

La gestione della memoria

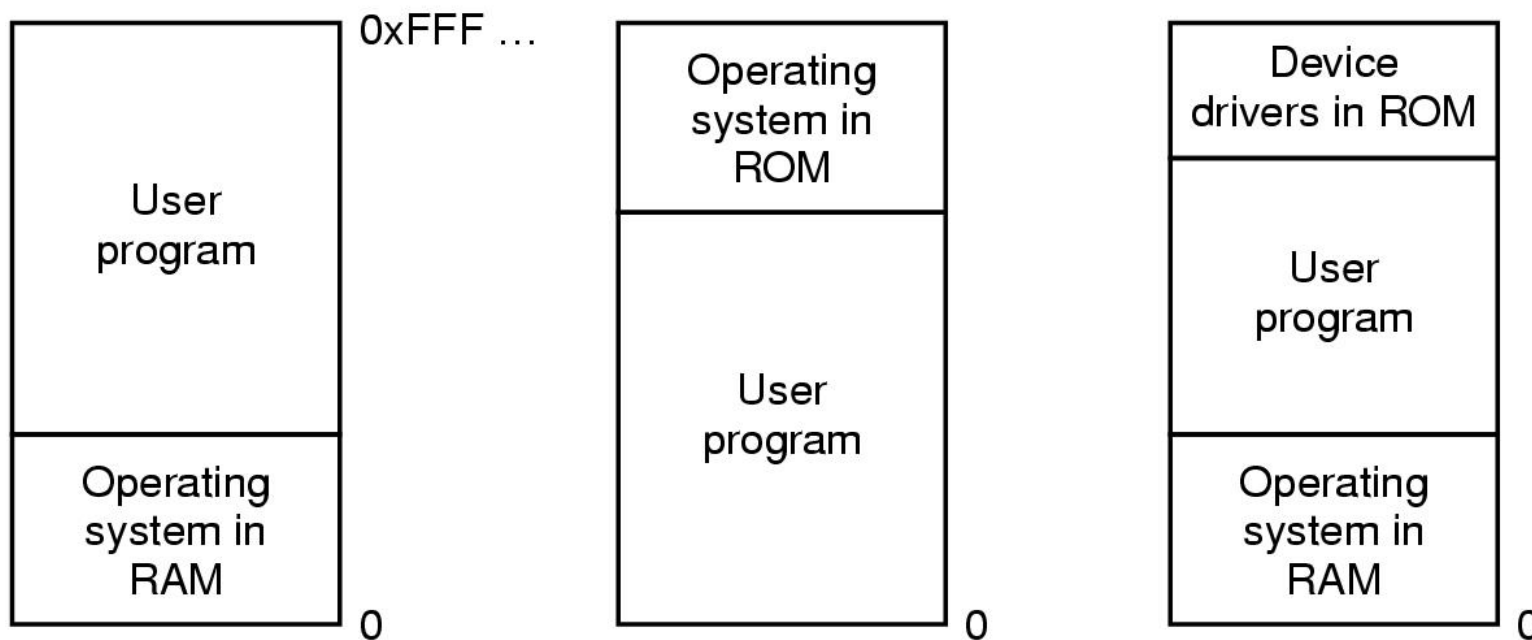
Lezione 16
Sistemi Operativi

La memoria come risorsa

- La memoria centrale è una risorsa fondamentale di un sistema di calcolo
- L'accesso a memoria centrale è una delle operazioni più frequenti
 - Accesso ai dati e al codice
- La tecnica di gestione della memoria ha un effetto determinante sulle prestazioni del sistema

Schema elementare di gestione della memoria

- Sistema mono programmato
 - Un solo programma e il sistema operativo



(a)

A.A. 2006/2007

(b)

3

(c)

Corso: Sistemi Operativi
© Danilo Bruschi

Memoria e prestazioni

- Calcoliamo l'utilizzazione della CPU in un sistema monoprogrammato
- Se l'unico processo spende in media 40% del tempo di esecuzione in attesa di I/O, l'utilizzo complessivo della CPU è

$$0.6 = 1 - 0.4$$

- Per migliorare questo parametro dobbiamo ridurre il tempo in cui la CPU è inattiva
 - Fare in modo che quando un processo esegue I/O il microprocessore venga usato per altre attività

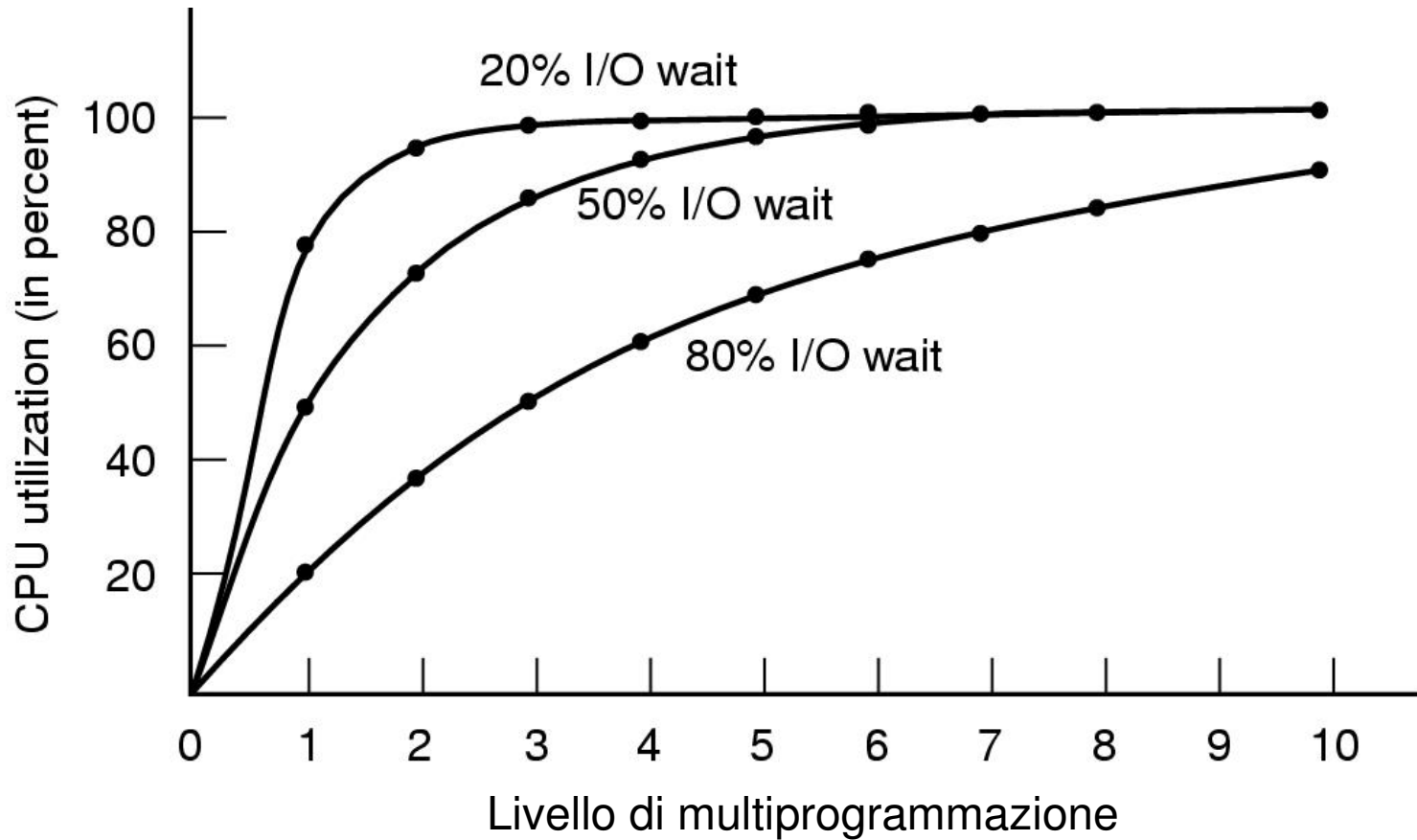
Memoria e prestazioni

- Con due processi in esecuzione, ciascuno con percentuale di I/O pari al 40%, l'utilizzo della CPU diventa

$$0.84 = 1 - 0.16$$

- Generalizzando
 - Siano dati n processi **indipendenti**
 - Ciascun processo spende una frazione p del proprio tempo di esecuzione in attesa di I/O
 - Non usa la CPU
 - La probabilità che tutti gli n processi siano in attesa è p^n
 - L'utilizzazione della CPU è $1 - p^n$
 - Quando tutti i processi sono in attesa, la CPU è idle

Memoria e prestazioni



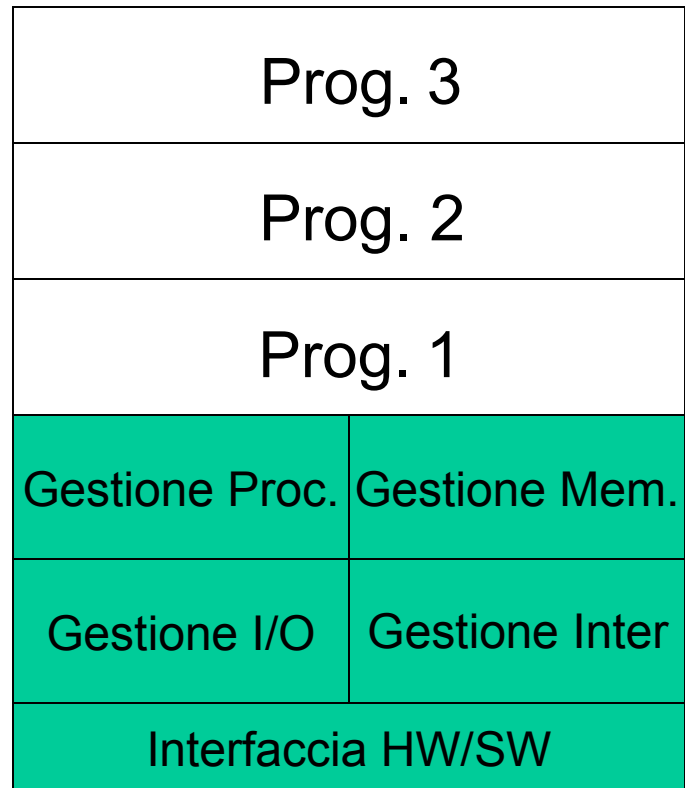
Osservazioni

- Se i programmi sono I/O bound, molti programmi devono essere in esecuzione per raggiungere un elevato utilizzo di CPU
- Aumentare la memoria centrale di un sistema non fa aumentare l'utilizzo di CPU in misura proporzionale all'incremento del livello di multiprogrammazione
 - L'efficacia dipende dal livello corrente di utilizzo
 - Pendenza della curva di utilizzo

Memoria e prestazioni

- Le soluzioni possibili sono due
 - Mantenere la stessa politica di gestione della memoria e ogni volta che un processo deve fare I/O effettuare uno swapping da memoria centrale a disco
 - Modificare la politica di gestione della memoria e consentire la presenza in memoria di più programmi
 - sistemi multiprogrammati

I Sistemi Multiprogrammati



Partizioni multiple fisse

- Questo schema di gestione della memoria richiede alcuni accorgimenti nella fase di preparazione dei programmi e generazione del codice
 - Compilatore
 - Linker
 - Loader
 - statico
 - Dinamico (run time)

Traduzione

- **Compilatore**
 - Traduce un programma scritto in linguaggio ad alto livello nel suo corrispettivo in linguaggio assembly
- **Assemblatore**
 - Traduce un programma scritto in linguaggio assembly nel suo corrispettivo in linguaggio macchina
- **Prodotti aggiuntivi della traduzione**
 - Symbol table
 - Informazioni sulle dimensioni del modulo oggetto
 - Informazioni per il debugging

Linker

- Collega i moduli oggetto compilati indipendentemente
 - Statico
 - Tutti i riferimenti sono risolti e tradotti in indirizzi logici dopo la compilazione, prima del caricamento
 - Dinamico
 - A tempo di caricamento
 - Moduli esterni (target) sono aggiunti all'applicazione dopo averla caricata in memoria con il loader
 - A tempo di esecuzione
 - I moduli target sono caricati e i riferimenti risolti solo durante l'esecuzione al momento in cui vengono richiamati
 - Con linking dinamico si facilita l'aggiornamento dei moduli target e la condivisione del codice

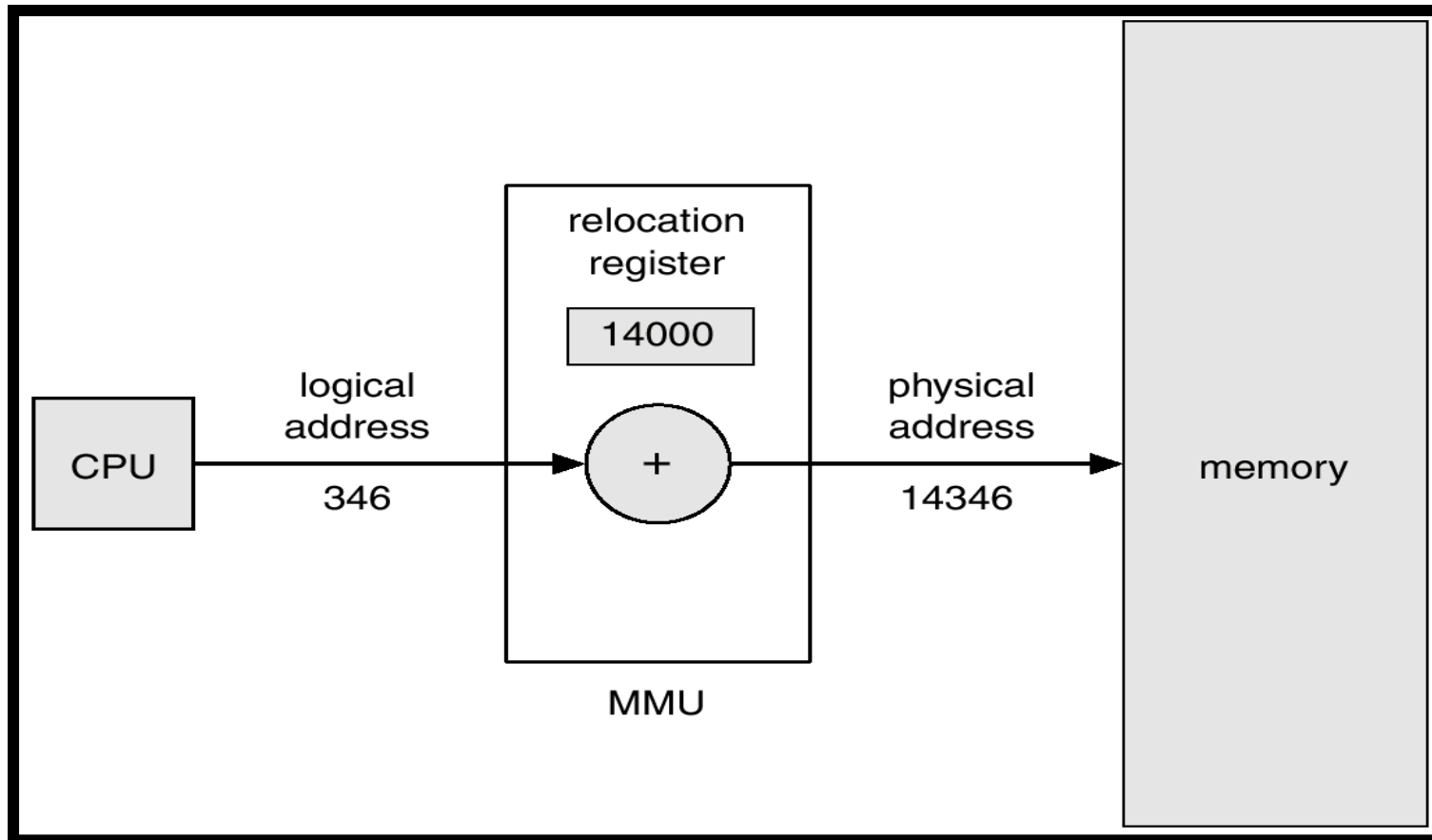
Loader

- Carica un programma in memoria
 - Usa le informazioni nella symbol table
 - Caricamento assoluto
 - Gli indirizzi usati dal programma sono fissati a tempo di compilazione
 - Caricamento rilocabile
 - Gli indirizzi generati dal compilatore sono tutti relativi ad un punto di inizio (indirizzi logici) e vengono tradotti in indirizzi fisici prima del caricamento del programma
 - Caricamento dinamico a tempo di esecuzione
 - I riferimenti alla memoria sono mantenuti in formato relativo e tradotti in indirizzi assoluti solo nel momento in cui vengono utilizzati

Sistema multiprogrammato: requisiti

- Rilocazione
 - Il programmatore non sa a priori dove il programma verrà caricato in memoria per essere eseguito
 - Durante l'esecuzione, il programma può essere scaricato sul disco e ricaricato successivamente in memoria ad un indirizzo diverso (rilocazione)
 - Gli indirizzi di memoria nel codice devono essere tradotti in indirizzi fisici di memoria, questa funzione viene svolta da hw dedicato (Memory Management Unit)

Rilocazione dinamica



Sistema multiprogrammato: requisiti

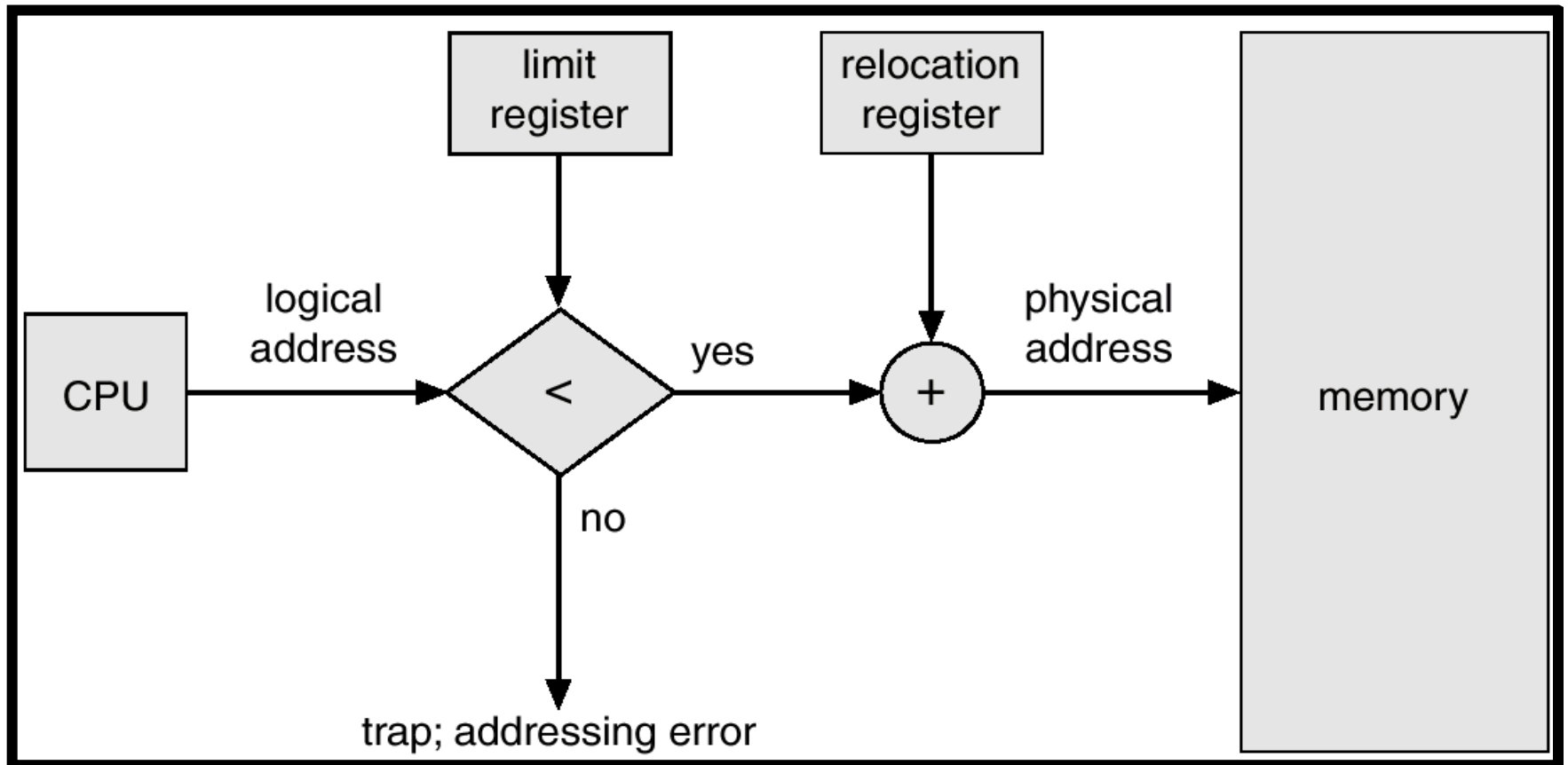
- Protezione

- I processi non devono poter accedere ad aree di memoria di altri processi senza permesso

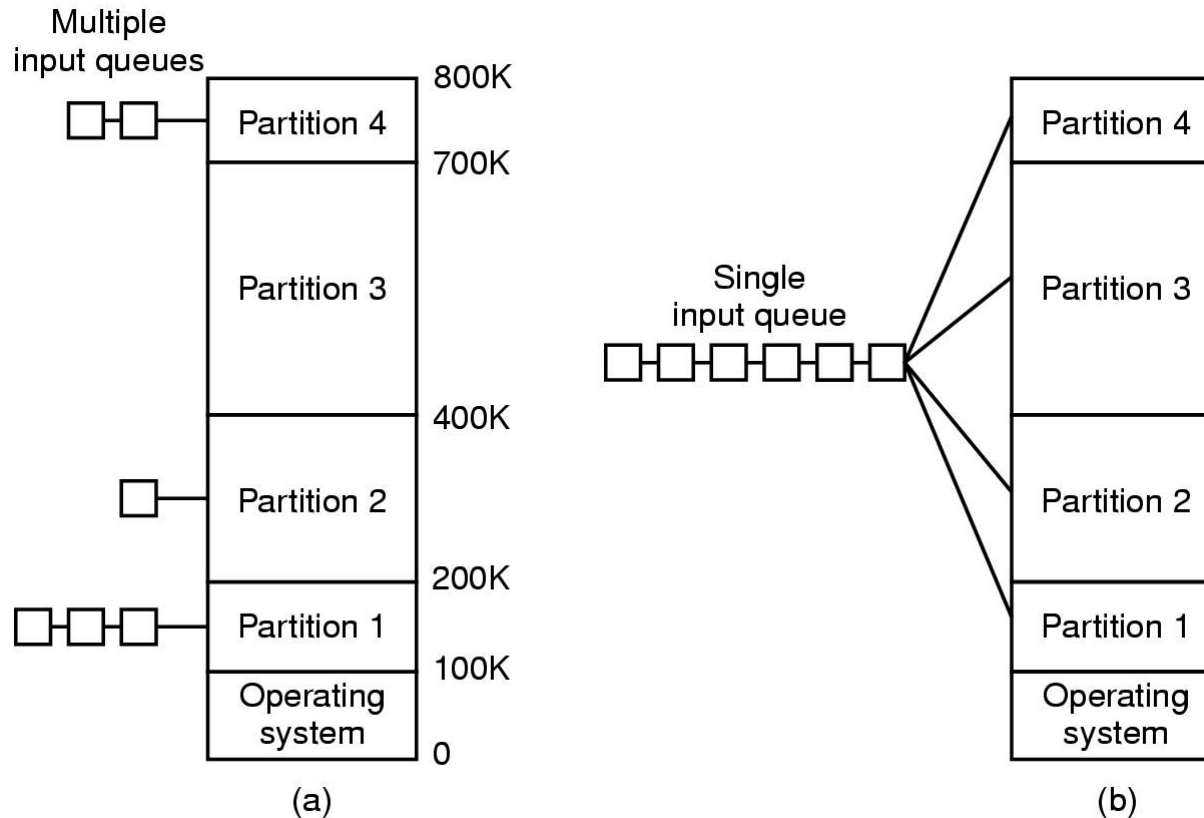
```
        mov  eax,end
loop   mov  (eax),0
        add  eax,4
        jmp  loop
end    ret
```

- Poiché è impossibile controllare a priori quali indirizzi di memoria un programma referenzierà i riferimenti alla memoria devono essere verificati durante l'esecuzione

Protezione



Partizioni multiple fisse

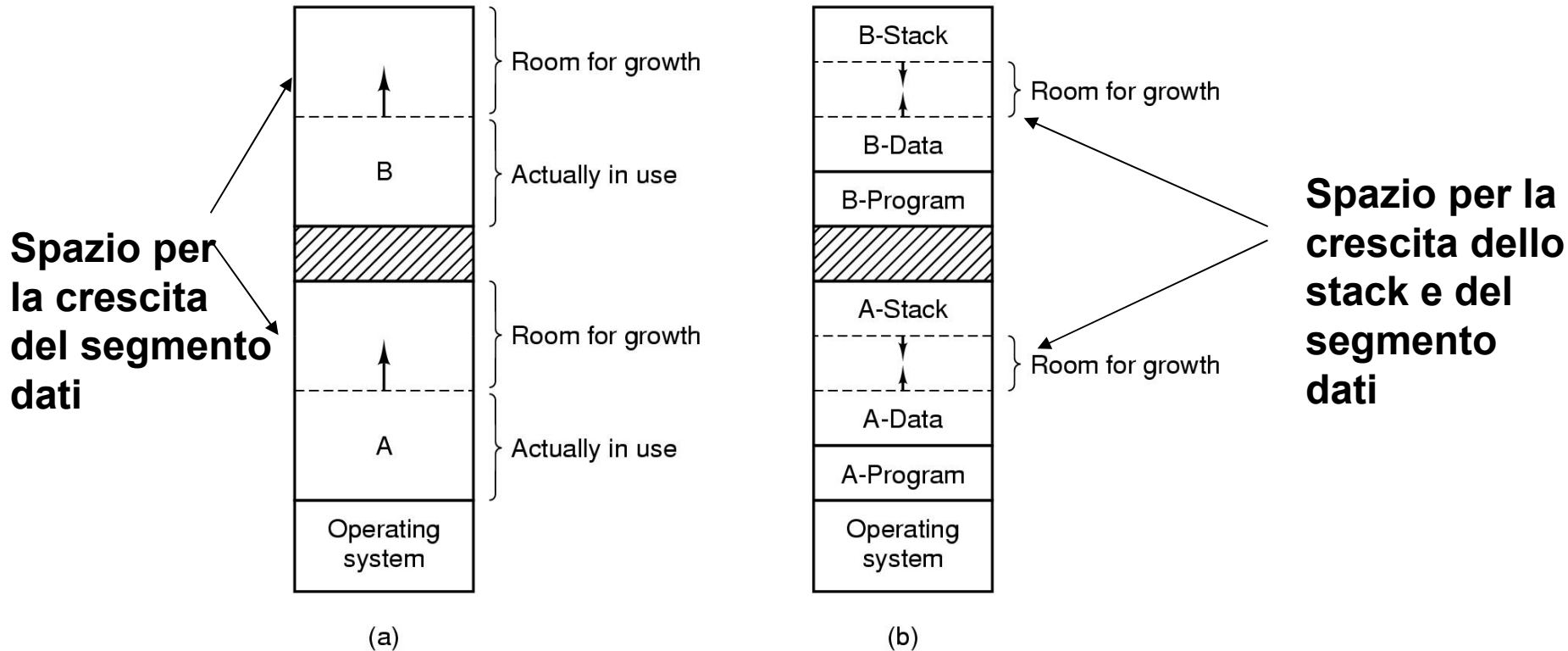


- Memoria a partizioni fisse
 - Code separate per ogni partizione
 - possibile sotto utilizzo del sistema
 - Singola coda di input

Partizioni multiple fisse: criticità

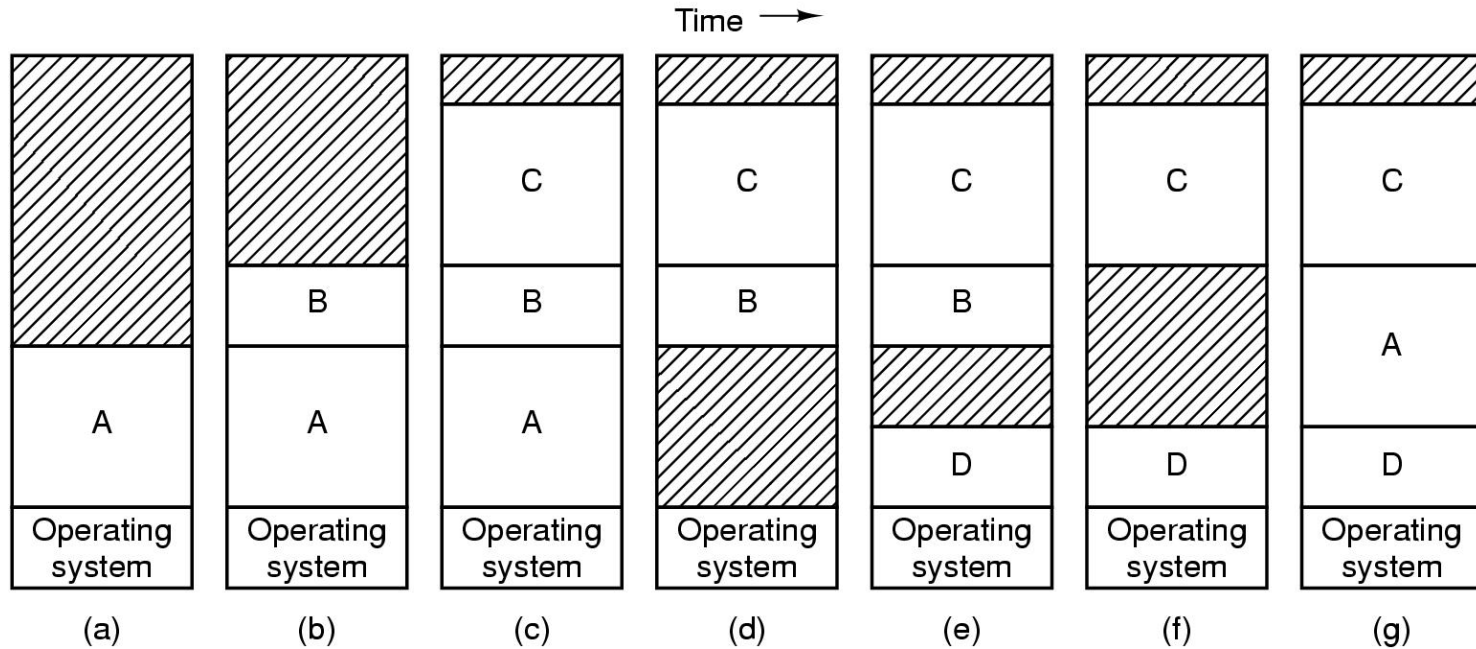
- Le partizioni fisse hanno avuto un certo successo in sistemi di tipo batch
 - Relativa staticità dei programmi eseguiti su un sistema
- Con l'avvento dei sistemi time sharing, i problemi principali da risolvere sono diventati:
 - dover gestire processi la cui dimensione, nel complesso, era di gran lunga superiore di quella della memoria centrale
 - Dover gestire processi la cui dimensione era fortemente variabile

Partizionamento guidato dai processi



- Le dimensioni di un processo cambiano durante l'esecuzione va quindi prevista la possibilità di l'espansione

Partizioni multiple variabili



- Numero, dimensione e posizione delle partizioni cambiano col variare dei processi in memoria
- Lo stato della memoria si modifica continuamente
 - I processi accedono alla memoria centrale
 - Lasciano la memoria per poi rientrarvi, non necessariamente nello stesso posto occupato in precedenza

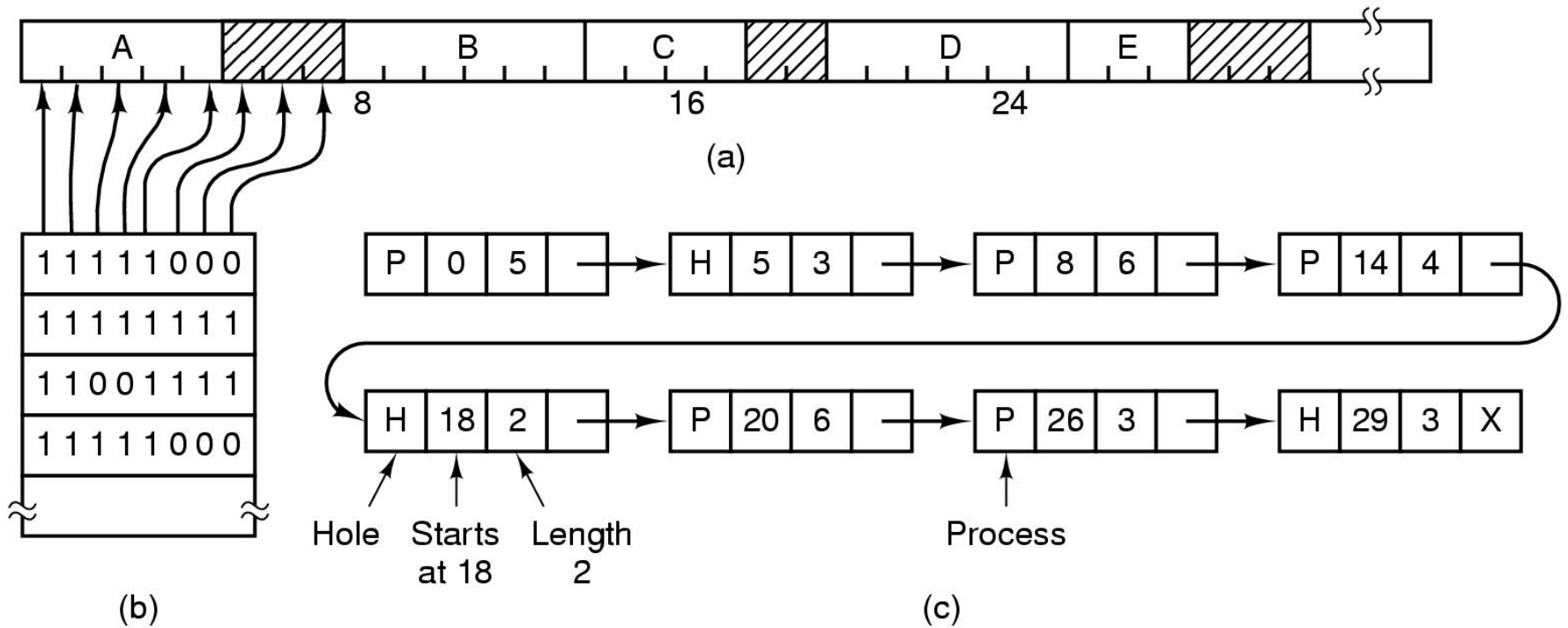
Partizioni multiple variabili: criticità

- Gestione delle zone libere e occupate della memoria centrale
- Frammentazione esterna
 - Degli spazi liberi

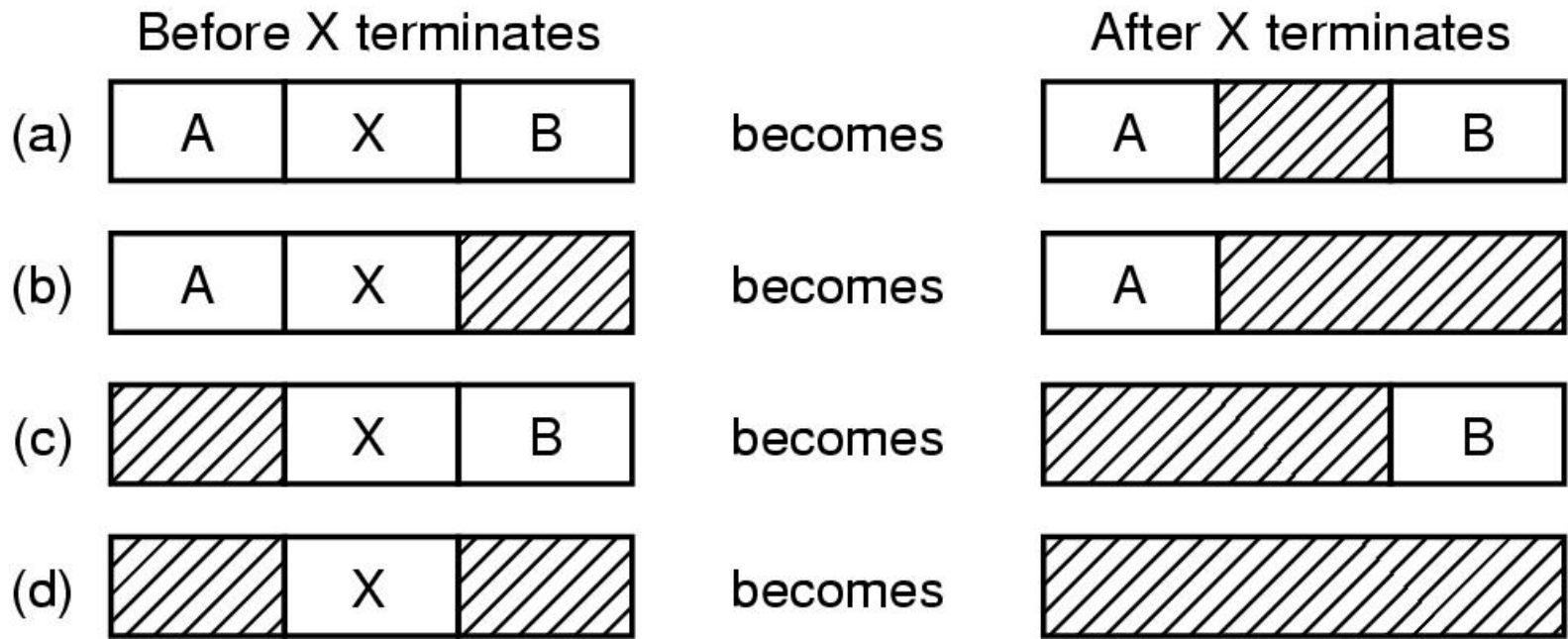
Gestione con bitmap

- Suddivisione della memoria in unità minime allocabili
- Associazione a ciascuna unità di un bit
 - Libera/occupata
- Dimensione della tabella
 - 1 bit ogni unità allocabile di k bit
- Ricerca di successioni di n bit a zero nella bitmap per trovare aree libere di dimensioni adeguate al programma da caricare
 - Semplice ma inefficiente

Gestione Memoria



Gestione della memoria con liste



- Aree libere contigue vengono compattate
- Quando si usano le liste linkate, può essere più efficiente fare ricorso a liste doppiamente linkate

Strategie di allocazione

- First Fit: il primo buco sufficientemente grande per contenere il processo
- Next Fit: si tiene traccia dell'ultima allocazione e si ricerca il prossimo buco sufficientemente grande
- Best Fit: il più piccolo buco sufficientemente grande per contenere il processo
- Worst Fit: il più grande buco che possa contenere il processo
- Quick Fit: mantenere delle liste separate per le richieste di spazi più frequenti

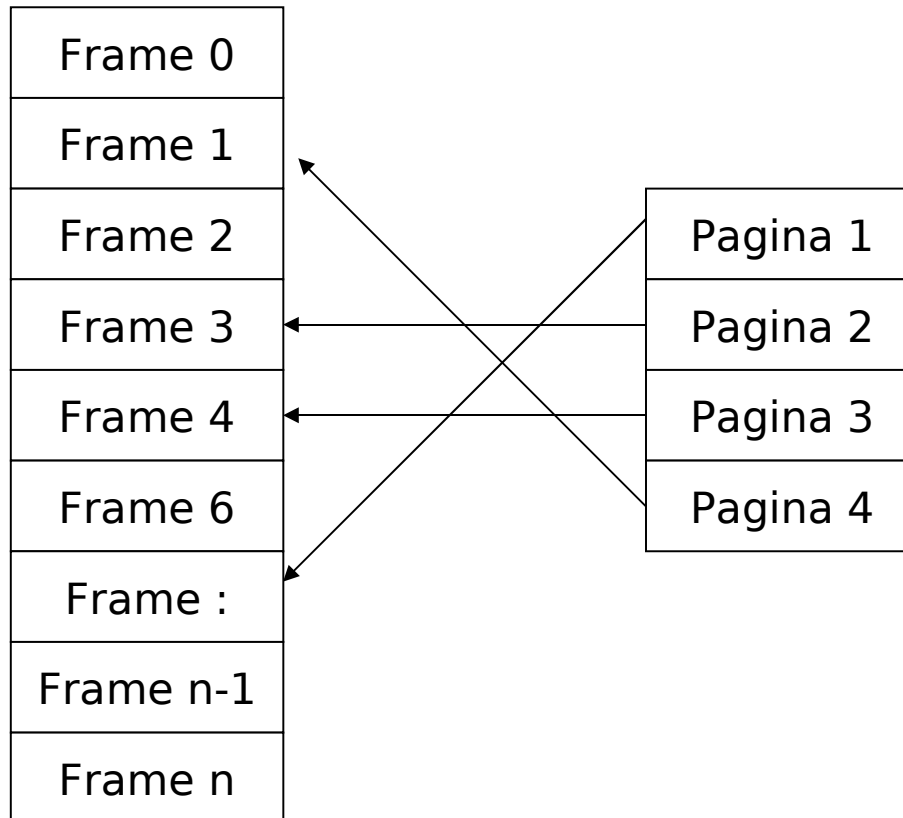
Frammentazione esterna

- Frazionamento della memoria libera in piccoli pezzi inutilizzabili
- Una possibile soluzione è l'adozione di strategie di compressione della memoria centrale
 - Operazione molto costosa in termini di efficienza del sistema
 - Non esistono strategie ottimali

Paginazione

- Problema: con gli schemi a partizione è possibile caricare in memoria un processo **solo** quando si trova una porzione di memoria **contigua**, che possa contenere l'intero processo => il processo di allocazione può richiedere parecchio tempo
- Soluzione
 - Suddivido la memoria fisica e logica in blocchi di dimensioni uguali (1024 byte), chiamo i primi "frame" e gli altri "pagine", carico un processo quando ho a disposizione un numero di pagine sufficiente a contenerlo

Paginazione



Paginazione

- In un sistema paginato gli indirizzi logici di memoria sono espressi da una copia $\langle p, d \rangle$, a cui va associato in fase di esecuzione un indirizzo fisico lineare, che esprime la locazione esatta del dato o istruzione in memoria

Calcolo degli indirizzi

- In un sistema paginato con pagine di 256 byte l'indirizzo 26251 sarà rappresentato dalla copia
 - $p = 26251 \text{ DIV } 256 = 102$
 - $d = 26251 \text{ MOD } 256 = 139$
- Per conoscere l'indirizzo fisico associato alla locazione di indirizzo logico $\langle 102, 139 \rangle$ dobbiamo conoscere in quale frame è stata caricata la pagina 102
 - $\text{Indirizzo fisico} = \text{n.ro frame} * 256 + d$

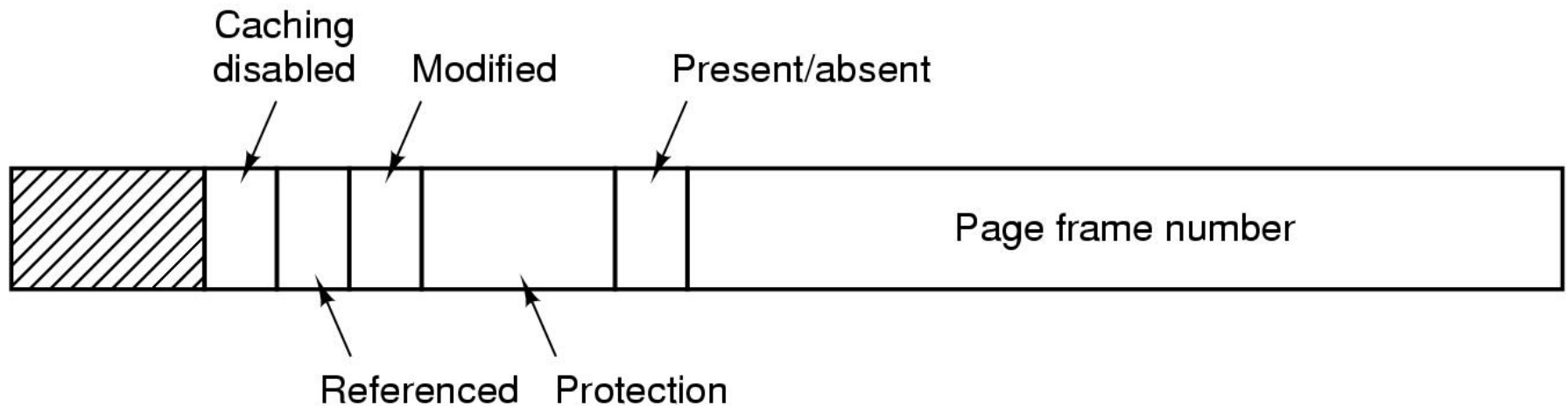
Paginazione

- Con pagine di dimensione 2^k , per il calcolo degli indirizzi fisici si sfrutta l'indirizzo logico
 - I k bit meno significativi dell'indirizzo logico individuano lo spiazzamento all'interno della pagina
 - I restanti bit più significativi individuano il numero della pagina
 - Il numero di pagina si usa come indice nella tabella delle pagine
 - Il valore trovato in corrispondenza è il numero di page frame associato alla pagina
 - Se la pagina è presente in memoria, il numero di page frame è copiato nei bit alti del registro di output e concatenato ai k bit bassi dell'offset
- ES: 26251: 000001100110 10001011

Supporti necessari

- Un sistema che offre la paginazione deve gestire le seguenti strutture dati
 - Tabella dei frame
 - Tabella delle pagine, che definisce l'associazione pagina → frame
 - una per ogni processo in esecuzione o in attesa di esecuzione
 - Tabella delle pagine del processo in esecuzione
 - Per ogni riferimento a memoria è necessario accedere alla tabella della pagine, l'operazione deve essere resa efficiente con il supporto dell'HW

Tabella delle pagine



Elemento di una tabella delle pagine

Paginazione

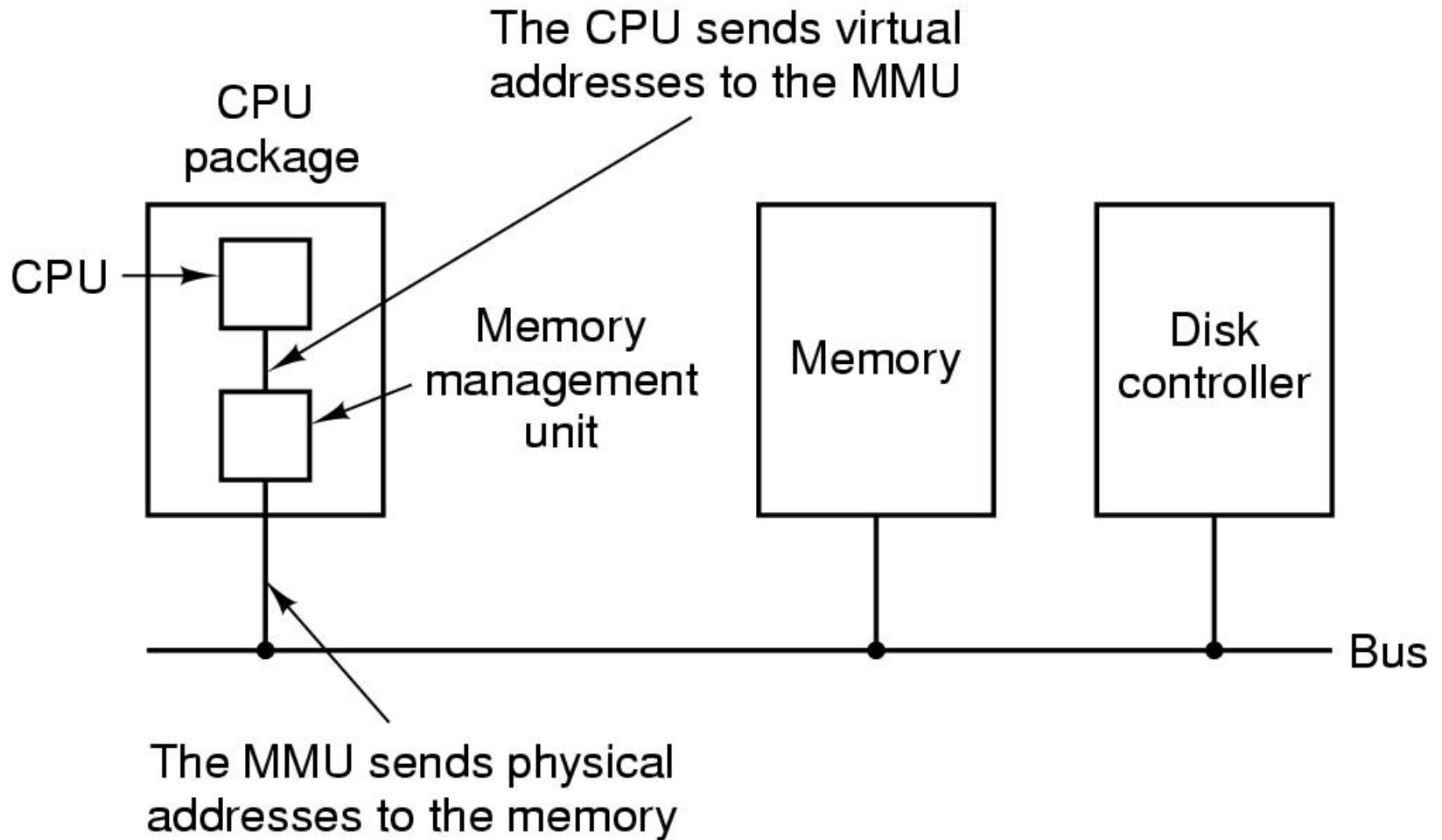
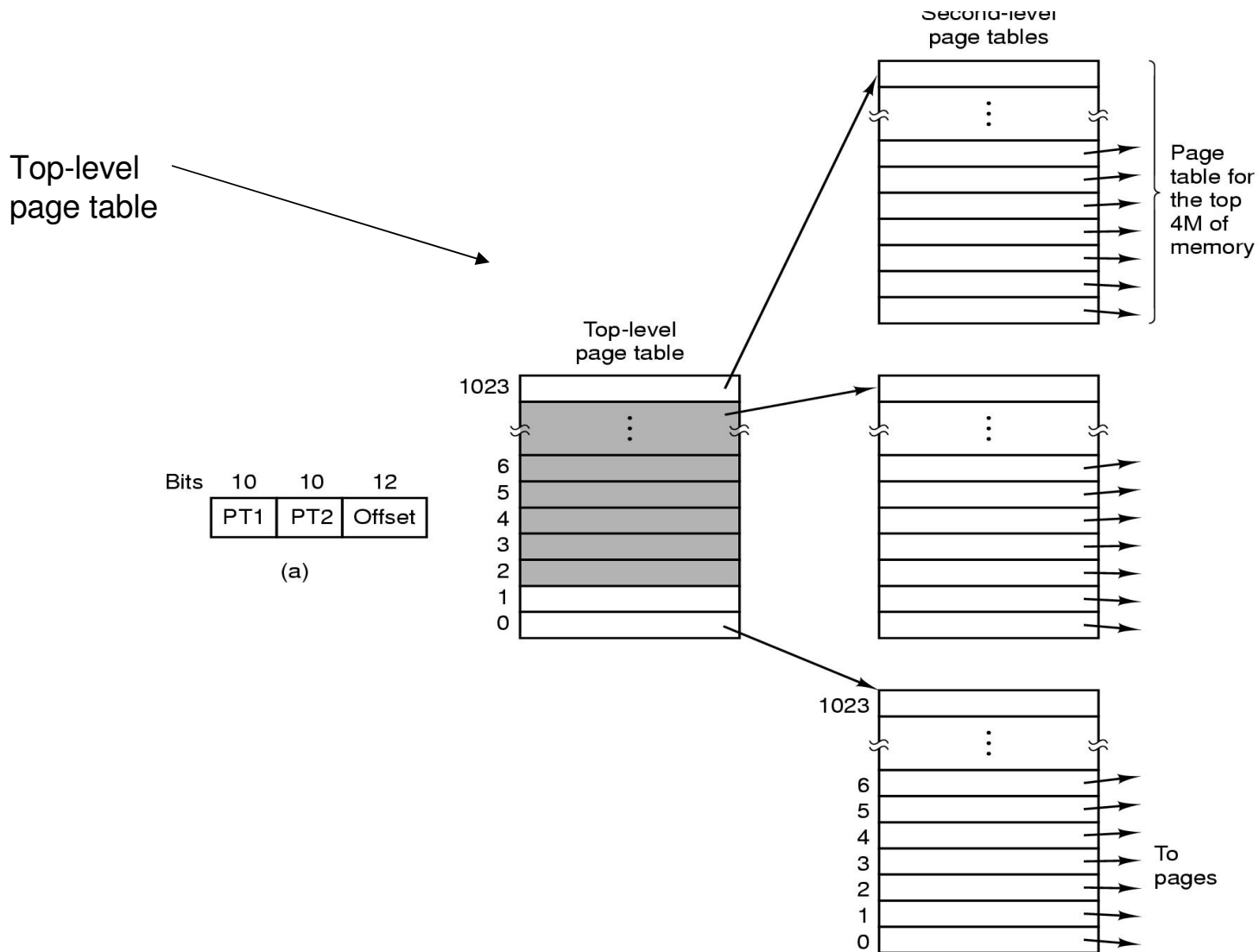


Tabella delle pagine

- La dimensione della tabella può essere molto elevata
 - Con pagine di 4KB
 - 1 milione di pagine con indirizzi di 32 bit, quindi 1 milione di elementi nella tabella
- L'associazione pagina- frame deve essere molto efficiente
 - Eseguita per ogni riferimento alla memoria
 - Un'istruzione richiede almeno 1 accesso
- Servono soluzioni specifiche

Tabella delle pagine a due livelli



Paginazione

- Procedura di caricamento di un programma in un ambiente multiprogrammato
 1. Long term scheduling verifica disponibilità dei frame in memoria centrale
 2. Il processo viene caricato in MC e la sua tabella delle pagine compilata e agganciata al PCB
 3. Quando il processo va in esecuzione la sua tabella delle pagine viene caricata in quella del sistema

Tabella delle pagine di sistema

- Esistono tre alternative per implementarla
 - Registri specializzati
 - Molto efficiente in esecuzione ma molto costosa per tabelle di grosse dimensioni, soprattutto in context switch
 - Esiste un registro che indirizza la tabella delle pagine del processo in esecuzione, tra quelle di tutti i processi del sistema; la tabella sta in memoria
 - Lenta richiede due accessi a memoria centrale per recuperare le informazioni
 - Registri associativi
 - Implementano un Translation Lookaside Buffer di 8-64 elementi

TLBs – Translation Lookaside Buffers

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

TLB

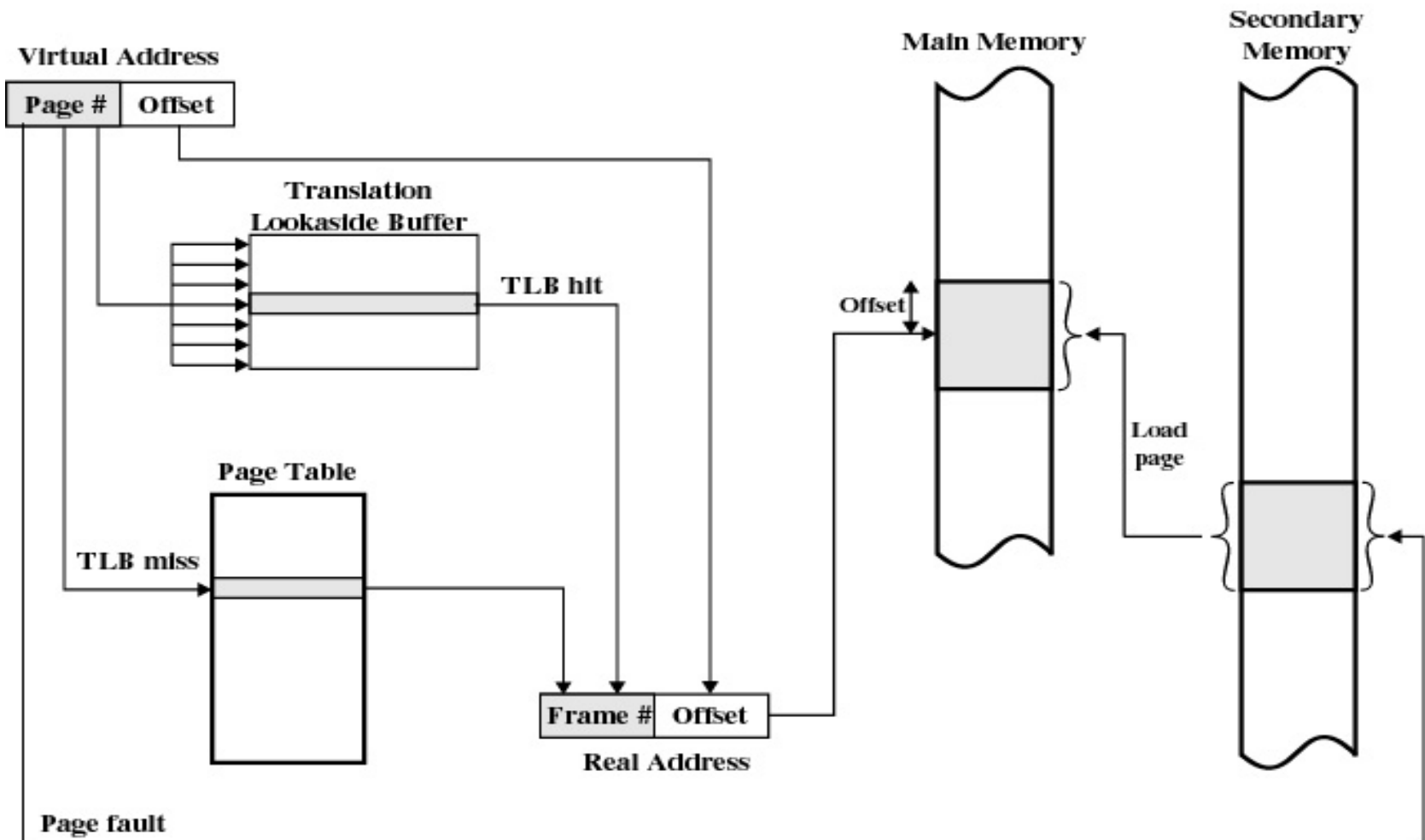
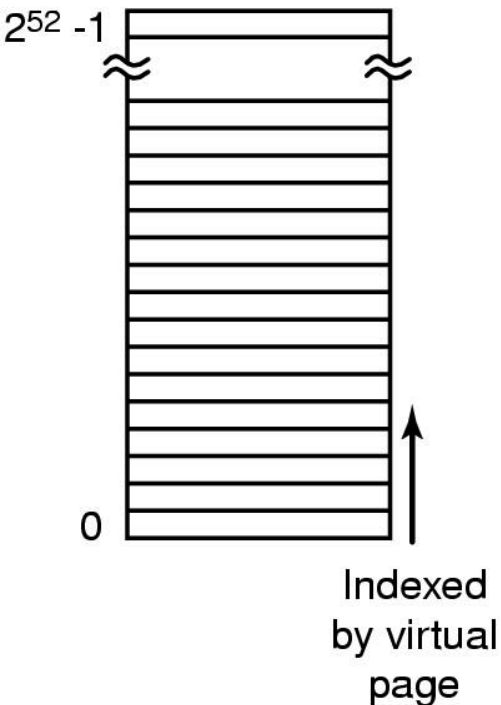


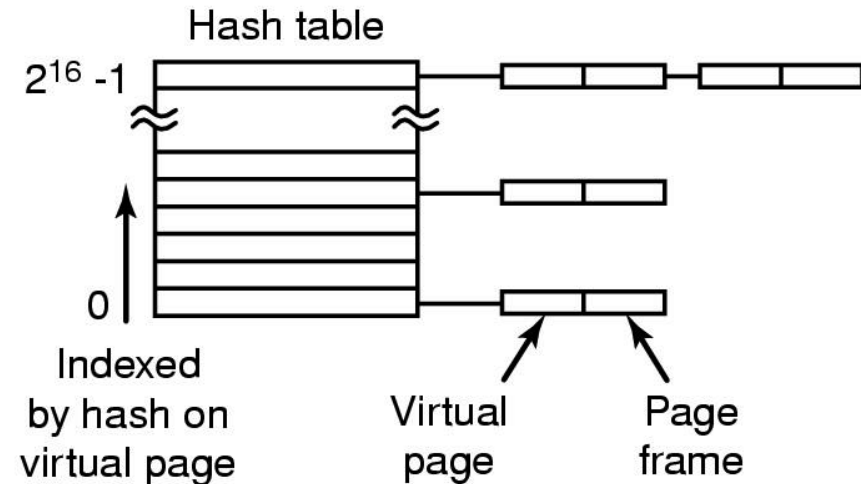
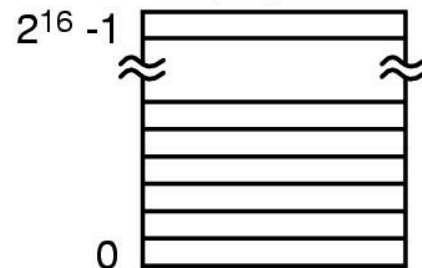
Figure 8.7 Use of a Translation Lookaside Buffer

Tabella delle pagine invertita

Traditional page table with an entry for each of the 2^{52} pages



256-MB physical memory has 2^{16} 4-KB page frames



La tabella indicizza i frame di memoria fisica

Un TLB permette la ricerca delle pagine più usate