9' Esercitazione

https://politecnicomilano.webex.com/meet/gianenrico.conti

16 maggio 2022

Gian Enrico Conti

Memoria virtuale

Memoria virtuale: recap

- Riduce costi
- Spazio Logico (virtuale) >= spazio fisico
- Pagine stessa grandezza
- (Stesso offset)
- NPV>=NPF
- Page fault
- Bit x validita'
- Mapping fatto in HW (con supporto nel .S.O. ...)
- Tabella della pagine

Memoria virtuale: esempio 1

- Es1

32 bit spazio logico (virtuale)

30 bit spazio fisico

Calcoliamo NPF NPV bit necessari.

Memoria virtuale: esempio 1

32 bit spazio logico (virtuale)

4 GB

30 bit spazio fisico

1 GB

Supponiamo page Size 4 K

…

Memoria virtuale: esempio 1

32 bit spazio logico (virtuale)	4 GB
30 bit spazio fisico	1 GB
Page Size	4 K

NPL = 4GB / 4 KB = 2^32 / 2 ^12 = 2^20 -> 20 BIT x individuarle

..

Memoria virtuale: esempio 1

32 bit spazio logico (virtuale)	4 GB
30 bit spazio fisico	1 GB
Page Size	4 K

NPL = 4GB / 4 KB = 2^32 / 2 ^12 = 2^20 -> 20 BIT x individuarle

NPF = 2^30 / 2^12 = 2^18 -> 18 bit

La Page Table 20 bit

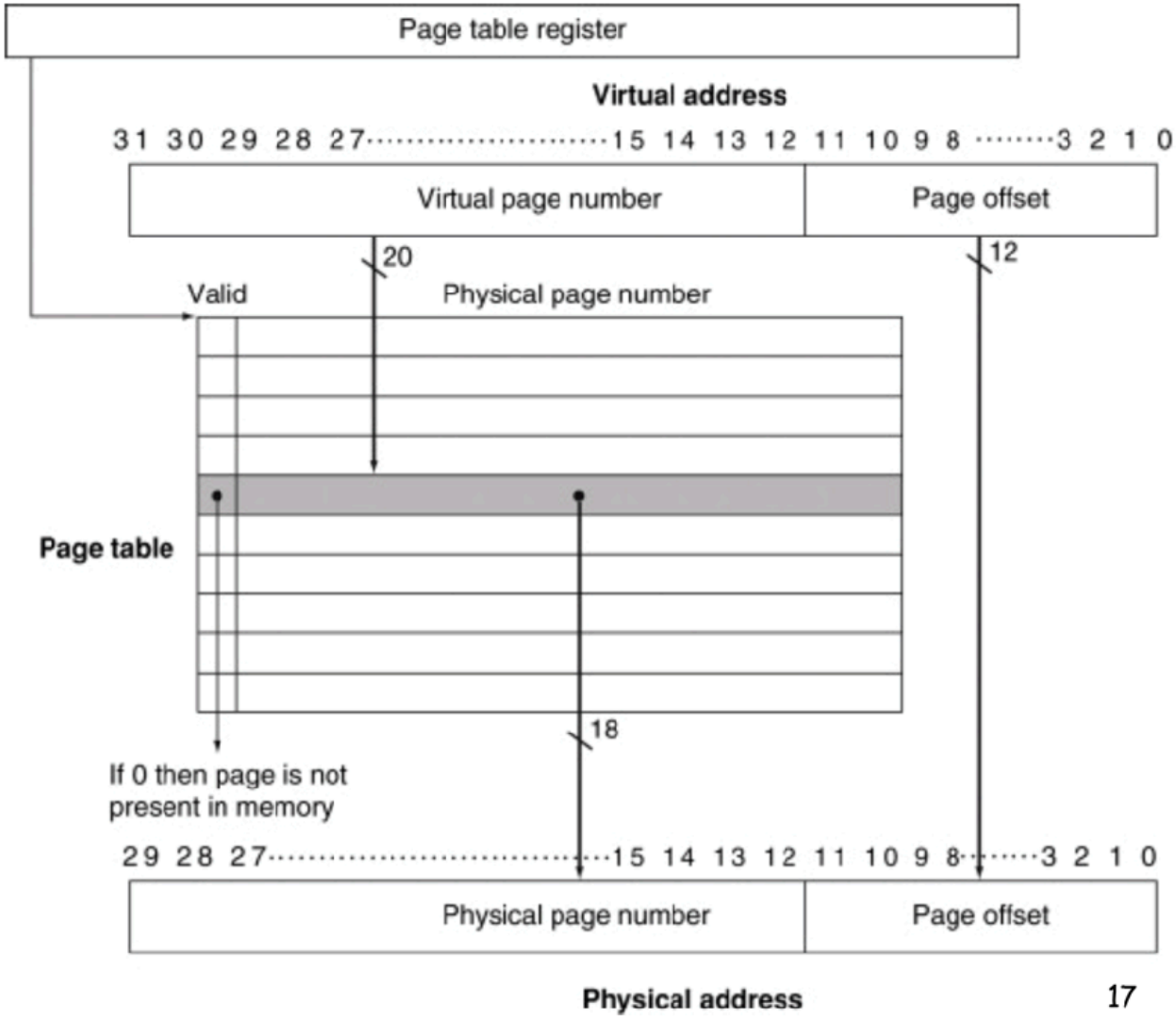
Memoria virtuale: esempio 1

NPL = 4GB / 4 KB = 2^32 / 2 ^12 = 2^20 -> 20 BIT x individuarle

NPF = 2^30 / 2^12 = 2^18 -> 18 bit

La Page Table 20 righe

VPN fa da indice



Bit di validita': 0 == page fault

Memoria virtuale: esempio 2

48 bit spazio logico (virtuale)

40 bit spazio fisico

page Size 4 K

Calcolare NPF NPV e offset.

Memoria virtuale: esempio 2

48 bit spazio logico (virtuale) (256 TB) 40 bit spazio fisico (1TB)

page Size 4 K (12 bit)

48 -12 36 bit x NPV

40 - 12 = 28 bit x NPF

Offset 12 bit (4K pages)

Notazioni x esercizi

La pagina virtuale n dell'area virtuale A del programma o processo P è indicata con la notazione APn, dove:

- A indica un tipo di area virtuale secondo la convenzione:

C (codice)

D (dati),

P (pila)

COND (area dati condivisa)

- P indica il programma o il processo
- n indica il numero di pagina nell'ambito dell'area virtuale

quindi

<tipo><processo><n.pagina>

Notazioni x esercizi: <tipo><processo><n.pagina>

- A indica un tipo di area virtuale secondo la convenzione:

C (codice)

D (dati),

P (pila)

COND (area dati condivisa)

- P indica il programma o il processo
- n indica il numero di pagina nell'ambito dell'area virtuale

Esempio:

DQ3 indica la pagina 3 dell'area dati del processo Q

Nota:

non è la pagina con NPV = 3,

perchè in generale l'area DP non inizia con NPV = 0

VM: Esercizio 1

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti:

la memoria centrale fisica ha capacità di 32 Kbyte,

quella logica di 32 Kbyte e la pagina ha dimensione 4 Kbyte.

Si chiede di svolgere i punti seguenti...

A b c d e, li vedremo passo passo..

(esercizio-memvirt-1.pdf)

VM: Esercizio 1 richiesta a

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti:

la memoria centrale fisica ha capacità di 32 Kbyte,

quella logica di 32 Kbyte e la pagina ha dimensione 4 Kbyte.

A) Si definisca la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi

NPF, Spiazzamento fisico, NPL, Spiazzamento logico

VM: Esercizio 1 richiesta a: Soluzione

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti:

la memoria centrale fisica ha capacità di 32 Kbyte,

quella logica di 32 Kbyte e la pagina ha dimensione 4 Kbyte.

A) Si definisca la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi:

NPF, Spiazzamento fisico, NPL, Spiazzamento logico

NPF:

32 KByte / 4 KByte = 8, di solito si scrivono i Bit necessari quindi

NPF 2^3 -> 3 bit

Nota: talora la soluzione dice semplicemente NPF = 3...

VM: Esercizio 1 richiesta a: Soluzione

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti:

la memoria centrale fisica ha capacità di 32 Kbyte,

quella logica di 32 Kbyte e la pagina ha dimensione 4 Kbyte.

A) Si definisca la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi:

NPF, Spiazzamento fisico, NPL, Spiazzamento logico

NPF: 3,

Spiazzamento fisico: 4K = 2^12 quindi bastano 12 bit.

VM: Esercizio 1 richiesta a: Soluzione

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti:

la memoria centrale fisica ha capacità di 32 Kbyte,

quella logica di 32 Kbyte e la pagina ha dimensione 4 Kbyte.

A) Si definisca la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi:

NPF, Spiazzamento fisico, NPL, Spiazzamento logico

NPF: 3,

Spiazzamento fisico: 12,

NPL: identico al NPF, 32K /4 K bastano 3 bit.

VM: Esercizio 1 richiesta a: Soluzione

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti:

la memoria centrale fisica ha capacità di 32 Kbyte,

quella logica di 32 Kbyte e la pagina ha dimensione 4 Kbyte.

A) Si definisca la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi:

NPF, Spiazzamento fisico, NPL, Spiazzamento logico

NPF: 3,

Spiazzamento fisico: 12,

NPL: 3,

Spiazzamento logico: non cambi nulla rispetto a quello fisico:

12

VM: Esercizio 1 richiesta b

Nel sistema vengono creati alcuni processi, indicati nel seguito con P, Q, R, S.

I programmi eseguiti da tali processi sono due: X e Y.

La dimensione iniziale dei segmenti dei programmi è la seguente:

CX: 8K DX: 4K PX:4K

CY:12K DY: 8K PY:4K

Si inserisca in tabella 1

L a struttura in pagine della memoria virtuale

indir. virtuale	prog. X	prog. Y
0		
1		
2		
3		
4		
5		
6		
7		

(mediante la notazione definita sopra: CX0 CX1 DX0 PX0 … CY0 …).

VM: Esercizio 1 richiesta b: Soluzione

CX: 8K DX: 4K PX:4K

CY:12K DY: 8K PY:4K

- Tutti i processi P, Q, R, S eseguono X e Y
- Quindi verrano caricati:
  Prima codice
  poi dati ed
  IN FONDO Lo stack..

Tutto nella VM e ovviamente UN SOLA copia di X e di Y....

VM: Esercizio 1 richiesta b: Soluzione

CX: 8K DX: 4K PX:4K

CY:12K DY: 8K PY:4K

CX 2 pagine...

indir. virtuale	prog. X	prog. Y
0	CX0	
1	CX1	
2		
3		
4		
5		
6		
7		

1) *memoria logica*

VM: Esercizio 1 richiesta b: Soluzione

CX: 8K DX: 4K PX:4K

CY:12K DY: 8K PY:4K

Idem

CY 3 pagine...

indir. virtuale	prog. X	prog. Y
0	CX0	CY0
1	CX1	CY1
2		CY2
3		
4		
5		
6		
7		

1) *memoria logica*

VM: Esercizio 1 richiesta b: Soluzione

CX: 8K DX: 4K PX:4K

CY:12K DY: 8K PY:4K

Data:

X 1 pagina,

Y 2 pagine

Subito adiacenti..

indir. virtuale	prog. X	prog. Y
0	CX0	CY0
1	CX1	CY1
2	DX0	CY2
3		DY0
4		
5		
6		
7		

1) *memoria logica*

VM: Esercizio 1 richiesta b: Soluzione

CX: 8K DX: 4K PX:4K

CY:12K DY: 8K PY:4K

STACK x definizione "in fondo"

indir. virtuale	prog. X	prog. Y
0	CX0	CY0
1	CX1	CY1
2	DX0	CY2
3		DY0
4		DY1
5		
6		
7	PX0	PY0

1) *memoria logica*

VM: Esercizio 1 richiesta c

A un certo istante di tempo $\mathbf{t_0}$ sono terminati, nell'ordine, gli eventi seguenti:

1. creazione del processo P e lancio del programma Y ("fork" di P ed "exec" di Y)
2. creazione del processo Q e lancio del programma X ("fork" di Q ed "exec" di X)
3. accesso a 1 pagina dati e creazione di una nuova pagina di pila da parte di P
4. accesso a 1 pagina dati da parte di Q
5. creazione del processo R come figlio di P ("fork" eseguita da P)
6. creazione di 1 pagina di pila da parte di R

Con le seguenti ipotesi...

VM: Esercizio 1 richiesta c: ipotesi

- il lancio di una programma avviene caricando solamente la pagina di codice con l'istruzione di partenza e una sola pagina di pila
- il caricamento di pagine ulteriori è in Demand Paging (cioè le pagine si caricano su richiesta senza scaricare le precedenti fino al raggiungimento del numero massimo di pagine residenti)
- l'indirizzo (esadecimale) dell'istruzione di partenza di X è 14AF
- l'indirizzo (esadecimale) dell'istruzione di partenza di Y è 0231
- il numero di pagine residenti R vale 3 (tre)
- viene utilizzato l'algoritmo LRU (ove richiesto prima si dealloca una pagina di processo e poi si procede alla nuova assegnazione)
- le pagine meno utilizzate in ogni processo sono quelle caricate da più tempo, con la sola eccezione seguente: se è residente una sola pagina di codice, quella è certamente stata utilizzata recentemente
- al momento di una fork viene duplicata solamente la pagina di pila caricata più recentemente
- dopo la fork le pagine di codice possono essere condivise tra i processi padre e figlio, se ambedue i processi usano la stessa pagina virtuale...

VM: Esercizio 1 richiesta c: ipotesi II

- ipotizzando che l'allocazione delle pagine virtuali nelle pagine fisiche avvenga sempre in sequenza, senza buchi, a partire dalla pagina fisica 0,

si indichi, completando tabella 2, l'allocazione fisica delle pagine dei tre processi all'istante t0 (notazione CP0 CP1 DP0 PP0 … CQ0 …).

indir. fisico	**pagine allocateal tempo t_0**
0	
1	
2	
3	
4	
5	
6	
7	

2) memoria fisica, istante t_0

VM: Esercizio 1 richiesta c: soluzione

1 step:

il processo P parte caricando la pagina CP0 (perché il programma Y ha istruzione di partenza in pagina 0) e una pagina di pila PP0:

(No data, vedi ipotesi)

....

indir. fisico	pagine allocateal tempo t_0
0	
1	
2	
3	
4	
5	
6	
7	

2) memoria fisica, istante t_0

VM: Esercizio 1 richiesta c: soluzione

1 step:

il processo P parte caricando la pagina CP0 (perché il programma Y ha istruzione di partenza in pagina **0**, addr. 0231)

e una pagina di pila PP0:

(No data, vedi ipotesi)

indir. fisico	pagine allocate al tempo t_0
0	CP0
1	PP0
2	
3	
4	
5	
6	
7	

2) memoria fisica, istante t_0

VM: Esercizio 1 richiesta c: soluzione

2 step:

successivamente Q carica CQ1 (perché il programma X ha istruzione di partenza in pagina 1) e PQ0

indir. fisico	pagine allocate al tempo t_0
0	CP0
1	PP0
2	**CQ1**
3	**PQ0**
4	
5	
6	
7	

2) memoria fisica, istante t_0

VM: Esercizio 1 richiesta c: soluzione

3 step:

P carica DP0, raggiungendo il limite delle pagine residenti (3), e quindi per caricare la pagina PP1 deve eliminare PP0 (vedi regole di utilizzazione indicate nel tema)

indir. fisico	pagine allocate al tempo t_0
0	CP0
1	~~PP0~~ **PP1**
2	CQ1
3	PQ0
4	DP0
5	
6	
7	

2) memoria fisica, istante t_0

VM: Esercizio 1 richiesta c: soluzione

4 step:

P carica DP0, raggiungendo il limite delle pagine residenti (3), e quindi per caricare la pagina PP1 deve eliminare PP0 (vedi regole di utilizzazione indicate nel tema)
Q carica DQ0 e raggiunge 3 pagine caricate

indir. fisico	pagine allocate al tempo t_0
0	CP0
1	~~PP0~~ PP1
2	CQ1
3	PQ0
4	DP0
5	**DQ0**
6	
7	

2) memoria fisica, istante t_0

VM: Esercizio 1 richiesta c: soluzione

5 step:

il nuovo processo R non ha bisogno di caricare CR0, perché esegue lo stesso programma di P e quindi CR0 è uguale a CP0,

quindi carica solamente PR1 (che è la copia della pagina PP1, ma conterrà un diverso pid)

indir. fisico	pagine allocate al tempo t_0
0	**CP0 (= CR0)**
1	~~PP0~~ PP1
2	CQ1
3	PQ0
4	DP0
5	DQ0
6	**PR1**
7	

2) memoria fisica, istante t_0

VM: Esercizio 1 richiesta c: soluzione

6 step:

R2 carica PR2

indir. fisico	pagine allocate al tempo t_0
0	**CP0 (= CR0)**
1	~~PP0~~ PP1
2	CQ1
3	PQ0
4	DP0
5	DQ0
6	PR1
7	**PR2**

2) memoria fisica, istante t_0

VM: Esercizio 1 richiesta d

- A un certo istante di tempo t1 > t0 sono terminati gli eventi seguenti:

7) terminazione del processo P (exit)

8) esecuzione della funzione “exec Y” (lancio di Y) nel processo Q e conseguente trasformazione di Q in processo S (si noti che Q si trasforma in S ma il pid resta lo stesso perché non c’è “fork”)

9) accesso a 2 pagine di dati per il processo S

.....

VM: Esercizio 1 richiesta d

Si completi la tabella 3 nelle medesime ipotesi delineate nel precedente punto (c) e supponendo che, dovendo utilizzare una pagina fisica libera, venga **sempre** utilizzata lapagina fisica libera avente indirizzo minore. Si aggiorni anche la tabella 1A (non è indispensabile).

indir. fisico	pagine allocateal tempo t_1
0	
1	
2	
3	
4	
5	
6	
7	

3)memoria fisica, istante t_1

VM: Esercizio 1 richiesta d

7 step:

P termina e libera le pagine fisiche 1 e 4

(non la 0, perché CR0 rimane necessaria)

indir. fisico	pagine allocate al tempo t_1
0	CP0 (= CR0)
1	
2	CQ1
3	PQ0
4	
5	DQ0
6	PR1
7	PR2

3)memoria fisica, istante t_1

VM: Esercizio 1 richiesta d

8 step:

la exec Y da parte di Q non alloca il codice, perché CS0 risulta identica a CR0, ma alloca la pagina PS0 nella pagina fisica 1, già libera; vengono inoltre liberate le pagine fisiche 2 e 3

indir. fisico	pagine allocate al tempo t_1
0	CP0 (= CR0)
1	**PS0**
2	
3	
4	
5	DQ0
6	PR1
7	PR2

3)memoria fisica, istante t_1

VM: Esercizio 1 richiesta d

9 step:

la nuova pagina DS0 viene allocata in pagina fisica 2, ma S raggiunge così il livello massimo di pagine residenti e quindi la successiva (DS1) deve essere allocata al posto di PS0

indir. fisico	pagine allocate al tempo t_1
0	CP0 (= CR0)
1	~~PS0~~ **DS1**
2	**DS0**
3	
4	
5	DQ0
6	PR1
7	PR2

3)memoria fisica, istante t_1

VM: Esercizio 1 richiesta e

Si indichi il contenuto della tabella delle pagine della MMU all'istante di tempo t_1 completando la tabella 4. Si ipotizzi che le righe della tabella siano state allocate ordinatamente man mano che venivano allocate le pagine di memoria virtuale e che gli eventi di cui ai punti (c, d), influenzanti la MMU, partano da una situazione di tabella vergine, abbiano utilizzato le righe lasciate libere e che se è richiesta una nuova riga si utilizzi **sempre** la prima riga libera. **Si indichi** anche il valore assunto dal bit di validità di pagina (il valore 1 significa che la pagina è caricata). Il numero di righe nella tabella sotto non è significativo.

PID *indicare P Q R o S comepid oppure ns se la riga non è significativa*	**NPV** *utilizzare la notazione CP0/0 per indicare "segmento di codice di P numero 0 / pagina virtuale numero 0", e similmente per le altre*	**NPF**	**Bit di Validità**

VM: Esercizio 1 richiesta e: Soluzione

Per permettere di interpretare il risultato riportato nella seguente tabella si indicano, oltre al contenuto della tabella all'istante finale, anche i contenuti precedenti.

PID *indicare P Q R o S comepid oppure ns se la riga non è significativa*	**NPV** *utilizzare la notazione CP0/0 per indicare "segmento di codice di P numero 0 / pagina virtuale numero 0", e similmente per le altre*	**NPF**	**Bit di Validità**
~~P ns~~ S	~~CP0~~ CS0 / 0	0	1
~~P ns~~ S	~~PP0 PP1 PS0~~ DS1 / 4	1	1
~~Q ns~~ S	~~CQ1~~ DS0 / 3	2	1
~~Q~~ ns	~~PQ0~~	3	0
~~P~~ ns	~~DP0~~	4	0
~~Q~~ ns	~~DQ0~~	5	0
R	CR0 / 0	0	1
R	PR1 / 6	6	1
R	PR2 / 5	7	1

VM: Esercizio 2

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità di 32 K byte, quella logica di 32 K byte e la pagina ha taglia di 4 K byte. Si chiede di svolgere i punti seguenti:

(a) **Si definisca** la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi

VM: Esercizio 2

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità di 32 K byte, quella logica di 32 K byte e la pagina ha taglia di 4 K byte. Si chiede di svolgere i punti seguenti:

(a) **Si definisca** la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi

NPF: 3 bit
spiazzamento fisico: 12 bit
NPL: 3 bit
spiazzamento logico: 12 bit

VM: Esercizio 2

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità di 32 K byte, quella logica di 32 K byte e la pagina ha taglia di 4 K byte. Si chiede di svolgere i punti seguenti:

(b) **Si inserisca** nella tabella a fianco la struttura in pagine della memoria vir- tuale di due programmi X e Y (mediante la notazione CX0 CX1 DX0 PX0 ... CY0 ...), sapendo che la dimensione iniziale dei segmenti di tali programmi è la seguente:

CX: 4 K CY:16K

DX:12 K DY: 4K

PX:4 K PY:4K

indir. virtuale	prog. **X**	prog. **Y**
0		
1		
2		
3		
4		
5		
6		
7		

VM: Esercizio 2

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità di 32 K byte, quella logica di 32 K byte e la pagina ha taglia di 4 K byte. Si chiede di svolgere i punti seguenti:

(b) **Si inserisca** nella tabella a fianco la struttura in pagine della memoria vir- tuale di due programmi X e Y (mediante la notazione CX0 CX1 DX0 PX0 ... CY0 ...), sapendo che la dimensione iniziale dei segmenti di tali programmi è la seguente:

CX: 4 K CY:16K

DX:12 K DY: 4K

PX:4 K PY:4K

indir. virtuale	**prog. X**	**prog. Y**
0	CX0	CY0
1	DX0	CY1
2	DX1	CY2
3	DX2	CY3
4		DY0
5		
6		
7	PX0	PY0

VM: Esercizio 2

C) Nel sistema vengono creati alcuni processi, indicati nel seguito con P, Q, R e S.

A pagina seguente sono indicate due serie di eventi; la prima serie termina all'istante t_0, la seconda all'istante t_1.

Si compilino le tabelle che descrivono i contenuti della memoria fisica e della MMU agli istanti t_0 e t_1, utilizzando la notazione CP0 CP1 DP0 PP0 … CQ0 … per indicare le pagine virtuali dei processi; nelle ta- belle della MMU si aggiunga anche il NPV effettivo con la notazione CP0/0 ecc …; in tutte le tabelle si usi la notazione (CP0) ecc … per indicare un dato non più valido e la notazione CP0 = CQ0 ecc … per indica-re che una pagina fisica contiene più di una pagina logica.

Si considerino valide le seguenti ipotesi relative al sistema:

- il lancio di una programma avviene caricando solamente la pagina di codice con l'istruzione di par-tenza e una sola pagina di pila
- il caricamento di pagine ulteriori è in Demand Paging (cioè le pagine si caricano su richiesta senza scaricare le precedenti fino al raggiungimento del numero massimo di pagine residenti)
- l'indirizzo (esadecimale) dell'istruzione di partenza di X è 0AAA
- l'indirizzo (esadecimale) dell'istruzione di partenza di Y è 39F2
- il numero di pagine residenti **R** vale **4**
- viene utilizzato l'algoritmo LRU (ove richiesto prima si dealloca una pagina di processo e poi si pro- cede alla nuova assegnazione)
- in assenza di indicazioni esplicite relative all'accesso alle pagine, le pagine meno utilizzate in ogni processo sono quelle caricate da più tempo; inoltre la pagina di codice caricata più di recente è acceduta continuamente
- al momento di una "fork" viene duplicata solamente la pagina di pila caricata più recentemente, ma tutte le pagine virtuali del padre sono considerate condivise con il figlio
- dopo una "fork", se uno dei due processi padre o figlio scrive in una pagina condivisa, la nuova pagina fisica che viene allocata appartiene al processo che ha eseguito la scrittura
- l'allocazione delle pagine virtuali nelle pagine fisiche avviene **sempre** in sequenza, senza buchi, a partire dalla pagina fisica 0
- le righe della tabella della MMU vengono allocate ordinatamente man mano che vengono allocate le pagine di memoria virtuale
- gli eventi influenzanti la MMU partono da una situazione di tabella vergine
- se è richiesta una nuova riga di MMU, si utilizza **sempre** la prima riga libera

VM: Esercizio 2

(d) A un certo istante di tempo t_0 sono terminati, nell'ordine, gli eventi seguenti:

1. creazione del processo P e lancio del programma X ("fork" di P ed "exec" di X)
2. P accede a pagine nel seguente ordine: dati 1, dati 0, pila
3. P crea due pagine dati dinamiche tramite BRK
4. creazione del processo Q e lancio del programma Y ("fork" di Q ed "exec" di Y)
5. Q salta all'istruzione di indirizzo 2B35, poi accede alla pila
6. Q crea 2 pagine di pila

Si compilino le tabelle, al tempo t_0.

memoria fisica	
indir. fisico	**pagine allocate**
0	
1	
2	
3	
4	
5	
6	
7	

MMU			
proc.	**NP V**	**NP F**	**valid bit**
P			
P			
P			
P			
Q			
Q			
Q			
Q			

VM: Esercizio 2

(d) A un certo istante di tempo t_0 sono terminati, nell'ordine, gli eventi seguenti:

1. creazione del processo P e lancio del programma X ("fork" di P ed "exec" di X)
2. P accede a pagine nel seguente ordine: dati 1, dati 0, pila
3. P crea due pagine dati dinamiche tramite BRK
4. creazione del processo Q e lancio del programma Y ("fork" di Q ed "exec" di Y)
5. Q salta all'istruzione di indirizzo 2B35, poi accede alla pila
6. Q crea 2 pagine di pila

Si compilino le tabelle, al tempo t_0.

memoria fisica	
indir.fisico	**pagine allocate**
0	CP0
1	PP0
2	(DP1) DP4
3	(DP0) DP5
4	(CQ3) PQ2
5	PQ0
6	CQ2
7	PQ1

MMU			
proc.	**NP V**	**NP F**	**valid bit**
P	CP0/0	0	1
P	PP0/7	1	1
P	(DP1/2) DP4/5	2	1
P	(DP0/0 DP5/6	3	1
Q	(CQ3/3) PQ2/5	4	1
Q	PQ0/7	5	1
Q	CQ2/2	6	1
Q	PQ1/6	7	1
			0
			0
			0
			0

VM: Esercizio 2

A un certo istante di tempo $t_1 > t_0$ sono terminati, nell'ordine, gli eventi seguenti:

P termina ("exit" di P)

Q esegue una fork e crea il processo R

R esegue una fork e crea il processo S

Si compilino le tabelle sotto, al tempo t_1.

memoria fisica	
indir. fisico	**pagine allocate**
0	
1	
2	
3	
4	
5	
6	
7	

MMU			
proc.	**NP V**	**NP F**	**validbit**
(P) R			
(P) R			
(P) R			
(P) R			
Q			
Q			
Q			
Q			
S			
S			
S			
S			

VM: Esercizio 2

A un certo istante di tempo $t_1 > t_0$ sono terminati, nell'ordine, gli eventi seguenti:

P termina ("exit" di P)

Q esegue una fork e crea il processo R

R esegue una fork e crea il processo S

Si compilino le tabelle sotto, al tempo t_1.

memoria fisica	
indir. fisico	**pagine allocate**
0	(CP 0) PR2
1	(PP 0) PS2
2	(DP 1) (DP4)
3	(DP 0) (DP5)
4	(CQ 3) PQ2
5	PQ0 = PR0 = PS0
6	CQ2 = CR2 = CS2
7	PQ1 = PR1 = PS1

MMU			
proc.	**NPV**	**NPF**	**validbit**
(P) R	(CP0/0) PR2/5	(0) 0	(1)(0)1
(P) R	(PP0/7) PR0/7	(1) 5	(1)(0)1
(P) R	(DP1/2) (DP4/5) CR2/2	(2) 6	(1)(0)1
(P) R	(DP0/0) (DP5/6) PR1/6	(3) 7	(1)(0)1
Q	(CQ3/3) PQ2/5	4	1
Q	PQ0/7	5	1
Q	CQ2/2	6	1
Q	PQ1/6	7	1
S	PS2/5	1	(0) 1
S	PS0/7	5	(0) 1
S	CS2/2	6	(0) 1
S	PS1/6	7	(0) 1

VM: Esercizio 3

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica è di 64 Kbyte, l'indirizzo logico è di 16 bit, la dimensione di pagina è di 4 Kbyte. Si svolgano i punti seguenti:

Si definisca la struttura dell'indirizzo fisico dando la lunghezza in bit dei campi costituenti:

NPF: 4 bit Spiazzamento fisico: 12 bit

SOLUZIONE:

NPF: 4 bit Spiazzamento fisico: 12 bit

VM: Esercizio 3

b) I tre programmi PGP, PGQ e PGR, che verranno eseguiti dai tre processi P, Q e R, rispettivamente,hanno la struttura di segmentazione iniziale seguente (e condividono il segmento COND):

CP: 8 K	DP: 4 K	PP: 8 K	COND: 4 K
CQ: 12 K	DQ: 4 K	PQ: 4 K	COND: 4 K
CR: 8 K	DR: 4 K	PR: 4 K	COND: 4 K

La dimensione complessiva dello spazio di indirizzi di ogni processo è di 64 K. Il segmento pila di ogni processo inizia dal fondo di tale spazio e cresce verso l'indirizzo iniziale. Onde permettere la crescita dello Heap (dati dinamici, funzione `malloc`), nei processi P, Q e R il segmento condiviso COND è allocato lasciando 2, 1 e 2 pagine libere dopo i segmenti dati DP, DQ e DR, rispettivamente.

.....

VM: Esercizio 3

Si definisca in tabella 1A il significato delle varie pagine di memoria logica, tramite la notazione usuale: CP0, CP1, DP0, PP0, …, CQ0, …, COND0, … (ecc).

CP: 8 K	DP: 4 K	PP: 8 K	COND: 4 K
CQ: 12 K	DQ: 4 K	PQ: 4 K	COND: 4 K
CR: 8 K	DR: 4 K	PR: 4 K	COND: 4 K

Indirizzo di pagina virtuale	Processo **P**	Processo **Q**	Processo **R**
0	CP0	CQ0	CR0
1	CP1	CQ1	CR1
2	DP0	CQ2	DR0
3		DQ0	
4		DQ1	
5	COND0	COND0	COND0
6			
7			
8			
9			
A			
B	PP4		
C	PP3		
D	PP2		
E	PP1		
F	PP0	PQ0	PR0