Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative
Istituto sistemi informativi e networking

SUPSI

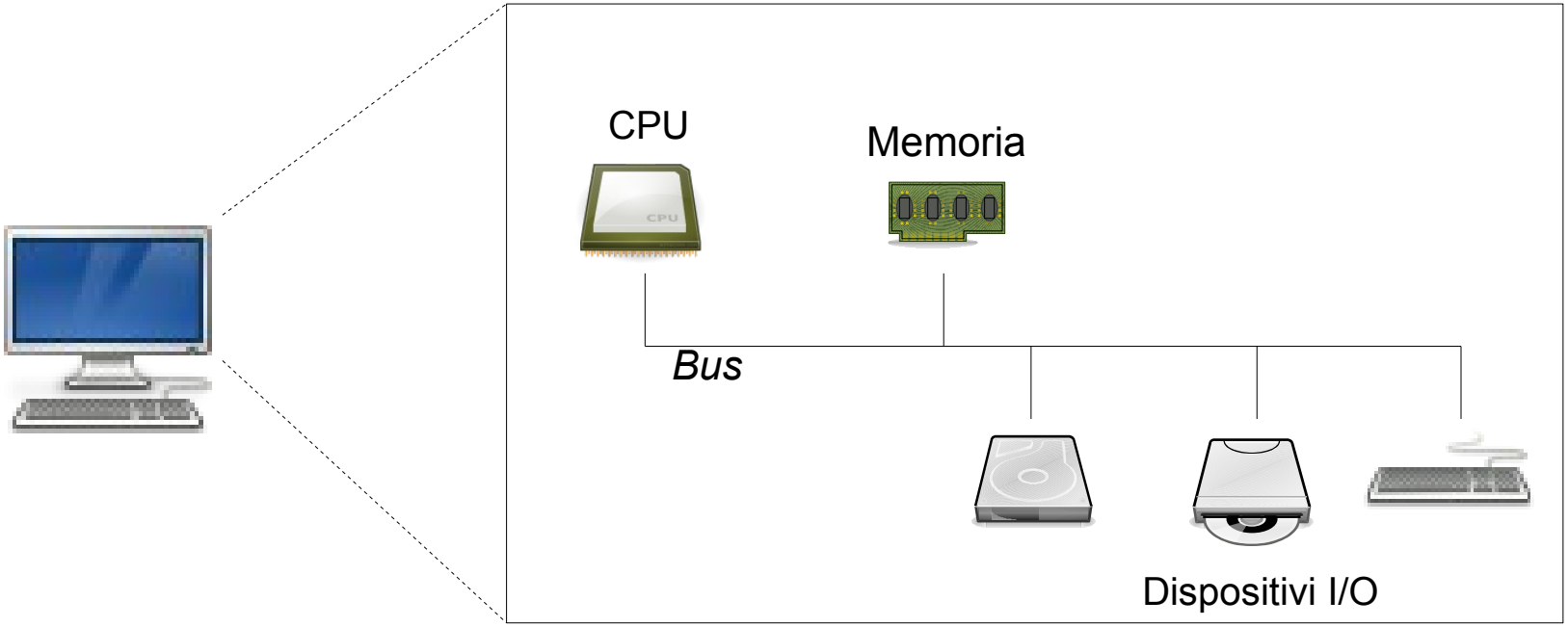
Ambienti Operativi: Memoria

Amos Brocco, Ricercatore, ISIN / DTI

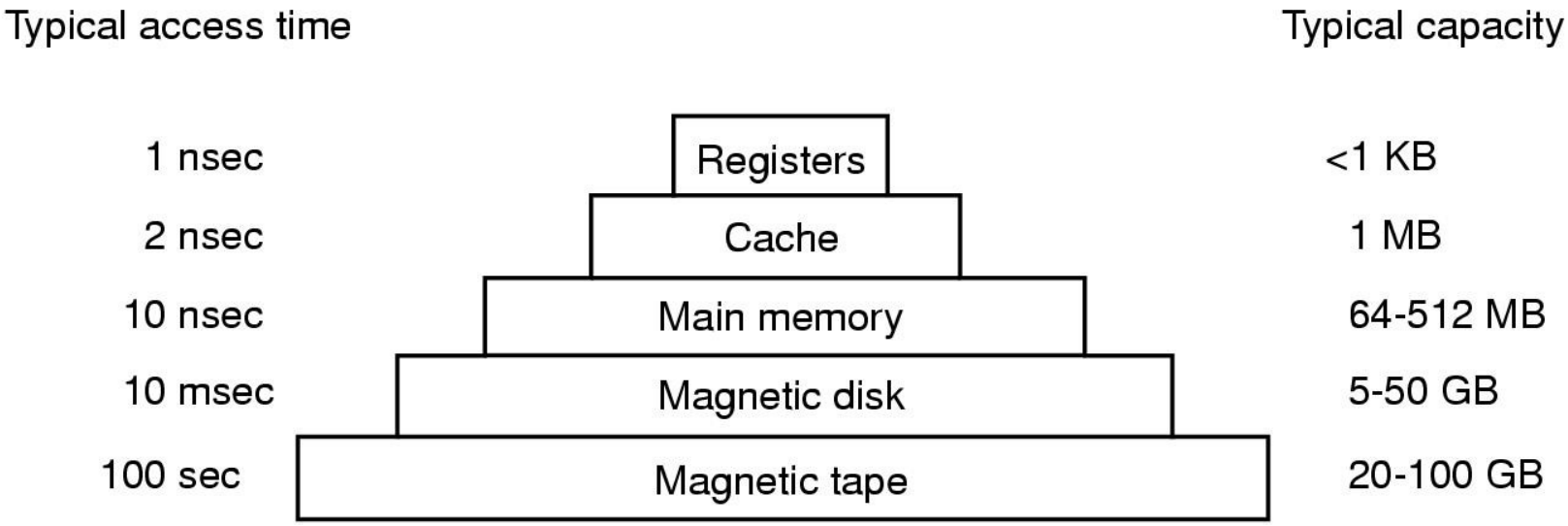
Motivazione

- Perché studiare come la memoria viene gestita in un sistema operativo moderno?
  - per capire come utilizzarla meglio
    - quali sono i limiti dell'hardware e del software
  - per capire cosa significano termini come swapping, memoria virtuale, segmentazione, segmentation fault, paginazione,...

Architettura di un computer

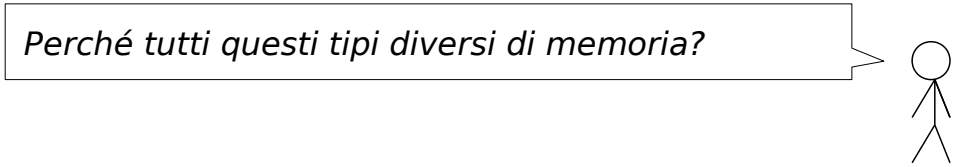


Gerarchia della memoria



Da A. Tanenbaum, "Modern Operating Systems", 2ª Edizione

Gerarchia di memoria



- Idealmente i programmatori e gli utenti vorrebbero disporre di memoria che è allo stesso tempo
  - Infinita
  - Veloce
  - A basso costo
  - Non volatile
- *Purtroppo questo obiettivo non è ancora stato raggiunto*
- È compito del sistema operativo gestire efficientemente queste memorie e offrire un'**astrazione** che "nasconda" i limiti dell'hardware

All'inizio... nessuna astrazione

- I primi computer (pre-1960) non offrono nessun tipo di astrazione della memoria
- Ogni programma ha **accesso diretto alla memoria fisica**
  - gli indirizzi da utilizzare sono codificati nel programma, per es.

```
MOV registro1, 1981
```

sposta il contenuto della posizione di memoria 1981 nel registro numero 1

→ difficile poter eseguire più di un processo contemporaneamente:

- processi diversi devono essere scritti in modo da utilizzare indirizzi differenti

Perché astrarre

- Per **ovviare alle limitazioni** fisiche (es. dimensione finita della memoria)
- Per **semplificare la multi-programmazione**
  - è praticamente impossibile garantire che ogni programma utilizzi effettivamente indirizzi diversi
- Per **proteggere** i dati
  - se ogni processo ha accesso alla memoria fisica può sovrascrivere i dati di altri processi (o del sistema operativo!)

Semplificare e proteggere: spazi di indirizzamento

- Invece di dare libero accesso a tutta memoria, ogni processo riceve uno **spazio di indirizzamento**:
  - lo spazio di indirizzamento definisce l'insieme di indirizzi che il processo può utilizzare per accedere alla memoria

- Uno spazio di indirizzamento è definito da
  - un indirizzo **base** nella memoria fisica
  - una dimensione **limite** (quanto è grande lo spazio)

Semplificare e proteggere: spazi di indirizzamento

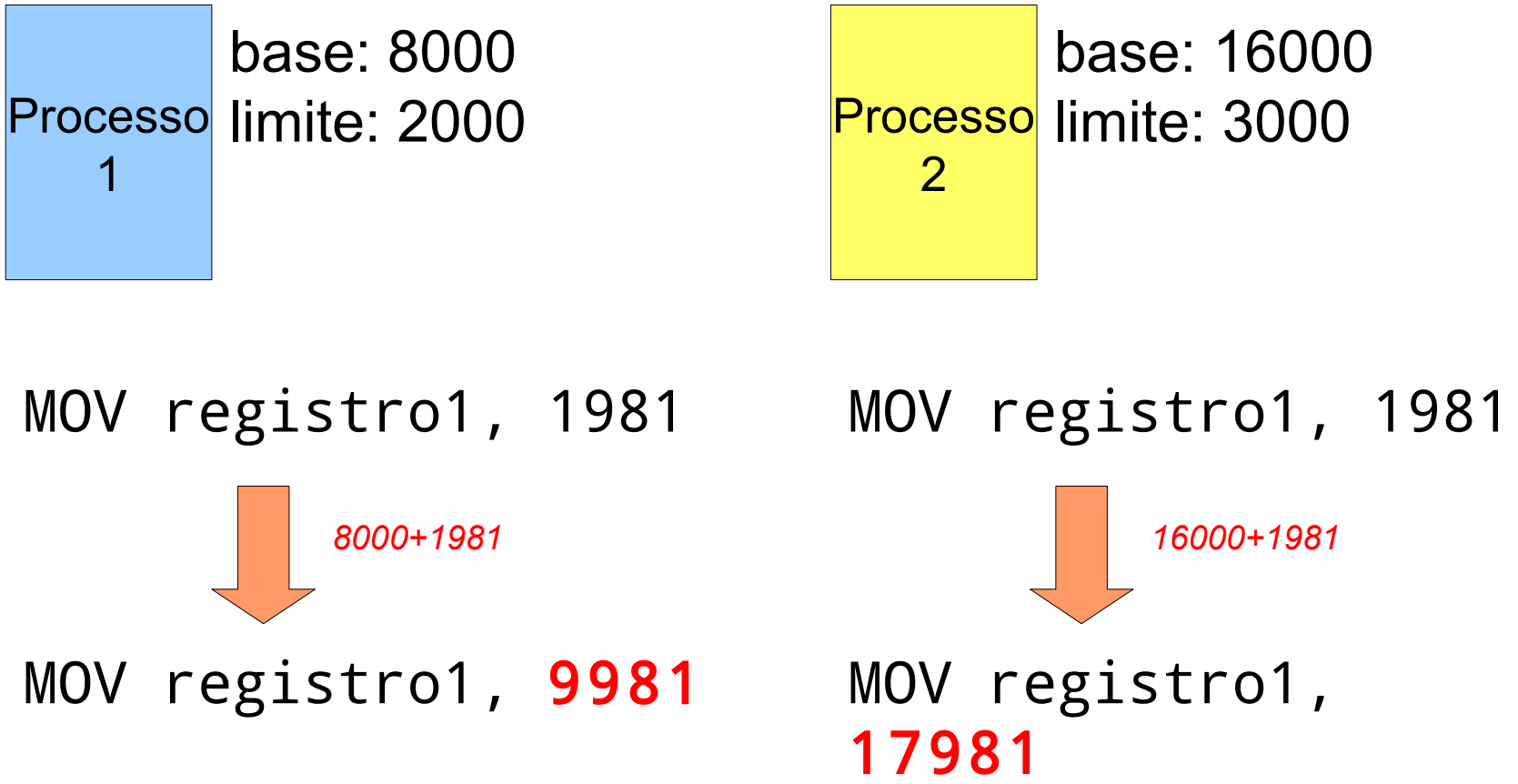
- Quando un processo viene eseguito, base e limite vengono caricati in due registri del processore

- Gli indirizzi codificati in un programma non vengono più considerati come indirizzi fisici ma come **offset** relativi alla base dello spazio di indirizzamento assegnato al processo
  - per ottenere un indirizzo nella memoria fisica **l'offset viene sommato alla base**

- Il sistema controlla anche che l'offset utilizzato non sia maggiore del limite, evitando che un processo possa curiosare o modificare altri spazi di indirizzamento → **protezione della memoria**

Semplificare la multiprogrammazione: spazi di indirizzamento

- Esempio di traduzione:



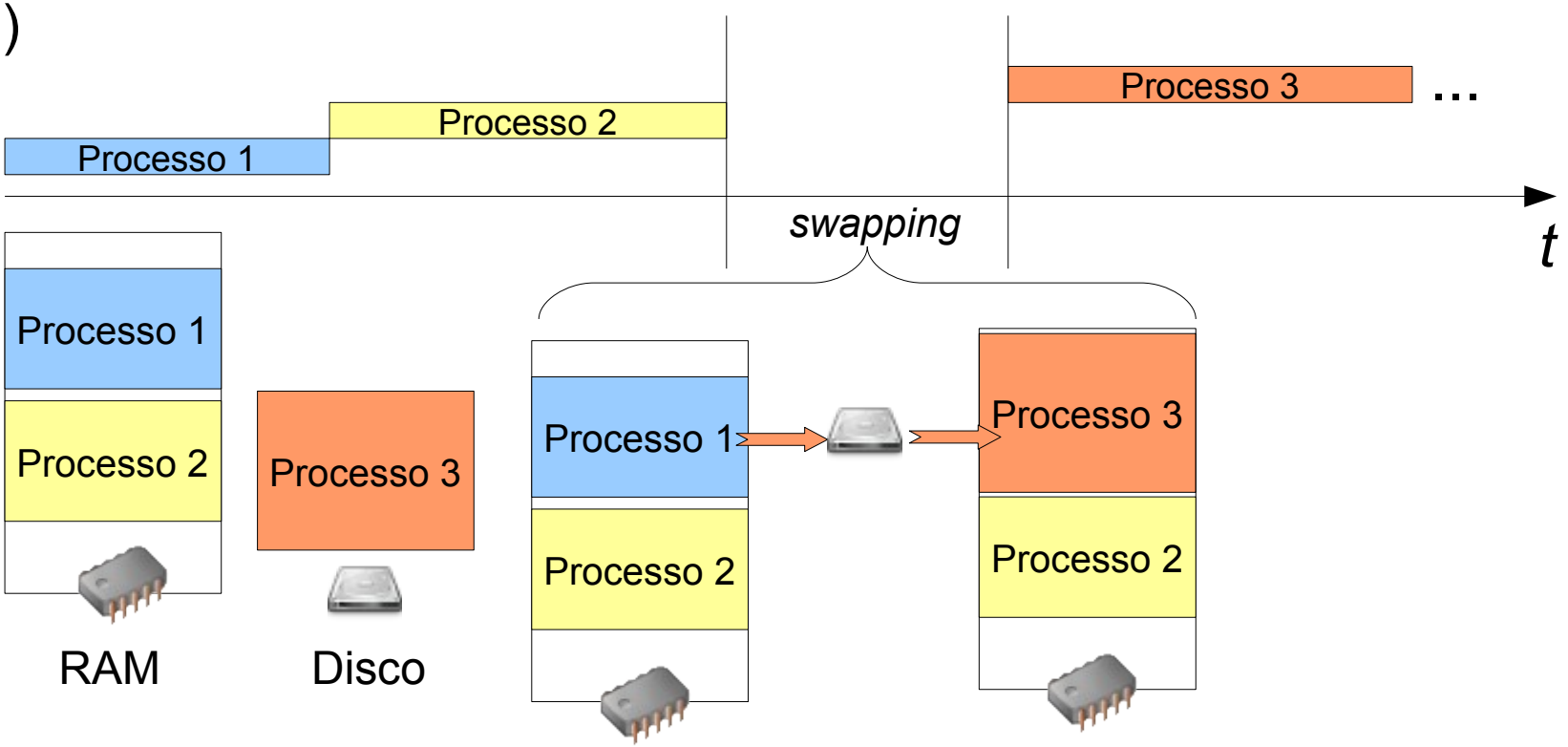
Ovviare alle limitazioni: Swapping e memoria virtuale

- Spesso la memoria RAM richiesta dai processi è maggiore di quella libera disponibile

- Per risolvere questo problema ci sono due soluzioni
  - **swapping**
  - **memoria virtuale**

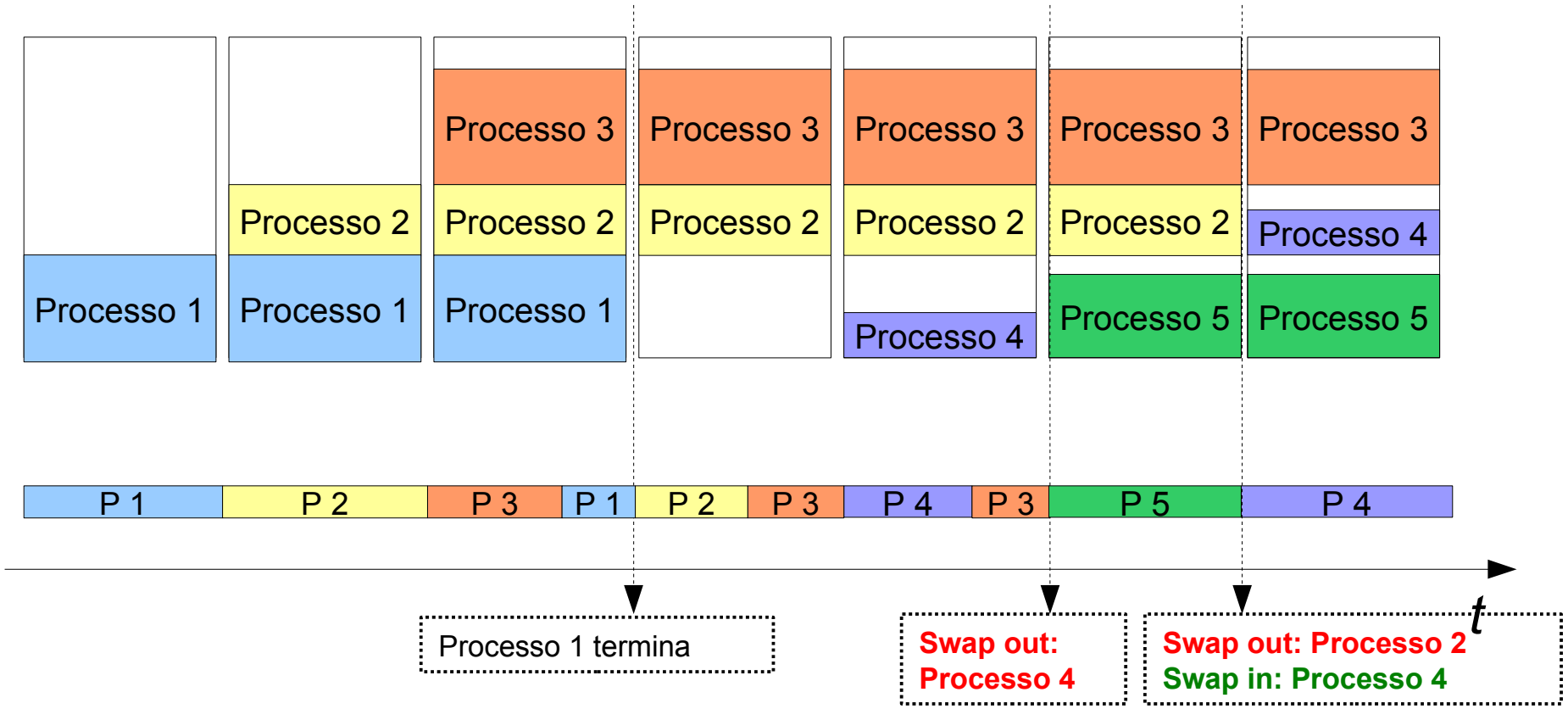
Swapping

- Lo **swapping** (*scambio*) consiste nello spostare la memoria occupata dai processi che non sono correntemente in esecuzione su disco (*swap out*) per poi ricaricarla quando necessario (*swap in*)



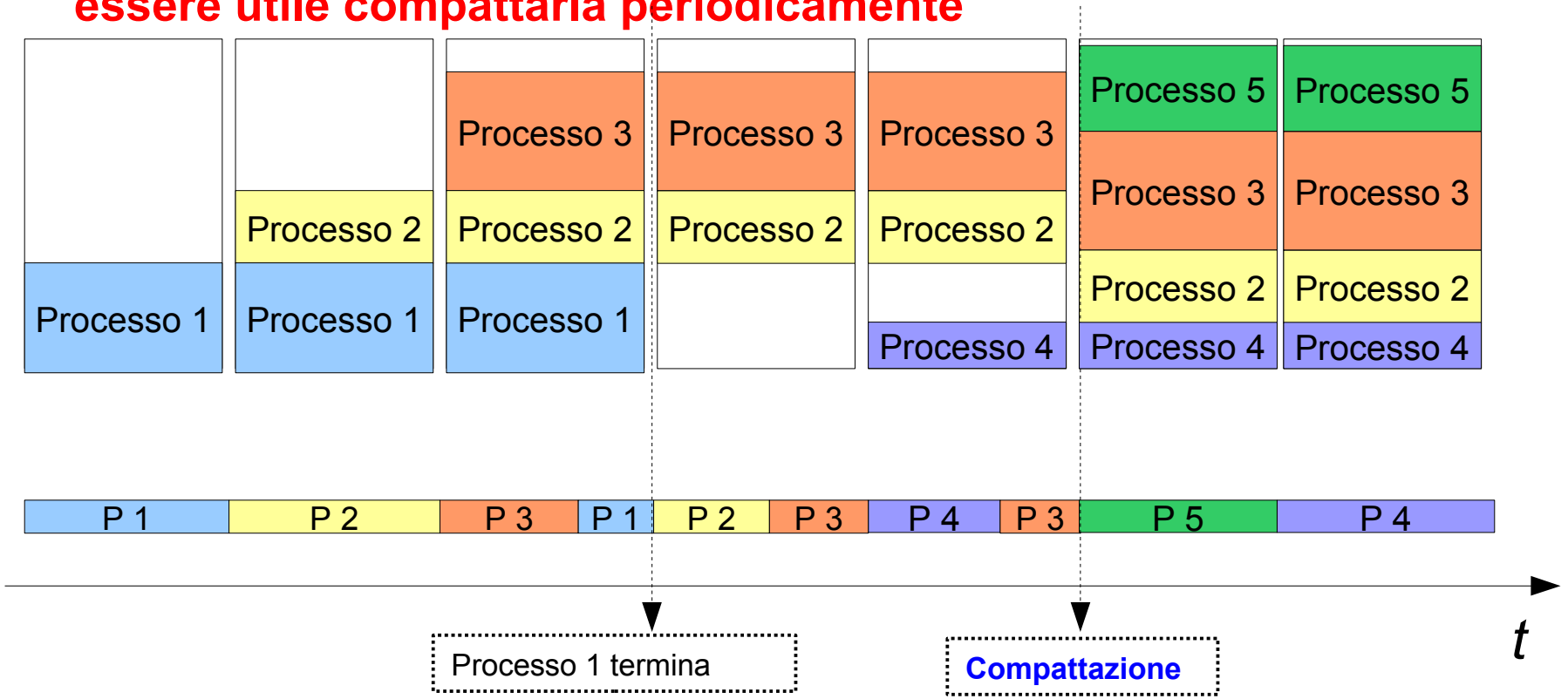
Problemi con lo swapping: frammentazione esterna

- Con lo swapping la memoria può diventare frammentata, causando swap non necessari che rallentano l'esecuzione dei processi



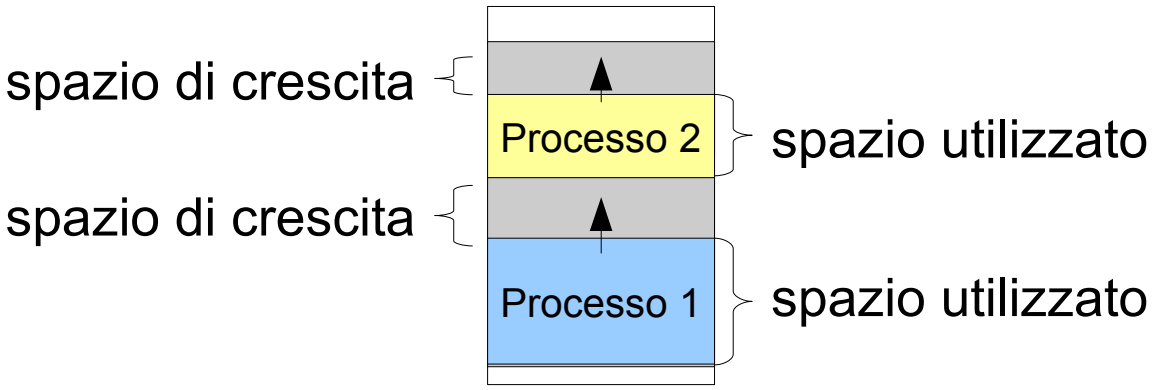
Frammentazione esterna: compattazione

- Con lo swapping la memoria può diventare frammentata, causando swap non necessari che rallentano l'esecuzione dei processi **→ può essere utile compattarla periodicamente**



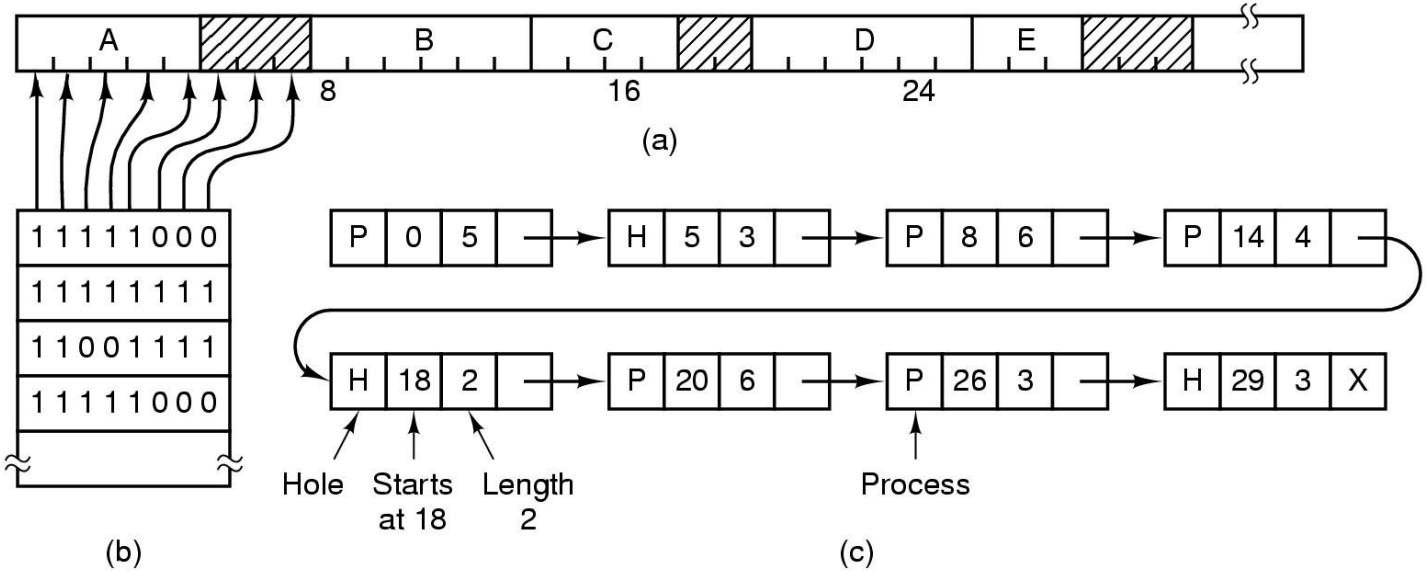
Processi che richiedono più memoria

- Un programma in esecuzione può richiedere memoria dinamicamente...
  - il sistema operativo deve "prevedere" questa situazione e lasciare della memoria disponibile tra quella allocata per evitare swap inutili



Gestione dello spazio libero

- Per sapere dove trovare spazio libero in memoria il sistema operativo deve tener traccia della memoria allocata:
  - **bitmap** (0 = spazio non allocato, 1 = spazio allocato) *(b)*
  - **lista concatenata** *(c)*



da A. Tanenbaum, Modern Operating Systems, 2nd Ed.

Sempre più memoria...

- Sui moderni computer i programmi spesso richiedono più della memoria effettivamente disponibile...
  - astrazioni
  - sistemi multi-utente, sempre più applicazioni
  - audio / video processing (grandi quantità di dati)
- **Come fare?**
  - tipicamente un programma non utilizza tutta la memoria che ha richiesto nello stesso momento, quindi è possibile dividerla a pezzi che il sistema può gestire indipendentemente con lo swapping
  - visto che dividere grandi programmi in pezzi più piccoli è difficile il compito è lasciato al **sistema operativo, che gestisce un'astrazione detta memoria virtuale**

Memoria virtuale: Paginazione

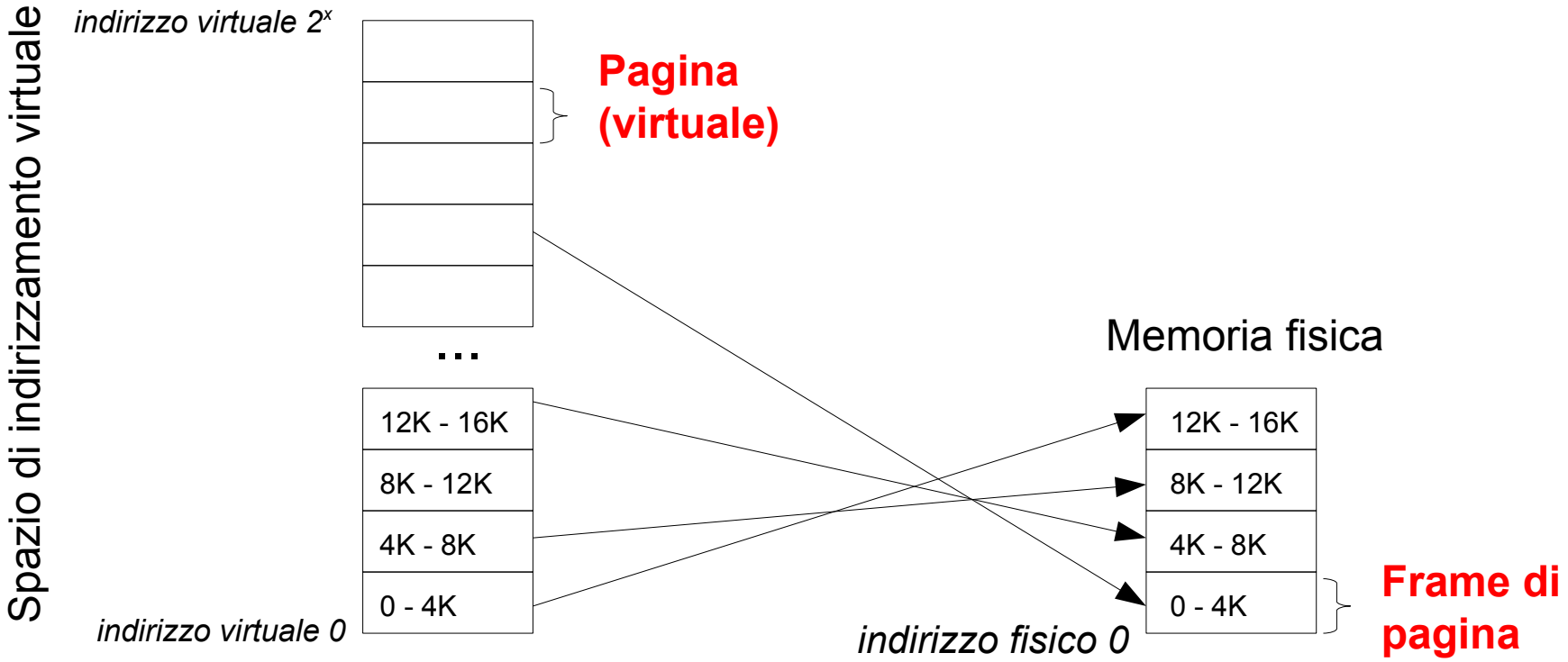
- **Memoria virtuale**: ogni processo ha il suo spazio di indirizzamento, che è diviso in pagine di ugual dimensione (p.es. 4 KB)

- Ogni pagina contiene un intervallo contiguo di indirizzi
  - le pagine sono caricate nella memoria fisica in unità di ugual dimensione dette **frame di pagina**

- Il numero di pagine utilizzate da un processo dipende dalla memoria allocata e dalla dimensione di ogni pagina
- Il numero di frame di pagina dipende dalla quantità di memoria fisica disponibile e dalla dimensione di ogni pagina

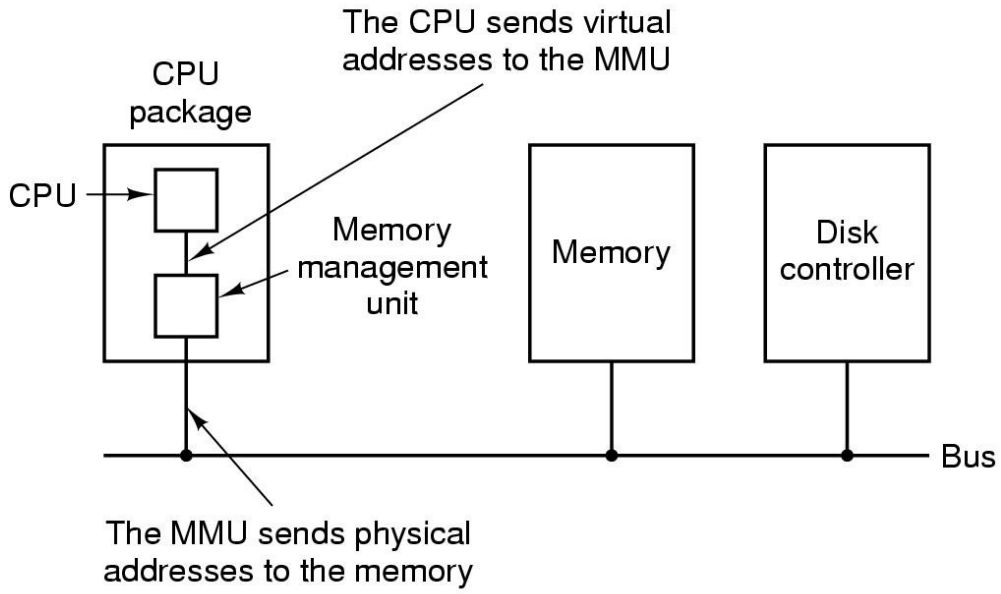
Paginazione

- I processi non utilizzano indirizzi nella memoria fisica, ma in uno **spazio di indirizzamento virtuale:** prima accedere alla memoria, l'indirizzo virtuale deve essere "tradotto" in un indirizzo fisico



Traduzione degli indirizzi

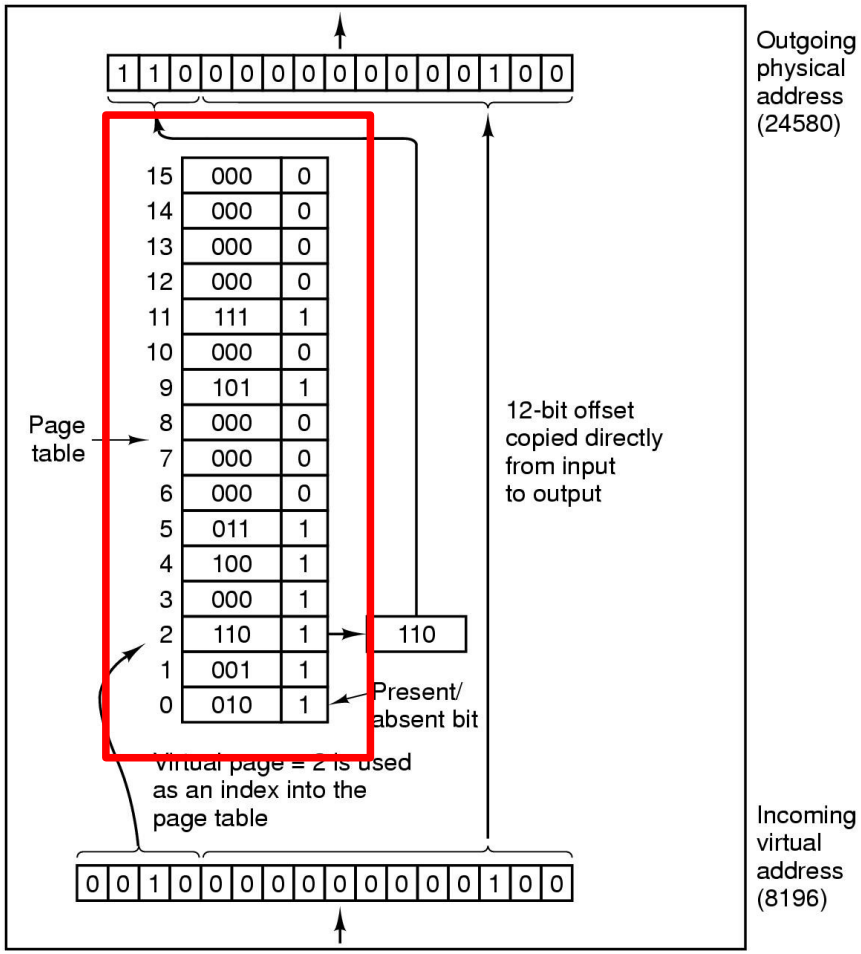
- Quando un processo cerca di accedere ad un indirizzo, una componente hardware chiamata **MMU** (Memory Management Unit) traduce l'indirizzo virtuale in un indirizzo fisico



da A. Tanenbaum, Modern Operating Systems, 2nd Ed.

Traduzione degli indirizzi: tabella delle pagine

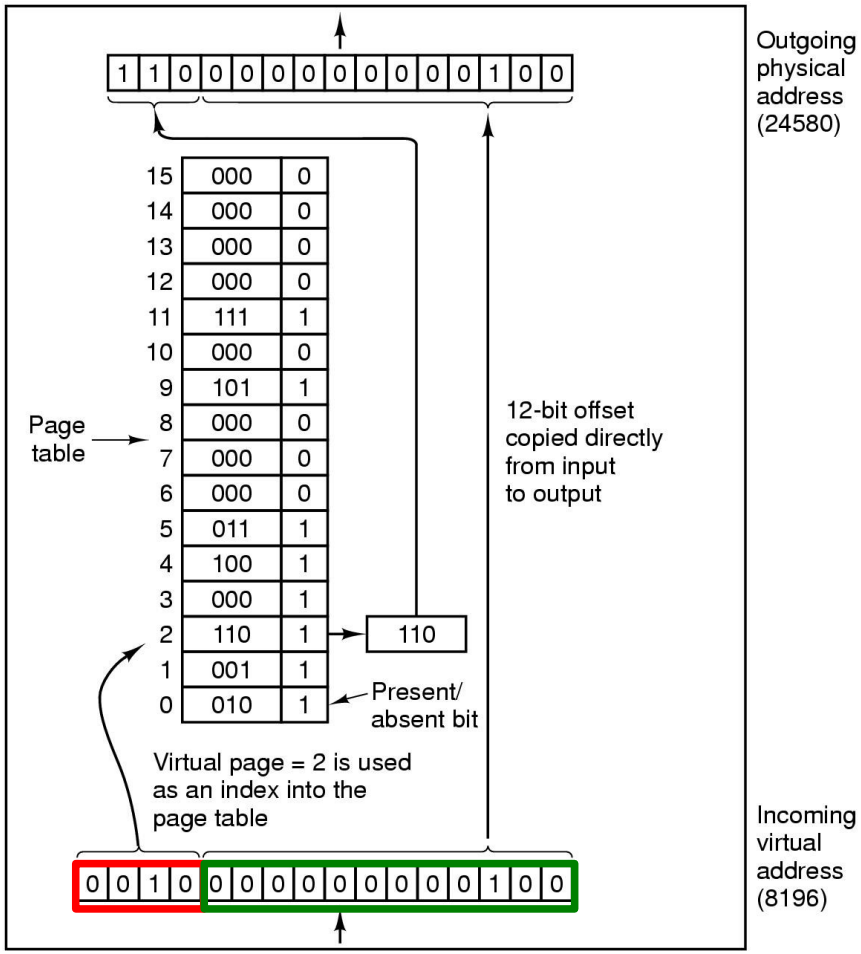
- La MMU gestisce una tabella delle pagine (che risiede in memoria!) con il **mapping tra il numero della pagina e l'indirizzo base del frame in memoria**



da A. Tanenbaum, Modern Operating Systems, 2nd Ed.

Traduzione degli indirizzi: tabella delle pagine

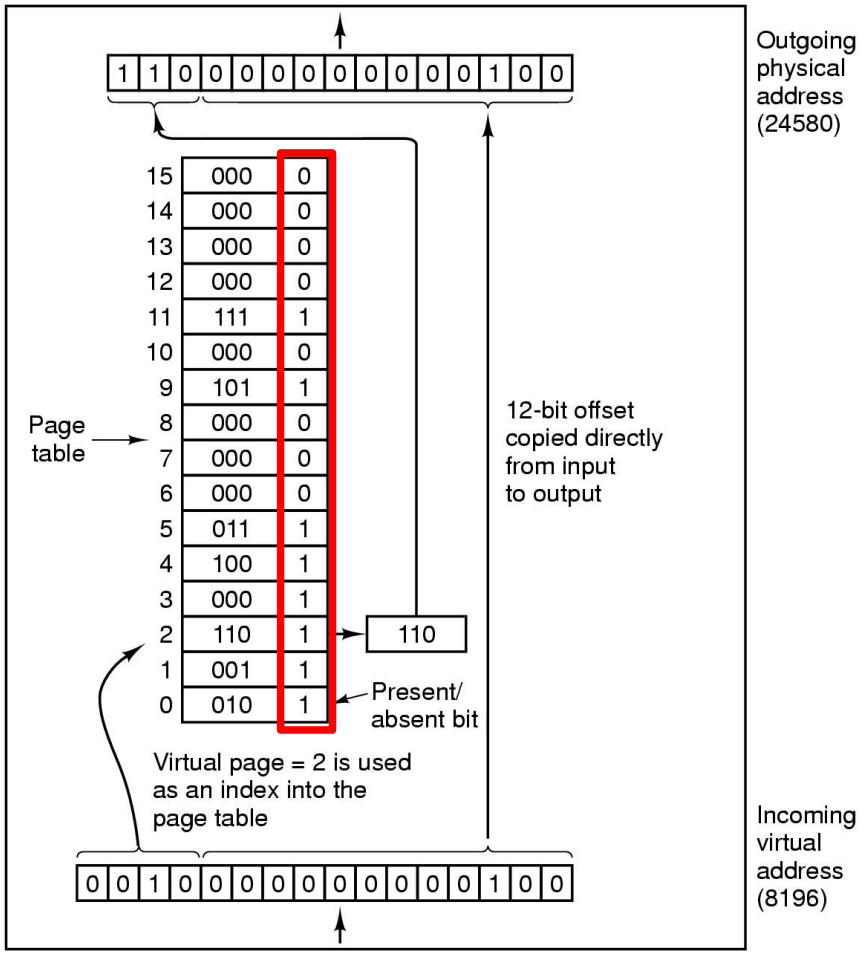
- Una parte dell'indirizzo virtuale identifica il **numero di pagina**, mentre il resto è un ***offset* all'interno della pagina**



da A. Tanenbaum, Modern Operating Systems, 2nd Ed.

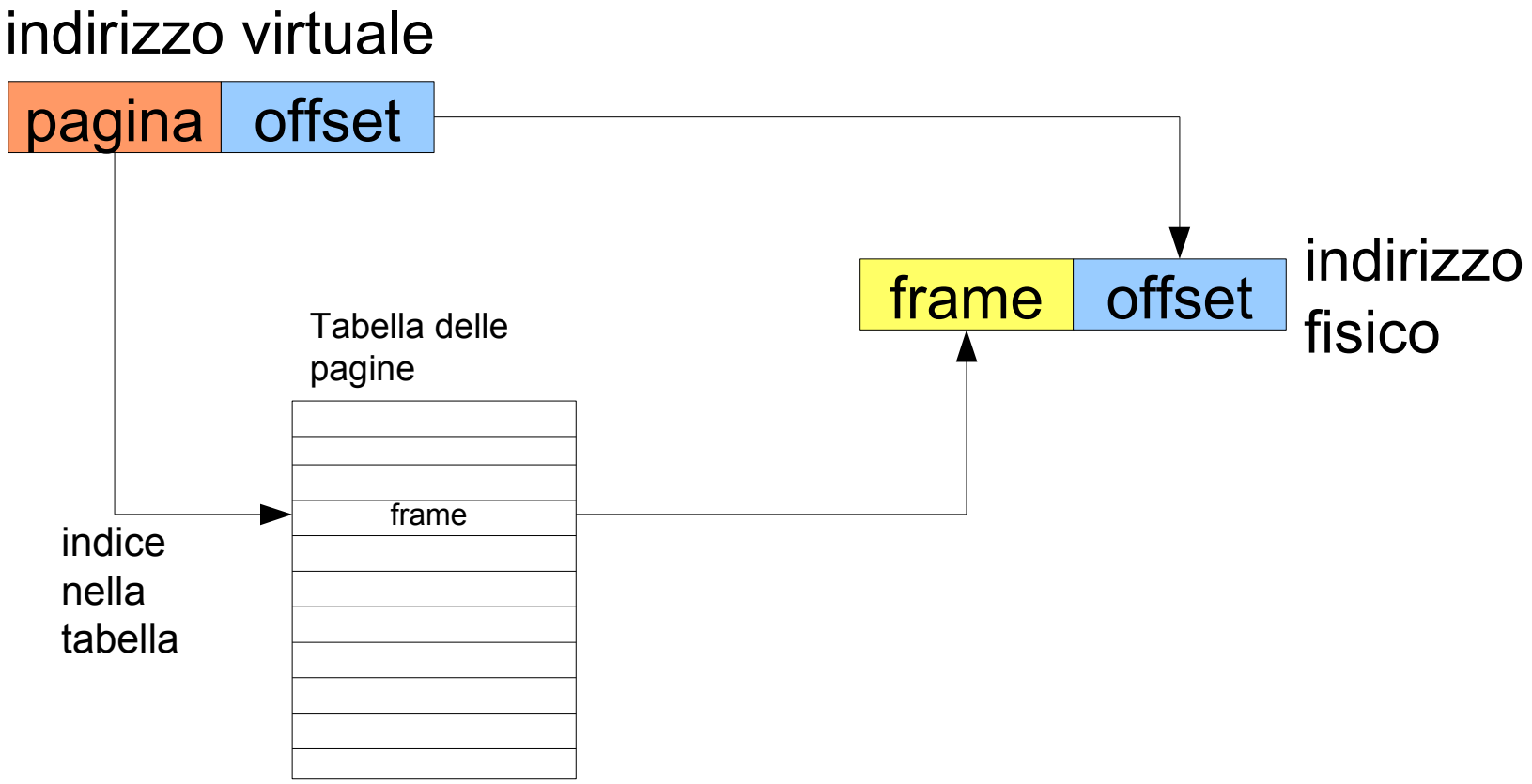
Traduzione degli indirizzi: tabella delle pagine

- **un bit indica se la pagina è in memoria oppure no**
  - se non è in memoria il sistema operativo riceve un segnale (*page fault*) e carica la pagina dal disco
  - se la pagina non esiste neppure su disco il processo riceve un segnale di terminazione



da A. Tanenbaum, Modern Operating Systems, 2nd Ed.

Traduzione degli indirizzi



Swapping di pagine

- Non tutte le pagine devono trovarsi nella memoria fisica durante l'esecuzione
  - il sistema operativo può spostare (page swapping) su disco le pagine che non sono correntemente utilizzate, liberando la memoria fisica
  - quando un processo fa riferimento a una pagina che non è in memoria, il sistema operativo viene notificato con un page fault e provvede a caricarla dal disco

Pagine dirty

- Non sempre è necessario ri-salvare una pagina su disco quando vogliamo liberare la memoria fisica
  - se la pagina è già sul disco e non è stata modificata dall'ultimo "swap in"
  - altrimenti la pagina è detta "**dirty**" e il sistema deve riscriverla su disco
- Nella tabella delle pagine viene utilizzato un bit (**dirty bit**) per indicare se il contenuto è stato modificato oppure no

Trashing

- Solitamente il page swapping è abbastanza veloce perché coinvolge solo alcune pagine

- Se la memoria fisica a disposizione non è sufficiente il numero di page fault può essere molto alto, e il sistema operativo inizia a fare swapping aggressivo
  - si parla di trashing

Algoritmi di rimpiazzamento delle pagine

- Quando è necessario "liberare" la memoria fisica, un algorithmo deve scegliere quale pagina spostare su disco
  - esempi:
    - **FIFO** (First In First Out)
    - **LRU** (Least Recently Used): sposta la pagina che è stata meno recentemente utilizzata
    - **Working Set**: cerca di mantenere le pagine più frequentemente utilizzate in memoria

Partizione swap e file di paging

- Windows utilizza **un file su disco** (pagefile.sys) detto file di paging
- Linux tipicamente utilizza **una partizione speciale** del disco come spazio per lo swap, ma è possibile utilizzare anche un file
  - **`mkswap`** (comando per creare un'area di swap su disco o su una partizione)
  - **`swapon, swapoff`** (comandi per attivare/disattivare un'area di swap)

Dividere la memoria

- I processi utilizzano più intervalli di memoria, ognuno in modo diverso
  - codice macchina
  - variabili globali
  - stack
  - ...
- Questi intervalli hanno bisogno di privilegi diversi
  - non tutti gli intervalli contengono codice eseguibile
  - non tutti gli intervalli devono essere modificabili durante l'esecuzione

Segmentazione

- Il meccanismo della segmentazione permette di dividere le diverse parti logiche di un processo (dati, codice,...) in spazi di indirizzamento separati

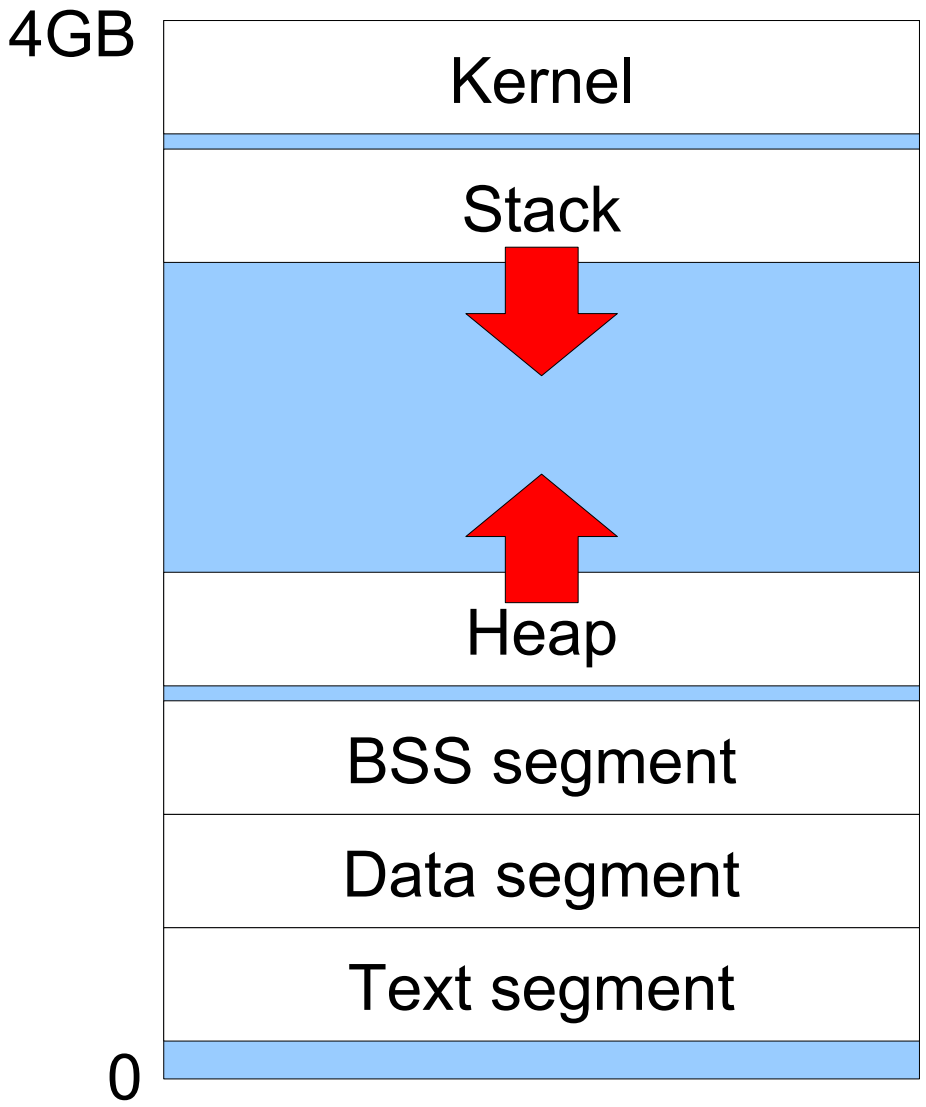
- Ogni "sezione" di memoria è chiamata **segmento**
  - è definito da un **indirizzo base** e una **dimensione** (o lunghezza) che determinano un nuovo spazio di indirizzamento
  - può avere privilegi specifici (r, w, x,...) → **protezione**
  - può essere utilizzato da più processi → **condivisione**

Segmentazione

- la "traduzione" degli indirizzi è affidata alla MMU
- a differenza del paging è utilizzabile direttamente dal programmatore (o dal compilatore) per **dividere la memoria in unità logiche differenti**

Mappa di un processo in memoria

- Codice
- Dati
  - Data segment (variabili inizializzate)
  - BSS * segment (variabili non inizializzate)
  - Heap (variabili allocate dinamicamente)
- Stack



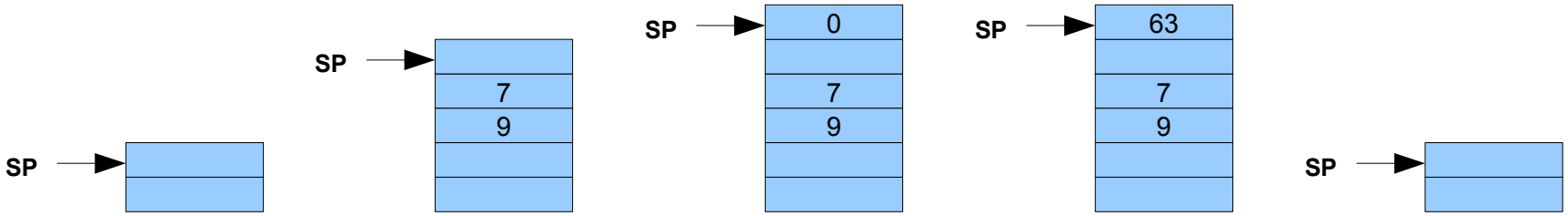
* Block Started by Symbol

Piccola parentesi: lo stack

- Lo **stack** (pila) è una struttura dati con semantica LIFO (Last-In, First-Out)
  - L'ultimo elemento a venir aggiunto è il primo a venir rimosso
- È utilizzato durante l'esecuzione dei processi per mantenere le variabili locali e per il passaggio di parametri
- Un registro del processore (SP, stack pointer) contiene l'indirizzo della cima dello stack

Piccola parentesi: lo stack

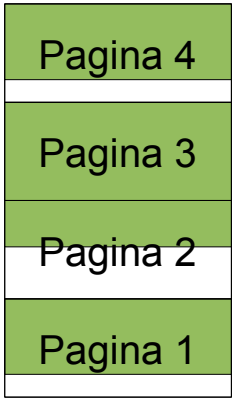
```
int moltiplica(int a, int b) {
	int risultato;
	risultato = a * b;
	return risultato;
}

void main(void) {
	int k;
	k = moltiplica(7,9);
}
```



Frammentazione

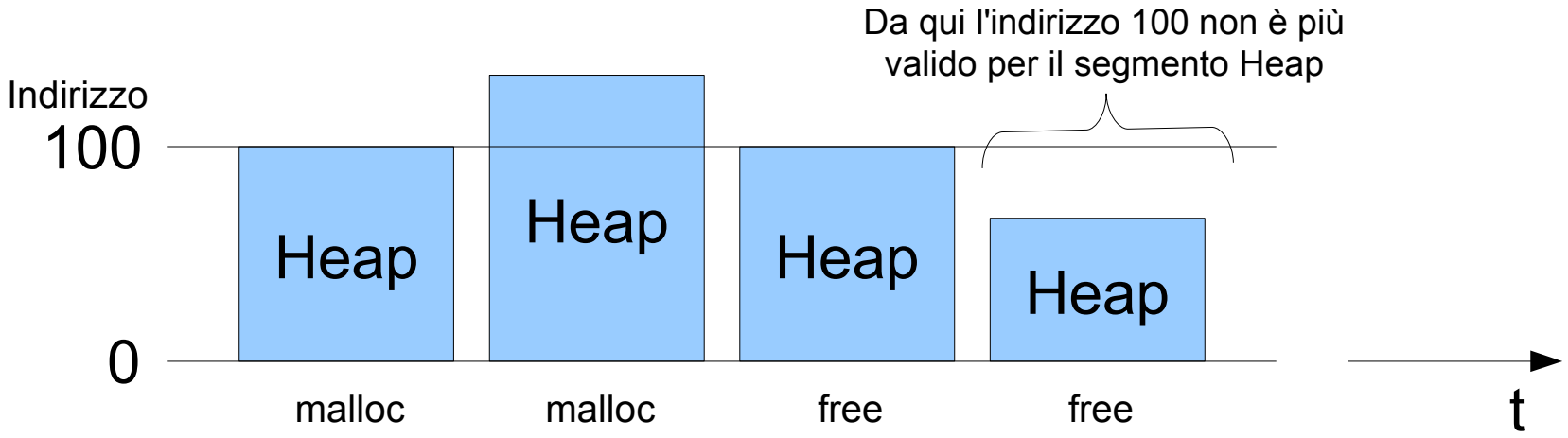
- I segmenti hanno dimensione variabile
  - possono portare a **frammentazione esterna**

- Le pagine hanno dimensione fissa
  - possono portare a **frammentazione interna** quando non tutto lo spazio di una pagina viene utilizzato



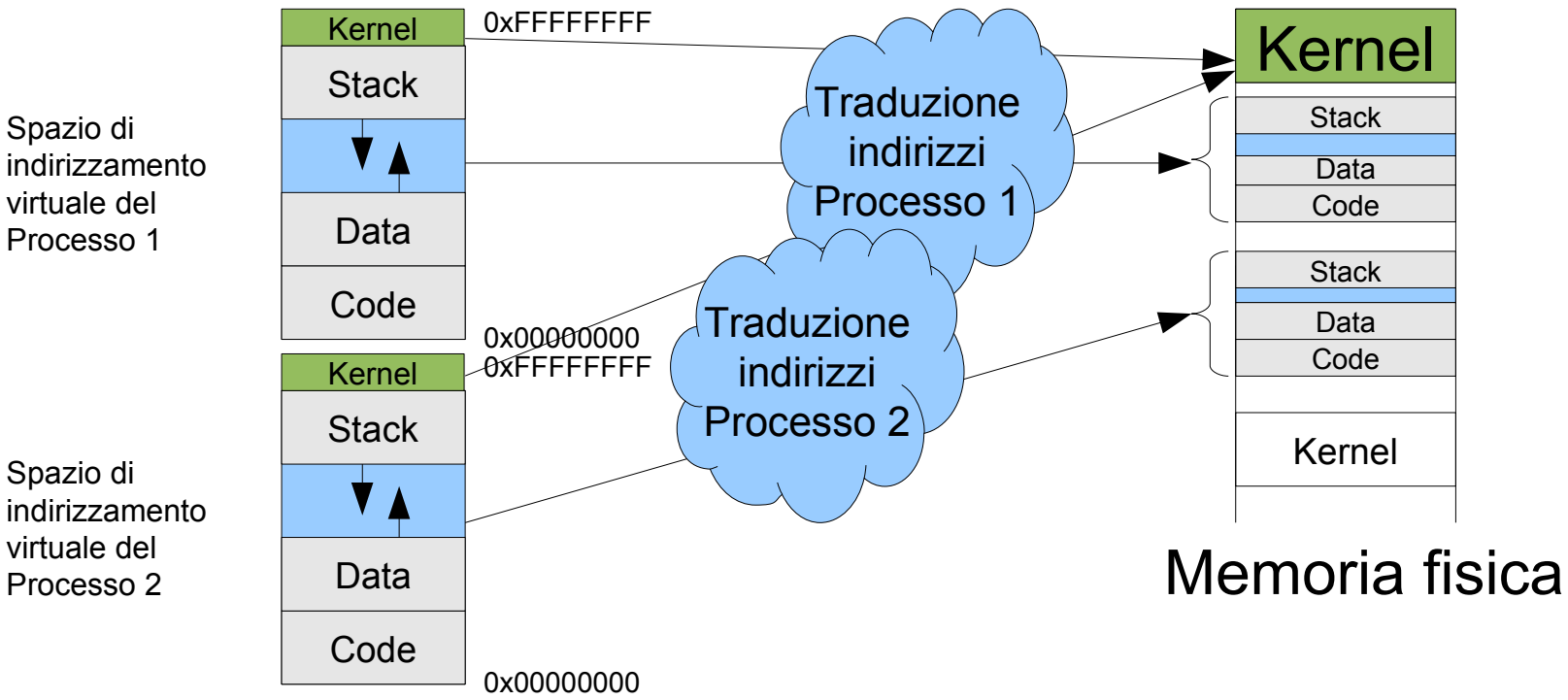
Segmentation fault

- Quando la memoria viene allocata dinamicamente (es. malloc/free in C) il segmento Heap viene ridimensionato
- Quando un processo cerca di accedere a un indirizzo al di fuori di un segmento valido il sistema genera un errore di segmentazione, il **segmentation fault**



Segmentazione e paginazione

- I segmenti possono essere divisi su più pagine



Confronto tra segmentazione e paginazione

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

da A. Tanenbaum, Modern Operating Systems, 2nd Ed.

Memoria fisica occupata, memoria virtuale: **ps aux**

[X] bash

```
utente@host:~/Documenti$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
utente    2734  0.0  0.0  50204  3624 ?        Sl   Dec05   0:00 /usr/bin/gnome-keyring-daemon --daemonize
--login
utente    2743  0.0  0.1  29792  6096 ?        Ssl  Dec05   0:00 gnome-session
utente    2752  0.0  0.0   3588   632 ?        S    Dec05   0:00 dbus-launch --sh-syntax --exit-with-session
utente    2753  0.0  0.0  14484  1752 ?        Ssl  Dec05   0:04 /bin/dbus-daemon --fork --print-pid 5
--print-address 7 --session
utente    2842  0.0  0.1   8768  4360 ?        S    Dec05   0:32 /usr/libexec/gconfd-2
utente    2850  0.0  0.3 149372  3884 ?        Ssl  Dec05   0:24 /usr/libexec/gnome-settings-daemon
utente    2852  0.0  0.1  27364  7904 ?        Ss   Dec05   0:00 seahorse-daemon
utente    2859  0.0  0.0   7544  2356 ?        S    Dec05   0:00 /usr/libexec/gvfsd
utente    2866  0.0  0.0  39520  2436 ?        Ssl  Dec05   0:00 /usr/libexec//gvfs-fuse-daemon
/home/utente/.gvfs
utente    2871  0.0  0.4  54632  6716 ?        S    Dec05   0:14 gnome-panel
utente    2875  0.2  0.1 113152  4936 ?        S<sl Dec05   3:22 /usr/bin/pulseaudio --start --log-
target=syslog
utente    2881  0.0  0.0  11156  2732 ?        S    Dec05   0:00 /usr/libexec/pulse/gconf-helper
utente    2891  0.0  4.4 793812 80788 ?        S    Dec05   0:48 nautilus
utente    2893  0.0  0.0  46872  3420 ?        Ssl  Dec05   0:00 /usr/libexec/bonobo-activation-server --ac-
activate --ior-output-fd=18
```

Resident Set Size memoria fisica utilizzata (non swappata su disco)

Memoria virtuale (kilobytes)

% della memoria fisica occupata (RSS / RAM)

Mappa della memoria, segmenti : **pmap -x PID**

[X] bash

```
utente@host:~/Documenti$ pmap -x 13242
13242:   gnome-terminal
Address   Kbytes     RSS   Dirty Mode  Mapping
00101000     332     196       0 r-x-- libORBit-2.so.0.1.0
(...)
00177000       8       8       8 rw--- libICE.so.6.3.0
00179000       4       0       0 rw---   [ anon ]
0017a000      48      16       0 r-x-- libnss_files-2.12.so
(...)
003d3000     160      56       0 r-x-- libm-2.12.so
003fb000       4       4       0 r---- libm-2.12.so
003fc000       4       4       4 rw--- libm-2.12.so
b609b000     608     124       0 r---- DejaVuSans.ttf
b6133000     296      44       0 r---- DejaVuSansMono-Bold.ttf
b617d000     316      64       0 r---- DejaVuSansMono.ttf
b61cf000      24      24       0 r--s- b79f3aaa7d385a141ab53ec885cc22a8-le32d4.cache-3
b61d5000       8       8       0 r--s- 0b1bcc92b4d25cc154d77dafe3bceaa0-le32d4.cache-3
(...)
bfb70000     108      96      96 rw---   [ stack ]
--------- ------- ------- ------- -----
total kB   5804       -       -       -
```

Permessi

Pagine dirty (kilobytes)

Resident Set Size memoria fisica utilizzata

Dimensione (kilobytes)

Indirizzo base (virtuale)

Memoria utilizzata, libera e swapping : **free** e **vmstat**

[X] bash

```
utente@host:~/Documenti$ free
             total       used       free     shared    buffers     cached
Mem:       4030168    2696576    1333592          0     273684    1351132
-/+ buffers/cache:    1071760    2958408
Swap:      6127608          0    6127608
```

Page Cache per operazioni su file

Memoria fisica e swap (kilobytes)

Buffer per operazioni I/O

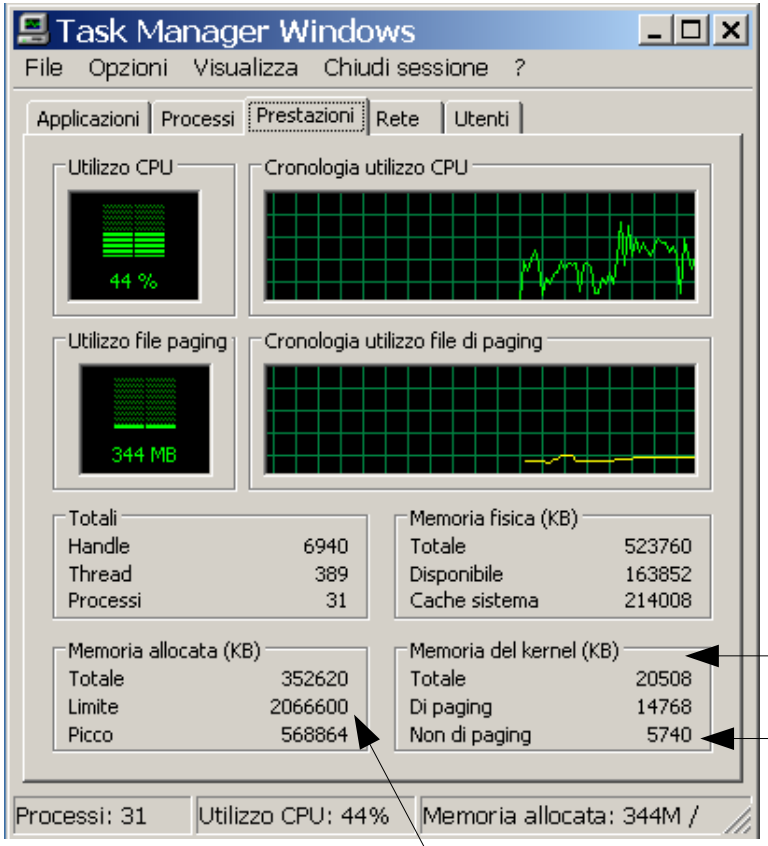
[X] bash

```
utente@host:~/Documenti$ vmstat
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 1332684 273552 1349896    0    0     3     8   10   84  3  1 96  0  0
```

swpd: swap usato
free: libera
buff: buffers
cache: page cache

si: swap-in / secondo
so: swap out / secondo

Come viene utilizzata la memoria: Task Manager Windows

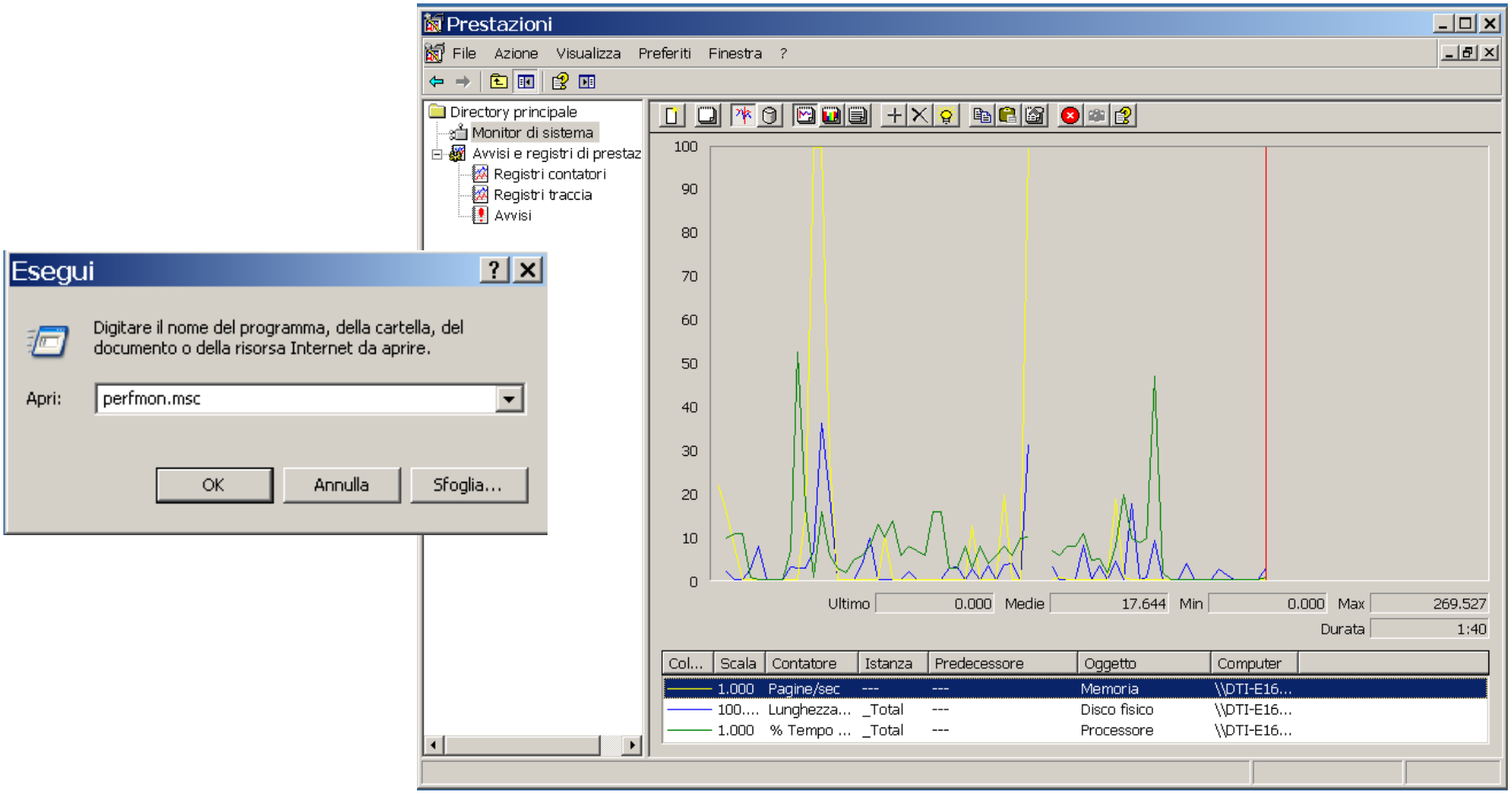


CTRL + ALT + DEL

Memoria usata dal sistema operativo
Non di paging: memoria che non deve essere mai spostata su disco

Memoria virtuale massima che il SO può dare ai programmi senza dover ingrandire il file di paging

Come viene utilizzata la memoria: System Monitor



Come viene utilizzata la memoria: System Monitor

- È possibile aggiungere nuovi contatori, per esempio il numero di pagine lette o scritte su disco