

ONE PIECE



3 - ENIES LOBBY

Enies Lobby

Nico Robin

Ohara

Imperatori

Gen. - Feb. - Mar.

Introduzione

Storia

Architetture

Processi



NICO ROBIN



MEMORIA

"La storia dell'informatica è la storia del tentativo di usare al meglio risorse estremamente costose."

(Andrew S. Tanenbaum)



Storia dei Sistemi Operativi

Sistemi Operativi

Prof. Andrea Bianco

Storia dei Sistemi Operativi

Argomenti del capitolo

- Introduzione
- Generazione 1 - Tubi a vuoto
- Generazione 2 - Transistor
- Generazione 3 - Circuiti integrati
- Generazione 4 - Circuiti integrati LSI/VLSI (Personal Computer)

Generazione 3 - Circuiti integrati

Storia dei Sistemi Operativi

Storia dei Sistemi Operativi

Generazione 3

Circuiti integrati (1965-1980)

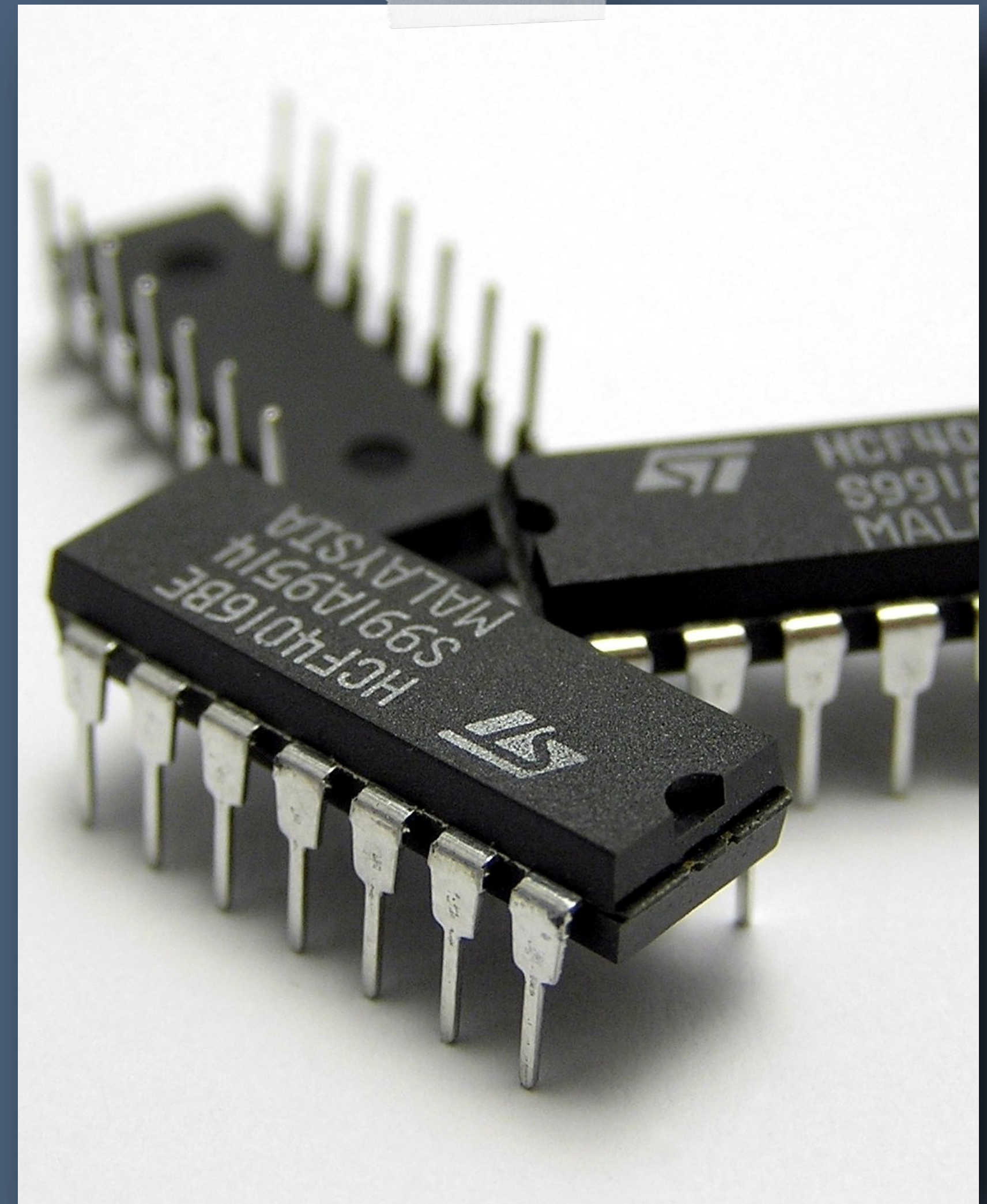
Uso

Le macchine vengono ancora usate per calcoli scientifici, che richiedono parecchia CPU, ma anche per elaborazione di dati commerciali, che richiedono meno CPU ma più I/O.

Hardware

Circuiti integrati, introduzione degli hard disk e dei terminali (terminale = tastiera + video).

Microchip



Storia dei Sistemi Operativi

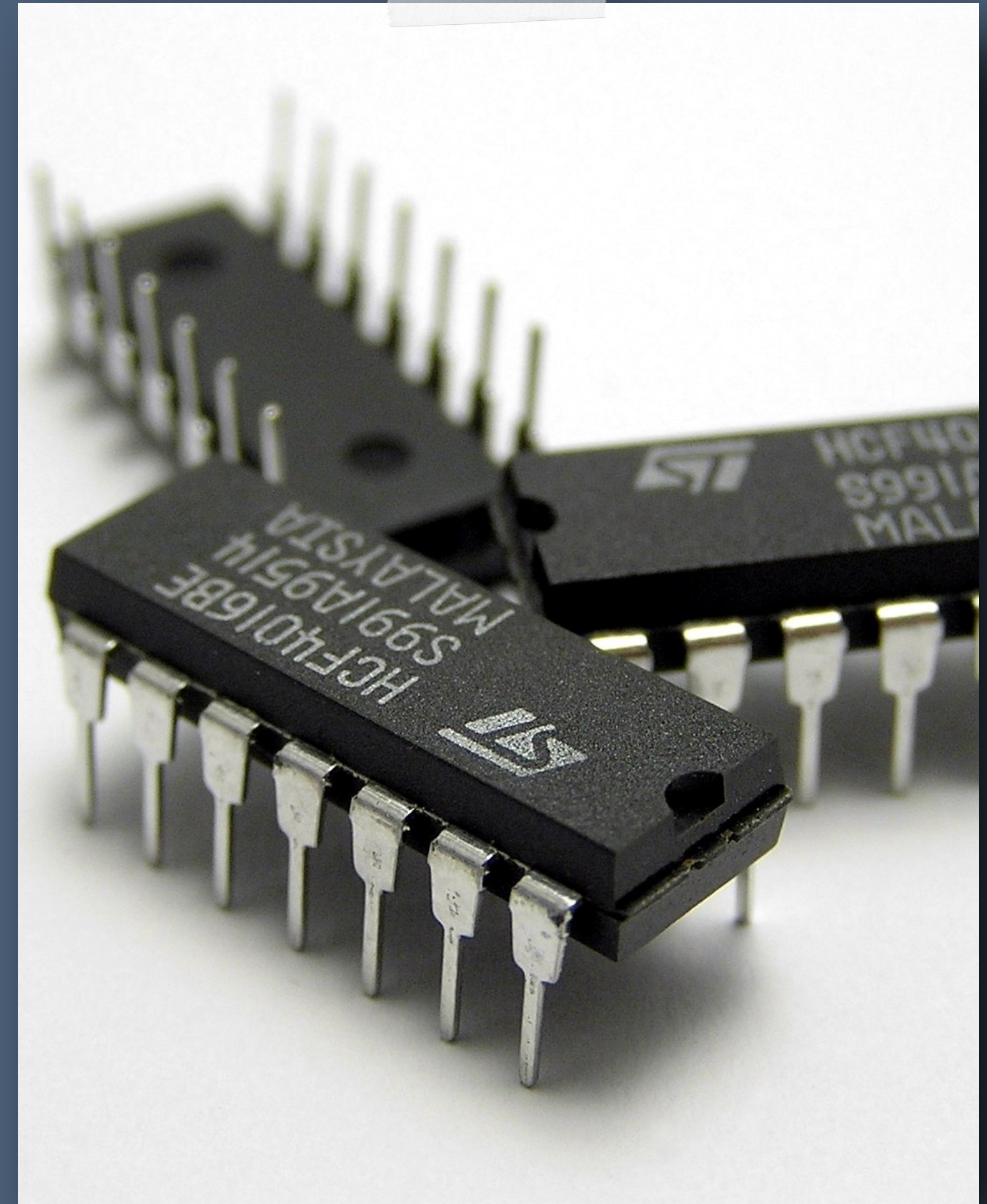
Generazione 3

Programmazione

Linguaggi ad alto livello (C, script di shell,..), favorita dall'introduzione degli editor.

Sistema Operativo

Interattivi, con multiprogrammazione o timesharing.



Storia dei Sistemi Operativi

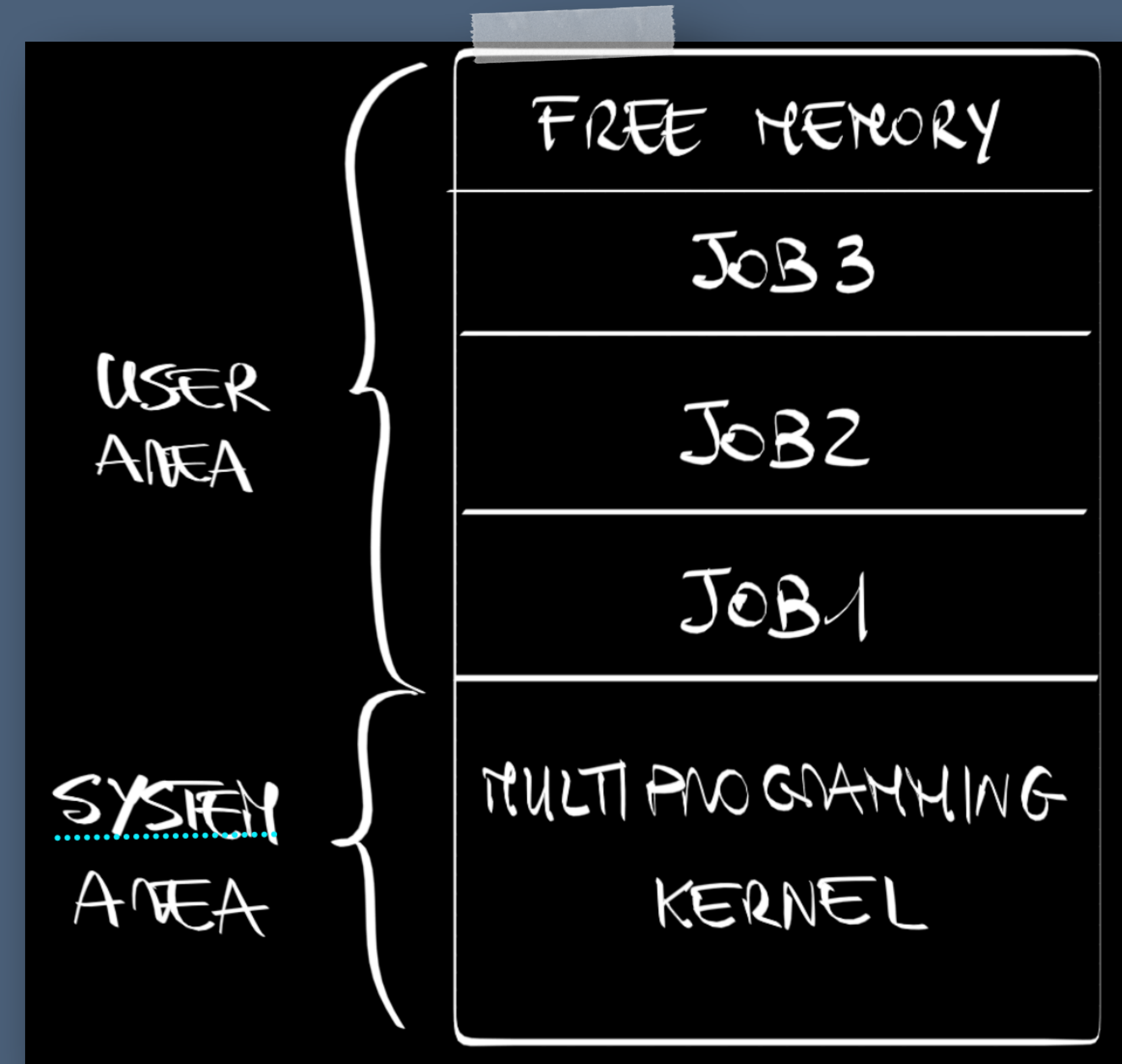
Generazione 3

Multiprogrammazione

Più job vengono caricati in memoria, che viene quindi partizionata.

La CPU viene tolta a un job solo quando termina completamente la sua esecuzione.

L'hardware deve consentire a CPU e dispositivi di lavorare in parallelo.



BUG

La CPU viene tolta a un job solo quando termina completamente la sua esecuzione.

Correzione:

Quando un job inizia un'operazione di I/O, la CPU non rimane inattiva (idle), ma viene assegnata ad un altro job.



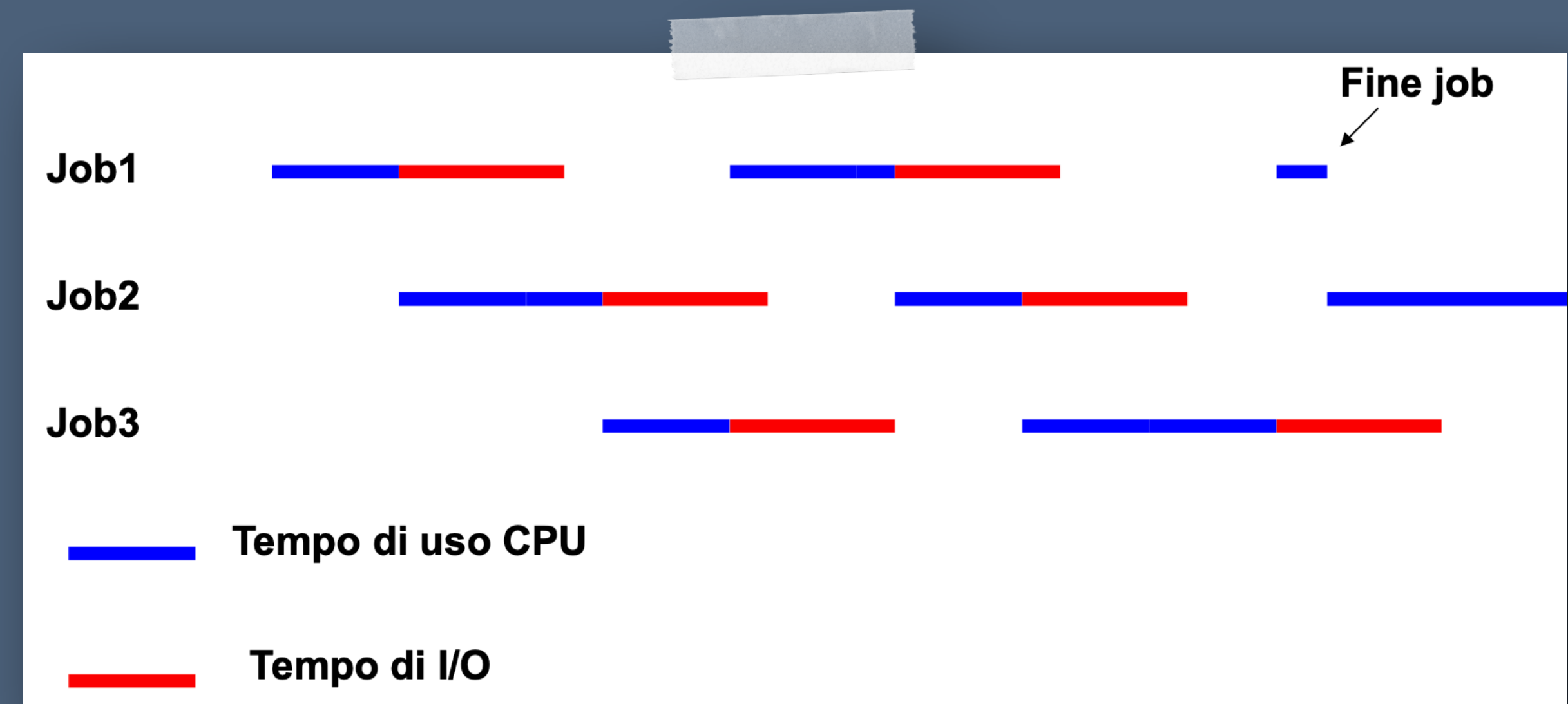
Storia dei Sistemi Operativi

Generazione 3

Multiprogrammazione

Quando un job ottiene la CPU, la mantiene finché non inizia un'operazione di I/O.

Questo schema richiede di avere più job contemporaneamente in memoria.



Storia dei Sistemi Operativi

Generazione 3

CPU Scheduling

Quando il job in esecuzione avvia un'operazione di I/O, la CPU deve essere assegnata ad un job tra quelli in memoria non impegnati in attività di I/O.

E' necessaria una politica di scheduling per scegliere il job da eseguire tra quelli eseguibili, cioè tra quelli caricati in memoria che non stanno eseguendo I/O.

Storia dei Sistemi Operativi

Generazione 3

Gestione della memoria

Ogni job deve poter accedere ai propri dati.

Con i registri LBR e UBR i job possono accedere liberamente a tutta la memoria fisica, purché non accedano alla system area.

BUG

Con i registri LBR e UBR i job possono accedere liberamente a tutta la memoria fisica, purché non accedano alla system area.

Correzione:

Nessun job deve riuscire ad accedere ai dati degli altri. L'hardware deve offrire meccanismi a supporto di tale compito (capitolo Memoria).



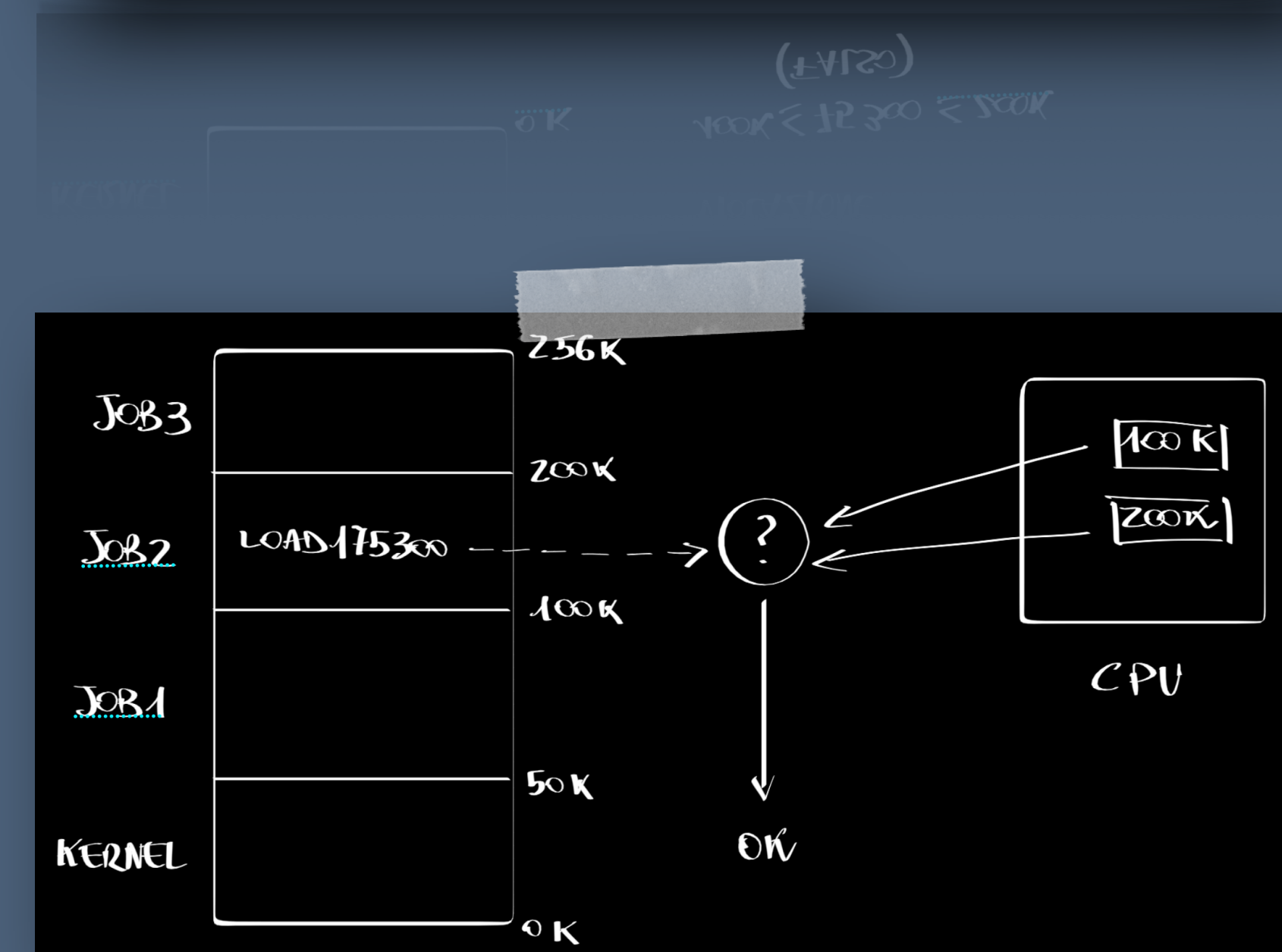
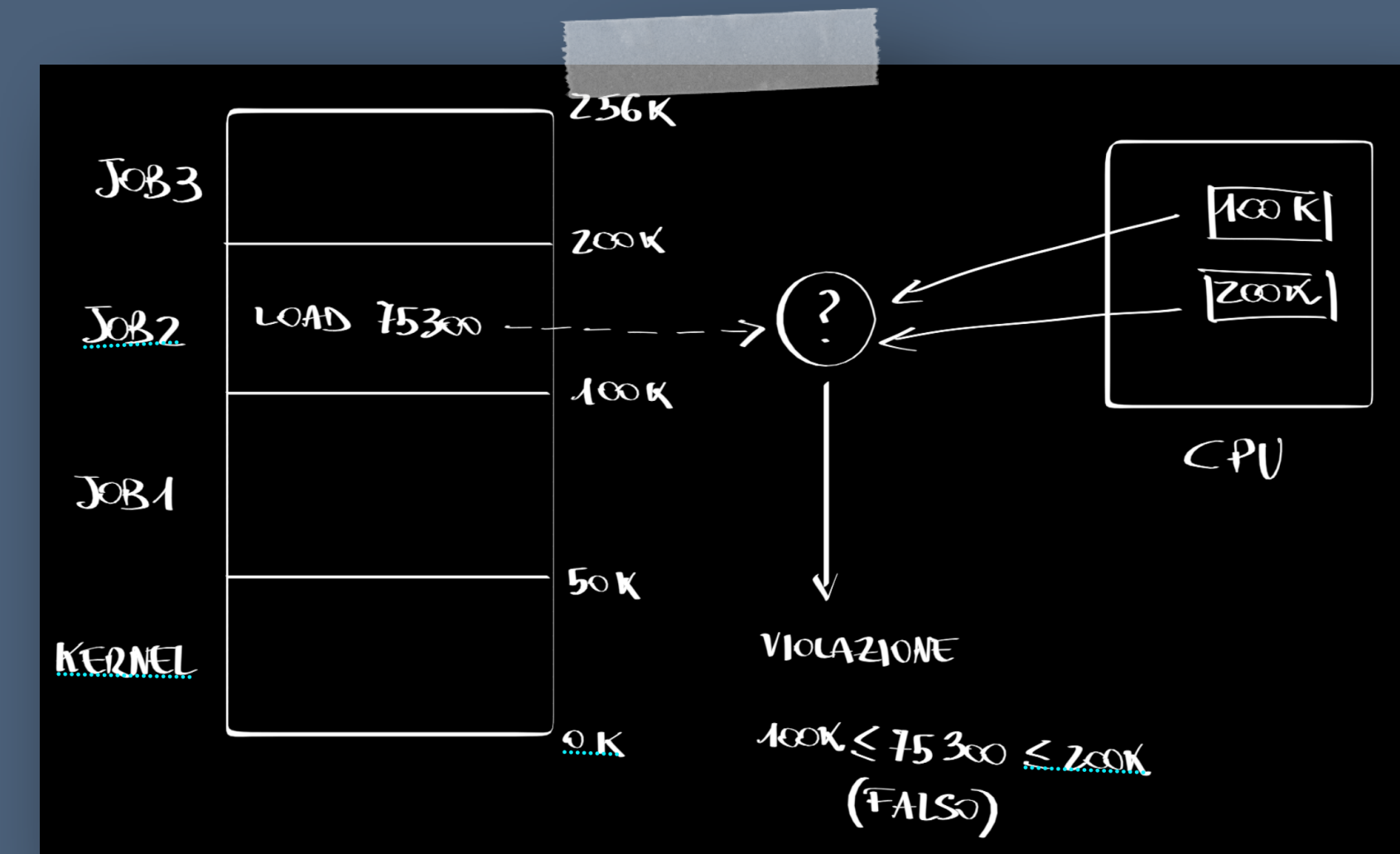
Storia dei Sistemi Operativi

Generazione 3

Gestione della memoria

Esempio:

- la CPU ha i registri Lower Bound Register (LBR) ed Upper Bound Register (UBR);
- gli indirizzi non compresi tra LBR e UBR non sono validi, cioè causano un'interruzione del programma (capitolo Architetture);
- le istruzioni per modificare i valori di LBR e UBR sono privilegiate (disponibili solo in modalità kernel);
- i valori di LBR e UBR vengono modificati dal SO quando inizia l'esecuzione di un job.



Storia dei Sistemi Operativi

Generazione 3

Gestione dispositivi

Ogni job deve poter usare i dispositivi.

Nessun job deve poter interferire sull'uso dei dispositivi da parte degli altri job.

Due modalità principali:

- **partitioning**: ad ogni job vengono assegnati i dispositivi staticamente (esempio, porzione riservata del disco).

Uso poco efficiente dei dispositivi.

- **pooling**: ad ogni job vengono assegnati i dispositivi dinamicamente.

Uso più efficiente dei dispositivi, ma il SO è più complesso.

ANSIA

Cos'è un batch?

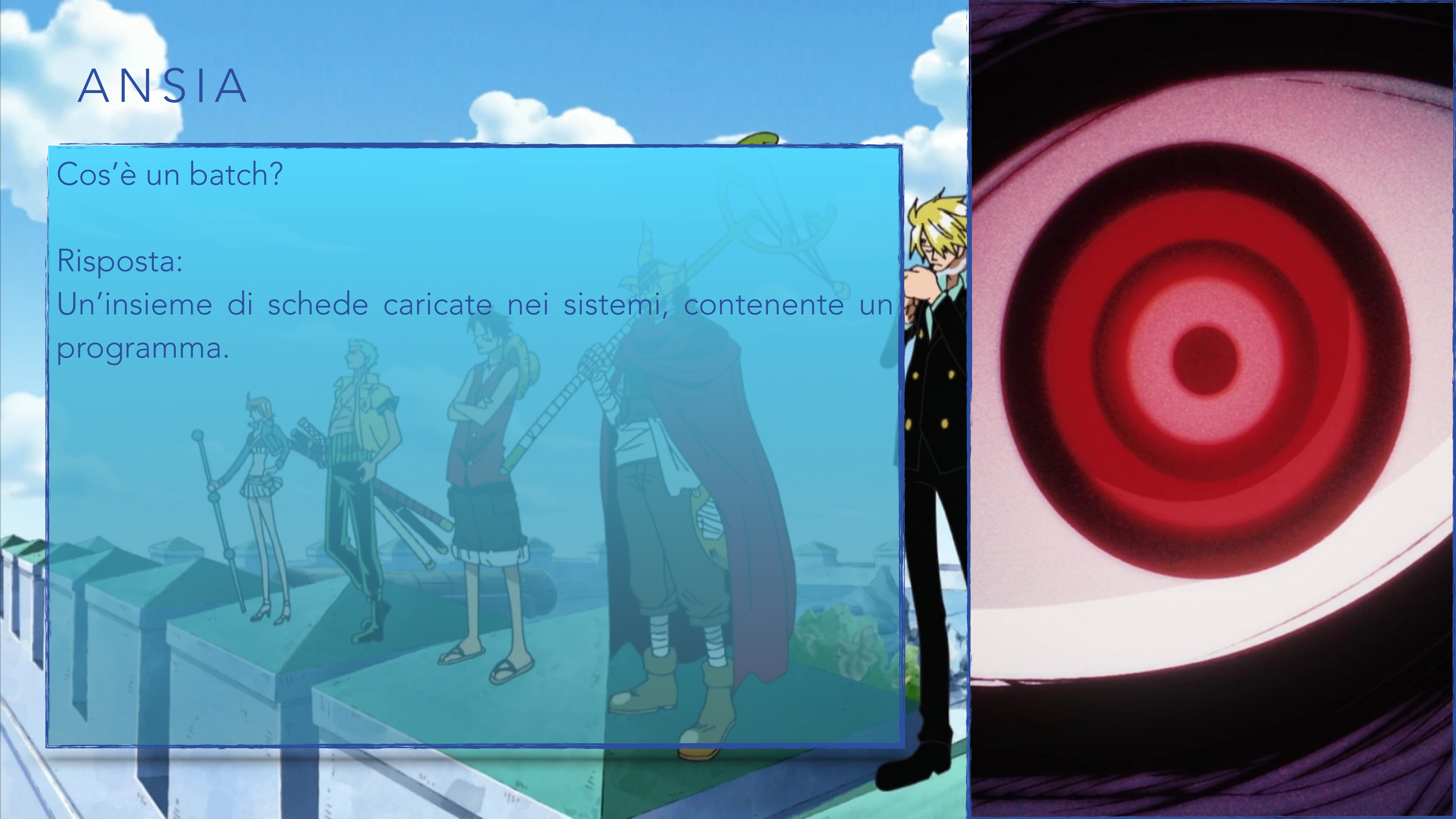


ANSIA

Cos'è un batch?

Risposta:

Un'insieme di schede caricate nei sistemi, contenente un programma.



Storia dei Sistemi Operativi

Generazione 3

Requisiti hw

- L'hardware deve consentire alla CPU ed ai dispositivi di I/O di lavorare in parallelo (cioè contemporaneamente).

DMA (Direct Memory Access)

- Quando un dispositivo di I/O ha terminato di eseguire un'operazione per conto di un job, il job deve essere inserito nell'elenco dei job schedulabili.

Interrupt

(capitolo Architetture)

Storia dei Sistemi Operativi

Generazione 3

Definizioni

- **Throughput**: il rapporto tra il numero di programmi eseguiti ed il tempo.
- Programma **CPU-bound**: uso intenso CPU, poco I/O.
- Programma **I/O-bound**: tanto I/O, poco uso CPU.

Per massimizzare il throughput conviene dare priorità ai programmi CPU-bound rispetto a quelli I/O-bound.

BUG

Per massimizzare il throughput conviene dare priorità ai programmi CPU-bound rispetto a quelli I/O-bound.

Correzione:

Per ottimizzare il throughput, è opportuno che lo scheduler privilegi i programmi I/O-bound rispetto a quelli CPU-bound, per far sì che siano frequenti gli usi concorrenti di I/O e CPU.

Privilegiare i programmi CPU-bound non consentirebbe di migliorare il throughput rispetto ai sistemi batch.



Storia dei Sistemi Operativi

Generazione 3

Definizioni

- **Program priority:** ad ogni programma è assegnata una priorità, che viene considerata in fase di scheduling per scegliere il programma da schedulare.
- **Preemption:** in alcuni casi, la CPU viene tolta forzatamente ad un programma in esecuzione (esempio precedente).

Per ottimizzare il throughput:

- i programmi CPU-bound devono avere priorità inferiore rispetto ai programmi I/O-bound;
- quando un programma I/O-bound termina un'operazione di I/O, se è in esecuzione un programma CPU-bound questo è preempted (subisce la preemption).

Storia dei Sistemi Operativi

Generazione 3

Definizioni

- **Spooling**: SPOOL (Simultaneous Peripheral Operation On Line).

I programmi possono essere caricati su disco: non appena un job termina e libera una partizione di memoria, il SO può assegnarla ad uno dei job su disco, che diventa pertanto eseguibile. Quale job andrebbe scelto?

Per migliorare il throughput, è bene avere sempre sia programmi CPU-bound sia programmi I/O-bound in memoria.

Anche i dispositivi di output (esempio, stampanti) possono essere gestiti in modo analogo.

Di fatto, non servono più le macchine dedicate alla preparazione dei batch ed alla stampa dei risultati.

Storia dei Sistemi Operativi

Generazione 3

Timesharing

Estensione della multiprogrammazione: ad ogni job viene assegnato un quanto di tempo (time slice), allo scadere del quale la CPU viene assegnata ad un altro job anche in assenza di I/O o preemption.

Adatto per sistemi dotati di più terminali su cui lavorano utenti diversi: Ogni utente ha l'impressione di interagire continuamente con la macchina, in quanto nessun job di altri utenti può monopolizzare la CPU. (Ricordiamo che siamo in un periodo storico in cui la CPU è estremamente costosa e va condivisa da vari utenti).

Il SO, con l'aiuto dell'hardware, deve garantire la protezione dei dati di ogni singolo utente.

Storia dei Sistemi Operativi

Generazione 3

Timesharing

Il timesharing penalizza il throughput, ma migliora i tempi di risposta offerti agli utenti interattivi, per esempio utenti che compilano ripetutamente programmi e li testano inserendo dati da tastiera. Si va a privilegiare la user convenience rispetto all'efficienza dell'uso delle risorse.

Lo scheduler non può essere basato sulle priorità, per evitare che i programmi di alcuni utenti vengano penalizzati. I programmi vanno schedulati introducendo il concetto di turno, con la tecnica del round robin. Non indaghiamo oltre.

Storia dei Sistemi Operativi

Generazione 3

Multiprogrammazione - Timesharing

Multiprogrammazione: priorità, preemption, throughput.

Timesharing: round robin, time slice, tempi di risposta.

Sottrarre la CPU ad un job ed assegnarla ad un altro (context switch) è un'operazione costosa in termini di tempo. E' necessario:

- schedulare il job, tenendo conto delle priorità/turno
- effettuare alcune operazioni (capitolo Processi)

Durante il context switch la CPU continua a svolgere lavoro utile per i job.

Avere context switch frequenti garantisce maggior interattività agli utenti ma accresce l'overhead a danno del throughput: è necessario trovare un equilibrio.

BUG

Durante il context switch la CPU continua a svolgere lavoro utile per i job.

Correzione:

Durante il context switch la CPU non lavora per nessun job, quindi sotto un certo punto di vista non effettua lavoro utile. —> Overhead

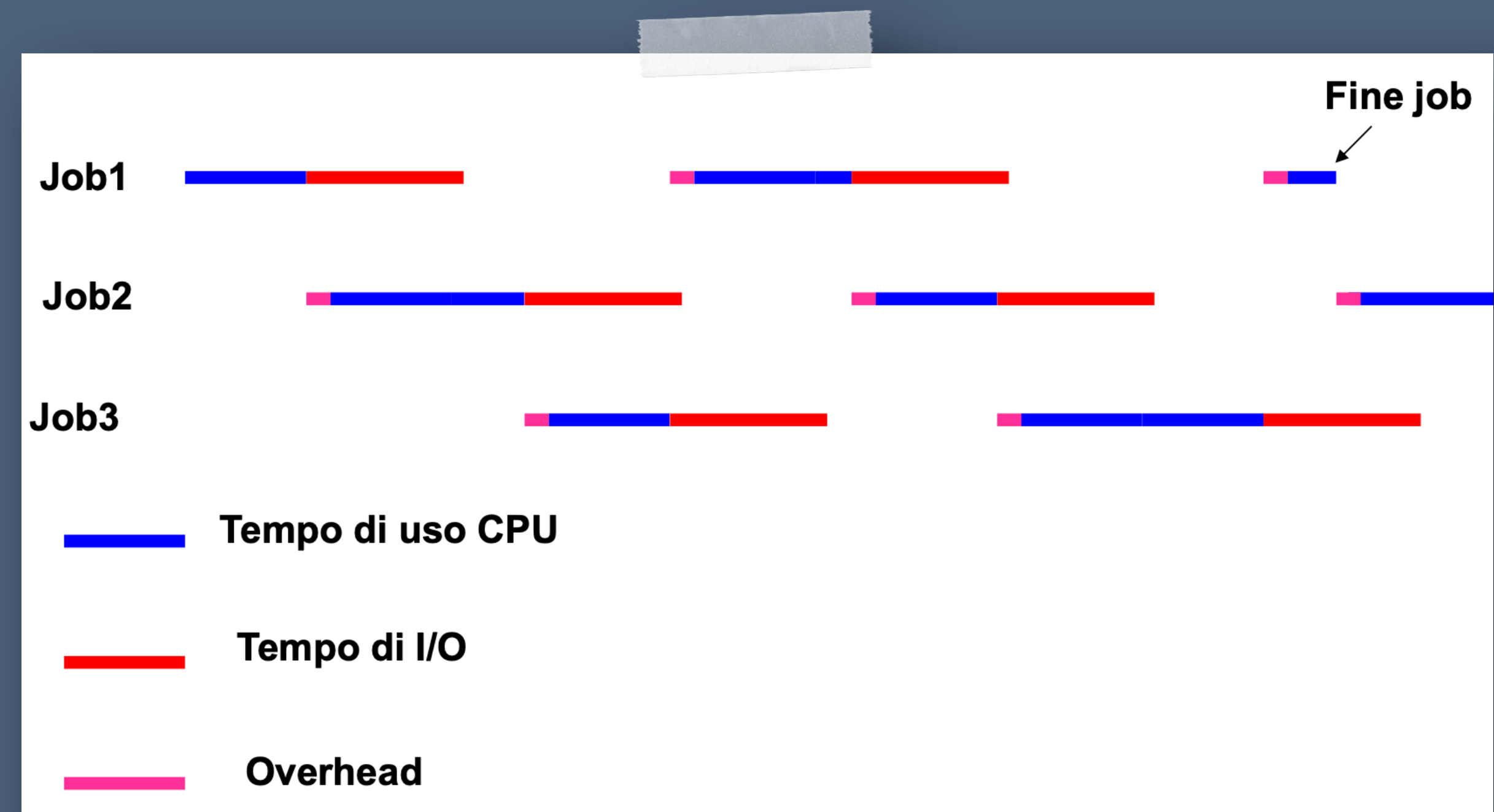


Storia dei Sistemi Operativi

Generazione 3

Multiprogrammazione - Timesharing

Teniamo conto dell'overhead.

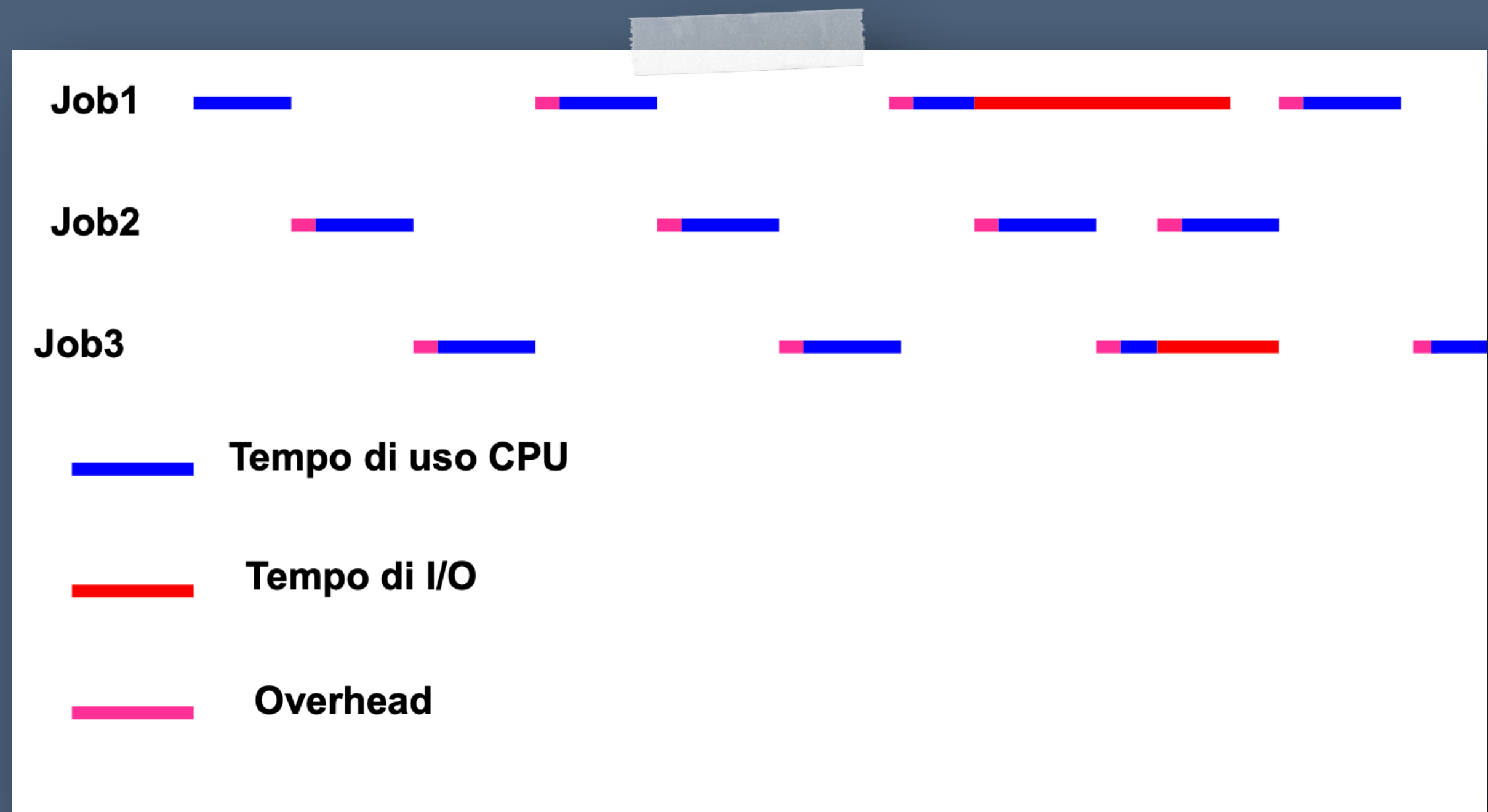


Storia dei Sistemi Operativi

Generazione 3

Multiprogrammazione - Timesharing

Tre job con timesharing.



Storia dei Sistemi Operativi

Generazione 3

Esempi

OS/360 IBM, introduce la multiprogrammazione.

CTSS (Compatible Time Sharing System) MIT (1962) per IBM 7094, introduce il timesharing.

MULTICS (MULTiplexed Information and Computing Service) MIT, GE, Bell Labs (AT&T) (1965), pensato per sistemi multiutente, introduce il concetto di processo.

UNIX Bell Labs (Ken Thompson, 1970 per sistemi monoutente, derivato da CTSS e MULTICS, gira inizialmente su PDP-7, poi viene scritto in C e diventa portabile.

Ken Thompson e Dennis Ritchie



Generazione 4 - Circuiti integrati LSI/VLSI (Personal Computer)

Storia dei Sistemi Operativi

Storia dei Sistemi Operativi

Generazione 4

Circuiti integrati LSI/VLSI (1980-...)

Hardware

Circuiti integrati LSI/VLSI

Dagli anni'80 si diffondono i PC: la facilità d'uso diventa un elemento determinante, per questo vengono introdotte le GUI (Graphical User Interface).

Circuito integrato Texas Instruments r/electronics

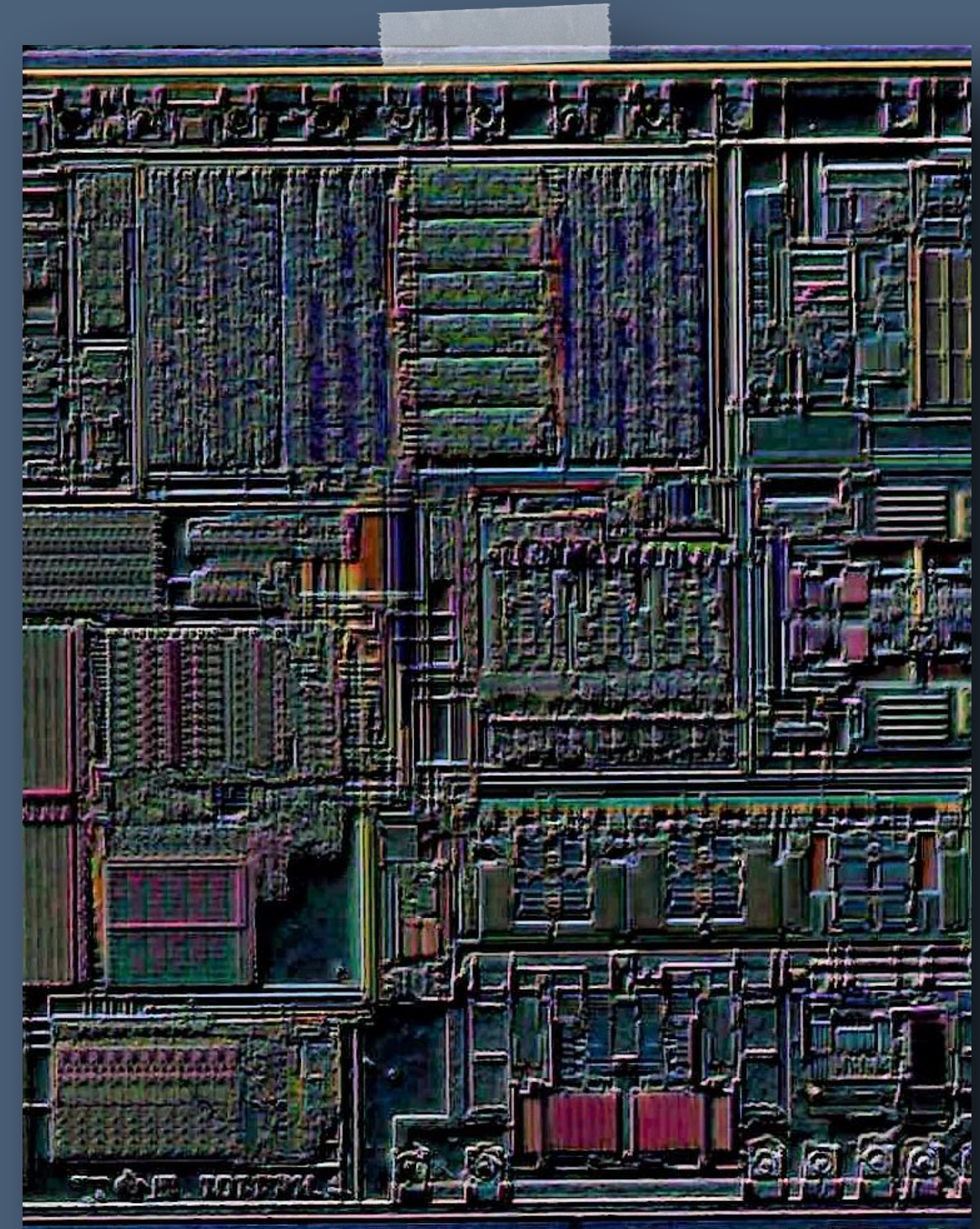


Storia dei Sistemi Operativi

Generazione 4

Dagli anni '90 si diffondono i sistemi operativi distribuiti per la gestione dei sistemi distribuiti. Diventano centrali i concetti di controllo distribuito, trasparenza delle risorse/servizi, remote procedure call (RPC).

Nei sistemi operativi moderni vengono combinate le varie tecniche menzionate nelle slide precedenti, al fine di soddisfare richieste di natura diversa da parte degli utenti.



ESERCIZIO

Disegna una linea temporale che confronti:

- multiprogrammazione
- timesharing

indicando:

- uso della CPU
- uso dell'I/O
- momenti di preemption
- presenza di overhead



MEMORIA

"Un buon progetto nasce sempre da un compromesso tra obiettivi in conflitto."

(Edsger W. Dijkstra)



COMPITI

Spiega in 15 righe massimo:

Perché il timesharing peggiora il throughput ma migliora la user convenience.

Descrivi un sistema reale (UNIX/Linux, Windows, macOS) e indica:

- un esempio di timesharing
- un esempio di preemption
- un caso in cui l'overhead è visibile all'utente

Rispondi: perché non è possibile scegliere un quanto di tempo (time slice) troppo piccolo né troppo grande?

