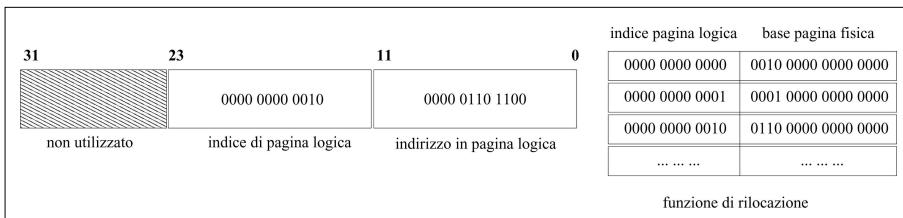


## Esercizio: memoria virtuale

Siano dati un indirizzo logico con la struttura ed il contenuto mostrati in figura, dove è anche riportata la funzione di rilocazione. Si indichi l'indirizzo fisico corrispondente all'indirizzo logico mostrato in figura e degli indirizzi logici  $4000_{10}$  e  $6000_{10}$ .



## Soluzione

Rappresentando i tre valori di indirizzo logico dati in formato esadecimale ed interpretandoli secondo la struttura dell'indirizzo logico mostrata in precedenza:

indice pagina logica	indirizzo in pagina logica	base pagina fisica corrispondente	indirizzo fisico risultante
2	06C	6000	606C
0	FA0	2000	2FA0
1	770	1000	1770

## Esercizio: memoria virtuale

Sia dato un calcolatore con memoria virtuale paginata (dimensione pagina 8KB), con spazio di indirizzamento su 48 bit e memoria principale con spazio di indirizzamento su 32 bit.

Si calcoli:

1. L'ampiezza complessiva della tabella di gestione delle pagine, assumendo un costo di 8B per pagina;
2. Se  $P$  è la percentuale dello spazio di memoria virtuale occupato dalla tabella di gestione delle pagine, si calcoli l'ampiezza  $A$  della porzione di tabella ospitabile in memoria principale che causa la stessa percentuale  $P$  di occupazione della memoria fisica;
3. Si dica se  $A$  è sufficiente per permettere alla porzione di tabella in memoria fisica di contenere la totalità delle pagine presenti in memoria fisica.

## Soluzione

1. Spazio indirizzamento logico  $2^{48}B = 2^{18}GB$ ; una pagina contiene  $8KB = 2^{13}B$ , quindi il numero di pagine dello spazio logico è  $2^{48} / 2^{13} = 2^{48-13} = 2^{35} = 2^5 GB = 32 GB$ . Poiché ogni riga della tabella occupa  $8B = 2^3$ , la tabella occupa  $2^{35}2^3 = 2^{38} = 2^8 GB = 256 GB$ .
2.  $P = 256 GB / 2^{18}GB \times 100 = 2^{-10}B \times 100$ , quindi  $A = 2^{-10}2^{32} = 2^{22} = 4MB$ .
3. La memoria principale ha  $2^{32}B = 4GB$ , cioè un numero di pagine pari a  $2^{32} / 2^{13} = 2^{32-13} = 2^{19} = 2^9 KB = 512 KB$ . La porzione di tabella in memoria fisica ha dimensione  $4MB$  e quindi può contenere  $4MB/8B = 512 KB$  righe, cioè quanto basta per contenere la totalità delle pagine presenti in memoria fisica.

## Esercizio: memoria virtuale

Si consideri un sistema di gestione della memoria virtuale basato su paginazione. Si discuta, in non piu' di 5 righe, la congettura secondo la quale l'aumento di capacita' della memoria fisica comporti una diminuzione delle occorrenze di accesso a pagina mancante (page fault). Si discuta ora cio' che accade nella situazione descritta di seguito e lo si metta in relazione con la congettura di cui sopra:

Una memoria fisica M1, inizialmente vuota, ha capacita' pari a 3 pagine, a fronte di una memoria virtuale V di ampiezza 5 pagine, numerate da 0 a 4. Il gestore della memoria virtuale adotta la politica di rimpiazzo detta LRU (least recently used) implementata in versione a basso costo (lazy), dove cioe' non si aggiorna la lista ordinata per frequenza d'accesso a seguito di un accesso avvenuto con successo (hit).

L'esecuzione del programma causa la sequenza di chiamata di pagine:

**0,1,2,3,0,1,4,0,1,2,3,4.**

Si mostri l'effetto di tale sequenza sul contenuto di M1, indicando ove essa provochi page fault. Si mostri poi l'effetto della medesima sequenza sulla memoria fisica M2, inizialmente vuota, e di capacita' pari a 4 pagine.

## Soluzione

Per quanto intuitivamente plausibile, non è sempre il caso che all'aumento di capacità della memoria fisica corrisponda una diminuzione del numero di eccezioni di pagina: un certo numero di fattori concomitanti (quali la politica di rimpiazzo e la sequenza di chiamata) deve concorrere affinché la congettura sia verificata.

Questa incongruenza è nota come anomalia di Belady, dal nome dello studioso che per primo la rilevò.

Come mostrato di seguito, la sequenza indicata causa 9 eccezioni di pagina (marcate con '\*') su M1, meno capiente, e 10 su M2, più capiente:

(VEDI LUCIDO SUCCESSIVO)

## Soluzione

pagina richiesta	0	1	2	3	0	1	4	0	1	2	3	4
accesso più recente	0	1	2	3	0	1	4	4	4	2	3	3
accesso meno recente		0	1	2	3	0	1	1	1	4	2	2
eccezione di pagina			0	1	2	3	0	0	0	1	4	4
mancato riordinamento di lista	*	*	*	*	*	*	*	*	*	*	*	*

$M1$

pagina richiesta	0	1	2	3	0	1	4	0	1	2	3	4
accesso più recente	0	1	2	3	3	3	4	0	1	2	3	4
accesso meno recente		0	1	2	2	2	3	4	0	1	2	3
eccezione di pagina			0	1	1	1	2	3	4	0	1	2
mancato riordinamento di lista	*	*	*	*		*	*	*	*	*	*	*

$M2$

Si verifichi, per esercizio, cosa succede nel caso si utilizzi una implementazione rigorosa della LRU

## Esercizio: memoria virtuale

Sia dato un calcolatore con memoria virtuale con segmentazione paginata (dimensione pagina 8KB), con 7 segmenti (ognuno di 1023 pagine) e memoria principale di 512 MB con spazio di indirizzamento su 32 bit.

Si calcoli:

1. Quanti bit dovranno essere usati per la rappresentazione degli indirizzi virtuali ?
2. Quanti bit dovranno essere usati per la rappresentazione degli indirizzi fisici ?

## Soluzione

1. Quanti bit dovranno essere usati per la rappresentazione degli indirizzi virtuali ?

Ris.: occorrono 3 bit ( $7 < 2^3 = 8$ ) per individuare il segmento, 10 bit ( $1023 < 2^{10} = 1024$ ) per individuare la pagina e 13 bit ( $7 < 2^{13} = 8KB$ ) per individuare un indirizzo all'interno di una pagina.

$$\text{Totale} = 3 + 10 + 13 = 26 \text{ bit}$$

2. Quanti bit dovranno essere usati per la rappresentazione degli indirizzi fisici ?

Ris.: per indirizzare 512 MB di memoria fisica occorrono 29 bit, di cui 13 per individuare un indirizzo fisico all'interno della pagina

## Esercizio: memoria virtuale

**Si consideri un sistema monoprocessoresso multi-utente con memoria virtuale segmentata paginata (con aging di 8 bit, e altri bit di controllo, inclusi diritti di accesso, per un totale di altri 8 bit). Si assuma che:**

1. **Ciascun processo possa possedere al massimo 16 segmenti ed accederne, ad ogni istante, non più di 1**
2. **A ciascun segmento sia associato un descrittore di 8B**
3. **Il descrittore di segmento designi la base del segmento in memoria fisica, espressa su 32 bit, e l'ampiezza effettiva del segmento (limite), espressa in 24 bit, con i restanti 8 bit contenenti informazione di controllo**

## Esercizio: memoria virtuale

4. Un processo emetta indirizzi logici di 32 bit con la seguente struttura: bit 0-11 indirizzo in pagina logica, bit 12-23 indirizzo di pagina logica, bit 24-31 non utilizzati. Tali indirizzi designano un indirizzo fisico entro il segmento attivo mediante paginazione

**Si determini l'ampiezza minima dell'area di memoria occorrente per ospitare tutte le strutture di controllo associate alla memoria virtuale assegnata a ciascun processo, assumendo l'uso di unità di informazione di ampiezza minima di 1B.**

## Soluzione

1. Ciascun processo può possedere fino a 16 segmenti di cui al più 1 in uso. Quindi struttura dati con 16 posizioni per indicare il segmento in uso: bastano 2B.
2. Inoltre c'è un descrittore di 8B per ogni segmento: quindi  $16 \times 8B = 128B$ .
3. Un segmento contiene fino a 16MB. Quindi un segmento paginato con 12 bit di indirizzo logico determinano 4K pagine, ognuna di dimensione 4KB (bit da 0 a 11). Assumendo 2B per informazioni di controllo (inclusi diritti di accesso) e informazioni per la politica di rimpiazzo (vedi aging), e 4B per la base fisica, la tabella di rilocazione occuperà  $4K \times (2+4)B = 24576B$

**Quindi in totale ogni processo ha bisogno di**

$$2B + 128B + 24576B = 24706B$$

↑                   ↑                   ↑  
1.               2.               3.

## Esercizio: memoria virtuale

Si assuma una memoria segmentata con paginazione, dove ogni indirizzo logico può fare riferimento a non più di 8 segmenti. Ogni segmento può contenere non più di 2047 pagine, ognuna di 4KB. L'unità di indirizzamento logico è 1B e la memoria fisica è di 16MB.

Dati i seguenti due indirizzi logici, in esadecimale, 1001EC3 e 080281B e le tabelle dei segmenti e delle pagine mostrate nel lucido successivo, si dica a quali indirizzi fisici corrispondono, assumendo che la prima pagina fisica sia memorizzata a partire dall'indirizzo 0.

## Esercizio: memoria virtuale

# segmento	pagine	rif. tab. pag.
0	4	•
1	4	•
2	4	•
3	0	
4	0	
5	0	
6	0	
7	0	

# pagina logica	# pagina fisica
0	1
1	3
2	A
3	C

# pagina logica	# pagina fisica
0	4
1	6
2	F
3	7

# pagina logica	# pagina fisica
0	2
1	5
2	D
3	E

## Soluzione

Per l'indirizzamento dello spazio logico occorrono 3 bit per individuare il segmento, 11 bit per selezionare la pagina in un segmento, e 12 per individuare la locazione in una pagina di 4KB. Quindi in totale abbiamo 26 bit. Un indirizzo fisico ha bisogno di 24 bit (per indirizzare tutte le 16M locazioni da 1B), di cui i 12 meno significativi per individuare la locazione fisica all'interno della pagina fisica (offset).

Se rappresentiamo in binario (su 26 bit) l'indirizzo 1001EC3 otteniamo:

01 0000 0000 0001 1110 1010 0011

di cui i primi 3 bit rappresentano il segmento (quindi 2), i successivi 11 bit rappresentano la pagina nel segmento (quindi 1) e i rimanenti 12 bit rappresentano l'offset.

Dobbiamo generare un indirizzo fisico di 24 bit. I primi bit sono generati usando l'indirizzo fisico alla riga di pagina 1 della tabella puntata dal segmento 2 in tabella dei segmenti (quindi 6 esadecimale) e completandolo con l'offset (quindi EC3 esadecimale), ottenendo in binario (su 24 bit)

0000 0000 0110 1110 1010 0011

Di seguito riportiamo la traduzione in esadecimale per i due indirizzi logici

indirizzo logico		indirizzo fisico
1001EC3	segmento 2, pagina 1	006EC3
080281B	segmento 1, pagina 2	00D81B

## Esercizio: memoria virtuale

Si assuma una memoria segmentata con paginazione, dove ogni indirizzo logico può fare riferimento a non più di 8 segmenti. Ogni segmento può contenere non più di 511 pagine, ognuna di 16KB. L'unità di indirizzamento logico è 1B e la memoria fisica è di 16MB.

Dati i seguenti due indirizzi logici, in esadecimale, 100AFF1 e 0009004 e le tabelle dei segmenti e delle pagine mostrate nel lucido successivo, si dica a quali indirizzi fisici corrispondono, assumendo che la prima pagina fisica sia memorizzata a partire dall'indirizzo 0.

## Esercizio: memoria virtuale

The diagram illustrates the mapping of logical pages to physical pages. It consists of three tables:

- Table 1: Segment Table** (Left):
 

# segmento	pagine	rif. tab. pag.
0	5	●
1	4	●
2	6	●
3	0	
4	0	
5	0	
6	0	
7	0	
- Table 2: Page Table 1** (Middle):
 

# pagina logica	# pagina fisica
0	0
1	C
2	7
3	8
4	B
- Table 3: Page Table 2** (Right):
 

# pagina logica	# pagina fisica
0	1
1	2
2	D
3	E

Arrows from the 'rif. tab. pag.' column in Table 1 point to the corresponding rows in Tables 2 and 3. Specifically, row 0 of Table 1 points to row 0 of Table 2, and rows 1, 2, and 3 of Table 1 point to row 1 of Table 3.

Lezione

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 172

## Soluzione

Per l'indirizzamento dello spazio logico occorrono 3 bit per individuare il segmento, 9 bit per selezionare la pagina in un segmento, e 14 per individuare la locazione in una pagina di 16KB. Quindi in totale abbiamo 26 bit. Un indirizzo fisico ha bisogno di 24 bit (per indirizzare tutte le 16M locazioni da 1B), di cui i 14 meno significativi per individuare la locazione fisica all'interno della pagina fisica (offset).

Se rappresentiamo in binario (su 26 bit) l'indirizzo 100AFF1 otteniamo:

01 0000 0000 1010 1111 1111 0001

di cui i primi 3 bit rappresentano il segmento (quindi 2), i successivi 9 bit rappresentano la pagina nel segmento (quindi 2) e i rimanenti 14 bit rappresentano l'offset.

Dobbiamo generare un indirizzo fisico di 24 bit. I primi 10 bit sono generati usando l'indirizzo fisico alla riga di pagina 2 della tabella puntata dal segmento 2 in tabella dei segmenti (quindi 6 esadecimale rappresentato su 24 bit indirizzo fisico) – 14 (offset) = 10 bit:

0000 0001 10

I restanti 14 bit corrisponderanno all'offset dell'indirizzo logico, ottenendo in binario (su 24 bit)

0000 0001 1010 1111 1111 0001

Di seguito riportiamo la traduzione in esadecimale per i due indirizzi logici

indirizzo logico	indirizzo fisico
100AFF1	01AFF1
0009004	01D004

Lezione

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 173

## Esercizio: memoria virtuale

Assumendo una memoria fisica di capienza 4 pagine, inizialmente vuota, una memoria virtuale di capienza 8 pagine (numerate da 0 a 7) e la seguente sequenza di riferimenti:

**0(0),1(3),7(5),2(6),3(8),2(9),7(9),1(11),0(12),3(15)**

dove la notazione  $X(Y)$  indica una richiesta dalla pagina  $X$  effettuata al tempo  $Y$ , mostrare l'evoluzione della lista mantenuta dall'algoritmo *second chance* (seconda possibilità) e confrontarne comportamento e costo realizzativo con quello dell'algoritmo LRU (least recently used).

## Soluzione

In tabella (vedi lucido successivo) possiamo seguire il variare delle informazioni di controllo associate alle pagine presenti in memoria principale al succedersi delle richieste di pagina, dove  $T$  denota il tempo di caricamento della pagina,  $P$  l'indice di pagina ed  $R$  il bit di riferimento. Dalla tabella è facile rilevare come l'algoritmo dato degeneri in puro FIFO laddove tutte le pagine presenti siano state recentemente riferite. Gli istanti in cui ciò avviene sono marcati con un \*.

In rapporto all'algoritmo LRU, l'algoritmo dato presenta un minor costo implementativo (in quanto quest'ultimo non riordina la lista delle pagine ad ogni riferimento, come invece previsto dall'LRU) ed una minore efficacia (in quanto esso può rimpiazzare la pagina più recentemente riferita, come marcato in tabella con un \*, la quale ha, secondo il principio di località, elevata probabilità di essere riferita nell'immediato futuro, ciò che non accade con l'LRU).

# Soluzione

sequenza richieste					
0(0)	1(3)	7(5)	2(6)*	3(8)	2(9)
T 0	0 ← 3	0 ← 3 ← 5	0 ← 3 ← 5 ← 6	8 ← 8 ← 8 ← 8	8 ← 8 ← 8 ← 8
P 0	0 ← 1	0 ← 1 ← 7	0 ← 1 ← 7 ← 2	1 ← 7 ← 2 ← 3	1 ← 7 ← 2 ← 3
R 1	1 ← 1	1 ← 1 ← 1	1 ← 1 ← 1 ← 1	0 ← 0 ← 0 ← 1	0 ← 0 ← 1 ← 1
7(9)	1(11)*•	0(12)	3(15)		
T 8 ← 8 ← 8 ← 8	8 ← 8 ← 8 ← 8	12 ← 12 ← 12 ← 12	12 ← 12 ← 12 ← 12		
P 1 ← 7 ← 2 ← 3	1 ← 7 ← 2 ← 3	7 ← 2 ← 3 ← 0	7 ← 2 ← 3 ← 0		
R 0 ← 1 ← 1 ← 1	1 ← 1 ← 1 ← 1	0 ← 0 ← 0 ← 1	0 ← 0 ← 1 ← 1		