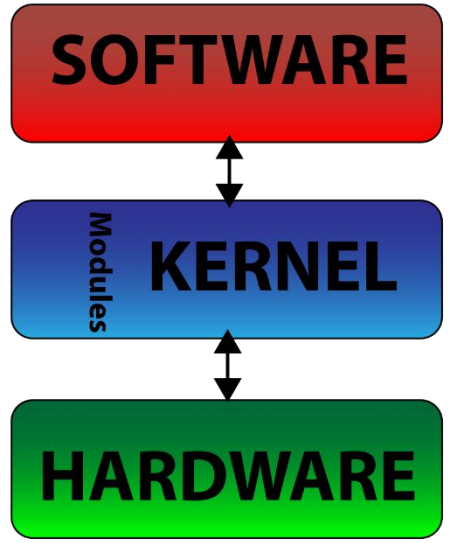
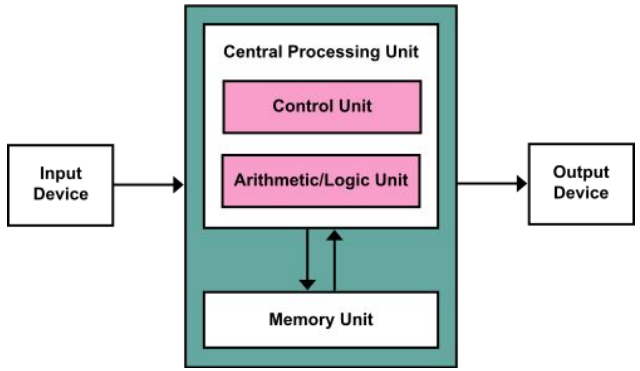
UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

Corso di Sistemi Operativi

Paginazione



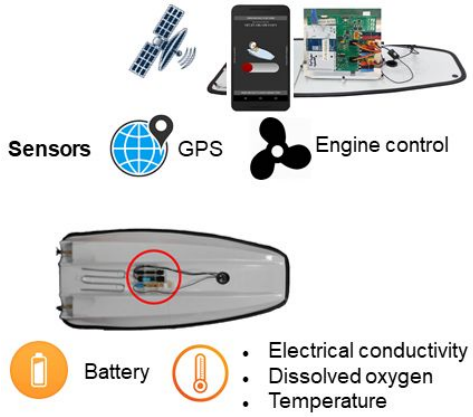
Docente:
Domenico Daniele Bloisi

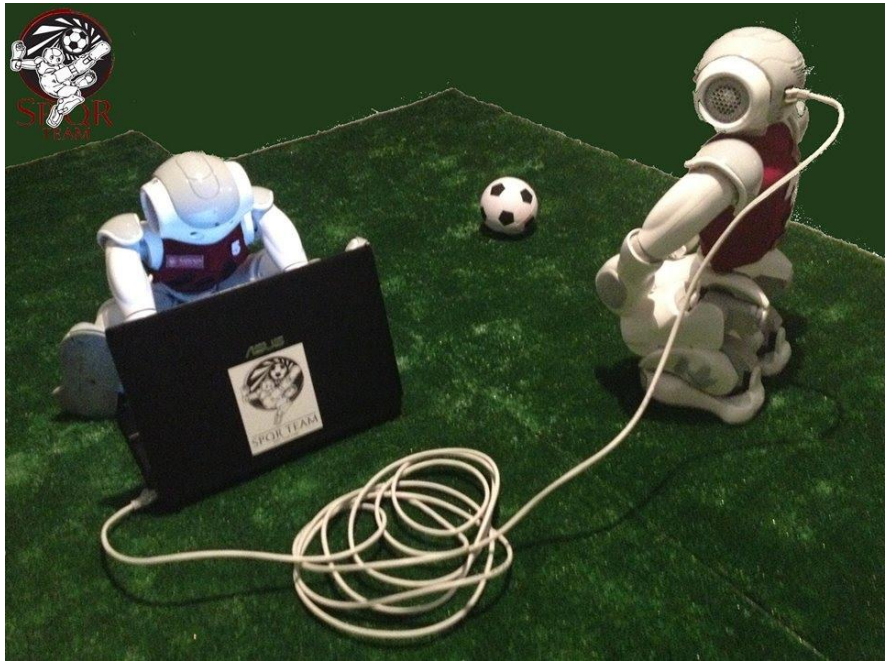


Domenico Daniele Bloisi

- Professore Associato
  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



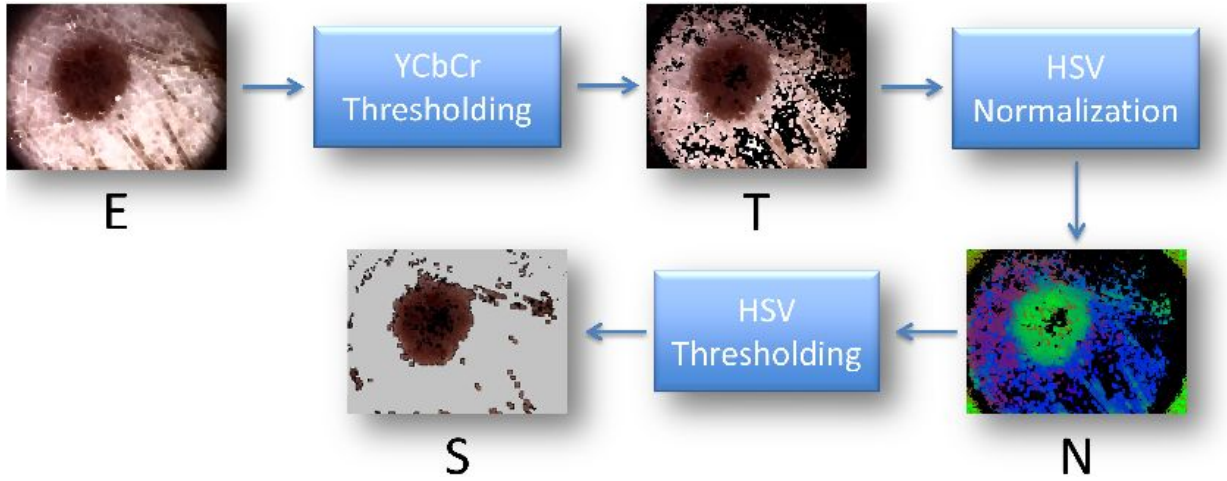


Interessi di ricerca

- Intelligent surveillance
- Robot vision
- Medical image analysis

https://youtu.be/2KHNZX7UIWQ

https://youtu.be/9a70Ucgbi_U



UNIBAS Wolves https://sites.google.com/unibas.it/wolves

- UNIBAS WOLVES is the robot soccer team of the University of Basilicata. Established in 2019, it is focussed on developing software for NAO soccer robots participating in RoboCup competitions.

- UNIBAS WOLVES team is twinned with SPQR Team at Sapienza University of Rome

https://youtu.be/ji0OmkaWh20

Informazioni sul corso

- Home page del corso:
  http://web.unibas.it/bloisi/corsi/sistemi-operativi.html
- Docente: Domenico Daniele Bloisi
- Periodo: I semestre ottobre 2022 – gennaio 2023

  - Lunedì dalle 15:00 alle 17:00 (Aula Leonardo)
  - Martedì dalle 08:30 alle 10:30 (Aula 1)

Ricevimento

- In presenza, durante il periodo delle lezioni:
  Lunedì dalle 17:00 alle 18:00
  presso Edificio 3D, II piano, stanza 15
  Si invitano gli studenti a controllare regolarmente la bacheca degli avvisi per eventuali variazioni
- Tramite google Meet e al di fuori del periodo delle lezioni:
  da concordare con il docente tramite email

Per prenotare un appuntamento inviare una email a
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

Paginazione

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

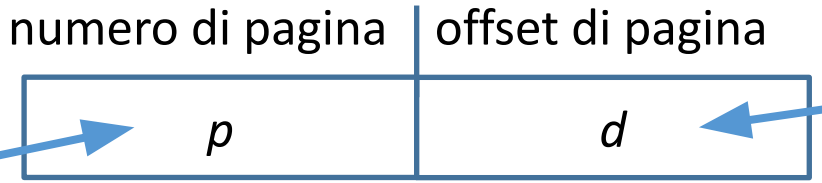
La paginazione è implementata sfruttando la cooperazione tra sistema operativo e hardware del computer

Pagine vs. Frame

Pagine
suddivisione della
memoria logica in blocchi di
pari dimensione

Frame
suddivisione della
memoria fisica in blocchi
di dimensione fissa

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
- l'offset è la posizione nel frame a cui viene fatto riferimento

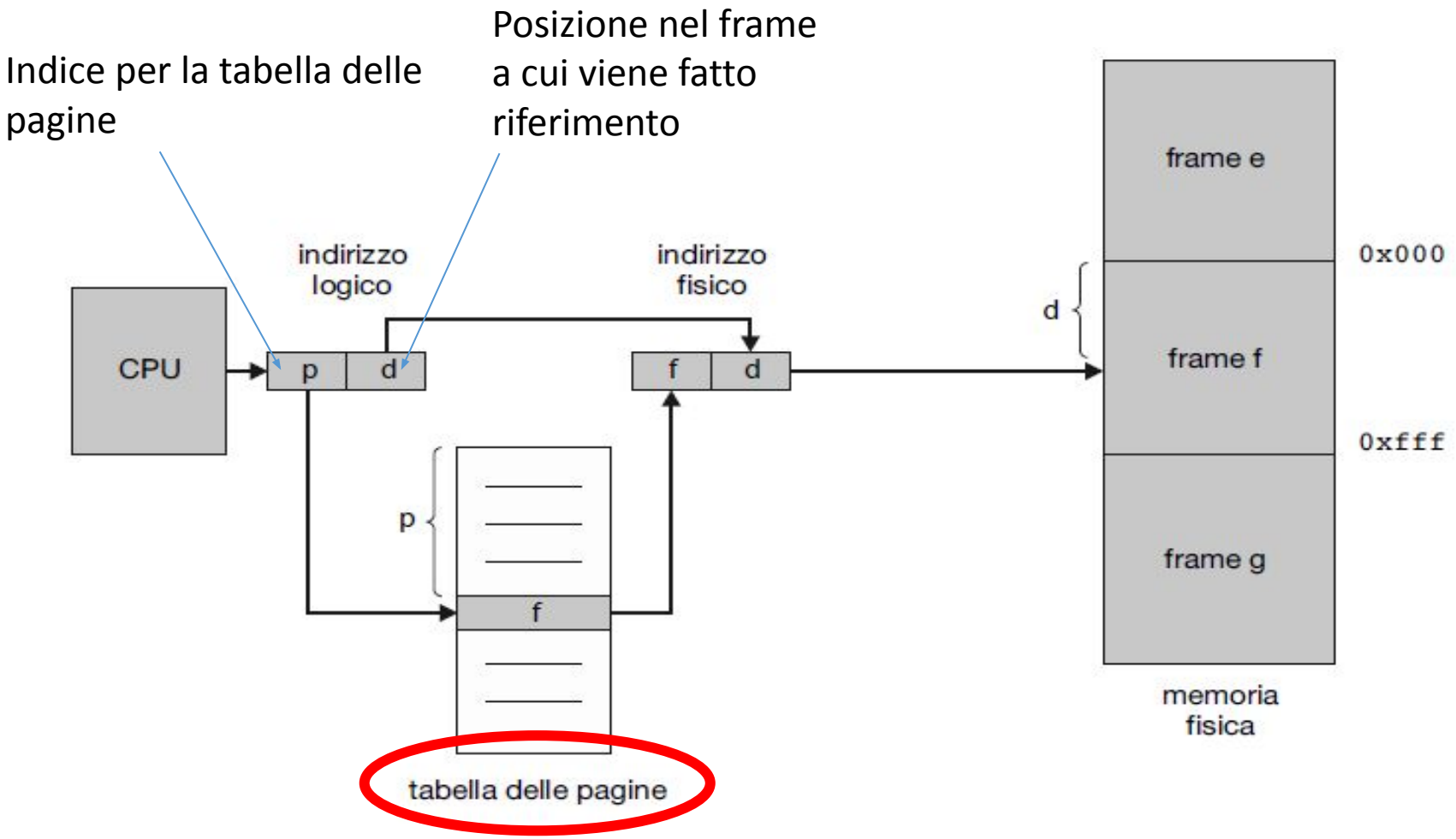


Figura 9.8 Hardware di paginazione.

Modello di paginazione della memoria

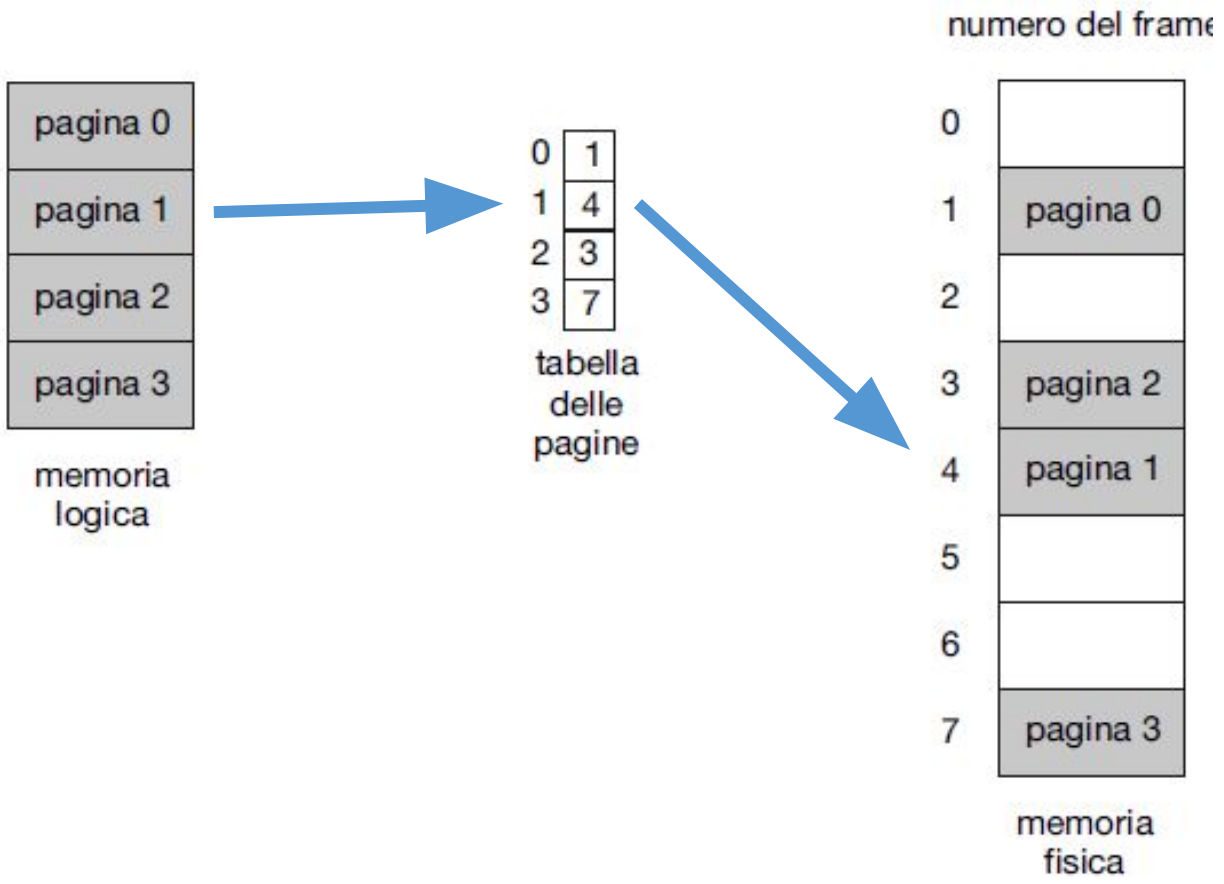


Figura 9.9 Modello di paginazione di memoria logica e memoria fisica.

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*

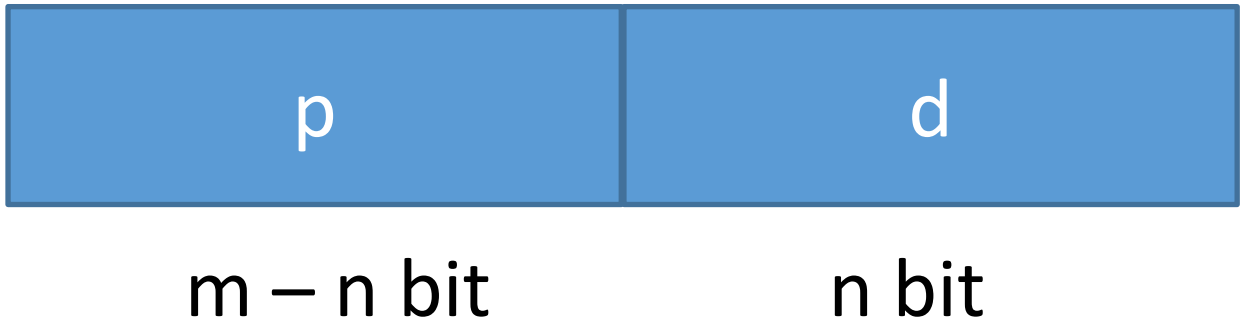
indirizzo fisico ottenuto 

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?

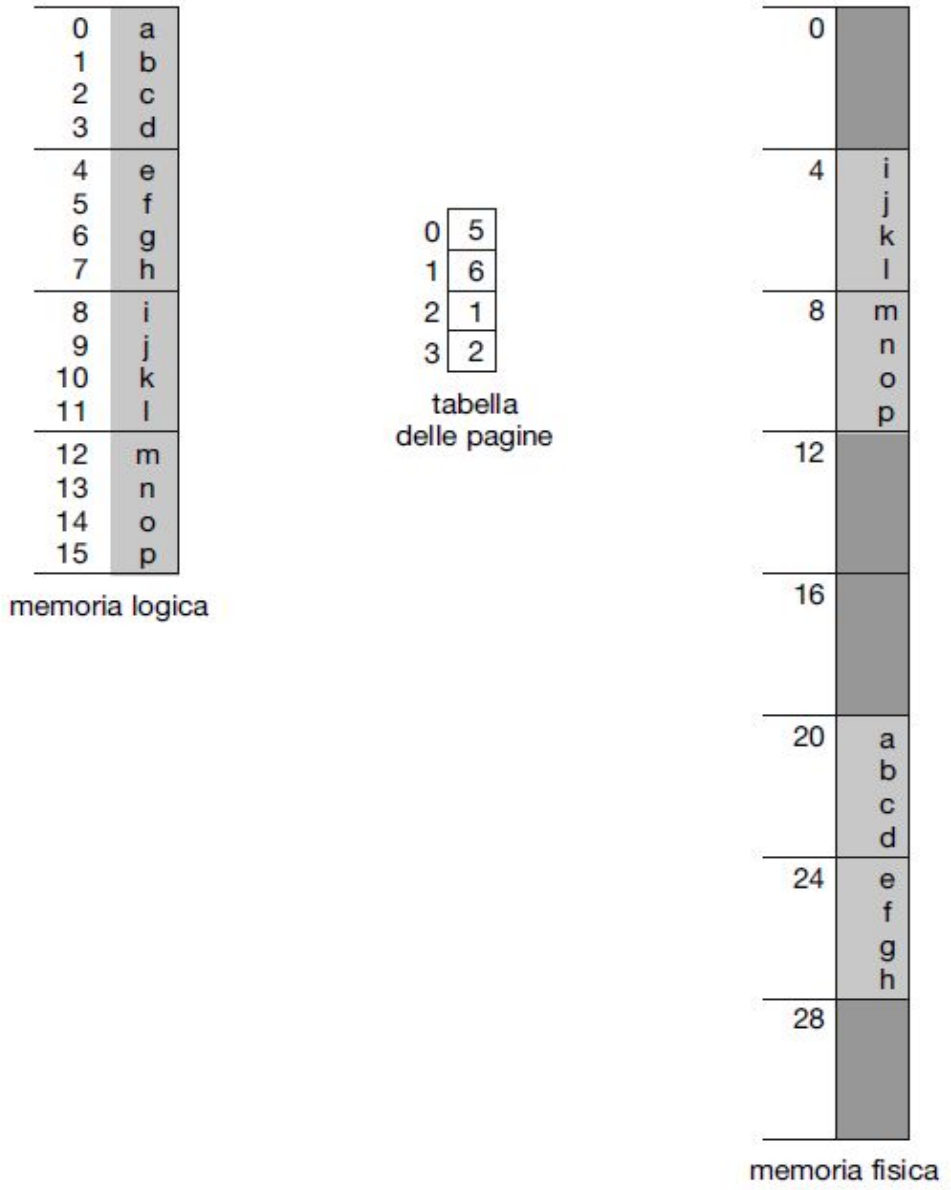


Figura 9.10 Esempio di paginazione per una memoria di 32 byte con pagine di 4 byte.

Soluzione

Che indirizzo fisico corrisponde all'indirizzo logico 3?

indirizzo logico 3

pagina	offset
0	3

indirizzo fisico 23

frame	offset
5	3

$(5 \times 2^2) + 3 = 23$

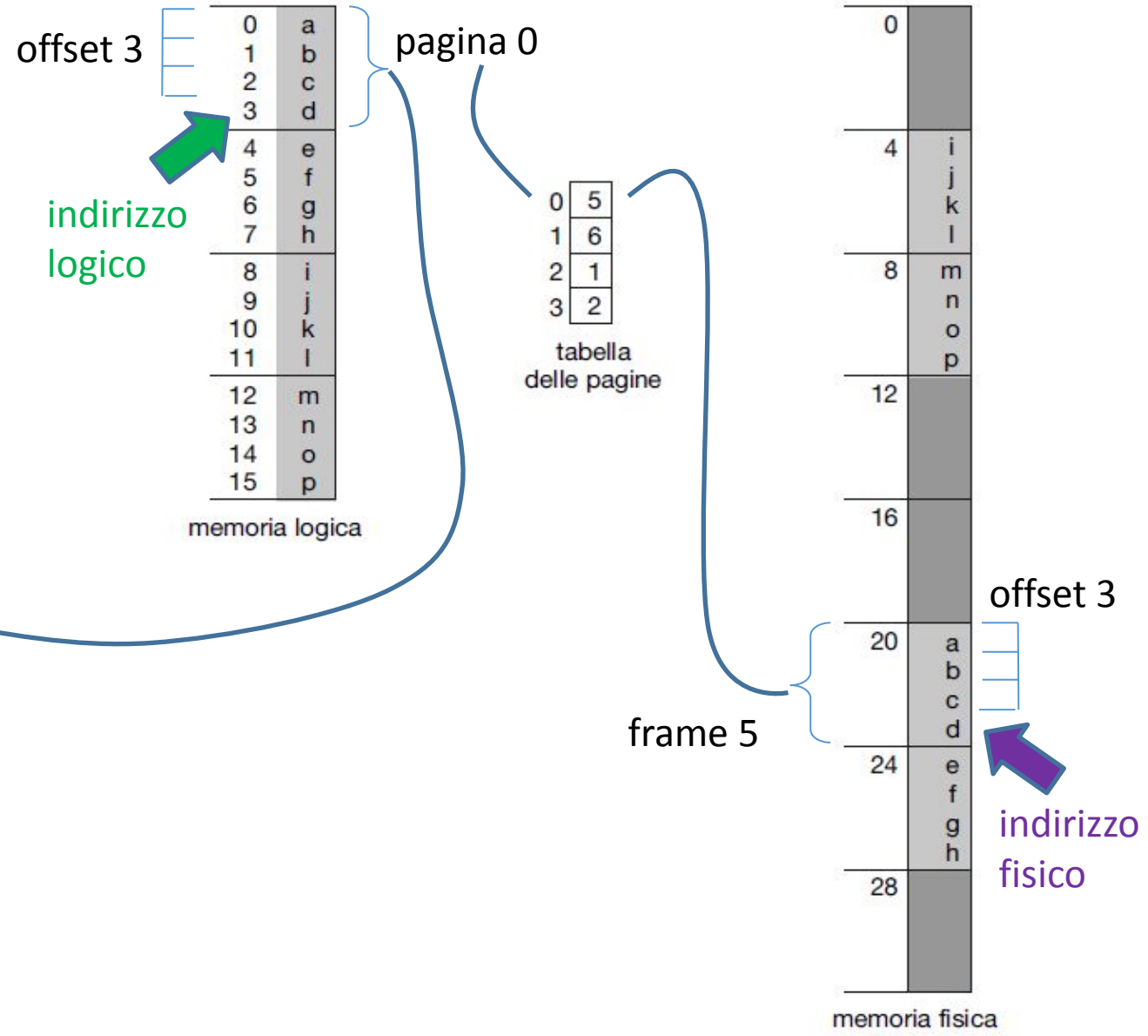


Figura 9.10 Esempio di paginazione per una memoria di 32 byte con pagine di 4 byte.

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

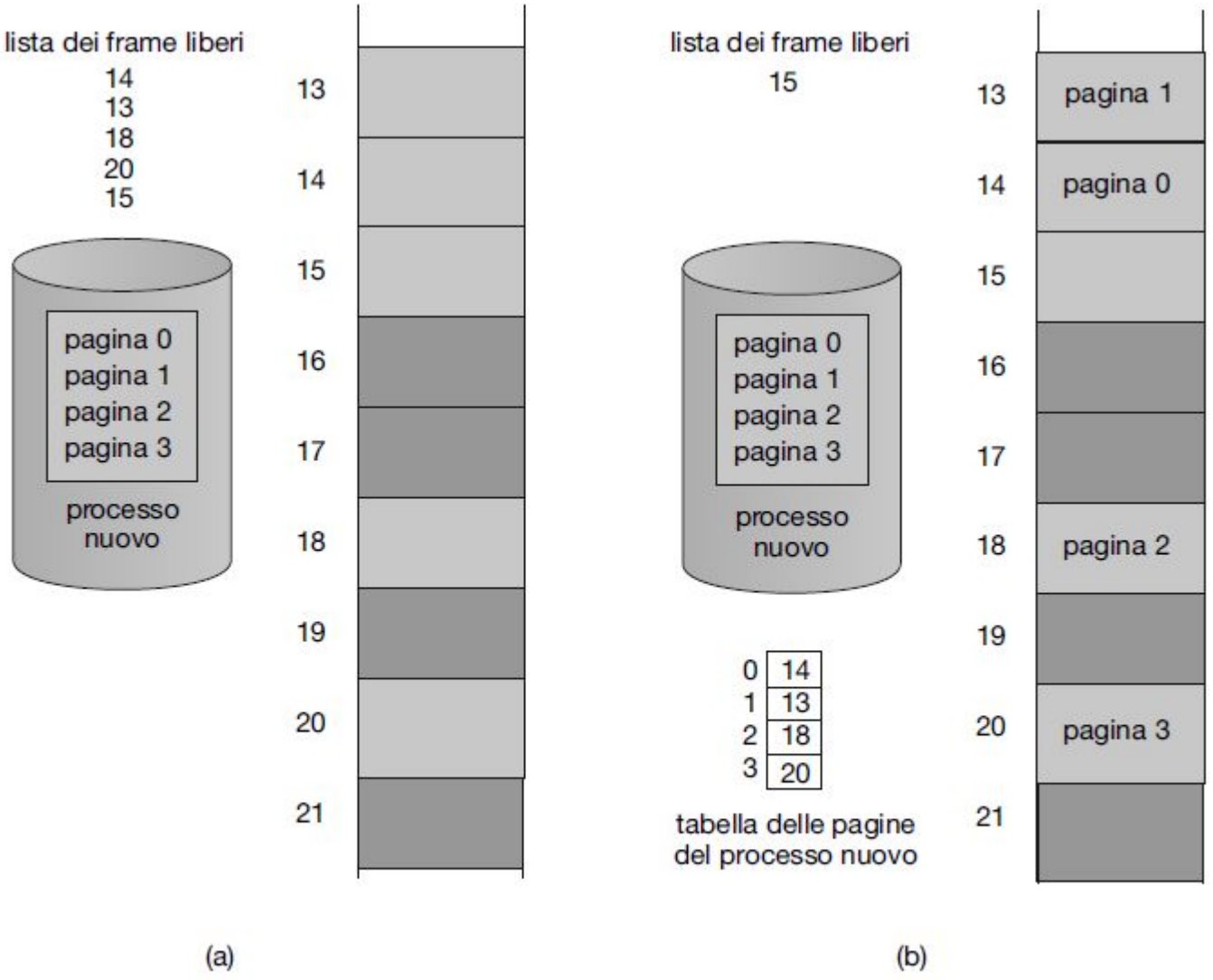


Figura 9.11 Frame liberi; (a) prima e (b) dopo l'allocazione.

Ogni processo ha la sua tabella delle pagine

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

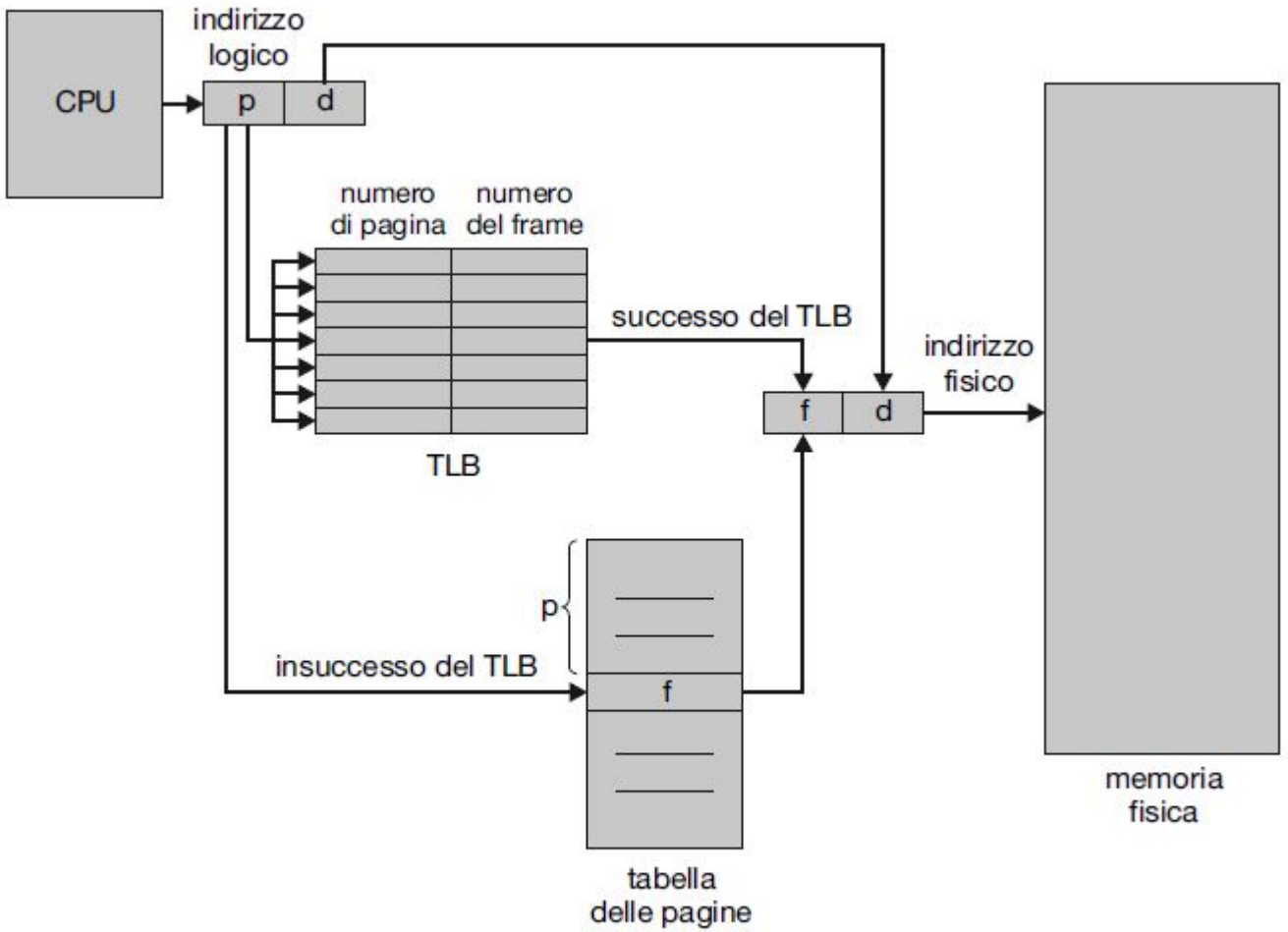


Figura 9.12 Hardware di paginazione con TLB.

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

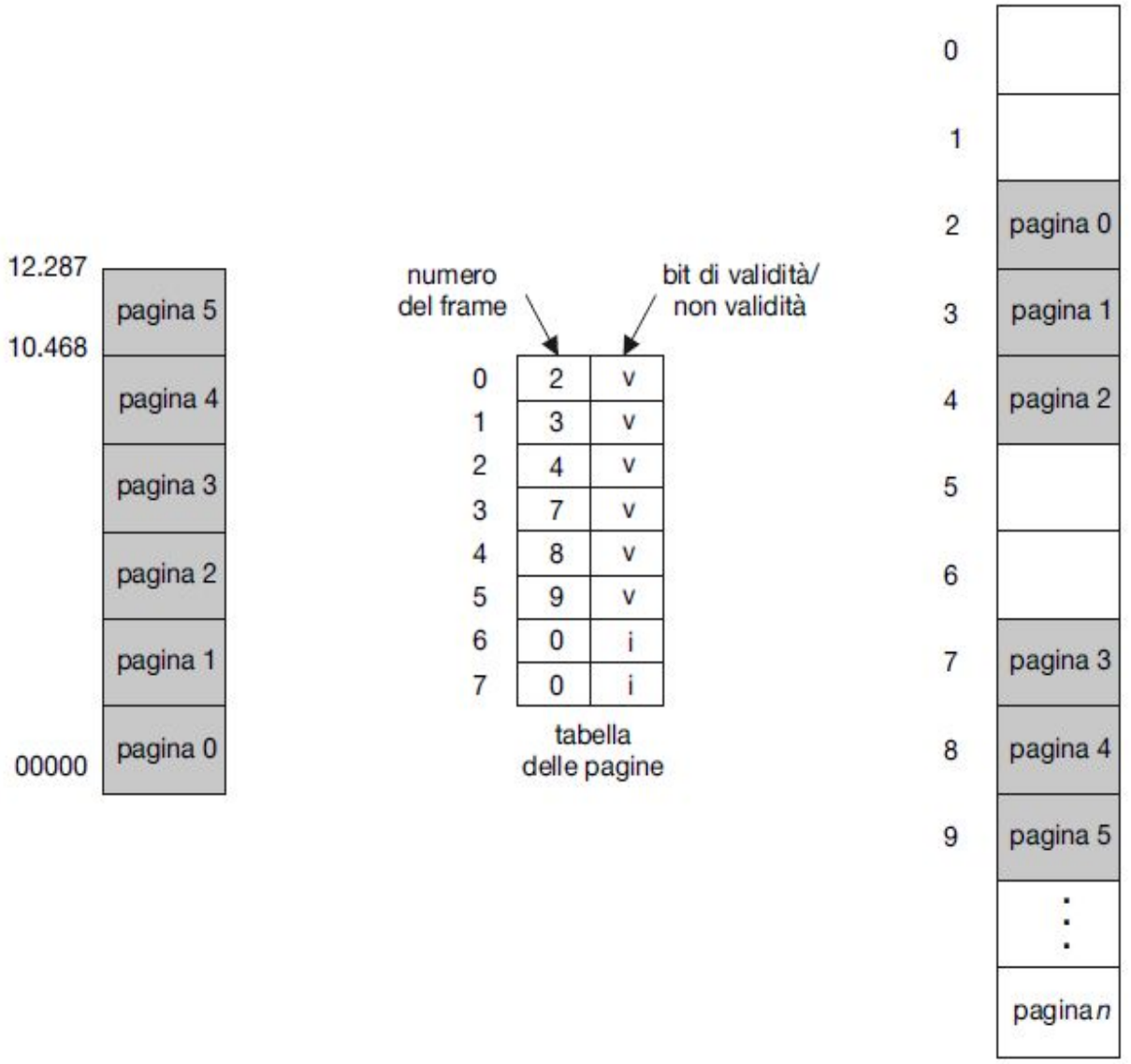


Figura 9.13 Bit di validità (v) o non validità (i) in una tabella delle pagine.

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.

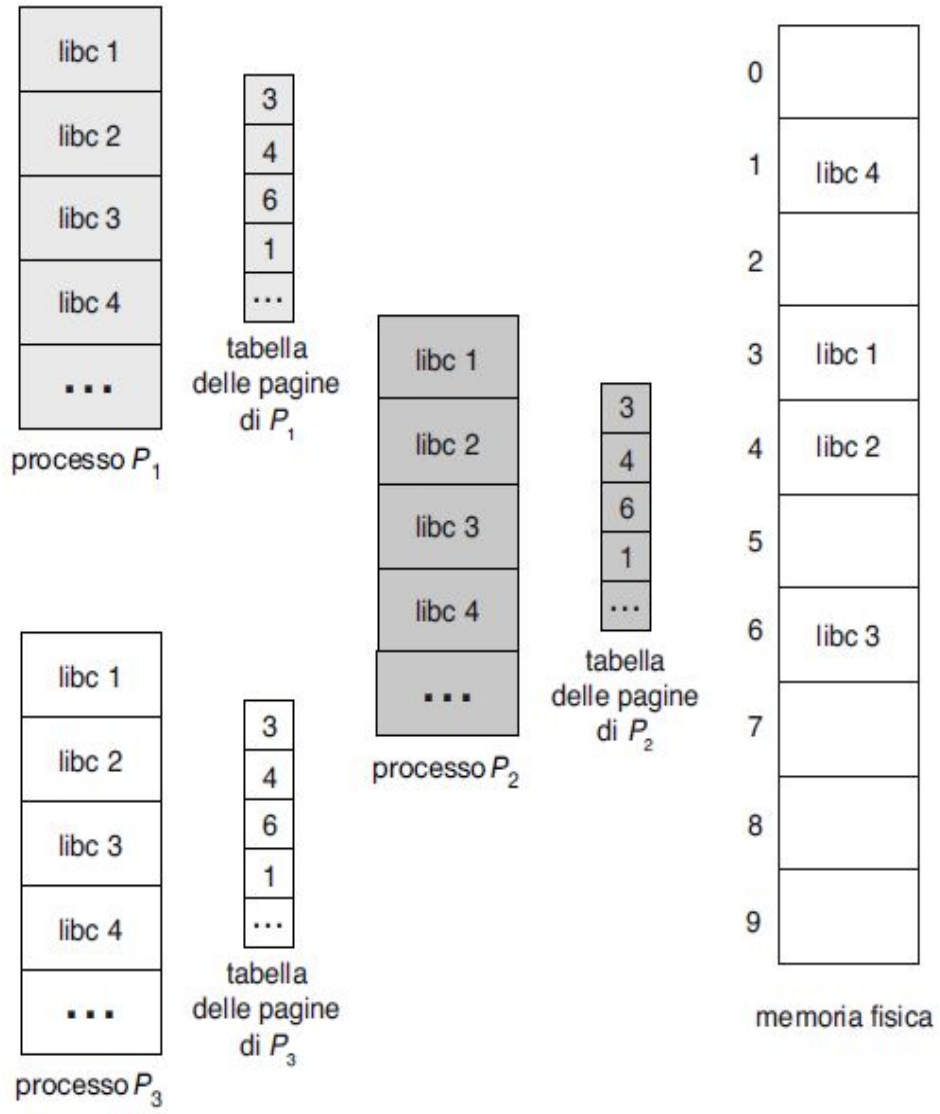


Figura 9.14 Condivisione della libreria standard del C in un ambiente paginato.

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**

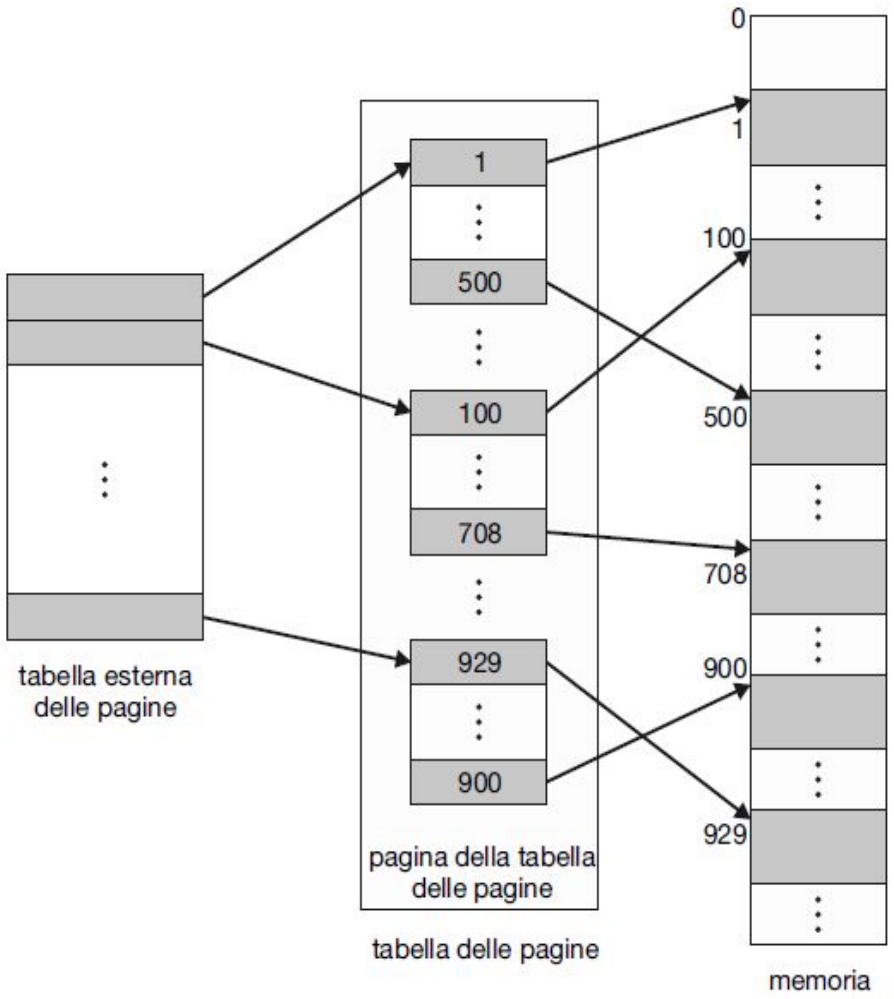


Figura 9.15 Schema di una tabella delle pagine a due livelli.

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*).

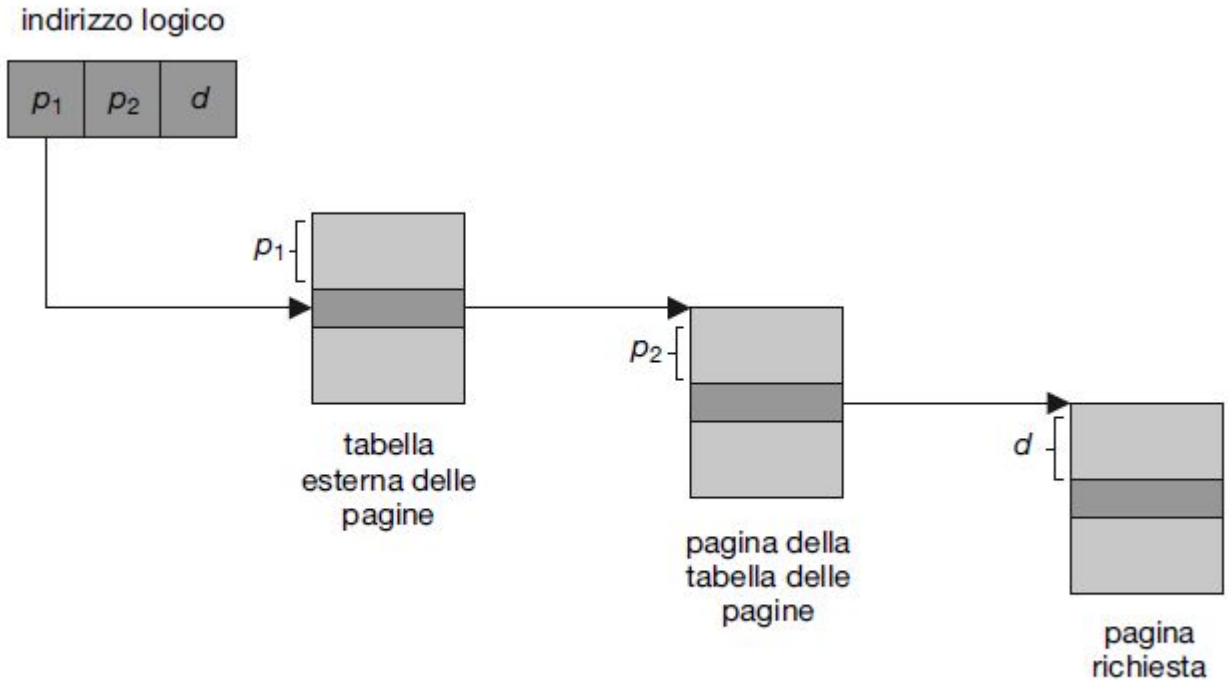


Figura 9.16 Traduzione degli indirizzi per un'architettura a 32 bit con paginazione a due livelli.

Hashing

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

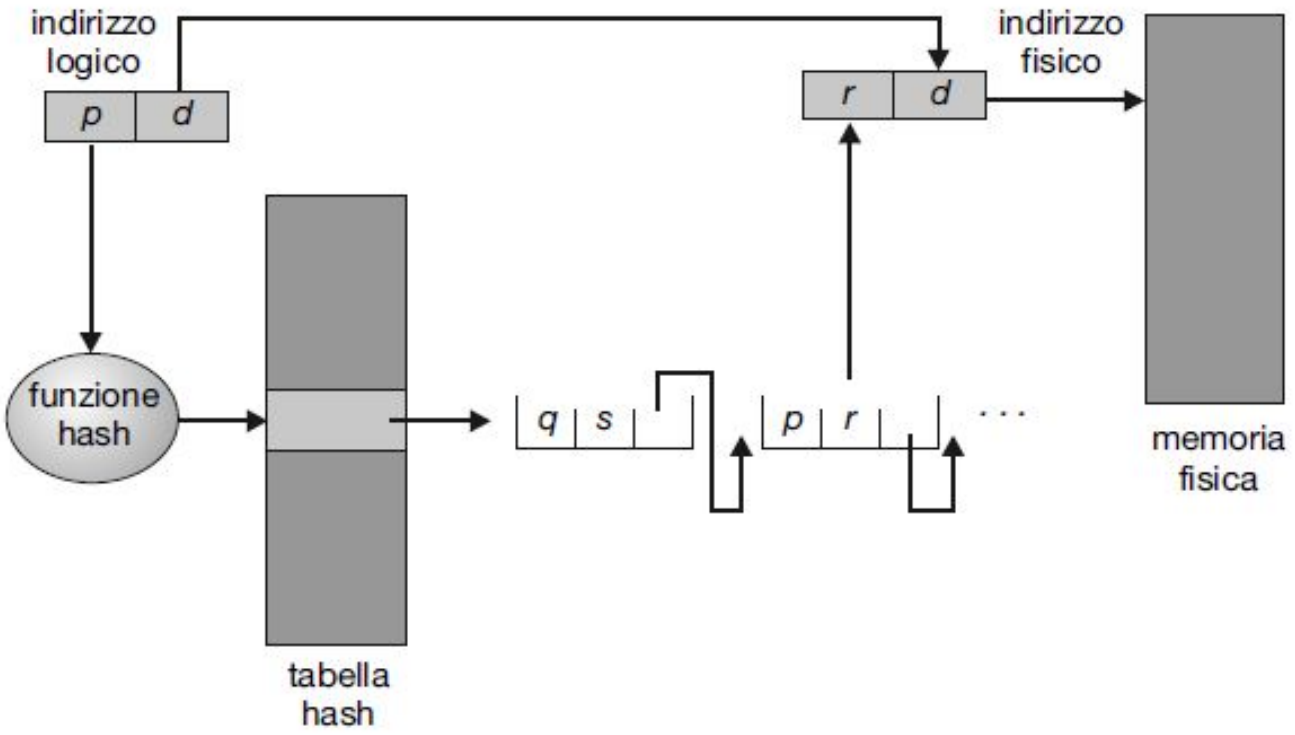


Figura 9.17 Tabella delle pagine di tipo hash.

Tabella delle pagine invertita

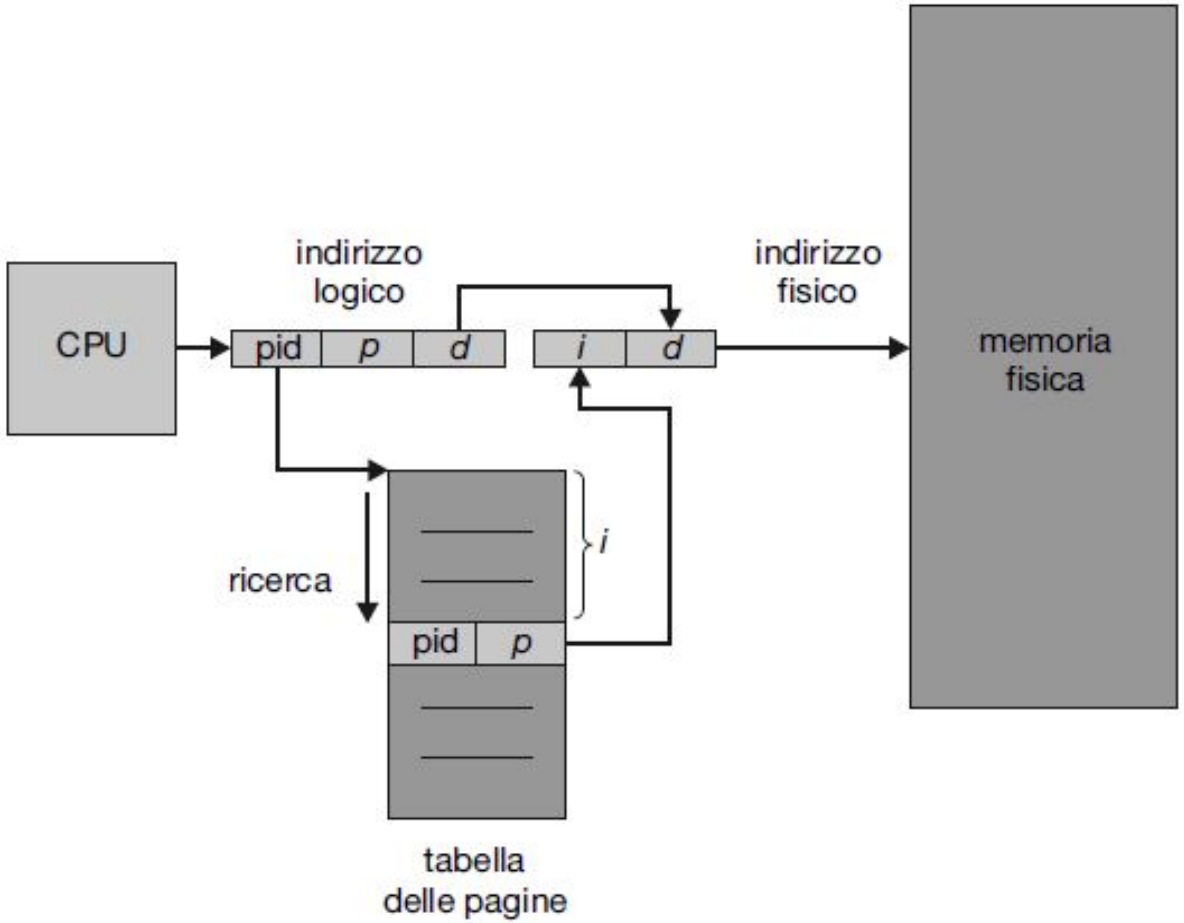


Figura 9.18 Tabella delle pagine invertita.

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

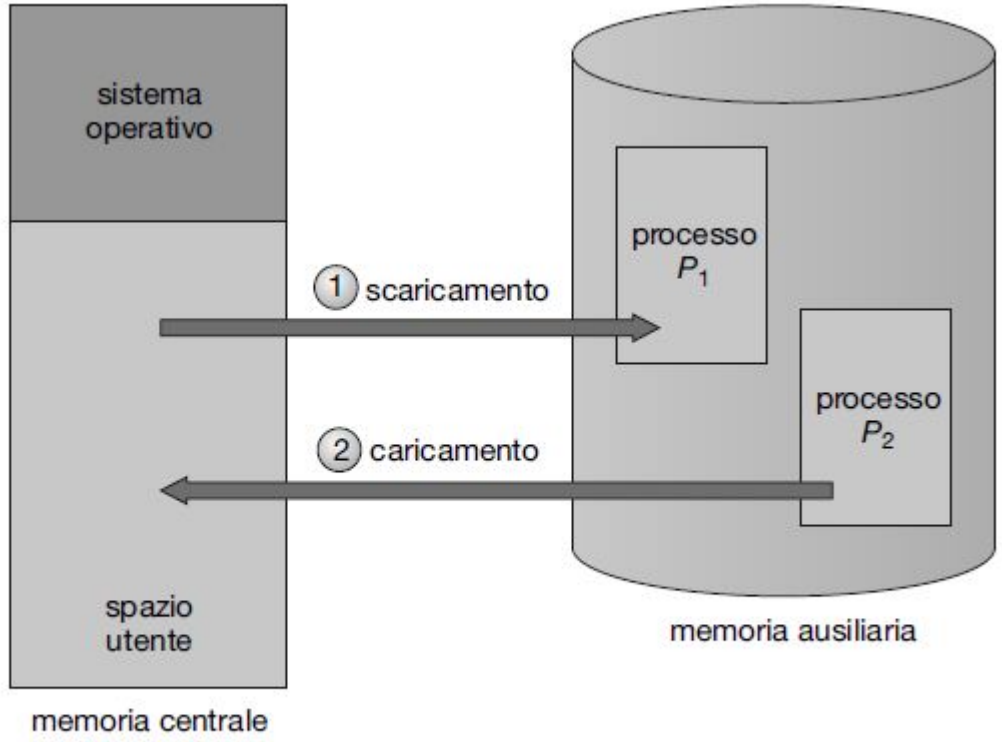


Figura 9.19 Avvicendamento standard di due processi con un disco come memoria ausiliaria.

Swapping con paginazione

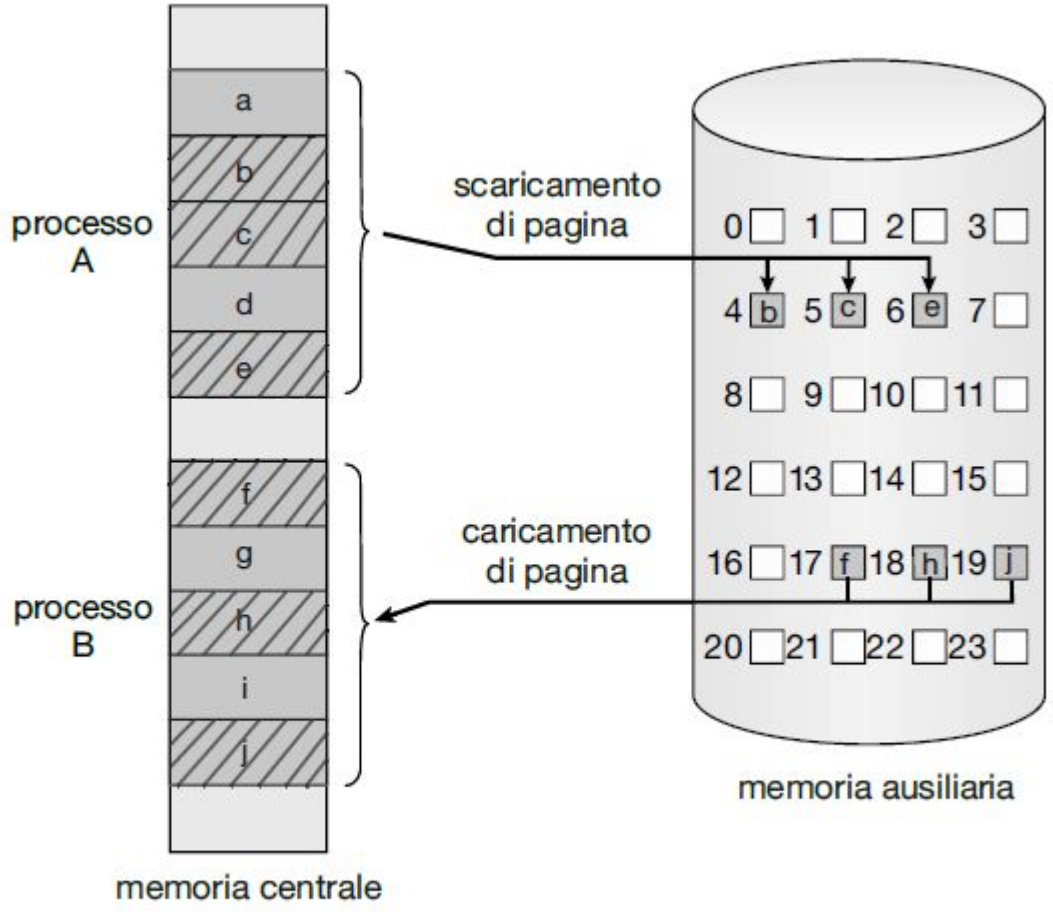


Figura 9.20 Avvicendamento con paginazione.

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)

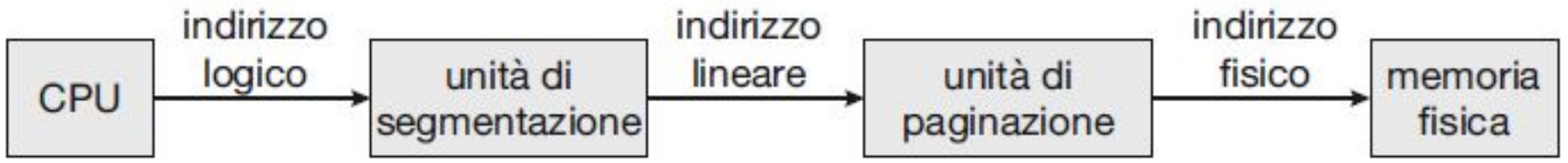


Figura 9.21 Traduzione degli indirizzi logici in indirizzi fisici in IA-32.

Segmentazione in IA-32

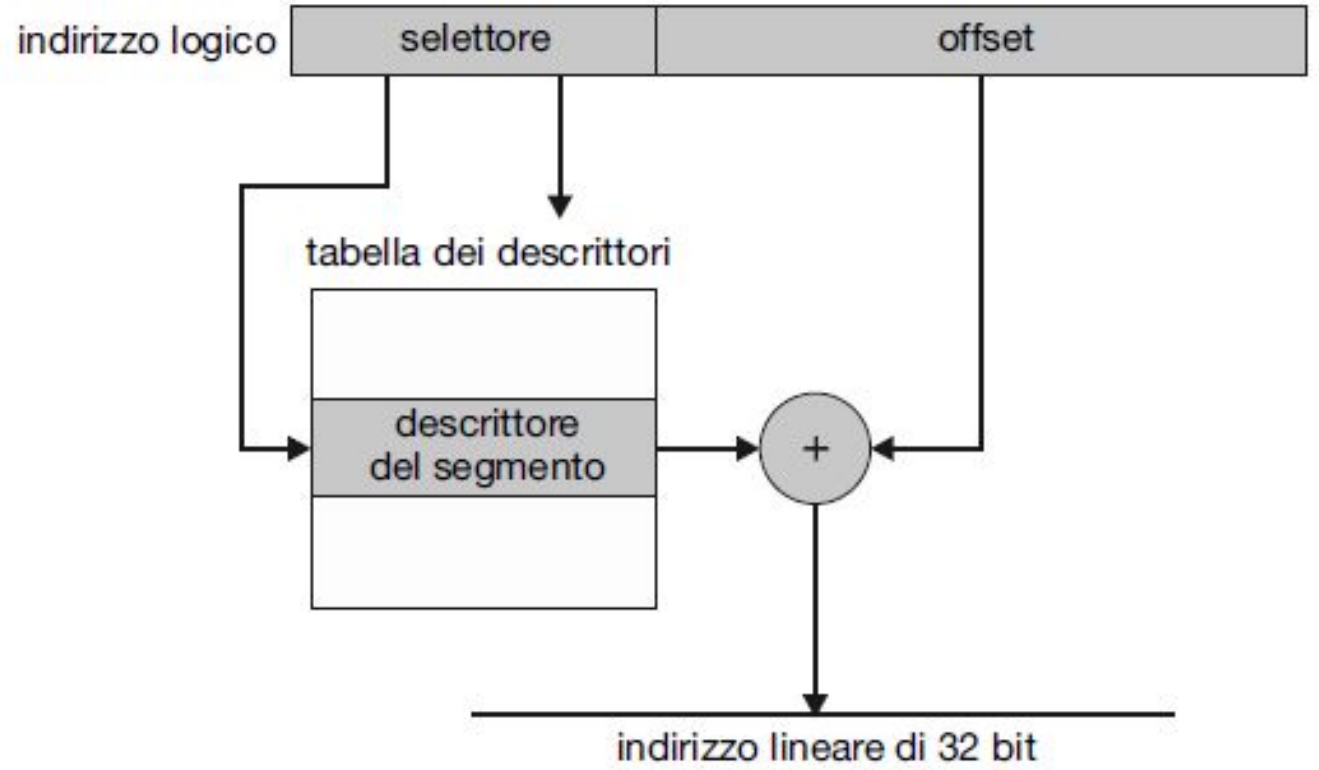


Figura 9.22 Segmentazione in IA-32.

Paginazione in IA-32

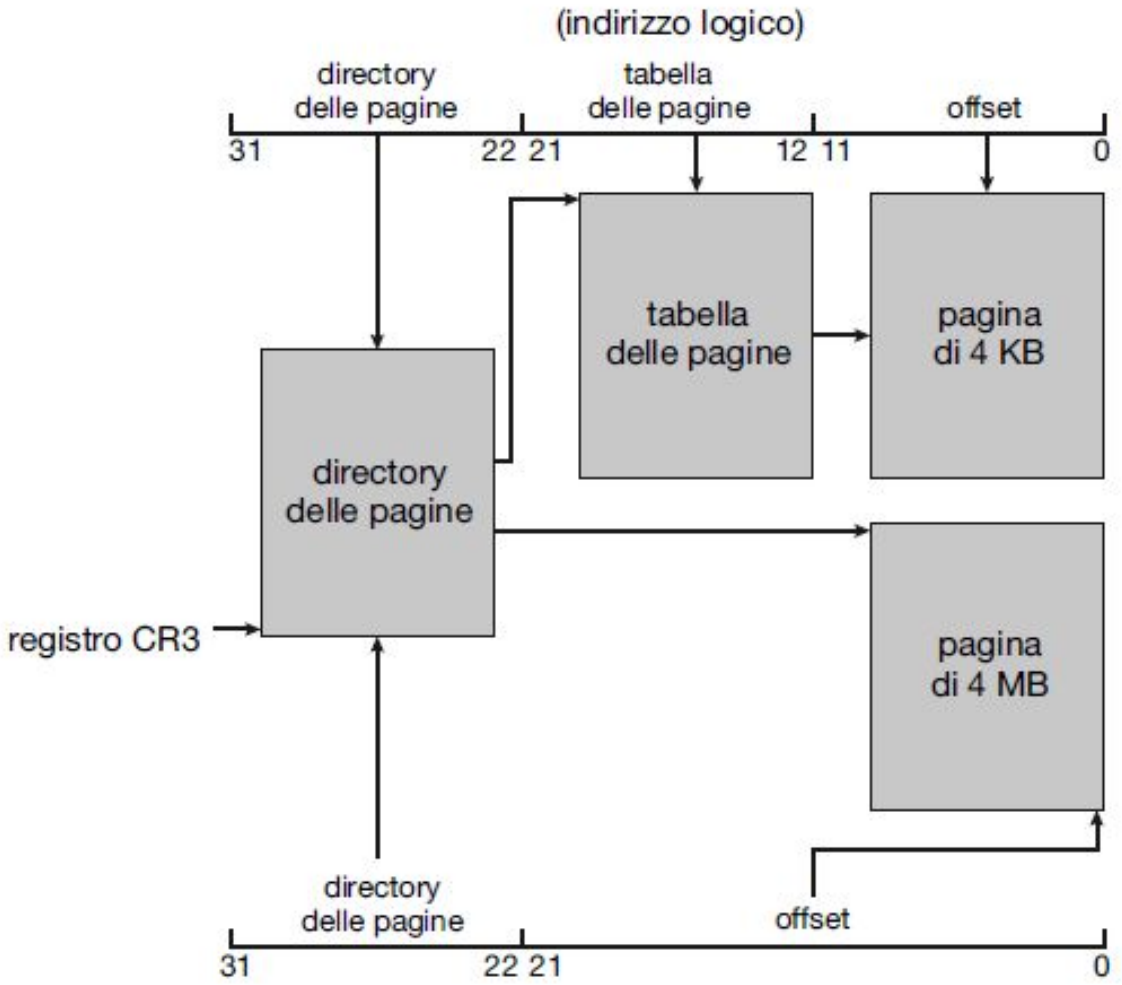


Figura 9.23 Paginazione nell'architettura IA-32.

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.

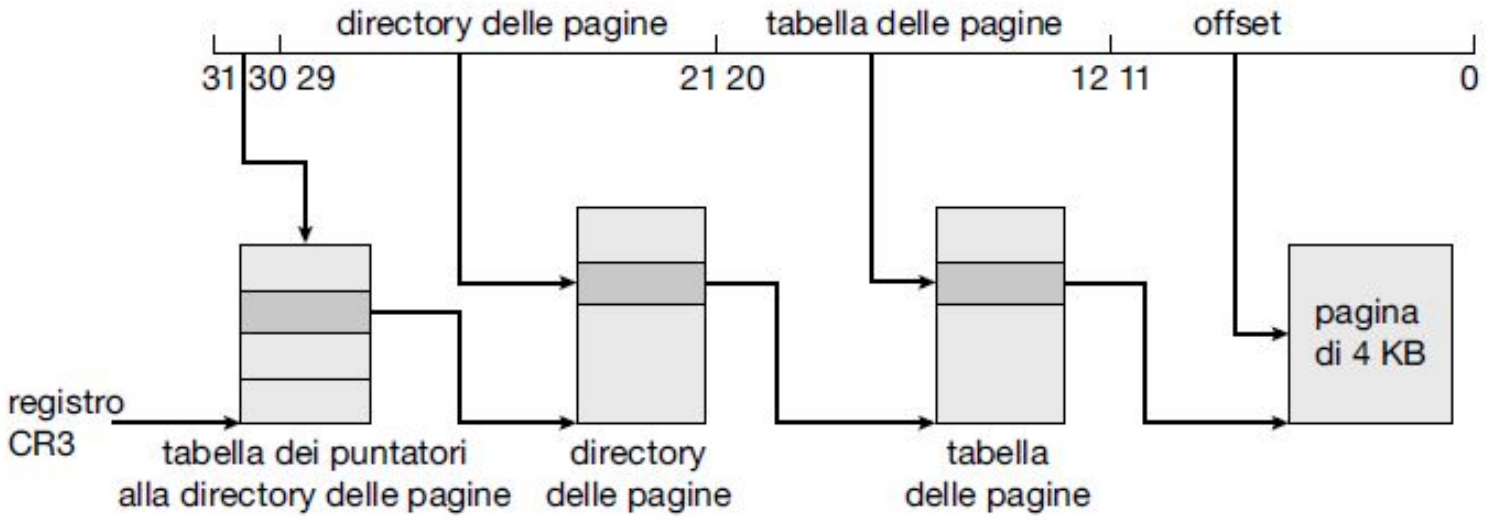


Figura 9.24 Estensione degli indirizzi di pagina.

Architettura x86-64

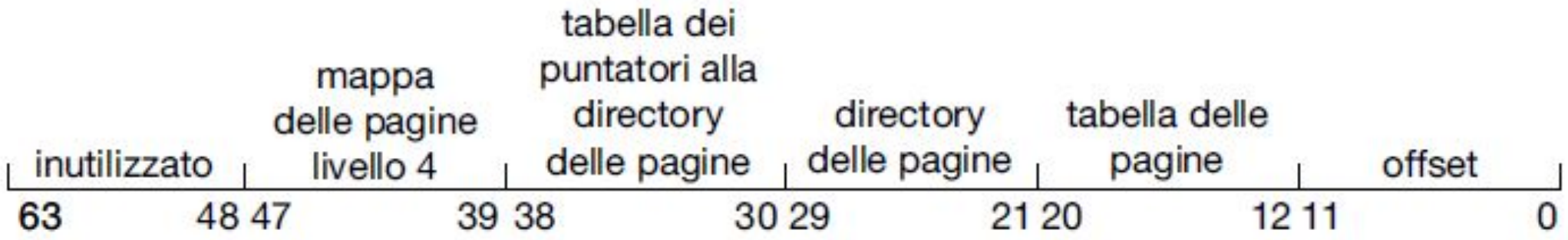


Figura 9.25 Indirizzo lineare in x86-64.

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.

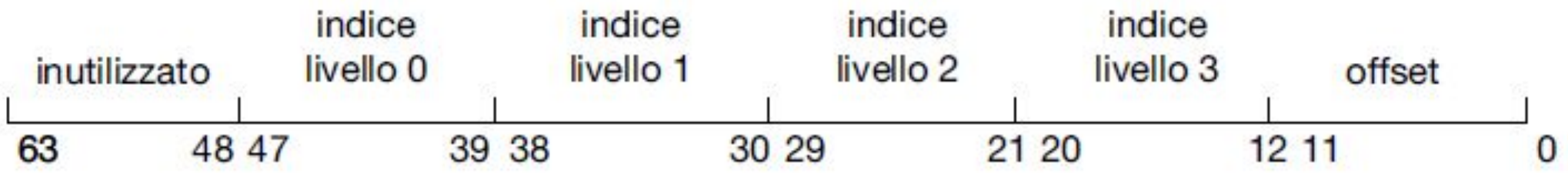


Figura 9.26 Indirizzo in ARM, con granularità di 4 KB.

Paginazione ARMv8

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

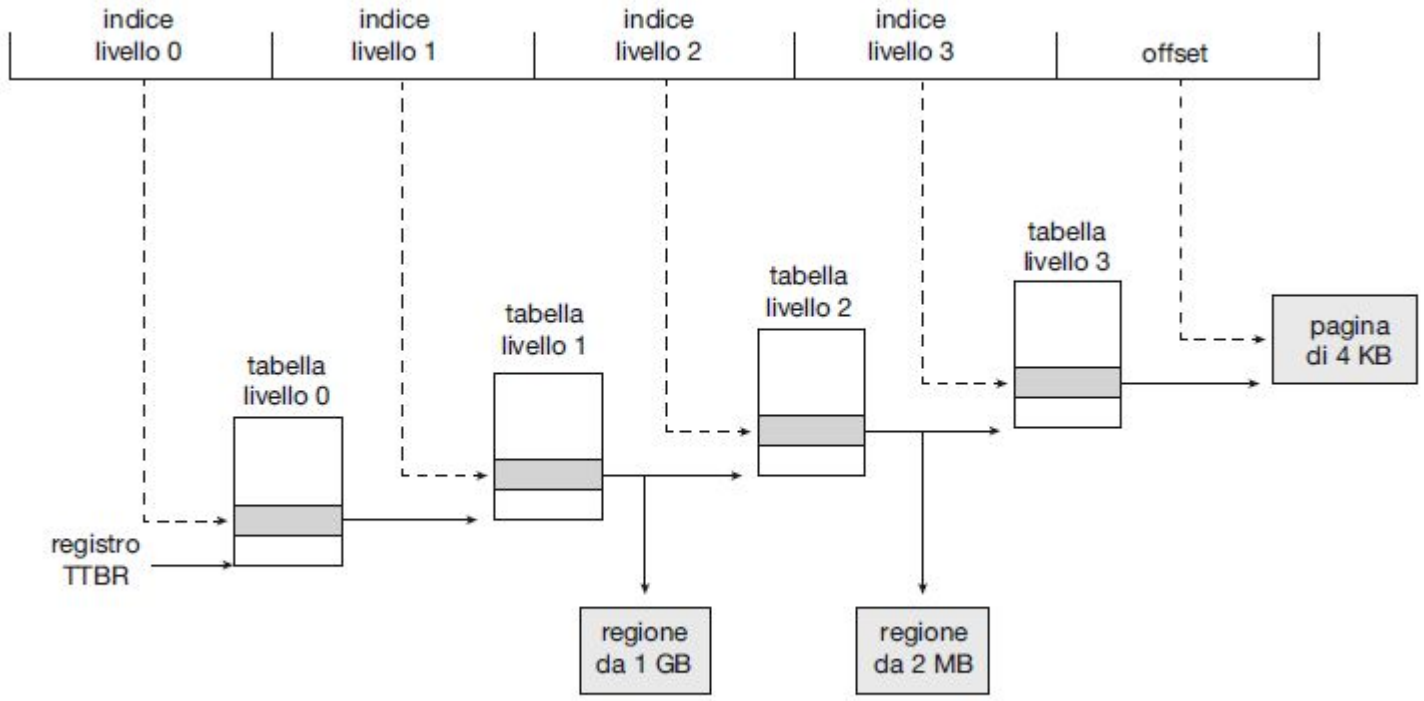
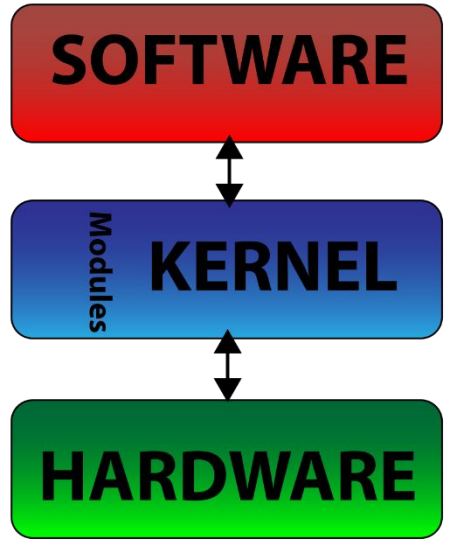


Figura 9.27 Paginazione gerarchica su 4 livelli in ARM.

UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

Corso di Sistemi Operativi

Paginazione



Docente:
Domenico Daniele Bloisi