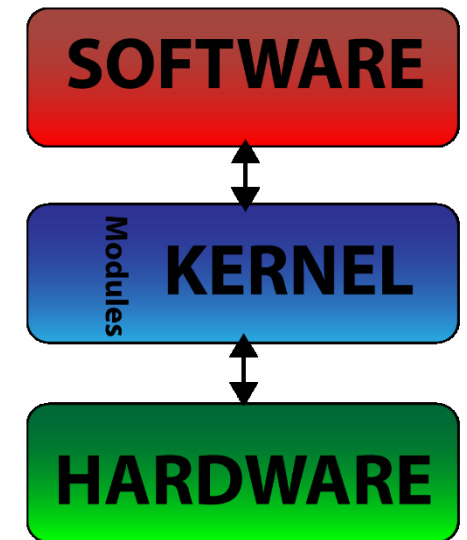
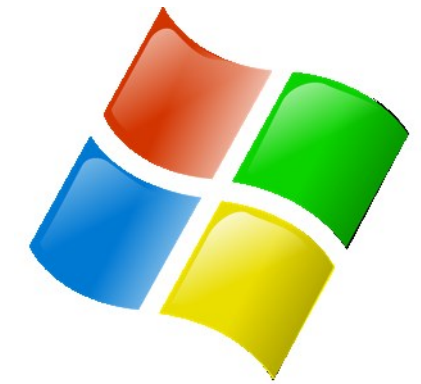




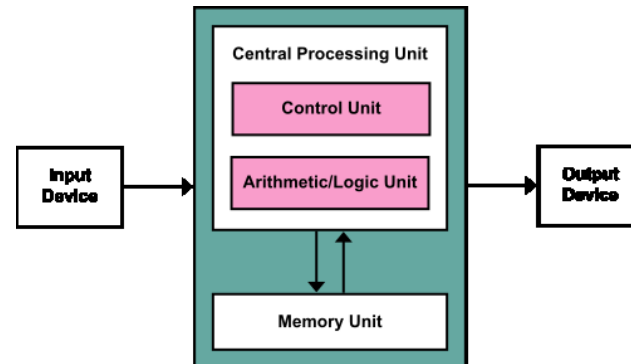
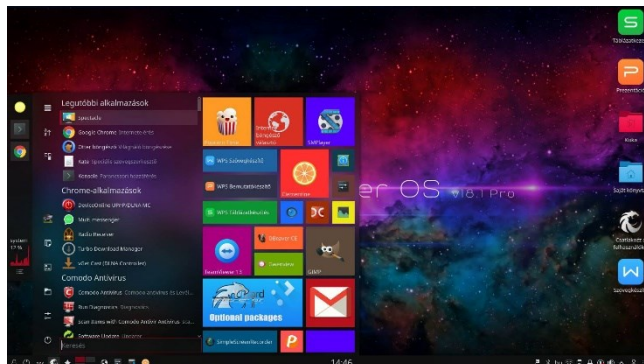
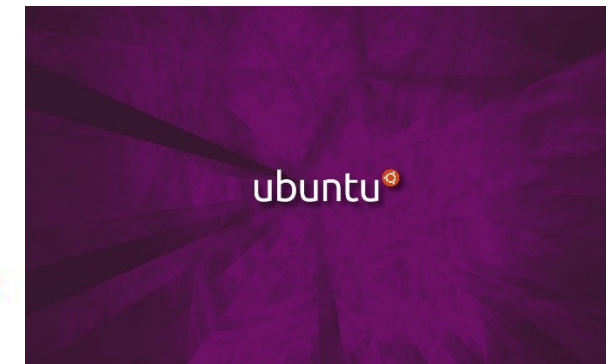
**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

*Corso di Sistemi Operativi
A.A. 2019/20*

Memoria centrale



Docente:
**Domenico Daniele
Bloisi**



Novembre 2019

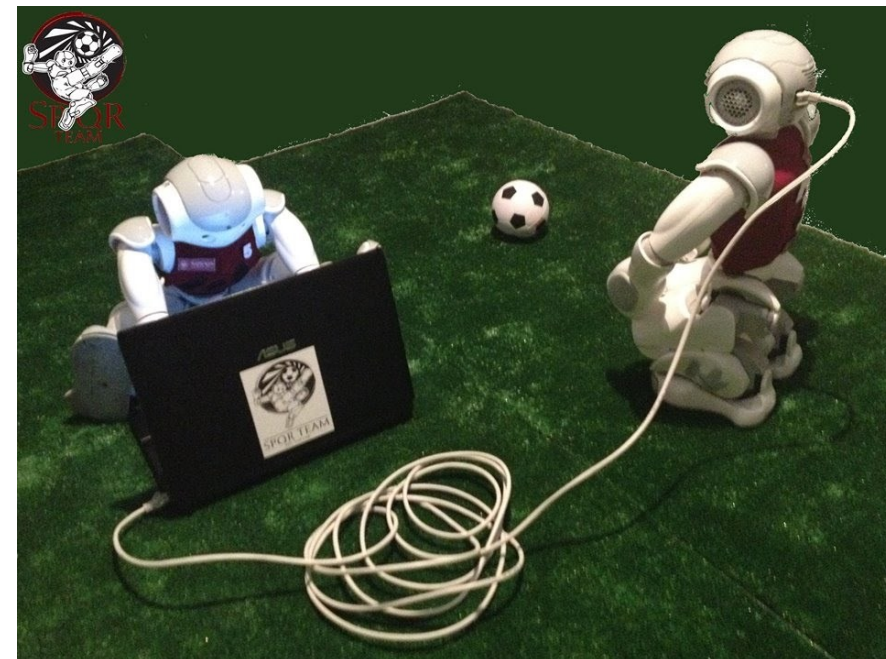
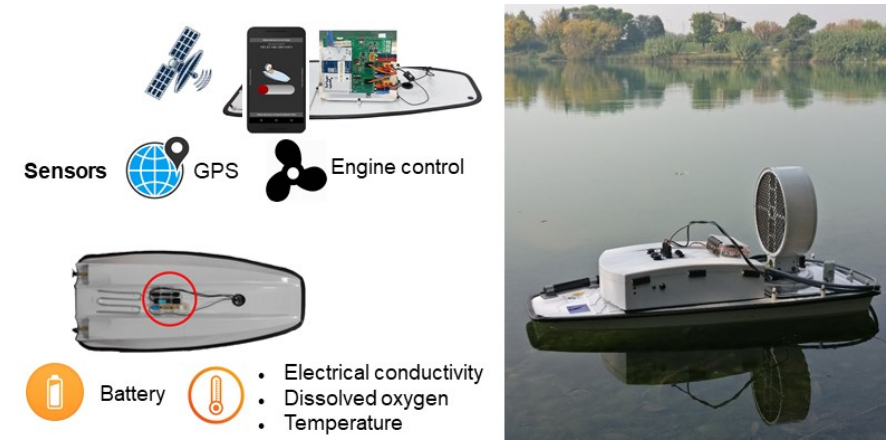
Domenico Daniele Bloisi

- Ricercatore RTD B
Dipartimento di Matematica, Informatica
ed Economia
Università degli studi della Basilicata

<http://web.unibas.it/bloisi>

- SPQR Robot Soccer Team
Dipartimento di Informatica, Automatica
e Gestionale Università degli studi di
Roma “La Sapienza”

<http://spqr.diag.uniroma1.it>



Ricevimento

- In aula, subito dopo le lezioni
- Martedì dalle 11:00 alle 13:00 presso:
Campus di Macchia Romana
[Edificio 3D](#) (Dipartimento di Matematica,
Informatica ed Economia)
[Il piano, stanza 15](#)

Email: domenico.bloisi@unibas.it



Programma – Sistemi Operativi

- Introduzione ai sistemi operativi
- Gestione dei processi
- Sincronizzazione dei processi
- **Gestione della memoria centrale**
- Gestione della memoria di massa
- File system
- Sicurezza e protezione

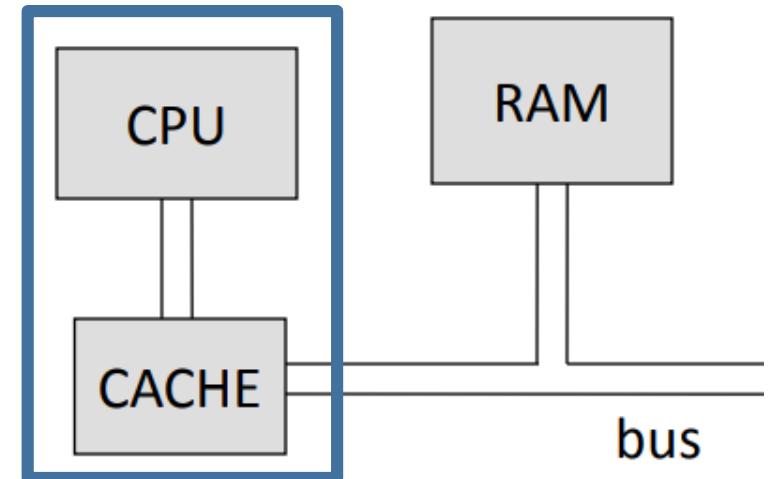
Esecuzione di un programma

Durante l'esecuzione, i **programmi** e i **dati** cui essi accedono devono trovarsi, almeno parzialmente, in memoria centrale.

La scelta di un metodo di gestione della memoria, per un sistema specifico, dipende da molteplici fattori, in particolar modo dall'*architettura hardware*.

Memoria centrale

- **RAM** (Random-Access Memory - memoria ad accesso casuale) è una memoria volatile, cioè essa perde il proprio contenuto allo spegnimento del computer
- **ROM** (Read-Only Memory - memoria a sola lettura) è una memoria non volatile in grado di mantenere memorizzati i dati anche in assenza di alimentazione elettrica
- **CACHE** (dal termine francese caché che significa “nascosto”) un tipo di memoria volatile come la RAM, ma molto più veloce e più costosa di quest’ultima



Spazio di memoria

Ciascun processo deve avere uno **spazio di memoria separato**, in modo da proteggere i processi l'uno dall'altro:

- ciò è fondamentale per avere più processi caricati in memoria per l'esecuzione concorrente.

Bisogna proteggere il sistema operativo dall'accesso dei processi utenti e, in sistemi multiutente, salvaguardare i processi utenti uno dall'altro.

- Tale **protezione** è implementata a **livello hardware** poiché il sistema operativo (per questioni di prestazioni) non interviene negli accessi della CPU alla memoria.

Registro base e registro limite

Si può implementare il meccanismo di protezione tramite due registri, detti **registro base** e **registro limite**

Registro base contiene il più piccolo indirizzo legale della memoria fisica (per es. 300040)

Registro limite determina la dimensione dell'intervallo ammesso (per es. 120900)

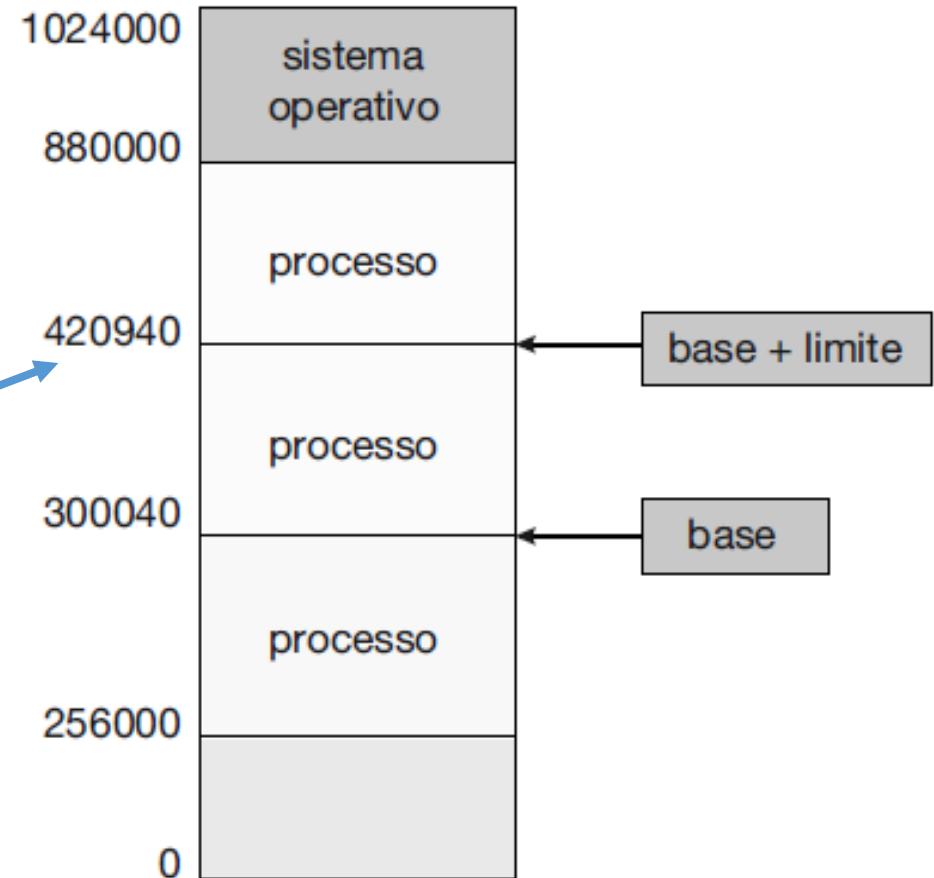


Figura 9.1 I registri base e limite definiscono lo spazio degli indirizzi logici.

Protezione delle aree di memoria

Qualsiasi tentativo da parte di un programma eseguito in modalità utente di accedere alle aree di memoria riservate al sistema operativo o a una qualsiasi area di memoria riservata ad altri utenti comporta l'invio di una **eccezione** (*trap*) che restituisce il controllo al sistema operativo che, a sua volta, interpreta l'evento come un errore fatale.

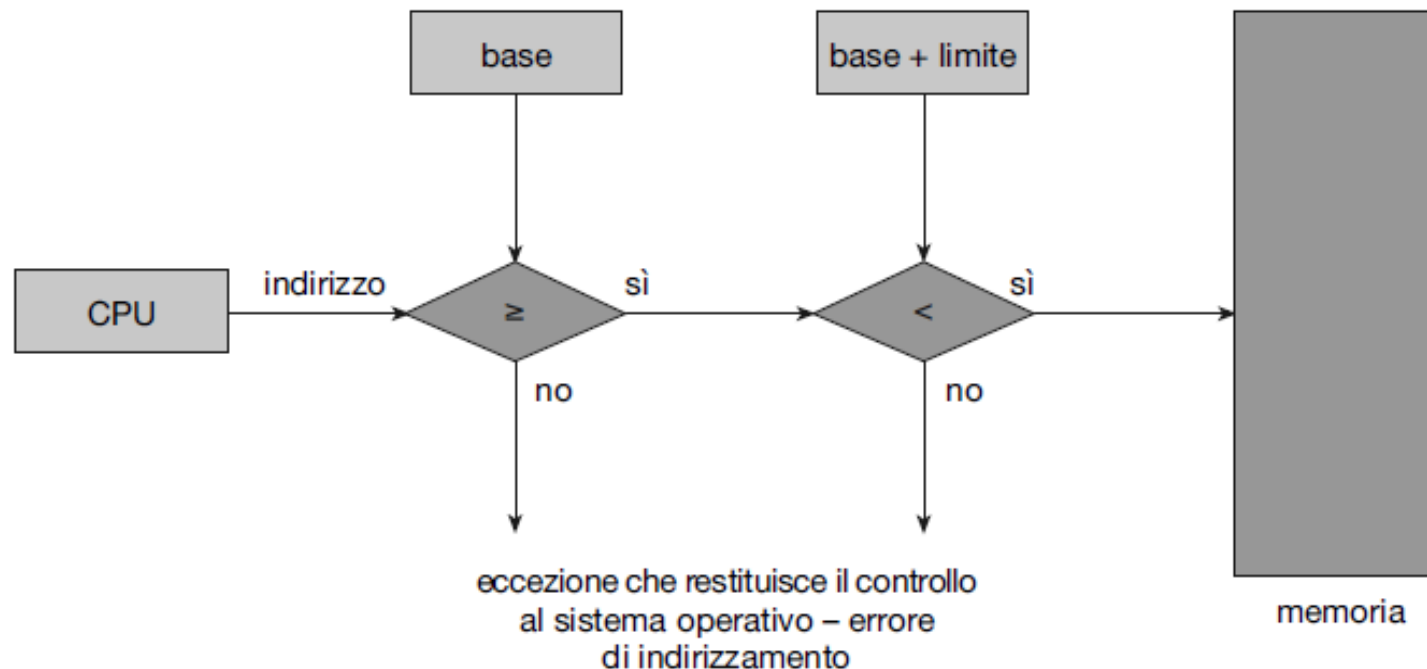


Figura 9.2 Protezione hardware degli indirizzi tramite registri base e limite.

Associazione degli indirizzi

Nella maggior parte dei casi un programma utente, prima di essere eseguito, deve passare attraverso *varie fasi*, alcune delle quali possono essere facoltative

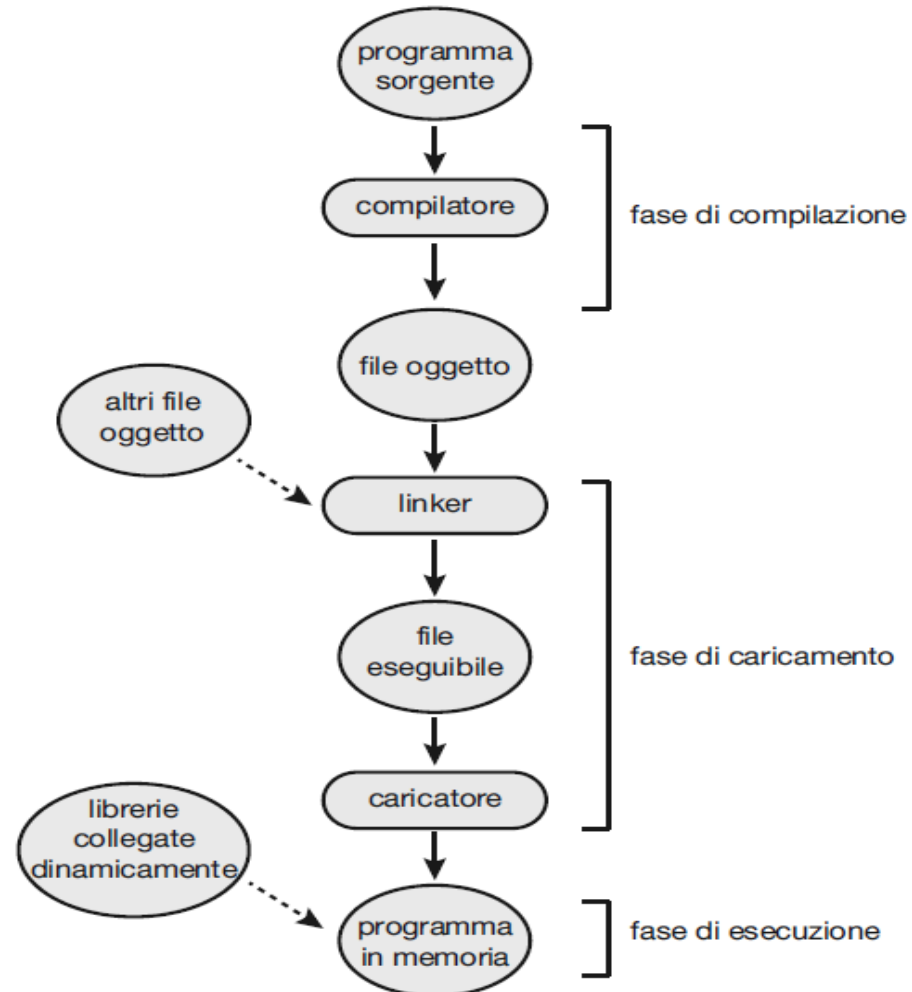


Figura 9.3 Fasi di elaborazione di un programma utente.

Associazione degli indirizzi

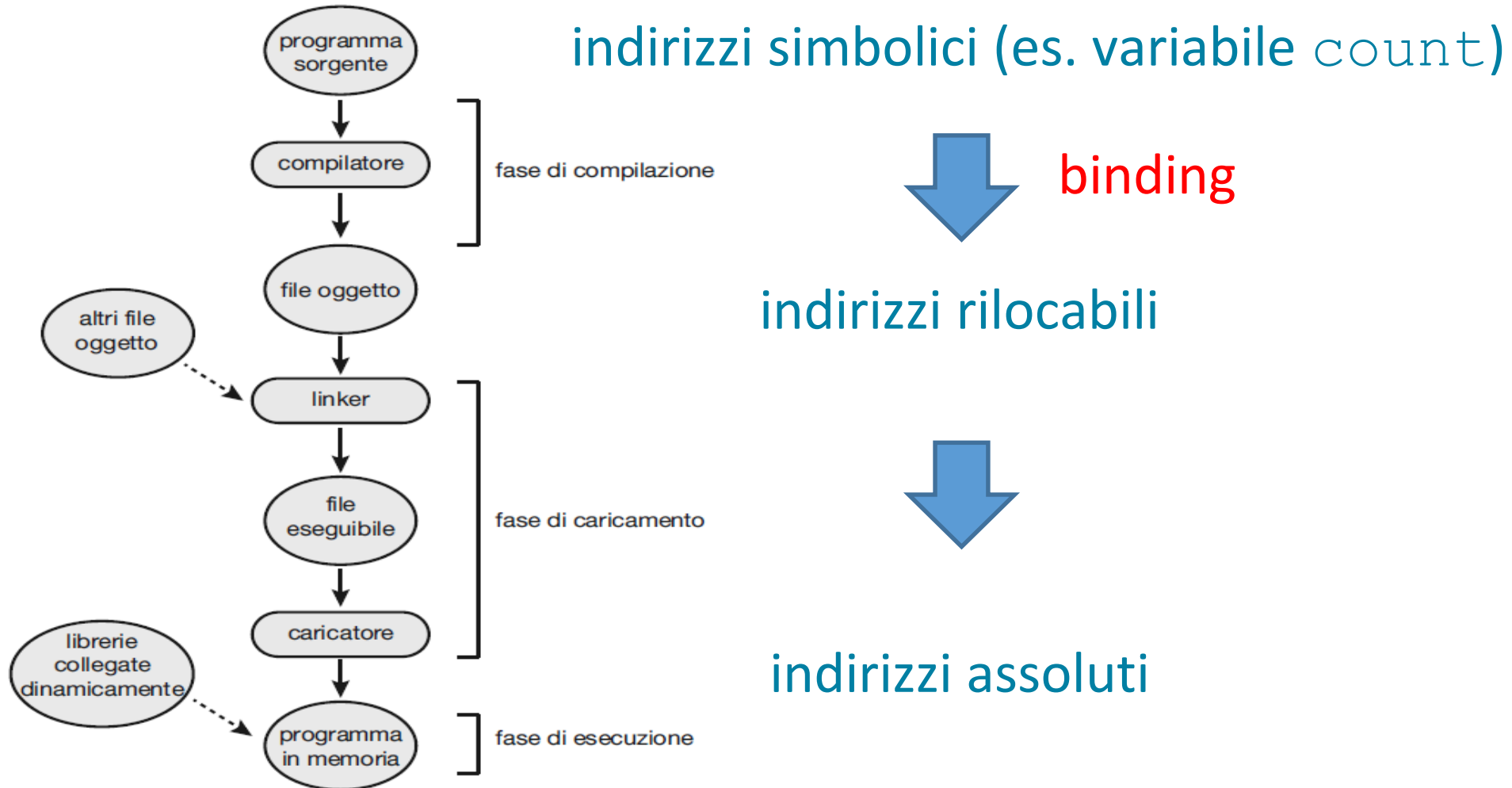


Figura 9.3 Fasi di elaborazione di un programma utente.

Associazione degli indirizzi

Generalmente, l'associazione di istruzioni e dati a indirizzi di memoria si può compiere in qualsiasi fase del seguente percorso.



Associazione degli indirizzi



Se nella fase di compilazione si conosce in che punto della memoria risiederà il processo, si può generare **codice assoluto**

Se nella fase di compilazione non è possibile conoscere in che punto della memoria risiederà il processo, il compilatore deve generare **codice rilocabile**

Se durante l'esecuzione il processo può essere spostato da un segmentato di memoria ad un altro, si deve **ritardare** l'associazione degli indirizzi fino alla fase di esecuzione

Spazio di indirizzi

indirizzo logico → indirizzo generato dalla CPU

indirizzo fisico → indirizzo caricato nel **registro dell'indirizzo di memoria**

indirizzi virtuali → perché con il metodo di associazione nella fase d'esecuzione gli indirizzi logici non coincidono con gli indirizzi fisici

Unità di gestione della memoria

L'unità di gestione della memoria

(*memory management unit*, **MMU**)

svolge l'associazione nella fase d'esecuzione dagli indirizzi virtuali agli indirizzi fisici.

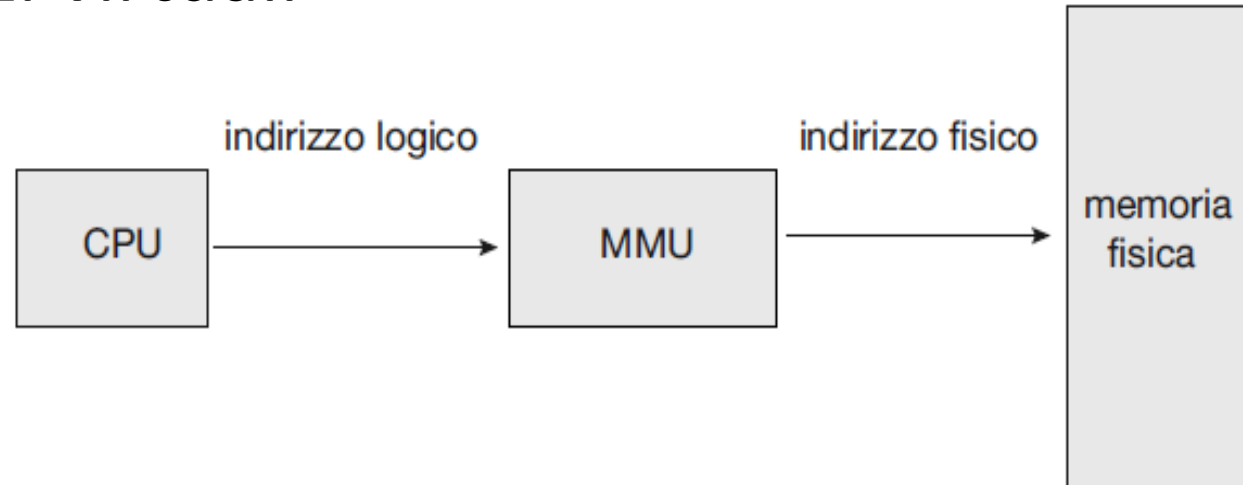


Figura 9.4 Unità di gestione della memoria (MMU).

Registro di rilocazione

Quando un processo utente genera un **indirizzo**, prima dell'**invio all'unità di memoria**, si somma a tale indirizzo il valore contenuto nel **registro di rilocazione**.

il registro di base è ora denominato **registro di rilocazione**

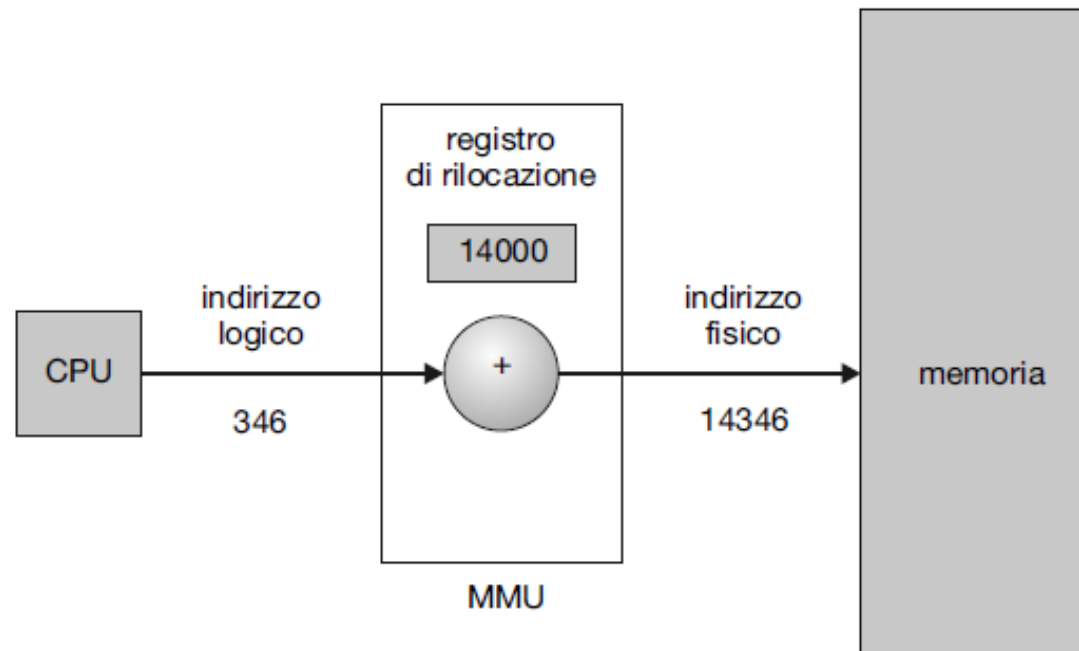


Figura 9.5 Rilocazione dinamica tramite un registro di rilocazione.

Allocazione contigua della memoria

Con l'**allocazione contigua della memoria** ciascun processo è contenuto in una singola sezione di memoria contigua a quella che contiene il processo successivo.

Protezione della memoria

Ogni **indirizzo logico** deve cadere nell'intervallo specificato dal registro limite

la **MMU** fa corrispondere *dinamicamente* l'indirizzo fisico all'indirizzo logico sommando a quest'ultimo il valore contenuto nel registro di rilocazione (Figura 9.6) e invia l'indirizzo risultante alla memoria

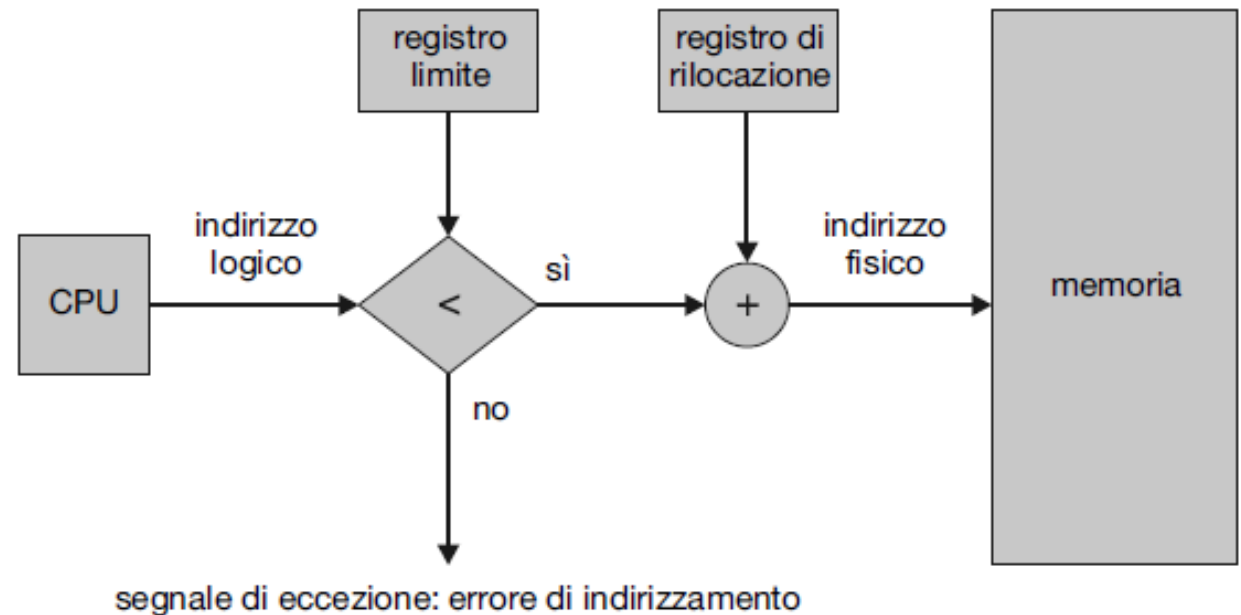


Figura 9.6 Registri di rilocazione e limite.

Vantaggi del registro di rilocazione

Lo schema con registro di rilocazione consente al sistema operativo di cambiare **dinamicamente** le proprie dimensioni.

Tale flessibilità è utile in molte situazioni. Per esempio, se un driver di periferica non è attualmente in uso, può essere rimosso dalla memoria.

Metodi per l'allocazione della memoria

Uno dei metodi più semplici per l'allocazione della memoria consiste nel suddividere la stessa in **partizioni di dimensione variabile**, dove ciascuna partizione può contenere esattamente un processo.

Schema a partizione variabile

Nello **schema a partizione variabile** il sistema operativo conserva una tabella in cui sono indicate le partizioni di memoria disponibili e quelle occupate.

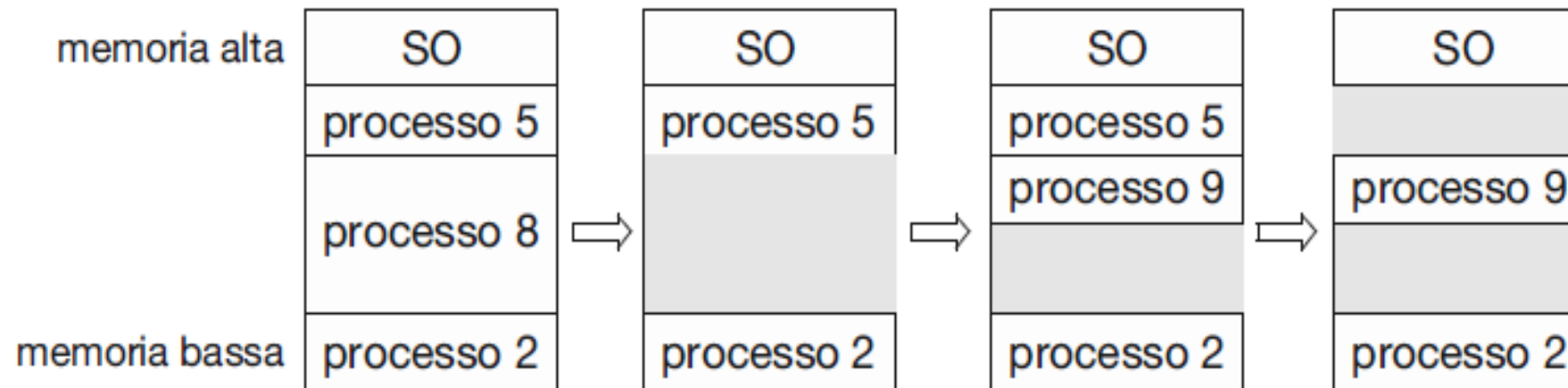


Figura 9.7 Schema a partizione variabile.

Allocazione dinamica

L'**allocazione dinamica della memoria** consente di soddisfare una richiesta di dimensione n data una lista di **buchi** liberi

I criteri più usati per scegliere un **buco** libero tra quelli disponibili nell'insieme sono i seguenti:

First-fit

Si assegna il primo buco abbastanza grande

Best-fit

Si assegna il più piccolo buco in grado di contenere il processo

Worst-fit

Si assegna il buco più grande

Frammentazione

First-fit

Si assegna il primo buco abbastanza grande

Best-fit

Si assegna il più piccolo buco in grado di contenere il processo

Worst-fit

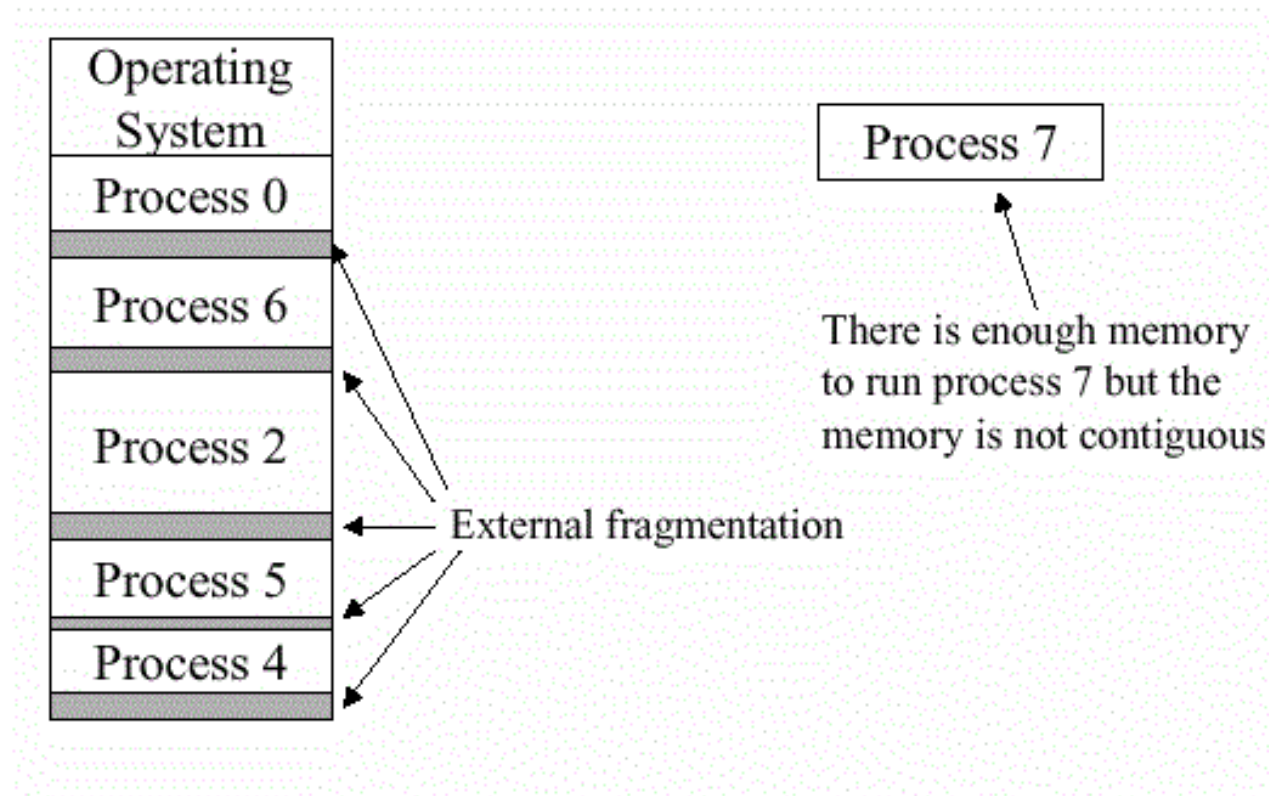
Si assegna il buco più grande

Entrambi i criteri *first-fit* e *best-fit* di allocazione della memoria soffrono di **frammentazione esterna**

Frammentazione esterna

External fragmentation

Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous so it can not be used.



Regola del 50%

L'analisi statistica dell'algoritmo first-fit rileva che per n blocchi assegnati si perdono altri $0,5n$ blocchi a causa della frammentazione.

Ciò significa che potrebbe essere inutilizzabile ben un terzo della memoria.

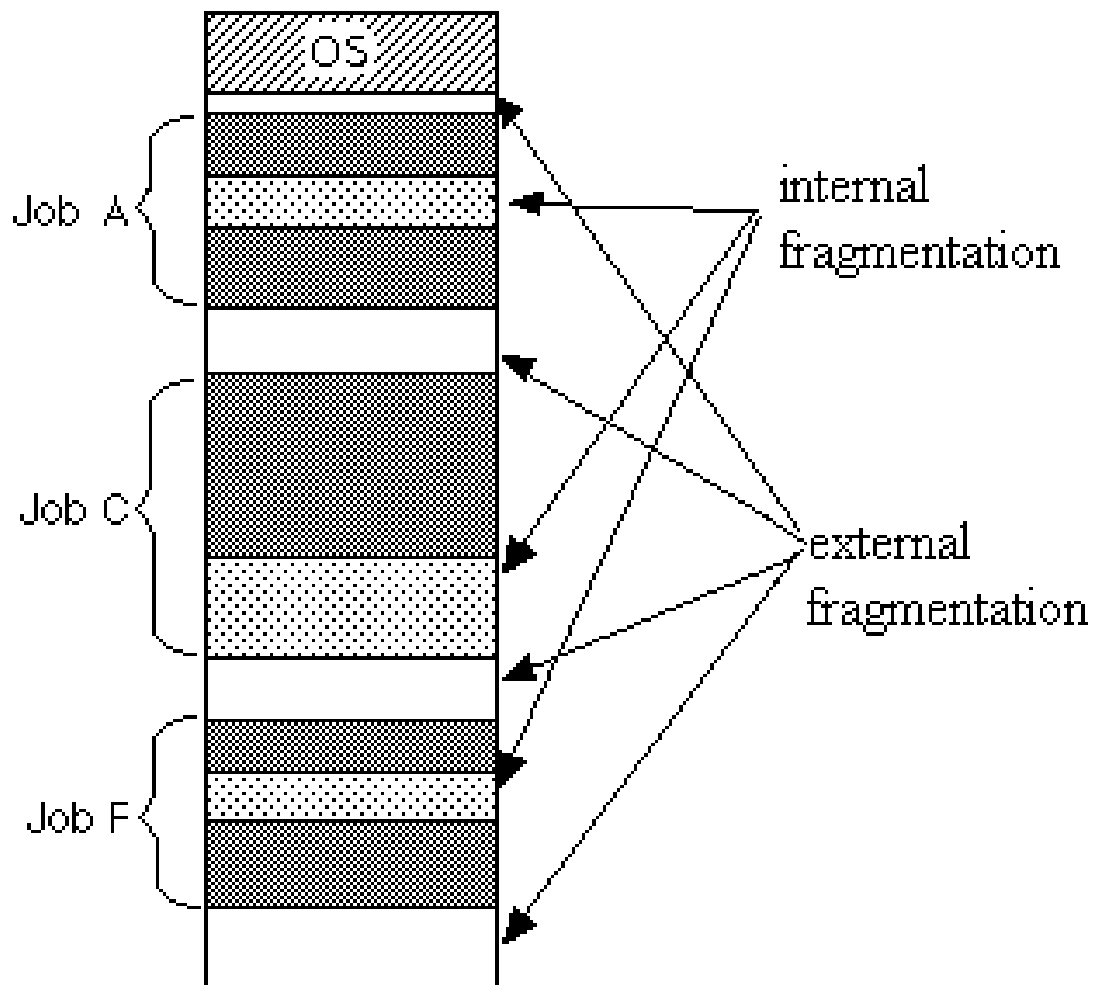
Questa caratteristica è nota come regola del 50%

Frammentazione interna

Internal fragmentation

Memory block assigned to process is bigger.

Some portion of memory is left unused as it cannot be used by another process.



Compattazione

Una soluzione al problema della frammentazione esterna è data dalla compattazione.

Si riordina il contenuto della memoria per riunire le porzioni di memoria libera in un unico grosso blocco.

La compattazione è possibile solo se la rilocazione è dinamica e si effettua nella fase di esecuzione.

Paginazione

Paginazione → schema di gestione della memoria che consente allo spazio di indirizzamento fisico di un processo di essere non contiguo

La paginazione è implementata sfruttando la cooperazione tra sistema operativo e hardware del computer

Paginazione

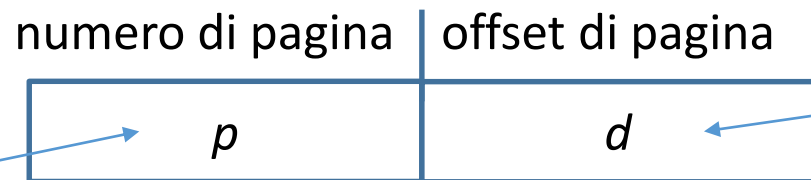
Frame

suddivisione della
memoria fisica in blocchi
di dimensione fissa

Pagine

suddivisione della
memoria logica in blocchi di
pari dimensione

Ogni indirizzo generato dalla CPU è diviso in



Indice per la tabella
delle pagine

Posizione nel frame
a cui viene fatto
riferimento

Paginazione

La **tabella delle pagine** contiene l'indirizzo di base di ciascun frame nella memoria fisica e l'**offset** è la posizione nel frame a cui viene fatto riferimento.

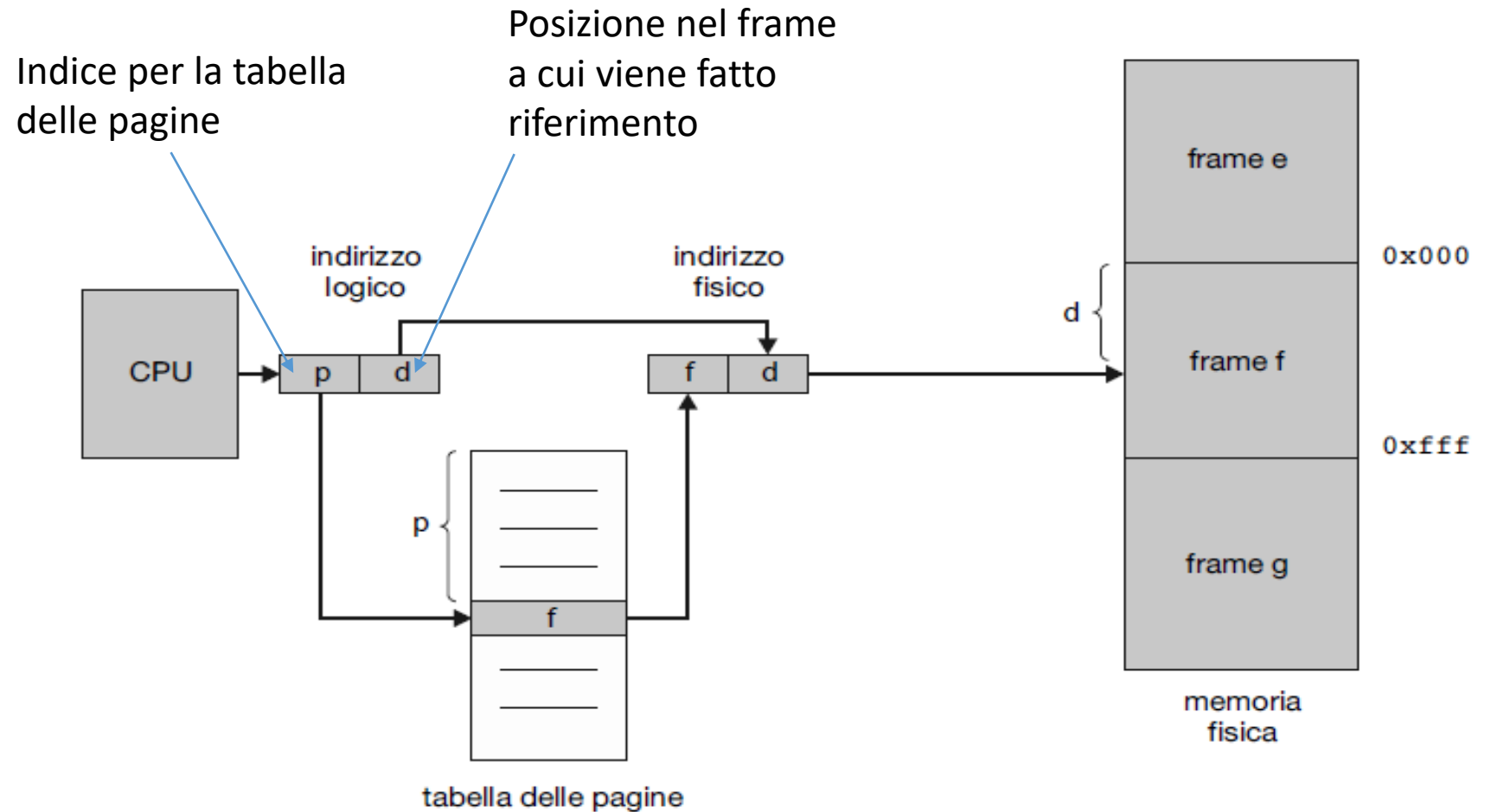


Figura 9.8 Hardware di paginazione.

Modello di paginazione della memoria

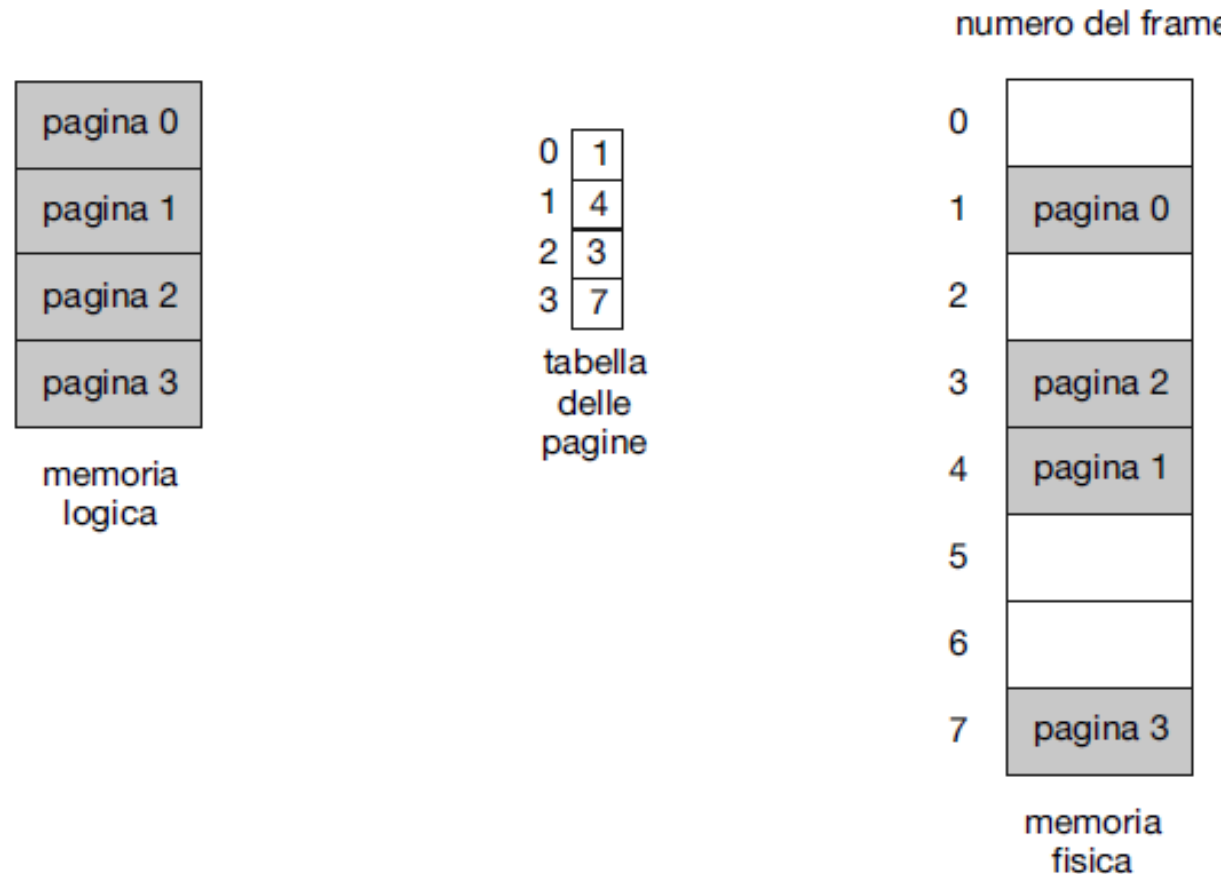


Figura 9.9 Modello di paginazione di memoria logica e memoria fisica.

Traduzione degli indirizzi

indirizzo logico di partenza 

Per tradurre un indirizzo logico in un indirizzo fisico la MMU compie i seguenti passi:

1. Estrae il numero di pagina p e lo utilizza come indice nella tabella delle pagine
2. Estrae il numero di frame f corrispondente dalla tabella delle pagine
3. Sostituisce il numero di pagina p nell'indirizzo logico con il numero di frame f

indirizzo fisico ottenuto 

Dimensione degli indirizzi

Dati

- dimensione dello spazio degli indirizzi logici: 2^m
- dimensione di una pagina: 2^n byte

Si avrà il seguente indirizzo logico



Esercizio

dimensione dello spazio degli indirizzi: 2^4

dimensione di una pagina: 2^2 byte

memoria fisica: 32 byte (8 pagine)

Che indirizzo fisico corrisponde all'indirizzo logico 3?

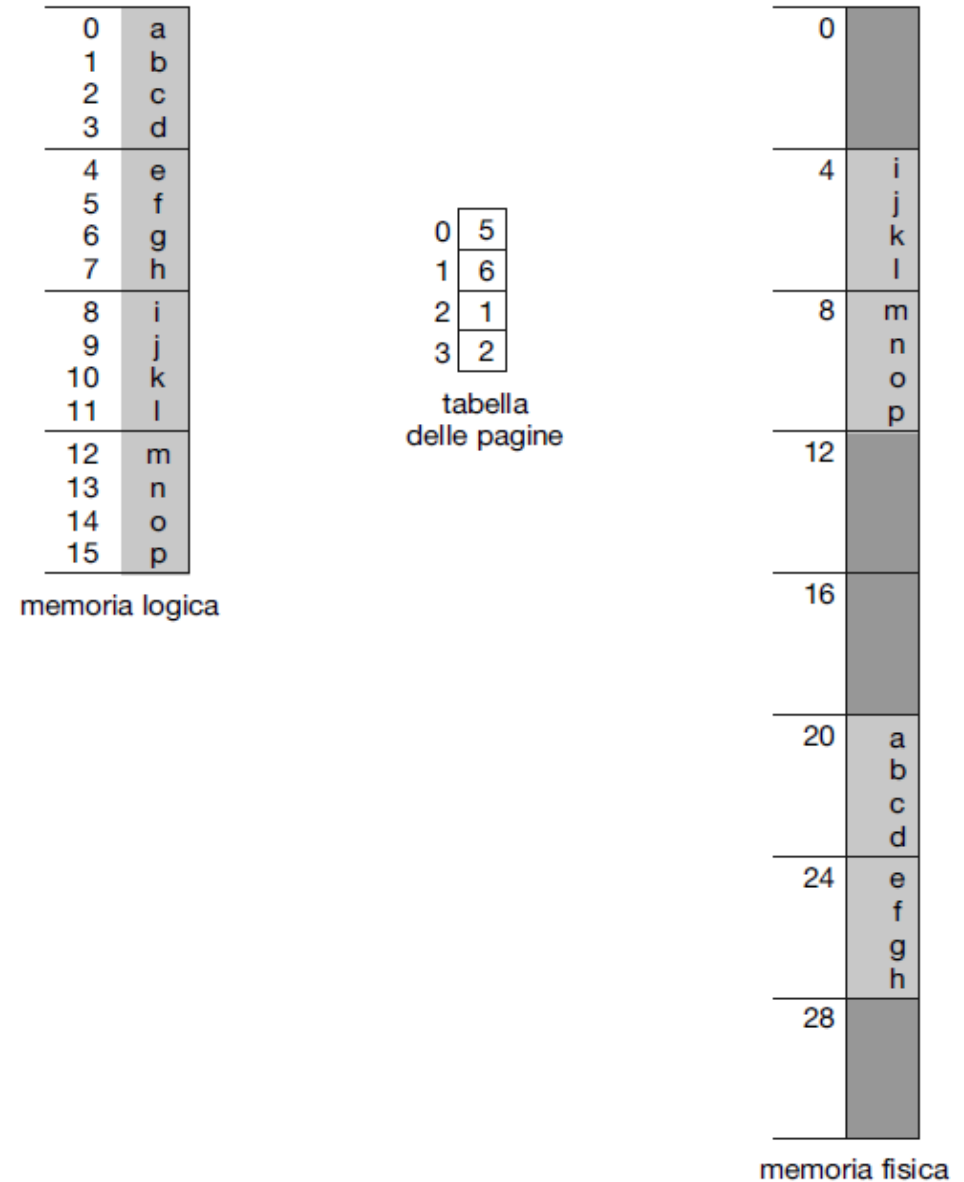
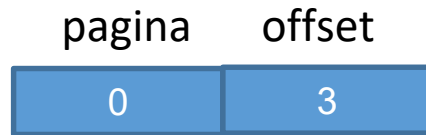


Figura 9.10 Esempio di paginazione per una memoria di 32 byte con pagine di 4 byte.

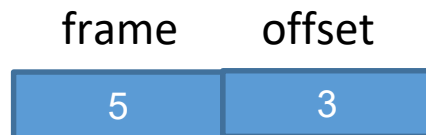
Esercizio

Che indirizzo fisico corrisponde all'indirizzo logico 3?

indirizzo logico 3



indirizzo fisico 23



$$(5 \times 4) + 3 = 23$$

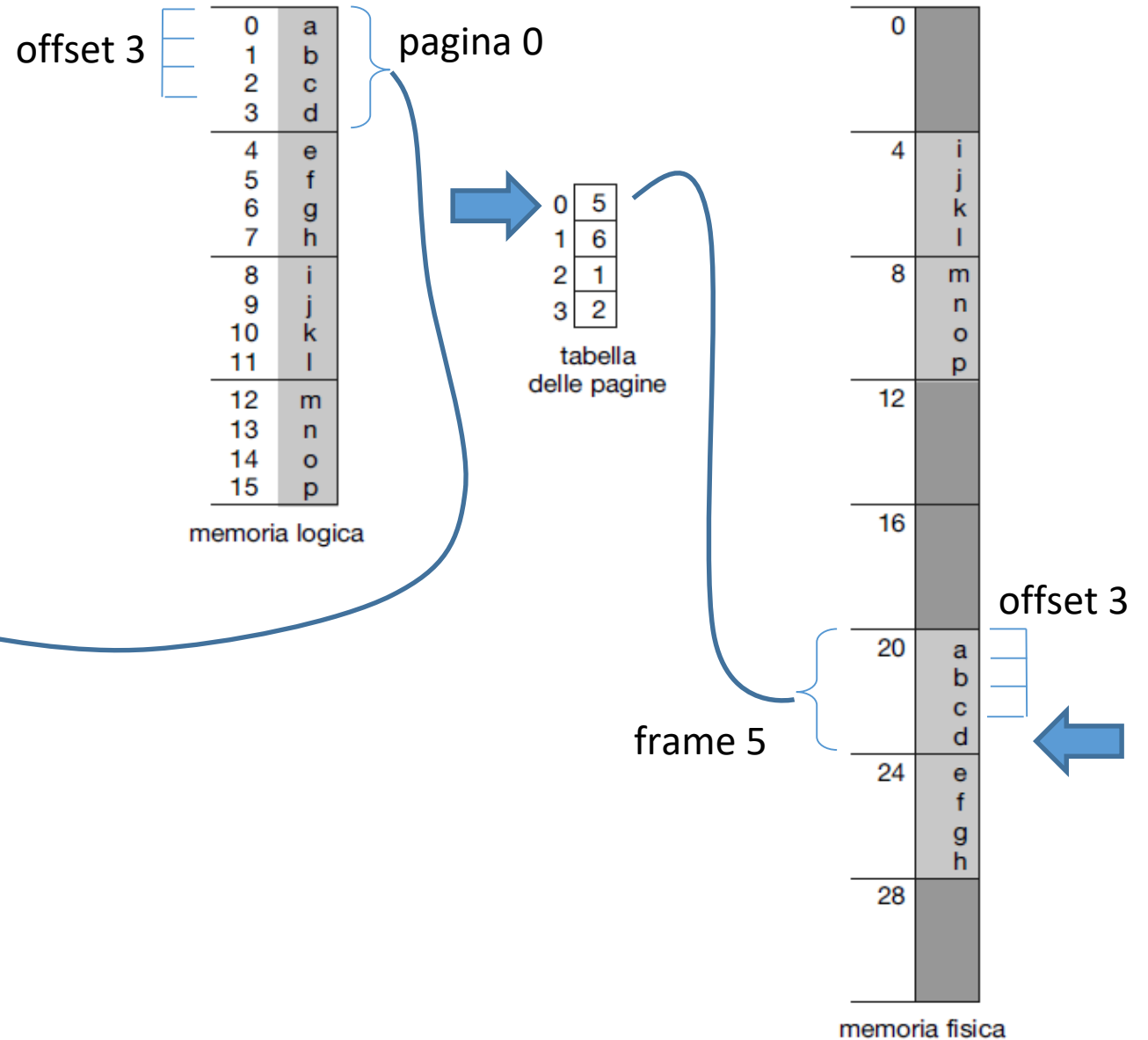


Figura 9.10 Esempio di paginazione per una memoria di 32 byte con pagine di 4 byte.

Ricapitolando: paginazione

La **paginazione** non è altro che una forma di **rilocazione dinamica**: a ogni indirizzo logico l'architettura di paginazione fa corrispondere un indirizzo fisico.

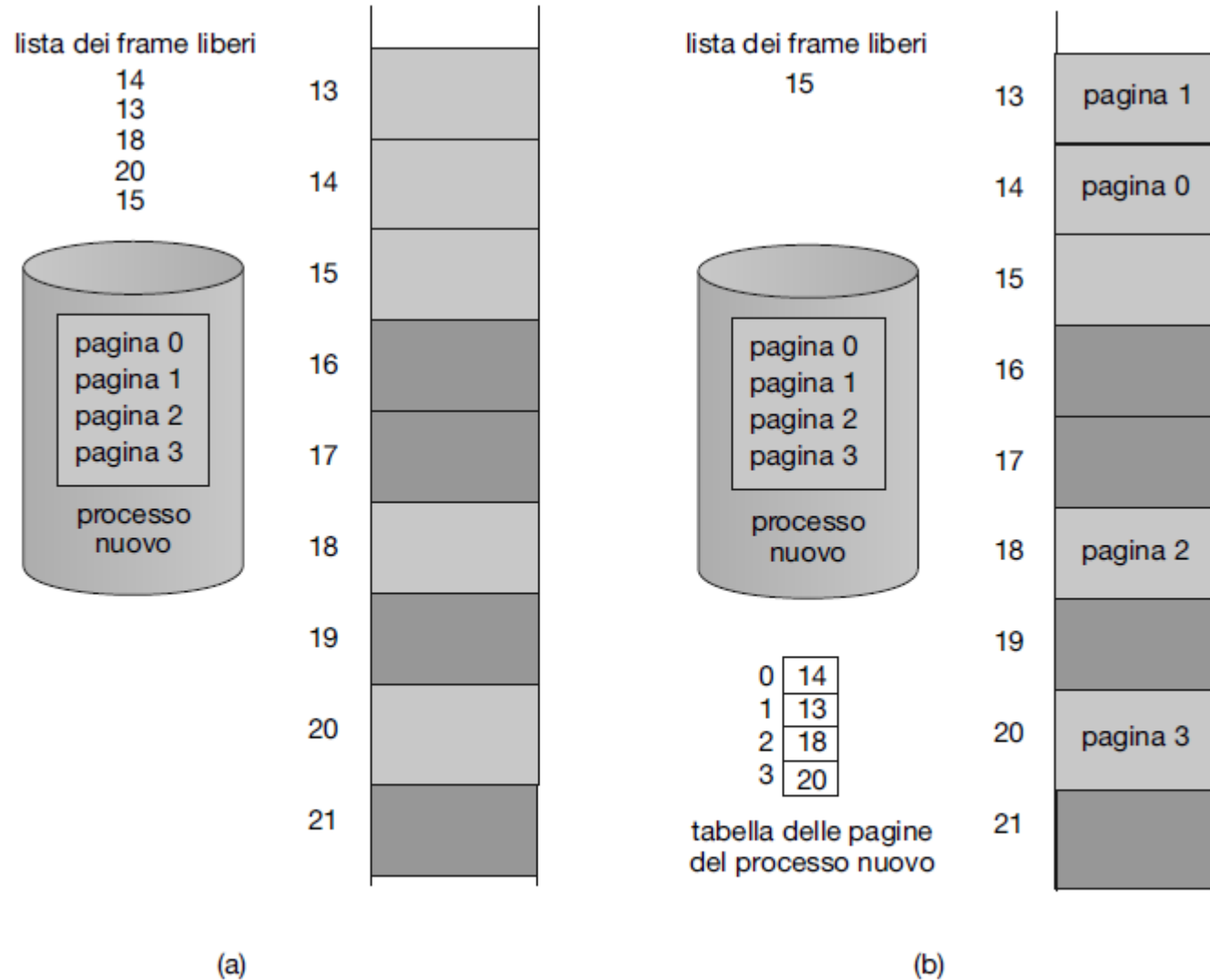
L'uso della **tabella delle pagine** è simile all'uso di una tabella di registri base (o di rilocazione), uno per ciascun frame.

Paginazione e frammentazione

La **paginazione** evita la frammentazione esterna. Qualsiasi frame libero può essere assegnato a un processo che ne abbia bisogno.

La **paginazione** non evita la frammentazione interna. Tipicamente parte dello spazio nell'ultimo frame assegnato a un processo sarà sprecato.

Frame liberi



Ogni processo ha la sua tabella delle pagine

Figura 9.11 Frame liberi; (a) prima e (b) dopo l'allocatione.

TLB (supporto hardware alla paginazione)

TLB (*translation look-aside buffer*) → speciale, piccola cache hardware. Il **TLB** è una **memoria associativa** ad alta velocità in cui ogni elemento consiste di due parti: una chiave (o *tag*) e un valore.

- il **TLB** contiene una piccola parte degli elementi della tabella delle pagine
- quando la CPU genera un indirizzo logico, si presenta il suo numero di pagina al **TLB**
- se tale numero è presente, il corrispondente numero del frame è immediatamente disponibile e si usa per accedere alla memoria ...

TLB

...

- se tale numero è presente, il corrispondente numero del frame è immediatamente disponibile e si usa per accedere alla memoria

Altrimenti



Insuccesso del TLB (*TLB miss*)



si deve consultare la tabella delle pagine in memoria

Paginazione con TLB

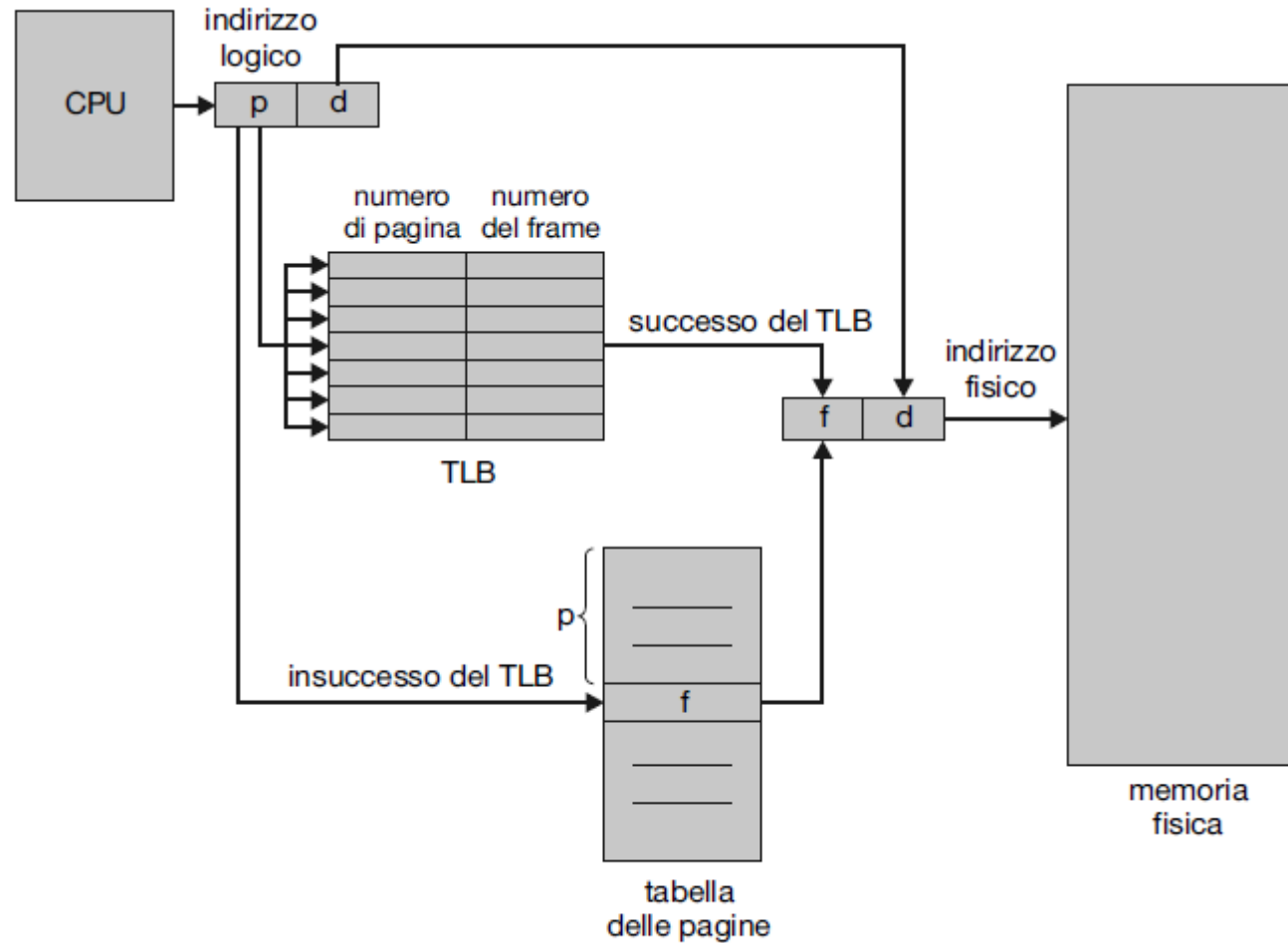


Figura 9.12 Hardware di paginazione con TLB.

ASID

Alcuni **TLB** memorizzano gli **identificatori dello spazio d'indirizzi** (*address-space identifier, ASID*) in ciascun elemento del **TLB**.

Un **ASID** identifica in modo univoco ciascun processo e si usa per fornire al processo corrispondente la **protezione del suo spazio d'indirizzi**.

Hit ratio

tasso di successi



percentuale di volte che il numero di pagina di interesse si trova nel TLB



tempo effettivo d'accesso alla memoria

Bit di validità

bit di validità → ulteriore bit che si associa a ciascun elemento della tabella delle pagine

impostato a *valido*



la pagina corrispondente è nello spazio d'indirizzi logici del processo

impostato a *non valido*



la pagina *non è* nello spazio d'indirizzi logici del processo

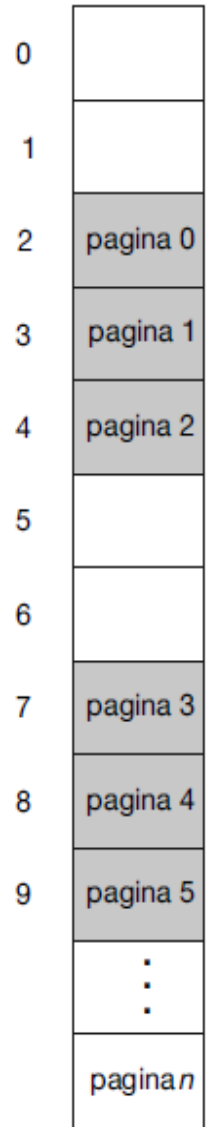
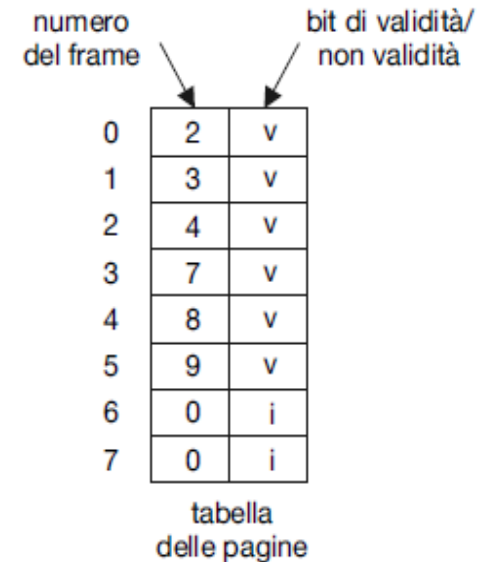
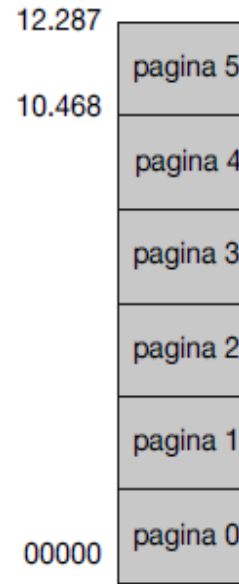


Figura 9.13 Bit di validità (v) o non validità (i) in una tabella delle pagine.

Condivisione

Un vantaggio della **paginazione** risiede nella possibilità di **condividere codice comune**, il che è particolarmente importante in un ambiente con più processi.

La **condivisione della memoria tra processi** di un sistema è simile al modo in cui i thread condividono lo spazio d'indirizzi di un task.

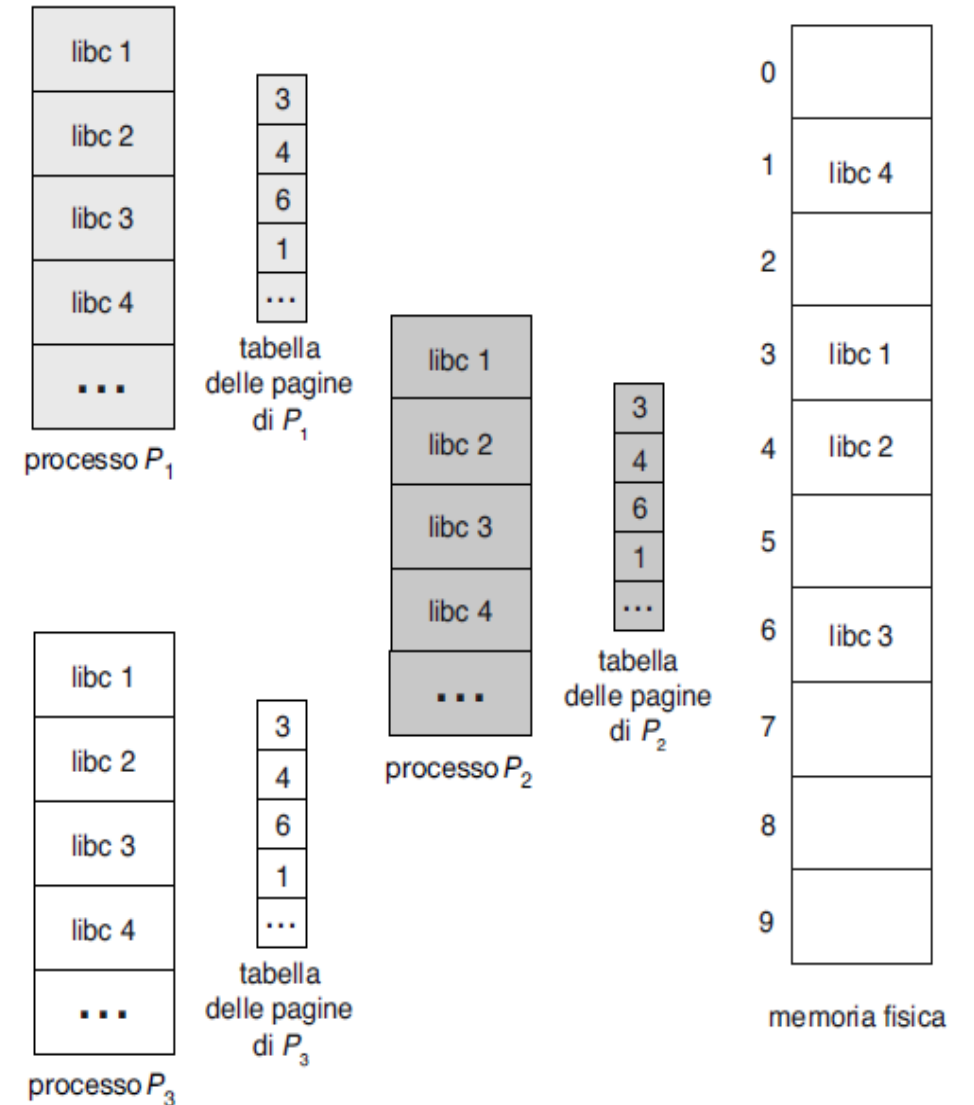


Figura 9.14 Condivisione della libreria standard del C in un ambiente paginato.

Struttura della tabella delle pagine

tecniche più comuni per *strutturare*
la tabella delle pagine

paginazione
gerarchica

tabella delle
pagine di
tipo hash

tabella delle
pagine
invertita

Paginazione gerarchica

Lo spazio degli indirizzi logici in un moderno calcolatore è molto grande → la stessa **tabella delle pagine** diventa eccessivamente grande.



suddividere la tabella delle pagine in parti più piccole



algoritmo di **paginazione a due livelli**

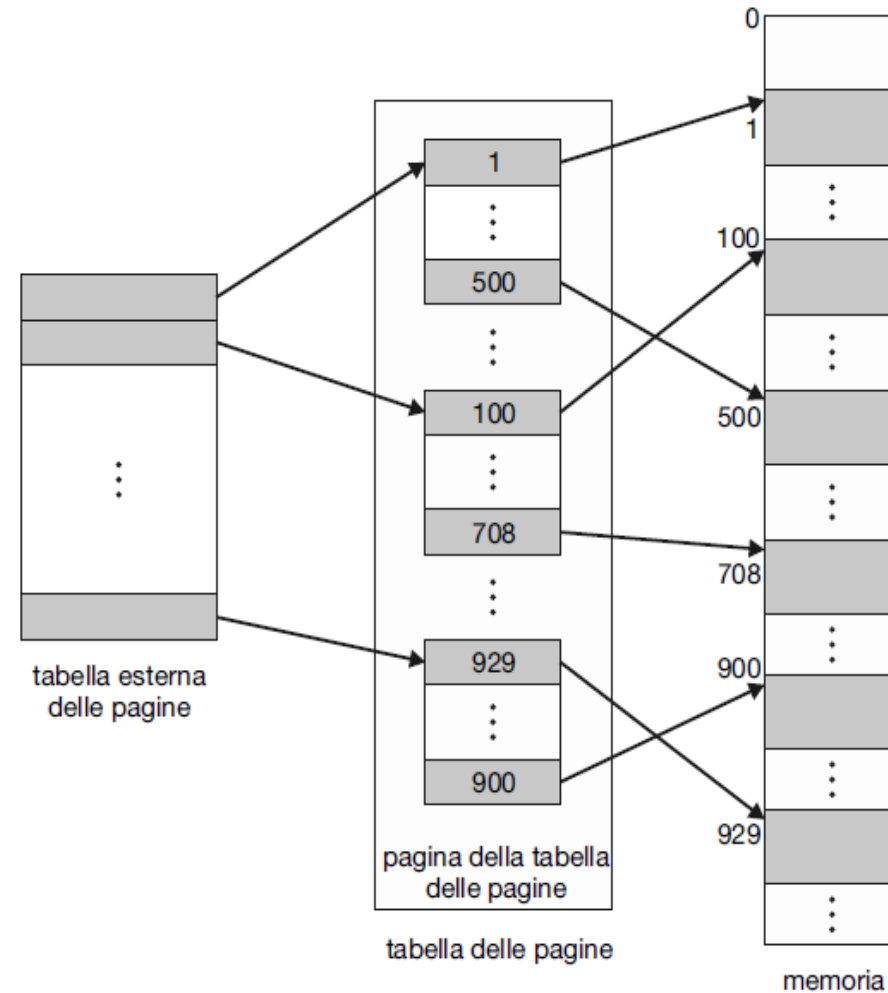


Figura 9.15 Schema di una tabella delle pagine a due livelli.

Associazione diretta

Poiché la traduzione degli indirizzi si svolge dalla tabella esterna delle pagine verso l'interno, questo metodo è anche noto come **tabella delle pagine ad associazione diretta** (*forward-mapped page table*).

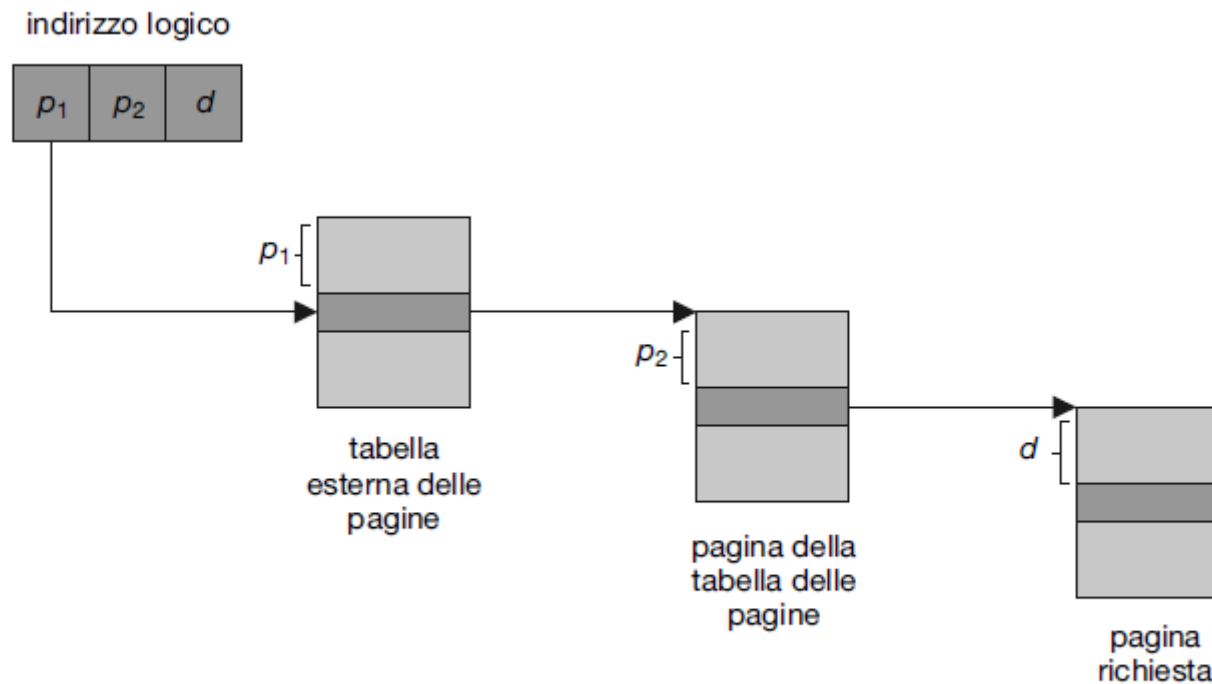


Figura 9.16 Traduzione degli indirizzi per un'architettura a 32 bit con paginazione a due livelli.

Hashing

Tabella delle pagine di tipo hash metodo di gestione molto comune degli spazi d'indirizzi **oltre i 32 bit**

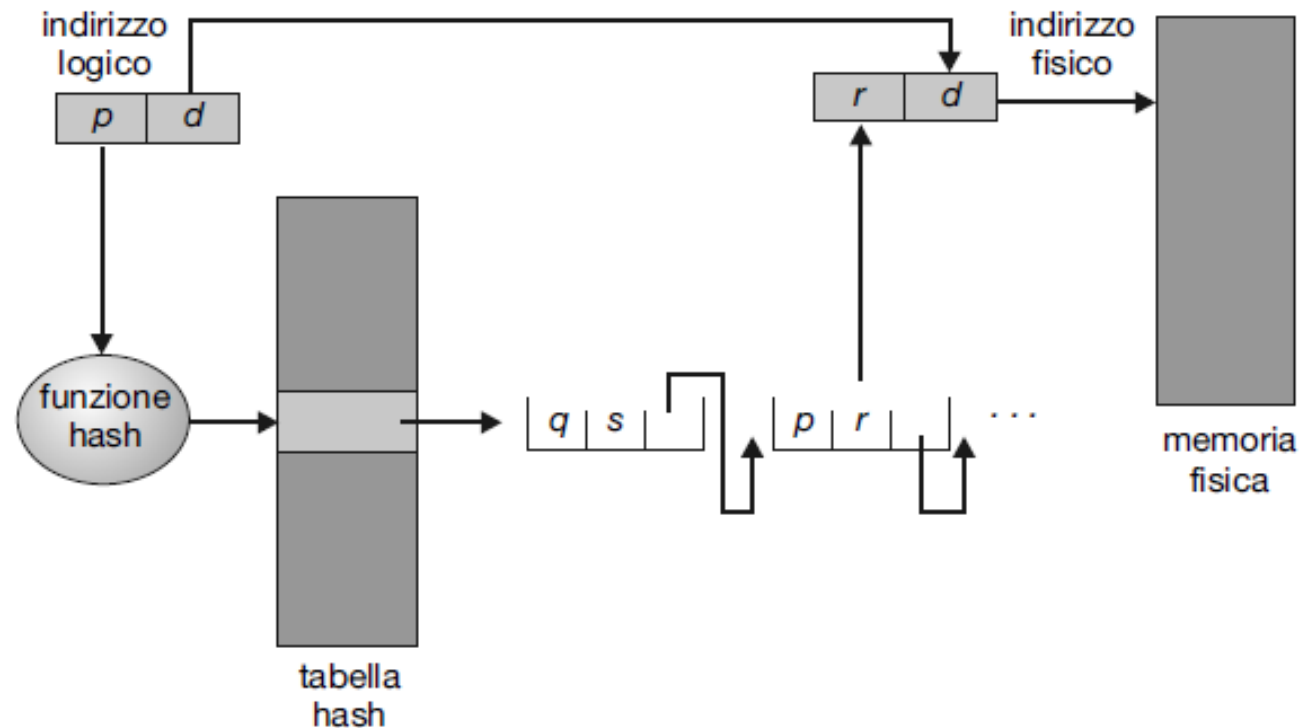


Figura 9.17 Tabella delle pagine di tipo hash.

Tabella delle pagine invertita

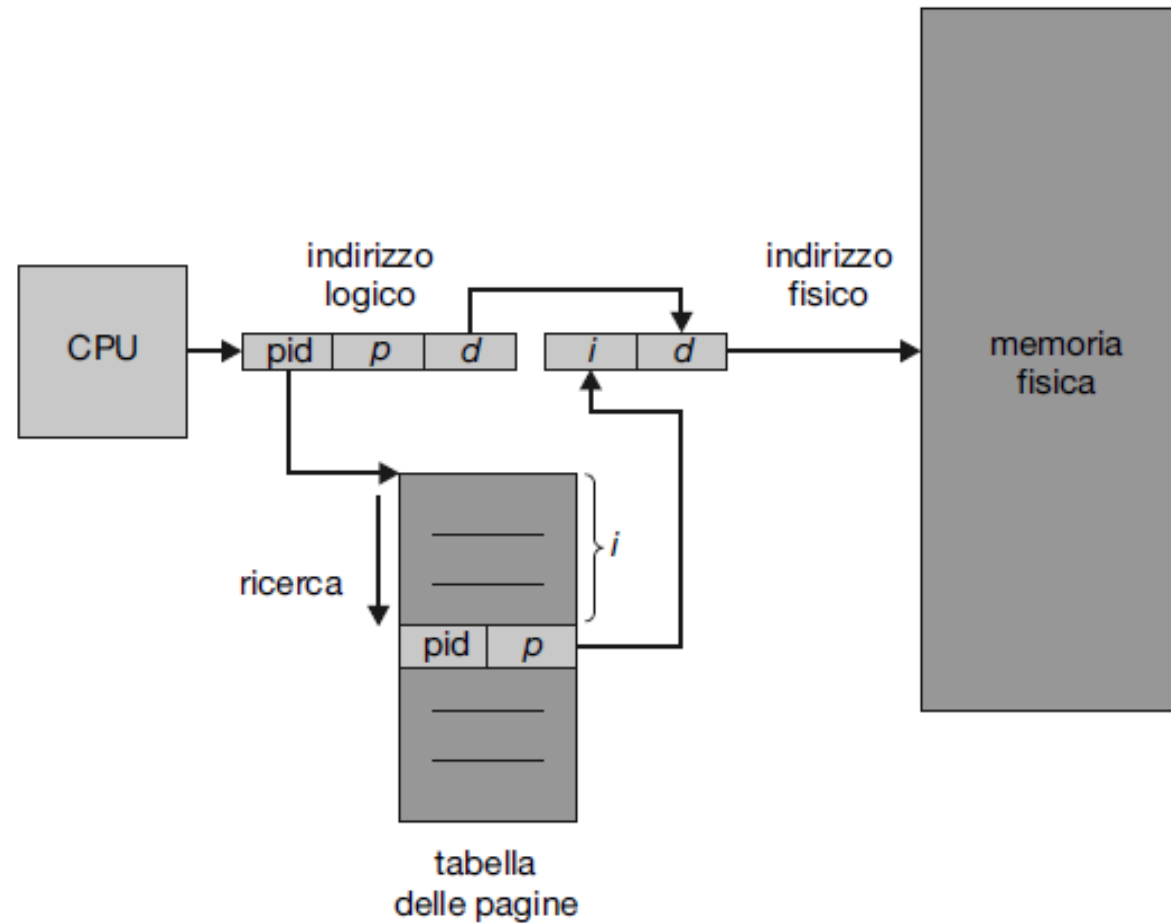


Figura 9.18 Tabella delle pagine invertita.

Swapping

Un processo, o una sua parte, può essere rimosso temporaneamente dalla memoria centrale (mediante un'operazione detta di **swap out, scaricamento**), spostato in una **memoria ausiliaria** (*backing store*) e in seguito riportato in memoria (**swap in, caricamento**) per continuare la sua esecuzione

Swapping (avvicendamento)

Esistono tre procedure di **avvicendamento** o **swapping**

Avvicendamento
standard

Avvicendamento
con paginazione

Avvicendamento
di processi nei
sistemi mobili

Swapping standard

- Lo **swapping standard** tra memoria centrale e memoria secondaria riguarda l'intero processo
- Vantaggio: si può sovrascrivere la memoria fisica

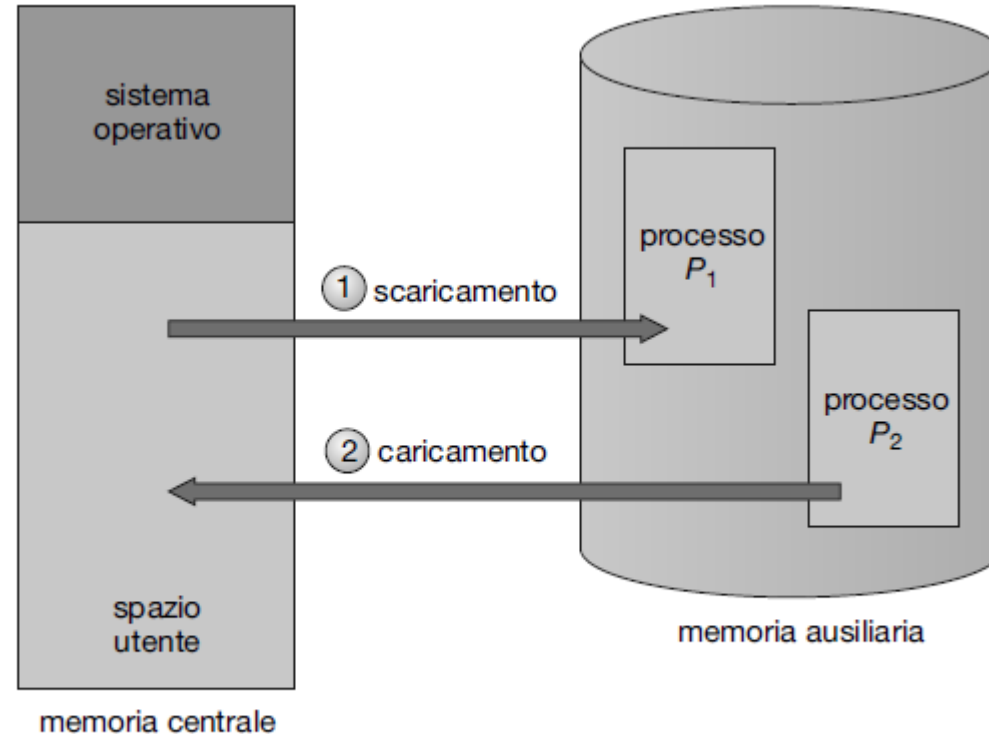


Figura 9.19 Avvicendamento standard di due processi con un disco come memoria ausiliaria.

Swapping con paginazione

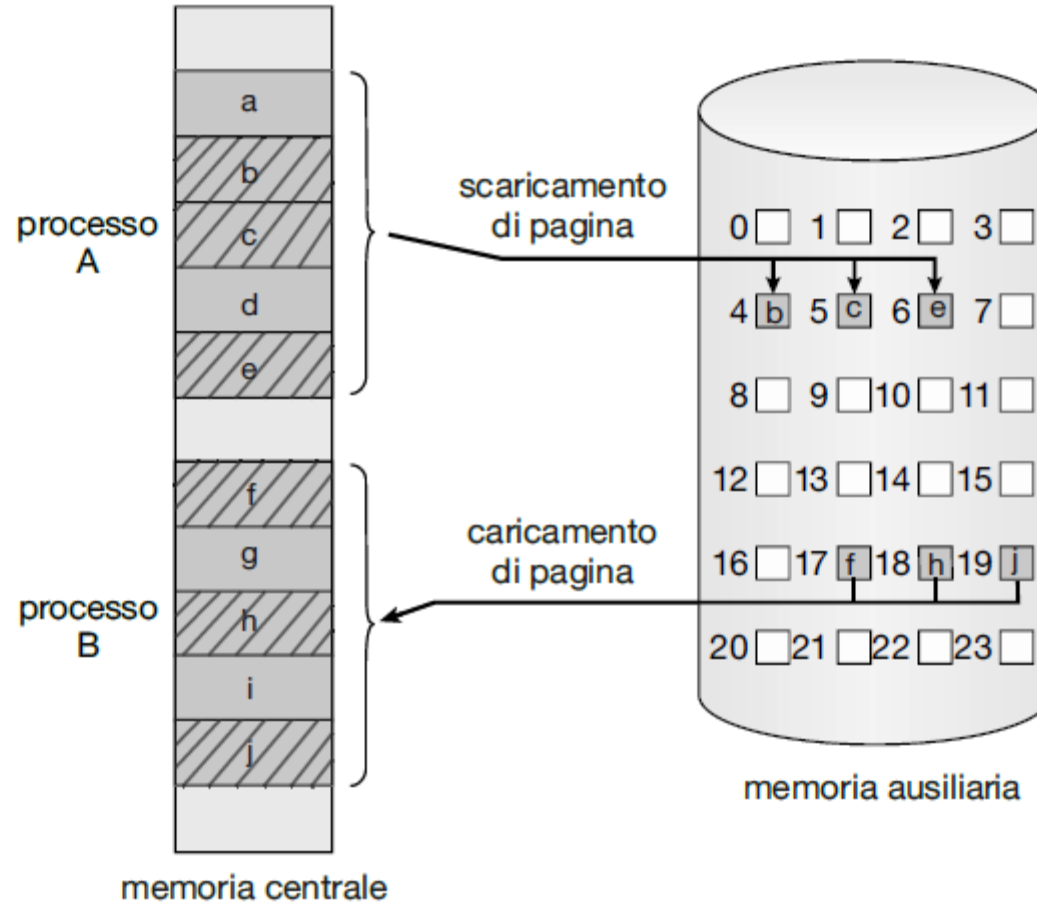


Figura 9.20 Avvicendamento con paginazione.

Swapping nei sistemi mobili

Lo **swapping nei sistemi mobili non viene generalmente supportato:**

- Le memorie flash a disposizione per l'archiviazione secondaria non sono molto capienti
- la memoria flash può tollerare un numero limitato di scritture prima di diventare inaffidabile
- scarsa velocità di trasferimento tra la memoria centrale e la memoria flash

Architettura Intel IA-32

La gestione della memoria nei sistemi IA-32 è suddivisa in due componenti: **segmentazione** e **paginazione**

L'unità di segmentazione e l'unità di paginazione formano insieme l'equivalente dell'unità di gestione della memoria (MMU)

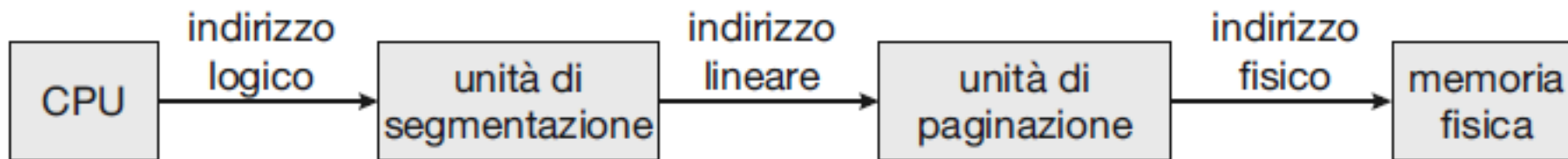


Figura 9.21 Traduzione degli indirizzi logici in indirizzi fisici in IA-32.

Segmentazione in IA-32

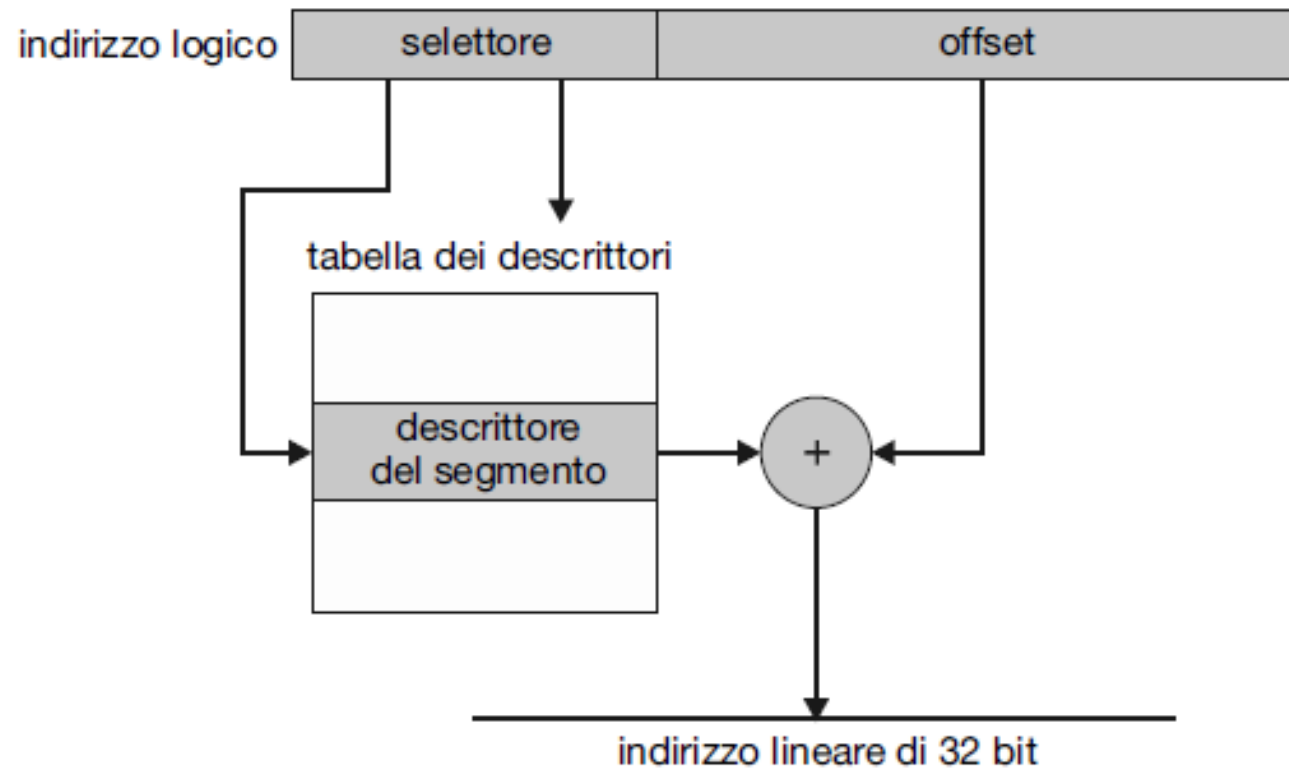


Figura 9.22 Segmentazione in IA-32.

Paginazione in IA-32

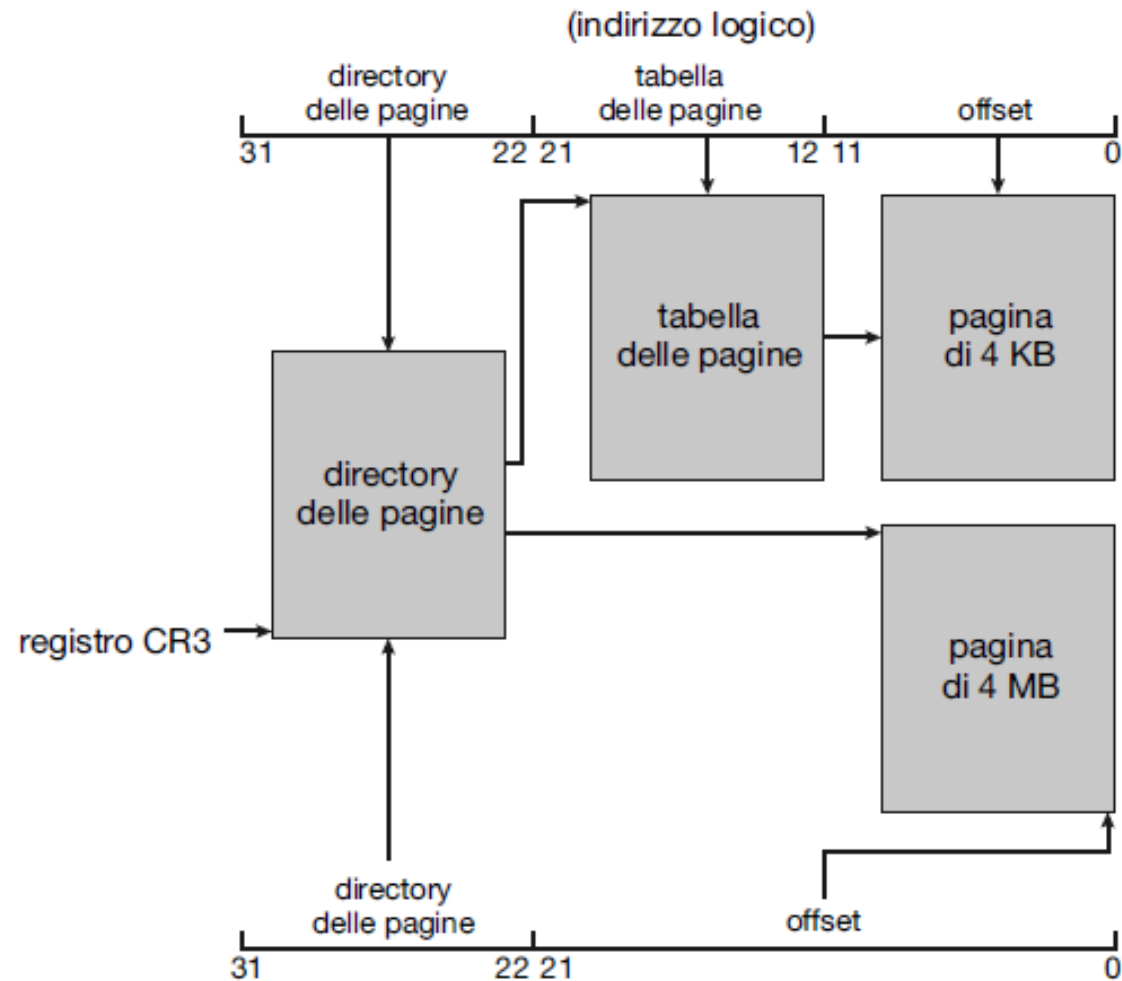


Figura 9.23 Paginazione nell'architettura IA-32.

Estensione PAE

L'**estensione di indirizzo della pagina (PAE, page address extension)**, consente ai processori a 32 bit di accedere a uno spazio di indirizzamento fisico più grande di 4 GB.

Con **PAE** è stata inoltre aumentata la dimensione degli elementi della directory delle pagine e della tabella delle pagine, che passa da 32 a 64 bit, permettendo di estendere l'indirizzo di base delle tabelle delle pagine e dei frame da 20 a 24 bit.

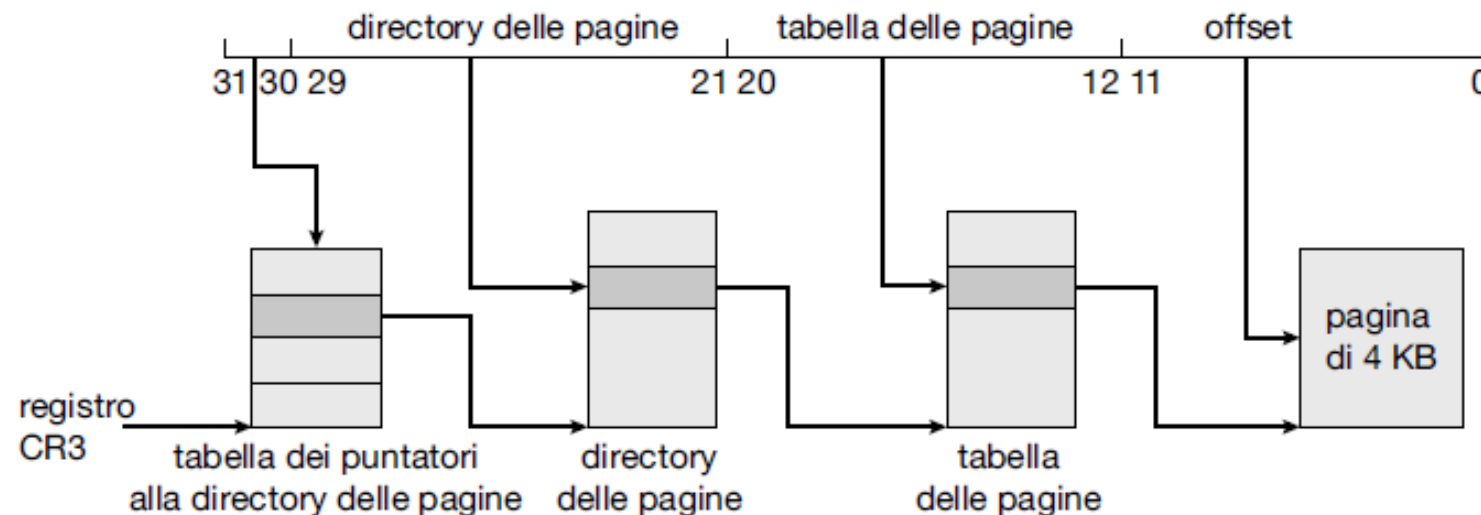


Figura 9.24 Estensione degli indirizzi di pagina.

Architettura x86-64

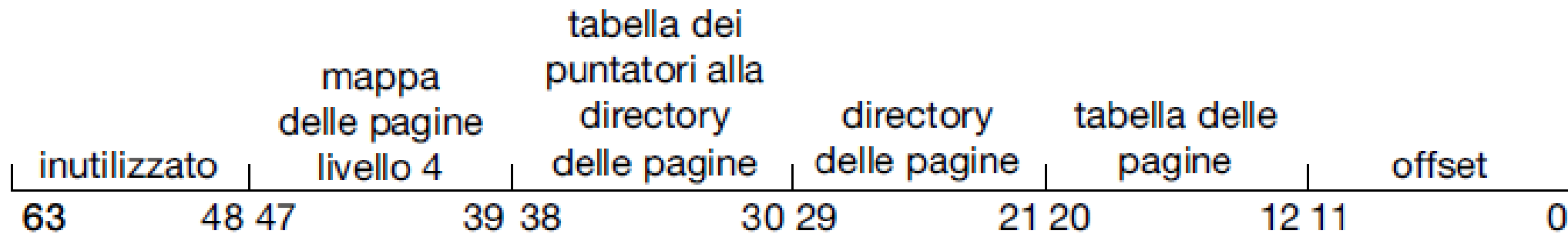


Figura 9.25 Indirizzo lineare in x86-64.

Architettura ARMv8

ARMv8 supporta tre diverse **granularità di traduzione** (*translation granule*):

4 KB, 16 KB e 64 KB.

La Figura 9.26 illustra la struttura degli indirizzi ARMv8 con una **granularità di 4 KB** con un massimo di quattro livelli di paginazione.

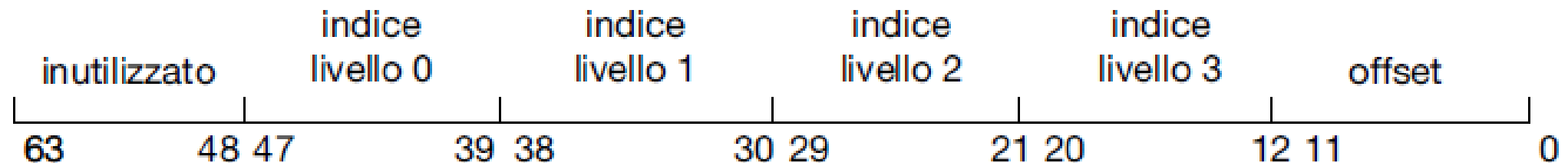


Figura 9.26 Indirizzo in ARM, con granularità di 4 KB.

Paginazione ARMv8

registro TTBR → è il registro di base della tabella di traduzione e punta alla **tabella del livello 0** per il thread corrente

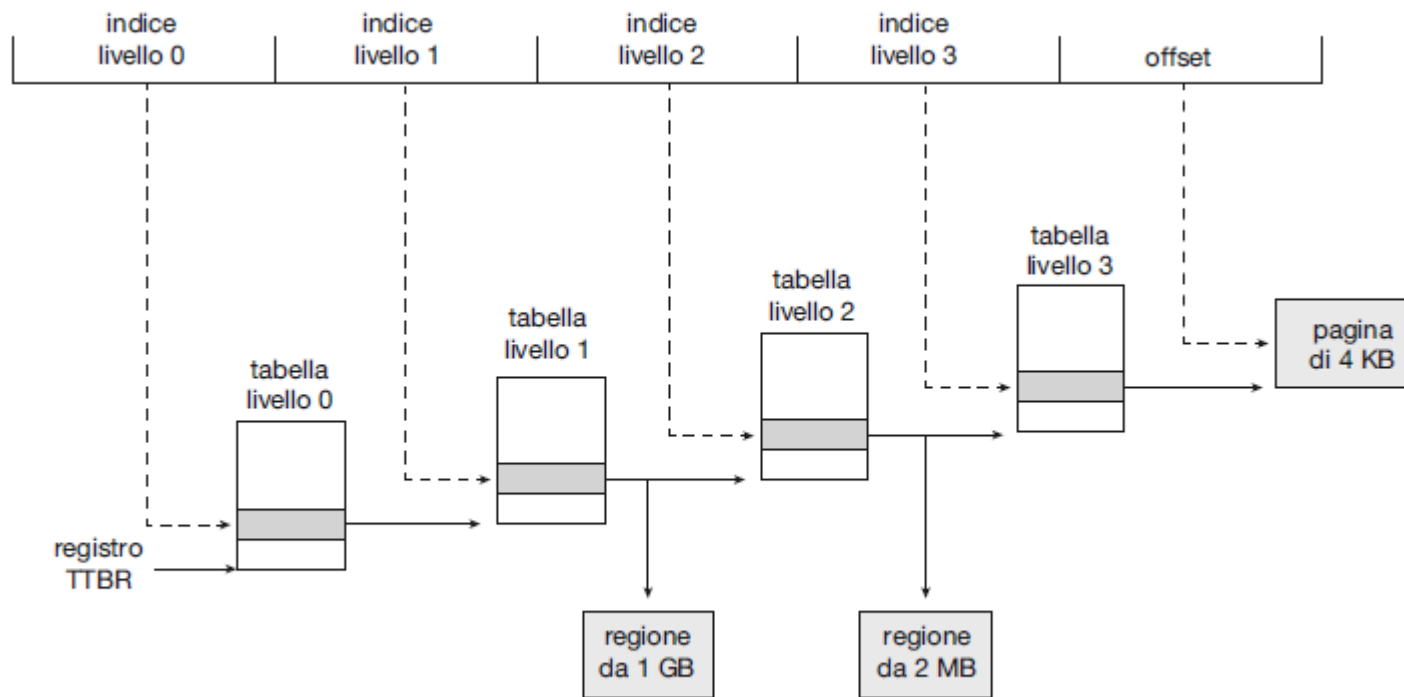
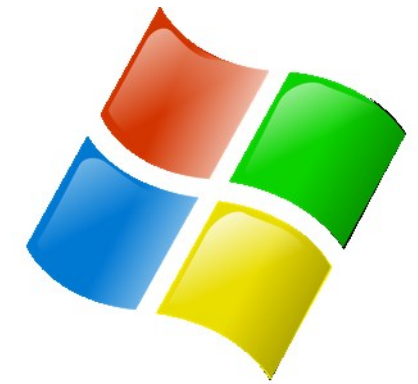


Figura 9.27 Paginazione gerarchica su 4 livelli in ARM.



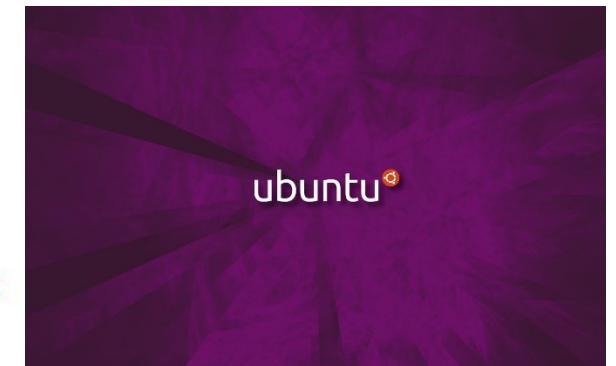
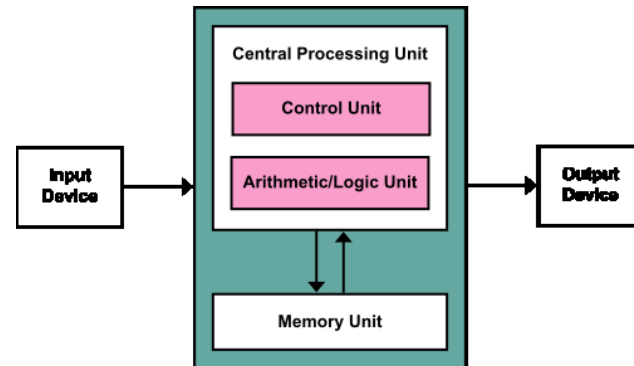
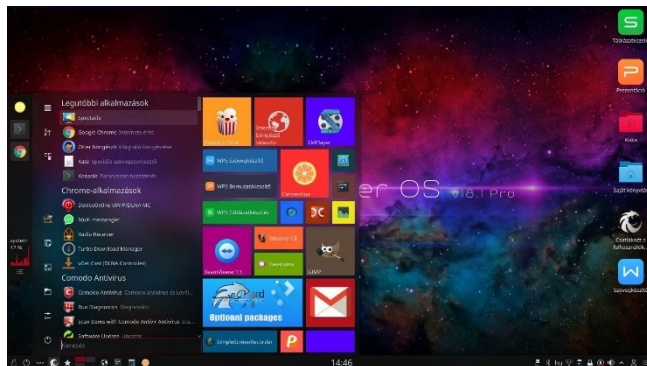
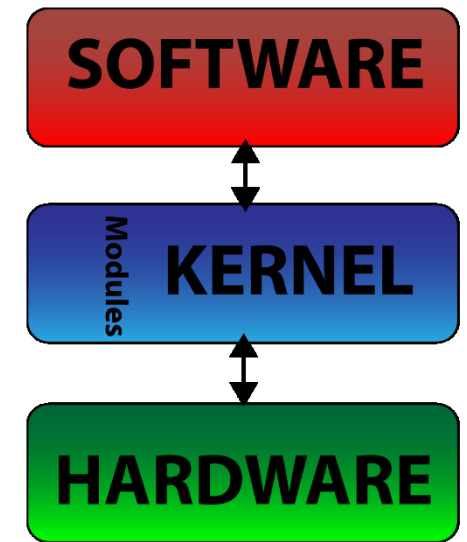
**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

*Corso di Sistemi Operativi
A.A. 2019/20*



Memoria centrale

Docente:
Domenico Daniele
Bloisi



Novembre 2019