Approccio con Memoria virtuale

Consiste nel consentire a più programmi di essere caricati anche **PARZIALMENTE** nella memoria.

> *I programmi richiedono una dimensione superiore a quella che si ha realmente (fisicamente)*

I piccoli pezzi parziali di programmi sono detto ***OVERLAY***. All'inizio parte il gestore di overlay che carica e avvia l'overlay 0, poi l'overlay 1 e così via..

Problema: l'idea è quella di mantenere pezzi di programma in memoria ma gli overlay si trovano nel disco (operazioni `disco->memoria` **frequenti**) ma il limite principale è che gli overlay venivano **gestiti dal programmatore** e per questo motivo gli **overlay richiedono più tempo** e quindi si è più soggetti a **errori**.

L'idea è di affidare la gestione degli overlay ***AL SISTEMA OPERATIVO***. Si assegna uno spazio degli indirizzi personale a ogni programma. Ogni pezzo di programma è detto ***PAGINA***

> *La memoria virtuale prende l'idea precedente e la affida al SO*

Pagina

La pagina rappresenta **insieme di indirizzi *CONTIGUI* mappati sulla memoria fisica**. Più pezzi dello stesso programma possono essere mappati su memoria fisica ***ANCHE NON CONTIGUA*** quindi i vari pezzi del programma si possono trovare **sparsi per la memoria fisica**.

- Lo spazio di indirizzamento virtuale viene assegnato a ogni programma ed è diviso in pagine. Ogni pagina è associata ad un indirizzzo fisico
- Il SO (in particolare la component `MMU`) farà la ***traduzione da indirizzo logico(virtuale)->indirizzo fisico*** e funziona nei sistemi moderni dove si usa un *sistema multiprogrammato* (*più pezzi di programmi in memoria allo stesso momenti*)
- Lo spazio degli indirizzi virtuale è diviso in blocchi di ***DIMENSIONI FISSE***. Ogni blocco è detta ***PAGINA***.
- Le unità (blocco di memoria fisica) corrispondenti della memoria fisica vengono chiamate ***FRAME***.
- Le *pagine* e i *frame* hanno la **STESSA DIMENSIONE**, in genere, *MA* il *numero di pagine* è **SUPERIORE** al numero di frame

  > *Quindi gli indirizzi virtuali devono essere di più rispetto al numero di indirizzi fisici altrimenti questo discorso non avrebbe senso*

- Si ha una **protezione implicita fra processi**: nessun processo ha modo di **accedere** a parti di memoria che **NON gli sono state assegnate**, o che non possiede.

Si ha una netta separazione fra la ***memoria virtuale dell'utente*** e fra ***quella fisica effettiva***. L'utente vede un unico spazio di memoria che contiene il programma considerato ma in realtà si riesce a gestire l'esecuzione del programma dell'utente dividendo il programma in pezzi sparsi per la memoria.

Lo ***spazio virtuale degli indirizzi*** è l'*insieme* degli indirizzi virtuali generati *dal programma*.

Paginazione

La paginazione è il meccanismo sopra descritto.

> *Quando si parlava* `enter_region` *e* `leave_region` *nei processi, c'era l'istruzione* `MV R2, #1000` *che sono istruzioni macchina che copiano il contenuto di* `1000` *nel registro* `R2`*.*

Il **programma genera da sè** gli indirizzi virtuali e foma lo spazio degli indirizzi virtuali. Gli indirizzi virtuali vengono inviati al MMU che li traduce in indirizzo fisico e lo invia all'unità di elaborazione.

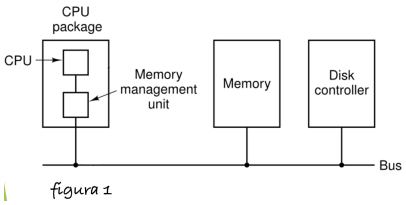


figura 1

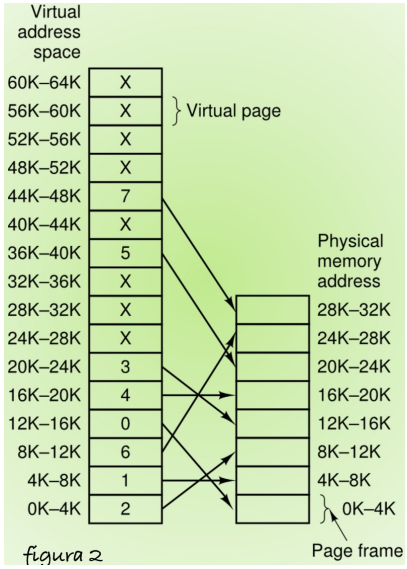


figura 2

Nell'esempio sopra in figura:

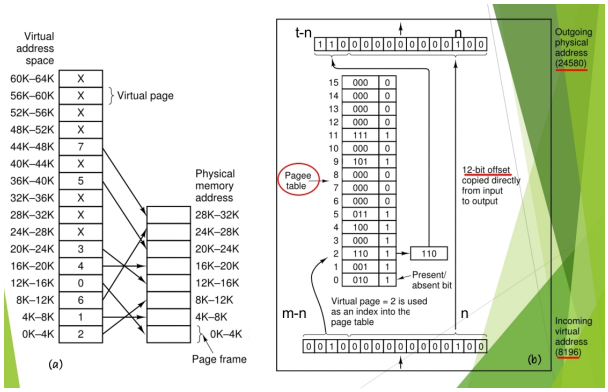
- Si hanno 16 pagine virtuali e 8 pagine fisiche, e ogni cella è grande $4K$
- Se ho `MOV REG #0`, si accede all'indirizzo virtuale `0` e viene spedito a MMU e si nota che corrisponde il ***frame 2*** nella memoria fisica e corrisponde, quindi, **all'inizio** del frame 2 (perchè non ho nessuno sfasamento, o ***offset***)
  - *quindi all'indirizzo virtuale 0 corrisponde l'indirizzo fisico 8K* (*inizio della frame*)
    - Inizio della pagina = Inizio del frame (*se non c'è offset*)
- Può capitare che una ***pagina virtuale non sia assegnata ad alcun frame fisico*** (si vede la X al posto di un numero), quindi si ha necessità di ***trovare un modo per tenere traccia di quello che è associato a un frame*** per distinguerlo da quando non lo è.
  - In questo caso si usa un bit (***BIT DI PRESENZA/ASSENZA***) che, settandolo a 0 o a 1, si usa per capire se una pagina è associata a un frame oppure no.
- Ad ogni processo si associa una ***TABELLA DELLE PAGINE*** e contiene: *il bit di presenza/assenza*, *frame associato* (*se associato a un frame*)
- Un frame allocato ad una pagina di **un processo**, ovviamente, **non può essere allocata** a una pagina **di un altro processo.**

Esempio (*vedi* `figura2`)

Data l'istruzione `MOV REG #32780`:

- L'indirizzo deve essere preima tradotto
- *32780* appartiene al blocco *32K-36K*
- MMU riceve l'indirizzo, accede alla tabella degli indirizzi associata al processo.

- Si accede *all'ottava* pagina corrispondente e accede al bit presenza/assenza e ne controlla il valore. In questo caso è 0 (X).
- In questo caso MMU genera una ***TRAP*** per chiamare il Sistema Operativo e questo evento viene detto **PAGE FAULT** (*Errore di Pagina*)
- Il SO associa un frame alla pagina che non la ha. **Associa** la pagina a qualcosa di **libero** oppure effettua una **sostituzione di pagina**.
  - Se c'è una ***pagina di lettura*** (es: *ascolto musica*) allora si può rimuovere.
  - Se c'è una ***pagina di lettura/scrittura***, allora i dati devono ***PRIMA essere scritti in memoria*** e poi può essere rimossa.



- Ogni pagina è grande 4K.
- Per identificare univocamente le pagine virtuali servono **16 bit** mentre per quelli fisici servono **15 bit**
- MMU riceve indirizzo virtuale (16 bit). I tot bit vengono suddivisi in **2 parti** non uguali:
  - ***I più significativi*** (4) indicano il **numero di pagina** che accede alla pagina stessa
  - ***i meno significativi*** (12) vengono detti ***OFFSET*** e sono messi da parte
- Verifica il bit di presenza/assenza
- Per esempio ho `0010` che corrisponde alla pagine di indice *2*. Controllo bit di preseza e trovo 1, prendo i bit `110` e **li aggancio all'offset** e trovo l'indirizzo fisico (vedi disegno figura `b)`)

In generale:

- spazio indirizzi virtuali: 2^m
- dimensione pagina: 2^n

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

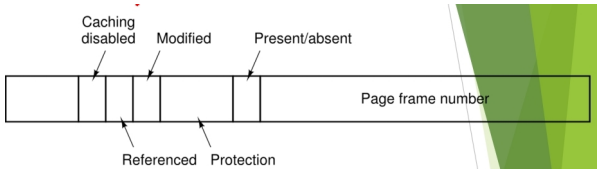
- spazio indirizzi fisici: 2^t

  → 2^{t-n} frame
  → $m > t$

> La suddivisione può essere differente in base alle dimensioni delle pagine. **Lo SCOPO** della tabella delle pagine è mappare le pagine virtuali in frame delle pagine.

Dettaglio su una voce della tabella delle pagine

Le informazioni su una pagina servono per gestire nel migliore dei modi la rimozione di una pagina per dare spazio ad un'altra ed eventualmente per altre operazioni.



- `Numero del frame`: indica il numero del frame associato a quella pagina (la mappatura dalla quale si ricava l'indirizzo fisico)
- bit `presente/assente`
- bit `modificato` (***DIRTY BIT***): indica che è a *1* quando il contenuto di una pagina è modificato rispetto alle modifiche presenti in memoria in quell'istante
- bit `referenziato`: indica quando una **pagina è stata referenziata/utilizzata**. Vale *1* quando si fa riferimento ad un'altra pagina. E' sempre 1 se la pagina è modificata e viceversa.
  - i bit `referenziato` e `modificata` servono per capire quando una pagina è *usata e modificata*
  - Una pagina chiamata in causa per operare comporta il bit `referenziato` a 1. Se poi deve essere modificata, ciò comporta il bit `modificato` a 1.
- bit di `protezione`: identifica la ***tipologia di accesso*** che si può fare alla pagina: *lettura/scrittura/esecuzione*
  - Recentemente questo campo è formato da 3 bit e quindi permette le combinazioni lettura/scrittura, lettura/esecuzuione ecc ecc..
- bit per `disabilitare la cache` per la specifica pagina e si usa quando si lavora sui ***registri dei dispositivi***. In particolar modo quando le pagine vengono mappate ***direttamente*** sui registri dei *dispositivi I/O* e serve per ***evitare accessi inutili alla cache*** proprio perchè non viene *mai chiamata in causa*
- bit di `validità` (o di `allocazione`): indica se la pagina è in uno spazio di indirizzi valido oppure no. Serve anche per ***impedire/permettere l'accesso alla pagina di riferimento***.
  - Se è 0 (non valido) allora la pagina non è associata nello spazio degli indirizzi logici di quel determinato processo. Consegue che le attività vengono bloccate perchè non si ha accesso a quella pagina (perchè *non valida*).
- Le pagine che sono state **usate nel breve** intervallo di tempo, con molta probabilità verranno usate di nuovo **nel breve futuro** e, seguendo questa logica, si decide quale pagina rimuovere dalla tabella in caso di sostituzione.
- Si usano i bit `modificato` e `referenziato` per capire se una pagina è stata usata di recente o no.

> *Con questo modo di agire non si annulla il page fault ma si riduce la probabilità che esso accada*