Memoria virtuale

Introduzione

Paginazione

Fault

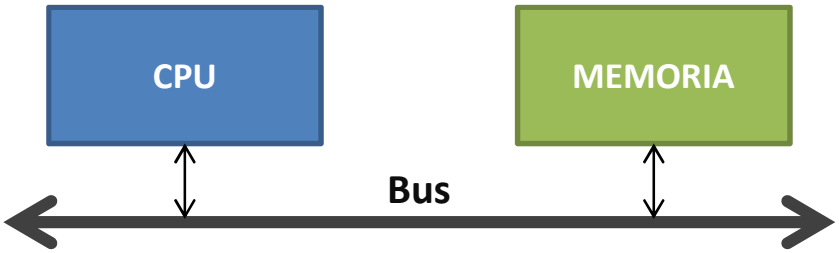
Introduzione

- **La memoria di un calcolatore sottende un modello lineare**
  - La memoria è un insieme ordinato di celle contigue
- **Dimensione delle celle**
  - La dimensione delle celle è sempre di 1 byte, cioè 8 bit
  - Spesso è possibile accedere alla memoria per gruppi di 2, 4 o 8 byte consecutivi
    - Tali gruppi sono detti "parole"
  - In genere l'accesso alle parole avviene in modo "allineato"
    - L'indirizzo d'inizio di una parola è un multiplo della sua dimensione
- **Dimensione della memoria**
  - La dimesnione della memoria indica il numero di byte che essa contiene
    - Non di parole!
  - La dimensione è sempre una potenza del 2
  - La dimensione si esprime in multipli del byte
    - Kbyte 2^{10} = 1,024 byte
    - Mbyte 2^{20} = 1,048,576 byte
    - Gbyte 2^{30} = 1,073,741,824 byte

(c) 2012 - Carlo Brandolese

Introduzione

- **In un sistema di calcolo general purpose, la memoria**
  - E' esterna al microprocessore
  - E' connessa a questo mediante un "bus"



- **Per poter accedere a tutte le celle della memoria**
  - E' necessario riferirvisi per inidirizzo
  - L'indirizzo (in binario) deve essere grande a sufficienza per riferirsi a tutte le celle
- **A determinare la dimensione dell'indirizzo sono principlamente**
  - La dimensione di parola del microprocessore
  - La dimensione del bus indirizzi del sistema
- **Nessuno di questi fattori è strettamente vincolante**
  - Vi sono strategie per "comporre" diverse parti a formare un indirizzo completo

(c) 2012 - Carlo Brandolese

Introduzione

- **In molti sistemi la dimensione degli indirizzi è fissata**
  - E in genere indipendente dalla largezza del bus e della parola del microprocessore
- **Ad esempio**
  - Sistema con indirizzi a 16bit
    - Possibilità di indirizzare $2^{16} = 2^6 \times 2^{10}$ byte = 2^6 Kbyte = 64 Kbyte
  - Sistema con indirizzi a 24bit
    - Possibilità di indirizzare $2^{24} = 2^4 \times 2^{20}$ byte = 2^4 Mbyte = 16 Mbyte
  - Sistema con indirizzi a 32bit
    - Possibilità di indirizzare $2^{32} = 2^2 \times 2^{30}$ byte = 2^2 Gbyte = 4 Gbyte
- **Tuttavia i sistemi spesso dispongono di una quantità di memoria inferiore a quella potenzialmente indirizzabile**
- **Sorge quindi una necessità**
  - Svincolare la dimensione di un indirizzo dalla quantità di memoria fisica disponibile
- **Il programmatore può "immaginare" di avere a disposizione tutti gli indirizzi**
  - Sarnno l'hardware e il sistema operativo a gestire questa situazione

(c) 2012 - Carlo Brandolese

Memoria fisica e memoria virtuale

- **Memoria fisica**
  - Numero di byte della memoria effettiavemente disponibile sul sistema

- **Memoria logica o virtuale**
  - Numero di byte indirizzabili da una parola di microprocessore
  - Se la parola del microprocessore è di N bit
    - Si possono generare 2^N indirizzi diversi
    - Si dice che lo spazio di indirizzamento è di 2^N celle
  - Lo spazio di indirizzamento costituisce una memoria "virtuale" appunto

- **Dato che deve sempre essere possibile accedere a tutta la memoria fisica**
  - Lo spazio di indirizzamento virtuale è maggiore o uguale alla memoria fisica

- **Il modello di memoria virtuale fornisce una astrazione**
  - Della memoria fisica
  - Indipendente dalla reale dimensione della memoria fisica
  - Il modello di riferimento del programmatore

(c) 2012 - Carlo Brandolese

Memoria fisica e memoria virtuale

- **In altre parole, la virtualizzazione della memoria**
  - Fornisce al programmatore un modello secondo cui tutto lo spazio di indirizzamento consentito dalla dimensione della parola è disponibile
  - Il prgramma, dunque, fa riferimento a indirizzi "virtuali" o "logici"
- **Gli indirizzi virtuali**
  - Sono generati dal linker alla fine del processo di compilazione
  - Iniziano tutti da un valore fisso, diciamo 0, per semplicità
- **All'atto del caricamento di un programma in memoria, si hanno sue situazioni**
  - In presenza di memoria virtuale
    - Ogni processo ha il suo spazio di indirizzamento
    - Tutti i processi possono essere caricati a partire dall'indirizzo logico 0
  - In assenza di memroia virtuale
    - Il programma andrà a trovarsi ad un indirizzo fisico in generale diverso da 0
    - Per risolvere questo problema il sistema operativo deve "aggiornare" gli indirizzi in base alla posizione effettiva del codice
    - Questo processo prende il nome di "rilocazione"
- **Gli indirizzi generati dal processo di rilocazione**
  - Sono ancora indirizzi fisici
  - Il processo di rilocazione avviene anche in presenza di memoria virtuale

(c) 2012 - Carlo Brandolese

Memoria fisica e memoria virtuale

- **Per accedere effettivamente all ememori afisica del clacolatore**
  - E' necessario disporre di un meccansimo di traduzione
    - Da indirizzi logici
    - A indirizzi fisici
- **Quest'operazione**
  - E' svolta da un insieme di componenti
    - Hardware: MMU (Memory Management Unit)
    - Software: Il gestore della memoria virtuale nel sistema operativo
  - Prende il nome di "memory mapping"
- **Grazie alla combinazione di rilocazione e memory mapping**
  - Un indirizzo logico può sempre essere tradotto automaticamente in uno fisico
  - Il programmatore può
    - Disinteressarsi della posizione reale in memoria del proprio programma
    - Immaginare di disporre di una memoria grande quanto tutto lo spazio di indirizzamento
- **Il sistema operativo**
  - Può caricare un programma ovunque il memoria (rilocazione)
  - Può modificare dinamicamente la memoria assegnata al processo

(c) 2012 - Carlo Brandolese

Memoria fisica e memoria virtuale

- **Dal momento che**
  - Un programma può indirizzare una memoria viruale più grande di quella fisica
  - Più programmi possono essere caricati in memoria contemporaneamente

- **Se ne deduce che**
  - Un programma in esecuzione non risiede, in generale, completamente nella memoria fisica del sistema

- **Ciò comporta una ulteriore complicazione del sistema di gestione della memoria**
  - Parti di uno stesso programma devono essere caricate e scaricate dinamicamente
    - Dal sistema operativo
    - In modo trasparente al programmatore
    - In base alle esigenze di quel preciso momento

(c) 2012 - Carlo Brandolese

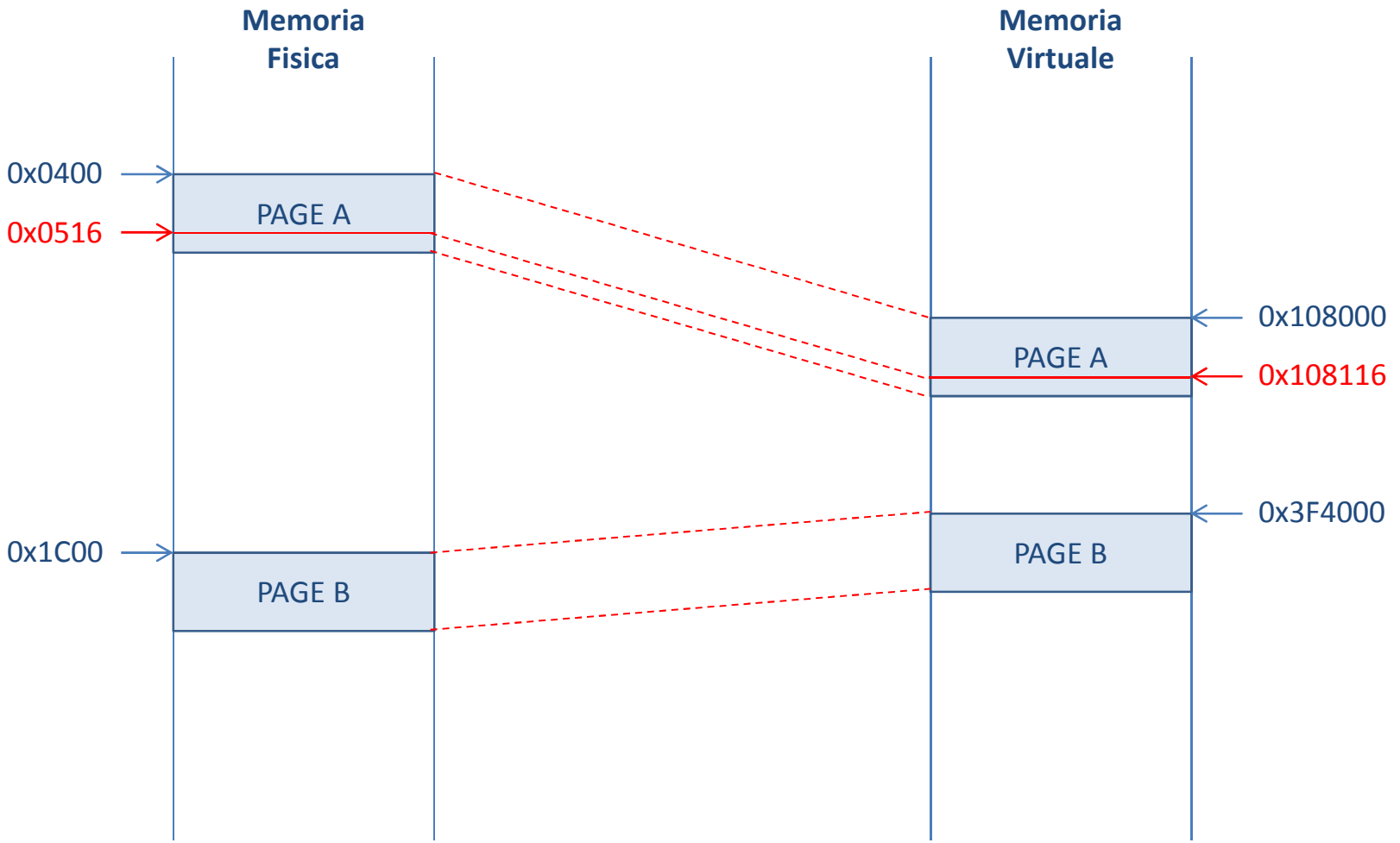
Paginazione

- **La soluzione si basa sul concetto di pagina**
  - Una pagina è una parte di memoria di dimensione fissa
    - Tipicamente da 1Kbyte a 64Kb
- **La memoria fisica e la memoria virtuale**
  - Sono viste come una sequenza di pagine contigue
- **Esempio**
  - Dimensione pagina: 1Kbye
  - Memoria fisica:
    - Dimensione: 64Kbyte
    - Indirizzi: 16 bit
    - Numero di pagine: 64K / 1K = 64
  - Memoria virtuale:
    - Dimensione 16MByte
    - Indirizzi: 24 bit
    - Numero di pagine: 16M / 1K = 16K

(c) 2012 - Carlo Brandolese

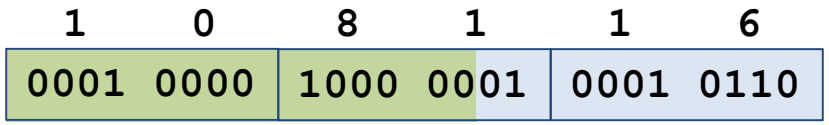
Paginazione

- Visione della memoria



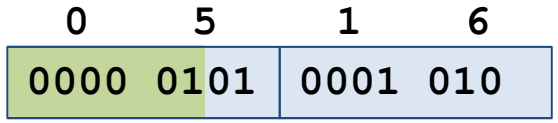
Paginazione

- **Abbiamo quindi, riferendoci alla pagina A**
  - Indirizzo virtuale d'inizio: 0x108000
  - Indirizzo virtuale del dato: 0x108116
  - Indirizzo fisico d'inizio: 0x0400
  - Indirizzo fisico del dato: 0x0516
- **Consideriamo l'indrizzo del e scomponiamolo in due parti**
  - Numero di pagina (verde) e offset (azzurro)
  - Considerando gli indirizzi virtuali, si ha



Numero di pagina: 00 0100 0010 0000 = 0x0420
Offset: 01 0001 0110 = 0x116

  - Mentre nel caso di indirizzi fisici, si ha

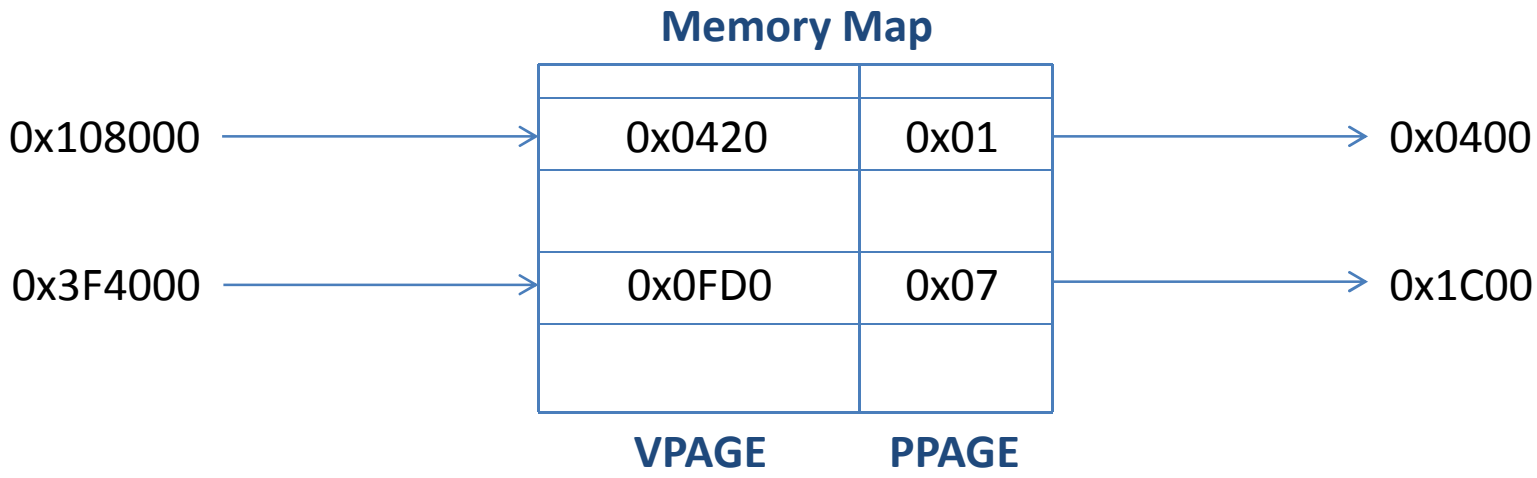


Numero di pagina: 00 0001 = 0x01
Offset: 01 0001 0110 = 0x116

(c) 2012 - Carlo Brandolese

Paginazione

- **Si nota che**
  - L'offset all'interno delle pagine sia locgica che fisica è identico
  - Il numero di pagina logica e il numero di pagina fisica differiscono
- **Per poter quindi associare un indirizzo logico ad uno fisico**
  - E' necessario disporre di una mappa che, per ogni pagina allocata, associ
    - Il numero di pagina logica al numero di pagina fisica
- **Nel nostro caso, considerando entrambe le pagine dell'esempio avremmo**



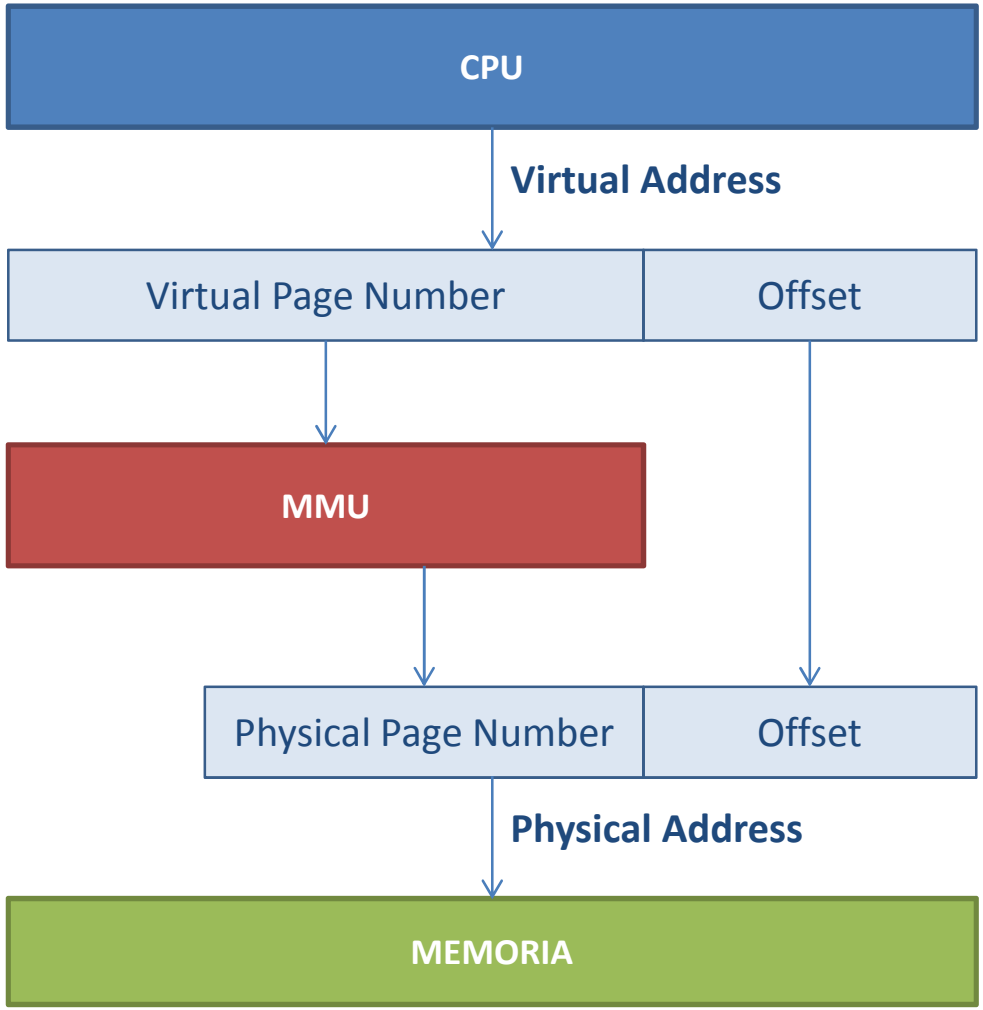
Paginazione

- **Dato quindi un indirizzo logico**
  - Si spezza tale indirizzo in due campi VPAGE e OFFSET
  - Si individua nella memory map la riga corrispondente all pagina VPAGE
  - Si preleva dalla memory map la corrispondente PPAGE
  - Si combina la PPAGE con il campo OFFSET ricavato all'inizio
- **Il meccanismo, seppur semplice**
  - Richiede diverse operazioni
  - Deve essere trasparente al programmatore
- **Se fosse totalmente a carico del sistema operativo**
  - Ogni accesso in memoria richiederebbe diverse istruzioni assembly
- **E' necessario disporre di un'apposita unità hardware preposta allo scopo**
  - Si tratta della MMU o Memory Management Unit
  - E' una memoria associativa
  - Prende in ingresso un indirizzo virtuale
  - Produce in uscita un indirizzo fisico
- **La MMU deve essere "configurata" con una specifica mappa di memoria**

(c) 2012 - Carlo Brandolese

Paginazione: Memory Management Unit

- Possiamo vedere la MMU in questo modo



(c) 2012 - Carlo Brandolese

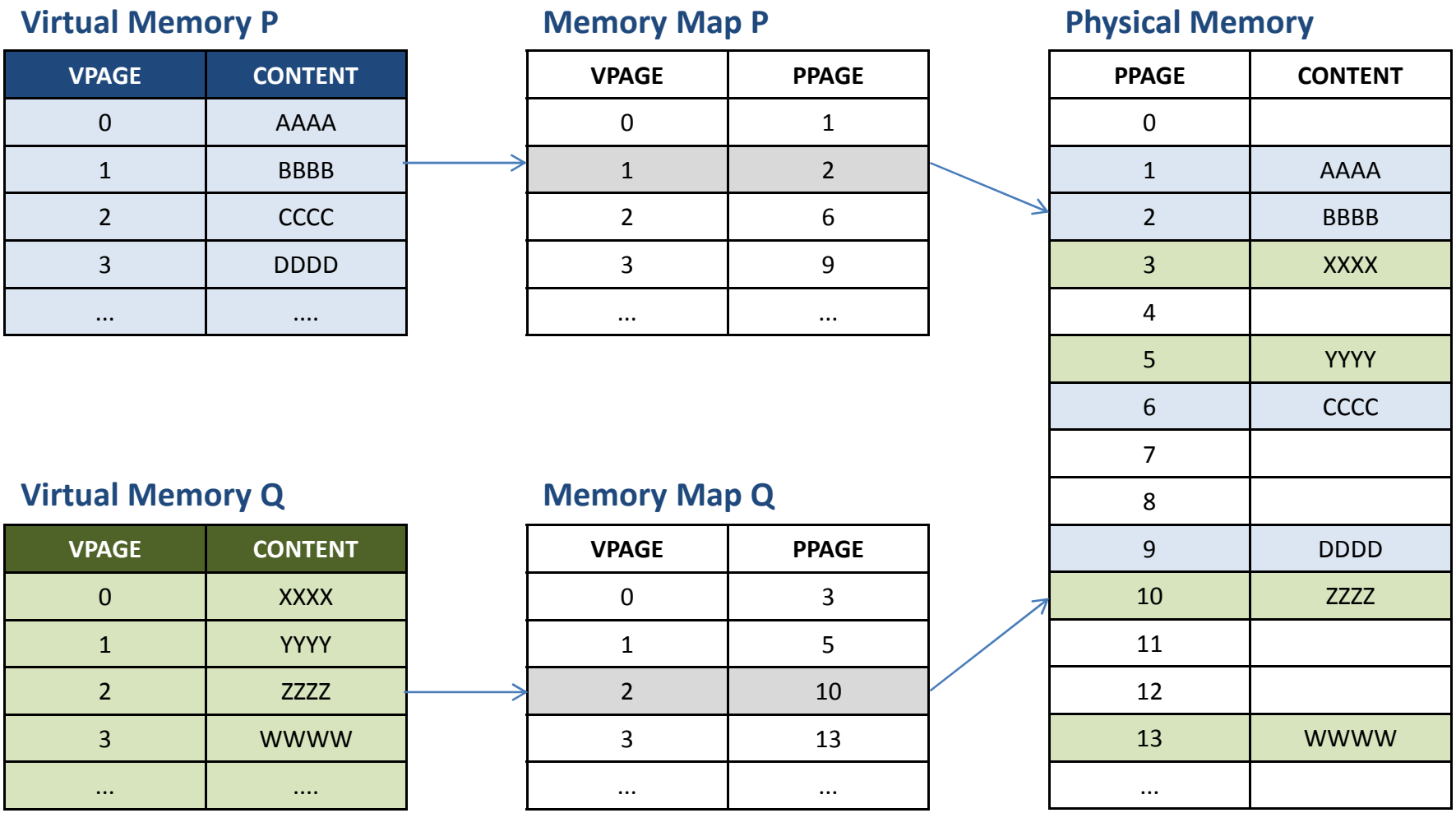
Memory Management e processi

- **Affinché il porgrammatore possa riferirsi alla memoria virtuale**
  - Ogni processo deve disporre di tutto lo spazio di indirizzamento
- **Un processo occuperà solo una parte esigua di memoria fisica**
  - Rispetto alla dimensione della memoria virtuale
- **In generale quindi un processo sarà costituito**
  - Da un insieme contiguo di pagine logiche
  - Da un insieme arbitrariamente distribuito di pagine fisiche
- **Affinché la combinazione tra MMU e concetto di processo sia consistente**
  - Ogni processo deve disporre di una propria mappa di memoria
  - Nel momento in cui il processo entra in esecuzione
    - La MMU deve essere configurata con la mappa del processo corrente
  - In questo modo le pagine logiche del processo in esame vengono associate alle corrispondenti pagine fisiche

(c) 2012 - Carlo Brandolese

Memory Management e processi

- **Esempio: Due processi P e Q**



(c) 2012 - Carlo Brandolese

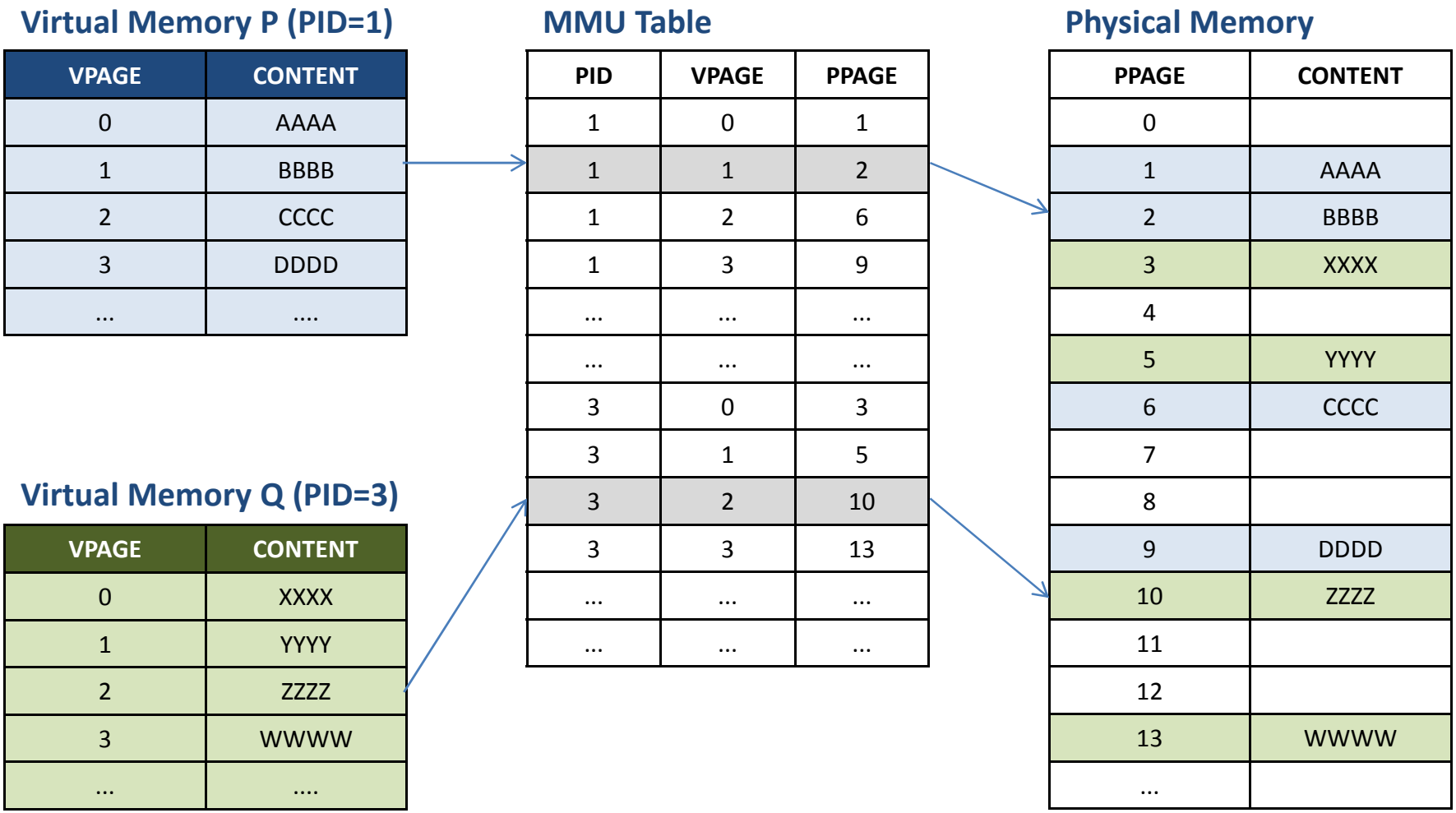
Memory Management e processi

- **Le tabelle delle pagine di ogni processo**
  - Devono contenere, in linea di principio, una riga per ogni pagina della memoria virtuale
  - Una tale tabella può essere molto grande
    - Difficile gestione
- **La tabella utilizzata per configurare la MMU (detta MMU Table)**
  - Contiene, in genere, solo una parte della tabella delle pagine di ogni processo
    - Sarebbe opportuno contenesse i riferimenti delle pagine utilizzate più spesso
  - La MMU contiene in genere parte delle tabelle di più processi
- **Ne consegue**
  - Il numero di pagina virtuale VPAGE
    - Non può essere utilizzato come indice della tabella
    - Deve in realtà essere un campo di una opportuna struttura dati
  - La memoria che contiene la MMU table
    - Deve poter accedere alla tabella corretta in base al processo corrente
    - Il PID del processo è un'altro campo di tale struttura

(c) 2012 - Carlo Brandolese

Memory Management e processi

- Una sola MMU table per più processi



(c) 2012 - Carlo Brandolese

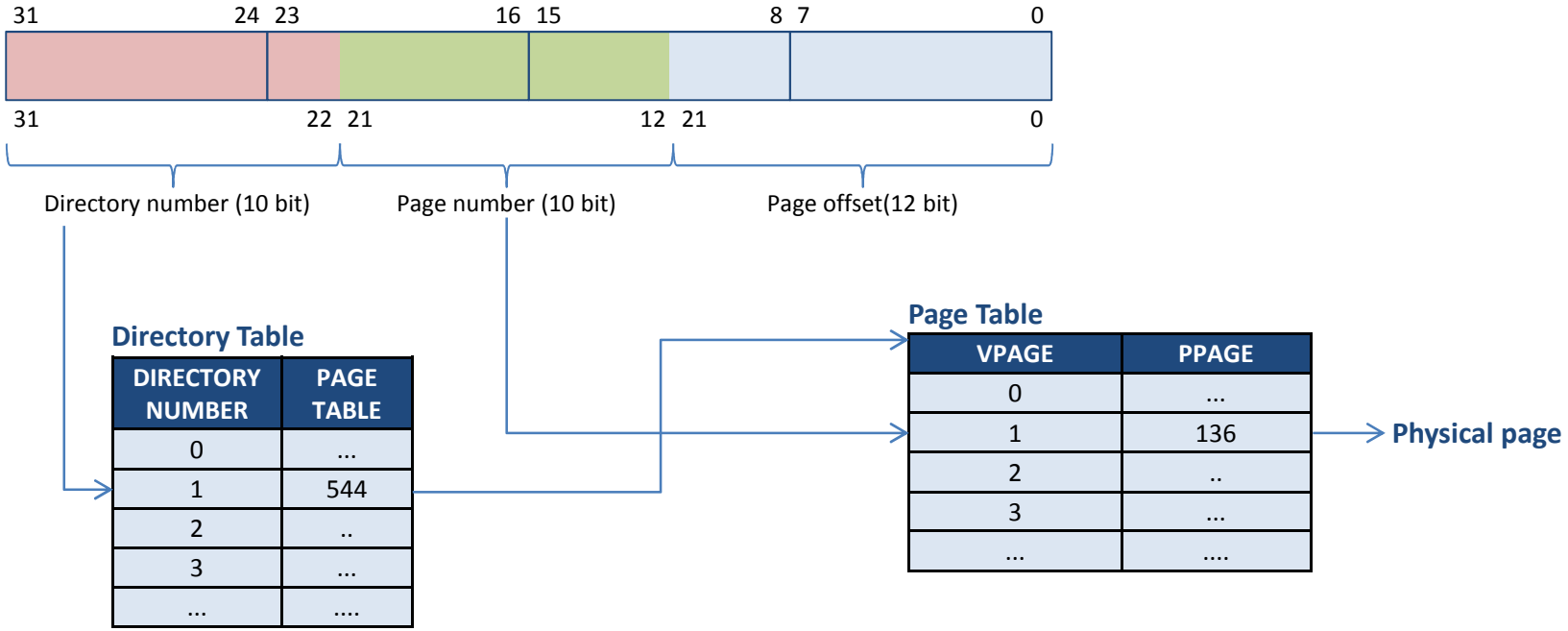
Paginazione: Struttura della tabella

- **Consideriamo un caso realeistico**
  - Memoria virtuale 4Gbyte
  - Memoria fisica 512Mbyte
  - Dimensione pagina 4Kbyte
- **Ne consegue**
  - Indirizzo virtuale 32 bit
  - Indirizzo fisico 29 bit
  - Offset di pagina 12 bit
  - Numero di pagine logiche $2^{32} / 2^{12} = 2^{20}$
  - Numero pagine fisiche $2^{29} / 2^{12} = 2^{17}$
- **La dimensione complessiva della tabella sarebbe**
  - Dimensione di una riga 4 byte
  - Dimensione della tabella 4×2^{20} = 4Mbyte = 1K pagine fisiche
- **Tale dimensione**
  - E' eccessiva, a maggior ragione in quanto è relativa ad ogni processo (vedi oltre)
  - Utilizzando una hash-map, la ricerca sarebbe troppo lenta e complessa

(c) 2012 - Carlo Brandolese

Paginazione: Struttura della tabella

- **Linux ricorre ad una struttura gerarchica**
  - Sull'esempio della struttura delle directory, ma limitata a due livelli
- **L'idea è quella di scomporre un indirizzo logico in tre parti**
  - Page directory
  - Page number
  - Page offest
- **La figura seguente schematizza tale soluzione per l'esempio in esame**



(c) 2012 - Carlo Brandolese

Table miss

- **Abbiamo detto che il contenuto della tabella associativa della MMU**
  - Consiste solamente in una parte della tabella delle pagine di un processo
- **La tabella delle pagine di un processo**
  - E' memorizzata in una struttura dati del Sistema Operativo
    - Può assumere grandi dimensioni
- **La dimensione della tabella associativa della MMU**
  - E' in generale molto ridotta rispetto alla dimensione delle tabelle delle pagine
- **Poiché la tabella associativa contiene solo una parte della tabella delle pagine**
  - E' possibile che venga richiesto l'accesso ad una pagina virtuale non attualmente presente nella tabella associativa della MMU
  - In questo caso si parla di "table miss"
    - Una porzione della MMU table deve essere aggiornata
  - Un numero elevato di table miss comporta una perdita di efficienza
- **Per ridurre la probabilità di table miss**
  - E' necessario che la dimensione della porzione di MMU table associata ad un processo si avvicini il più possibile al numero R di pagine del processo residenti in memoria
    - Vedi oltre per la definizione di R e del working set

(c) 2012 - Carlo Brandolese

Condivisione delle pagine

- **Alcuni processi possono condividere alcune informazioni**
  - Codice, si pensi alla fork()
  - Dati, si pensi a due processi con memoria condivisa
- **Quindi alcune pagine della memoria fisica sono associate a più di un processo**
  - Nella memoria fisica si ha una sola pagina
  - Nella memoria virtuale si ha una pagina per ogni processo che la condivide
- **Nella MMU table, quindi**
  - Ad una pagina condivisa saranno associate tante righe quanti sono i processi associati
  - I numeri delle pagine virtuali saranno in generale diversi da processo a processo
  - Il numero di pagina fisica è sempre lo stesso
    - Ciò realizza di fatto la condivisione
- **Dal punto di vista della MMU**
  - Questa situazione è identica alla gestione di pagine non condivise
  - La traduzione degli indirizzi avviene in modo identico

(c) 2012 - Carlo Brandolese

Protezione delle pagine

- **Il meccanismo di paginazione**
  - Consente di rilevare, durante l'esecuzione, accessi a zone di memoria
    - Che non appartengono allo spazio di indirizzamento virtuale del processo in esecuzione
    - Sulle quali non è consentito compiere una data operazione (lettura, scrittura, esecuzione)
  - Ciò si verifica quando viene generato un numero di pagina virtuale che non esiste nella tabella delle pagine del processo
    - In questo caso dobbiamo fare riferimento alla tabella completa, non alla MMU table
- **Quando ciò accade la MMU o il sistema operativo**
  - Generano un interrupt di violazione di memoria
- **Al fine di migliorare la protezione**
  - E' possibile associare ad ogni pagina virtuale di un processo alcuni bit di protezione
  - Tali bit definiscono le modalità di accesso consentite
    - Lettura (R)
    - Scrittura (W)
    - Esecuzione (X)
  - Il tipo di operazione richiesta e i permessi associati alla pagina determinano l'evetuale violazione ed il conseguente interrupt

(c) 2012 - Carlo Brandolese

Page fault

- **Ricordiamo che la memoria fisica è**
  - Limitata
  - Comunque più piccola della memoria virtuale
- **Può quindi accadere che non tutte le pagine necessarie possano risiedere nella memoria fisica contemporaneamente**
  - In questo caso il sistema operativo provvede a liberare alcune pagine di memoria fisica, salvandone il contenuto su disco
    - Si dice che si esegue uno "swap-out" delle pagine
  - Indicando nella tabella delle pagine e nella MMU table che la pagina non è più in RAM
    - A tale scopo si utilizza un flag di validità detto "valid bit"
- **Quando viene richiesto l'accesso ad una data pagina virtuale**
  - Si verifica che il "valid bit" sia ad 1, cioè che la pagina sia in memoria
    - In tal caso tutto procede come visto finora
  - Se il "valid bit" indica che la pagina non è in memori si ha un "page fault"
- **In corrispondenza di un page fault**
  - Il processo viene sospeso in attesa che la pagina venga caricata nuovamente
    - Si dice che la pagina subisce uno "swap-in"

(c) 2012 - Carlo Brandolese

Page fault

- **Durante l'esecuzione di un processo**
  - Solo un numero limitato di pagine virtuali è presente nella memoria fisica
  - Chiamaiamo tali pagine "residenti"
- **In caso di page fault**
  - Uno specifico interrupt passa il controllo al sistema operativo
  - Il processo in esecuzione viene interrotto
  - Il controllo viene passato al sistema operativo
- **A questo punto il sistema operativo deve**
  - Individuare su disco la pagina virtuale richiesta
    - A tale scopo è necessario ricorrere alle tabelle complete delle pagine
    - La tabella delle pagine contiene anche un riferimento alla posizione su disco
  - Trovare uno spazio disponibile in memoria per caricare la pagina richiesta
    - Cio può richiedere di "liberare" memoria scaricando un'altra pagina (swap-out)
  - Caricare la pagina da disco
    - Swap-in
  - Richiedere nuovamente l'esecuzione dell'istruzione che aveva generato il page fault

(c) 2012 - Carlo Brandolese

Page fault

- **Si hanno quindi due problemi**
  - Caricare una pagina in memoria
    - Risolto grazie all'informazione relativa alla posizione su disco presente nella tabella delle pagine
  - Liberare spazio nella memoria fisica per caricare una nuova pagina
    - Il sistema operativo deve scegliere una pagina su cui eseguire lo swap-out
- **Per la scelta della pagina si utilizzano due ulteriori bit**
  - Associati ad ogni pagina
- **Access bit**
  - Viene posto a 0 quando non appena la pagina è caricata in memoria
  - Viene posto a 1 ogni volta che viene richiesto un accesso alla pagina
    - Utile per decidere quale pagina scaricare
- **Dirty bit**
  - Viene posto a 0 quando non appena la pagina è caricata in memoria
  - Viene posto a 1 quando si accede in scrittura ad una parola della pagina
  - Il valore di tale bit
    - Permette di decidere se aggiornare la copia su disco di una pagina al momento dello swap-out
    - Se vale 0, nulla è stato modificato dall'ultimo caricamento per cui la copia su disco è superflua

(c) 2012 - Carlo Brandolese

Sostituzione delle pagine

- **Si deve adottare una politica per la scelta della pagina da scaricare**
  - Molti algoritmi possibili
- **I pù comunemente utilizzati sono i seguenti**

- **Politica "random"**
  - Si sceglie una pagina a caso

- **Politica "least recently used" (LRU)**
  - Si sceglie la pagina utilizzata meno di recente
    - E' più probabile che non appartenga più al working set
  - Si veda la descrizione nella slide seguente

- **Politica "first in first out" (FIFO)**
  - Si sceglie la pagina caricata meno di recente
    - E' una politica semplice ma non tiene conto degli accessi effettivi

(c) 2012 - Carlo Brandolese

Sostituzione delle pagine

- **Algoritmo LRU**
  - Utilizza l'access bit e il dirty bit memorizzati nella tabella delle pagine
- **Per misurare l'"invecchiamento" di una pagina si gestisce l'access bit come segue**
  - Al caricamento della pagina il bit viene posto a 0
  - Ad ogni accesso bit viene posto a 1
  - Periodicamente il sistema operativo riporta il bit a 0
- **Algoritmo LRU semplice**
  - Si sceglie una delle pagine con access bit pari a 0
  - Quelle con access bit pari a 1 sono state accedute più di recente
    - Precisamente nell'ultimo periodo di "reset" periodico
- **Algorimo LRU avanzato**
  - Il sistema operativo mantiene un contatore per ogni pagina
  - Prima dell'azzeramento periodico il sistema operativo incrementa tale contatore per tutte le pagine che hanno access bit pari a 0
  - Si sceglie la pagina con access bit pari a zero e valore del contatore più alto

(c) 2012 - Carlo Brandolese

Working set

- **Il problema del page fault impatta significativamente sulle prestazioni**
  - Richiede accessi ad un supporto di memorizzazione di massa
  - Le memorie di massa sono migliaia/milioni di volte più lente della memoria centrale
- **Si è verificato sperimentalmente che per un programma valgono due principi**
  - Località spaziale
    - Elevata probabilità che un accesso a memoria sia ad un indirizzo vicino a quello dell'ultimo accesso effettuato
  - Località temporale
    - Elevata probabilità un accesso a memoria sia ad un indirizzo cui si è acceduto recentemente
- **Si definisce "working set" di ordine k**
  - L'insieme delle pagine utilizzate negli ultimi k accessi a memoria
- **Grazie ai principi di località e per k sufficientemente grande**
  - Il working set cambia molto lentamente nel tempo
    - Il valore di k dipende dal programma
  - Mantenedo in memoria fisica le pagine del working set, si riduce significativamente il numero di page fault
    - Si ha dunque un impatto minore sulle prestazioni

(c) 2012 - Carlo Brandolese

Working set

- **La dimensione del working set è indicata con R (numero di pagine)**
- **Nella scelta del parametro R si hanno due esigenze contrastanti**
  - Minimizzare i page fault, aumentando R
  - Minimizzare il numero di pagine residenti in memoria, dimunuendo R
- **Fissato R, si hanno due possibili situazioni**
  - Durante la normale esecuzione del programma
    - Si avrà un numero di page fault limitato ma dipendente dalla bontà della scelta di R
  - All'inizio dell'esecuzione di un programma
    - Si avrà una serie di page fault
    - Ad ogni page fault verrà caricata una nuova pagina
    - Dopo il caricamento di R pagine si giunge a regime
- **La sequenza di caricamento iniziale delle pagine è detta di "demand paging"**
  - Una pagina viene caricata quando si tenta di accedervi per la prima volta
  - Anche detto "lazy loading"
- **Un'alternativa consiste nel caricare subito tutte le pagine del programma**

(c) 2012 - Carlo Brandolese