POLITECNICO DI MILANO

Memoria Virtuale

Alessandro A. Nacci
alessandro.nacci@polimi.it

ACSO

2014/2014


START

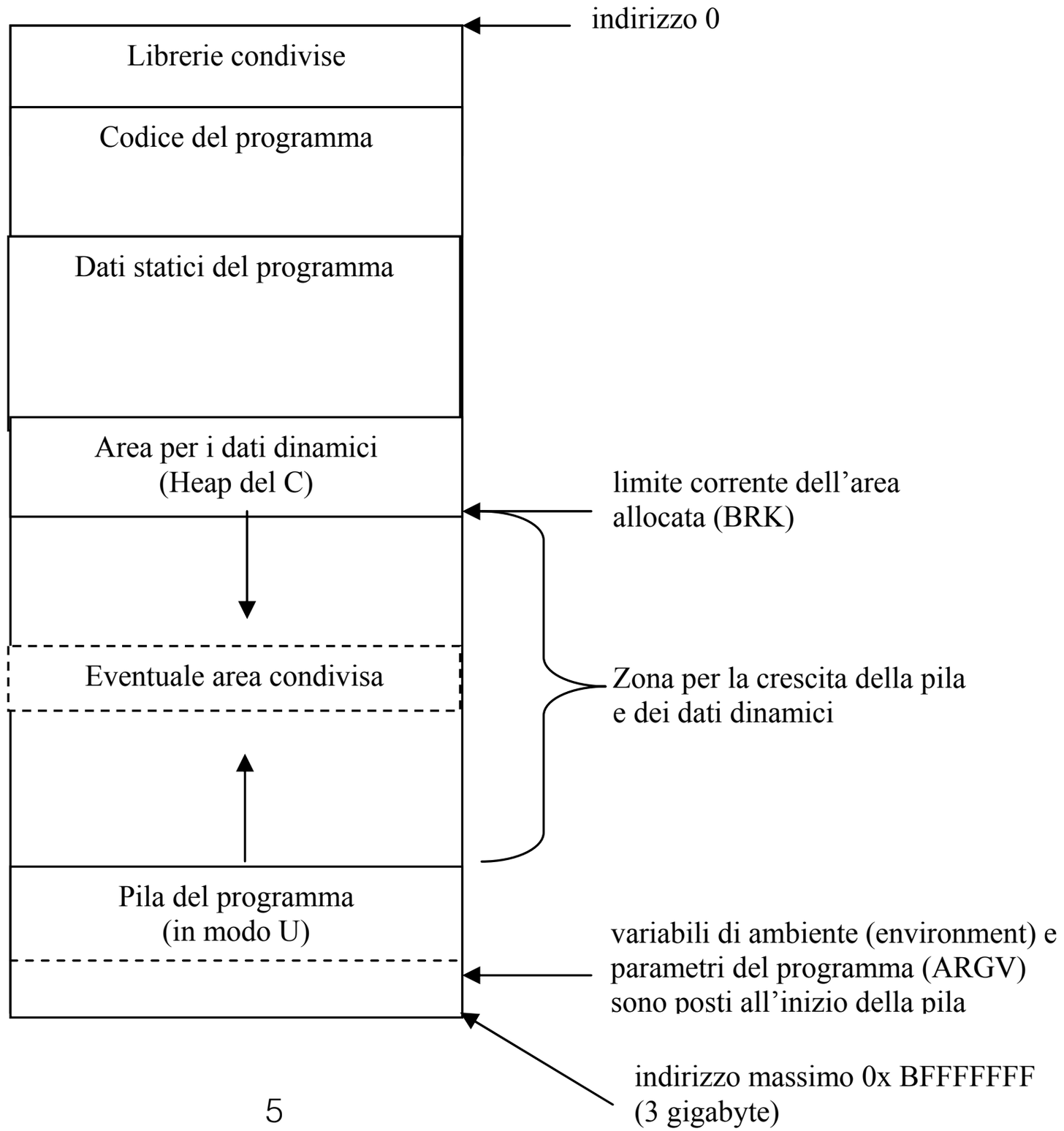
Algoritmo LRU

- Buone prestazioni in media
- Utilizza bit di controllo che riportano le informazioni sugli accessi alle pagine
- La gestione più semplice prevede un bit di accesso nella tabella delle pagine
  - il bit viene posto a 1 quando la pagina viene acceduta e azzerato periodicamente da SO
  - in corrispondenza dell'azzeramento periodico, il SO incrementa una variabile di conteggio interna, per tutte le pagine con bit di accesso a 0
  - viene sostituita la pagina con bit di accesso a 0 e valore di conteggio più alto
- Esistono situazioni particolari in cui l'algoritmo ha prestazioni pessime

Fork()

- L’esecuzione della fork crea un nuovo processo che è l’immagine del padre
- Viene aggiornata la tabella dei processi, ma non viene allocata memoria fisica e i segmenti virtuali vengono condivisi
- All’atto di una scrittura in memoria, da parte di uno dei due processi, la pagina scritta viene effettivamente allocata (le pagine relative a segmenti virtuali contenenti dati vengono inizialmente poste con diritto di accesso in sola lettura)

Struttura Programma Eseguibile



Esercizio Riassuntivo (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. **Si definisca** la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi NPF, Spiazzamento fisico, NPL, Spiazzamento logico

Esercizio Riassuntivo (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. **Si definisca** la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi NPF, Spiazzamento fisico, NPL, Spiazzamento logico

a) NPF: 3, Spiazzamento fisico: 12, NPL: 3, Spiazzamento logico: 12

Esercizio Riassuntivo (2)

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: 8 K DX: 4 K PX: 4 K

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

Si inserisca in tabella 1 la struttura in pagine della memoria virtuale (mediante la notazione definita sopra: CX0 CX1 DX0 PX0 … CY0 …).

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

1) memoria logica

Esercizio Riassuntivo (2)

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: 8 K DX: 4 K PX: 4 K

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

Si inserisca in tabella 1 la struttura in pagine della memoria virtuale (mediante la notazione definita sopra: CX0 CX1 DX0 PX0 … CY0 …).

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

1) memoria logica

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

Esercizio Riassuntivo (3)

c. A un certo istante di tempo 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

- 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

Esercizio Riassuntivo (3) *soluzione*

c) Per facilitare la comprensione del contenuto della tabella 2 (tempo $t_{0)}$, riportato sotto, si forniscono le seguenti spiegazioni, seguendo la numerazione degli eventi:

1. 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
2. successivamente Q carica CQ1 (perché il programma X ha istruzione di partenza in pagina 1) e PQ0
3. 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)
4. Q carica DQ0 e raggiunge 3 pagine caricate
5. 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)
6. R 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*

Esercizio Riassuntivo (4)

d) A un certo istante di tempo $t_1 > t_0$ 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

d) Il contenuto della tabella 3 si spiega nel modo seguente

7. P termina e libera le pagine fisiche 1 e 4 (non la 0, perché CR0 rimane necessaria)
8. 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
9. 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	*CR0 (= CS0)*
1	~~*PS0*~~ *DS1*
2	~~*CQ1*~~ *DS0*
3	~~*PQ0*~~ ---
4	
5	
6	*PR1*
7	*PR2*

TdE / 11 Marzo 2011

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX, è caratterizzato dai parametri seguenti: l'indirizzo logico è di **16 bit**; l'indirizzo fisico è di **16 bit**; e la dimensione delle pagine è di **4096 byte**.

(a) Nel sistema vengono **attivati** i processi P, Q e R. Essi eseguono i programmi X e Y, e condividono un segmento dati. La dimensione **iniziale** dei segmenti dei due programmi è la seguente:

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

CY: 8 K DY: 4 K PY: 4 K COND: 4 K

Nel programma X il segmento COND è allocato lasciando **due** pagine libere dopo il segmento dati, mentre nel programma Y il segmento COND è allocato lasciando **una** pagina libera dopo il segmento dati.

Inserire in tabella la struttura in pagine della memoria virtuale dei due programmi X e Y (notazione: CX0, CX1, DX0, PX0, … CY0, …, COND0, ...).

indirizzo di pagina virtuale	X	Y
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
A		
B		
C		
D		
E		
F		

TdE / 11 Marzo 2011

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX, è caratterizzato dai parametri seguenti: l'indirizzo logico è di **16 bit**; l'indirizzo fisico è di **16 bit**; e la dimensione delle pagine è di **4096 byte**.

(a) Nel sistema vengono **attivati** i processi P, Q e R. Essi eseguono i programmi X e Y, e condividono un segmento dati. La dimensione **iniziale** dei segmenti dei due programmi è la seguente:

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

CY: 8 K DY: 4 K PY: 4 K COND: 4 K

Nel programma X il segmento COND è allocato lasciando **due** pagine libere dopo il segmento dati, mentre nel programma Y il segmento COND è allocato lasciando **una** pagina libera dopo il segmento dati.

Inserire in tabella la struttura in pagine della memoria virtuale dei due programmi X e Y (notazione: CX0, CX1, DX0, PX0, … CY0, …, COND0, ...).

indirizzo di pagina virtuale	X	Y
0	***CX0***	***CY0***
1	***CX1***	***CY1***
2	***DX0***	***DY0***
3	***DX1***	
4		***COND0***
5		
6	***COND0***	
7		
8		
9		
A		
B		
C		
D		
E		
F	***PX0***	***PY0***

TdE / 11 Marzo 2011

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

- le pagine fisiche di indirizzo 0 e 5 vengono allocate a un processo non significativo
- P viene creato (fork di P ed exec di X)
- Q viene creato (fork di Q ed exec di Y)
- Q alloca una pagina di memoria tramite sbrk, e accede a tale pagina
- Q accede alla pagina COND

Considerando le ipotesi seguenti, **compilare** le tabelle della situazione al tempo t_0 relative alla memoria fisica e al contenuto della MMU:

- il lancio in esecuzione di un programma avviene caricando solo la **pagina di codice** con l'istruzione di partenza, la **prima pagina dati** e una **pagina di pila**, in quest'ordine
- il caricamento di ulteriori pagine in memoria avviene **su richiesta** (on demand)
- il numero di pagine residenti di R è pari a **quattro**
- l'indirizzo esadecimale di partenza di X è **1800 Hex**
- l'indirizzo esadecimale di partenza di Y è **0404 Hex**
- si utilizza l'**algoritmo LRU** per la sostituzione di pagine di memoria, considerando che almeno una pagina di pila debba sempre rimanere in memoria
- l'allocazione delle pagine virtuali nelle pagine fisiche avviene sempre **in sequenza**
- all'inizio della sequenza di eventi la MMU è **vuota**

TdE / 11 Marzo 2011

situazione al tempo t_0

memoria fisica	
indirizzo fisico	**pagine allocate**
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
A	
B	
C	
D	
E	
F	

MMU			
proc.	**NPV**	**NPF**	**valid bit**

TdE / 11 Marzo 2011

situazione al tempo t_0

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*occupato*
1	*CP1*
2	*DP0*
3	*PP0*
4	*CQ0*
5	*occupato*
6	*(DQ0) COND*
7	*PQ0*
8	*DQ1*
9	
A	
B	
C	
D	
E	
F	

MMU			
proc.	**NPV**	**NPF**	**valid bit**
P	*CP1 / 1*	*1*	*1*
P	*DP0 / 2*	*2*	*1*
P	*PP0 / F*	*3*	*1*
Q	*CQ0 / 0*	*4*	*1*
Q	*(DQ0 / 2) COND / 4*	*6*	*1*
Q	*PQ0 / F*	*7*	*1*
Q	*DQ1 / 3*	*8*	*1*

TdE / 11 Marzo 2011

(c) A un certo istante $t_1 > t_0$ è terminata la sequenza di eventi seguente:

- P chiama una funzione che si trova nella pagina di codice corrente, e tale funzione esegue un accesso alla pagina dati il cui indirizzo virtuale assoluto è 3332 Hex
- dopo essere rientrato dalla funzione, P esegue fork e crea il processo R
- P accede alla pagina COND
- Q esegue exit
- R accede in scrittura a una struttura dati di tipo array, il cui indirizzo iniziale è 2002 Hex

situazione al tempo t_1

memoria fisica	
indirizzo fisico	**pagine allocate**
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
A	
B	
C	

MMU			
proc	**NPV**	**NPF**	**valid bit**

TdE / 11 Marzo 2011

(c) A un certo istante $t_1 > t_0$ è terminata la sequenza di eventi seguente:

- P chiama una funzione che si trova nella pagina di codice corrente, e tale funzione esegue un accesso alla pagina dati il cui indirizzo virtuale assoluto è 3332 Hex
- dopo essere rientrato dalla funzione, P esegue fork e crea il processo R
- P accede alla pagina COND
- Q esegue exit
- R accede in scrittura a una struttura dati di tipo array, il cui indirizzo iniziale è 2002 Hex

situazione al tempo t_1

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*occupato*
1	*CP1*
2	*(DP0) DR0*
3	*PP0*
4	*(CQ0)*
5	*occupato*
6	*(DQ0) COND*
7	*(PQ0)*
8	*(DQ1)*
9	*DP1*
A	*PR0*
B	
C	

MMU			
proc	**NPV**	**NPF**	**valid bit**
P	*CP1 / 1*	*1*	*1*
P	*(DP0 / 2) COND / 6*	*(2) 6*	*1*
P	*PP0 / F*	*3*	*1*
(Q)	*(CQ0 / 0*	*4*	*1 / 0*
(Q)	*(DQ0 / 2) (COND / 4)*	*6*	*1 / 0*
(Q)	*(PQ0 / F)*	*7*	*1 / 0*
(Q)	*(DQ1 / 3)*	*8*	*1 / 0*
P	*DP1 / 3*	*9*	*1*
R	*CR1 / 1*	*1*	*1*
R	*DR0 / 2*	*2*	*1*
R	*DR1 / 3*	*9*	
R	*PR0 / F*	*A*	

TdE / 7 Luglio 2010

Un sistema dotato di memoria virtuale con paginazione e segmentazione tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità 32 Kbyte, la memoria logica ha capacità di 64 KByte e la dimensione delle pagine è di 4096 byte.

(a) Nel sistema saranno attivati i processi P, Q e R, che eseguono i programmi X e Y, che condividono un segmento dati. La dimensione iniziale dei segmenti dei tre programmi è la seguente:

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

CY: 4 K DY: 8 K PY: 4 K COND: 4 K

Il segmento COND è allocato lasciando 2 pagine libere dopo il segmento dati di X e Y.

Inserire in tabella la struttura in pagine della memoria virtuale dei due programmi X e Y (notazione: CX0, CX1, DX0, PX0,… CY0, …, COND0, ...).

indirizzo di pagina virtuale	X	Y
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
A		
B		
C		
D		
E		
F		

TdE / 7 Luglio 2010

Un sistema dotato di memoria virtuale con paginazione e segmentazione tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità 32 Kbyte, la memoria logica ha capacità di 64 KByte e la dimensione delle pagine è di 4096 byte.

(a) Nel sistema saranno attivati i processi P, Q e R, che eseguono i programmi X e Y, che condividono un segmento dati. La dimensione iniziale dei segmenti dei tre programmi è la seguente:

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

CY: 4 K DY: 8 K PY: 4 K COND: 4 K

Il segmento COND è allocato lasciando 2 pagine libere dopo il segmento dati di X e Y.

Inserire in tabella la struttura in pagine della memoria virtuale dei due programmi X e Y (notazione: CX0, CX1, DX0, PX0,… CY0, …, COND0, ...).

indirizzo di pagina virtuale	X	Y
0	**CX0**	**CY0**
1	**CX1**	**DY0**
2	**DX0**	**DY1**
3		
4		
5	**COND0**	**COND0**
6		
7		
8		
9		
A		
B		
C		
D		
E		
F	**PX0**	**PY0**

TdE / 7 Luglio 2010

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

- creazione del processo P (fork di P ed exec di X)
- creazione del processo Q (fork di Q ed exec di Y)
- accesso alla pagina condivisa COND da parte di Q
- allocazione e accesso in sequenza di due pagine di memoria per P tramite sbrk
- accesso alla pagina DP1 da parte di P

Considerando le ipotesi seguenti, **<u>compilare</u>** le tabelle della situazione al tempo t_0 relative alla memoria fisica e al contenuto della MMU:

- il lancio in esecuzione di un programma avviene caricando solo la pagina di codice con l'istruzione di partenza, la prima pagina dati e una pagina di pila, in quest'ordine
- il caricamento di ulteriori pagine in memoria avviene on demand
- il numero di pagine residenti R è pari a 4
- l'indirizzo esadecimale di partenza di **X è 1200 Hex**
- si utilizza l'algoritmo LRU per la sostituzione di pagine di memoria, considerando che almeno una pagina di pila debba sempre rimanere in memoria
- l'allocazione delle pagine virtuali nelle pagine fisiche avviene sempre in sequenza
- all'inizio della sequenza di eventi la MMU è vuota, e che se è richiesta una nuova riga si utilizzi **sempre** la prima riga libera.

TdE / 7 Luglio 2010

situazione al tempo t_0

memoria fisica	
indirizzo fisico	**pagine allocate**
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
A	
B	
C	
D	
E	
F	

MMU			
proc.	**NPV**	**NPF**	**valid bit**

TdE / 7 Luglio 2010

situazione al tempo t_0

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*CP1*
1	*(DP0) DP2*
2	*PP0*
3	*CQ0*
4	*DQ0*
5	*PQ0*
6	*COND*
7	*DP1*
8	
9	
A	
B	
C	
D	
E	
F	

MMU			
proc.	**NPV**	**NPF**	**valid bit**
P	*CP1 / 1*	*0*	*1*
P	*(DP0 / 2) DP2 / 4*	*1*	*1*
P	*PP0 / F*	*2*	*1*
Q	*CQ0 / 0*	*3*	*1*
Q	*DQ0 / 1*	*4*	*1*
Q	*PQ0 / F*	*5*	*1*
Q	*COND / 5*	*6*	*1*
P	*DP1 / 3*	*7*	*1*

TdE / 7 Luglio 2010

(c) A un certo istante $t_1 > t_0$ è terminata la seguente sequenza di eventi:

- accesso alla pagina condivisa COND da parte di P
- Q esegue una fork e crea il processo R
- P esegue la exit e termina
- R richiede una nuova pagina di pila
- R esegue l'accesso in scrittura a una struttura dati il cui indirizzo iniziale è 1400 Hex

Completare le tabelle con la situazione al tempo t_1.

situazione al tempo t_1

memoria fisica	
indirizzo fisico	**pagine allocate**
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
A	
B	
C	

MMU			
proc	**NPV**	**NPF**	**Valid Bit**

TdE / 7 Luglio 2010

(c) A un certo istante $t_1 > t_0$ è terminata la seguente sequenza di eventi:

- accesso alla pagina condivisa COND da parte di P
- Q esegue una fork e crea il processo R
- P esegue la exit e termina
- R richiede una nuova pagina di pila
- R esegue l'accesso in scrittura a una struttura dati il cui indirizzo iniziale è 1400 Hex

Completare le tabelle con la situazione al tempo t_1.

situazione al tempo t_1

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*(CP1) PR1*
1	*(DP0) (DP2) PR0*
2	*(PP0) DR0*
3	*CQ0 = CR0*
4	*DQ0 (= DR0)*
5	*PQ0*
6	*COND*
7	*(DP1)*
8	
9	
A	
B	
C	

MMU			
proc	**NPV**	**NPF**	**Valid Bit**
(P) R	*(CP1 / 1) PR1 / E*	*0*	*1 / 0 / 1*
P	*(DP0/2) (DP2/4) (COND/5)*	*(6)*	*1 / 0*
P	*(PP0 / F)*	*2*	*1 / 0*
Q	*CQ0 / 0*	*3*	*1*
Q	*DQ0 /1*	*4*	*1*
Q	*PQ0 / F*	*5*	*1*
Q	*COND / 5*	*6*	*1*
P	*(DP / 3)*	*7*	*1 / 0*
R	*CR0 / 0*	*3*	*1*
R	*DR0 / 1*	*4 / 2*	*1*
R	*COND/5*	*6*	*1*
R	PR0 / F	*1*	*1*

TdE / 03 Settembre 2010

Un sistema dotato di memoria virtuale con paginazione e segmentazione tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità 64 Kbyte, la memoria logica ha capacità di 64 KByte e la dimensione delle pagine è di 4096 byte.

(a) Nel sistema vengono attivati i processi P, Q e R, che eseguono i programmi X e Y, che condividono un segmento dati. La dimensione iniziale dei segmenti dei tre programmi è la seguente:

CX:8 K DX:12 K PX: 4 K COND: 4 K

CY:8 K DY: 8 K PY: 4 K COND: 4 K

Il segmento COND è allocato lasciando 2 pagine libere dopo il segmento dati di X e Y.

Inserire in tabella la struttura in pagine della memoria virtuale dei due programmi X e Y (notazione: CX0, CX1, DX0, PX0,… CY0, …, COND0, ...).

TdE / 03 Settembre 2010

Un sistema dotato di memoria virtuale con paginazione e segmentazione tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità 64 Kbyte, la memoria logica ha capacità di 64 KByte e la dimensione delle pagine è di 4096 byte.

(a) Nel sistema vengono attivati i processi P, Q e R, che eseguono i programmi X e Y, che condividono un segmento dati. La dimensione iniziale dei segmenti dei tre programmi è la seguente:

CX:8 K DX:12 K PX: 4 K COND: 4 K

CY:8 K DY: 8 K PY: 4 K COND: 4 K

Il segmento COND è allocato lasciando 2 pagine libere dopo il segmento dati di X e Y.

Inserire in tabella la struttura in pagine della memoria virtuale dei due programmi X e Y (notazione: CX0, CX1, DX0, PX0,… CY0, …, COND0, …).

indirizzo di pagina virtuale	X	Y
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
A		
B		
C		
D		
E		
F		

TdE / 03 Settembre 2010

Un sistema dotato di memoria virtuale con paginazione e segmentazione tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità 64 Kbyte, la memoria logica ha capacità di 64 KByte e la dimensione delle pagine è di 4096 byte.

(a) Nel sistema vengono attivati i processi P, Q e R, che eseguono i programmi X e Y, che condividono un segmento dati. La dimensione iniziale dei segmenti dei tre programmi è la seguente:

CX:8 K DX:12 K PX: 4 K COND: 4 K

CY:8 K DY: 8 K PY: 4 K COND: 4 K

Il segmento COND è allocato lasciando 2 pagine libere dopo il segmento dati di X e Y.

Inserire in tabella la struttura in pagine della memoria virtuale dei due programmi X e Y (notazione: CX0, CX1, DX0, PX0,… CY0, …, COND0, ...).

indirizzo di pagina virtuale	X	Y
0	**CX0**	**CY0**
1	**CX1**	**CY1**
2	**DX0**	**DY0**
3	**DX1**	**DY1**
4	**DX2**	
5		
6		**COND**
7	**COND**	
8		
9		
A		
B		
C		
D		
E		
F	**PX0**	**PY0**

TdE / 03 Settembre 2010

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

- creazione del processo P (fork di P ed exec di X)
- accesso di P alla sua seconda pagina di dati
- creazione del processo Q (fork di Q ed exec di Y)
- allocazione e accesso in sequenza di una pagina di memoria per Q tramite sbrk
- accesso alla pagina COND da parte di P

Considerando le ipotesi seguenti, **<u>compilare</u>** le tabelle della situazione al tempo t_0 relative alla memoria fisica e al contenuto della MMU:

- il lancio in esecuzione di un programma avviene caricando solo la pagina di codice con l'istruzione di partenza, la prima pagina dati e una pagina di pila, in quest'ordine
- il caricamento di ulteriori pagine in memoria avviene on demand
- il numero di pagine residenti R è pari a 4
- l'indirizzo esadecimale di partenza di **X è 300 Hex** e quello di **Y è 1300 Hex**
- si utilizza l'algoritmo LRU per la sostituzione di pagine di memoria, considerando che almeno una pagina di pila debba sempre rimanere in memoria
- l'allocazione delle pagine virtuali nelle pagine fisiche avviene sempre in sequenza
- all'inizio della sequenza di eventi la MMU è vuota, e che se è richiesta una nuova riga si utilizzi **sempre** la prima riga libera.

TdE / 03 Settembre 2010

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

- creazione del processo P (fork di P ed exec di X)
- accesso di P alla sua seconda pagina di dati
- creazione del processo Q (fork di Q ed exec di Y)
- allocazione e accesso in sequenza di una pagina di memoria per Q tramite sbrk
- accesso alla pagina COND da parte di P

situazione al tempo t_0

memoria fisica	
indirizzo fisico	**pagine allocate**
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
A	
B	
C	
D	
E	
F	

MMU			
proc.	**NPV**	**NPF**	**valid bit**

TdE / 03 Settembre 2010

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

- creazione del processo P (fork di P ed exec di X)
- accesso di P alla sua seconda pagina di dati
- creazione del processo Q (fork di Q ed exec di Y)
- allocazione e accesso in sequenza di una pagina di memoria per Q tramite sbrk
- accesso alla pagina COND da parte di P

situazione al tempo t_0

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*CP0*
1	*(DP0) COND*
2	*PP0*
3	*DP1*
4	*CQ1*
5	
6	
7	
8	
9	
A	
B	
C	
D	
E	
F	

MMU			
proc.	**NPV**	**NPF**	**valid bit**
P	*CP0 / 0*	*0*	*1*
P	*(DP0 / 2) COND / 7*	*1*	*1*
P	*PP0 / F*	*2*	*1*
P	*DP1 / 3*	*3*	*1*
Q	*CQ1 / 1*	*4*	*1*
Q	*DQ0 / 2*	*5*	*1*
Q	*PQ0 / F*	*6*	*1*
Q	*DQ2 / 4*	*7*	*1*

TdE / 03 Settembre 2010

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

- creazione del processo P (fork di P ed exec di X)
- accesso di P alla sua seconda pagina di dati
- creazione del processo Q (fork di Q ed exec di Y)
- allocazione e accesso in sequenza di una pagina di memoria per Q tramite sbrk
- accesso alla pagina COND da parte di P

situazione al tempo t_0

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*CP0*
1	*(DP0) COND*
2	*PP0*
3	*DP1*
4	*CQ1*
5	*DQ0*
6	
7	
8	
9	
A	
B	
C	
D	
E	
F	

MMU			
proc.	**NPV**	**NPF**	**valid bit**
P	*CP0 / 0*	*0*	*1*
P	*(DP0 / 2) COND / 7*	*1*	*1*
P	*PP0 / F*	*2*	*1*
P	*DP1 / 3*	*3*	*1*
Q	*CQ1 / 1*	*4*	*1*
Q	*DQ0 / 2*	*5*	*1*
Q	*PQ0 / F*	*6*	*1*
Q	*DQ2 / 4*	*7*	*1*

TdE / 03 Settembre 2010

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

- creazione del processo P (fork di P ed exec di X)
- accesso di P alla sua seconda pagina di dati
- creazione del processo Q (fork di Q ed exec di Y)
- allocazione e accesso in sequenza di una pagina di memoria per Q tramite sbrk
- accesso alla pagina COND da parte di P

situazione al tempo t_0

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*CP0*
1	*(DP0) COND*
2	*PP0*
3	*DP1*
4	*CQ1*
5	*DQ0*
6	*PQ0*
7	
8	
9	
A	
B	
C	
D	
E	
F	

MMU			
proc.	**NPV**	**NPF**	**valid bit**
P	*CP0 / 0*	*0*	*1*
P	*(DP0 / 2) COND / 7*	*1*	*1*
P	*PP0 / F*	*2*	*1*
P	*DP1 / 3*	*3*	*1*
Q	*CQ1 / 1*	*4*	*1*
Q	*DQ0 / 2*	*5*	*1*
Q	*PQ0 / F*	*6*	*1*
Q	*DQ2 / 4*	*7*	*1*

TdE / 03 Settembre 2010

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

- creazione del processo P (fork di P ed exec di X)
- accesso di P alla sua seconda pagina di dati
- creazione del processo Q (fork di Q ed exec di Y)
- allocazione e accesso in sequenza di una pagina di memoria per Q tramite sbrk
- accesso alla pagina COND da parte di P

situazione al tempo t_0

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*CP0*
1	*(DP0) COND*
2	*PP0*
3	*DP1*
4	*CQ1*
5	*DQ0*
6	*PQ0*
7	*DQ2*
8	
9	
A	
B	
C	
D	
E	
F	

MMU			
proc.	**NPV**	**NPF**	**valid bit**
P	*CP0 / 0*	*0*	*1*
P	*(DP0 / 2) COND / 7*	*1*	*1*
P	*PP0 / F*	*2*	*1*
P	*DP1 / 3*	*3*	*1*
Q	*CQ1 / 1*	*4*	*1*
Q	*DQ0 / 2*	*5*	*1*
Q	*PQ0 / F*	*6*	*1*
Q	*DQ2 / 4*	*7*	*1*

TdE / 03 Settembre 2010

(c) A un certo istante $t_1 > t_0$ è terminata la sequenza di eventi seguente:

- P chiama una funzione che si trova nella pagina di codice corrente, e tale funzione esegue un a cesso alla pagina dati il cui indirizzo virtuale assoluto è 3332 Hex
- dopo essere rientrato dalla funzione, P esegue fork e crea il processo R
- P accede alla pagina COND
- Q esegue exit
- R accede in scrittura a una struttura dati di tipo array, il cui indirizzo iniziale è 2002 Hex

Completare le tabelle con la situazione al tempo t_1.

situazione al tempo t_1

memoria fisica	
indirizzo fisico	**pagine allocate**
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
A	
B	
C	

MMU			
proc	**NPV**	**NPF**	**valid bit**

TdE / 03 Settembre 2010

(c) A un certo istante $t_1 > t_0$ è terminata la sequenza di eventi seguente:

- P chiama una funzione che si trova nella pagina di codice corrente, e tale funzione esegue un a cesso alla pagina dati il cui indirizzo virtuale assoluto è 3332 Hex
- dopo essere rientrato dalla funzione, P esegue fork e crea il processo R
- P accede alla pagina COND
- Q esegue exit
- R accede in scrittura a una struttura dati di tipo array, il cui indirizzo iniziale è 2002 Hex

Completare le tabelle con la situazione al tempo t_1.

situazione al tempo t_1

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*occupato*
1	*CP1*
2	*(DP0) DR0*
3	*PP0*
4	*(CQ0)*
5	*occupato*
6	*(DQ0) COND*
7	*(PQ0)*
8	*(DQ1)*
9	*DP1*
A	*PR0*
B	
C	

MMU			
proc	**NPV**	**NPF**	**valid bit**
P	*CP1 / 1*	*1*	*1*
P	*(DP0 / 2) COND / 6*	*(2) 6*	*1*
P	*PP0 / F*	*3*	*1*
(Q)	*(CQ0 / 0*	*4*	*1 / 0*
(Q)	*(DQ0 / 2) (COND / 4)*	*6*	*1 / 0*
(Q)	*(PQ0 / F)*	*7*	*1 / 0*
(Q)	*(DQ1 / 3)*	*8*	*1 / 0*
P	*DP1 / 3*	*9*	*1*
R	*CR1 / 1*	*1*	*1*
R	*DR0 / 2*	*2*	*1*
R	*DR1 / 3*	*9*	
R	*PR0 / F*	*A*	

TdE / 03 Settembre 2010

(c) A un certo istante $t_1 > t_0$ è terminata la sequenza di eventi seguente:

- P chiama una funzione che si trova nella pagina di codice corrente, e tale funzione esegue un a cesso alla pagina dati il cui indirizzo virtuale assoluto è 3332 Hex
- dopo essere rientrato dalla funzione, P esegue fork e crea il processo R
- P accede alla pagina COND
- Q esegue exit
- R accede in scrittura a una struttura dati di tipo array, il cui indirizzo iniziale è 2002 Hex

Completare le tabelle con la situazione al tempo t_1.

situazione al tempo t_1

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*occupato*
1	*CP1*
2	*(DP0) DR0*
3	*PP0*
4	*(CQ0)*
5	*occupato*
6	*(DQ0) COND*
7	*(PQ0)*
8	*(DQ1)*
9	*DP1*
A	*PR0*
B	
C	

MMU			
proc	**NPV**	**NPF**	**valid bit**
P	*CP1 / 1*	*1*	*1*
P	*(DP0 / 2) COND / 6*	*(2) 6*	*1*
P	*PP0 / F*	*3*	*1*
(Q)	*(CQ0 / 0*	*4*	*1 / 0*
(Q)	*(DQ0 / 2) (COND / 4)*	*6*	*1 / 0*
(Q)	*(PQ0 / F)*	*7*	*1 / 0*
(Q)	*(DQ1 / 3)*	*8*	*1 / 0*
P	*DP1 / 3*	*9*	*1*
R	*CR1 / 1*	*1*	*1*
R	*DR0 / 2*	*2*	*1*
R	*DR1 / 3*	*9*	
R	*PR0 / F*	*A*	

TdE / 03 Settembre 2010

(c) A un certo istante $t_1 > t_0$ è terminata la sequenza di eventi seguente:

- P chiama una funzione che si trova nella pagina di codice corrente, e tale funzione esegue un a cesso alla pagina dati il cui indirizzo virtuale assoluto è 3332 Hex
- dopo essere rientrato dalla funzione, P esegue fork e crea il processo R
- P accede alla pagina COND
- Q esegue exit
- R accede in scrittura a una struttura dati di tipo array, il cui indirizzo iniziale è 2002 Hex

Completare le tabelle con la situazione al tempo t_1.

situazione al tempo t_1

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*occupato*
1	*CP1*
2	*(DP0) DR0*
3	*PP0*
4	*(CQ0)*
5	*occupato*
6	*(DQ0) COND*
7	*(PQ0)*
8	*(DQ1)*
9	*DP1*
A	*PR0*
B	
C	

MMU			
proc	**NPV**	**NPF**	**valid bit**
P	*CP1 / 1*	*1*	*1*
P	*(DP0 / 2) COND / 6*	*(2) 6*	*1*
P	*PP0 / F*	*3*	*1*
(Q)	*(CQ0 / 0*	*4*	*1 / 0*
(Q)	*(DQ0 / 2) (COND / 4)*	*6*	*1 / 0*
(Q)	*(PQ0 / F)*	*7*	*1 / 0*
(Q)	*(DQ1 / 3)*	*8*	*1 / 0*
P	*DP1 / 3*	*9*	*1*
R	*CR1 / 1*	*1*	*1*
R	*DR0 / 2*	*2*	*1*
R	*DR1 / 3*	*9*	
R	*PR0 / F*	*A*	

TdE / 03 Settembre 2010

(c) A un certo istante $t_1 > t_0$ è terminata la sequenza di eventi seguente:

- P chiama una funzione che si trova nella pagina di codice corrente, e tale funzione esegue un a cesso alla pagina dati il cui indirizzo virtuale assoluto è 3332 Hex
- dopo essere rientrato dalla funzione, P esegue fork e crea il processo R
- P accede alla pagina COND
- Q esegue exit
- R accede in scrittura a una struttura dati di tipo array, il cui indirizzo iniziale è 2002 Hex

Completare le tabelle con la situazione al tempo t_1.

situazione al tempo t_1

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*occupato*
1	*CP1*
2	*(DP0) DR0*
3	*PP0*
4	*(CQ0)*
5	*occupato*
6	*(DQ0) COND*
7	*(PQ0)*
8	*(DQ1)*
9	*DP1*
A	*PR0*
B	
C	

MMU			
proc	**NPV**	**NPF**	**valid bit**
P	*CP1 / 1*	*1*	*1*
P	*(DP0 / 2) COND / 6*	*(2) 6*	*1*
P	*PP0 / F*	*3*	*1*
(Q)	*(CQ0 / 0*	*4*	*1 / 0*
(Q)	*(DQ0 / 2) (COND / 4)*	*6*	*1 / 0*
(Q)	*(PQ0 / F)*	*7*	*1 / 0*
(Q)	*(DQ1 / 3)*	*8*	*1 / 0*
P	*DP1 / 3*	*9*	*1*
R	*CR1 / 1*	*1*	*1*
R	*DR0 / 2*	*2*	*1*
R	*DR1 / 3*	*9*	
R	*PR0 / F*	*A*	

TdE / 19 Novembre 2010

Un sistema dotato di memoria virtuale con paginazione e segmentazione tipo UNIX è caratterizzato dai parametri seguenti: la memoria fisica ha capacità 32 K byte, la memoria logica ha capacità di 64 K byte e la dimensione delle pagine è di 4096 byte.

(a) **Indicare** il numero di pagine fisiche e il numero di pagine logiche del sistema.

(b) Nel sistema saranno attivati i processi P e Q che eseguono i programmi X e Y, che condividono un segmento dati. La dimensione iniziale dei segmenti dei due programmi è la seguente:

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

CY: 12 K DY: 4 K PY: 4 K COND: 4 K

Il segmento COND è allocato lasciando 2 pagine libere dopo il segmento dati di X e Y.

Inserire in tabella la struttura in pagine della memoria virtuale dei due programmi X e Y (notazione: CX0, CX1, DX0, PX0,… CY0, …, COND0, …).

indirizzo di pagina virtuale	X	Y
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
A		
B		
C		
D		
E		
F		

TdE / 19 Novembre 2010

Un sistema dotato di memoria virtuale con paginazione e segmentazione tipo UNIX è caratterizzato dai parametri seguenti: la memoria fisica ha capacità 32 K byte, la memoria logica ha capacità di 64 K byte e la dimensione delle pagine è di 4096 byte.

(a) **Indicare** il numero di pagine fisiche e il numero di pagine logiche del sistema.

(b) Nel sistema saranno attivati i processi P e Q che eseguono i programmi X e Y, che condividono un segmento dati. La dimensione iniziale dei segmenti dei due programmi è la seguente:

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

CY: 12 K DY: 4 K PY: 4 K COND: 4 K

Il segmento COND è allocato lasciando 2 pagine libere dopo il segmento dati di X e Y.

Inserire in tabella la struttura in pagine della memoria virtuale dei due programmi X e Y (notazione: CX0, CX1, DX0, PX0,… CY0, …, COND0, …).

indirizzo di pagina virtuale	X	Y
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
A		
B		
C		
D		
E		
F		

TdE / 19 Novembre 2010

Un sistema dotato di memoria virtuale con paginazione e segmentazione tipo UNIX è caratterizzato dai parametri seguenti: la memoria fisica ha capacità 32 K byte, la memoria logica ha capacità di 64 K byte e la dimensione delle pagine è di 4096 byte.

(a) **Indicare** il numero di pagine fisiche e il numero di pagine logiche del sistema.

(b) Nel sistema saranno attivati i processi P e Q che eseguono i programmi X e Y, che condividono un segmento dati. La dimensione iniziale dei segmenti dei due programmi è la seguente:

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

CY: 12 K DY: 4 K PY: 4 K COND: 4 K

Il segmento COND è allocato lasciando 2 pagine libere dopo il segmento dati di X e Y.

Inserire in tabella la struttura in pagine della memoria virtuale dei due programmi X e Y (notazione: CX0, CX1, DX0, PX0,… CY0, …, COND0, …).

indirizzo di pagina virtuale	X	Y
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
A		
B		
C		
D		
E		
F		

TdE / 19 Novembre 2010

Un sistema dotato di memoria virtuale con paginazione e segmentazione tipo UNIX è caratterizzato dai parametri seguenti: la memoria fisica ha capacità 32 K byte, la memoria logica ha capacità di 64 K byte e la dimensione delle pagine è di 4096 byte.

(a) **Indicare** il numero di pagine fisiche e il numero di pagine logiche del sistema.

(b) Nel sistema saranno attivati i processi P e Q che eseguono i programmi X e Y, che condividono un segmento dati. La dimensione iniziale dei segmenti dei due programmi è la seguente:

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

CY: 12 K DY: 4 K PY: 4 K COND: 4 K

Il segmento COND è allocato lasciando 2 pagine libere dopo il segmento dati di X e Y.

Inserire in tabella la struttura in pagine della memoria virtuale dei due programmi X e Y (notazione: CX0, CX1, DX0, PX0,… CY0, …, COND0, …).

indirizzo di pagina virtuale	X	Y
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
A		
B		
C		
D		
E		
F		

TdE / 19 Novembre 2010

Un sistema dotato di memoria virtuale con paginazione e segmentazione tipo UNIX è caratterizzato dai parametri seguenti: la memoria fisica ha capacità 32 K byte, la memoria logica ha capacità di 64 K byte e la dimensione delle pagine è di 4096 byte.

(a) **Indicare** il numero di pagine fisiche e il numero di pagine logiche del sistema.

(b) Nel sistema saranno attivati i processi P e Q che eseguono i programmi X e Y, che condividono un segmento dati. La dimensione iniziale dei segmenti dei due programmi è la seguente:

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

CY: 12 K DY: 4 K PY: 4 K COND: 4 K

Il segmento COND è allocato lasciando 2 pagine libere dopo il segmento dati di X e Y.

Inserire in tabella la struttura in pagine della memoria virtuale dei due programmi X e Y (notazione: CX0, CX1, DX0, PX0,… CY0, …, COND0, …).

indirizzo di pagina virtuale	X	Y
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
A		
B		
C		
D		
E		
F		

TdE / 19 Novembre 2010

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

- P viene creato ("fork" di P ed exec di X)
- P esegue una "fork" e crea il processo figlio Q
- Q accede alla pagina condivisa COND
- Q chiama una funzione che si trova nella pagina di codice corrente, la quale funzione esegue un accesso in scrittura a una struttura dati il cui indirizzo virtuale assoluto è **2000 hex**
- P accede alla pagina dati

Considerando le ipotesi seguenti, **compilare** le tabelle della situazione al tempo t_0 relative alla memoria fisica e al contenuto della MMU:

- il lancio in esecuzione di un programma avviene caricando solo la pagina di codice con l'istruzione di partenza, la prima pagina dati e una pagina di pila, in quest'ordine
- il caricamento di ulteriori pagine in memoria avviene "on demand"
- il numero di pagine residenti **R** è pari a **3**
- l'indirizzo esadecimale di partenza di **X è 11AA hex** (indirizzo virtuale assoluto)
- l'indirizzo esadecimale di partenza di **Y è 2222 hex** (indirizzo virtuale assoluto)
- si utilizza l'algoritmo LRU per la sostituzione di pagine di memoria, considerando che almeno una pagina di pila debba sempre rimanere in memoria
- l'allocazione delle pagine virtuali nelle pagine fisiche avviene sempre in sequenza
- all'inizio della sequenza di eventi la MMU è vuota, e se è richiesta una nuova riga si utilizzi **sempre** la prima riga libera

TdE / 19 Novembre 2010

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

- P viene creato ("fork" di P ed exec di X)
- P esegue una "fork" e crea il processo figlio Q
- Q accede alla pagina condivisa COND
- Q chiama una funzione che si trova nella pagina di codice corrente, la quale funzione esegue un accesso in scrittura a una struttura dati il cui indirizzo virtuale assoluto è **2000 hex**
- P accede alla pagina dati

Considerando le ipotesi seguenti, **compilare** le tabelle della situazione al tempo t_0 relative alla memoria fisica e al contenuto della MMU:

- il lancio in esecuzione di un programma avviene caricando solo la pagina di codice con l'istruzione di partenza, la prima pagina dati e una pagina di pila, in quest'ordine
- il caricamento di ulteriori pagine in memoria avviene "on demand"
- il numero di pagine residenti **R** è pari a **3**
- l'indirizzo esadecimale di partenza di **X è 11AA hex** (indirizzo virtuale assoluto)
- l'indirizzo esadecimale di partenza di **Y è 2222 hex** (indirizzo virtuale assoluto)
- si utilizza l'algoritmo LRU per la sostituzione di pagine di memoria, considerando che almeno una pagina di pila debba sempre rimanere in memoria
- l'allocazione delle pagine virtuali nelle pagine fisiche avviene sempre in sequenza
- all'inizio della sequenza di eventi la MMU è vuota, e se è richiesta una nuova riga si utilizzi **sempre** la prima riga libera

TdE / 19 Novembre 2010

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

- P viene creato ("fork" di P ed exec di X)
- P esegue una "fork" e crea il processo figlio Q
- Q accede alla pagina condivisa COND
- Q chiama una funzione che si trova nella pagina di codice corrente, la quale funzione esegue un accesso in scrittura a una struttura dati il cui indirizzo virtuale assoluto è **2000 hex**
- P accede alla pagina dati

Considerando le ipotesi seguenti, **compilare** le tabelle della situazione al tempo t_0 relative alla memoria fisica e al contenuto della MMU:

- il lancio in esecuzione di un programma avviene caricando solo la pagina di codice con l'istruzione di partenza, la prima pagina dati e una pagina di pila, in quest'ordine
- il caricamento di ulteriori pagine in memoria avviene "on demand"
- il numero di pagine residenti **R** è pari a **3**
- l'indirizzo esadecimale di partenza di **X è 11AA hex** (indirizzo virtuale assoluto)
- l'indirizzo esadecimale di partenza di **Y è 2222 hex** (indirizzo virtuale assoluto)
- si utilizza l'algoritmo LRU per la sostituzione di pagine di memoria, considerando che almeno una pagina di pila debba sempre rimanere in memoria
- l'allocazione delle pagine virtuali nelle pagine fisiche avviene sempre in sequenza
- all'inizio della sequenza di eventi la MMU è vuota, e se è richiesta una nuova riga si utilizzi **sempre** la prima riga libera

TdE / 19 Novembre 2010

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

- P viene creato ("fork" di P ed exec di X)
- P esegue una "fork" e crea il processo figlio Q
- Q accede alla pagina condivisa COND
- Q chiama una funzione che si trova nella pagina di codice corrente, la quale funzione esegue un accesso in scrittura a una struttura dati il cui indirizzo virtuale assoluto è **2000 hex**
- P accede alla pagina dati

Considerando le ipotesi seguenti, **compilare** le tabelle della situazione al tempo t_0 relative alla memoria fisica e al contenuto della MMU:

- il lancio in esecuzione di un programma avviene caricando solo la pagina di codice con l'istruzione di partenza, la prima pagina dati e una pagina di pila, in quest'ordine
- il caricamento di ulteriori pagine in memoria avviene "on demand"
- il numero di pagine residenti **R** è pari a **3**
- l'indirizzo esadecimale di partenza di **X è 11AA hex** (indirizzo virtuale assoluto)
- l'indirizzo esadecimale di partenza di **Y è 2222 hex** (indirizzo virtuale assoluto)
- si utilizza l'algoritmo LRU per la sostituzione di pagine di memoria, considerando che almeno una pagina di pila debba sempre rimanere in memoria
- l'allocazione delle pagine virtuali nelle pagine fisiche avviene sempre in sequenza
- all'inizio della sequenza di eventi la MMU è vuota, e se è richiesta una nuova riga si utilizzi **sempre** la prima riga libera

TdE / 19 Novembre 2010

situazione al tempo t_0

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

MMU			
proc.	**NPV**	**NPF**	**valid bit**

TdE / 19 Novembre 2010

situazione al tempo t_0

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*CP1= CQ1*
1	*DP0 = (DQ0)*
2	*PP0*
3	*PQ0*
4	*(COND) DQ0*
5	
6	
7	

MMU			
proc.	**NPV**	**NPF**	**valid bit**
P	*CP1 / 1*	*0*	*1*
P	*DP0 /2*	*1*	*1*
P	*PP0/F*	*2*	*1*
Q	*CQ1 / 1*	*0*	*1*
Q	*(DQ0/2)(COND / 5) DQ0 / 2*	*(1) 4*	*1 / 0 / 1/0/1*
Q	*PQ0 /F*	*3*	*1*

TdE / 19 Novembre 2010

situazione al tempo t_0

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*CP1= CQ1*
1	*DP0 = (DQ0)*
2	*PP0*
3	*PQ0*
4	*(COND) DQ0*
5	
6	
7	

MMU			
proc.	**NPV**	**NPF**	**valid bit**
P	*CP1 / 1*	*0*	*1*
P	*DP0 /2*	*1*	*1*
P	*PP0/F*	*2*	*1*
Q	*CQ1 / 1*	*0*	*1*
Q	*(DQ0/2)(COND / 5) DQ0 / 2*	*(1) 4*	*1 / 0 / 1/0/1*
Q	*PQ0 /F*	*3*	*1*

TdE / 19 Novembre 2010

situazione al tempo t_0

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*CP1= CQ1*
1	*DP0 = (DQ0)*
2	*PP0*
3	*PQ0*
4	*(COND) DQ0*
5	
6	
7	

MMU			
proc.	**NPV**	**NPF**	**valid bit**
P	*CP1 / 1*	*0*	*1*
P	*DP0 /2*	*1*	*1*
P	*PP0/F*	*2*	*1*
Q	*CQ1 / 1*	*0*	*1*
Q	*(DQ0/2)(COND / 5) DQ0 / 2*	*(1) 4*	*1 / 0 / 1/0/1*
Q	*PQ0 /F*	*3*	*1*

TdE / 19 Novembre 2010

situazione al tempo t_0

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*CP1= CQ1*
1	*DP0 = (DQ0)*
2	*PP0*
3	*PQ0*
4	*(COND) DQ0*
5	
6	
7	

MMU			
proc.	**NPV**	**NPF**	**valid bit**
P	*CP1 / 1*	*0*	*1*
P	*DP0 /2*	*1*	*1*
P	*PP0/F*	*2*	*1*
Q	*CQ1 / 1*	*0*	*1*
Q	*(DQ0/2)(COND / 5) DQ0 / 2*	*(1) 4*	*1 / 0 / 1/0/1*
Q	*PQ0 /F*	*3*	*1*

TdE / 19 Novembre 2010

(d) A un certo istante $t_1 > t_0$ è terminata la seguente sequenza di eventi:

- Q esegue una "exec" e passa a eseguire il programma Y
- P esegue un accesso a COND
- Q richiede una nuova pagina di pila
- Q esegue un accesso a COND

Completare le tabelle con la situazione al tempo t_1.

situazione al tempo t_1

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

MMU			
proc	**NPV**	**NPF**	**valid bit**

TdE / 19 Novembre 2010

(d) A un certo istante $t_1 > t_0$ è terminata la seguente sequenza di eventi:

- Q esegue una "exec" e passa a eseguire il programma Y
- P esegue un accesso a COND
- Q richiede una nuova pagina di pila
- Q esegue un accesso a COND

Completare le tabelle con la situazione al tempo t_1.

situazione al tempo t_1

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

MMU			
proc	**NPV**	**NPF**	**valid bit**

TdE / 19 Novembre 2010

(d) A un certo istante $t_1 > t_0$ è terminata la seguente sequenza di eventi:

- Q esegue una "exec" e passa a eseguire il programma Y
- P esegue un accesso a COND
- Q richiede una nuova pagina di pila
- Q esegue un accesso a COND

Completare le tabelle con la situazione al tempo t_1.

situazione al tempo t_1

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

MMU			
proc	**NPV**	**NPF**	**valid bit**

TdE / 19 Novembre 2010

(d) A un certo istante $t_1 > t_0$ è terminata la seguente sequenza di eventi:

- Q esegue una "exec" e passa a eseguire il programma Y
- P esegue un accesso a COND
- Q richiede una nuova pagina di pila
- Q esegue un accesso a COND

Completare le tabelle con la situazione al tempo t_1.

situazione al tempo t_1

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

MMU			
proc	**NPV**	**NPF**	**valid bit**

TdE / 19 Novembre 2010

(d) A un certo istante $t_1 > t_0$ è terminata la seguente sequenza di eventi:

- Q esegue una "exec" e passa a eseguire il programma Y
- P esegue un accesso a COND
- Q richiede una nuova pagina di pila
- Q esegue un accesso a COND

Completare le tabelle con la situazione al tempo t_1.

situazione al tempo t_1

memoria fisica	
indirizzo fisico	**pagine allocate**
0	*CP1 = (CQ1)*
1	*(DP0) = (DQ0) COND*
2	*(PP0)*
3	*(PQ0)CQ2*
4	*(COND)(DQ0)(DQ0) PQ1*
5	*(PQ0)*
6	
7	

MMU			
proc	**NPV**	**NPF**	**valid bit**
P	*CP1 / 1*	*0*	*1*
P	*(DP0/2) COND/5*	*(1)1*	*1/0/1*
P	*PP0/F*	*2*	*1*
Q	*(CQ1/1) CQ2/2*	*(0) 3*	*1 / 0 / 1*
Q	*(DQ0/2)(COND/5)(DQ0/2) (DQ0/3) PQ1/E*	*(1) (4) (4)(4) 4*	*1/0/1/ 0/1/0/1/0/1*
Q	*(PQ0/F) (PQ0 / F) COND/6*	*(3) (5)1*	*1/0/1/0/1*

Alla prossima lezione

alessandro.nacci@polimi.it