POLITECNICO
MILANO 1863

Architettura dei calcolatori e sistemi operativi

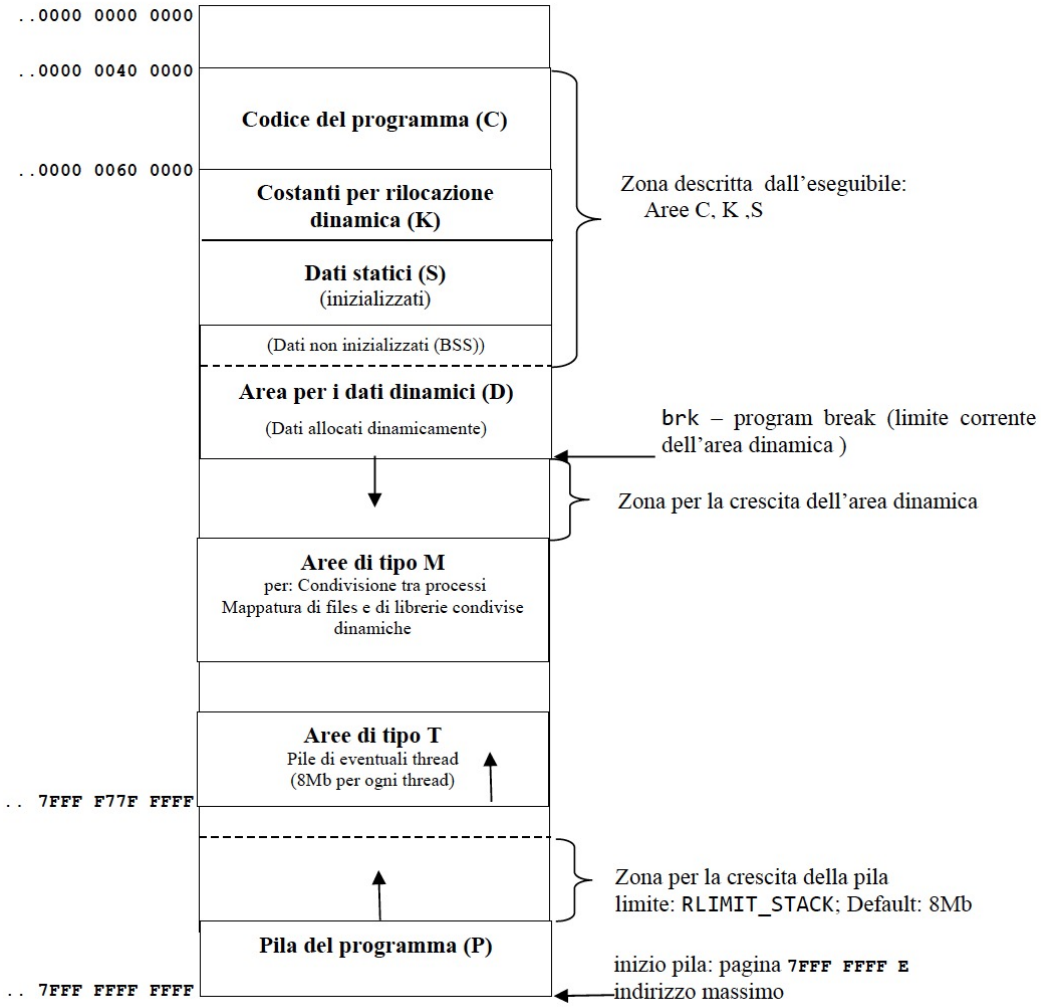
Esercizio completo di gestione della memoria

Testo

Data una memoria fisica di 10 pagine, con MAX_FREE = 3, MIN_FREE = 2, si esegua la simulazione della memoria con l'esecuzione dei seguenti eventi, considerando la dimensione iniziale della pila di 3 pagine

1. exec(2, 0, 0, 4, 4001FA,"X")
2. fork("Q")
3. write ("Pd0", "Pp0", "Pp1")
4. read("Pc0", "Pp0"), 4 kswapd
5. fork("R")
6. write("Pd1", "Pd2")
7. read("Pc0", "Pp0", "Pp1"), 4 kswapd
8. write("Pp2")
9. read("Pd0")
10. write("Pd0")
11. contextSwitch("R"), write ("Rp0")
12. write("Rd0")
13. exit("P"), write ("Pp2")
14. clone (S, c0)
15. mmap (0x30000000, 2, W, S, M, "F", 2)
16. read (Pm00), Write (Pm01)

Aree di memoria virtuale



exec(2, 0, 0, 4, 4001FA,"X") – creazione processo P

VMA di P

VMA	Start Address (NPV)	Dim	R/W	P/S	M/A	mapping
C	000000400	2	R	P	M	<X,0>
D	000000600	4	W	P	A	<-1,0>
P	7FFFFFFC	3	W	P	A	<-1,0>

La pagina iniziale del codice è all'indirizzo 4001FA quindi pagina c0

Tabella delle pagine:
<c0 :1 R> <c1 :- -> <d0 :- -> <d1 :- -> <d2 :- -> <d3 :- -> <p0 :2 W>
<p1 :- -> <p2 :- ->

NPV di PC: c0, NPV di SP: p0

exec(2, 0, 0, 4, 4001FA,"X") – creazione processo P

Memoria fisica

Numero pagina fisica (NPF)	Numero Pagina Virtuale	Numero Pagina Fisica	Numero pagina virtuale
00	<ZP>	01	Pc0 <X,0>
02	Pp0	03	
04		05	
06		07	
08		09	

exec(2, 0, 0, 4, 4001FA,"X") – creazione processo P

Stato del TLB

Pc0 : 01 - 0: 1: || Pp0 : 02 - 1: 1: ||

SWAP FILE: ----, ----, ----, ----, ----, ----,

LRU ACTIVE: PP0, PC0,

LRU INACTIVE:

fork("Q")

VMA di P invariata

Tabella delle pagine di P:

<c0 :1 R> <c1 :- -> <d0 :- -> <d1 :- -> <d2 :- -> <d3 :- ->

<p0 :3 W> <p1 :- -> <p2 :- ->

Stato del TLB (solo processo in esecuzione quindi P)

Pc0 : 01 - 0: 1: || Pp0 : 03 - 1: 1: ||

fork("Q")

VMA di Q

VMA	Start Address (NPV)	Dim	R/W	P/S	M/A	mapping
C	000000400	2	R	P	M	<X,0>
D	000000600	4	W	P	A	<-1,0>
P	7FFFFFFC	3	W	P	A	<-1,0>

Tabella delle pagine di Q:

c0 :1 R> <c1 :- -> <d0 :- -> <d1 :- -> <d2 :- -> <d3 :- ->
<p0 :2 D W> <p1 :- -> <p2 :- ->

fork("Q")

Memoria fisica

Numero pagina fisica (NPF)	Numero Pagina Virtuale	Numero Pagina Fisica	Numero pagina virtuale
00	<ZP>	01	Pc0/Qc0 <X,0>
02	Qp0 D	03	Pp0
04		05	
06		07	
08		09	

LRU ACTIVE: QP0, QC0, PP0, PC0,

write ("Pd0", "Pp0", "Pp1")

Tabella delle pagine di P

<c0 :1 R> <c1 :- -> <d0 : 4 W> <d1 :- -> <d2 :- -> <d3 :- -> <p0 :3 W> <p1 : 5 W> <p2 :- ->

Stato del TLB (solo processo in esecuzione quindi P)

Pc0 : 01 – 0: 1:	**Pp0 : 03 – 1: 1:**
Pd0 : 04 – 1: 1:	**Pp1 : 05 – 1: 1:**

NPV di PC: c0, NPV di SP: p1

LRU ACTIVE: PP1, PD0, Qp0, QC0, PP0, PC0

Processo Q invariato

4. read("Pc0", "Pp0"), 4 kswapd

- Esegue 4 volte la lettura e kswapd e fa un'ultima lettura
- Nessuna modifica alla memoria fisica
- Si modifica la pila di P: la cima torna a essere p0

NPV di SP: p0 NPV di PC: c0

Stato del TLB

Pc0 : 01 – 0: 1:	**Pp0 : 03 – 1: 1:**
Pd0 : 04 – 1: 0:	**Pp1 : 05 – 1: 0:**

LRU ACTIVE: PP0, PC0

LRU INACTIVE: pp1, pd0, qp0, qc0

5. fork("R")

P esegue la fork: cambia la pagina fisica della cima della pila di P (p0) che viene lasciata al figlio e marcata D

Numero pagina fisica (NPF)	Numero Pagina Virtuale	Numero Pagina Fisica	Numero pagina virtuale
00	<ZP>	01	Pc0/Qc0/Rc0 <X,0>
02	Qp0 D	03	Rp0 D
04	Pd0/Rd0	05	Pp1/Rp1
06	Pp0	07	
08		09	

5. fork("R")

Tabella delle pagine di P

<c0 :1 R> <c1 :- -> <d0 : 4 R> <d1 :- -> <d2 :- -> <d3 :- -> <p0 :6 W> <p1 : 5 R> <p2 :- ->

Nota: le due pagine condivise con R (d1 e p1) per la COW sono diventate R (da W)

TLB (processo P in esecuzione)

Pc0 : 01 – 0: 1:	**Pp0 : 06 – 1: 1:**
Pd0 : 04 – 1: 0:	**Pp1 : 05 – 1: 0:**

LRU ACTIVE: RP0, RC0, PP0, PC0

LRU INACTIVE: rp1, rd0, pp1, pd0, qp0, qc0

5. fork("R")

Processo Q: invariato

Processo R:

- VMA identica a quella del padre P
- Tabella delle pagine

<c0 :1 R> <c1 :- -> <d0 : 4 R> <d1 :- -> <d2 :- -> <d3 :- ->
<p0 :3 D W> <p1 : 5 R> <p2 :- ->

- NPV di PC di R: c0 NPV di SP di R: p0

6. write("Pd1", “Pd2”)

Le due pagine devono essere allocate in memoria:

- FreePages iniziale = 3
- Posso allocare e scrivere Pd1
- FreePages = 2 = MinFree ➔ PFRA ➔ toFree = maxFree – FreePages + RequiredPages = 3 – 2 + 1 =2
- Scelta delle pagine da liberare dalla Inactive List: rp1, rd0, pp1, pd0, qp0, qc0
    - Partendo dalla coda verifico se la pagina è condivisa, nel caso la salto finché non trovo tutte le pagine condivise
    - Selezione: qp0 (pagina fisica 2), pd0/rd0 (pagina fisica 4)
    - Entrambe le pagine non sono mappate su file ➔ qp0 ha D =1 mentre pd0 ha bit D=1 nel TLB ➔ richiedono swap_out nello Swap File (s0, s1)

6. write("Pd1", “Pd2")

Memoria fisica

Numero pagina fisica (NPF)	Numero Pagina Virtuale	Numero Pagina Fisica	Numero pagina virtuale
00	<ZP>	01	Pc0/Qc0/Rc0 <X,0>
02	Pd2	03	Rp0 D
04		05	Pp1/Rp1
06	Pp0	07	Pd1
08		09	

SWAP FILE: Qp0, Pd0/Rd0

LRU ACTIVE: PD2, PD1, RP0, RC0, PP0, PC0

LRU INACTIVE: rp1, pp1, qc0

6. write("Pd1", “Pd2”)

Tabella delle pagine di P

<c0 : 1 R> <c1 :- -> <d0 : s1 R> <d1 :7 W> <d2 : 2 W> <d3 :- -> <p0 : 6 W> <p1 : 5 R> <p2 :- ->

Tabella delle pagine di Q

<c0 : 1 R> <c1 :- -> <d0 : - -> <d1 :- -> <d2 : - -> <d3 :- ->
<p0 : s0 W> <p1 : - -> <p2 :- ->

Tabella delle pagine di R

<c0 :1 R> <c1 :- -> <d0 : s1 R> <d1 :- -> <d2 :- -> <d3 :- ->
<p0 :3 D W> <p1 : 5 R> <p2 :- ->

TLB del processo in esecuzione P (elimino da TLB la riga della pagina swappata Pd0)

Pc0 : 01 – 0: 1:	**Pp0 : 06 – 1: 1:**
Pd2 : 02 – 1: 1:	**Pp1 : 05 – 1: 0:**
Pd1 : 07 – 1: 1:	

7. read("Pc0", "Pp0", ”Pp1"), 4 kswapd

Le 3 pagine sono presenti in memoria quindi cambiano solo TLB, liste LRU e SP.

NPV di SP: p1

Stato del TLB

Pc0 : 01 – 0: 1:	**Pp0 : 06 – 1: 1:**
Pd2 : 02 – 1: 0:	**Pp1 : 05 – 1: 1:**
Pd1 : 07 – 1: 0:	

LRU ACTIVE: PP0. PC0, PP1

LRU INACTIVE: pd2, pd1, rp0, rc0, rp1, qc0

8. write("Pp2”)

- 3 pagine libere in memoria fisica, quindi l’allocazione della pagina non richiede l’invocazione di PFRA
- Si modifica la VMA della pila di P perché sono allocate inizialmente 3 pagine, di cui l’ultima lasciata libera per la crescita. Va quindi modificato l’indirizzo di inizio della VMA della pila solo di P da 7FFFFFFFC a 7FFFFFFFB (4 pagine)
- Si modifica NPV di SP: p2
- Tabella delle pagine di P

<c0 : 1 R> <c1 :- -> <d0 : s1 R> <d1 :7 W> <d2 : 2 W> <d3 :- -> <p0 : 6 W> <p1 : 5 R> <p2 : 4 W> <p3: - ->

LRU ACTIVE: PP2, PP0, PC0, PP1

LRU INACTIVE: pd2, pd1, rp0, rc0, rp1, qc0

8. write("Pp2”)

Memoria fisica

Numero pagina fisica (NPF)	Numero Pagina Virtuale	Numero Pagina Fisica	Numero pagina virtuale
00	<ZP>	01	Pc0/Qc0/Rc0 <X,0>
02	Pd2	03	Rp0 D
04	Pp2	05	Pp1/Rp1
06	Pp0	07	Pd1
08		09	

Stato del TLB

Pc0 : 01 – 0: 1:	**Pp0 : 06 – 1: 1:**
Pd2 : 02 – 1: 0:	**Pp1 : 05 – 1: 1:**
Pd1 : 07 – 1: 0:	**Pp2: 04 – 1: 1:**

9. Read (“Pd0”)

- Pd0 non è in memoria fisica, FreePages = 2 ➔ PFRA: necessario liberare 2 pagine
- Analisi di LRU INACTIVE: pd2, pd1, rp0, rc0, rp1, qc0
- Possibile deallocare rp0 da pagina fisica 3 (marcata D) e pd1 da pagina fisica 7 (D a 1 nel TLB)
  - Rp0 e pd1 sono entrambe pagine anonime quindi necessario invocare swap_out per salvarle nello Swap File (s2 e s3)

  SWAP FILE: Qp0, Pd0/Rd0, Rp0, Pd1
- Swap_in di Pd0/Rd0: la pagina viene copiata in memoria fisica, la copia nella Swap Area non viene eliminata, la pagina letta in memoria è marcata in sola lettura (R) – pagina fisica 3
- Nel Swap_Cache_Index viene inserito un descrittore che contiene i riferimenti alla pagina fisica e al Page Slot < s1, 3>
  - Swap_map_counter = 2

9. Read (“Pd0”)

Inserimento nelle LRU list:

- la NPV che è stata richiesta (Pd0), che ha causato lo Swap_in, viene inserita in testa alla active con ref = 1
- eventuali altre NPV condivise all’interno della stessa pagina (Rd0) vengono poste in coda alla inactive con ref = 0

LRU ACTIVE: PD0, PP2, PP0, Pc0, PP1

LRU INACTIVE: pd2, rc0, rp1, qc0, rd0

9. Read (“Pd0”)

Memoria fisica

Numero pagina fisica (NPF)	Numero Pagina Virtuale	Numero Pagina Fisica	Numero pagina virtuale
00	<ZP>	01	Pc0/Qc0/Rc0 <X,0>
02	Pd2	03	Pd0/Rd0
04	Pp2	05	Pp1/Rp1
06	Pp0	07	
08		09	

Stato del TLB

Pc1 : 01 – 0: 1:	**Pp0 : 06 – 1: 1:**
Pd2 : 02 – 1: 0:	**Pp1 : 05 – 1: 1:**
Pd0 : 03 – 0: 1:	**Pp2 : 04 – 1: 1:**

9. Read (“Pd0”)

Tabella delle pagine di P

<c0 : 1 R> <c1 :- -> <d0 : 3 R> <d1 :s3 W> <d2 : 2 W> <d3 :- ->
<p0 : 6 W> <p1 : 5 R> <p2 : 4 W> <p3: - ->

NPV di PC: c0, NPV di SP: p2

Processo Q: invariato

Tabella delle pagine di R

<c0 :1 R> <c1 :- -> <d0 : 3 R> <d1 :- -> <d2 :- -> <d3 :- ->
<p0 :s2 W> <p1 : 5 R> <p2 :- ->

10. Write (“Pd0”)

La pagina fisica 3 è condivisa Pd0 e Rd0 e ha protezione R

- si verifica un Page Fault per errore di protezione:
  - Allocazione di una nuova pagina fisica
  - La pagina diventa PRIVATE e non appartiene più allo Swap File
  - La sua protezione cambia in W
  - Il contatore swap_map_counter viene decrementato (se swap_map_counter = 0 il Page Slot viene liberato nello Swap File)

Tabella delle pagine di P

<c0 : 1 R> <c1 :- -> <d0 : 7 W> <d1 :s3 W> <d2 : 2 W> <d3 :- ->
<p0 : 6 W> <p1 : 5 R> <p2 : 4 W> <p3: - ->

SWAP FILE: : Qp0, ~~(Pd0/)~~Rd0, Rp0, Pd1

10. Write (“Pd0”)

Memoria fisica

Numero pagina fisica (NPF)	Numero Pagina Virtuale	Numero Pagina Fisica	Numero pagina virtuale
00	<ZP>	01	Pc0/Qc0/Rc0 <X,0>
02	Pd2	03	Rd0
04	Pp2	05	Pp1/Rp1
06	Pp0	07	Pd0
08		09	

Stato del TLB

Pc1 : 01 – 0: 1:	**Pp0 : 06 – 1: 1:**
Pd2 : 02 – 1: 0:	**Pp1 : 05 – 1: 1:**
Pd0 : 07 – 1: 1:	**Pp2 : 04 – 1: 1:**

10. Write (“Pd0”)

Tabella delle pagine di R

<c0 :1 R> <c1 :- -> <d0 : 3 R> <d1 :- -> <d2 :- -> <d3 :- ->
<p0 :s2 W> <p1 : 5 R> <p2 :- ->

Nota: rimane invariata e non devo marcare D la pagina d0 perché quando avviene la duplicazione nel TLB la pagina non era marcata Dirty

11. contextSwitch("R"), write ("Rp0")

Va in esecuzione R: pagina di codice c0 (già presente in memoria fisica) e pagina della cima della pila p0 (non presente in memoria fisica)

- Necessario allocare una pagina fisica ma FreePages = 2
- Invoca PFRA per allocare 2 pagine: analisi della LRU INACTIVE list: pd2, rc0, rp1, qc0, rd0
  - rd0 (pagina fisica 3) e pd2 (pagina fisica 2) non sono condivisi con altri processi
  - Rd0 è già presente su Swap File e non è stata modificata quindi non deve essere copiata (verificato mediante lo swap cache index)
  - Pd2 non è presente nello Swap File e va quindi copiata nello slot s4

11. contextSwitch("R"), Write ("Rp0")

La scrittura di Rp0 fa diventare lo swap_map_counter = 0 quindi Rp0 viene cancellata dallo Swap File

SWAP FILE: Qp0, Rd0, ~~Rp0~~, Pd1, Pd2

Flush del TLB:

- Marcare con D nella tabella delle pagine di P e nel descrittore di pagina fisica della memoria le pagine di P che nel TLB hanno il bit Dirty = 1
- Caricare nel TLB le pagine di R in esecuzione c0 e p0
- Stato del TLB (R in esecuzione) dopo scrittura di Rp0

Rc0 : 01 – 0: 1:	**Rp0 : 02– 1: 1:**

11. contextSwitch("R"), write ("Rp0")

Memoria fisica

Numero pagina fisica (NPF)	Numero Pagina Virtuale	Numero Pagina Fisica	Numero pagina virtuale
00	<ZP>	01	Pc0/Qc0/Rc0 <X,0>
02	Rp0	03	
04	Pp2 D	05	Pp1/Rp1 D
06	Pp0 D	07	Pd0 D
08		09	

LRU ACTIVE: RP0, PD0, PP2, PP0, PC0, PP1
LRU INACTIVE: rc0, rp1, qc0

11. contextSwitch("R"), write ("Rp0")

Tabella delle pagine di P

<c0 : 1 R> <c1 :- -> <d0 : 7 D W> <d1 :s3 W> <d2 : s5 W> <d3 :- -> <p0 : 6 D W> <p1 : 5 D R> <p2 : 4 D W> <p3: - ->

Tabella delle pagine di R

<c0 : 1 R> <c1 :- -> <d0 : s1 R> <d1 :- -> <d2 :- -> <d3 :- -> <p0 : 2 W> <p1 : 5 D R> <p2 :- ->

NPV di PC: c0 NPV di SP: p0

12. Write (“Rd0”)

- Rd0 non è presente in memoria fisica, ma è presente nello Swap File, con swap_map_counter = 1
- FreePages = 3 quindi è possibile attivare Swap_in e copiare Rd0 da Swap File in pagina fisica 3 ed eliminarlo dallo Swap File essendo richiesta una scrittura

Tabella delle pagine di R

<c0 : 1 R> <c1 :- -> <d0 : 3 W> <d1 :- -> <d2 :- -> <d3 :- ->
<p0 : 2 W> <p1 : 5 D R> <p2 :- ->

Swap File: Qp0, ~~Rd0,~~ ---, Pd1, Pd2

LRU ACTIVE: RD0, RP0, PD0, PP2, PP0, PC0, PP1

LRU INACTIVE: rc0, rp1, qc0

12. Write (“Rd0”)

Memoria Fisica

Numero pagina fisica (NPF)	Numero Pagina Virtuale	Numero Pagina Fisica	Numero pagina virtuale
00	<ZP>	01	Pc0/Qc0/Rc0 <X,0>
02	Rp0	03	Rd0
04	Pp2 D	05	Pp1/Rp1 D
06	Pp0 D	07	Pd0 D
08		09	

Stato del TLB

Rc0 : 01 – 0: 1:	**Rp0 : 02– 1: 1:**
Rd0 : 03 – 1: 1:	

13. Exit (“P”), write (“Pp2”)

Il processo in esecuzione R esegue la exit e torna in esecuzione P che esegue una scrittura nella pagina in cima alla pila

La exit causa:

- Eliminazione memoria virtuale di R e della tabella delle pagine
- Eliminazione delle pagine di R dalla memoria fisica e dalle LRU list
- Flush del TLB e caricamento pagine di PC e SP del processo in esecuzione P
- Processo P NPV di PC: c0 NPV di SP: p2
- Entrambe le pagine sono già in memoria fisica ➔Tabella delle pagine di P rimane invariata
- LRU ACTIVE: PD0, PP2, PP0, PC0, PP1
- LRU INACTIVE: qc0
- Swap File: Qp0, ---, ---, Pd1, Pd2

13. Exit (“P”), write (“Pp2”)

Memoria Fisica

Numero pagina fisica (NPF)	Numero Pagina Virtuale	Numero Pagina Fisica	Numero pagina virtuale
00	<ZP>	01	Pc0/Qc0 <X,0>
02		03	
04	Pp2 D	05	Pp1 D
06	Pp0 D	07	Pd0 D
08		09	

Stato del TLB

Pc0 : 01 – 0: 1:	**Pp2 : 04– 1: 1:**

14. Clone (S, c0)

Creazione di un thread S la cui funzione inizia a pagina c0

- Thread opera sulle stesse pagine virtuali del processo che lo crea
- Tabella delle pagine condivisa con il processo che lo crea
- Indirizzo iniziale della pila: 0x7FFFF77FF (NPV)
- Assumiamo per esercizi la creazione di due pagine di pila nella tabella delle pagine (in realta' 8 MB)
- La prima pagina della pila viene allocata fisicamente in memoria perché scritta dalla clone
- Eventuali pile di altri thread sono allocate in sequenza lasciando libera una pagina di interposizione

14. Clone (S, c0)

VMA di P

VMA	Start Address (NPV)	Dim	R/W	P/S	M/A	mapping
C	000000400	2	R	P	M	<X,0>
D	000000600	4	W	P	A	<-1,0>
T0	7FFFF77FE	2	W	P	A	<-1,0>
P	7FFFFFFFC	3	W	P	A	<-1,0>

Tabella delle pagine di P

<c0 : 1 R> <c1 :- -> <d0 : 7 D W> <d1 :s3 W> <d2 : s5 W> <d3 :- -> <p0 : 6 D W> <p1 : 5 D R> <p2 : 4 D W> <p3: - -> <t00 : 2 W> <t01 : - ->

14. Clone (S, c0)

Memoria fisica

Numero pagina fisica (NPF)	Numero Pagina Virtuale	Numero Pagina Fisica	Numero pagina virtuale
00	<ZP>	01	PSc0/Qc0 <X,0>
02	PSt00	03	
04	PSp2 D	05	PSp1 D
06	PSp0 D	07	PSd0 D
08		09	

Stato del TLB

Pc0 : 01 – 0: 1:	**Pp2 : 04– 1: 1:**
PSt00: 02 - 1: 1	

15. mmap (0x30000000, 2, W, S, M, “F”, 2)

VMA di P

VMA	Start Address (NPV)	Dim	R/W	P/S	M/A	mapping
C	000000400	2	R	P	M	<X,0>
D	000000600	4	W	P	A	<-1,0>
M0	000030000	2	W	S	M	<F,2>
T0	7FFFF77FE	2	W	P	A	<-1,0>
P	7FFFFFFFC	3	W	P	A	<-1,0>

Tabella delle pagine di P

<c0 : 1 R> <c1 :- -> <d0 : 7 D W> <d1 :s3 W> <d2 : s5 W> <d3 :- -> <p0 : 6 D W> <p1 : 5 D R> <p2 : 4 D W> <p3: - -> <t00 : 2 W> <t01 : - -> <m00: - -> <m01: - ->

Memoria fisica invariata

16. Write (Pm01)

La VMA è shared, quindi la scrittura alloca una nuova pagina in memoria fisica

- se fosse stata private avrei dovuto caricare la pagina dal file e poi sdoppiarla in una nuova pagina fisica per la scrittura

Numero pagina fisica (NPF)	Numero Pagina Virtuale	Numero Pagina Fisica	Numero pagina virtuale
00	<ZP>	01	PSc0/Qc0 <X,0>
02	PSt00	03	Pm01 /<F,2>
04	PSp2 D	05	PSp1 D
06	PSp0 D	07	PSd0 D
08		09	

16. Read (Pm00), Write (Pm01)

Tabella delle pagine di P

<c0 : 1 R> <c1 :- -> <d0 : 7 D W> <d1 :s3 W> <d2 : s5 W> <d3 :- -> <p0 : 6 D W> <p1 : 5 D R> <p2 : 4 D W> <p3: - -> <t00 : 2 W> <t01 : - -> <m00: - -> <m01: 3 W>

Stato del TLB

Pc0 : 01 – 0: 1:	**Pp2 : 04– 1: 1:**
PSt00: 02 - 1: 1:	**Pm01: 03 – 1: 1:**