LA MEMORIA VIRTUALE ALGORITMI DI SOSTITUZIONE DELLE PAGINE

Danilo Croce

Novembre 2024

Negli anni ’80 molte università utilizzavano un sistema timesharing con dozzine di utenti (più o meno soddisfatti) che lavoravano contemporaneamente su un sistema VAX da 4 MB.

Oggi Microsoft raccomanda di avere almeno 2 GB per un sistema a 64 bit con Windows 10.

VIRTUAL MEMORY

GESTIONE DELLA MEMORIA: OUTLINE

- Memory Abstraction
- **Virtual Memory**
- Algoritmi di sostituzione delle pagine
- Problemi di Progettazione per Sistemi di Paging

IL PROBLEMA DEL BLOATWARE E LA CRESCITA DELLA MEMORIA

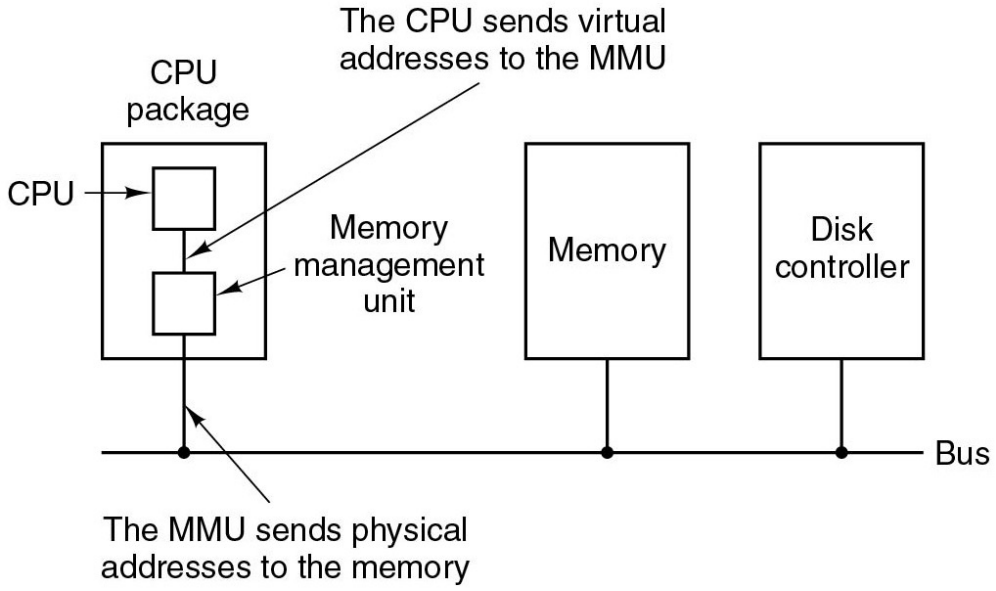
- Necessità di **gestire programmi che superano la capacità della memoria** disponibile.
  - Problema dei programmi più grandi della memoria esiste sin dalle origini, specialmente nelle scienze e nell'ingegneria.
- Negli anni ’60, introduzione di **tecniche per dividere programmi in parti gestibili.**
  - **Overlay**: sono piccole parti o segmenti di un programma.
  - **Solo l'overlay necessario viene caricato in memoria.**
  - Overlay successivi sovrascrivono o coesistono con quelli precedenti.
  - Gli overlay vengono scambiati tra memoria e disco.
- Originariamente, i programmatori dovevano suddividere manualmente i programmi in overlay.
  - Questa soluzione era tediosa e soggetta a errori.

MEMORIA VIRTUALE

- La **memoria virtuale estende l'idea dei registri base e limite.**
  - Ogni programma ha un proprio spazio degli indirizzi suddiviso in "**pagine**", che sono intervalli di indirizzi contigui.
  - Non tutte le pagine devono essere contemporaneamente nella memoria fisica:
    - l'hardware crea una mappa di quelle direttamente in memoria
    - se una pagina manca, il sistema operativo interviene
- La maggior parte dei sistemi moderni usa il "***paging***" (paginazione)
  - divisione dello spazio degli indirizzi in unità di dimensione fissa, es. 4 KB.
- **Alternativa**: "***segmentazione***" con unità di dimensione variabile:
  - **ora meno comune.**

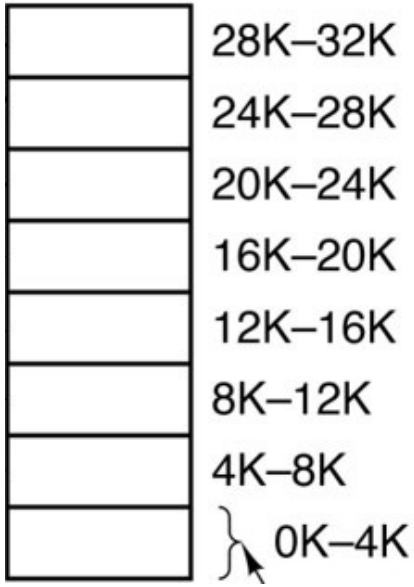
MEMORIA VIRTUALE (2)

- **Problema:**
  - Finora la memoria può essere assegnata ai processi solo in blocchi contigui.
- **Soluzione (e vantaggio dell'uso della Memoria Virtuale):**
  - Creare per il processo l'illusione di uno spazio di indirizzi ampio (ad esempio indicizzabile con 48 bit!!!).
  - Questo spazio è noto come spazio di indirizzi virtuale
  - La **RAM** (molto più limitata) è nota come **memoria fisica**.
  - **Memory Management Unit** (MMU): traduce gli indirizzi virtuali (come usati dal processo) in indirizzi fisici



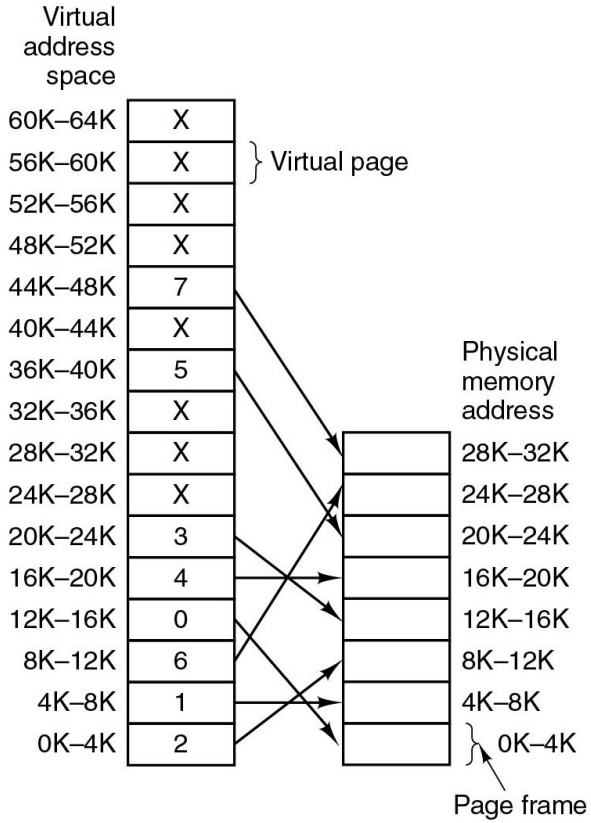
MEMORIA VIRTUALE E PAGINAZIONE

- I sistemi moderni utilizzano la **paginazione** (o ***paging***):
  - Dividendo la memoria fisica e virtuale in pagine di dimensioni fisse
    - ad esempio 4096 byte o 4 KB
  - Traducendo le pagine virtuali in **pagine fisiche** (**frame**)



SPAZIO DI INDIRIZZAMENTO VIRTUALE VS. SPAZIO DEGLI INDIRIZZI FISICI E PAGE TABLE

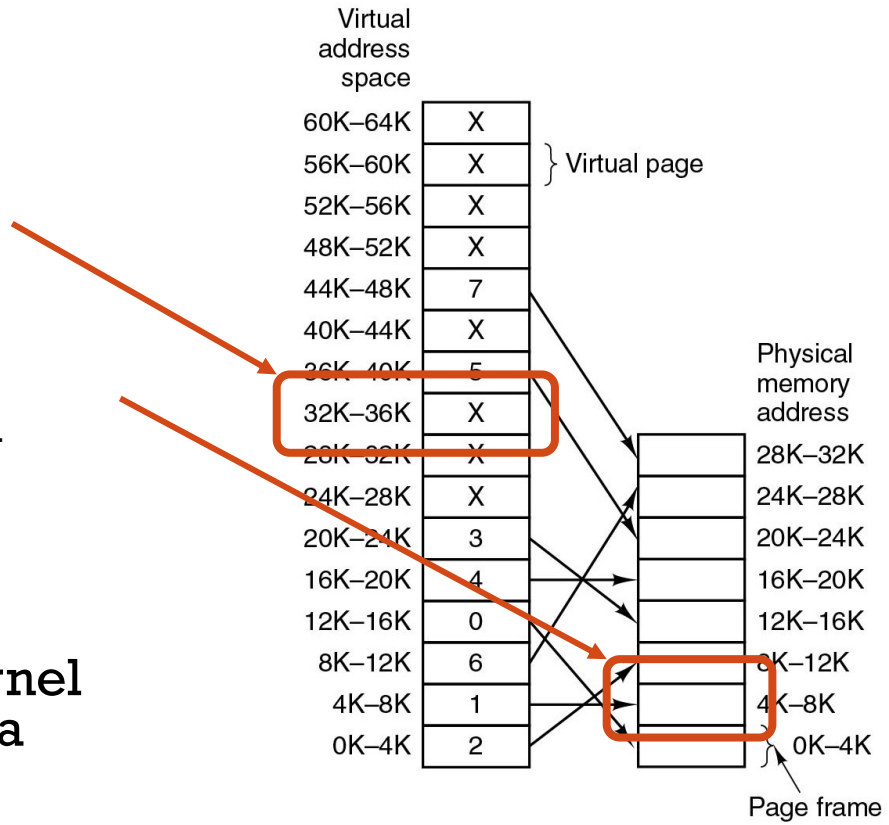
- **Mappatura Memoria:**
  - 16 pagine virtuali possono essere mappate in 8 frame fisici usando la Memory Management Unit (MMU).
  - Tuttavia, non tutte le pagine virtuali sono mappate fisicamente
    - quelle NON mappate sono contrassegnate con una X.
- Se un programma fa riferimento a una pagina non mappata, si verifica un «**Page fault**». Il sistema operativo allora:
  - Sposta un frame raramente usato su disco, se serve
    - Ma quale scegliere? Vedi più avanti
  - Carica la pagina richiesta nel frame libero o liberato.
  - Aggiorna la mappa della MMU per riflettere i cambiamenti.



La **relazione** tra gli indirizzi di memoria virtuale e fisica è data dalla **Page Table**.

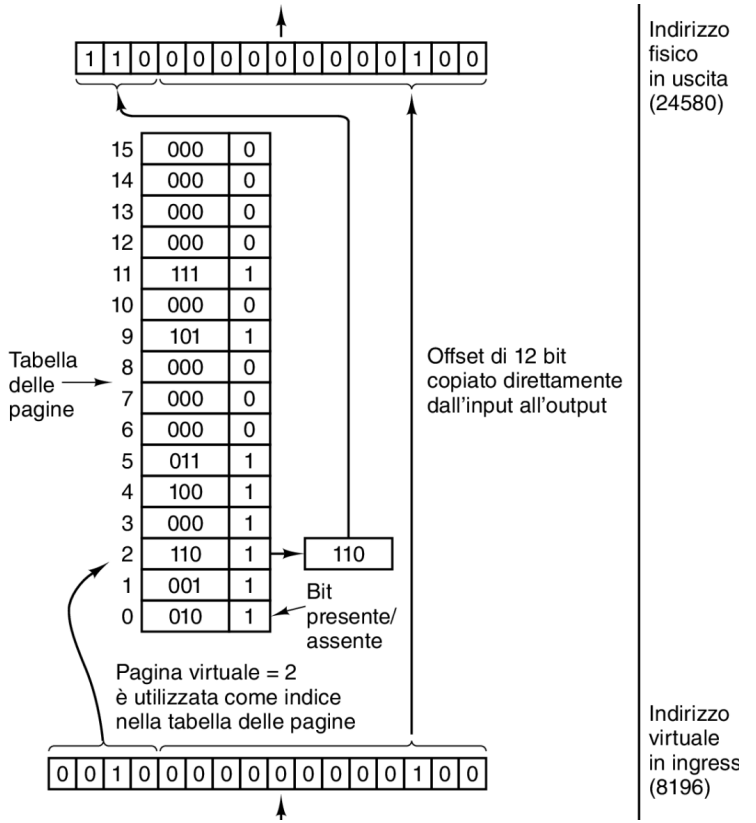
SPAZIO DI INDIRIZZAMENTO VIRTUALE VS. SPAZIO DEGLI INDIRIZZI FISICI E PAGE TABLE

- **Esempio:** gestire istruzione
  - `MOV REG,32780`
- Fa riferimento alla pagina virtuale 8.
  - Indirizzo `12` della pagina
  - `32780 - 2^15 (32768) = 12`
- Se non è mappata, il sistema operativo potrebbe decidere di sostituire il frame 1
  - Spostando il precedente su disco
  - Popolando il nuovo frame e puntando poi a
    - `4108 = 4096 + 12`
- Il page fault avviene nello spazio kernel durante il «**trap**» eseguito dal sistema operativo



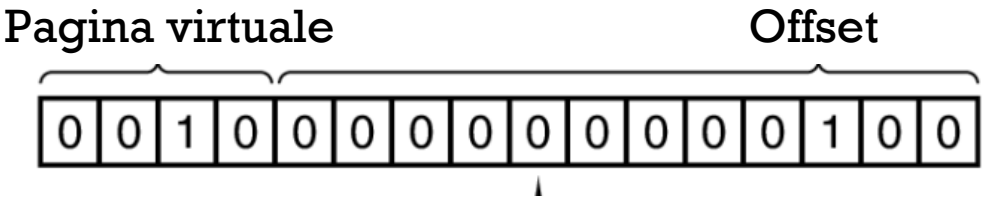
FUNZIONAMENTO INTERNO DELLA MMU

- **Indirizzo Virtuale:** 8196
  - **Rappresentazione Binaria:**
    - `0010 000000000100`
- **Suddivisione dell'Indirizzo Virtuale:**
  - **Numero di pagina**: 4 bit (permette di gestire 16 pagine)
  - **Offset**: 12 bit (indirizza 4096 byte per pagina che compongono ogni frame)
- **Mappatura tramite la Tabella delle Pagine:**
  - Numero di pagina →
    - Indice nella tabella delle pagine →
      - Numero di frame



Funzionamento interno della MMU con 16 pagine da 4 kB.

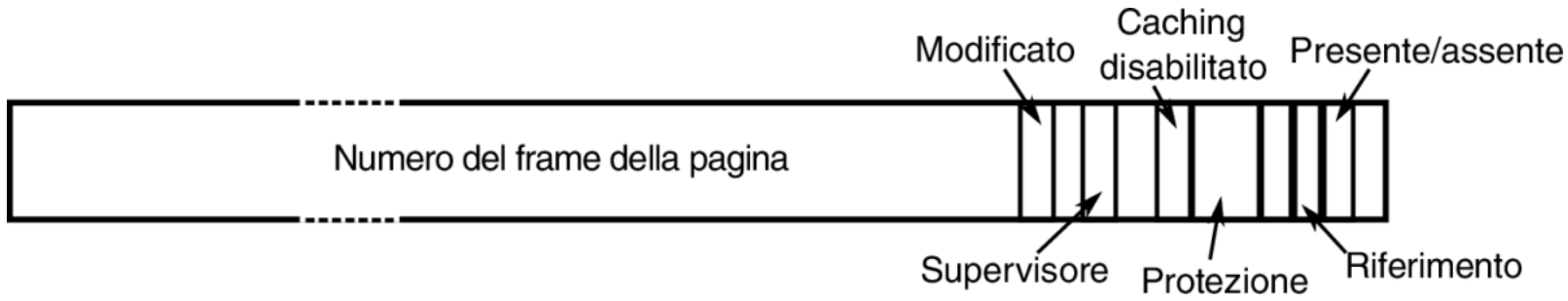
EVOLUZIONE DEGLI INDIRIZZI E TABELLA DELLE PAGINE



- **Indirizzi nei nostri esempi:** 16 bit (per chiarezza nelle illustrazioni)
- **Moderni PC:**
  - Usano indirizzi a 32 o 64 bit.
  - Con 32 bit e pagine da 4 KB:
    - 12 bit per indirizzare 4096 byte per pagina
    - Tabella delle pagine di $2^{(32-12)} = 2^{20}$ = 1.048.576 voci! Una taglia di 4GB è «fattibile» anche per PC con «pochi» GB di RAM.
- **Indirizzi a 64 bit e pagine da 4 KB:**
  - Richiede 2^{52} voci (~4,5 x 10^15) nella tabella.
  - In realtà nei sistemi a 64 bit si usano 48 bit
    - 256 Terabyte bastano e avanzano… gli altri bit sono riservati per il futuro …

COME È COMPOSTA UNA VOCE DELLA PAGE TABLE?

- Ogni voce ha informazioni cruciali come il numero del frame (es 12 bit per 4KB), come:
  - **Bit Presente/Assente**: Indica **se** la pagina virtuale **è in memoria**.
  - **Bit Protezione**: Specifica i **tipi di accesso consentiti** (lettura, scrittura, esecuzione).
  - **Bit Supervisor**: Stabilisce **se la pagina è accessibile** solo al sistema operativo o anche ai programmi utente.
  - **Bit Modificato (M) e Riferimento (R)**: Registrano l'uso della pagina.
    - Il bit Modificato si attiva quando la pagina viene scritta,
    - il bit Riferimento viene impostato ogni volta che si accede alla pagina.



- **Nota**: per un processo l'indirizzo in memora della «sua» tabella delle pagine è scritto nel registro Page Table Base Register (`PTBR`)

VELOCIZZARE LA PAGINAZIONE - PROBLEMI CHIAVE

- **Mappatura Veloce:** Necessaria a ogni riferimento alla memoria. Ogni istruzione può richiedere più riferimenti alla tabella delle pagine.
  - **Sfida:** Se un'istruzione impiega 1 ns, la ricerca nella tabella delle pagine deve essere inferiore a 0,2 ns per evitare colli di bottiglia.
- **Dimensione della Tabella delle Pagine:**
  - **Contesto:** Con 48 bit di indirizzamento e pagine di 4 KB, ci sono 64 miliardi di pagine. Una tabella delle pagine per questo spazio indirizzi richiederebbe voci enormi.
  - **Problema:** Usare centinaia di gigabyte solo per la tabella delle pagine è impraticabile. Ogni processo richiede una propria tabella delle pagine.

DOVE MEMORIZZARE LA TABELLA DELLE PAGINE?

- **Tabella delle Pagine in Registri Hardware:**
  - **Funzionamento:** Un registro hardware per ogni pagina virtuale, caricato all'avvio del processo.
  - **Vantaggi:** Semplice, non richiede accessi alla memoria durante la mappatura.
  - **Svantaggi:** Costoso con tabelle delle pagine grandi, ricaricare l'intera tabella ad ogni cambio di contesto è inefficiente.
- **Tabella delle Pagine in Memoria Principale:**
  - **Funzionamento:** La tabella delle pagine è interamente in RAM, con un registro che punta al suo inizio.
  - **Vantaggi:** Facile da cambiare a ogni cambio di contesto, richiede solo il ricaricamento di un registro.
  - **Svantaggi:** Richiede accessi frequenti alla memoria, rendendo la mappatura più lenta.

PROBLEMA DELLA PAGINAZIONE E TLB

- **Problema di Prestazioni nella Paginazione:**
  - **Ogni istruzione richiede l'accesso alla memoria** per prelevare l'istruzione stessa e un ulteriore accesso per la tabella delle pagine.
  - Raddoppio degli accessi alla memoria **riduce le prestazioni** di metà.
- **Ma …**
  - I programmi tendono a fare **molti riferimenti a un piccolo numero di pagine**.
  - **Solo una parte limitata** delle voci della tabella delle pagine viene **utilizzata frequentemente**.
- **Introduzione del *Translation Lookaside Buffer* (TLB):**
  - Dispositivo hardware che mappa indirizzi virtuali in fisici senza passare dalla tabella delle pagine.
  - Riduce gli accessi alla memoria durante la paginazione.

FUNZIONAMENTO E GESTIONE DEL TLB

- **Struttura:**
  - **Piccolo numero** di voci (es. 8-256), ciascuna con numero di pagina virtuale, **bit modificato**, **codice di protezione**, e frame fisico.
- **Funzionamento**:
  - Alla richiesta di un indirizzo virtuale, la MMU controlla prima nel TLB.
  - Se trovato e valido, il frame è prelevato direttamente dal TLB.
  - Se non trovato (`TLB miss`), avviene una ricerca normale nella tabella delle pagine e la voce trovata rimpiazza una voce nel TLB.
- **Gestione delle Modifiche**:
  - Le modifiche ai permessi di una pagina nella tabella delle pagine richiedono l'aggiornamento del TLB.
  - Per garantire la coerenza, la voce corrispondente nel TLB viene eliminata o aggiornata.

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

GESTIONE SOFTWARE DEL TLB

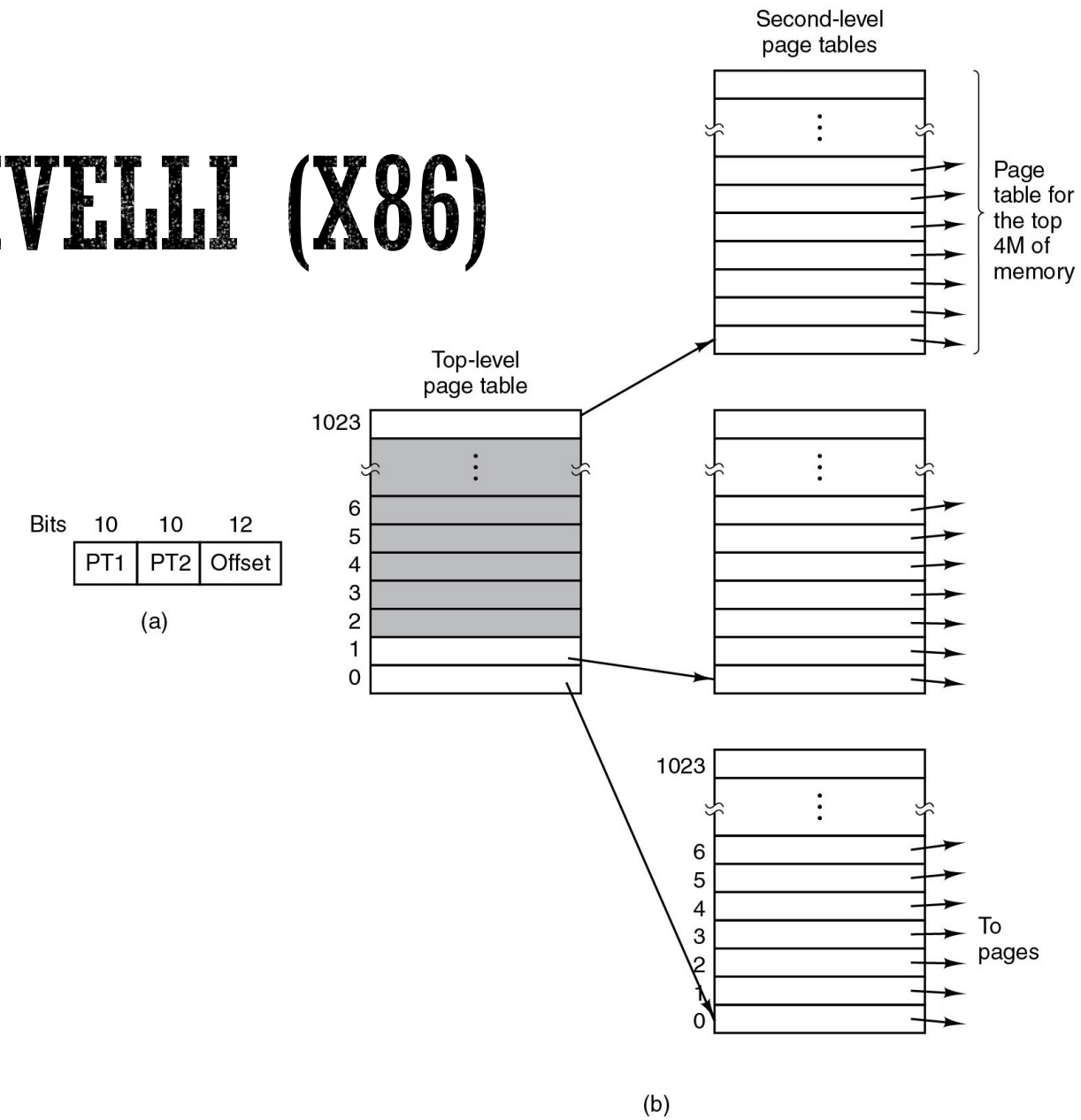
- **TLB in Architetture RISC:**
  - Alcune macchine RISC come SPARC, MIPS, e HP PA gestiscono le voci del TLB tramite software.
- **Processo in Caso di TLB Miss:**
  - Un TLB miss non porta a una ricerca automatica nella tabella delle pagine da parte della MMU.
  - Invece, si genera un errore di TLB e il sistema operativo deve intervenire.
  - Il sistema operativo cerca la pagina, aggiorna il TLB, e riavvia l'istruzione.

PAGE TABLE SIZES

- Poche slide fa: «*Con 32 bit e pagine da 4 KB*»:
  - 12 bit per indirizzare 4096 byte per pagina
  - tabella delle pagine di $2^{(32-12)} = 2^{20} = 1.048.576$ voci! Fattibile per PC con GB di RAM.
- Uno spazio di indirizzi virtuali molto grande porterebbe a una tabella di pagine molto grande
  - **Spreco di memoria** (senza contare cosa succederebbe per 64 bit!)
- **Possibili soluzioni**
  - Multi-level page table

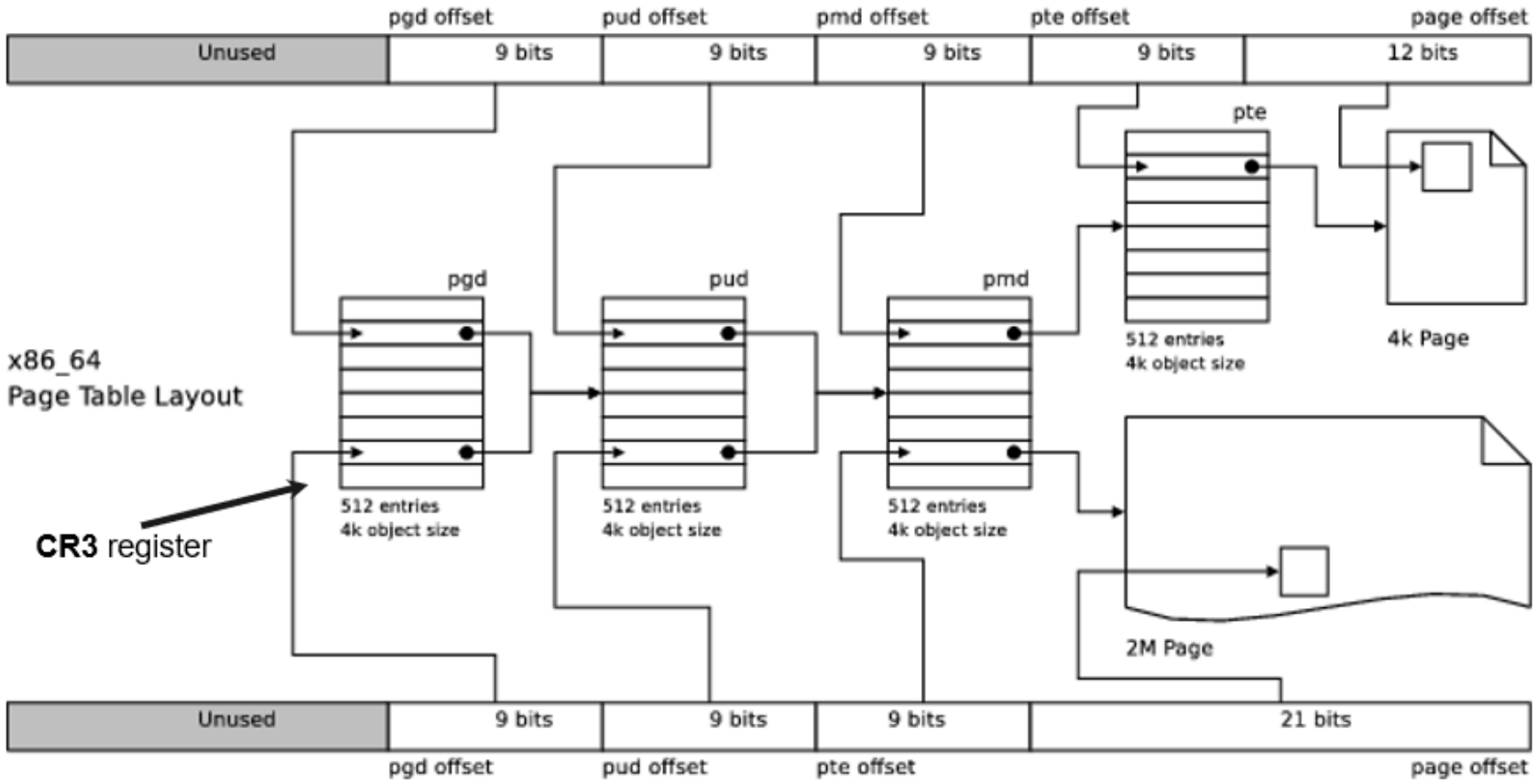
PAGE TABLE A DUE LIVELLI (X86)

- Le Page tables sono "attraversate" ("walked") dal Memory Management Unit
- **CR3** register
  - Registro speciale per puntare al vertice della gerarchia delle tabelle di pagina
- Esempio
  a) Un indirizzo a 32-bit con due campi
     - 10 +10 bit
  b) Una page table a due livelli



64 BIT: PAGE TABLE A 4 LIVELLI

- **PGD**: Page Global Directory
- **PUD**: Page Upper Directory
- **PMD**: Page Mid-level Directory
- **PTE**: page table entry



Nota: $2^9 \times 2^9 \times 2^9 \times 2^9 \times 2^{12} = 2^{48}$ byte. Ricordate i 48 bit? Permettono di puntare per il momento, 256 TB di memoria

TIPOLOGIE DI MISS E IMPLICAZIONI

- **Frequenza dei TLB Miss:**
  - I TLB miss **sono comuni a causa del numero limitato di voci** nel TLB (es. 64 voci).
  - **Aumentare la dimensione del TLB è costoso** e richiede compromessi nella progettazione dei chip.
- **Soft Miss vs Hard Miss:**
  - **Soft Miss:** La pagina è in memoria ma non nel TLB. Richiede solo l'aggiornamento del TLB.
  - **Hard Miss:** La pagina non è in memoria e richiede un accesso alla memoria non volatile (disco o SSD).
    - Un hard miss è significativamente più lento di un soft miss.
- **Page Table Walk e Diverse Tipologie di Miss:**
  - La ricerca nella gerarchia delle tabelle delle pagine è chiamata "**page table walk**".
  - I miss possono **variare in «gravità»** da minori (pagina in memoria ma non nella tabella delle pagine) a maggiori (pagina da caricare dalla memoria non volatile).
  - Un **accesso a un indirizzo non valido** può portare a un **segmentation fault** e alla **terminazione del programma**.