

Memoria virtuale e allocazione non contigua

Quando si parla di allocazione contigua ci si riferisce al fatto che informazioni contigue nello spazio logico del processo, sono allocate in modo contiguo anche nello spazio fisico nella memoria.

Con **allocazione non contigua** ci si riferisce invece ad un disallineamento tra i due spazi : quello del processo che parte da uno zero logico ad un certo limite viene sparpagliato all'interno della memoria fisica in locazioni non contigue.

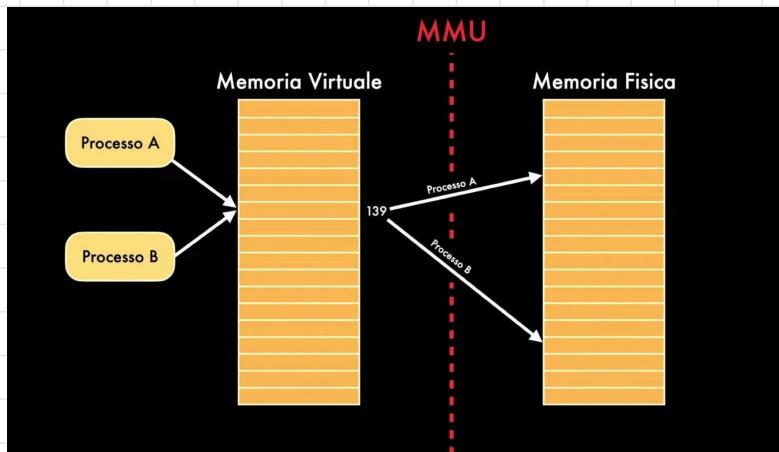
Quello che fino ad ora abbiamo chiamato spazio di indirizzamento logico , viene chiamato **virtuale**.

Il processo ha a disposizione il suo spazio di indirizzamento virtuale , dedicato al processo e potenzialmente illimitato (virtualmente). Questo spazio è suddiviso in **pagine** virtuali di dimensione fissa (solitamente 512 KB) , almeno nei sistemi più diffusi . Il meccanismo di impaginazione prevede che lo spazio di indirizzamento sia spezzettato in più pagine virtuali e solo alcune di queste sono impaginate realmente e memorizzate nella memoria fisica, anch'essa suddivisa in più pagine fisiche dette **frame**. La dimensione di tutti i frame è uguale a quella delle pagine virtuali. Un frame è solo uno slot idoneo a memorizzare una pagina qualsiasi e questo **grado di libertà** di immagazzinare pagine di qualsiasi processo è il centro del meccanismo. Due pagine virtuali che contengono due porzioni della stessa procedura cadono su pagine virtuali contigue ma non c'è nessuna contiguità a livello fisico. Grazie a questo grado di libertà non si presenta più il problema della frammentazione esterna, perché se vi è una richiesta di allocazione di un tot di pagine da immagazzinare , esse potranno essere memorizzate in qualsiasi slot senza alcun limite di contiguità. Ovviamente le pagine virtuali relative ai vari processi poi verranno mescolate tra i frame fisici.

Protezione implicita

Questo modello introduce implicitamente un meccanismo di protezione, in quanto il codice di un determinato processo, qualunque indirizzo generi , sarà interpretato come un indirizzo tra il suo zero logico ed il suo massimo (solitamente 2^{32} o 2^{64} nei casi reali) e **non ci sarà mai la possibilità di collisione con lo stesso indirizzo generato da un altro processo**.

Per fissare le idee i processi P1 e P2 possono eseguire un'operazione con riferimento alla locazione di memoria virtuale 1000 ma verranno tradotte differentemente dall'hardware in due indirizzi fisici differenti. Infatti l'MMU è la componente hardware che decide quali sono i range all'interno del quale si può muovere il meccanismo software.



In questo modello alcune pagine potrebbero non essere presenti in RAM in quanto non vi è spazio sufficiente.

Queste pagine vengono memorizzate in memoria secondaria, ma i processi che si componevano di quelle pagine possono continuare ad essere **eseguiti** nonostante dei pezzi (pagine) del loro codice e dati siano parcheggiati su disco (differentemente da quanto succede con lo swapping dove un intero processo viene spostato su disco e non più eseguibile).

Tabelle delle pagine

Attraverso la **tabella delle pagine**, l'MMU riesce a tenere conto di dove ha parcheggiato la pagina su disco. Queste sono delle strutture dati, locate in RAM ed implementate per ogni processo che hanno un **numero di voci pari al numero di pagine** dello spazio di indirizzamento virtuale del processo, ognuna delle quali mappa un indirizzo virtuale di una pagina del processo su un indirizzo fisico della memoria fisica (le frecce). Ogni frame è numerato e gli identificativi di ogni slot vengono detti **numeri di frame** ed ogni voce della tabella delle pagine contiene informazioni come il numero di frame fisico corrispondente all'indirizzo virtuale e informazioni sullo stato della pagina, come ad esempio se la pagina è stata caricata in memoria fisica o se si trova ancora sul disco (bit di presenza).

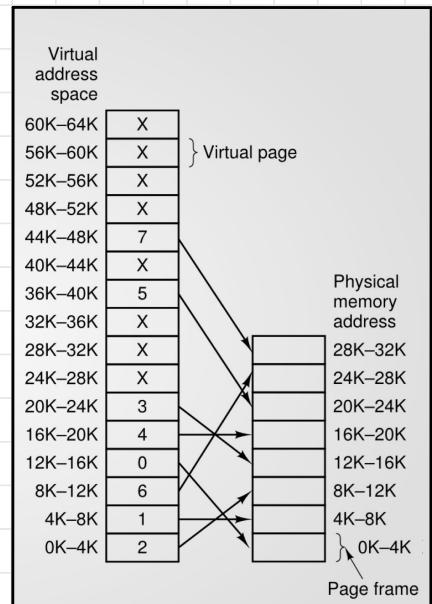
Quando il processo tenta di accedere ad un indirizzo virtuale, l'MMU del processore cerca nella tabella delle pagine la corrispondenza tra l'indirizzo virtuale richiesto e l'indirizzo fisico della pagina corrispondente. Se la pagina è già stata caricata in memoria fisica, l'MMU effettua la mappatura e consente al processo di accedere alla pagina richiesta. Se la pagina non è ancora presente in memoria fisica si ha un evento di page fault (la X), nel quale l'MMU genera un interrupt per il sistema operativo, che inizierà il processo di caricamento della pagina in memoria fisica e l'aggiornamento della tabella delle pagine.

Page fault

Se la pagina non presente serve al processo viene generato un indirizzo virtuale come gli altri e si avrà un evento di page fault, non è da intendere come errore ma come un evento dovuto alla configurazione del processo in quel momento. Questo evento indica che la pagina richiesta dal processo non è attualmente presente in RAM e deve essere caricata dalla memoria secondaria prima che il processo possa accedervi. La gestione del page fault avviene nel sistema operativo, che prenderà la pagina richiesta dal processo dalla memoria secondaria e la caricherà in un frame libero nella memoria fisica. In questo modo, il processo potrà accedere alla pagina richiesta dalla memoria fisica e proseguire la sua esecuzione. Durante questo processo di swap tra la memoria fisica e la memoria secondaria, il sistema operativo potrebbe dover liberare uno o più frame nella memoria fisica per fare spazio alla nuova pagina richiesta dal processo.

Questa gestione del page fault proseguirà con la **reiterazione dell'istruzione** che aveva generato l'evento di page fault, che stavolta potrà proseguire correttamente con la propria esecuzione. Nelle tabelle delle pagine, in ogni voce, è presente un **bit di presenza** in ogni record della tabella delle pagine, oltre al numero che identifica il corrispondente frame della RAM, che identifica se la pagina si trova su disco o se è già disponibile in memoria.

In questo esempio la dimensione delle pagine (quindi anche quella dei frame) è di 4KB. Con 64KB di spazio degli indirizzi virtuali e 32KB di memoria fisica otteniamo 16 pagine virtuali e 8 frame fisici. L'intervallo 0-4K indica che gli indirizzi fisici o virtuali in quel frame o in quella pagina vanno da 0 a 4095, ovvero sono 4096. Il secondo gruppo invece va da 4096 a 8K-1 e così via.

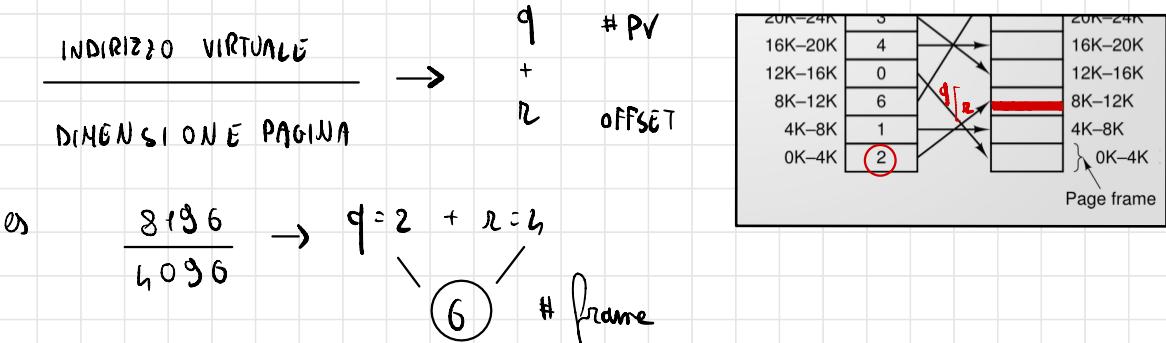


Traduzione

L'MMU riceve le richieste di operazioni con riferimento a determinati indirizzi virtuali e le inoltra alla RAM con riferimento ad indirizzi fisici, quindi traduce gli indirizzi virtuali in indirizzi fisici.

-Dato un indirizzo virtuale come capisco in che frame fisico questo ricade dopo la traduzione ?

L'MMU opera la traduzione da indirizzo virtuale, che viene generato dalla CPU, ad indirizzo fisico nel seguente modo : prende l'**indirizzo virtuale** e lo divide sulla **dimensione dei frame** (uguale alla dimensione di pagina) ottenendo un quoziente ed un resto . Il quoziente intero identifica il **numero di frame** in cui si trova quell'indirizzo virtuale, ovvero l'identificativo del gruppo in cui sono ricaduto con la traduzione. In particolare mi viene restituito il primo indirizzo del frame e il resto della divisione mi darà un **offset**, il quale sommato al numero di frame mi darà l'esatta word che rappresenta l'indirizzo fisico.



Basta moltiplicare l'identificativo del gruppo per la dimensione del frame e sommare l'offset per risalire all'indirizzo virtuale.

$$(\# \text{ frame} \cdot \text{DIMENSIONE FRAME}) + \text{offset} = \text{Indirizzo Virtuale}$$

L'MMU svolge questa traduzione degli indirizzi virtuali in indirizzi fisici in modo trasparente per il programma in esecuzione, garantendo che l'accesso alla memoria virtuale avvenga in modo efficiente e **senza conflitti** tra processi. Anche il program counter lavora con indirizzi virtuali e sarà l'MMU a tradurre gli indirizzi utilizzati, con riferimento alle relative istruzioni da eseguire, in indirizzi fisici, ed anche la CPU rimane ignara di questa traduzione.

Questa traduzione in caso di **page fault** non fornisce alcun risultato in quanto il frame in cui si troverebbe l'indirizzo non è presente in memoria . Si necessita di bloccare tutto con un IRQ, si carica dalla memoria secondaria il frame desiderato e si prosegue con la reiterazione del processo di traduzione che stavolta fornirà un risultato concreto.

Operazioni bitwise

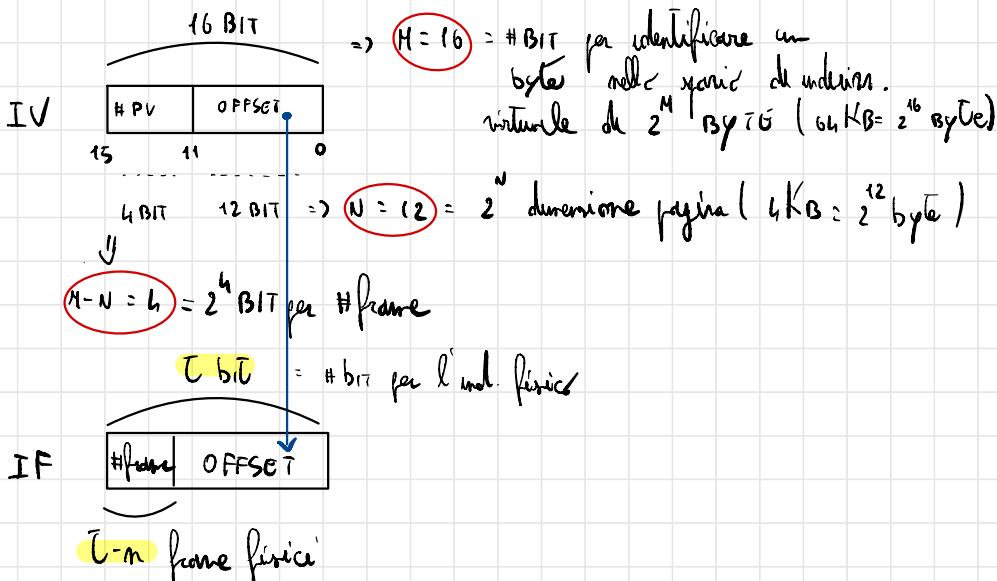
Se la dimensione della pagina virtuale (quindi anche del frame) è una potenza del due, questa traduzione diventa un calcolo più diretto ed efficiente da parte dell'MMU.

Normalmente la traduzione comporterebbe di dividere il numero di bit utilizzati per identificare un indirizzo virtuale per la dimensione della pagina, che essendo una potenza di 2 equivale ad effettuare uno shift logico a sinistra di N posizioni dell'indirizzo logico di M bit, ricordando che la **dimensione della pagina è di 2^N** e che **M è il numero di bit usati per identificare un singolo byte** nello spazio di indirizzamento. Le cifre meno significative scartate con lo shift a sinistra rappresentano l'offset. Quindi operare una maschera di bit in tal senso coincide con l'applicare un taglio di N posizioni, mentre le M-N posizioni più significative sono un numero a M-N bit che identifica il numero di pagina virtuale.

Nell'esempio si aveva un indirizzo fisico a 16 bit, perchè se si considera il log base due dello spazio di indirizzamento, nell'esempio 64K si ottiene 16 (infatti $2^{16} = 64K$), vuol dire che con un indirizzo a 16 bit ($M = 16$) riesco ad identificare ogni byte del mio spazio di indirizzamento. La dimensione di una singola pagina è di 4KB ovvero 2^{12} ($N = 12$).

Nell'esempio N è 12 (dimensione della pagina di 4KB ovvero 2¹² byte). Il taglio nell'esempio è fatto a 12 bit perchè la dimensione di pagina è 4096 bit, ovvero 2¹².

Concretamente operando lo shift logico si è effettuata una divisione per 2¹² e le dodici cifre meno significative del numero che ho ottenuto sono tutte nulle, per tanto sommare l'offset equivale ad eseguire un OR bit a bit che consentirà di trovare l'esatta word di riferimento (in pratica si ricopiano le cifre meno significative).



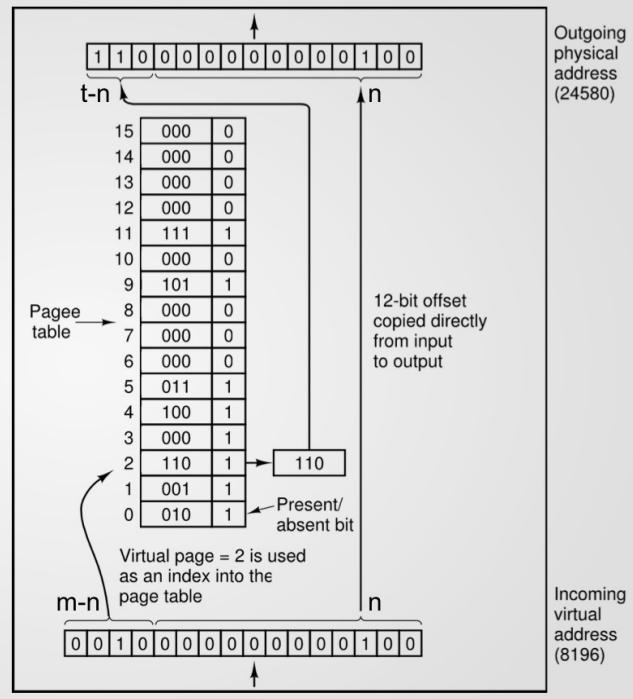
Uno spazio di indirizzamento a M bit ha una capienza di 2^M byte. La dimensione n dell'offset è in relazione alla dimensione della pagina, nell'esempio 2^{12} byte. La dimensione in bit del numero di pagina virtuale è data ovviamente da m-n e mi dà informazione su quante pagine devo gestire, ovvero $2^{(m-n)}$ pagine. Considerato che l'indirizzo fisico verrà tradotto in un numero da t bit, allora $2^{(t-n)}$ sono i frame all'interno della memoria fisica e si ha **m > t** sempre. Un ulteriore informazione, ovvero la **dimensione massima della memoria RAM gestibile dal sistema** è pari a 2^t byte.

Uso di una tabella delle pagine

- spazio indirizzi virtuali: 2^m
- dim. pagina: 2^n

→ numero di pagina: $(m-n)$ bit più significativi dell'indirizzo virtuale
→ 2^{m-n} pagine
→ offset: n bit meno significativi

- spazio indirizzi fisici: 2^t
→ 2^{t-n} frame
→ $m > t$



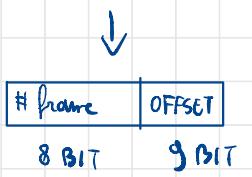
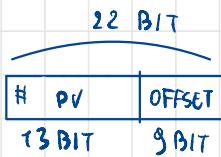
1) INPUT:

- Spazio di mem. richiesto = 4 MB
 - Nella Tabella delle pagine ho 2^{13} voci
 - # frame = 8 BIT
- Quanta RAM posso supportare al massimo?

$$4 \text{ MB} = 2^{22} \text{ Byte} \quad \text{quindi } M = 22 \text{ BIT}$$

$$2^{13} \text{ voci} \Rightarrow M-N = 13 \quad \text{ovvero } N = 22-13 = 9 \text{ BIT}$$

frame = 8 BIT significa che ho 2^8 frame
ognuna da 2^9 byte



$$\text{quindi } l = 8 + 9 = 17$$

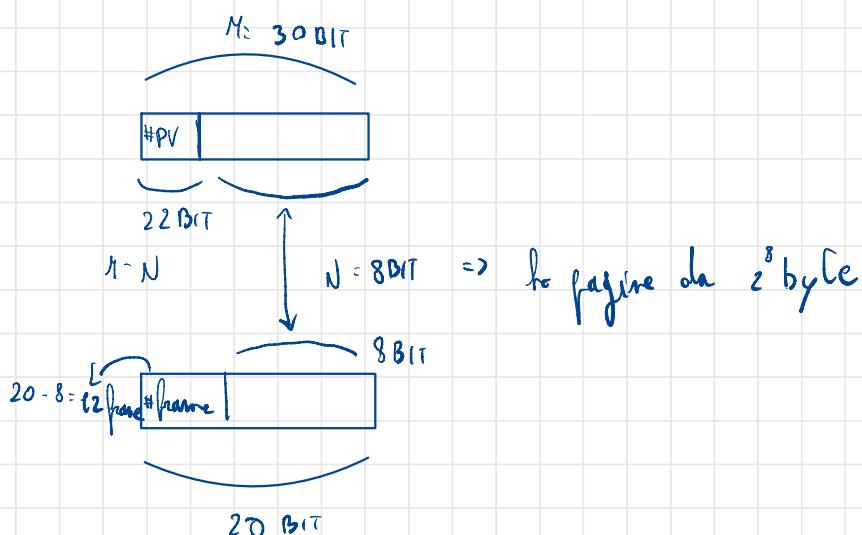
$$\Rightarrow 2^{17} \text{ Byte di RAM} \approx 128 \text{ KB}$$

2) INPUT

- Spanne der mem. mithilfe = 1 GB
- 2^{30} Bytes
- HPV = 22 Bit
- lmb. fizisch = 20 Bit
- 2^{20} Bytes \rightarrow 1 MB
- Anzahl Frame existenz?

$$1 \text{ GB} = 2^{30} \text{ Bytes} \Rightarrow N = 30$$

$$20 \text{ Bit} = 0$$



\Rightarrow No 12 frame da 2⁸ bytes

Dettaglio su una voce della tabella delle pagine

Oltre al bit di presenza/assenza e al numero del frame, nel generico record di pagina virtuale vi sono altri campi :

- **Numero frame**
- **Bit di presenza**
- **Protezione** : Lo spazio di indirizzamento virtuale può essere marcato con tipi di permessi diversi ad esempio lettura e scrittura. Rappresentato da tre bit, ovvero R W ed X (esecuzione). Il bit di scrittura è posto a zero per tutte le pagine del codice, rendendole quindi in sola lettura allo scopo di poter condividere memoria. Se ho due processi con lo stesso codice (due istanze dello stesso programma) in RAM andrò a caricare una sola copia del codice e sarà condivisa da entrambi i processi, si rendono le pagine del codice in sola lettura per evitare che uno dei due processi modifichi il codice visto dall'altro processo per non provocare anomalie. Il risparmio di memoria è ovvio ed è dato dal fatto che istanzio un solo processo al posto di due copie dello stesso processo. I permessi di scrittura sono invece attivati nell'heap e nello stack. Il terzo bit di esecuzione rappresenta un permesso di esecuzione che viene posto a zero. Il SO al momento dell'istanziazione del processo assegna gli opportuni permessi, mentre l'MMU gestisce questi bit in quanto riceve le opportune informazioni nella richiesta di traduzione. Prima ancora di fare la traduzione va a consultare la maschera dei permessi e se riscontra un conflitto genera un errore fatale, il processo viene segnalato la sua esecuzione viene interrotto. Quindi la gestione dei permessi è a carico della componente hardware MMU.
- **Dirty bit** : è un bit di stato che identifica se la pagina che ho in RAM differisce dalla copia che ho nel disco, ovvero se è stata modificata. Quando una pagina viene spostata su disco e ricaricata in RAM, la copia su disco rimane ed in questo caso il bit di modifica o dirty bit è utile perché se la copia che è presente in RAM non è stata modificata e ho necessità di ripostarla su disco, posso semplicemente scarstrarla e non ho un costo dovuto al processo di ricopiare sul disco (che contiene la copia che non era stata modificata). Ovviamente nel caso in cui la pagina era stata modificata, la copia sul disco non sarà in uno stato aggiornato e quindi si necessita di ricopiarla. Compito dell'MMU è mantenere coerente con lo stato del processo questo bit di modifica.
- **Bit di referenziamento** : questo bit deve essere posto a 1 se la pagina è stata referenziata da qualunque operazione di lettura o scrittura. Ovviamente una pagina appena caricata in RAM sarà posto a 0. Questo bit diventa una sorta di statistica a breve termine in quanto l'MMU ogni tot di millisecondi azzera tutti i bit di referenziamento, consentendo di poter fare una statistica di quali pagine sono state usate negli ultimi tot millisecondi. Viene utilizzato per prendere una decisione su quale pagina spostare du disco , ovviamente verrà scelta una pagina il cui bit di referenziamento è posto a zero.
- **Bit di cache** : consente di disabilitare la cache per la pagina. Questa caratteristica è importante per le pagine che mappano sui registri dei dispositivi piuttosto che in memoria. Se il sistema operativo è posizionato in un ciclo stretto in attesa che qualche dispositivo di I/O risponda a un comando appena inviato, è fondamentale che l'hardware continui a prelevare la parola da quel dispositivo e non usi quella vecchia messa in cache. Con questo bit, l'utilizzo della cache può essere disabilitato.
- **Bit di validità** : indica che la pagina è allocata per il processo se posto a 1. Identifica una porzione di memoria virtuale che fa parte di un processo ma che non è stata allocata, quindi se si fa accesso l'MMU genera un'eccezione in quanto quella pagina non contiene nulla di allocato per processo.

