Sistemi Operativi e Reti di Calcolatori (SOReCa)

Corso di Laurea in *Ingegneria Informatica e Automatica (BIAR)*
Terzo Anno | Primo Semestre
A.A. 2024/2025

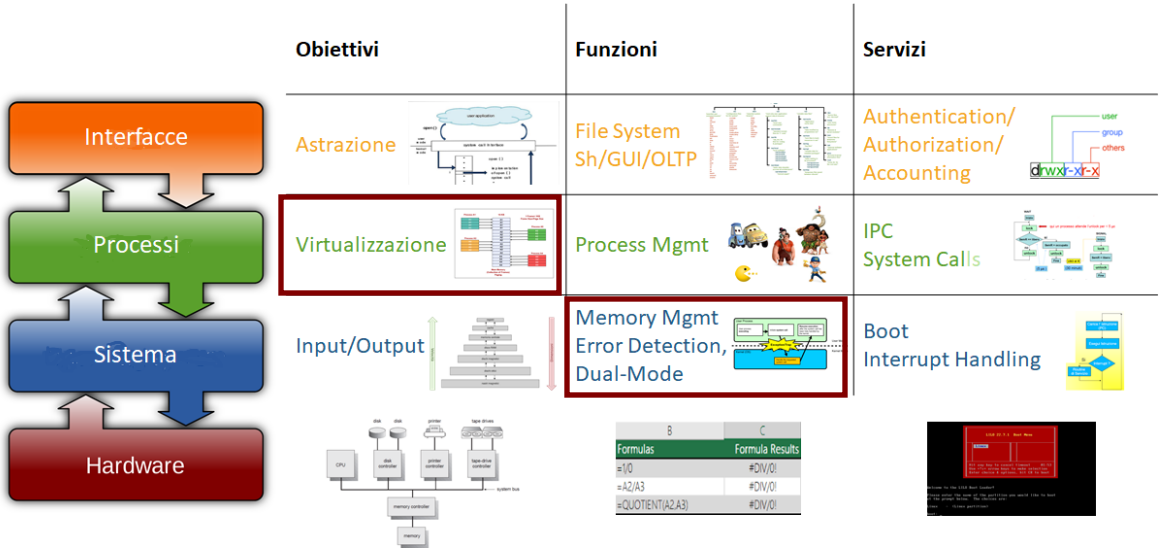
Memoria centrale e virtuale

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Sistemi operativi (3 CFU)

- Il sistema operativo
- Concorrenza e sincronizzazione
- Deadlock
- Inter-process communication (IPC)
- Scheduling
- Memoria centrale e virtuale
- Memoria di massa e File system
- Sicurezza informatica



Lezioni: Settembre - Ottobre

Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

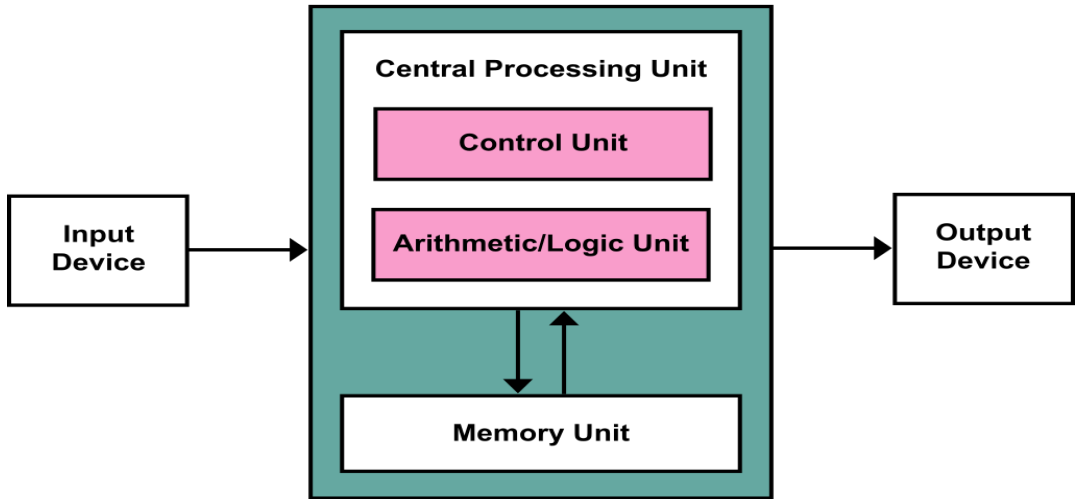
Sapienza
Università di Roma

Gestione della Memoria

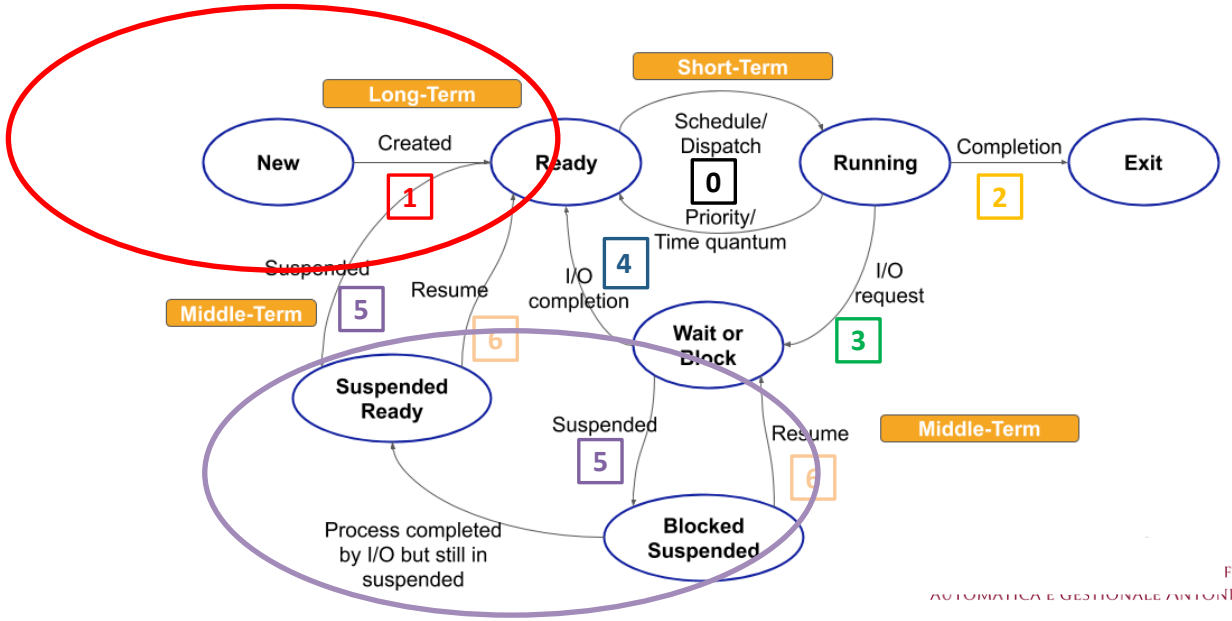
Operating Systems: Memory Management

Finalità

Per essere eseguito, un programma deve essere portato (almeno in parte) in memoria centrale ed “essere attivato come processo” a partire da un indirizzo ➔ **Architettura di von Neumann**



Quando un programma non è in esecuzione, non è strettamente necessario che stia in memoria centrale (*«Lemma di Garibaldi»*).

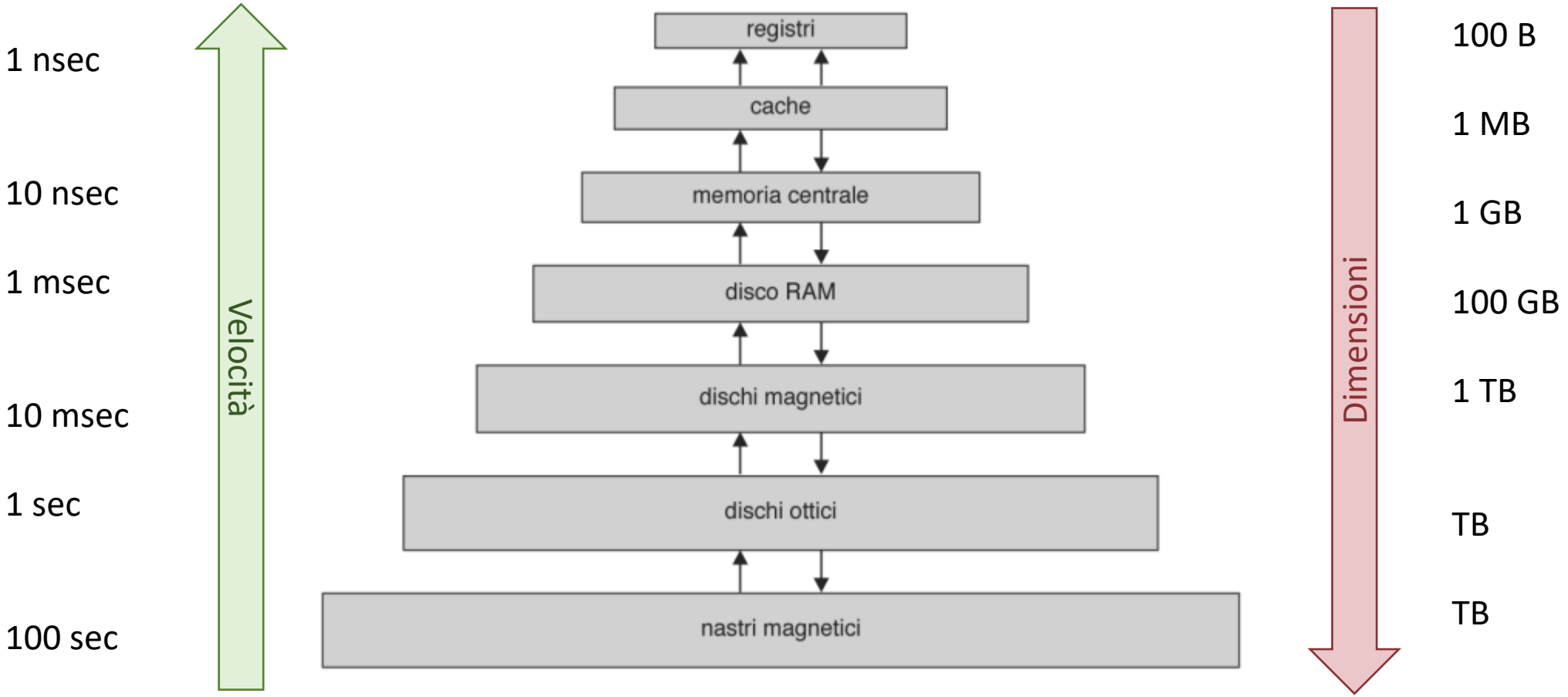


Coda di entrata: processi su disco che sono in attesa di essere caricati in memoria centrale per l’esecuzione ➔ **Long Term Scheduler**

Coda di ri-entrata: processi in swap su disco in attesa di essere caricati in memoria centrale per l’esecuzione ➔ **Medium Term Scheduler**

Operating Systems: Memory Management

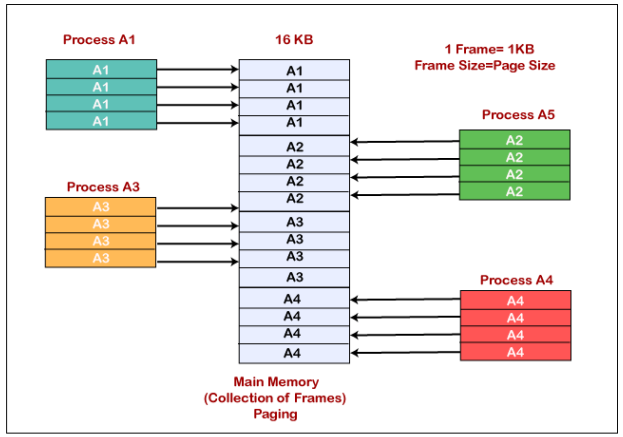
Gerarchie

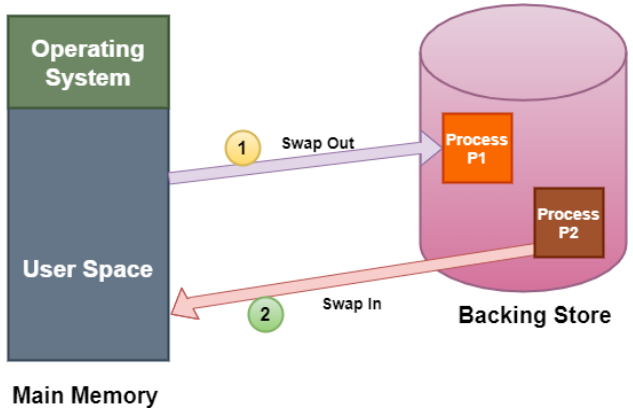


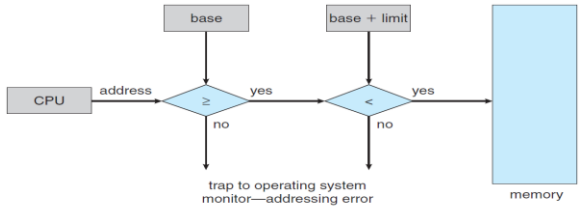
Il Memory Manager ha il compito di usare questa gerarchia creando l'astrazione (illusione) di facile accesso alla memoria.

Operating Systems: Memory Management

Funzioni







Memory Manager: la parte del sistema operativo che gestisce la gerarchia di memoria:

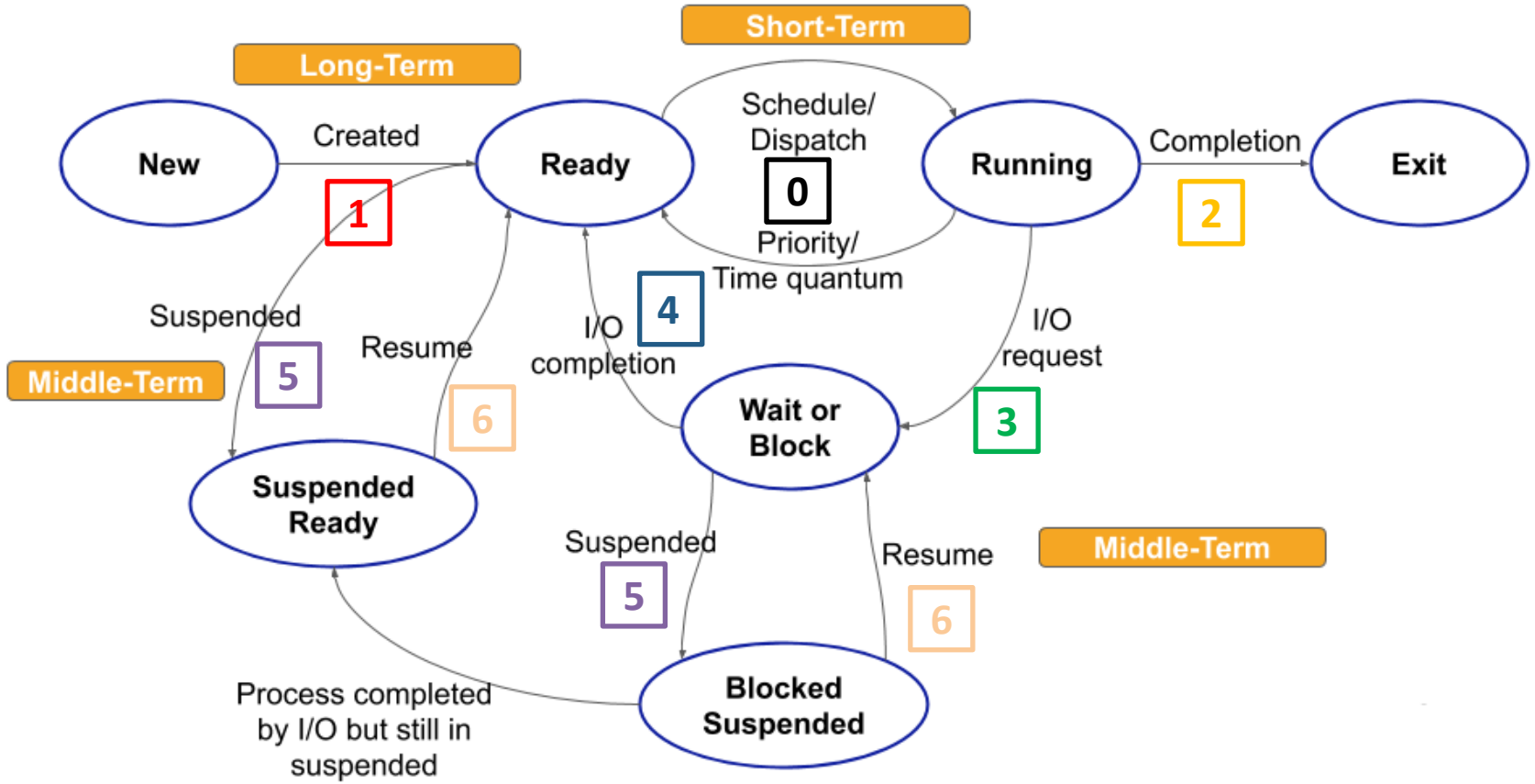
1. **Allocation**: alloccare la memoria ai processi che la richiedono.
2. **Deallocation**: liberare la memoria che non è più in uso.
3. **Use**: tenere traccia della memoria utilizzata.
4. **Swapping**: gestire lo spostamento da (rientro del processo in memoria centrale) e verso il disco (uscita del processo dalla memoria centrale, per mancanza di memoria fisica).
5. **Protect**: Ogni processo ha accesso solo alla porzione di memoria riservatagli dal S.O. Inoltre, possono essere implementati criteri di accesso (es. Read-Only: costanti; Write: Heap; eXecute: Stack).

Dipartimento di Ingegneria Informatica Automatica e Gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management

Coinvolgimento nello Scheduling



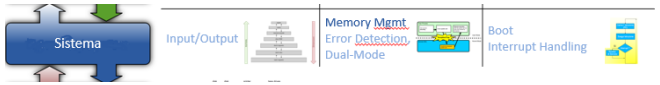
1. **Created**: riuscita allocazione della memoria
2. **Completion**: liberazione della memoria
5. **Swap**: sospensione del processo, con uscita dalla memoria centrale per spazio mancante
6. **Resume**: rientro del processo in memoria centrale

Altre transazioni di stato dei processi, non inerenti il memory management:

- **0.Dispatcher**: alternanza dei processi per ottimizzare CPU usage, Troughput, Turnaround Time, Response Time
- **3.I/O Request/Semaphore**: richiesta di una operazione dal processo (ora in attesa), qualcun altro lo deve sostituire
- **4.I/O Completion**: esecuzione della operazione di I/O, rendendo pronto il processo che l'ha chiesta

Operating Systems: Memory Management

Funzionamento



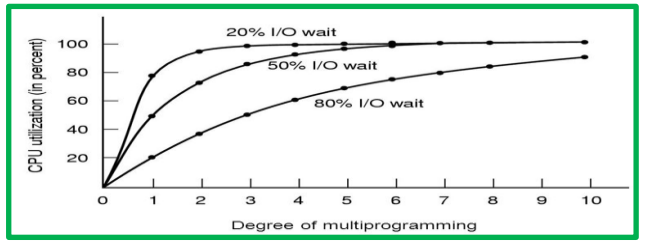
Memory Manager: si è evoluto notevolmente negli anni, elaborando via via concetti e meccanismi per implementare la gestione delle memoria al meglio:

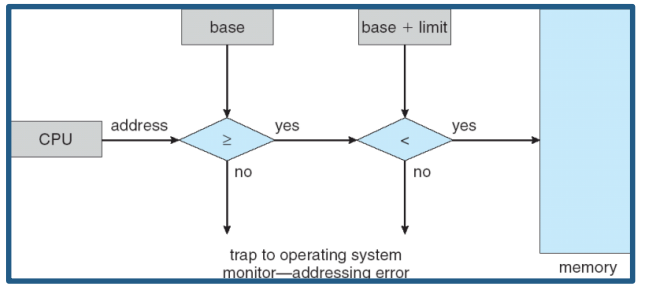
1. **Basic**: **Memory Partitioning**, **Degree of Multiprogramming**, **Base&Limit**
   - **Sistemi Mono-Programmati**: 1 processo alla volta.
   - **Sistemi Multi-Programmati a partizioni fisse (Fixed Partitions)**: più processi pronti in memoria per la elaborazione.

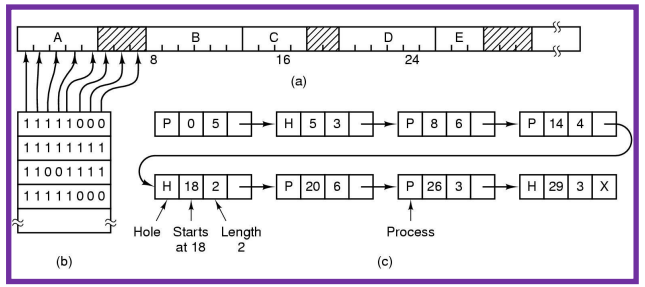
2. **Swapping**: **Linked List**
   caricare un processo per intero in memoria, eseguirlo per un certo tempo e poi segregarlo su disco.

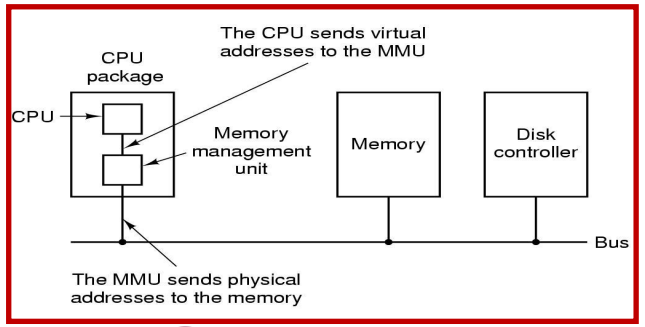
3. **Virtual Memory**: **MMU**, **Paging**, **TLB**, **LRU**, **Working Set**, **Backing Store**,
   gestire lo spostamento da e verso il disco (uscita del processo dalla memoria centrale, per mancanza di memoria fisica).

4. **Segmentation**: **Multidimensional Address Space**, **Different Protection**
   Ogni processo ha accesso solo alla porzione di memoria riservatagli dal S.O. Inoltre, possono essere implementati criteri di accesso (es. Read-Only: costanti; Write: Heap; eXecute: Stack).





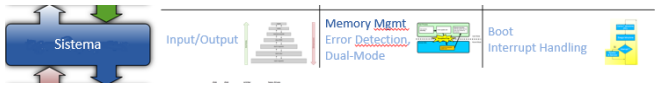




SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

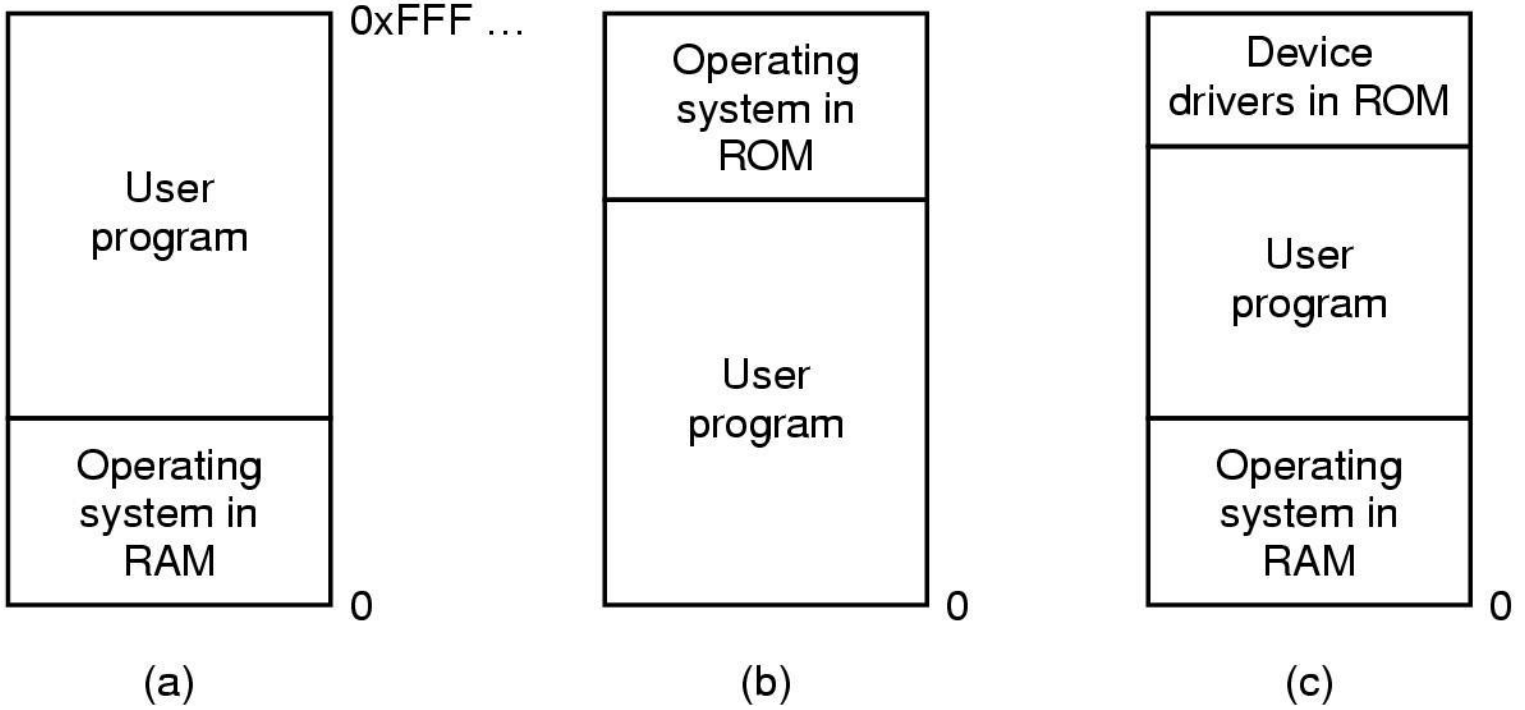
Basic: Memory Partitioning 1/4



Memory Manager: caricamento di un programma alla volta da disco, invocato tramite comando. Oltre al SO stesso.

1 Program at a time

- SO sempre indirizzabile nella memoria ➔ **il SO non è un processo come gli altri**
- Può avere un solo programma in memoria alla volta.
- Un bug nel programma utente può cestinare il sistema operativo: (a) e (c)
- (b) alcuni sistemi embedded **difendono il SO** ponendolo come **Read-Only**
- (c) MS-DOS (primi PC) - parte in ROM chiamata BIOS (Basic Input Output System), il **SO** può essere **ricaricato dal disco**

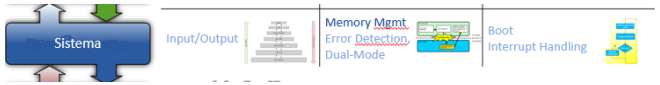


Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management

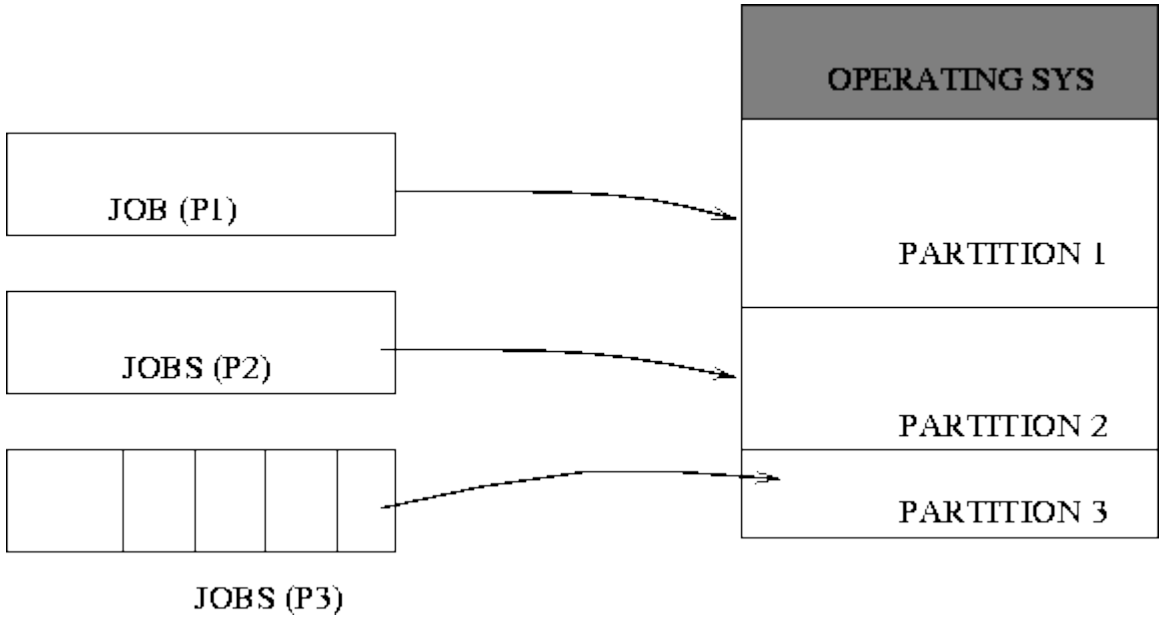
Basic: Memory Partitioning 2/4



Memory Manager: predisposizione delle partizioni (porzione separata della memoria).

Multiprogramming with Fixed Partitions

- La memoria viene divisa in ***n*** partizioni (possibilmente diseguali)➔ **Memory Partitioning** (no Overlap).
- Ogni partizione sarà assegnata ad uno ed uno processo in esecuzione
- Le partizioni (numero e dimensioni) sono determinate prima della esecuzione (es. ad inizio giornata, quando il sistema viene avviato). ➔ operazione di pianificazione **da eseguire con estrema cura**



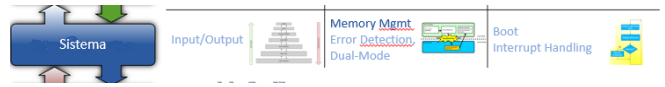
(es. IBM/360, **OS/MFT**: Multiprogramming with a Fixed number of Tasks

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

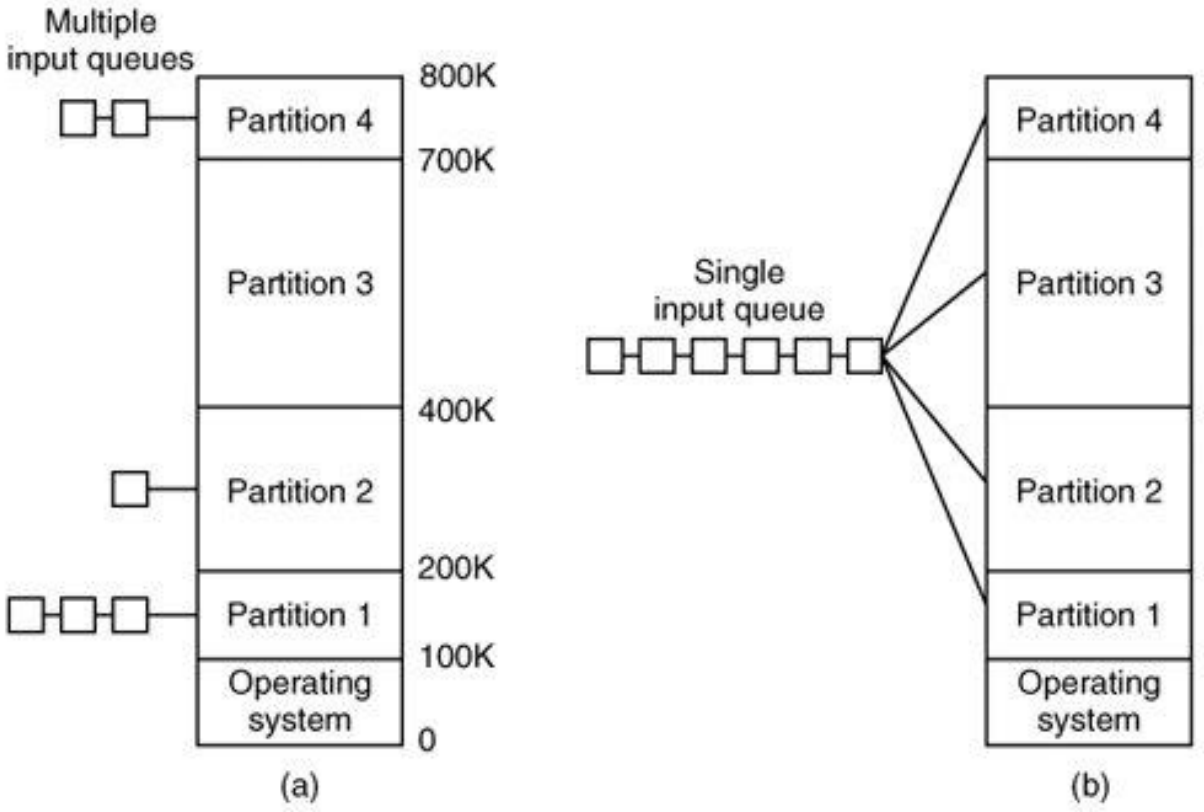
SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

Basic: Memory Partitioning 3/4



Memory Manager: scelta del programma da caricare, su quale partizione (porzione separata della memoria).



Partition Input Queue

- (a) **Separate Input Queues**: classificazione dei job in base allo spazio di memoria presumibilmente richiesto ➔ possibile attesa importante per processi «piccoli»
- (b) **Single Queue**: appena una partizione si libera, il primo processo cui può presumibilmente bastare la partizione viene caricato ➔ possibile sciupo di partizioni di grandi dimensioni
- **Free Small Partition**: lasciare sempre libera almeno una partizione di piccole dimensioni
- **Max-Skip**: numero massimo di volte in cui può essere rifiutata la esecuzione di un programma

Operating Systems: Memory Management

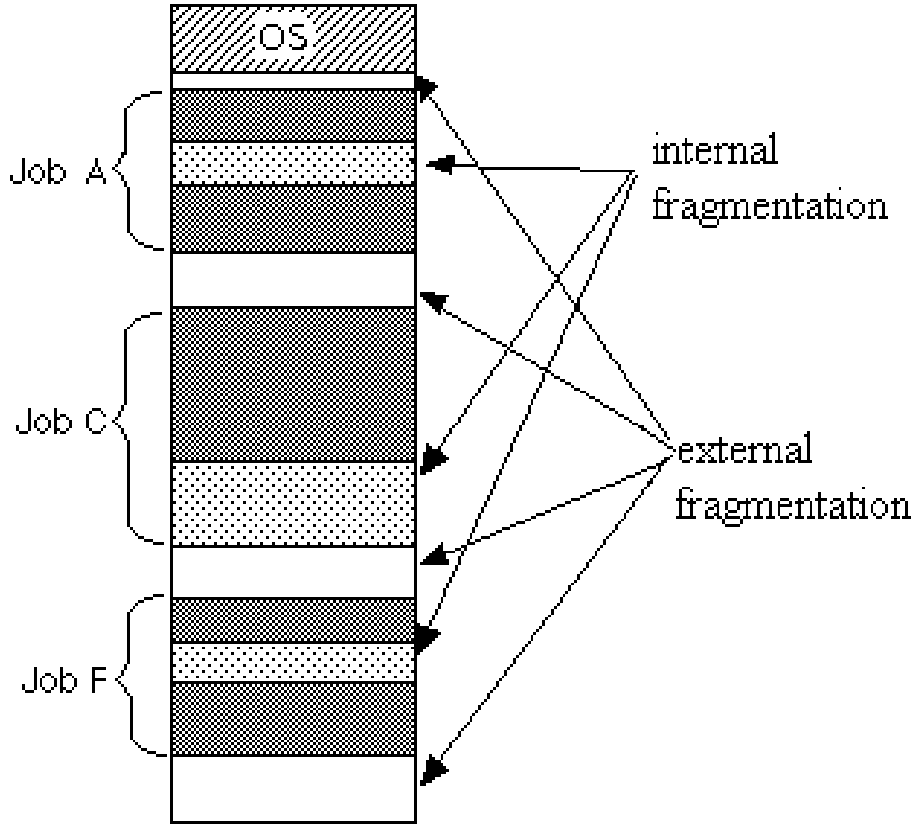
Basic: Memory Partitioning 4/4

Memory Manager: gestione delle porzioni di memoria frammentate (che non riescono ad essere utilizzate).

Frammentazione

- **Interna**: lo spazio allocato in eccesso rispetto alle esigenze dei processi, inutilizzabile perché allocato. Inevitabile. Se ne può ridurre l'impatto riducendo la dimensione delle partizioni

- **Esterna**: lo spazio libero in aree troppo piccole per essere utili. Può essere ridotta tramite:
  - **Rilocazione**: tecniche di compattamento. Efficaci ma computazionalmente pesanti
  - **Paginazione**: partizioni di dimensioni fisse



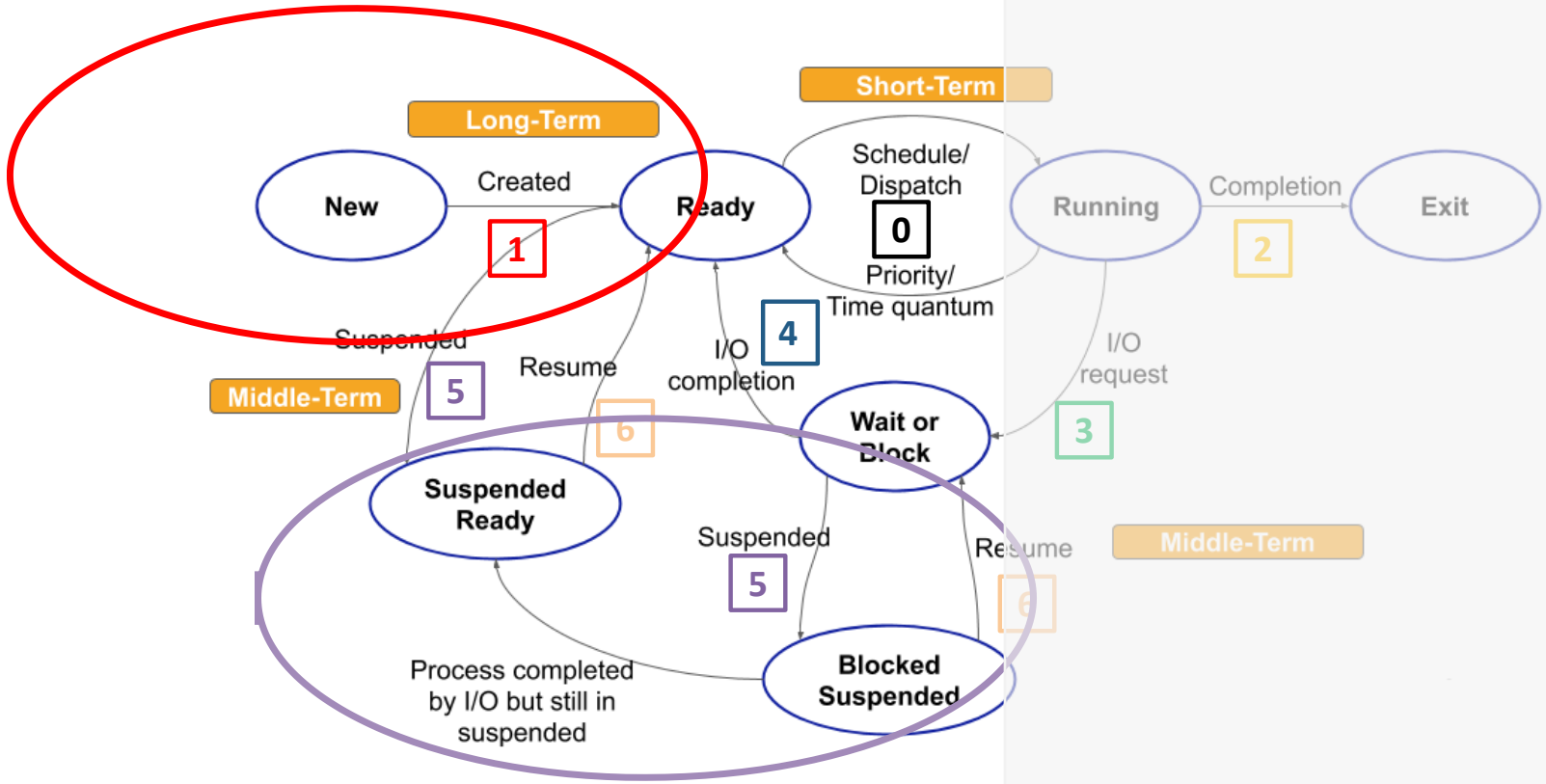
Operating Systems: Memory Management

Basic: Degree of Multiprogramming 1/4

Memory Manager: tenere presente anche il livello di utilizzazione della CPU.

Degree of Multiprogramming

- Finora ci si è preoccupati solo della allocazione (o ri-allocazione) di memoria
- È opportuno tener conto anche della **CPU-utilization**



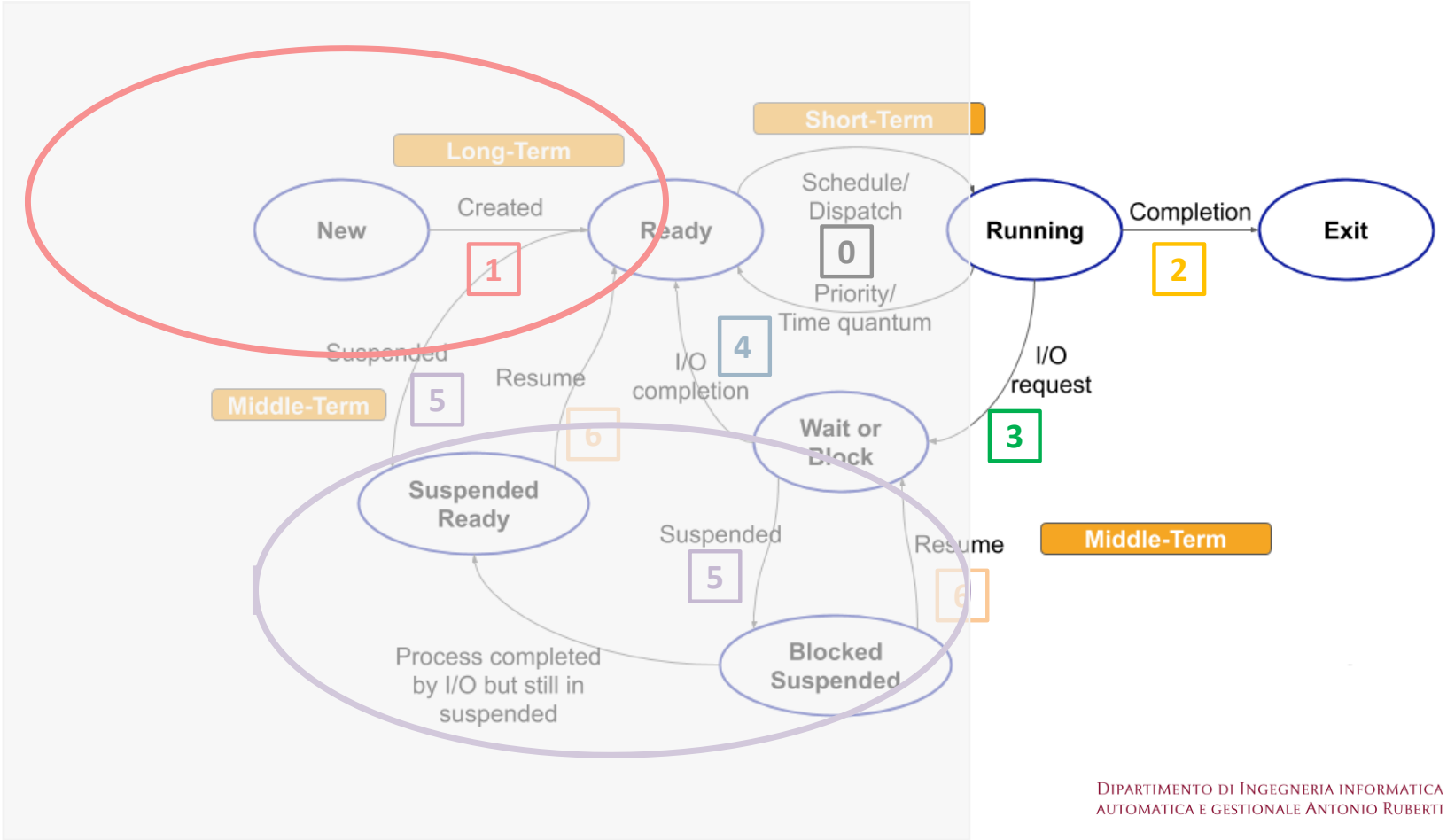
Operating Systems: Memory Management

Basic: Degree of Multiprogramming 2/4

Memory Manager: tenere presente anche il livello di utilizzazione della CPU.

Degree of Multiprogramming

- Finora ci si è preoccupati solo della allocazione (o ri-allocazione) di memoria
- È opportuno tener conto anche della **CPU-utilization**
- Ogni processo trascorre una percentuale *p* del suo tempo di esecuzione in stato «*Wait or Block*» ➔ ***CPU usage = 1 - p***

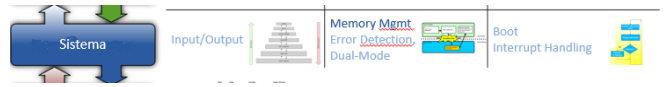


Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

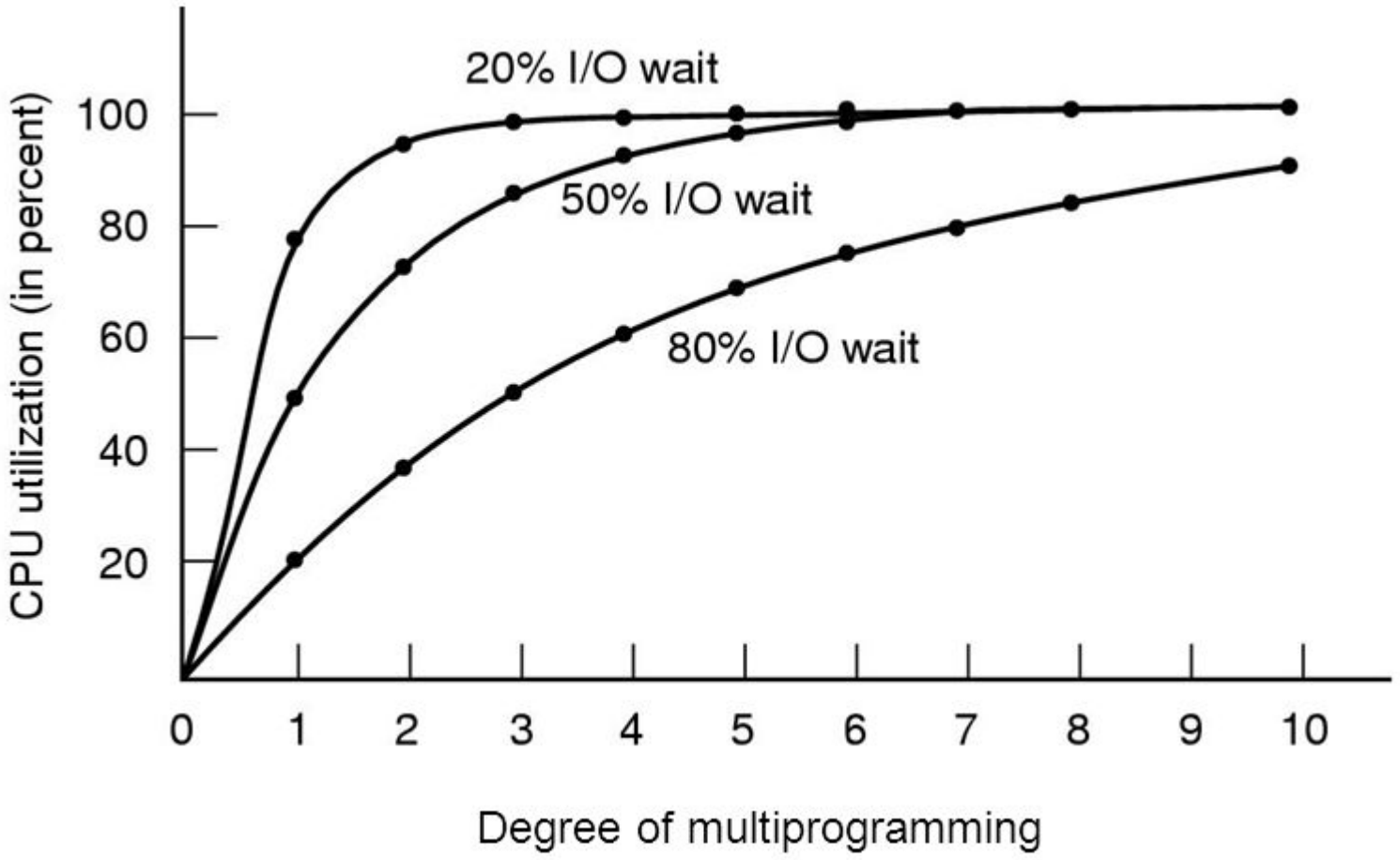
Sapienza Università di Roma

Operating Systems: Memory Management

Basic: Degree of Multiprogramming 3/4



Memory Manager: tenere presente anche il livello di utilizzazione della CPU.



Assumendo che:

- Tutti gli *n* processi in esecuzione presentano la stessa percentule *p* in stato «Wait or Block»
- La condizione di «Wait or Block» è fondamentalmente causata da **I/O Wait**

Si definisce **Degree of Multiprogramming** la funzione di n:

$CPU\text{-}utilization = 1 - p^n$

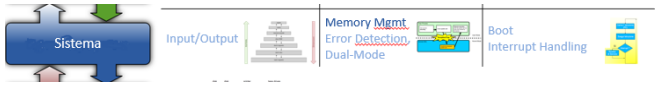
p	*n (40%)*	*n (90%)*
20%	1	2
50%	2	5
80%	3	10

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management

Basic: Degree of Multiprogramming 4/4

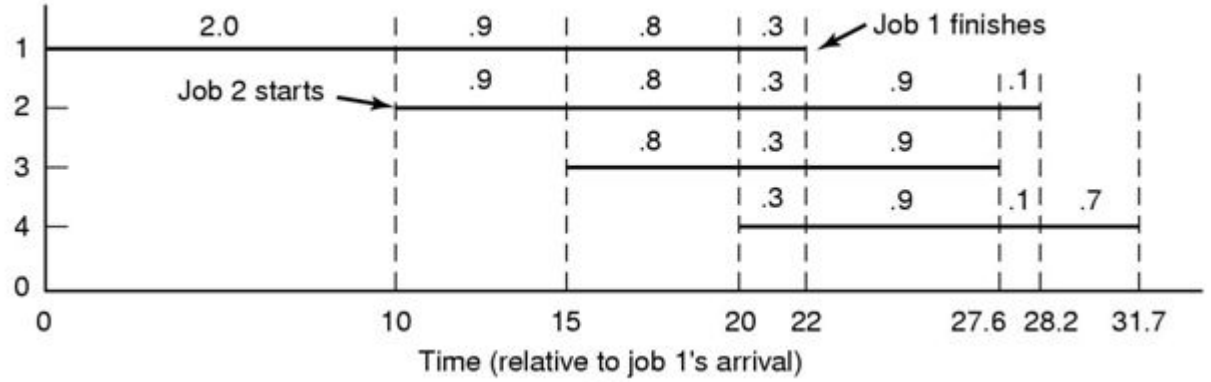


Memory Manager: tenere presente la performance generale, mediante gli altri parametri di scheduling.

- Tutti i *4* processi in esecuzione sono supposti avere *p = 80%*

Job	Arrival time	CPU minutes needed
1	10:00	4
2	10:10	3
3	10:15	2
4	10:20	2

# Processes	1	2	3	4
CPU idle	.80	.64	.51	.41
CPU busy	.20	.36	.49	.59
CPU/process	.20	.18	.16	.15



- **Throughput** = 4/31.7 ≈ 7,9 job/ora
- **Turnaround** = (22+18,2+12,6+11.7)/4 = 16,125 minuti
- **Waiting Time** = (5x16% + 5x29% + 2x39% + 5,6x29% + 0,6x16%)/4 ≈ (0,8+1,5+0,8+1,6+ 0.1)/4 ≈ 1,2 minuti
- **Response Time** = **Turnaround** (processi batch) = 16,125 minuti

Operating Systems: Scheduling

Criteri di Ottimizzazione

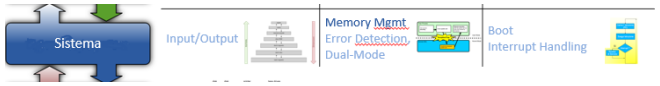
1. Utilizzo della CPU (**CPU usage**): la CPU deve essere attiva carico) al 90% (utilizzo intenso)
2. Frequenza di completamento (**throughput**): numero di p
3. Tempo di completamento (**turnaround time**) – intervallo momento del completamento (comprende tempi di esec
4. Tempo di attesa: somma dei tempi spesi in attesa nella c tempo di attesa, non sul tempo di esecuzione)
5. Tempo di risposta (**response time**): tempo che intercorre risposta (si conta il tempo necessario per iniziare la rispo

In genere si ottimizza:

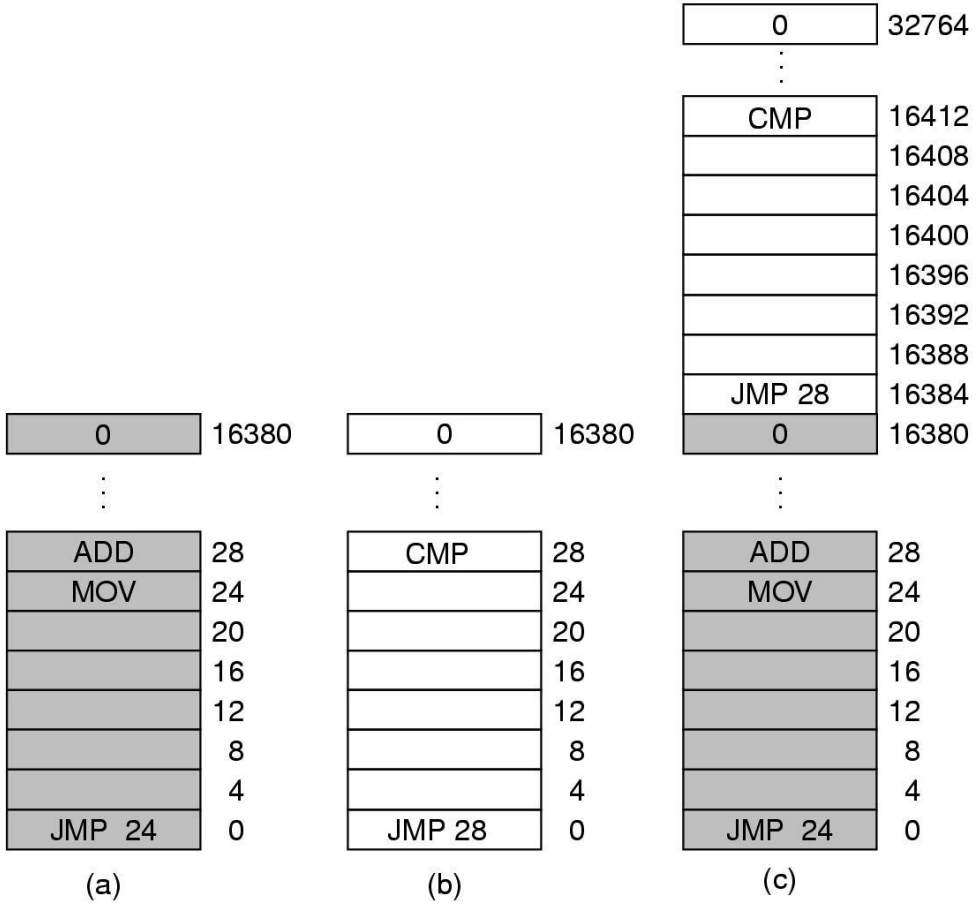
- il **valore medio** e/o
- Valore minimo/massimo e/o
- la **varianza** (per sistemi time-sharing, si preferisce minim predicibile che uno più veloce ma maggiormente variabile

Operating Systems: Memory Management

Basic: Relocation



Memory Manager: corretto utilizzo della memoria da parte del programma, qualsiasi partizione gli sia assegnata.



Rilocazione

(a), (b) programmi allocati ad inizio memoria, nessun problema.
Nella situazione (c), I 2 programmi sono in memoria in contemporanea: uno dei due deve risiedere in altra locazione. Allorchè il secondo programma esegue la JMP 28 (in 16384, a run-time) salta all'istruzione ADD nella posizione 28 e non alla CMP alla 16412 come desiderato: **Il programma si blocca**.

Rilocazione Statica

caricare la prima istruzione del programma all'indirizzo x, e aggiungere x ad ogni indirizzo successivo durante il caricamento:

➔ **troppo lento**: modificare ogni istruzione ad ogni riallocazione!

Rilocazione Dinamica: Spazio di Indirizzi (Address Space)

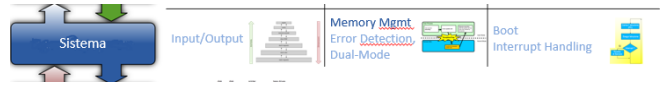
- Creare uno spazio di memoria astratto in cui il programma possa esistere
- Ogni programma ha il suo insieme di indirizzi
- Gli indirizzi sono diversi per ogni programma
- Chiamalo **spazio di indirizzi del programma**

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management

Basic: Protection



Memory Manager: garantire che ogni processo acceda solo alla sua area di memoria (a meno di condivisioni volute).

IBM 360

- dividere la memoria in blocchi di 2 KB
- associare una chiave di protezione di 4 bit al blocco.
- Tenere le chiavi nei registri.
- Mettere la chiave nel PSW (Program Status Word) del programma

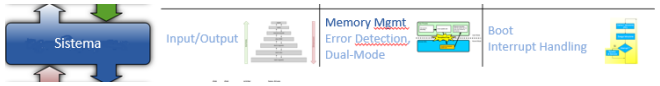
L'hardware impedisce al programma di accedere al blocco con un'altra chiave di protezione

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

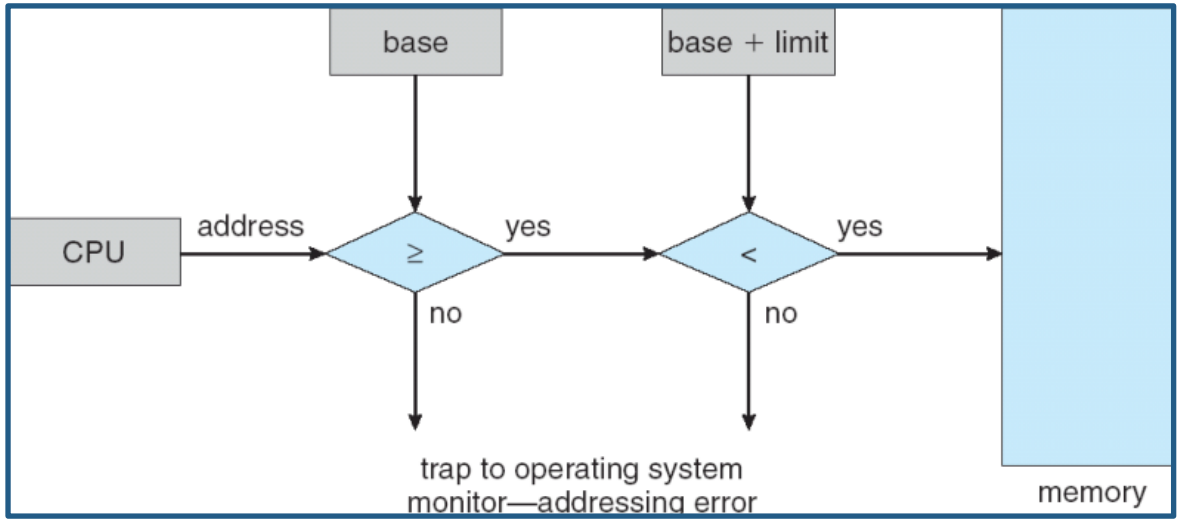
Basic: Relocation & Protection ➔ Base & Limit Registers 1/2



Memory Manager: corretto utilizzo della partizione di memoria assegnata, senza sconfinare in altre aree di memoria.

Registri Base & Limit

- Una forma di rilocazione dinamica
- **Base** contiene l'indirizzo iniziale del programmaII
- **Limit** contiene la lunghezza del programma
- Il programma fa riferimento alla memoria, aggiunge l'indirizzo di base all'indirizzo generato dal processo.
- Controlla se l'indirizzo è più grande del limite. Se è così, genera un errore



Svantaggio: l'aggiunta e il confronto devono essere fatti su ogni istruzione

Usato già nel CDC 6600 e nell'Intel 8088

Implementato in Hardware

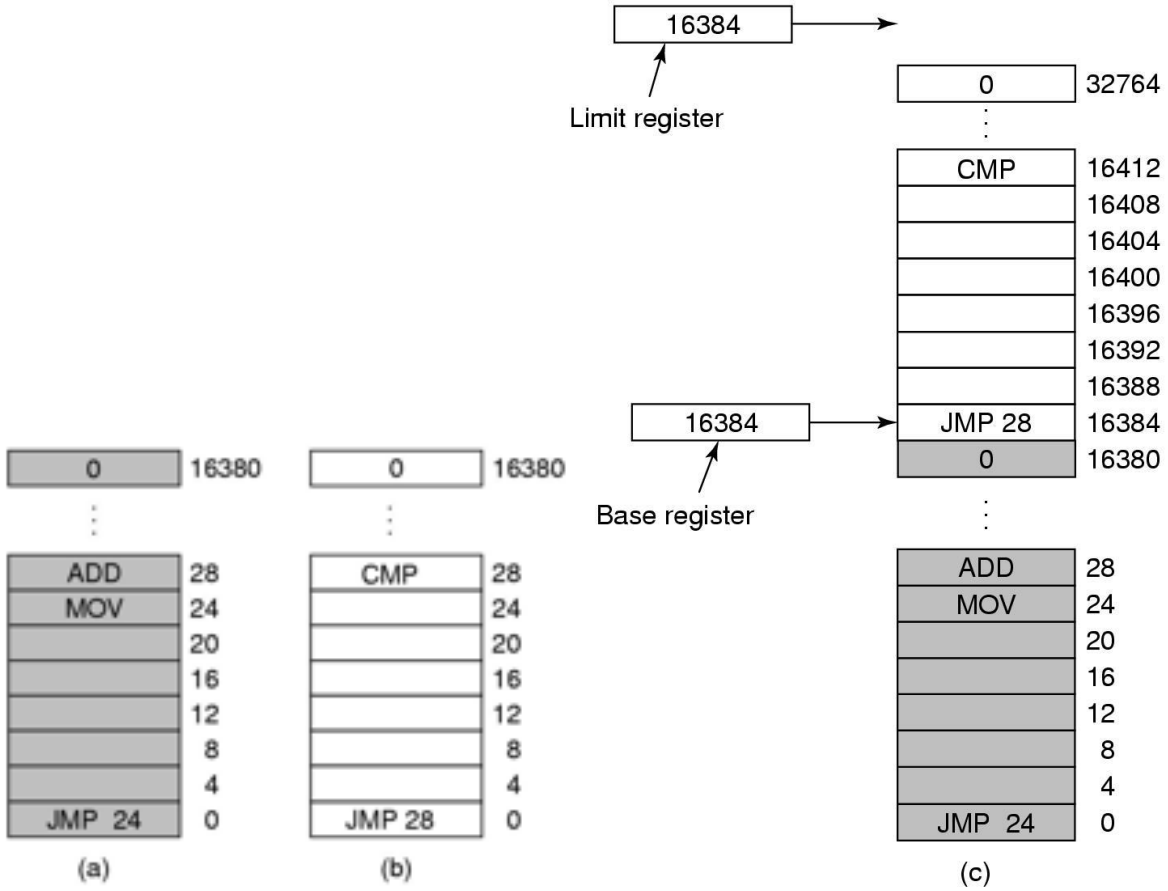
DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

Basic: Relocation & Protection ➔ Base & Limit Registers 2/2

Memory Manager: corretto utilizzo della partizione di memoria assegnata, senza sconfinare in altre aree di memoria.



Rilocazione Dinamica Base&Limit

(a), (b) programmi allocati ad inizio memoria, nessun problema.

Nella situazione (c), I 2 programmi oa sono allocati in memoria con il meccanismo Base&Limit. Allorchè il secondo programma esegue la JMP 28, (in 16384, a run-time) l'HW traduce in 28 + Base (16384) = 16412 come desiderato

➔ **il programma procede correttamente**

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

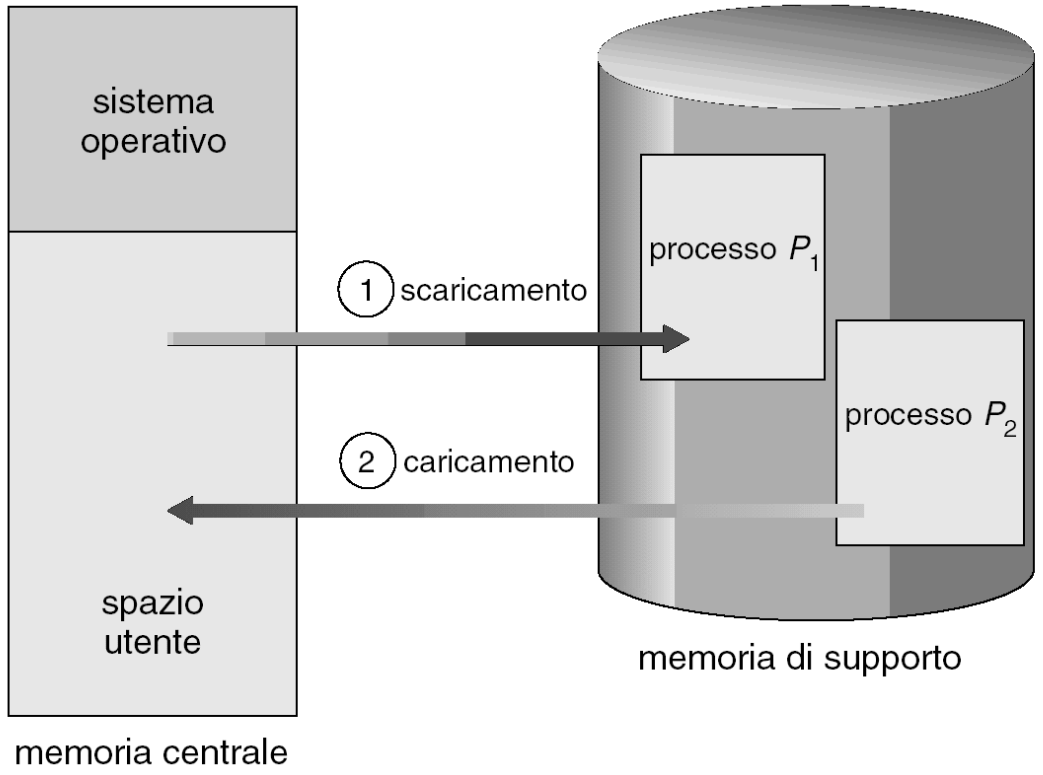
Operating Systems: Memory Management

Swapping: Avvicendamento

Memory Manager: può procedere all'avvicendamento dei processi da e verso una memoria secondaria (area di swap).

Swapping

- Permette di gestire più processi di quelli che fisicamente sarebbero caricabili in memoria
- Il periodico scambio tra processi in memoria centrale e secondaria (swapping) è controllato dallo scheduler a medio termine
- Quando il processo uscente subisce uno swap out viene copiato il descrittore di processo su memoria di massa
- È inutile salvare le istruzioni: basta ricaricarle dal testo del programma memorizzato nel file system

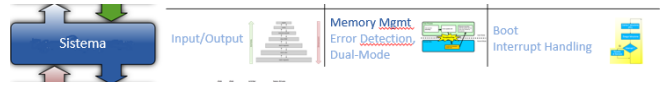


Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

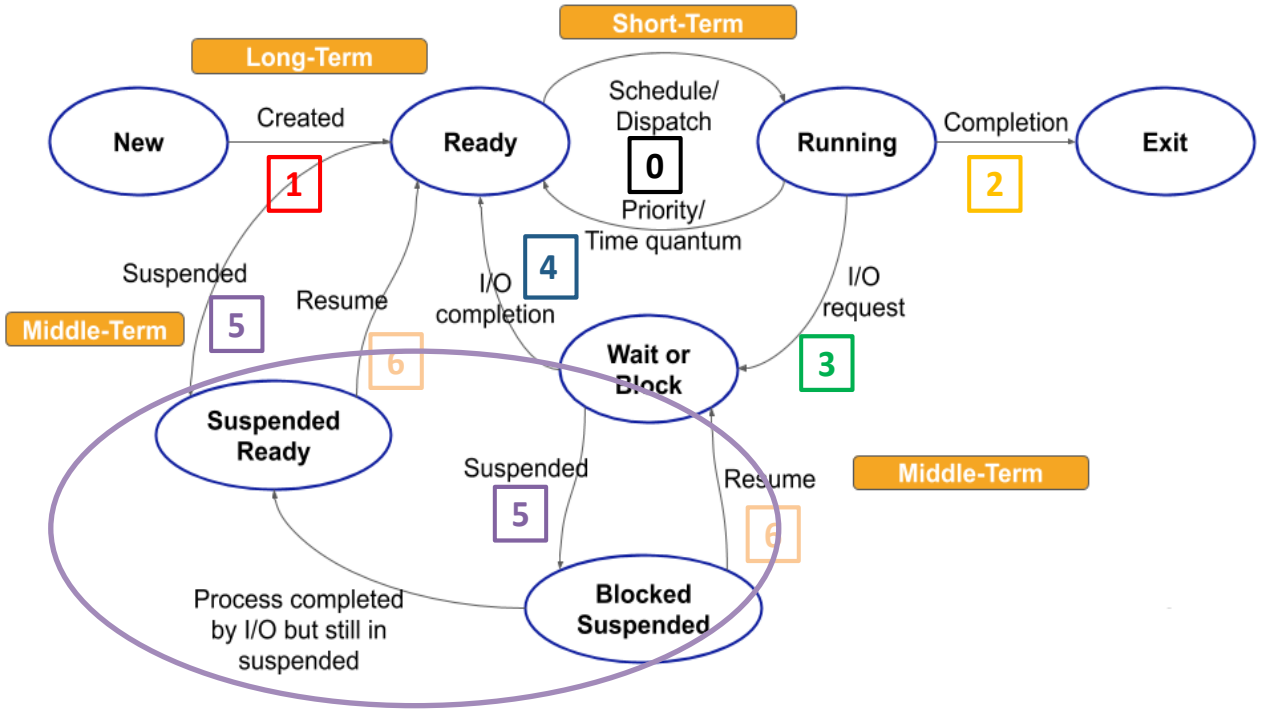
Sapienza Università di Roma

Operating Systems: Memory Management

Swapping: Stati dei Processi Addizionali



Memory Manager: procedere all'avvicendamento implica la introduzione della condizione Suspended, declinata in 2 stati: «Suspended Reay», «Blocked Suspended»).



Swapping

- Lo **swapping** è molto comune nei sistemi con schedulatore Round Robin (Il processo che finisce il quanto di tempo subisce lo swap-out)
- **Roll out**, **Roll in** è una variante dello swapping usata per algoritmi di schedulazione basati sulla priorità: un processo a bassa priorità è scambiato con un processo ad alta priorità, in modo che quest'ultimo possa essere caricato ed eseguito
- Versioni modificate di swapping si trovano in molti SO (ad esempio UNIX, Linux, e Windows) combinate con altre tecniche

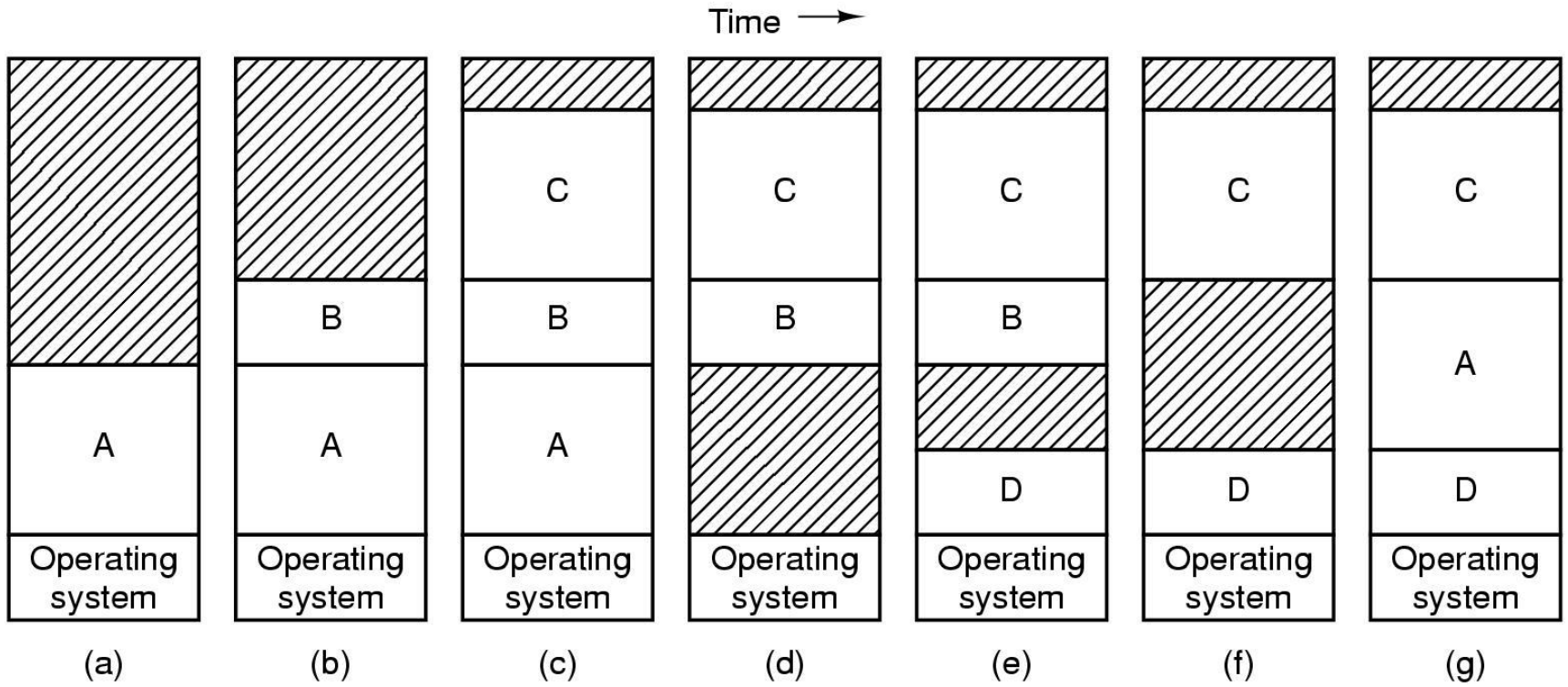
Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Operating Systems: Memory Management

Swapping: Compattazione della Memoria

Memory Manager: procedere all'avvicendamento è utile se le vengono rese utili le zone di memoria liberata.



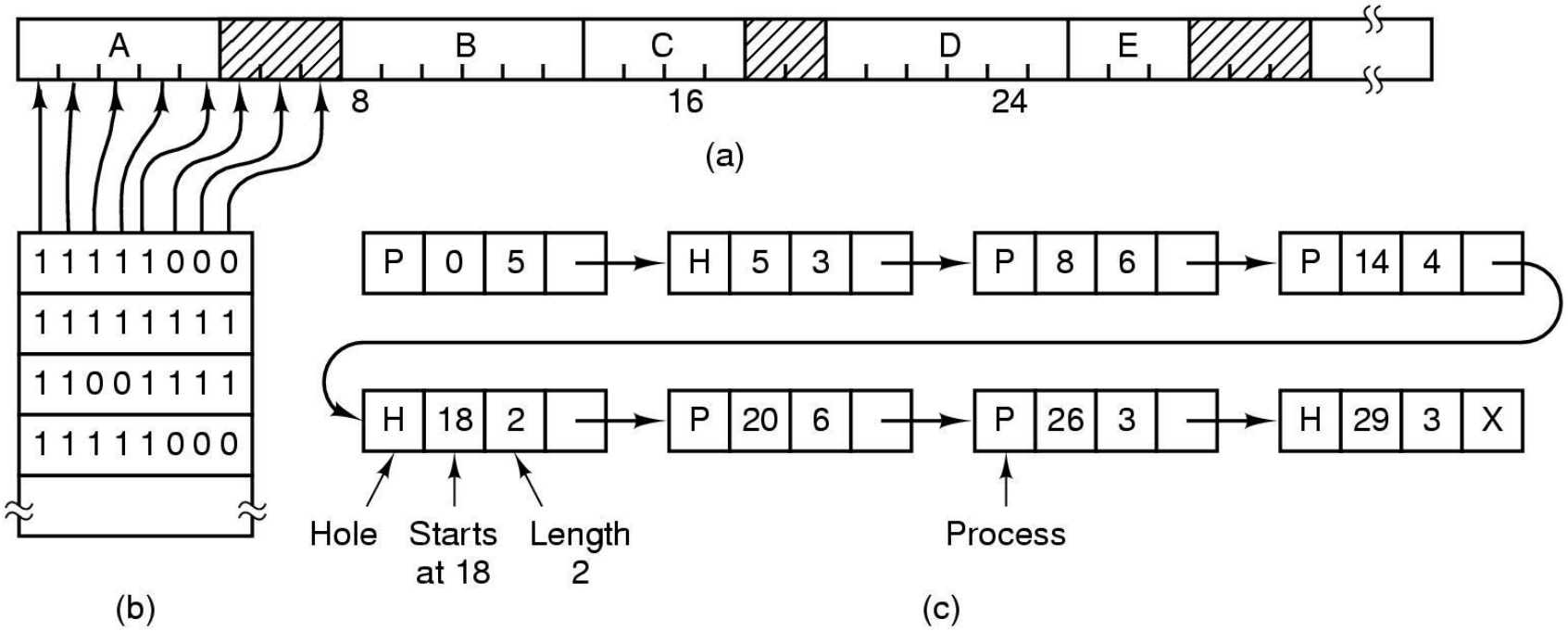
Swapping

- Richiede il compattamento dei buschi di memoria, generalmente risolti copiandovi i programmi ➔ **Richiede tempo**

Operating Systems: Memory Management

Swapping: Bitmap

Memory Manager: mappa della memoria in uso o libera bit a bit.



Bitmap

- (a) immagine della memoria
- (b) **Bitmap**: ogni bit nel bitmap corrisponde ad una unità della memoria (es. Word)
- (c) **Linked list** P: process H: hole

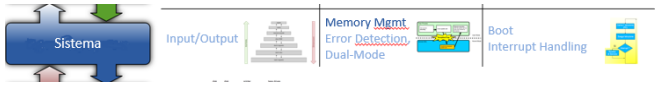
- **Pro**: modocompatto di tenere traccia della memoria
- **Contro**: è necessario cercare in memoria k zeri consecutivi per portare un file lungo k unità

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

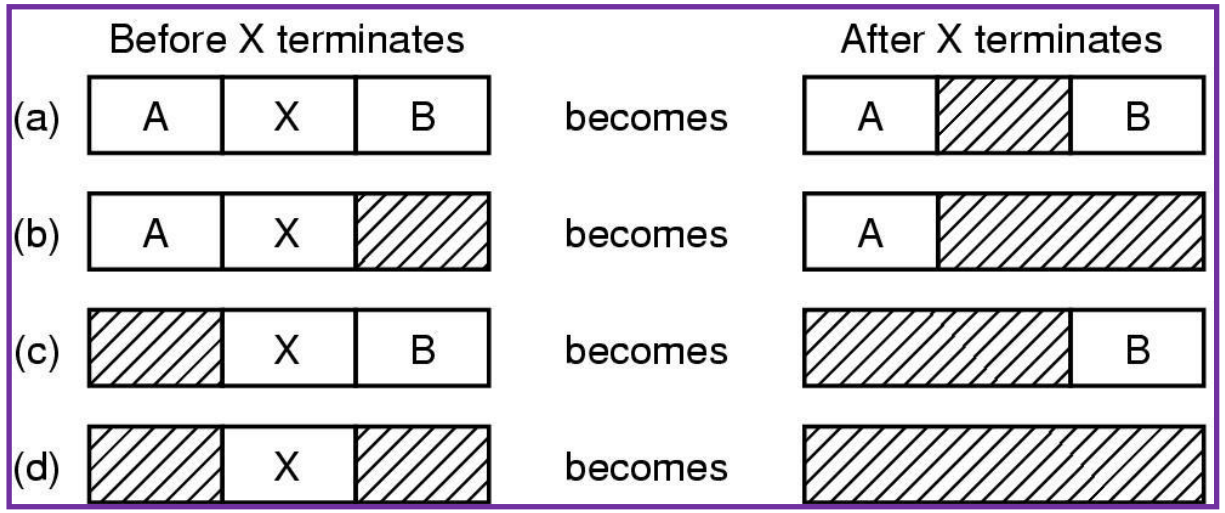
Sapienza Università di Roma

Operating Systems: Memory Management

Swapping: Linked List



Memory Manager: mappa della memoria in uso o libera tramite liste collegate.



- Quattro possibili combinazioni per i processi vicini, nel caso di terminazione di X
- Preferibile usare double-linked list per unire i buchi più facilmente

Fits

- Algoritmi per assegnare la memoria ai processi
- **First Fit**: scansione della lista dei segmenti dall'inizio finché non viene trovato il primo buco utile ➔ veloce, molti buchi
- **Next Fit**: scansione della lista dei segmenti dal punto in cui si trova il cursore finché non viene trovato il primo buco utile ➔ più veloce, più buchi
- **Best Fit**: scansione della lista per trovare il più piccolo buco utile ➔ più lento, molti buchi inutilizzabili
- **Worst Fit**: scansione della lista per trovare il più grande buco utile ➔ non buono
- **Quick Fit**: più liste per dimensioni diverse ➔ veloce, non riesce ad effettuare I merge tra aree vicine

Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti

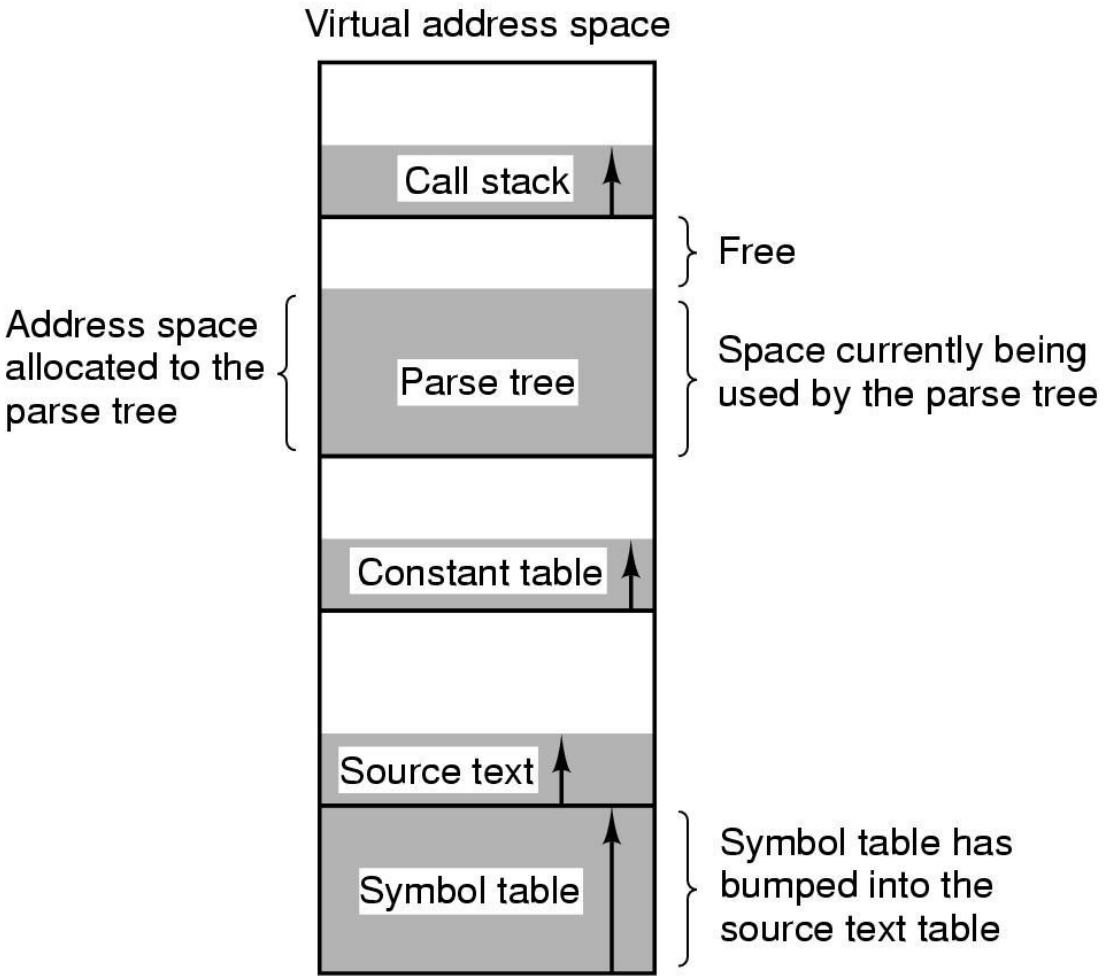
Sapienza
Università di Roma

Segmentazione

Operating Systems: Memory Management

Segmentation: Multidimensional Address Space 1/3

Memory Manager: divisione della memoria in dimensioni parallele.



Monodimensional Address Space

Se le tabelle crescono, possono collidere.

Es. Un compilatore ha molte tabelle che vengono costruite man mano che la compilazione procede, quali:

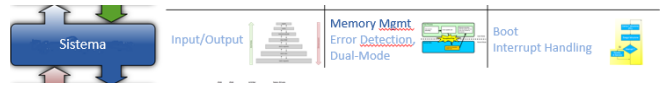
- testo sorgente che viene salvato per il listato stampato (su sistemi batch).
- simboli - i nomi e gli attributi delle variabili.
- costanti intere e a virgola mobile utilizzate.
- albero del parsing, - analisi sintattica del programma.
- stack utilizzato per le chiamate di procedura all'interno del compilatore.

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

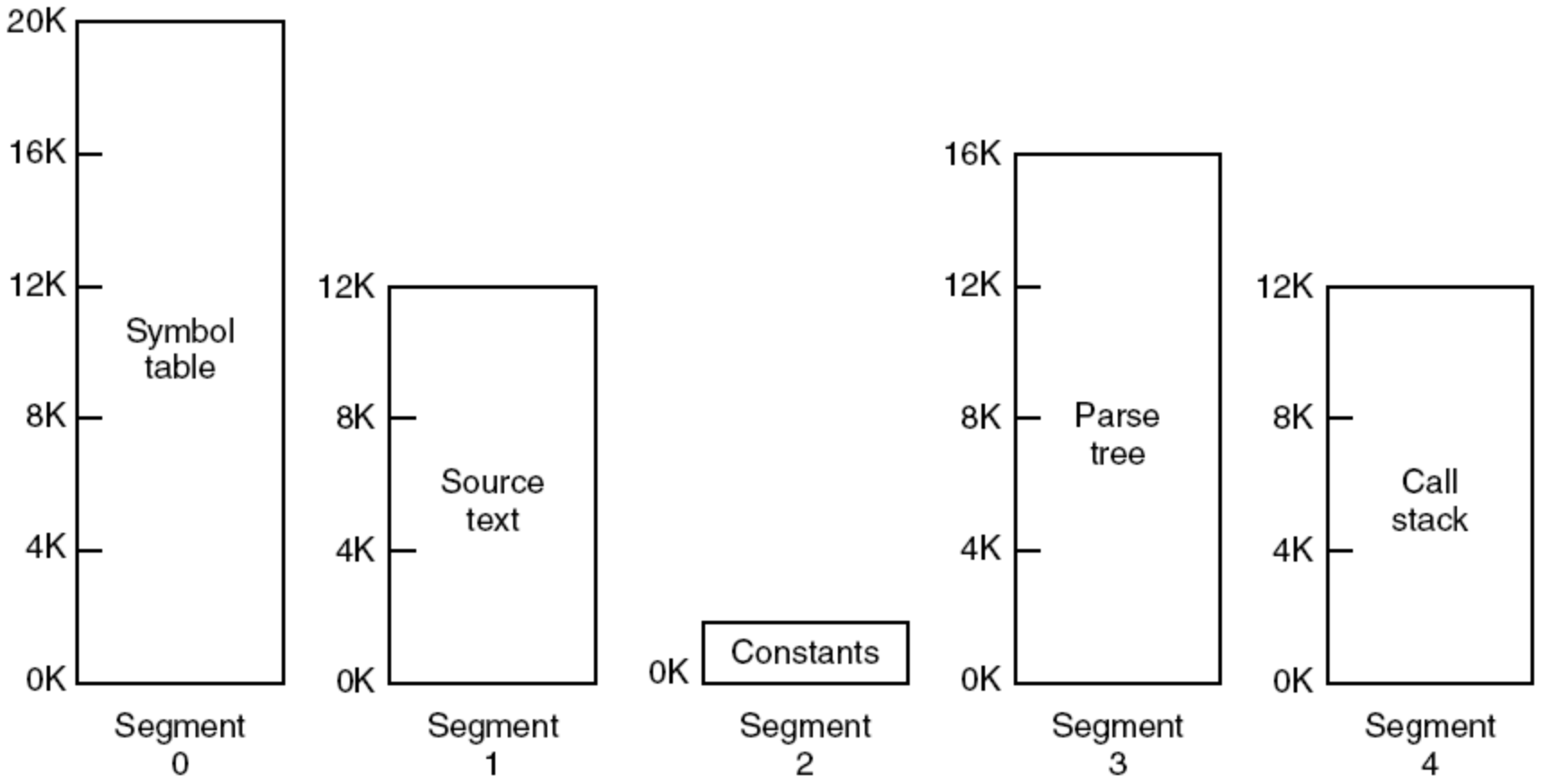
Sapienza Università di Roma

Operating Systems: Memory Management

Segmentation: Multidimensional Address Space 2/3



Memory Manager: divisione della memoria in dimensioni parallele.



Multidimensional Address Space

Segmentando la memoria, ogni tabella può crescere e restingersi indipendentemente.
Inoltre, è possibile configurare criteri di versi da segmento a segmento:

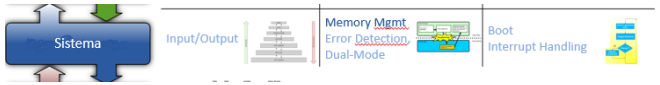
- Condivisione (es. librerie)
- Protezione (r,w,x)

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management

Segmentation: Multidimensional Address Space 3/3



Memory Manager: divisione della memoria in dimensioni parallele.

Comparazione Paging/Segmentation

Consideration	Paging	Segmentation
Need the programmer be aware that this technique is being used?	No	Yes
How many linear address spaces are there?	1	Many
Can the total address space exceed the size of physical memory?	Yes	Yes
Can procedures and data be distinguished and separately protected?	No	Yes
Can tables whose size fluctuates be accommodated easily?	No	Yes
Is sharing of procedures between users facilitated?	No	Yes
Why was this technique invented?	To get a large linear address space without having to buy more physical memory	To allow programs and data to be broken up into logically independent address spaces and to aid sharing and protection

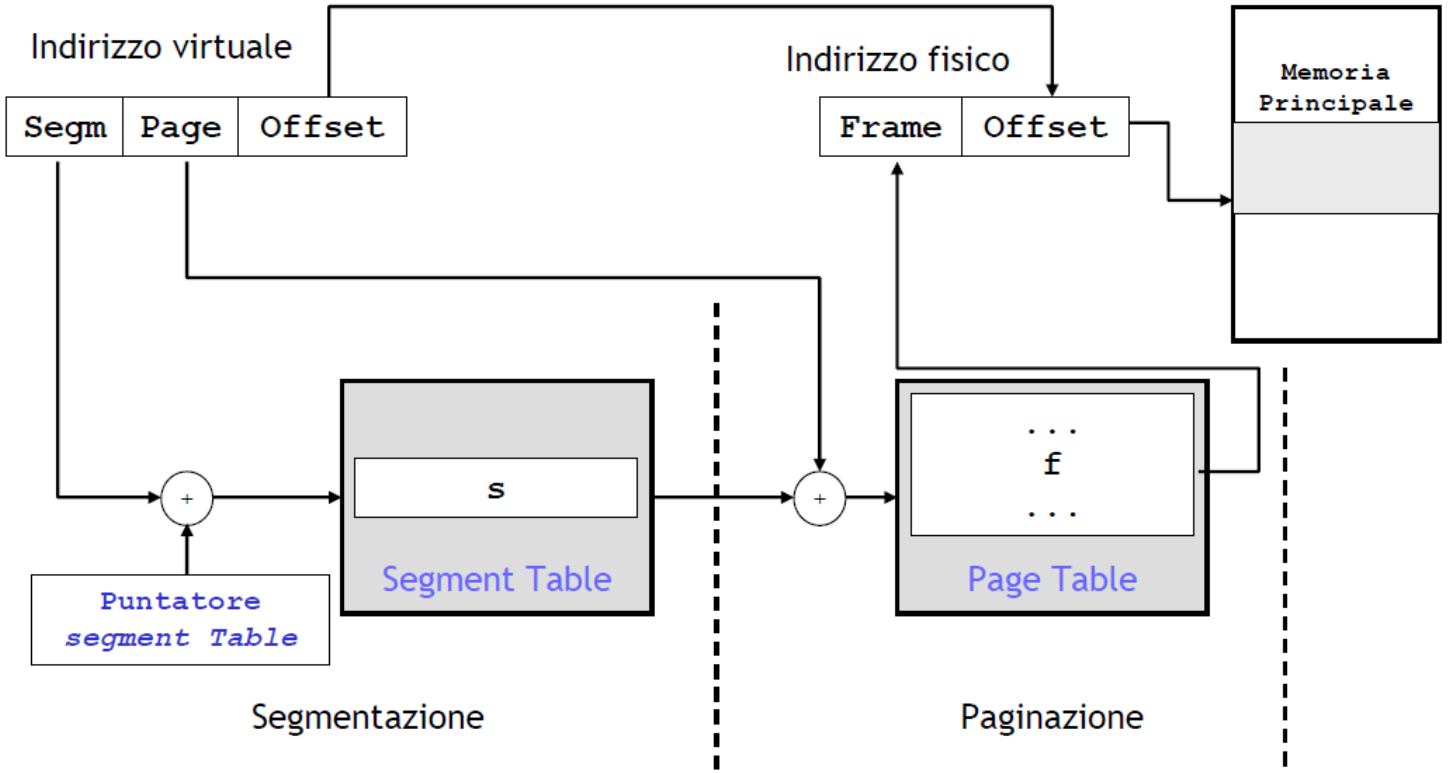
Operating Systems: Memory Management

Segmentation: Segmentation with Pagination

Memory Manager: permette di suddividere il segmento in pagine.

Segmentazione con Paginazione

Il descrittore di segmento dice se il segmento è nella memoria principale o no
Se una qualsiasi parte del segmento è in memoria, l'intero segmento è considerato in memoria



Virtual Address:

- **Segm**: numero di segmento
- **Page**: numero di pagina
- **Offset**: offset nella pagina

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Paginazione

Operating Systems: Memory Management
Paginazione 1/3


```
Architettura: B bit

Paginazione ad 1 livello:
Sizeof (Page): P KB ➔ bit di indirizzo = B – 10 – log2(P)
Reserved for Kernel: K KB

Paginazione a 2 livelli:
Indirizzo della page directory a D bit
```

Paginazione Semplice

- Numero massimo di pagine =
- Pagine tolte dal kernel =
- Pagine «paginabili»:

Paginazione a doppio livello

- Bit di indirizzo per il 2° livello =
- Numero totale di tabelle delle pagine =

Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Operating Systems: Memory Management
Paginazione 2/3


```
Architettura: B bit

Paginazione ad 1 livello:
Sizeof (Page): P KB ➔ bit di indirizzo = B – 10 – log2(P)
Reserved for Kernel: K KB

Paginazione a 2 livelli:
Indirizzo della page directory a D bit
```

Paginazione Semplice

- Numero massimo di pagine = dimensione massima indirizzabile / dimensione della pagina = $2^{(B-10)}/P$
- Pagine tolte dal kernel = K/P
- Pagine «paginabili»: $(2^{(B-10)} – K)/P$

Paginazione a doppio livello

- Bit di indirizzo per il 2° livello =
- Numero totale di tabelle delle pagine =

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management
Paginazione 3/3


```
Architettura: B bit

Paginazione ad 1 livello:
Sizeof (Page): P KB ➔ bit di indirizzo = B – 10 – log2(P)
Reserved for Kernel: K KB

Paginazione a 2 livelli:
Indirizzo della page directory a D bit
```

Paginazione Semplice

- Numero massimo di pagine = dimensione massima indirizzabile / dimensione della pagina = $2^{(B-10)}/P$
- Pagine tolte dal kernel = K/P
- Pagine «paginabili»: $(2^{(B-10)} – K)/P$

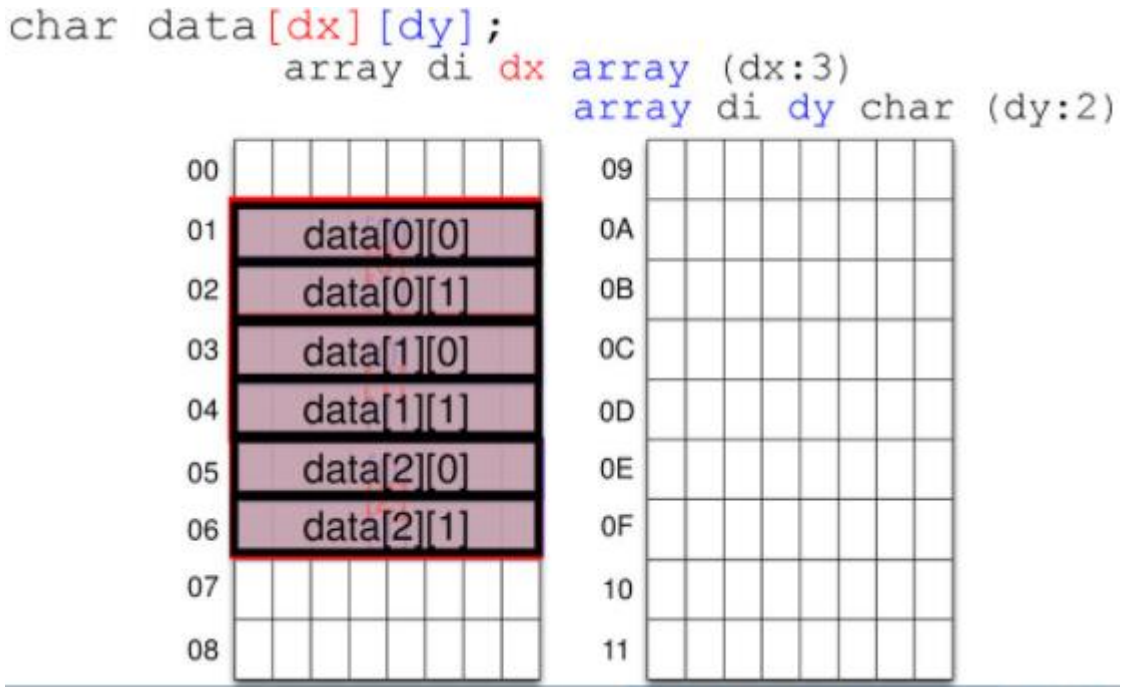
Paginazione a doppio livello

- Bit di indirizzo per il 2° livello = bit di indirizzo della soluzione ad 1 livello – bit indirizzamento della page directory = B – D-10 –$\log_2$ (P)
- Numero totale di tabelle delle pagine = Pagine di 1° livello + pagine di 2° livello = $2^D * (1+ 2^{<\text{bit di indirizzo del 2° livello}>}) = 2^D * (1+ 2^{(B – D\text{-}10 – \log2 (P))})$

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memoria
Array e Paginazione 1/5




```
#define dx 3
#define dy 2
char data[dx][dy];
```

Memoria occupata da data = 3 x 2 x dimensione (char) in byte:
= 6 x 4 (generalmente) = 24 byte (generalmente)

Organizzati in memoria in modo contiguo in memoria.
La memoria è lineare (stream).
La matrice è a 2 dimensioni ➔ immagazzinata per righe.

➔ Conviene scandire la matrice per righe (cioè avendo come ciclo for esterno l'indice di riga).

```
for (i = 0; i< dy; i++)
 for (j = 0; j< dx; j++)
   action();
```

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti
Sapienza Università di Roma

Operating Systems: Memoria
Array e Paginazione 2/5


```
#define Size 64
int A[Size][Size], B[Size][Size], C[Size][Size];
int register i, j;
for (j = 0; j< Size; j ++)
  for (i = 0; i< Size; i++)
    C[i][j] = A[i][j] + B[i][j];
```

Scansione delle colonne

Scansione delle righe

Sapendo che:
- sizeof (int) = 4 B
- sizeof (page) = 1 KB
- Allocazione in memoria tipica del C

Semplificazioni:
- Gli indici di riga e colonna sono in registro
- La paginazione è ad 1 livello (no fault tabella delle pagine)
- 3 pagine per le matrici ➔ 1 pagina a matrice

1. Fault di pagina attuali
2. Fault di pagina minimi

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

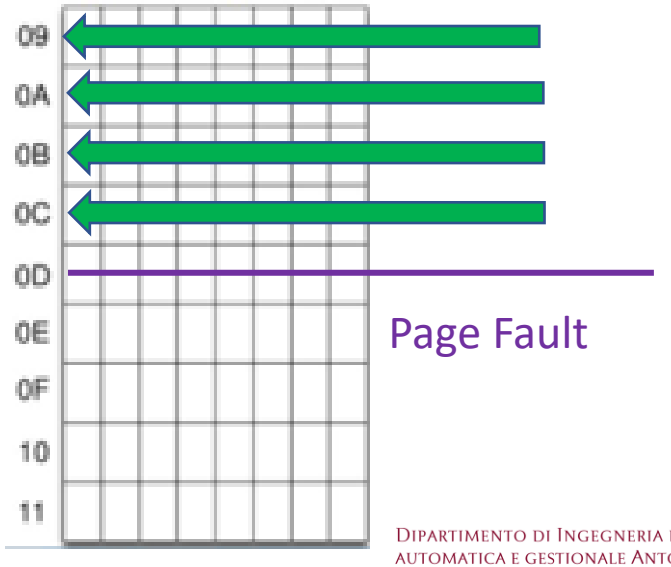
Operating Systems: Memoria
Array e Paginazione 3/5


```
#define Size 64
int A[Size][Size], B[Size][Size], C[Size][Size];
int register i, j;
for (i = 0; i< Size; i++)
  for (j = 0; j< Size; j ++)
    C[i][j] = A[i][j] + B[i][j];
```

Scansione per righe

Memoria occupata da A = 64 x 64 x dimensione (int) in byte
= 2^6 x 2^6 x 2^2 = 2^{14} = 16 KB
Se pagine da 1KB ➔ 16 pagine ➔ ogni pagina contiene 1/16 di matrice ➔ 4 righe

Se scansione per righe ➔ per ogni matrice: no page fault durante la scansione di 1/16 (=4) righe ➔ page fault = 48 - <pagine gia' allocate>



Operating Systems: Memoria
Array e Paginazione 4/5

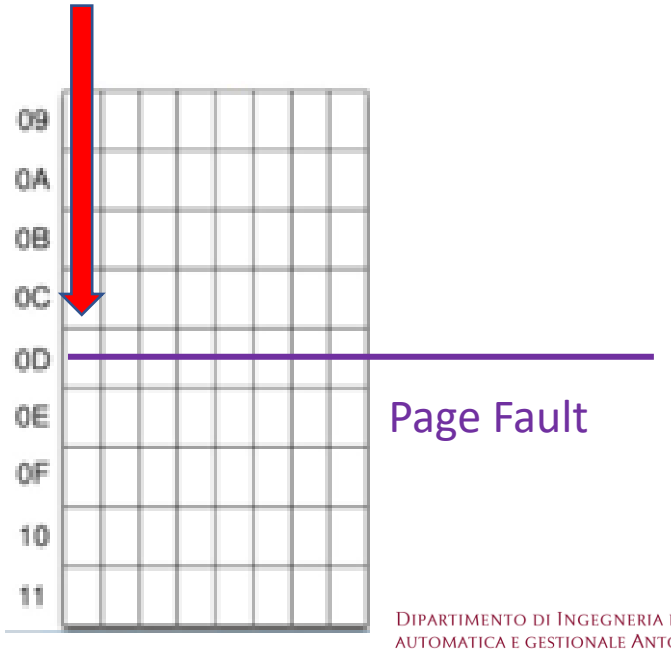

```
#define Size 64
int A[Size][Size], B[Size][Size], C[Size][Size];
int register i, j;
for (j = 0; j< Size; j ++)
  for (i = 0; i< Size; i++)
    C[i][j] = A[i][j] + B[i][j];
```

Scansione per colonne

Memoria occupata da A = 64 x 64 x dimensione (int) in byte = 2^6 x 2^6 x 2^2 = 2^{14} = 16 KB
Se pagine da 1KB ➔ 16 pagine ➔ ogni pagina contiene 64 tronconi delle colonne spezzate in 16 parti.

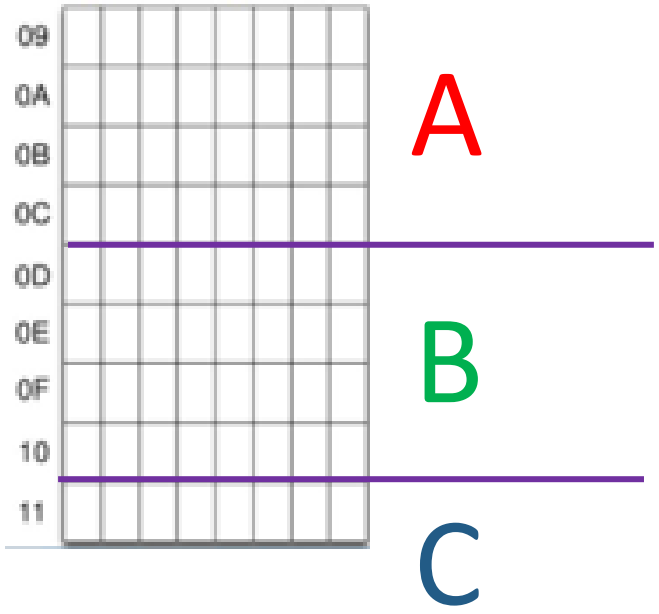
Se scansione per colonne ➔ per ogni matrice: 16 page fault per ogni scansione di colonna ➔ page fault = 16 x 64 - <pagine gia' allocate> = 2^{10} -1

➔ Conviene scandire la matrice per righe (e non per colonne).



Operating Systems: Memoria
Array e Paginazione 5/5


```
#define Size 64
int A[Size][Size], B[Size][Size], C[Size][Size];
int register i, j;
for (j = 0; j< Size; j ++)
  for (i = 0; i< Size; i++)
    C[i][j] = A[i][j] + B[i][j];
```



Buffer Overflow : possibile travalicare i confini delle strutture dati (di fatto, memorizzate in modo lineare).
Esempi:

A[64][2] coincide con B[0][2]

A[129][3] coincide con B[65][3] che coincide con C[1][3]

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

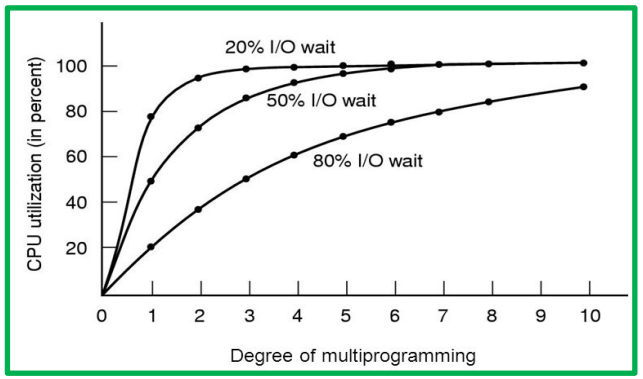
Sapienza Università di Roma

La Memoria Virtuale

Operating Systems: Memory Management

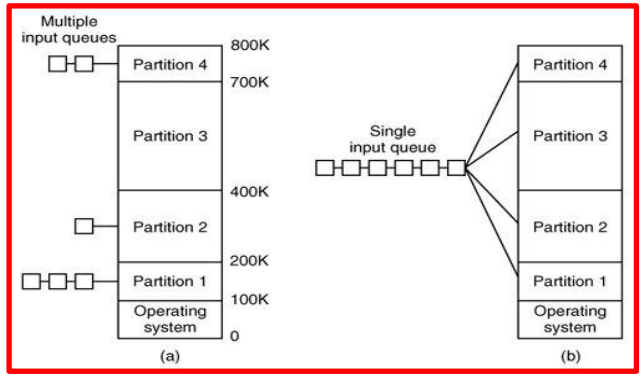
Virtual Memory: Lesson Learned

Memory Manager: deve possedere le seguenti proprietà



Memory Partitioning

➔ **Allocazione di Porzioni**

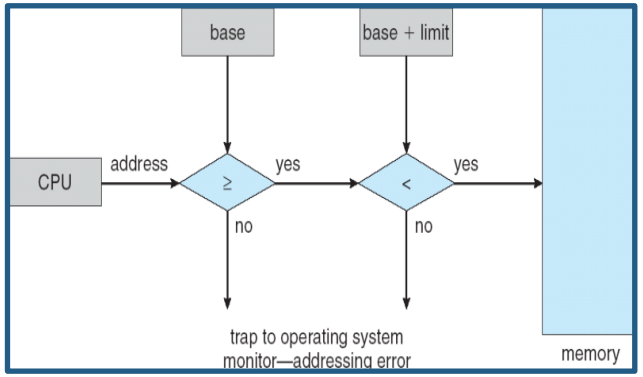


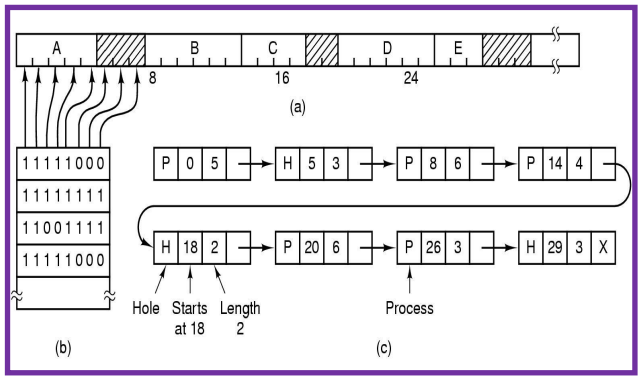
Degree of Multiprogramming

➔ **Performance**

Base&Limit

➔ **HW specifico**





Linked List

➔ **meta-dati più complessi se partizioni di differenti dimensioni**

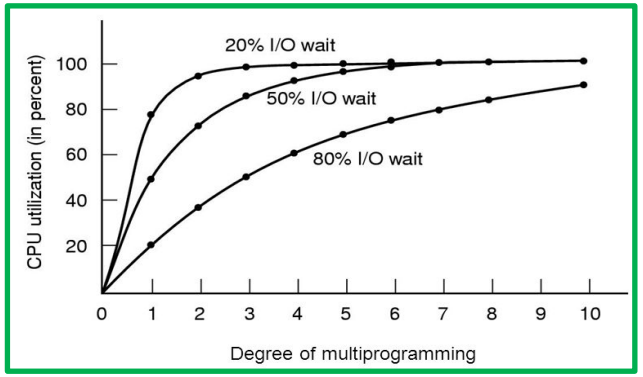
Dipartimento di Ingegneria Informatica Automatica e Gestionale Antonio Ruberti

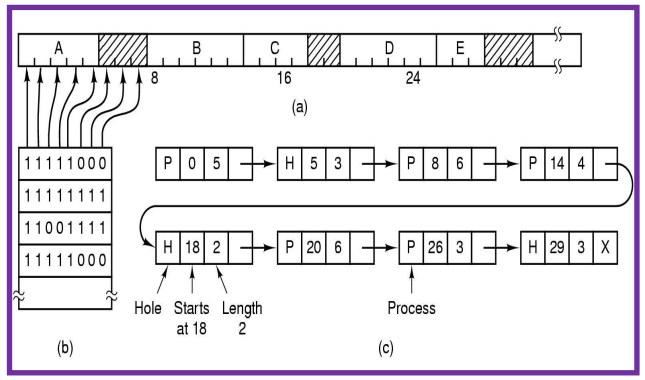
Sapienza Università di Roma

Operating Systems: Memory Management

Virtual Memory: Cenni Storici

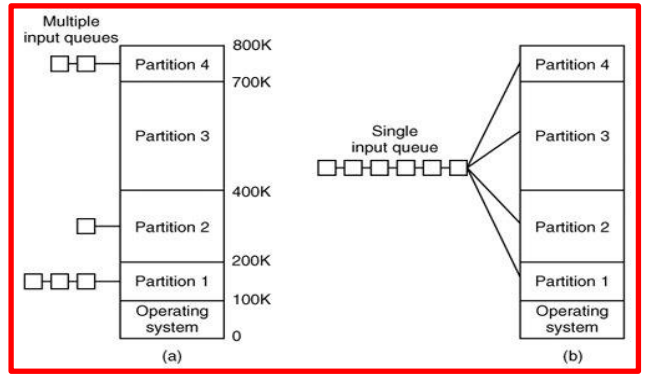
Memory Manager: soluzioni adottate e nel tempo abbandonate





Memory Partitioning

Partitioning: divisione dei programmi in parti (Overlay) da tenere in memoria separatamente

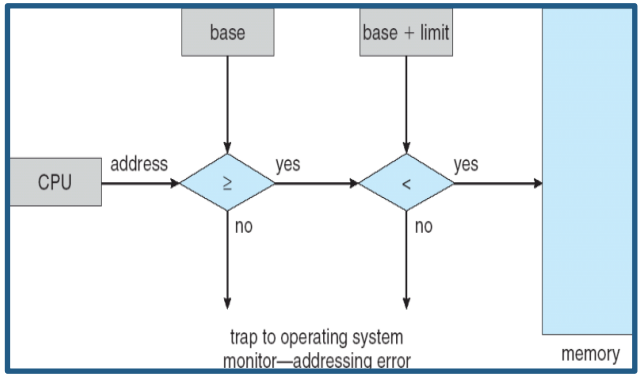


Degree of Multiprogramming

Swapping: troppo lento (richiede secondi)

Base&Limit

Overlay: definizione effettuata dal programmatore, in fase di sviluppo



Linked List

Macchinosa: la gestione degli Overlay

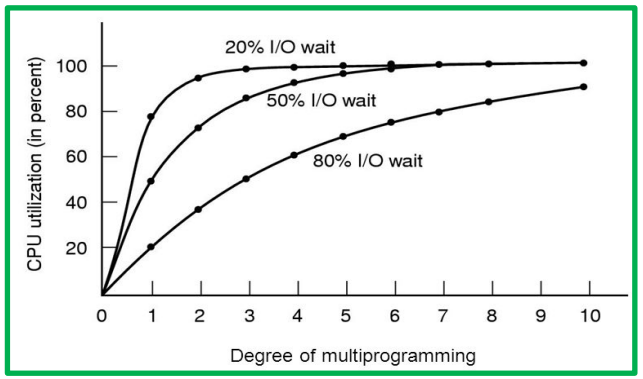
DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

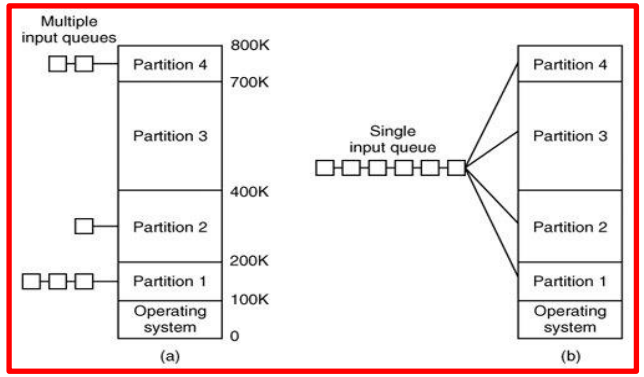
Virtual Memory: Concetti

Memory Manager: soluzioni adottate e confermate nel modello Virtual Memory



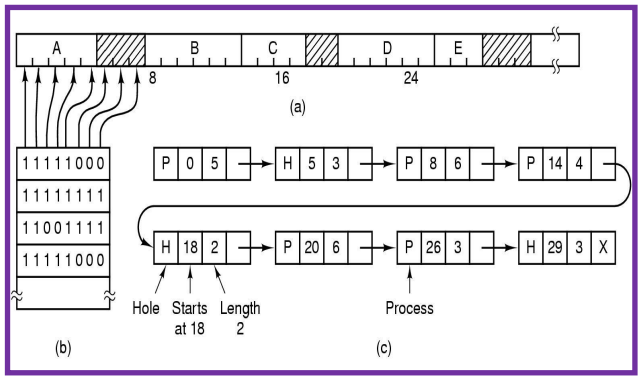
Memory Partitioning

Paging: divisione della memoria in porzioni di dimensioni fisse ed uguali per tutti



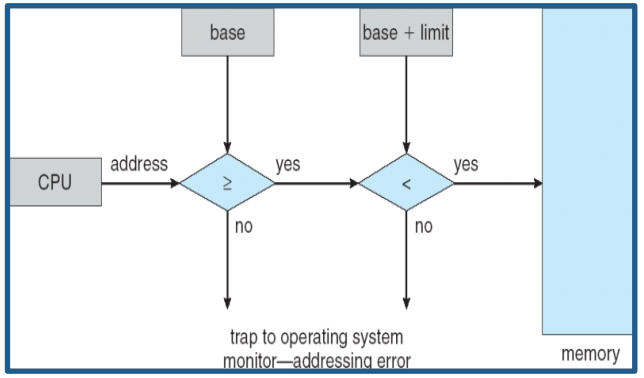
Degree of Multiprogramming

LRU, WSClock: algoritimi di rimozione delle pagine (Last Recently Used, Working Set Clock)



Base&Limit

TLB: tabella che tiene conto dell'**Address Space** e del **Working Set** di ogni processo



Linked List

Page Table: indicizzazione dei **Page Frame** e delle **Virtual Pages**.

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

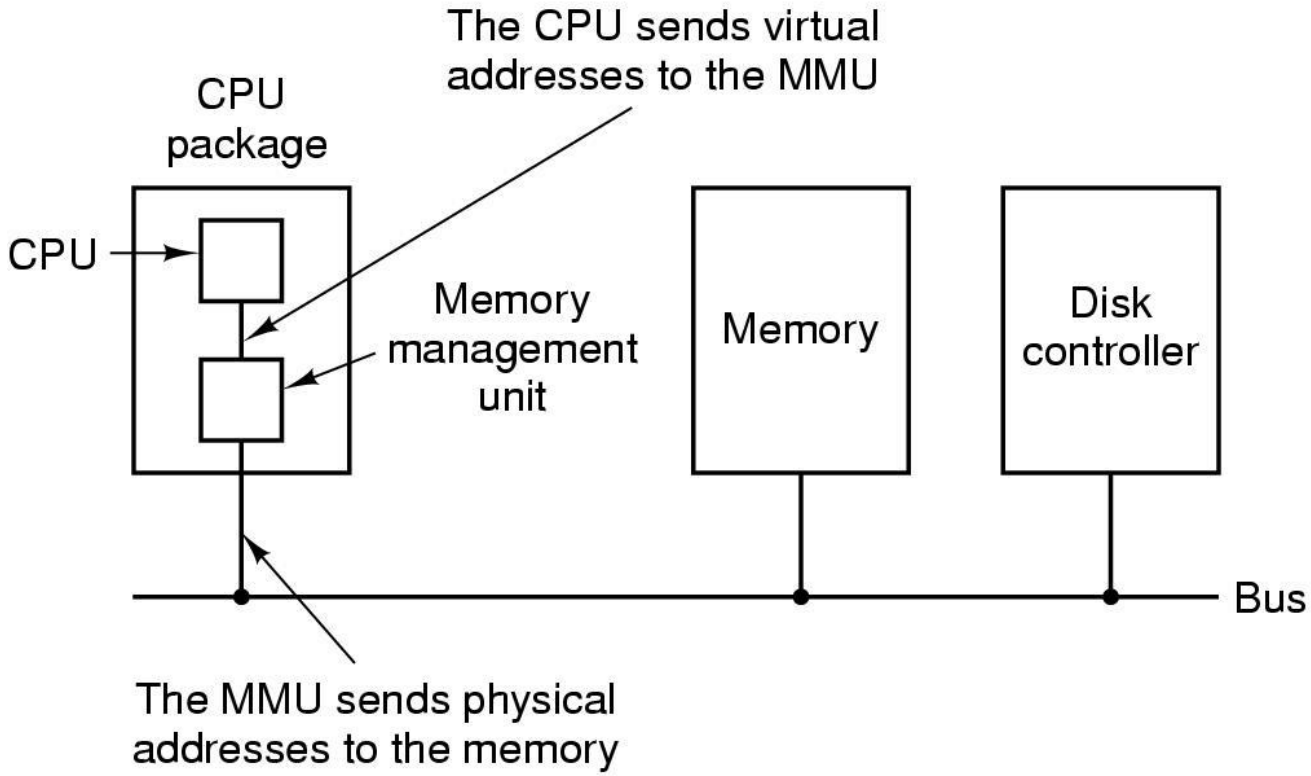
Operating Systems: Memory Management

Virtual Memory: MMU

Memory Manager: mappa gli indirizzi virtuali in indirizzi fisici e li mette sul bus di memoria.

Memory Management Unit

- **Dispositivo hardware** che realizza la trasformazione dagli indirizzi logici a quelli fisici in fase di esecuzione
- •Nello schema di MMU, il valore nel **registro di rilocazione** (r) è aggiunto ad ogni indirizzo logico generato da un processo nel momento in cui è trasmesso alla memoria
- Il **programma** utente interagisce con gli **indirizzi logici** (da 0 a max); non vede mai gli indirizzi fisici reali (da r a r+max)
- In questo modo il programma è indipendente da informazioni specifiche della memoria (valore di rilocazione, capacità della memoria, …)

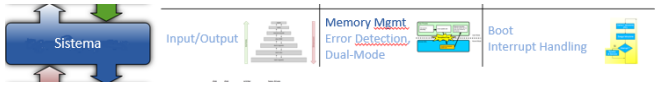


Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management

Virtual Memory: MMU ed Indirizzamento 1/6



Memory Manager: diversi spazi degli indirizzi, da associarsi in diverse fasi.

Indirizzamento

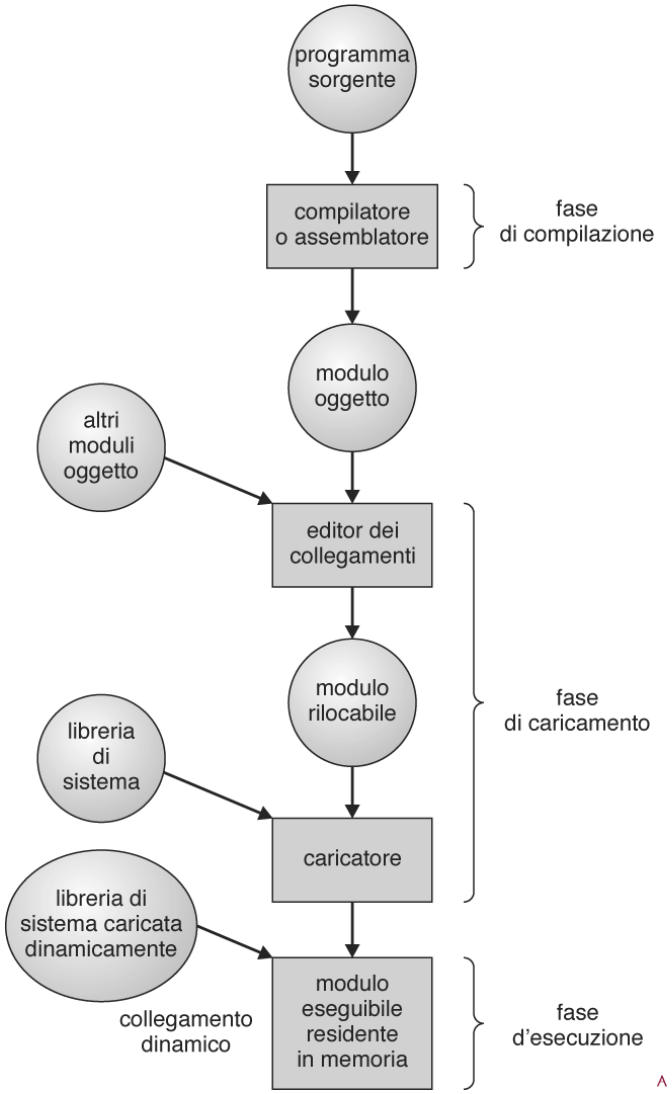
- La CPU esegue programmi utente che elaborano insiemi di dati in base ad una sequenza di istruzioni
- I programmi utente passano attraverso più stadi prima di essere eseguiti, e in tali stadi gli indirizzi cambiano la loro rappresentazione

Spazi degli Indirizzi:

- **Implementazione**: gli indirizzi sono simbolici (e.g. contatori)

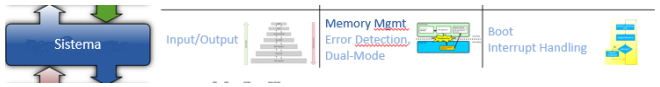
- **Compilazione**: il compilatore associa gli indirizzi simbolici ad indirizzi relativi (binding); Esempio di indirizzo relativo: n byte dall'inizio di un determinato modulo

- **Caricamento** (**run-time**): il linkage editor o il loader trasformano gli indirizzi relativi in indirizzi assoluti



Operating Systems: Memory Management

Virtual Memory: MMU ed Indirizzamento 2/6

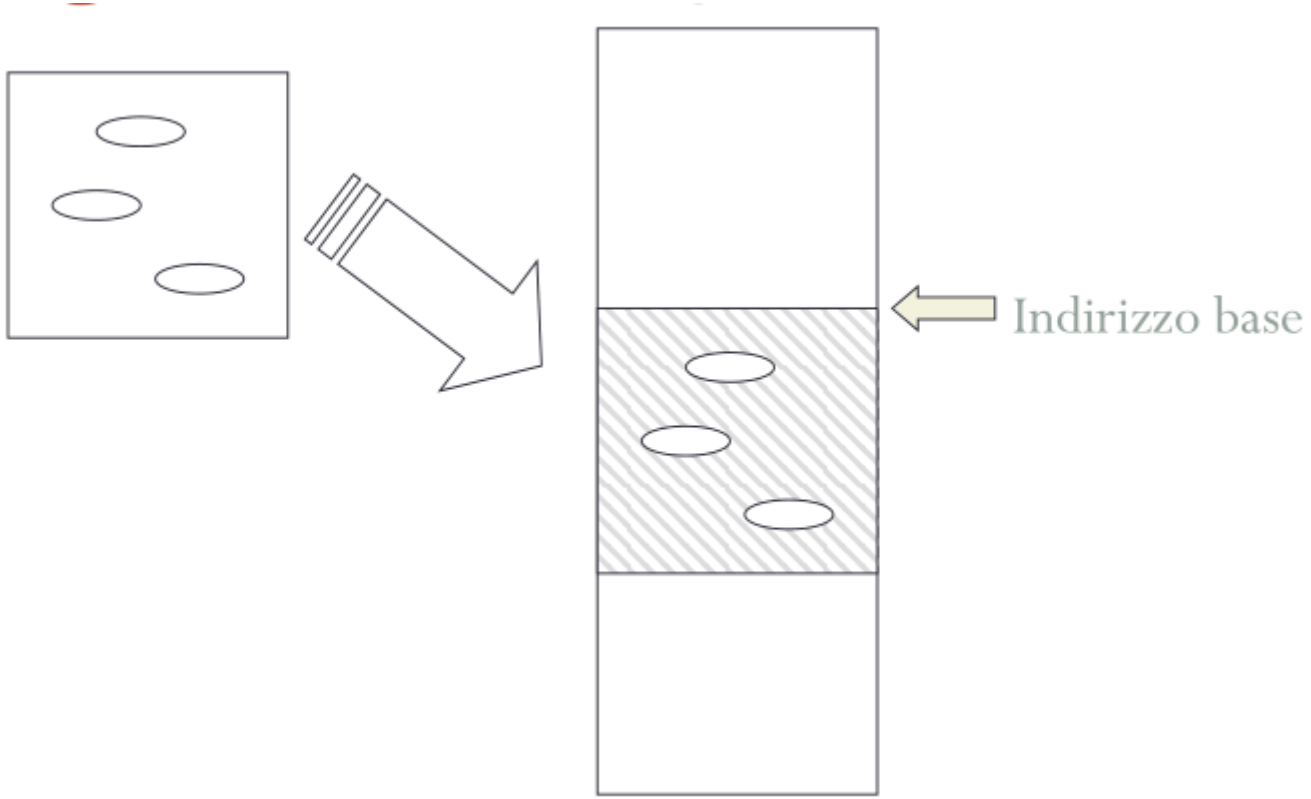


Memory Manager: diversi spazi degli indirizzi, da associarsi in diverse fasi.

Indirizzamento

L'associazione di istruzioni e dati ad indirizzi di memoria può essere
eseguita in diverse fasi:

- **Compilazione**: richiede di conoscere dove il processo risiederà in memoria
  - vengono generati solo indirizzi con riferimento a dove esattamente il codice dovrà risiedere in memoria durante l'esecuzione; **codice assoluto**
  - se la locazione di partenza cambia bisogna compilare di nuovo
  - se invece non si sa dove il processo risiederà è necessario generare del codice rilocabile

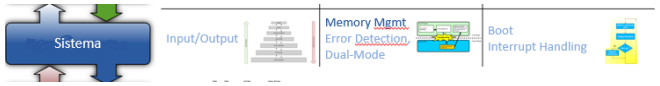


Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management

Virtual Memory: MMU ed Indirizzamento 3/6

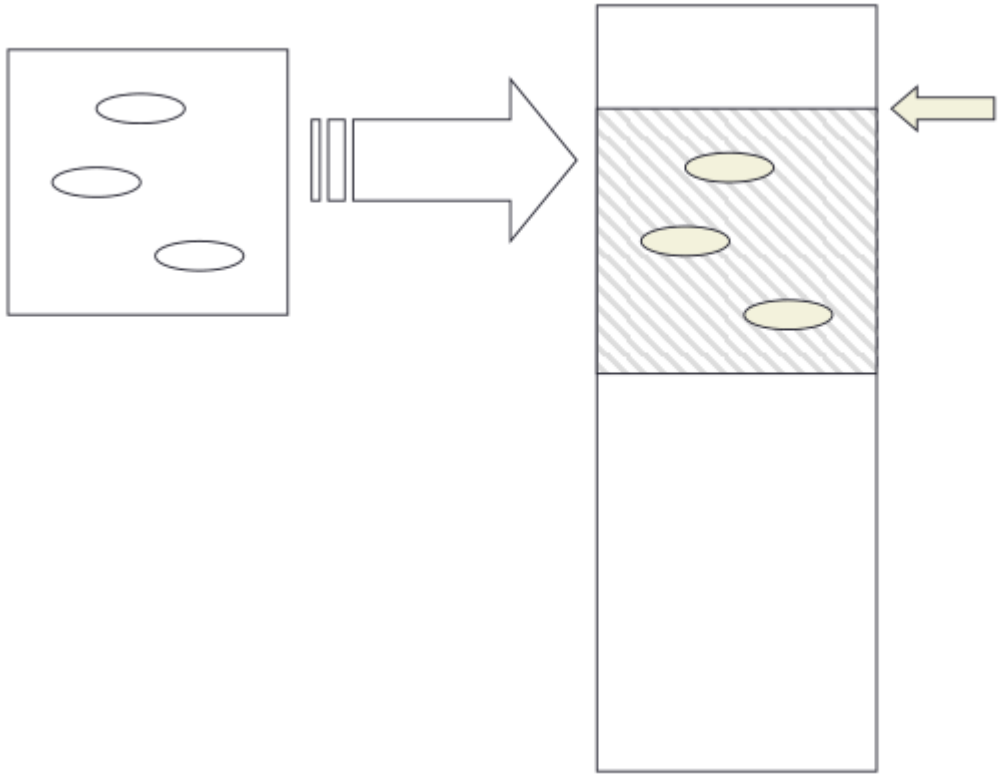


Memory Manager: diversi spazi degli indirizzi, da associarsi in diverse fasi.

Indirizzamento

L'associazione di istruzioni e dati ad indirizzi di memoria può essere eseguita in diverse fasi:

- **Caricamento**: gli eventuali indirizzi rilocabili (gli indirizzi fanno riferimento ad un indirizzo base non specificato) vengono associati ad indirizzi assoluti



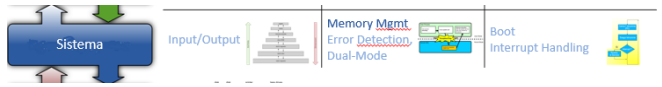
Indirizzo base contenuto nel **registro di rilocazione** (o **registro base**)

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management

Virtual Memory: MMU ed Indirizzamento 4/6



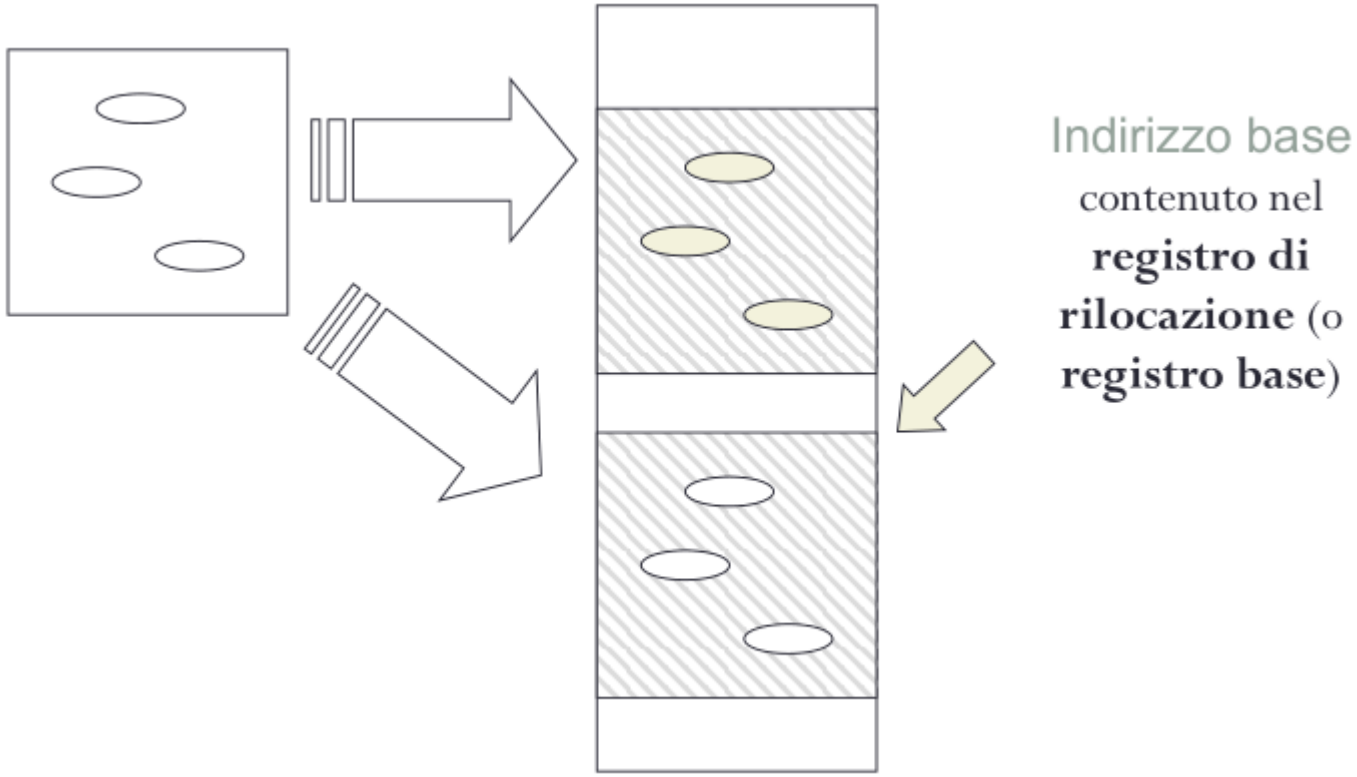
Memory Manager: diversi spazi degli indirizzi, da associarsi in diverse fasi.

Indirizzamento

L'associazione di istruzioni e dati ad indirizzi di memoria può essere eseguita in diverse fasi:

- **Rilocazione**: gli eventuali indirizzi rilocabili (gli indirizzi fanno riferimento ad un indirizzo base non specificato) vengono associati ad indirizzi assoluti (diversi da quelli precedentemente utati)

rilocazione del codice durante il caricamento

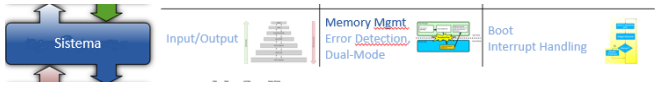


Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Operating Systems: Memory Management

Virtual Memory: MMU ed Indirizzamento 5/6



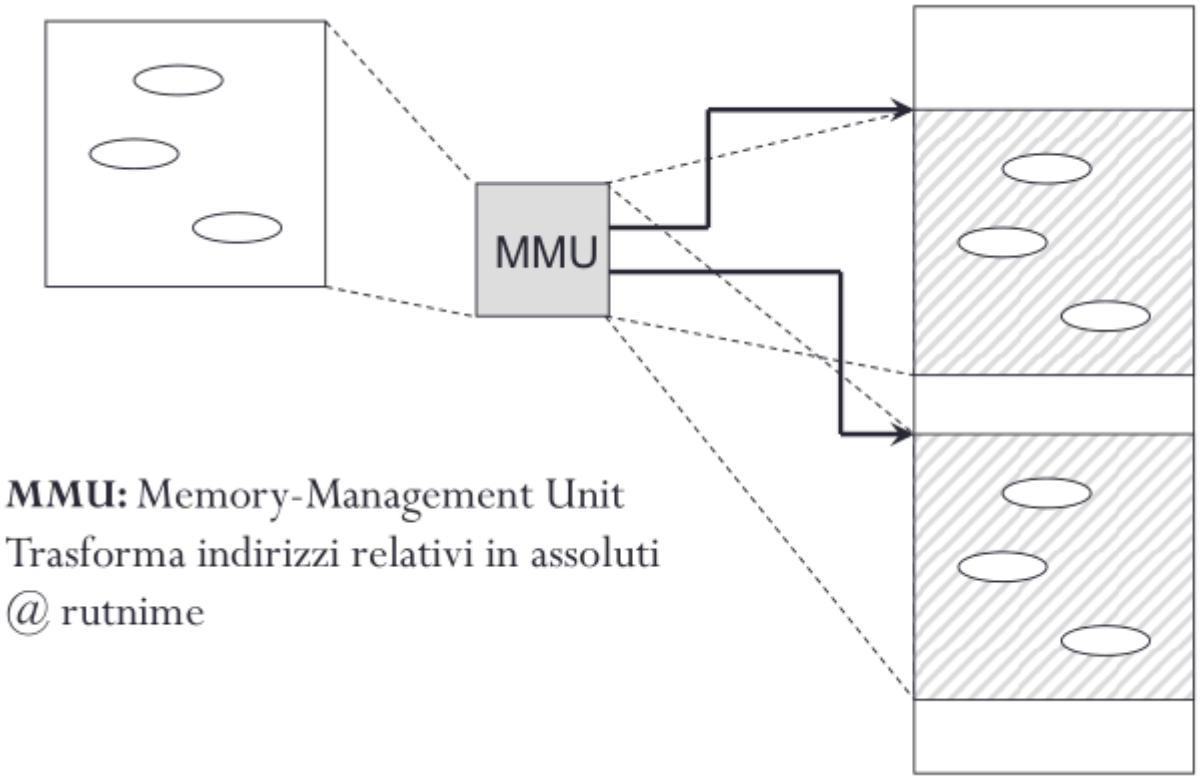
Memory Manager: diversi spazi degli indirizzi, da associarsi in diverse fasi.

Indirizzamento

L'associazione di istruzioni e dati ad indirizzi di memoria può essere eseguita in diverse fasi:

- **Esecuzione**: se il processo può venire spostato in memoria durante l'esecuzione, allora il collegamento deve essere ritardato fino al momento dell'esecuzione; codice dinamicamente rilocabile

- solo riferimenti relativi a se stesso
- Necessita della MMU
- Metodo più utilizzato

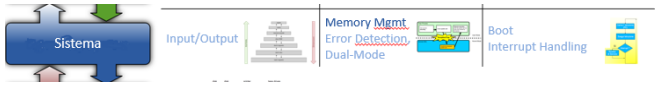


DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

Virtual Memory: MMU ed Indirizzamento 6/6

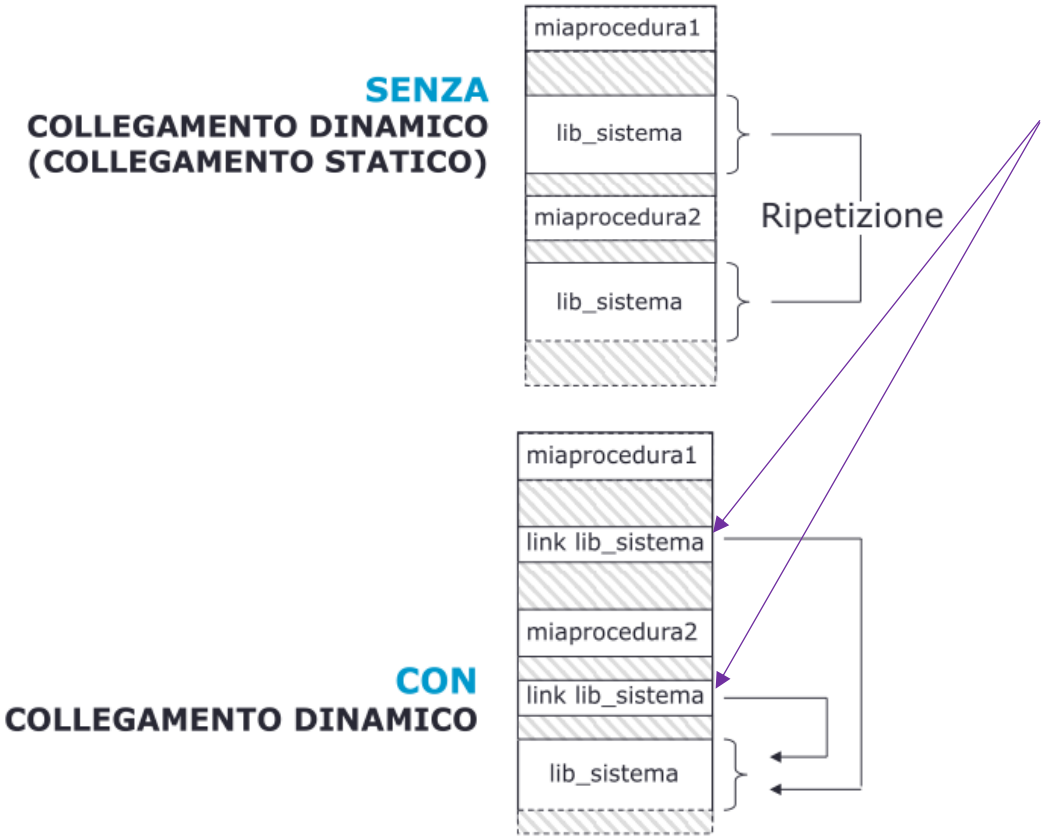


Memory Manager: utilizzo delle librerie esterne (e.g. librerie di sistema) mediante link del codice utente a queste

Collegamento (linking) dinamico

- **Statico**: le librerie esterne vengono incorporate nel file binario di esecuzione (file binari di dimensione molto grande)

- **Dinamico**: le librerie vengono collegate al codice utente solo in fase di esecuzione.



Una piccola parte di codice (**stub**) indica come individuare la procedura di libreria desiderata residente in memoria o come caricarla se non è già presente

- L'immagine rimpiazza se stessa con l'indirizzo della procedura e la esegue
- Si velocizzano eventuali chiamate successive

- Il SO deve controllare se la procedura necessaria è nello spazio di memoria di un altro processo o consentire l'accesso a più processi agli stessi indirizzi di memoria
- Il collegamento dinamico è particolarmente utile con le librerie condivise

Operating Systems: Memory Management

Virtual Memory: Paging 1/7

Memory Manager: suddivide la memoria in porzioni uguali e facilmente usabili.

Paginazione

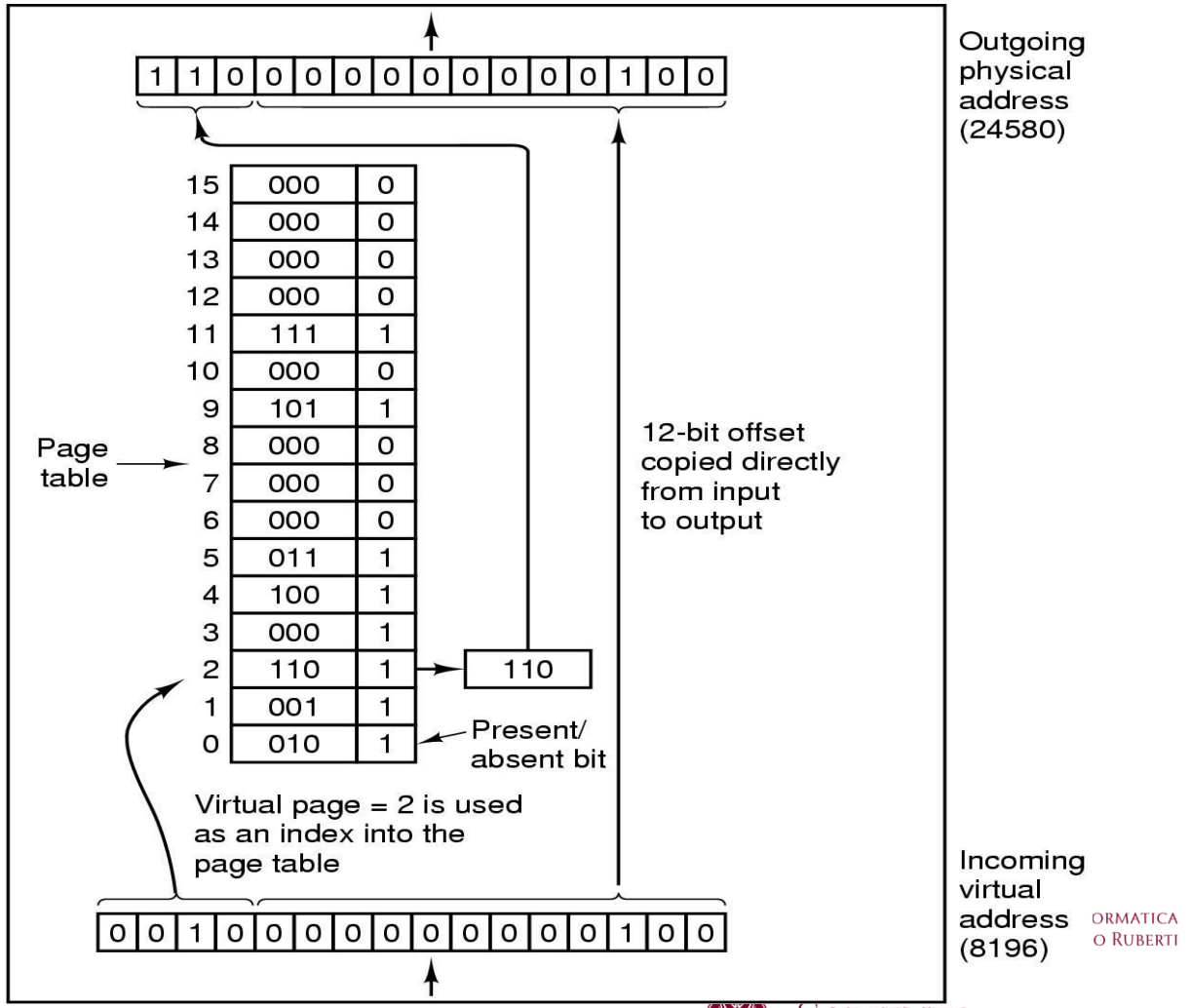
- **Efficienza**: I meccanismi a partizionamento fisso/dinamico non sono efficienti nell'uso della memoria (frammentazione interna/esterna)

- **Riduzione della Frammentazione**: allocando ai processi spazio di memoria non contiguo

- **Aumento del Degree of Multiprogramming**: si tiene in memoria solo una porzione del programma, aumentando il numero dei processi che possono essere contemporaneamente presenti in memoria

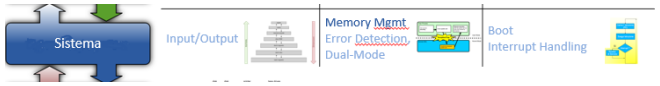
- **Memoria Virtuale**: possibilità di eseguire un processo più grande della memoria disponibile

- **MMU**: pieno utilizzo del dispositivo HW



Operating Systems: Memory Management

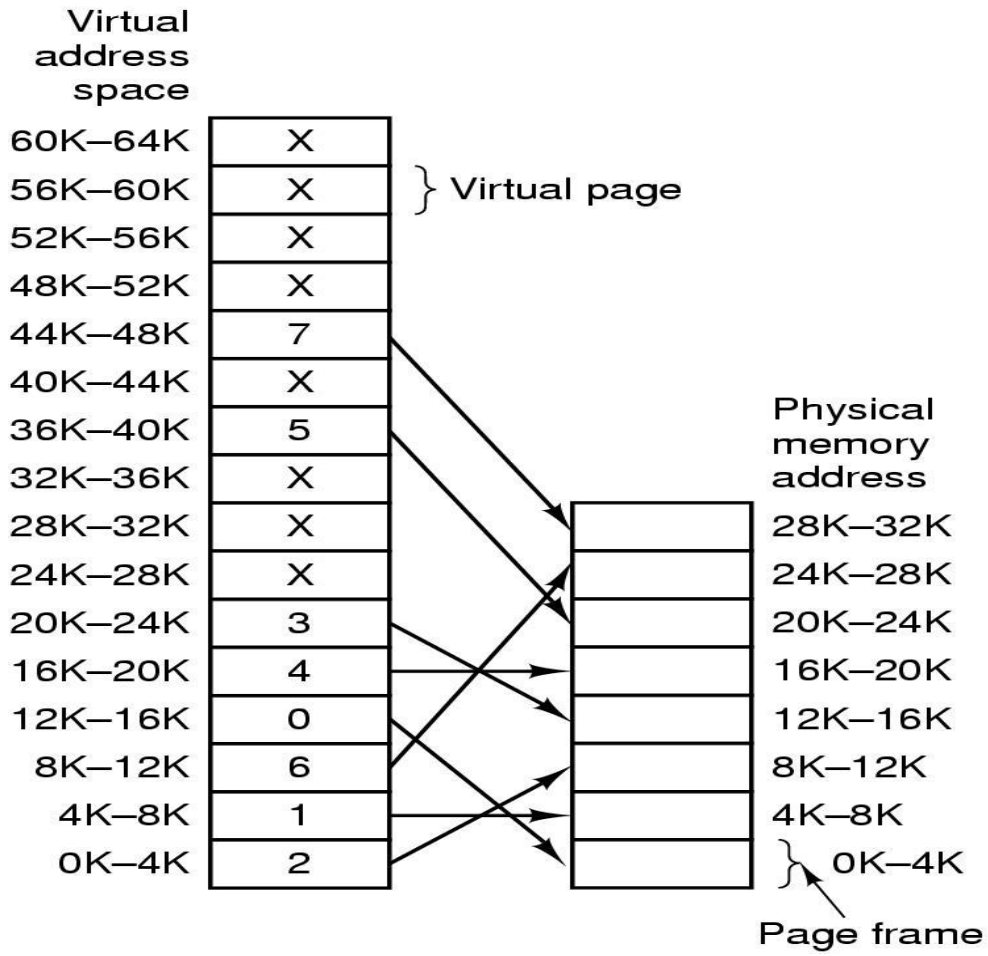
Virtual Memory: Paging 2/7



Memory Manager: suddivide la memoria in porzioni uguali e facilmente usabili.

Pagine Logiche <-> Fisiche

- Suddivide la memoria fisica (memoria centrale) in blocchi di **frame** della stessa dimensione (una potenza di 2, fra 512 byte e 16 MB). Lo spazio degli indirizzi fisici può essere non contiguo
- Divide la memoria logica in blocchi delle stesse dimensioni dei frame chiamati **pagine**. Il processo è sempre allocato all'interno della memoria logica in uno spazio contiguo
- Per eseguire un processo di dimensione di n pagine, bisogna trovare n frame liberi e caricare il programma
- Il SO imposta una **tabella** delle pagine per **tradurre** gli indirizzi logici in indirizzi fisici
- La tabella può avere grandi dimensioni
- La traduzione deve essere veloce



16 bit addresses,

4 KB pages

32 KB physical memory,

16 virtual pages

8 page frames

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

Virtual Memory: Paging 3/7

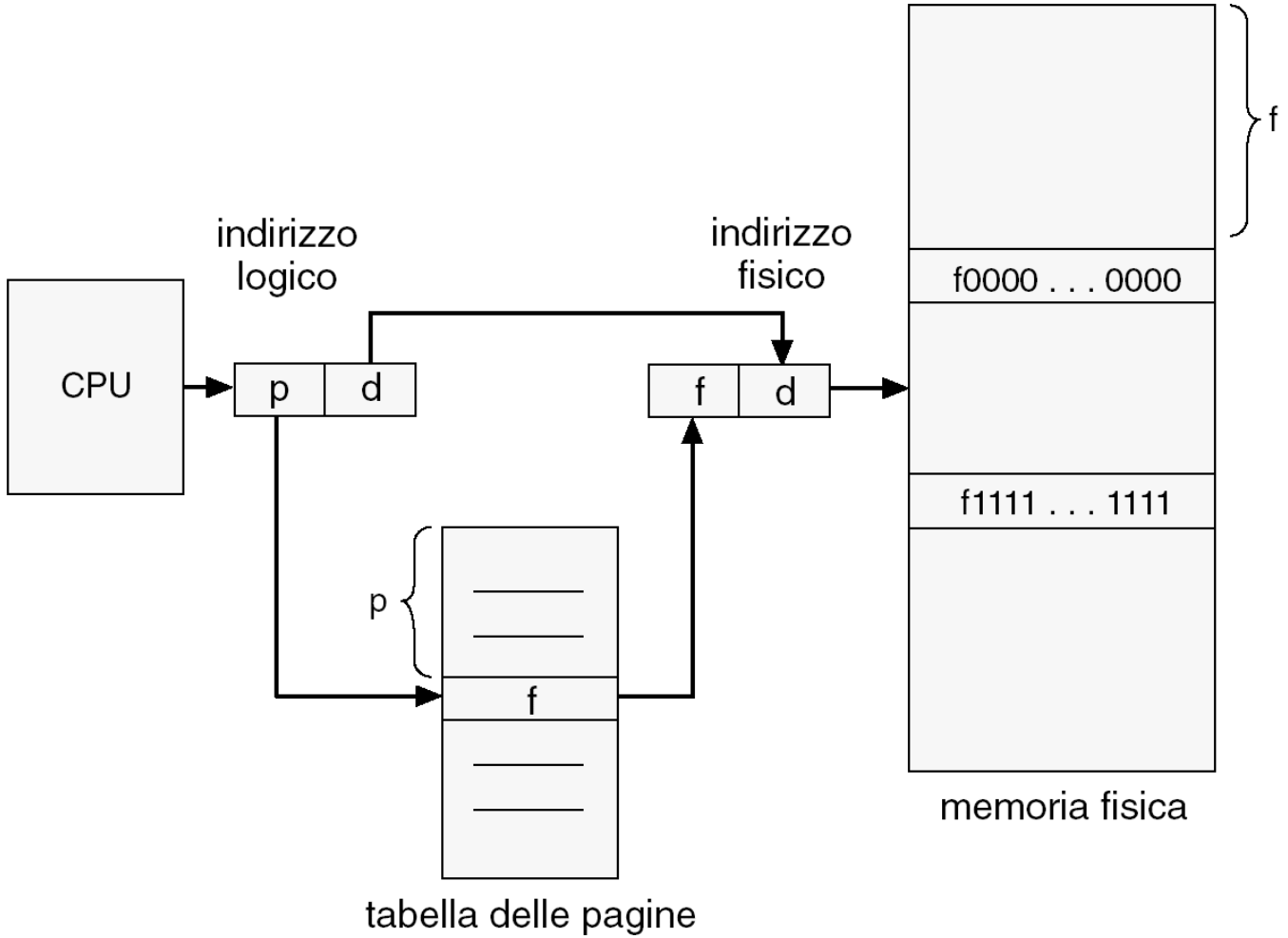
Memory Manager: suddivide la memoria in porzioni uguali e facilmente usabili.

Traduzione Indirizzi Logici <-> Fisici

Ogni indirizzo logico generato dalla CPU è diviso in due parti:

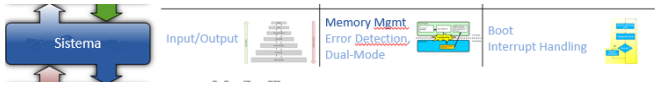
Indirizzo logico

- **Numero di pagina (p)**: usato come indice nella tabella delle pagine che contiene l'indirizzo di base di ogni frame nella memoria fisica
- **Spiazzamento** (**Displacement**) **nella pagina (d)**: combinato con l'indirizzo di base per calcolare l'indirizzo di memoria fisica che viene mandato all'unità di memoria centrale



Operating Systems: Memory Management

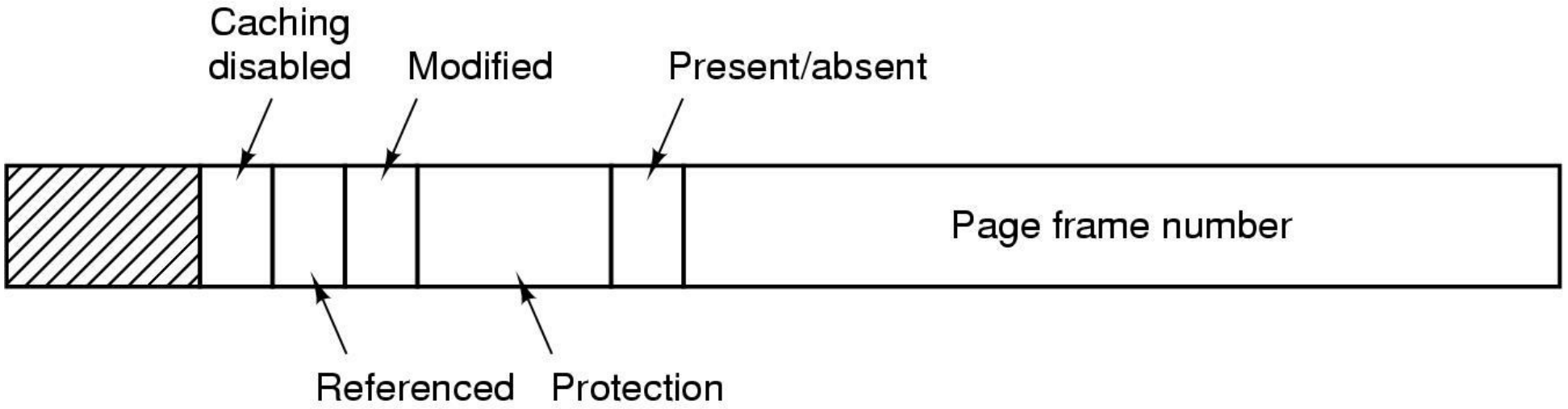
Virtual Memory: Paging 4/7



Memory Manager: suddivide la memoria in porzioni uguali e facilmente usabili.

Tabella delle Pagine (referenziata nel PCB)

Oltre a Page (p) e Displacement (d), per ogni pagina referenziata contiene le informazioni



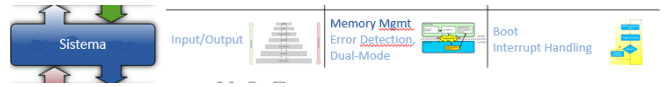
- **Present/Absent**: attualmente contenuta in memoria oppure no (➔ necessario accesso a disco)
- **Protection**: modalità di protezione (es. r, w, x)
- **Modified**: modificata di recente (deve essere salvata su disco) oppure no. Detto anche «**Dirty Bit**»
- **Referenced**: è stata letta di recente (probabilmente lo sarà ancora: non conviene segregarla su disco)
- **Caching Disabled**: la copia su memoria è aggiornata rispetto alla quella in cache

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

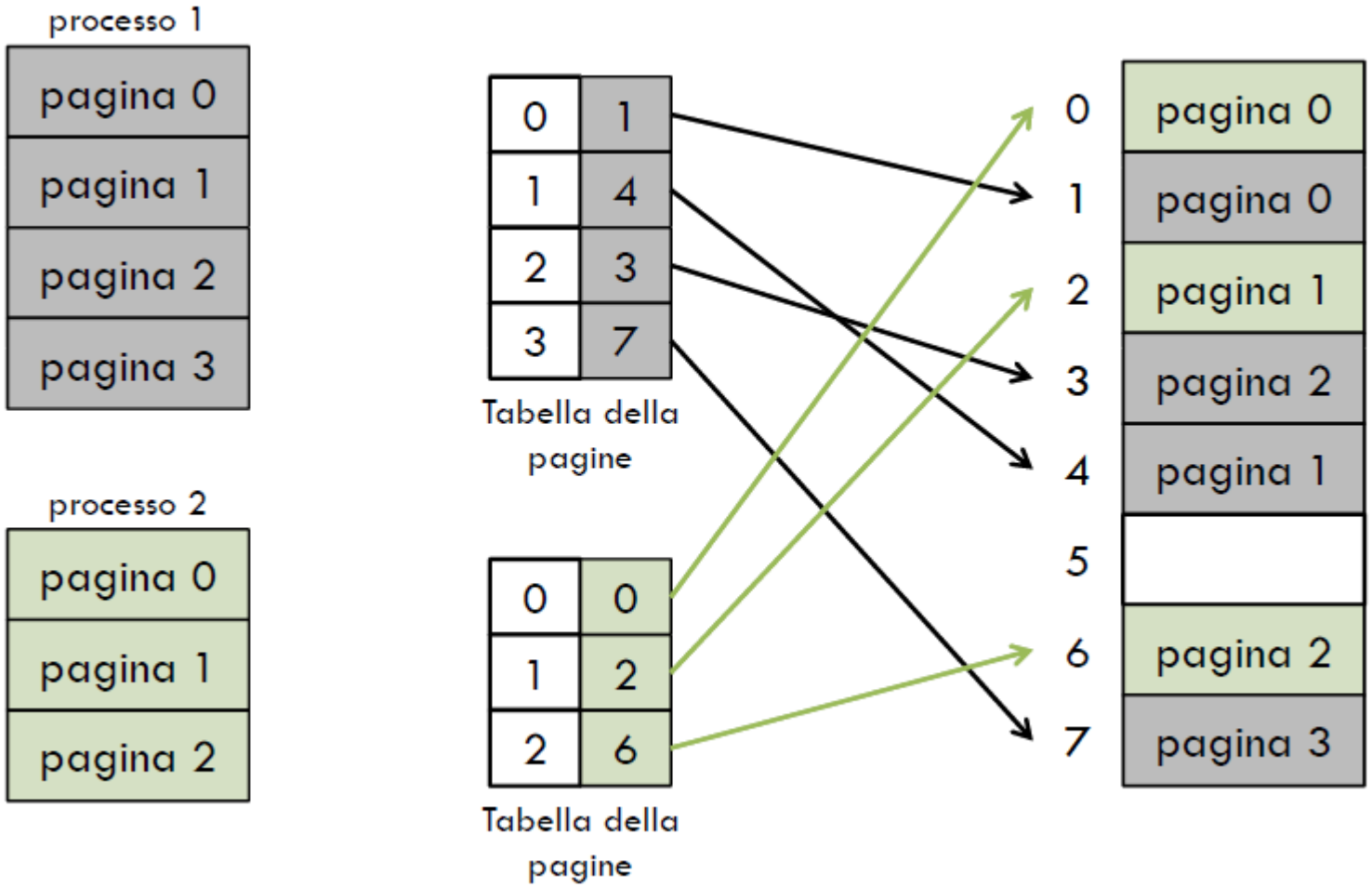
SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

Virtual Memory: Paging 5/7



Memory Manager: suddivide la memoria in porzioni uguali e facilmente usabili.



Esempio

La tabella delle pagine associa l'indirizzo di una pagina logica al corrispettivo indirizzo della pagina nella memoria fisica

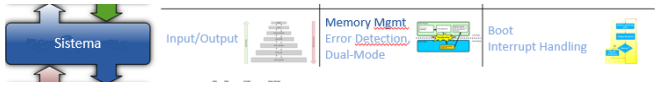
Le aree di memoria dei processi possono essere interfogliate

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management

Virtual Memory: Paging 6/7



Memory Manager: suddivide la memoria in porzioni uguali e facilmente usabili.

Vantaggi della Paginazione

- La paginazione implementa automaticamente una forma di **protezione dello spazio di indirizzamento**:
  ➔ Un programma può indirizzare solo i frame contenuti nella sua tabella delle pagine
- No frammentazione esterna
- Si frammentazione interna
  - L'ultimo frame assegnato però potrebbe non essere completamente occupato causando frammentazione interna
  - Il caso peggiore si verifica quando un processo è un multiplo della dimensione della pagina + 1 byte (un frame è completamente sprecato per l'ultimo byte)
- Pagine più grandi causano maggiore frammentazione interna, ma permettono di avere tabelle delle pagine più piccole (meno occupazione di memoria) e I/O più efficiente

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

Virtual Memory: Paging 7a/7

Memory Manager: strutture avanzate per la tabella delle pagine.

Paginazioni Alterrnative

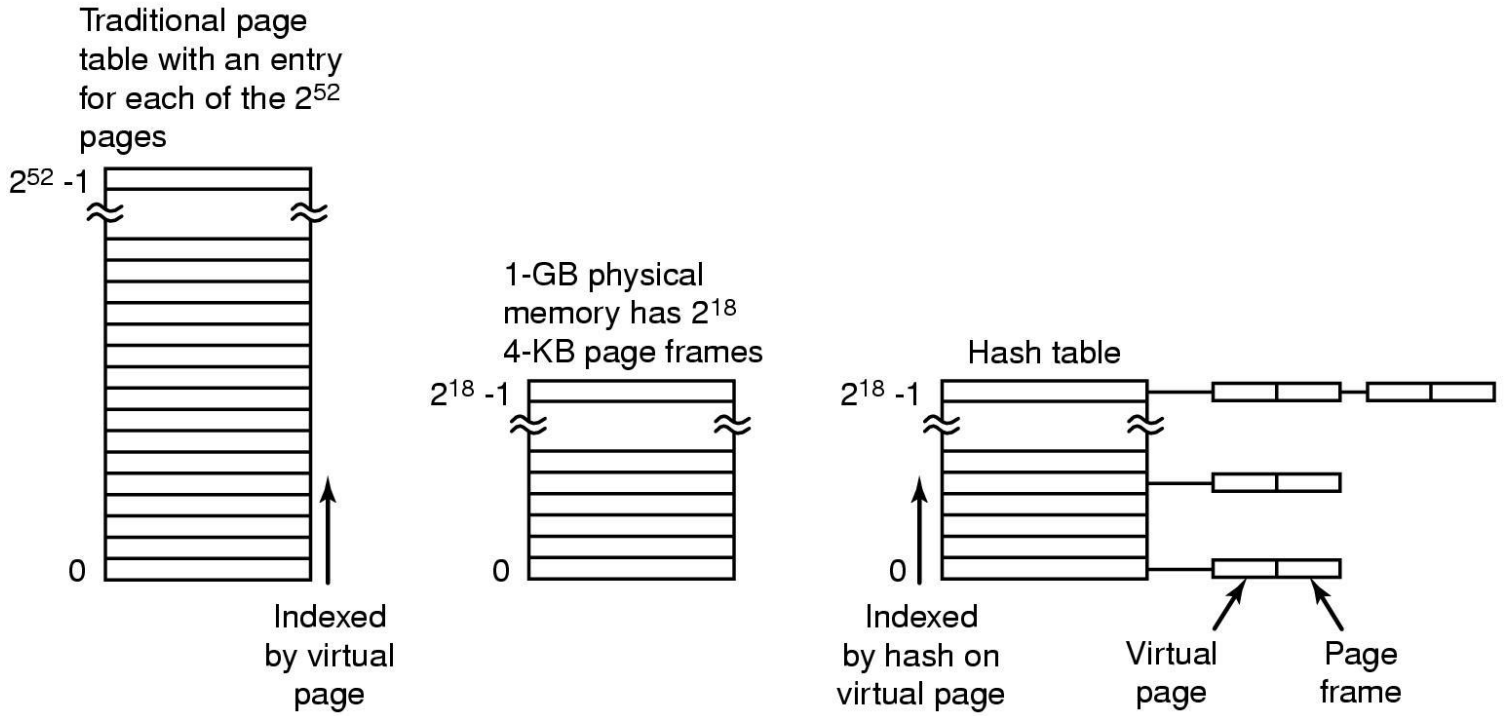
Nei computer moderni che supportano un vasto spazio di indirizzamento logico (da 2^{32} a 2^{64}), la tabella delle pagine è eccessivamente grande

(es. architettura a 32 bit con dimensione di pagina di 4 KB (2^{12}):

- la tabella può contenere fino a 1 milione di elementi (2^{32} / 2^{12} = 2^{20} = 1,048,576)
- ogni elemento occupa 4 Byte si ha una tabella da 4MB per ogni processo

➔ **Soluzione gerarchica**: suddividere lo spazio degli indirizzi logici in più tabelle di pagine (es. 2 Livelli)

➔ **Soluzione Hash**: Tabella le chiavi sono hash, contenente indirizzo di pagina e frame

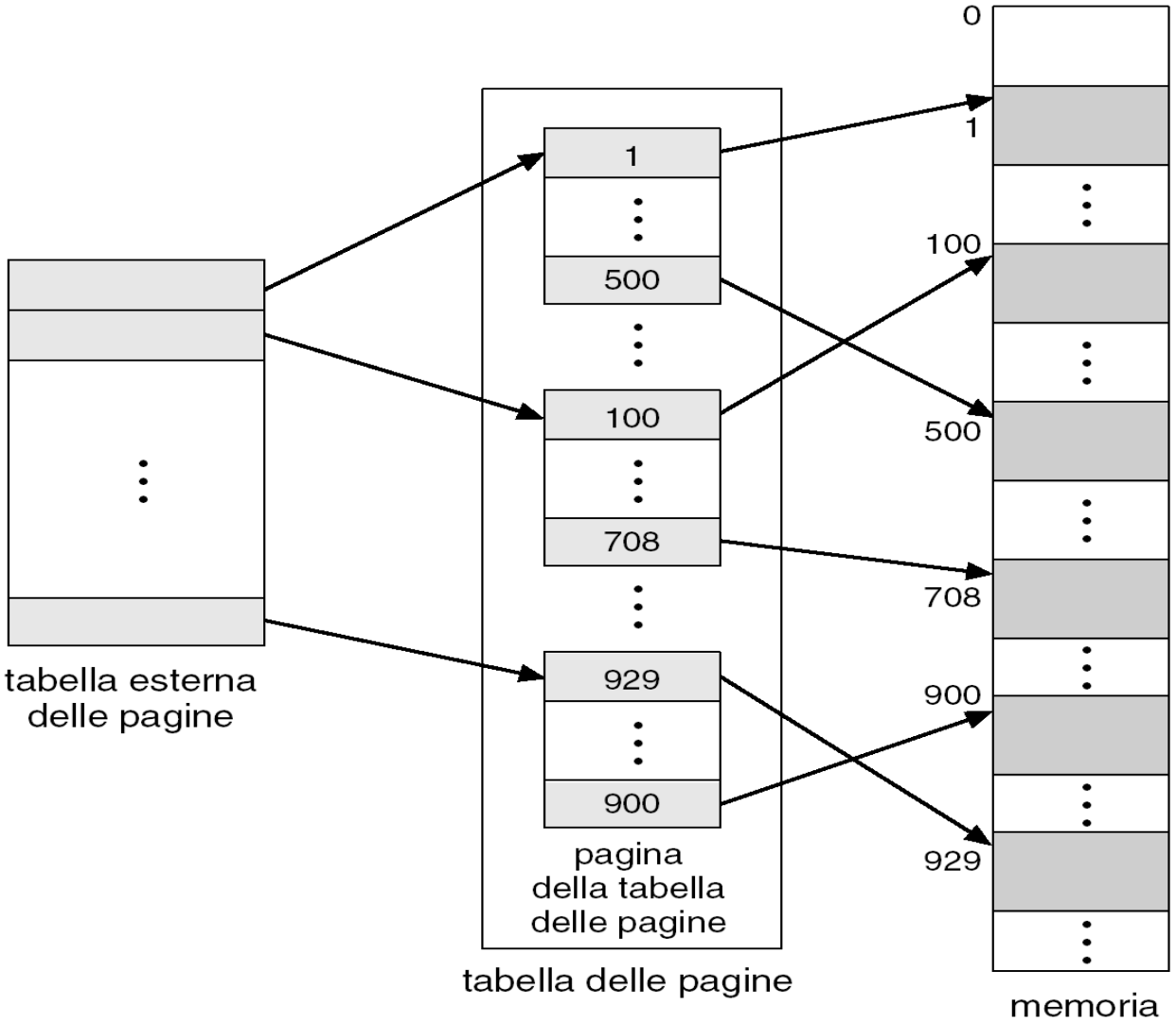


➔ **Soluzione a Tabella Invertita**: tabella dei frame, non delle pagine (si risparmia spazio, ma si perde in efficienza per cercare nella IPT l'entry che contiene la coppia (pid, p) Poiché la tabella è ordinata per indirizzo fisico e può essere necessario scorrerla tutta per trovare l'indirizzo logico desiderato

Operating Systems: Memory Management

Virtual Memory: Paging 7c/7

Memory Manager: strutture avanzate per la tabella delle pagine.



Paginazione Gerarchica

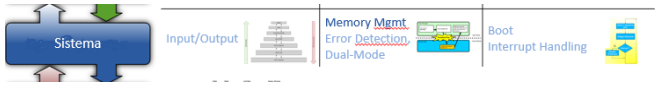
- **Soluzione gerarchica**: suddividere lo spazio degli indirizzi logici in più tabelle di pagine (es. 2 Livelli)
- Una directory delle pagine detta tabella esterna: ogni elemento punta ad una sotto-tabella delle pagine
- Un insieme di sotto-tabelle delle pagine (Tipicamente ampia quanto una pagina al fine di essere completamente contenuta in un frame)

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management

Virtual Memory: Paging 7d/7



Memory Manager: strutture avanzate per la tabella delle pagine.

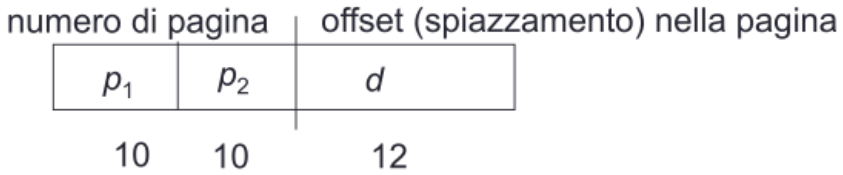
Esempio di Paginazione a 2 Livelli (Forward-mapped Page Table)

Un indirizzo logico su una macchina a 32-bit con pagine di 4K è diviso in un numero di pagina di 20 bit e un offset di 12 bit

Paginazione a due livelli ➔ numero di pagina ulteriormente diviso in :

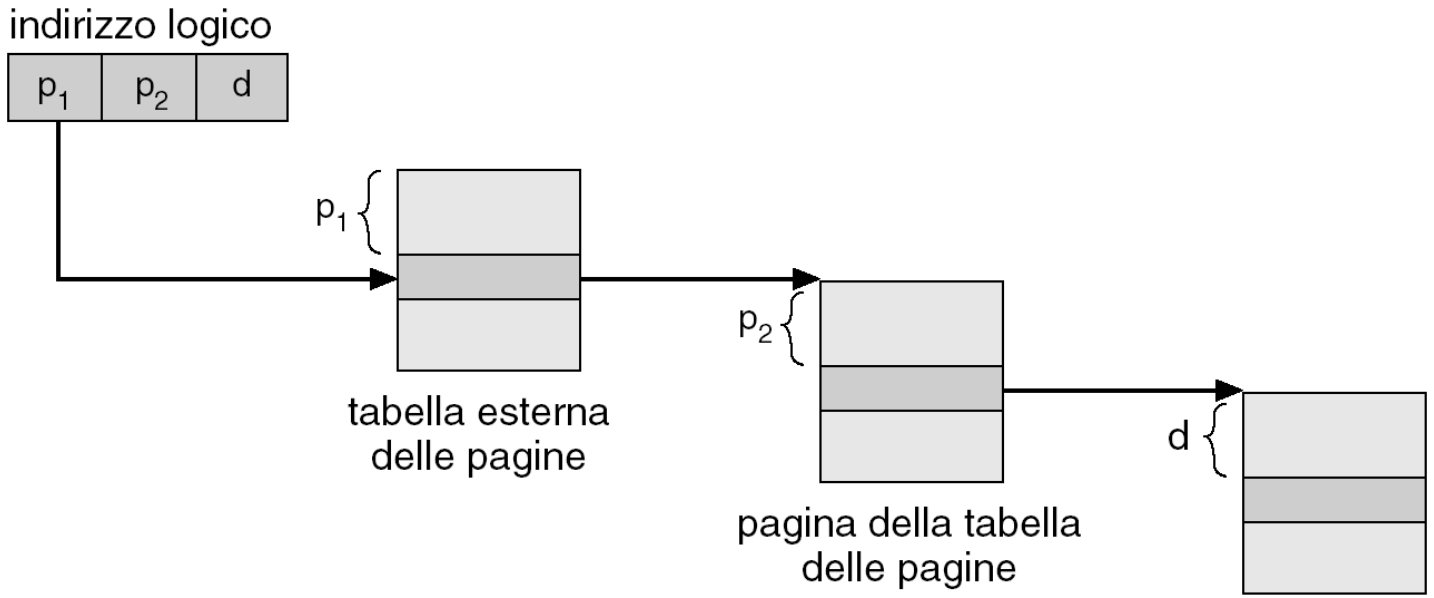
- p_1: numero di pagina da 10-bit
- p_2: spiazzamento nella pagina da 10-bit

Un indirizzo logico è diviso:



➔ p_1: indice della tabella esterna

➔ p_2: spostamento all'interno della pagina della tabella esterna



Ogni livello aggiunto implica un accesso alla memoria per la traduzione degli indirizzi ➔ raramente si superano i 3 livelli (SPARC a 32 bit)

Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti

Sapienza
Università di Roma

Operating Systems: Memory Management

Virtual Memory: Paging 7e/7

Memory Manager: strutture avanzate per la tabella delle pagine.

Tabelle delle Pagine con Hashing

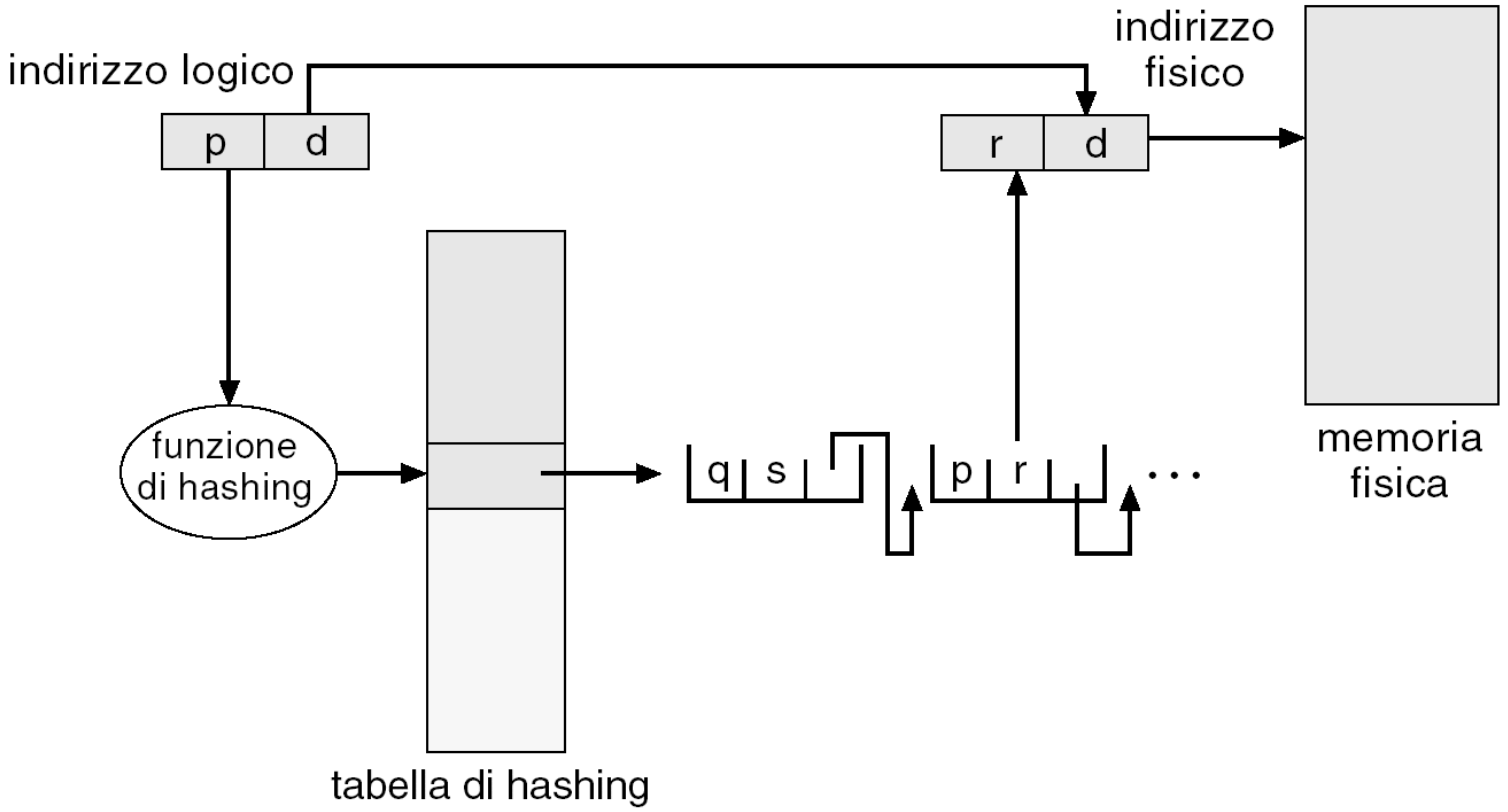
Comune per trattare gli spazi di indirizzamento più grandi di 32 bit :

- **Chiave**: hash dell'indirizzo virtuale
- **Valore**: una lista di elementi formati da (a) indirizzo virtuale, (b) indirizzo della pagina fisica, (c) puntatore al prossimo elemento

I numeri di pagina virtuali sono confrontati

con il campo (a) degli elementi della lista.

Se viene trovata una corrispondenza, il corrispondente frame fisico viene estratto.



Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management

Virtual Memory: Paging 7f/7

Memory Manager: strutture avanzate per la tabella delle pagine.

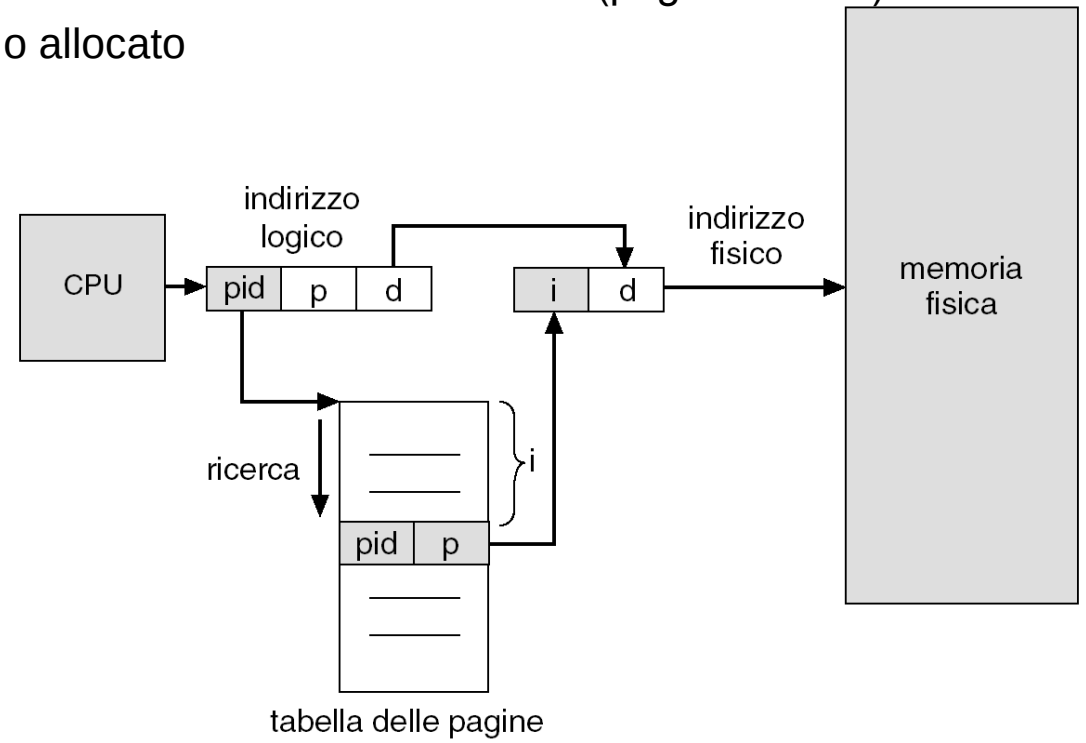
Tabella dei Frame (Inverted Page Table)

Oltre alla tabella delle pagine (logiche) di ciascun processo, il SO mantiene un'unica tabella dei frame (pagine fisiche):

- contiene un elemento per ogni frame, che indica se questo è libero o allocato
- (se allocato) a quale pagina di quale processo o processi

Una IPT descrive l'occupazione dei frame della memoria fisica con una entry per ogni frame:

- C'è una sola IPT per tutto il sistema (anziché una PT per processo)
- La dimensione della IPT dipende strettamente dalla dimensione della memoria primaria
- L'indice di ogni elemento della IPT corrisponde al numero del frame corrispondente



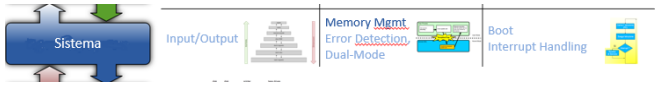
Ogni entry della IPT contiene una coppia (process-id, page-number)

- **process-id**: identifica il processo che possiede la pagina
- **page-number**: indirizzo logico della pagina contenuta nel frame corrispondente a quella entry

Si cerca nella IPT la coppia (pid, p), se la si trova all'i-esimo elemento, si genera l'indirizzo fisico (i,d), altrimenti un page-fault

Operating Systems: Memory Management

Virtual Memory: TLB 1/7



Memory Manager: una tabella delle pagine per ogni processo.

Tabella delle pagine nei Registri

specifico insieme di registri per salvare la tabella:

- Durante il context switch i valori della tabella delle pagine del processo vengono caricate nei registri
- Molto veloce
- Ma va bene solo per tabelle con pochi elementi (256 circa, quando tipicamente ne servono 1.000.000)

Tabella delle pagine in RAM + PTBR (Page-Table Base Register)

La tabella delle pagine viene mantenuta in memoria centrale:

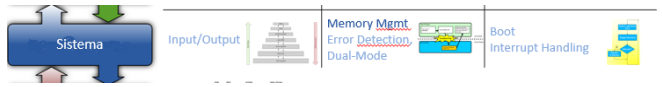
- Un registro chiamato page-table base register (PTBR) punta alla tabella del processo corrente
- Cambiare tabella delle pagine significa cambiare valore del registro
- Per accedere ad un dato occorrono due accessi alla memoria centrale
  - 1 per la tabella delle pagine
  - 1 per il byte stesso
- L'accesso alla memoria è rallentato di un fattore 2 (➔ preferibile lo swapping)

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

Virtual Memory: TLB 2/7

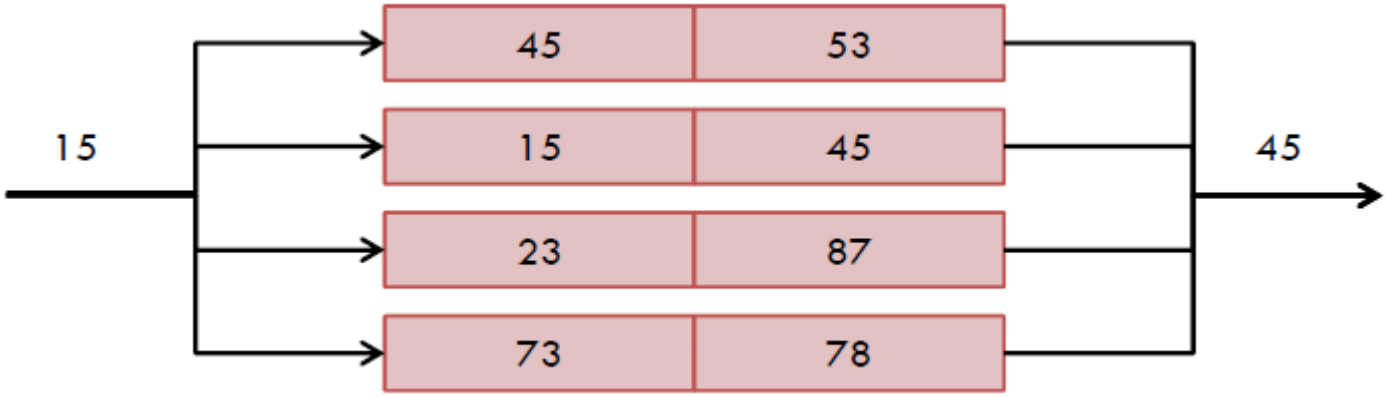


Memory Manager: una tabella delle pagine per ogni processo.

Translation Lookaside Buffer (TLB)

cache associativa: contiene una piccola parte delle righe della tabella delle pagine

- Associa ad una chiave un valore
  - Chiave: indirizzo logico - Valore: indirizzo fisico
  - Molto rapida ma anche costosa e piccola (max 1024 elementi)
- I record possono essere scritti e sovrascritti
- Ma possono anche essere vincolati (non modificabili)

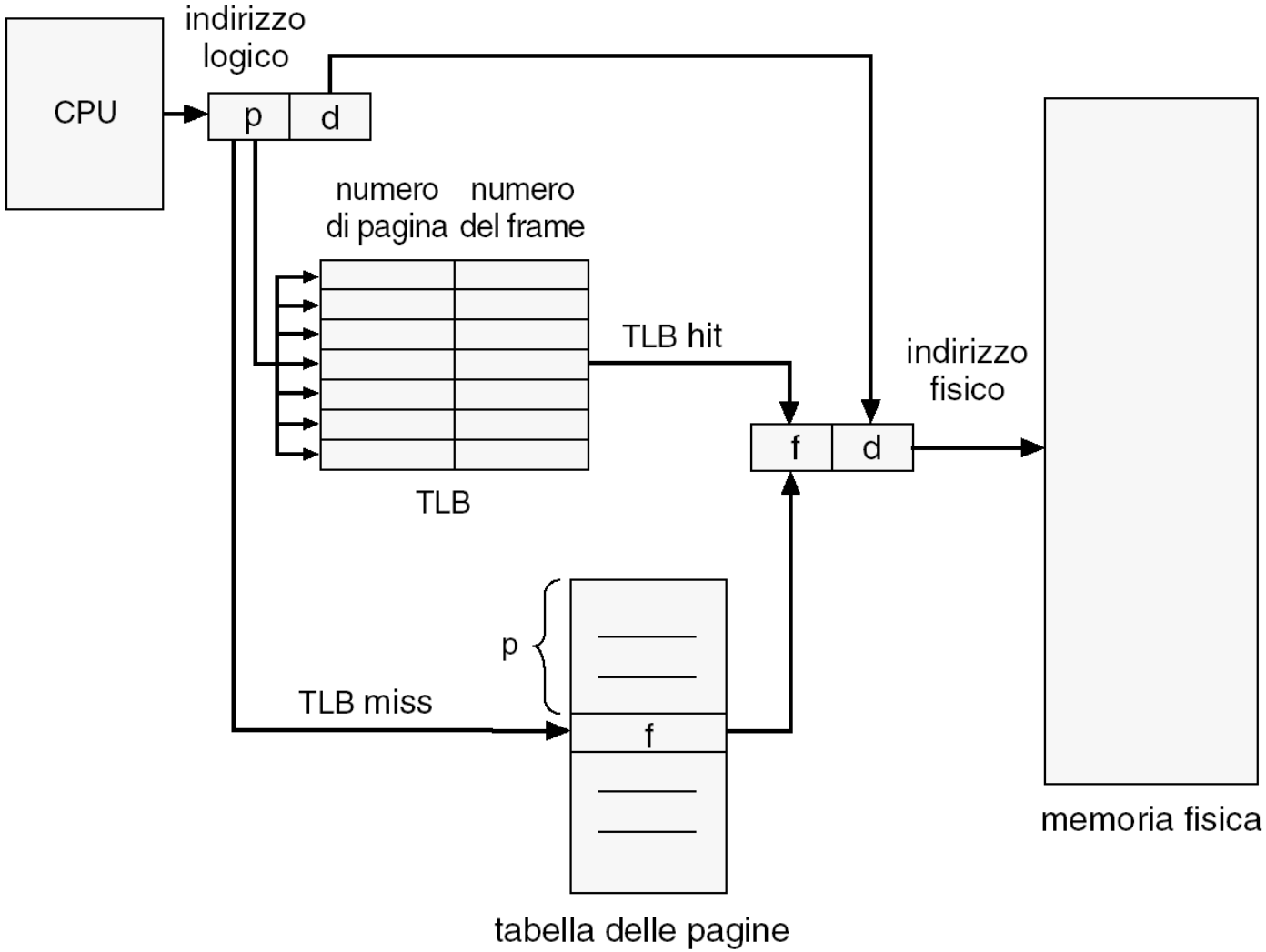


- Dato un indirizzo logico lo si passa alla TLB, se essa contiene tale chiave restituisce l'indirizzo fisico
- Altrimenti (TLB Miss) si cerca nella tabella residente in memoria centrale
- Si copiano poi i due indirizzi nella TLB in modo da velocizzare accessi futuri
- Se la tabella è piena si sceglie quale record togliere secondo diversi criteri (e.g. elemento usato meno di recente, casuale)

Operating Systems: Memory Management

Virtual Memory: TLB 3/7

Memory Manager: una tabella delle pagine per ogni processo.



Translation Lookaside Buffer (TLB)

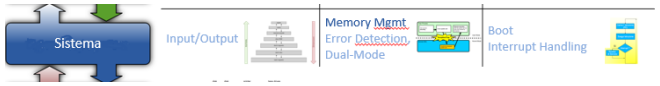
- Dato un indirizzo logico lo si passa alla TLB, se essa contiene tale chiave restituisce l'indirizzo fisico
- Altrimenti (TLB Miss) si cerca nella tabella residente in memoria centrale
- Si copiano poi i due indirizzi nella TLB in modo da velocizzare accessi futuri
- Se la tabella è piena si sceglie quale record togliere secondo diversi criteri (e.g. elemento usato meno di recente, casuale)

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management

Virtual Memory: TLB 4/7



Memory Manager: una tabella delle pagine per ogni processo.

Translation Lookaside Buffer (TLB)

Contiene anche le informazioni relative a:

- Protezione
- Modifica

desunte dalla tabella delle pagine

Il bit «Valid» indica se la pagina è in uso o no

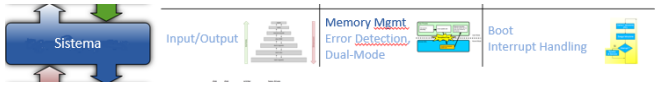
Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

Virtual Memory: TLB 5/7



Memory Manager: una tabella delle pagine per ogni processo.

TLB gestito via Software

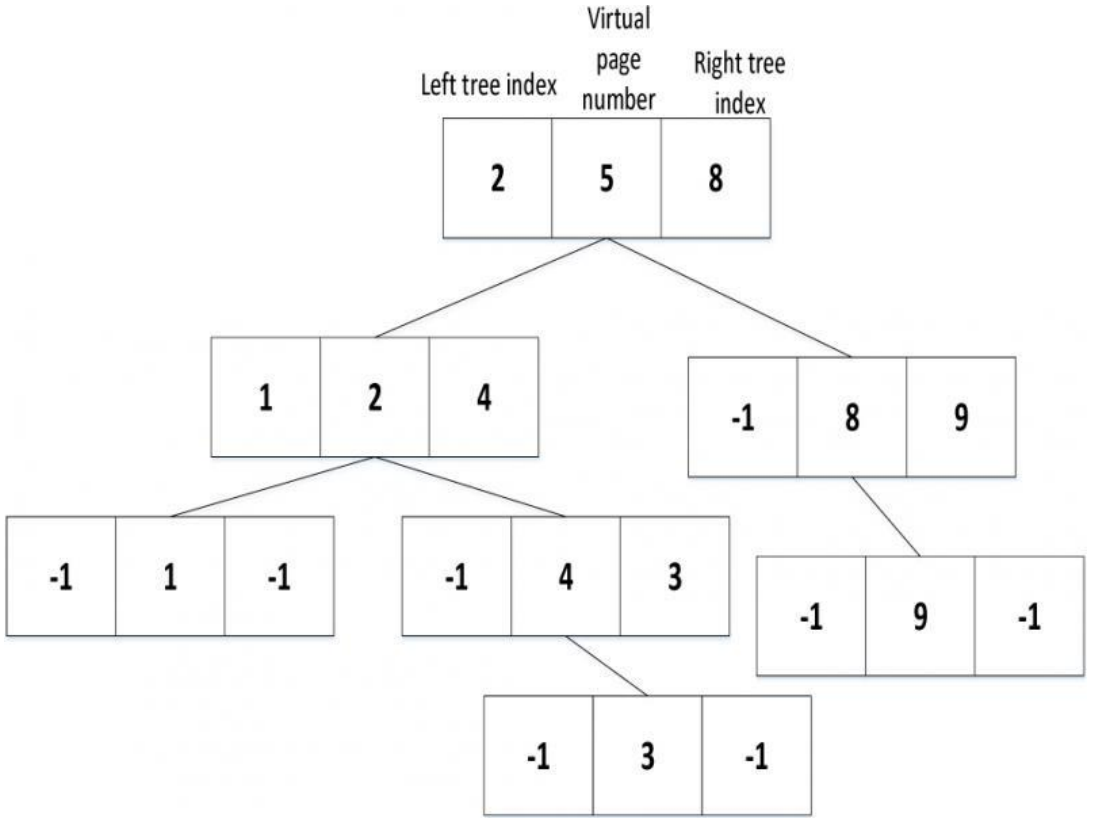
Le macchine RISC gestiscono TLB via software

TLB fault processato dal sistema operativo invece che dall'hardware della MMU

Risultati meno hardware in MMU e prestazioni OK

Il software può capire quali pagine precaricare nel TLB (es. server load dopo la richiesta del client)

Mantiene la cache delle pagine usate frequentemente

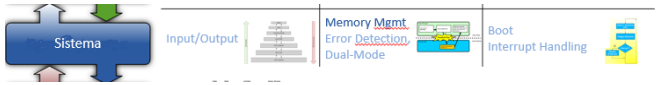


DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

Virtual Memory: TLB 6/7



Memory Manager: una tabella delle pagine per ogni processo.

Tempo di Accesso Effettivo (EAT: Effective Access Time) al TLB

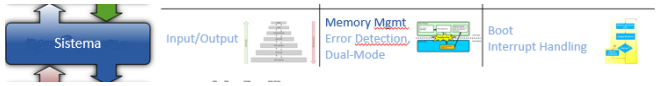
- Tempo di ricerca nella TLB = ε unità di tempo
- Tempo di accesso alla memoria = c unità di tempo
- Tasso di accesso con successo alla TLB (***hit ratio***) = α
  - percentuale delle volte che un numero di pagina si trova nella TLB

- Tempo di accesso effettivo (EAT)

$$\begin{aligned} \text{EAT} &= (c + \varepsilon)\,\alpha + (2c + \varepsilon)(1 - \alpha) \\ &= 2c + \varepsilon - c\,\alpha \end{aligned}$$

Operating Systems: Memory Management

Virtual Memory: TLB 7/7



Memory Manager: una tabella delle pagine per ogni processo.

Tempo di Accesso Effettivo (EAT: Effective Access Time) al TLB: esempio

- Tempo di ricerca nella TLB $= \varepsilon = 20$ ns
- Tempo di accesso alla memoria $= c = 100$ ns
- Tasso di accesso con successo $\alpha = 0.80$ (80%)

$$\mathbf{EAT} = 2c + \varepsilon - c\alpha = (200 + 20 - 80)\ \text{ns} = 140\text{ns}$$

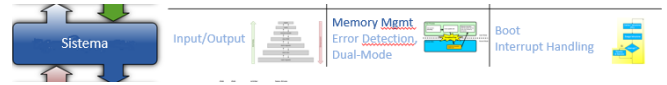
- Si ha un rallentamento del 40% del tempo di accesso alla memoria centrale (da 100 a 140 ns)
  - più alto è l'hit ratio α, più piccolo è il rallentamento

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

Virtual Memory: LRU 1/7



Memory Manager: avvicendamento delle pagine in memoria.

Page Replacement Algorithms

Se viene portata in memoria una nuova pagina, bisogna scegliere una pagina da ***sfrattare*** (eliminare dalla memoria).

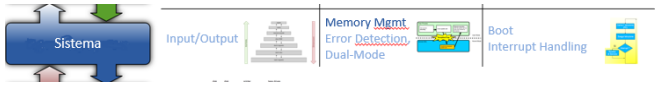
- È controproducente sfrattare le pagine molto usate (Recently Used) ➔ analizzare l'utilizzo delle pagine (Referenced bit).
- Se la pagina è stata scritta, bisogna copiarla sul disco, prima di rimuoverla ➔ analizzare la modifica (Write) delle pagine (Dirty bit). Altrimenti, una buona copia è sul disco (possibile scriverci sopra).
- **Principio di Località**: I riferimenti che un processo fa tendono ad essere vicini (sia dati, sia istruzioni).

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

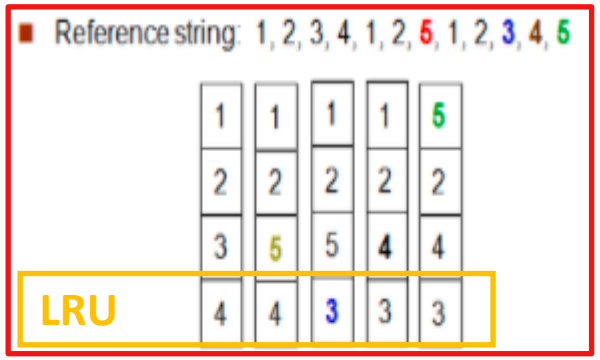
Virtual Memory: LRU 2/7

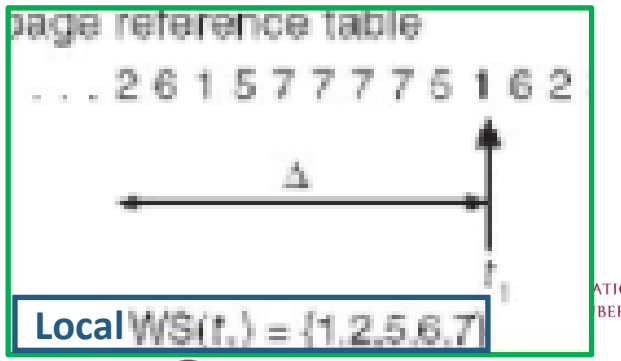


Memory Manager: scelta della pagina da rimuovere dalla memoria per far spazio a quella nuova.

Lista dei Page Replacement Algorithms

- **Optimal**: Scegliere quello che non sarà usato per maggior tempo. Impossibile: necessita la previsione delle referenziazioni future (*sfera di cristallo*). Algoritmo di riferimento ideale (Benchmark).
- **Recently Used**: analisi delle serie storiche per prevedere i futuri riferimenti
  - Not Recently Used (**NRU**): analisi dei bit **R** (Referenced) e **M** (Modified).
  - First-In, First-Out (**FIFO**): elimina la pagina caricata da più tempo. Facile da implementare ma pessime previsioni.
  - **Second Chance**: analogo al FIFO ma con analisi del bit **R** (Referenced). Efficiente,
  - **Clock**: disamina delle pagine «a giro», con analisi del bit **R** (Referenced). Efficiente.
  - Least Recently Used (**LRU**): eliminare la pagina usata meno di recente
- **Working Set**: insieme dei riferimenti caratteristici del processo (**Principio di Località**)
  - **WS**: analizza l'insieme dei riferimenti caratteristici
  - **WSClock**: disamina dei Working Set «a giro». Efficiente.

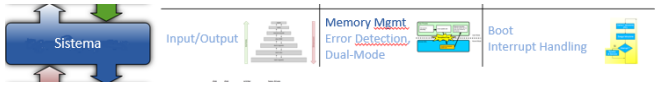




SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

Virtual Memory: LRU 3/7



Memory Manager: analisi delle serie storiche per identificare la pagina meno referenziata.

Not Recently Used (NRU)

analisi dei bit **R** (Referenced) e **M** (Modified).

Pulizia periodica del bit **R** (Referenced)

Sono definite 4 classi di priorità:

- **Class 0**: not referenced, not modified
- **Class 1**: not referenced, modified
- **Class 2**: referenced, not modified
- **Class 3**: referenced, modified

Sceglie una pagina a caso tra quelle con priorità minore.

Implementazione: moderatamente efficiente

Performance: moderatamente adeguata

First-In, First-Out (FIFO)

elimina la pagina caricata da più tempo.

Funzionamento:

- **Lista Ordinata**: mantenuta ordinate nel tempo (l'ultima pagina arrivata è in coda alla lista)
- **Eliminazione**: la pagina in testa viene sfrattata

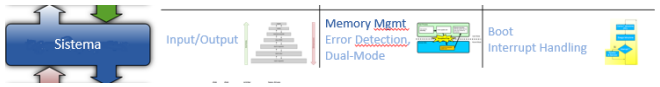
Non ha contezza del reale utilizzo delle pagine.

Implementazione: facile

Performance: disastrosa. Soffre della anomalia di Belady (frequenza dei page fault aumenta, nonostante l'incremento del numero di frame assegnati ai processi)

Operating Systems: Memory Management

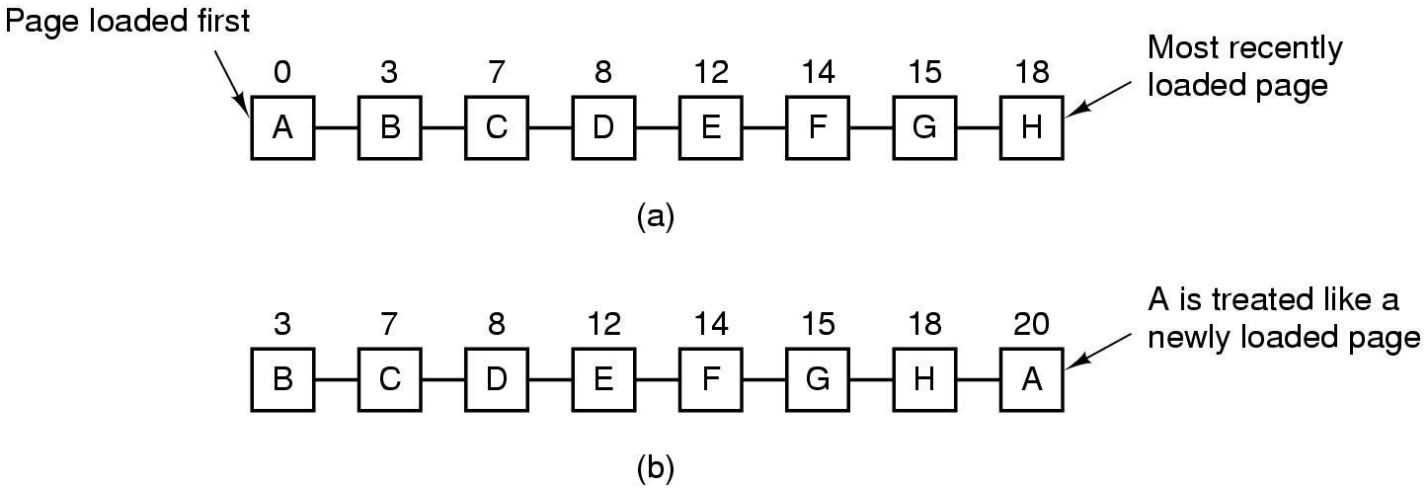
Virtual Memory: LRU 4/7



Memory Manager: analisi delle serie storiche per identificare la pagina meno referenziata.

Second Chance

Miglioramento del FIFO, con analisi dei bit **R** (Referenced) e **M** (Modified).



Funzionamento:

- **Lista Ordinata**: mantenuta ordinate nel tempo (l'ultima pagina arrivata è in coda alla lista): se il bit **R** (Referenced) è uno la pagina viene posta in coda ed il bit posto a zero
- **Eliminazione**: se il bit **R** (Referenced) è zero la pagina in testa viene sfrattata,

Se il WS è ampio (bit R resettato frequentemente) può portare ad eliminare una pagina utilizzata di recente.

Implementazione: facile

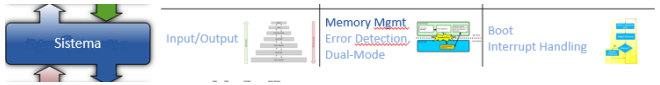
Performance: accettabile

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

Virtual Memory: LRU 5/7



Memory Manager: analisi delle serie storiche per identificare la pagina meno referenziata.

Clock

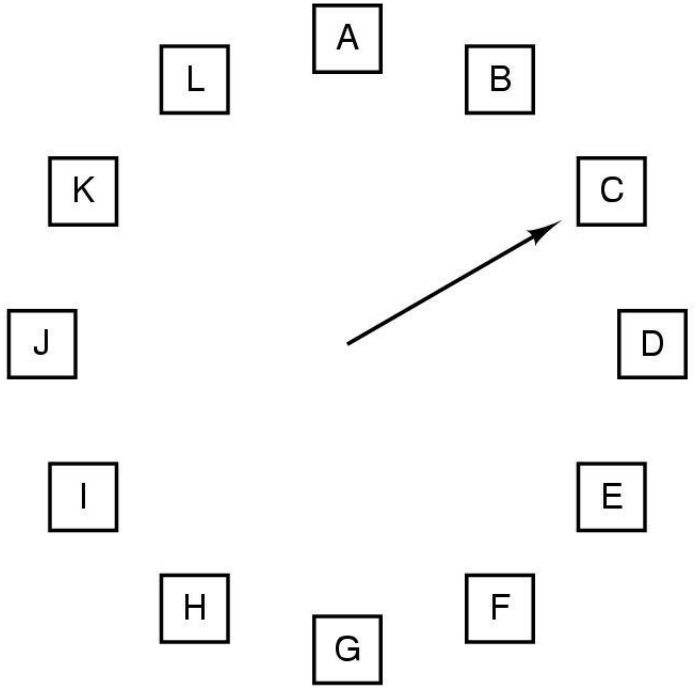
disamina delle pagine «a giro», con analisi del bit **R** (Referenced).

Funzionamento:

- **No Aging**: non viene computato il tempo da quanto la pagina è stata caricata
- **Lista Circolare**: scansione degli elementi della lista “a giro”
- **No Ordinamento**: più veloce perché non gestisce gli spostamenti di elementi nella lista
- **Eliminazione**: se il bit **R** (Referenced) è zero la pagina in testa viene sfrattata

Implementazione: facile

Performance: accettabile



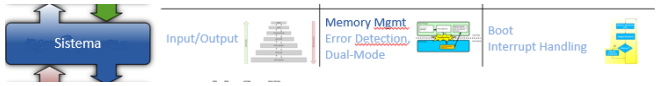
When a page fault occurs, the page the hand is pointing to is inspected. The action taken depends on the R bit:
R = 0: Evict the page
R = 1: Clear R and advance hand

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management

Virtual Memory: LRU 6/7



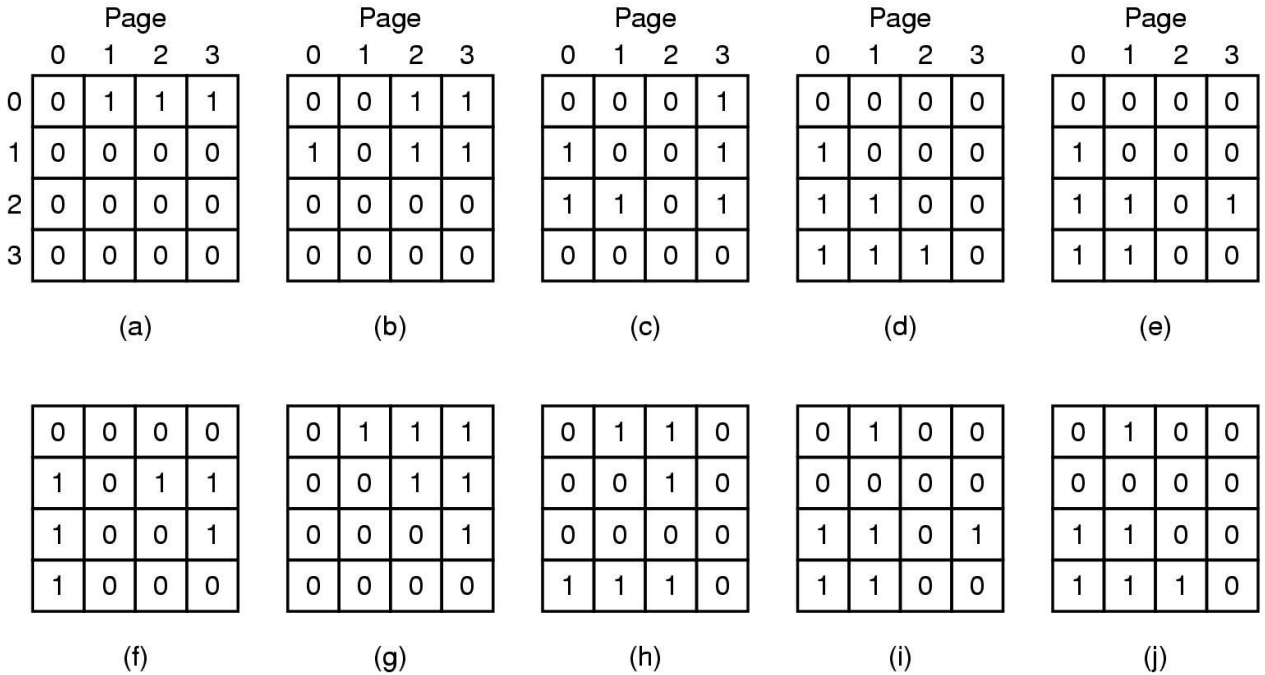
Memory Manager: analisi delle serie storiche per identificare la pagina meno referenziata.

Least Recently Used (LRU)

eliminare la pagina usata meno di recente.

Approssima LRU assumendo che l'uso recente della pagina approssimi l'uso della pagina a lungo termine

Potrebbe associare dei contatori ad ogni pagina ed esaminarli, ma questo è costoso



Funzionamento in Hardware:

- **Contatori**: associa un contatore per ogni pagina: matrice n x n per n pagine.
- **Incremento**: ad ogni referenza: quando si fa riferimento alla pagina k, mettere gli 1 nella riga k e gli 0 nella colonna k.
- **Eliminazione**: La riga con il valore binario più piccolo corrisponde alla pagina LRU. Elimina k

Implementazione: dispendiosa

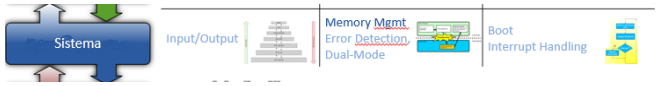
Performance: buona

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management

Virtual Memory: LRU 7/7



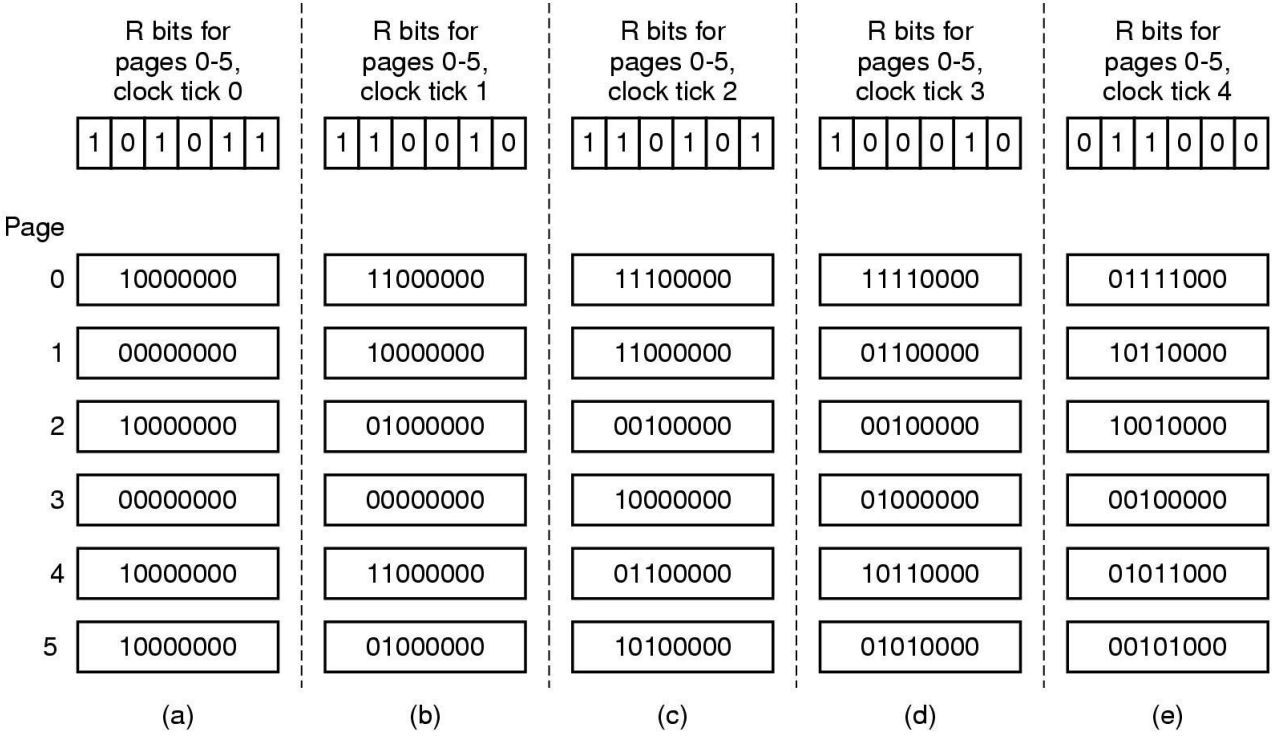
Memory Manager: analisi delle serie storiche per identificare la pagina meno referenziata.

Least Recently Used (LRU)

eliminare la pagina usata meno di recente, con analisi del bit **R** (Referenced).

Approssima LRU assumendo che l'uso recente della pagina approssimi l'uso della pagina a lungo termine

Potrebbe associare dei contatori ad ogni pagina ed esaminarli, ma questo è costoso



Funzionamento in Software (Aging):

- **Aging**: Mantenere una stringa di valori dei bit R per ogni tick di clock (fino a un certo limite)
- **Contatori**: fa uso di contatori software
- **Aggiornamento**: Dopo la spunta, spostare i bit a destra e aggiungere nuovi valori R a sinistra
- **Eliminazione**: elimina la pagina con il contatore più basso

Implementazione: facile

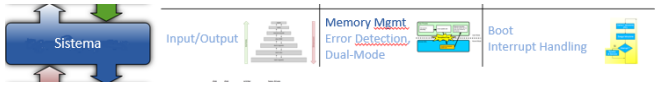
Performance: buona (importante la dimensione del contatore)

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management

Virtual Memory: Working Set 1/4

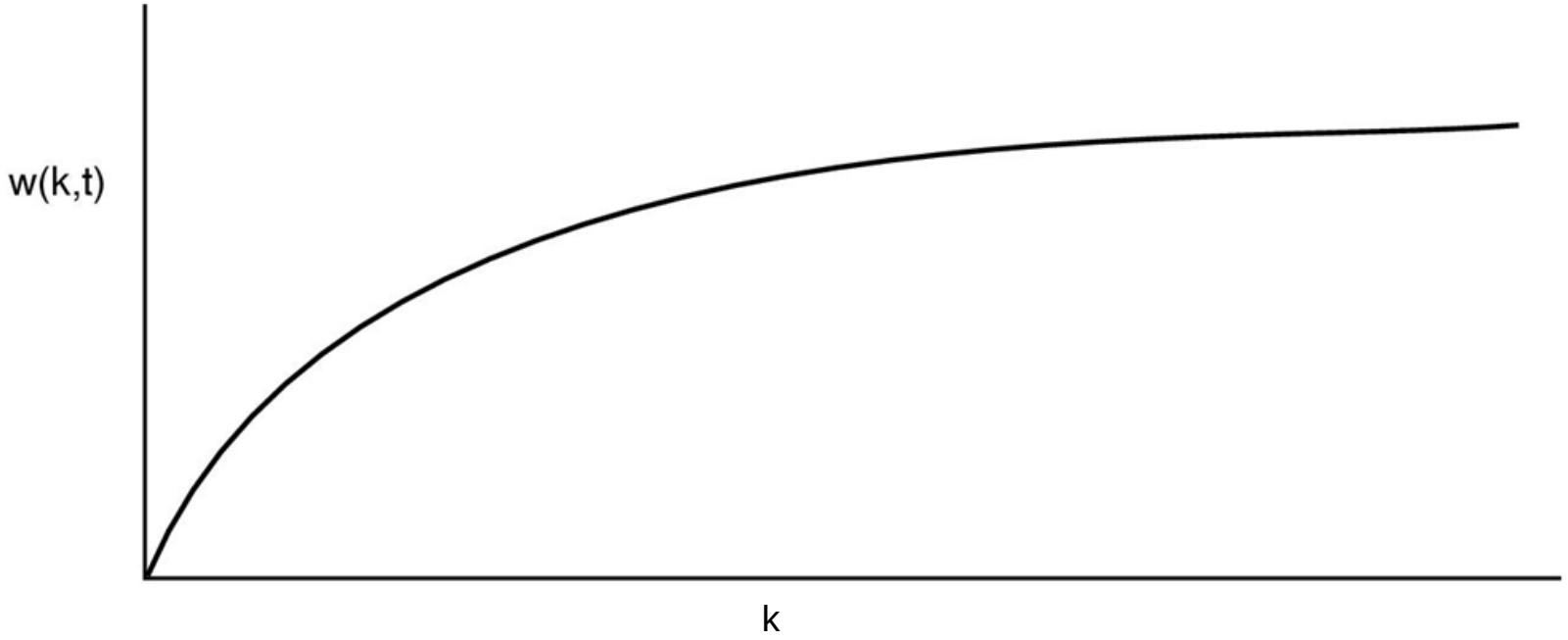


Memory Manager: analisi dell'insieme dei riferimenti caratteristici del processo per individuare il meno usato.

Working Set (WS)

Working Set: insieme delle pagine di cui un processo ha bisogno. In ogni istante t, considerando gli ultimi k riferimenti, è possibile calsolare la dimnesione del WS come w(k,t).

Rispetto al WS si possono avere 3 comportamenti:



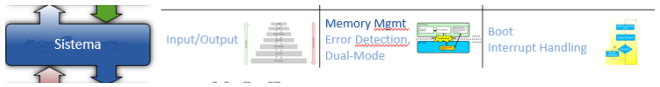
- **Demand Paging**: portare un processo in memoria cercando di eseguire la prima istruzione e ottenendo un page fault. Continuare fino a quando il WS non sia in memoria
- **Trashing**: la memoria è troppo piccola per contenere il WS, quindi il page fault è sempre presente
- **Pre-Paging**: Cercare di assicurarsi che il WS sia in memoria prima di far eseguire il processo

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

Virtual Memory: Working Set 2/4



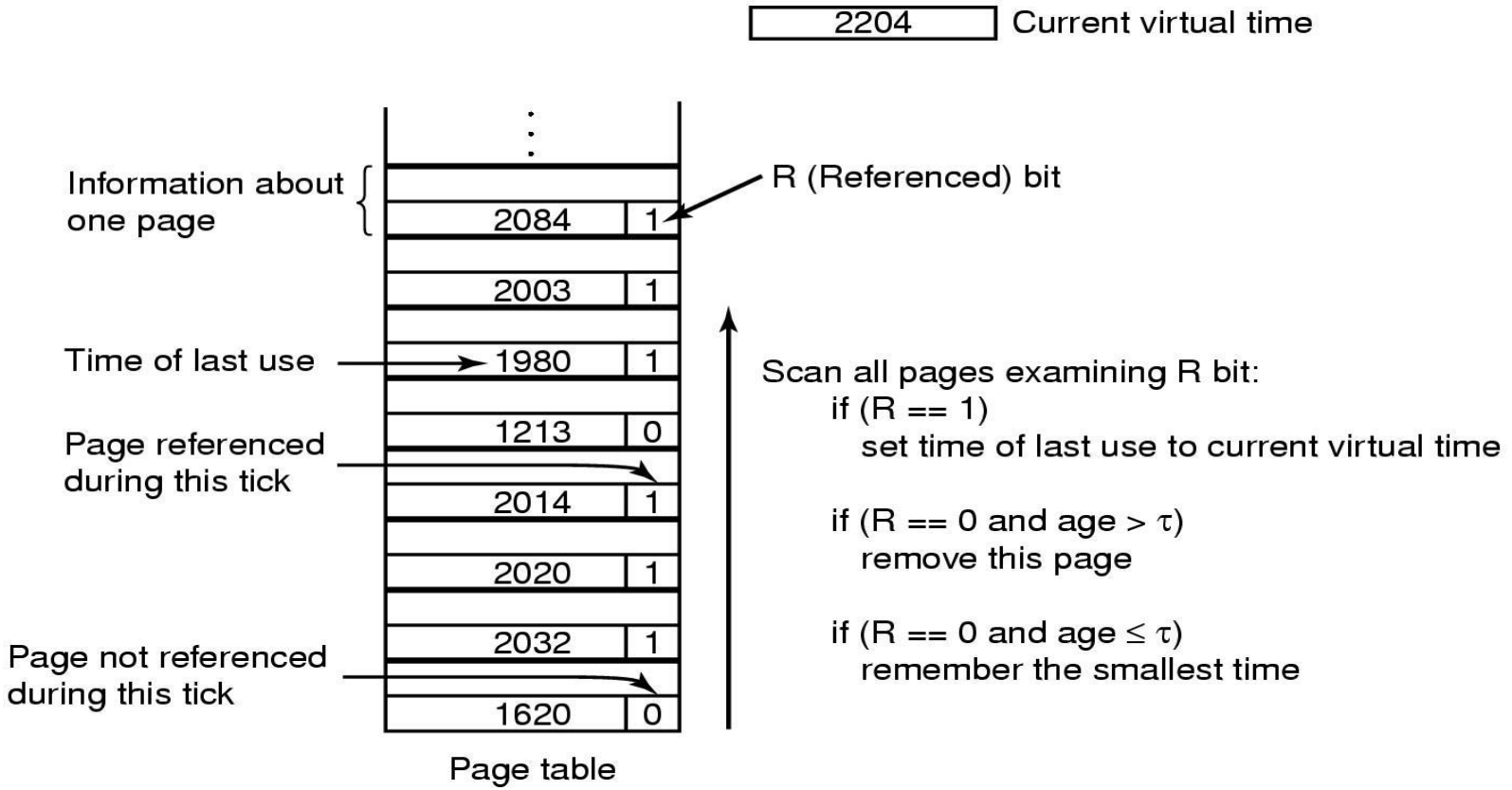
Memory Manager: analisi dell'insieme dei riferimenti caratteristici del processo per indiividuare il meno usato.

Funzionamento:

- **Virtual Time**: quantità di tempo della CPU utilizzato dall'inizio del processo
- **Time of Use**: tempo ultimo di riferimento alla pagina
- **Contatori**: traccia delle pagine in memoria ad ogni riferimento di memoria.
- **Aggiornamento**: tramite registro a scorrimento. Inserire il numero di pagina ad ogni riferimento.
- **Eliminazione**: elimina la pagina che non è nell'insieme di lavoro (se esiste una pagina siffatta). Altrimenti elimina quella on referenziata da più tempo.

Implementazione: dispendioso (scandire l'intera page table)

Performance: buona (importante la dimensione del contatore)

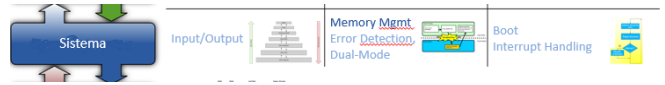


DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

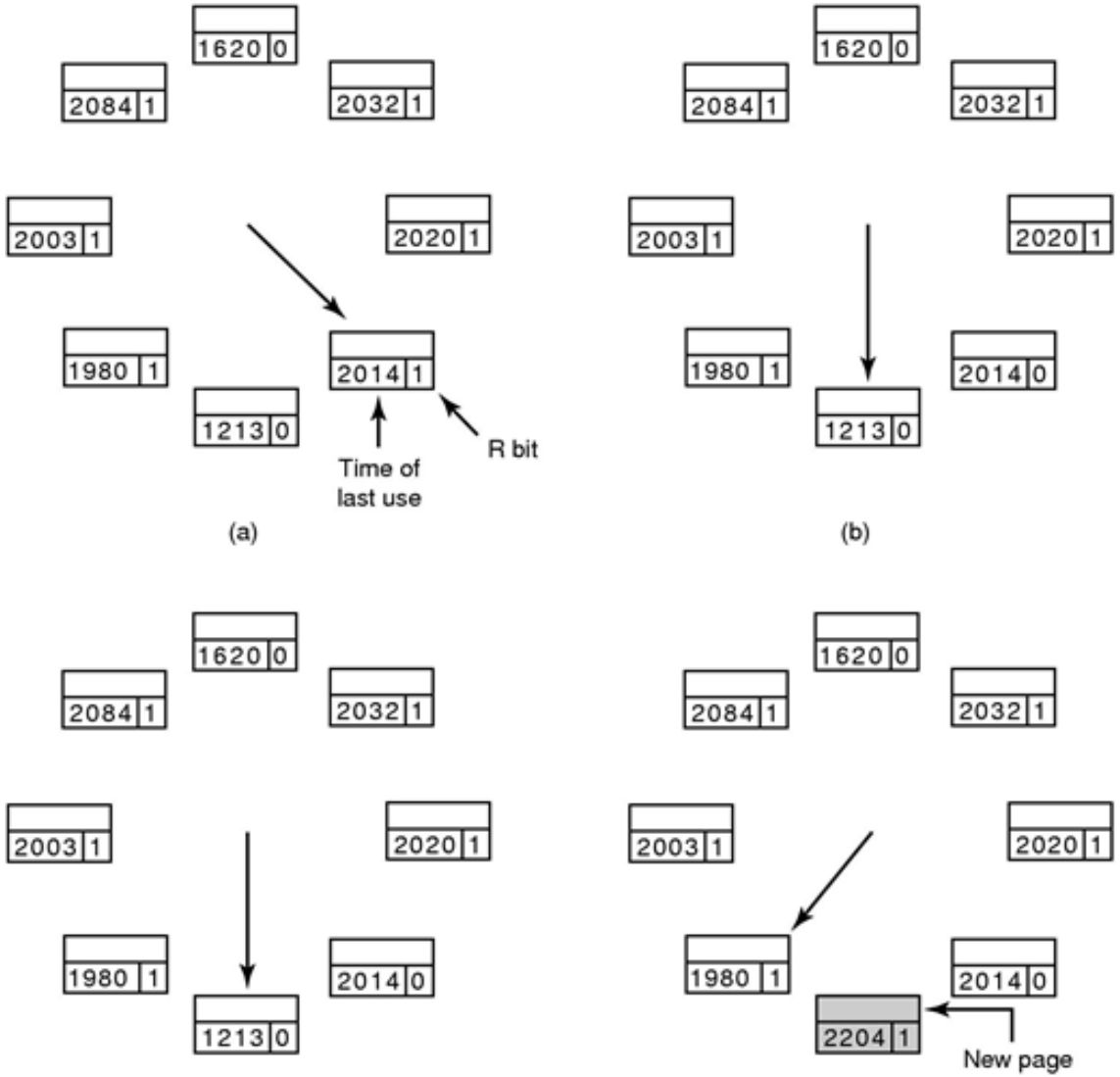
SAPIENZA UNIVERSITÀ DI ROMA

Operating Systems: Memory Management

Virtual Memory: Working Set 3/4



Memory Manager: analisi dell'insieme dei riferimenti caratteristici del processo per individuare il meno usato.



WSClock

disamina delle pagine «a giro», con analisi del bit **R** (Referenced) e del «Time of Last Use».

Funzionamento:

- **Time of Use**: viene computato il tempo ultimo di utilizzo della pagina
- **Lista Circolare**: scansione degli elementi della lista "a giro"
- **No Ordinamento**: più veloce perché non gestisce gli spostamenti di elementi nella lista
- **Eliminazione**: se il bit **R** (Referenced) è zero e il time of use è minore del valore stabilito, la pagina viene sfrattata. Altrimenti, si pone R a zero esi procede oltre.

Implementazione: facile

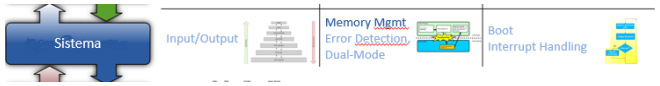
Performance: buona. Molto usato

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Operating Systems: Memory Management

Virtual Memory: Working Set 4/4



Memory Manager: riepilogo dei Page Replacement Algorithm.

Algorithm	Comment
Optimal	Not implementable, but useful as a benchmark
NRU (Not Recently Used)	Very crude approximation of LRU
FIFO (First-In, First-Out)	Might throw out important pages
Second chance	Big improvement over FIFO
Clock	Realistic
LRU (Least Recently Used)	Excellent, but difficult to implement exactly
NFU (Not Frequently Used)	Fairly crude approximation to LRU
Aging	Efficient algorithm that approximates LRU well
Working set	Somewhat expensive to implement
WSClock	Good efficient algorithm

Stack Algorithm

Algoritmi di Page Replacement per cui vale la disuguaglianza:

$M(m, r) \subseteq M(m+1, r)$

dove:

- **M**: insieme delle pagine del processo in memoria
- **m**: numero di pagine del processo in memoria
- **r**: indice scelta solo all'interno del processo con Page Fault

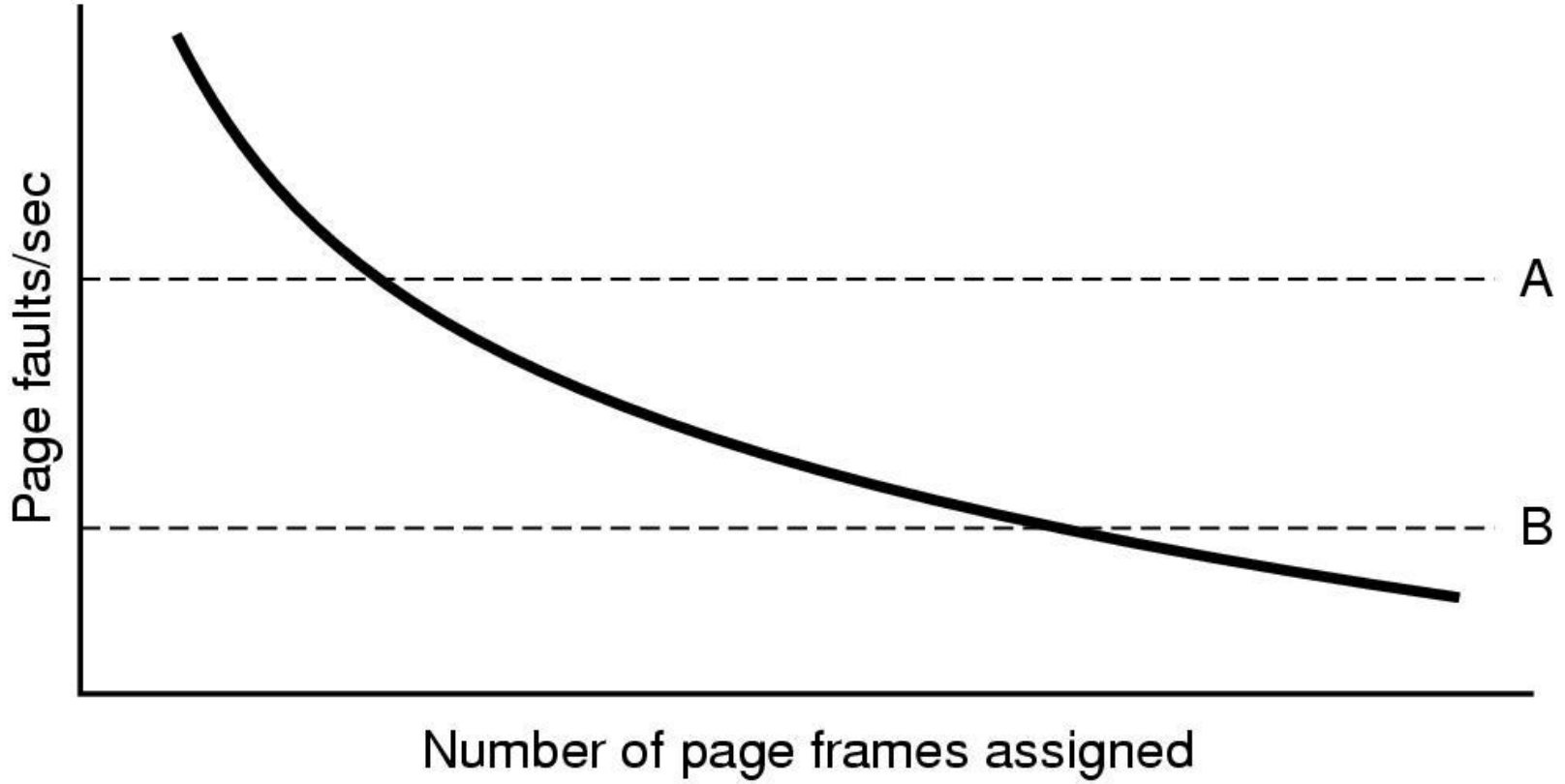
Operating Systems: Memory Management

Virtual Memory: Page Fault Frequency (PFF)

Memory Manager: considerare la frequenza di Page Fault.

Implementazione

- **Global**: scelta fra tutti i processi. Scelta migliore per la memoria:
  - WS crescono/si restringono nel tempo: i processi hanno dimensioni diverse
  - Assegnare un numero di pagine ad ogni processo proporzionale alla sua dimensione: Iniziare con un'allocazione basata sulla dimensione e usare la frequenza di errore delle pagine (**PFF**) per modificare la dimensione dell'allocazione per ogni processo
- **Local**: scelta solo all'interno del processo con Page Fault



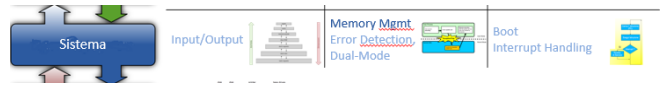
- **Lock**: bloccare lo swap delle pagine coinvolte in operazioni di I/O

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

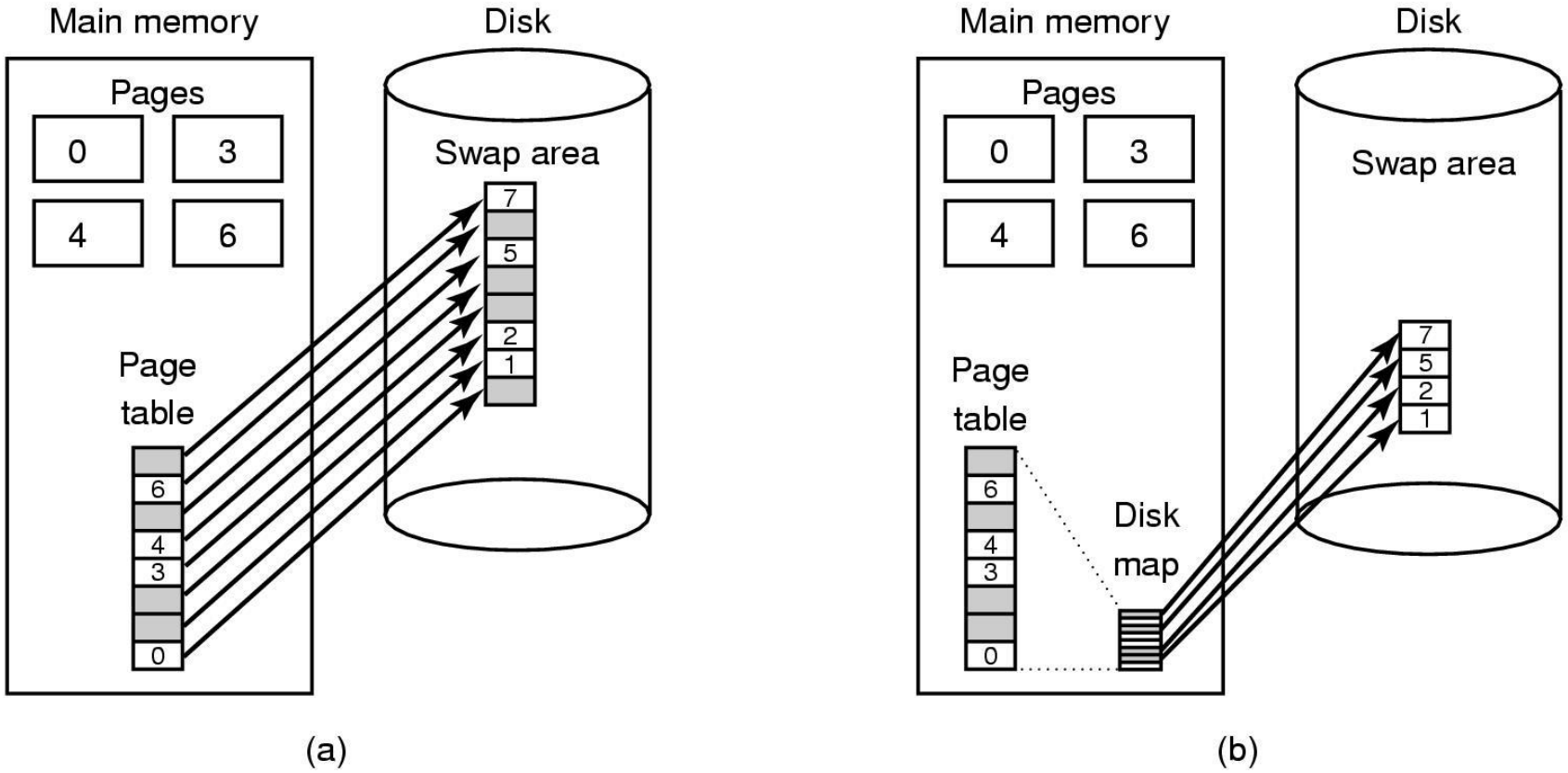
Sapienza Università di Roma

Operating Systems: Memory Management

Virtual Memory: Backing Store 1/2



Memory Manager: gestione della movimentazione delle pagine da e verso il disco.



(a) **Dynamic**: alloca le pagine secndo necessità, a *run-time*. Non assegna per il processo in anticipo lo spazio su disco. Necessita di un **Disk Map**

Backing Store

Gestione dell'area di swap su disco:

- Partizione separata senza file system
- File senza struttura

Allocazione spazio di swap:

(a) **Static**: Assegna una partizione fissa (chunk) abbastanza grande al processo all'avvio.

Lista di chunk liberi.

Indirizzo virtuale su disco:

- Indirizzo iniziale della partizione.
- offset di pagina.

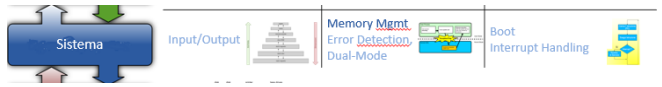
Aree diverse per:

- Dati (cresce)
- Testo (non cresce)
- Stack (cresce)

Dipartimento di Ingegneria Informatica Automatica e Gestionale Antonio Ruberti
Sapienza Università di Roma

Operating Systems: Memory Management

Virtual Memory: Backing Store 2/2



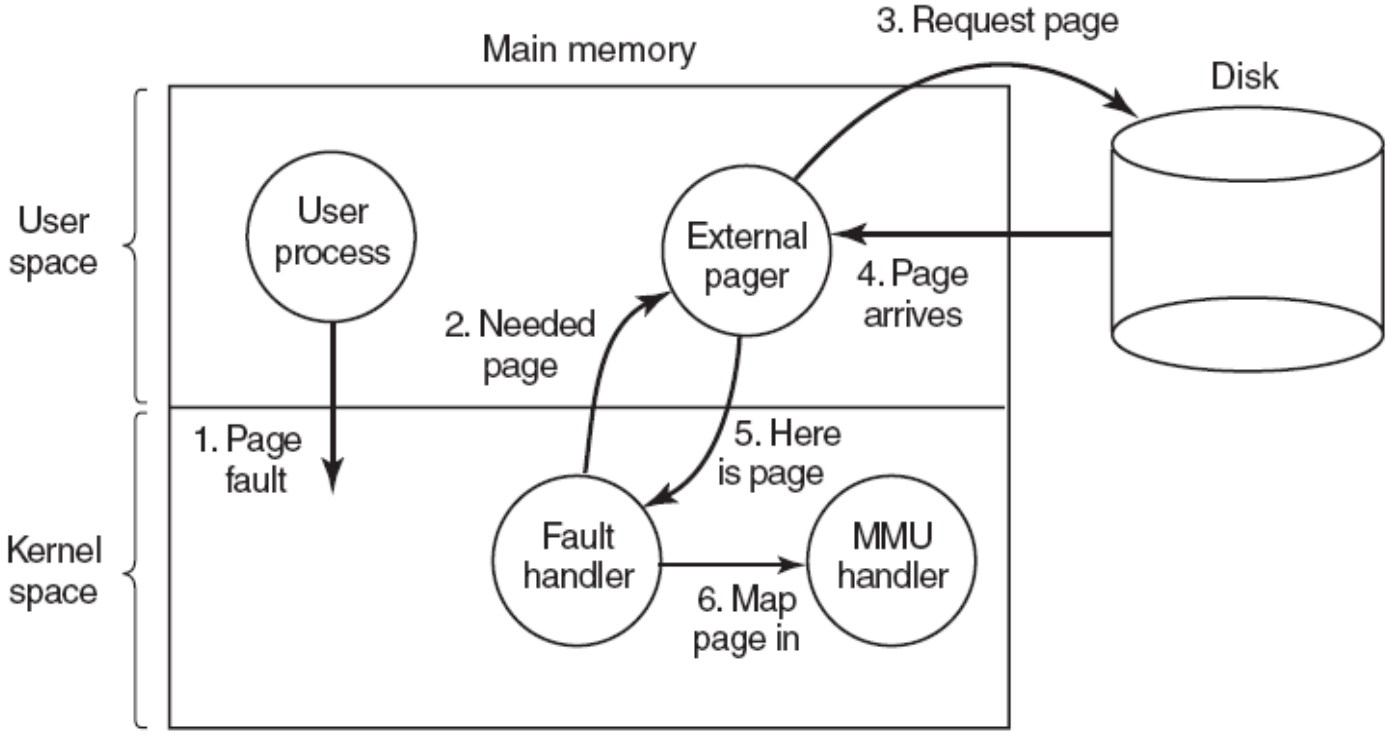
Memory Manager: gestione della movimentazione delle pagine da e verso il disco.

Separazione di Policy (User) e Mechanism (Kernel)
Separazione delle operazioni da eseguire per i page fault in 2 livelli:

User: contiene la **policy** per la sostituzione delle pagine e chiede/riceve pagine dal disco Lista di chunk liberi.

Kernel: composto da:

- gestore MMU di basso livello (dipendente dalla macchina)
- gestore di errori di pagina che è parte del kernel (indipendente dalla macchina) Chiede alla MMU di assegnare lo spazio per la pagina in arrivo nel processo



Caratteristiche:

- Cambiamento di Modo: più volte durante un page fault
- Modulare: grande flessibilità

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma