (1980-????)

S.O. per PC e workstation

- Uso “personale” dell’elaboratore

S.O. di rete

- Separazione logica delle risorse remote
- Accesso risorse remote ≠ accesso risorse locali

S.O. distribuiti

- Non-separazione logica delle risorse remote
- Accesso risorse remote = accesso risorse locali

S.O. real-time

- Vincoli sui tempi di risposta del sistema

S.O. embedded

- Per sistemi per applicazioni specifiche

Riassunto

1^a generazione

- Device driver
- Librerie
- Batching
- Automatic job sequencing
- Off-line processing
- Sovrapposizione CPU e I/O (buffering e spooling)

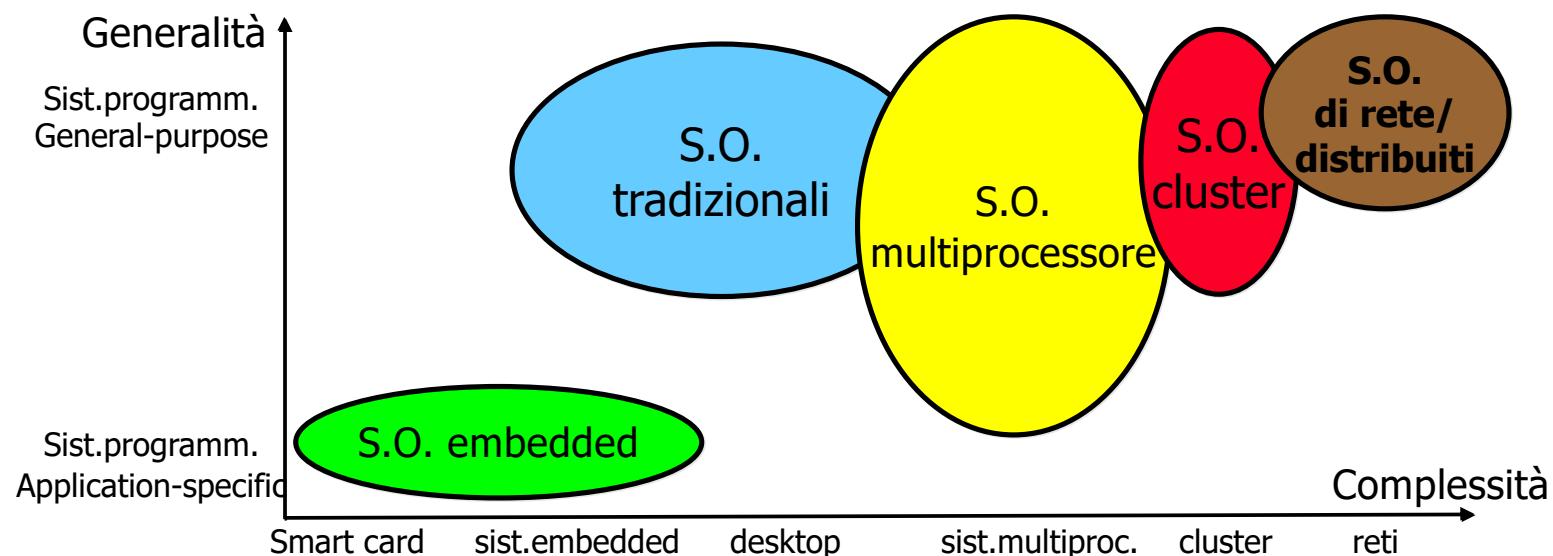
2^a generazione

- Multiprogrammazione
- Time sharing

3^a generazione

Lo spazio dei S.O.

Real-time = dimensione ulteriore!



Esempi di sistemi operativi

OS/360

- S.O. per sistemi batch multiprogrammati (IBM 360)

CP/M

- S.O. monoprogrammato, primo sistema per PC IBM

MS-DOS

- S.O. monoprogrammato, rimpiazzò CP/M

UNIX

- S.O. multiprogrammato, time-sharing, multiutente

MacOS/Windows/Linux

- S.O multiprogrammati, basati su interfaccia grafiche (paradigma WIMP - Window Icon Mouse Pointer)

Valutazione

- <https://kahoot.it>