3911037S
HARIHARAN BP

CAE-II OS

PART-B

(7) → Consider 5 philosophers who spend their lives thinking and eating. The philosophers share a circular table surrounded by 5 chairs; each belonging to one philosopher.

→ In center of the table is a bowl of rice, and the table is laid with 5 single chop sticks. When a philosopher thinks, he/she does not interact with her colleagues.

→ From time-to-time, a philosopher gets hungry & tries to pick up the chop stick closest to her.

→ They can pick only one chop stick at a time. He/she eats without releasing his/her chop sticks. When she's/he's finished eating, puts down both of chop sticks and starts thinking again.

→ The problem is considered as a classic synchronization problem as it an example of a large class of concurrency control problems. One simple solution is to represent each chop stick with a semaphore.

→ Problems with Semaphores

① Incorrect use of semaphore operations.

ⓐ Signal (mutex) .... wait (mutex)

ⓑ wait (mutex) ... wait (mutex)

ⓒ Omitting of wait (mutex) or signal (mutex) (or both)

② Deadlock and ~~starvation~~ are possible.

Monitor solution for Dining Philosopher

~~Mont~~

```
monitor Dining Phil
{
    enum (thinking; hungry, eating) state [5];
    condition self [5];
    void pickup (int i).
    {
        state [i] = hungry;
        test [i];
        ~~if state~~
        ~~if (state[i] = thinking~~
        if (state [i] != eating)
            self[i]. wait;
    }
```

```
void putdown (int i)
{
    state[i] = thinking;
        // test left and right neighbors.
    test((i+4) % 5);
    test((i+1) % 5);
}
void test( int i)
{
    if (( state[(i+4)%5] != eating) &&
        (state[i] == hungry) &&
        ( state[(i+1) % 5)] != eating))
    {
        state[i] = eating;
        self[i].signal();
    }
}
initialization_code()
{
    for(int i = 0; i < 5; i++)
        state[i] = thinking;
}
```

3911037J

→ Each Philosopher 'i' invokes the operations pickup() and putdown() in the following sequence:

DiningPhil.pickup(i);

eat

DiningPhil.putdown(i);

→ No deadlock but starvation is possible

9(i) The following are Page Replacement algorithm.

(1) FIFO (First in First Out)

→ The page has to be replaced first and which page has to be deleted first

(2) → Replace the pages that has been in the RAM / Rom for the longest time period (Least used)

for Eg:

String: 7 0 1 2 0 3 0 4 2 3 0 3 1 2 0

Page frames															
F_1	(7)	7	7	(2)	2	2	2	(4)	4	4	0	0	0	0	Present 0
F_2		(0)	0	0	Present 0	(3)	3	3	(2)	2	2	2	1	1	1
F_3			(1)	1	1	1	(0)	0	0	(3)	3	3 Present hit	3	2	2
	*	*	*	*	Hit	*	*	*	*	*	*		*	*	hit

Page hit : Hits → 3

Page faults :- * → 12

Ref string → 15

② LRU :- Last Recently Used.

→ 3 frames = 3 page no. (compare)

→ left side page no. are replaced with another

Eg :-

Consider two zero's as 1

7 0 1 2 0 3 0 4 2 3 0 3 1 2 0

Consider 2 3's as 1

F1	7	7	(7)	2	2	2	(2)	4	4	(4)	0	0	(0)	2	2
F2		0	(0)	0	0	0	0	0	(0)	3	3	3	3	(3)	0
F3			1	1	(1)	3	3	(3)	2	2	2	(2)	1	1	1
	*	*	*	*	hit	*	hit	~~hit~~ *	*	*	*	hit	*	*	*

Page hit = 3

" fault = 12

3911037+3

(iii) ~~OR ii OPTIMAL Alg~~

(iii) Optimal Algorithm

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7
7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7
	0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0
		1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1
											3						

* * * * * * * *

miss miss miss miss hit hit hit miss

Page hit → 9

" fault → 9

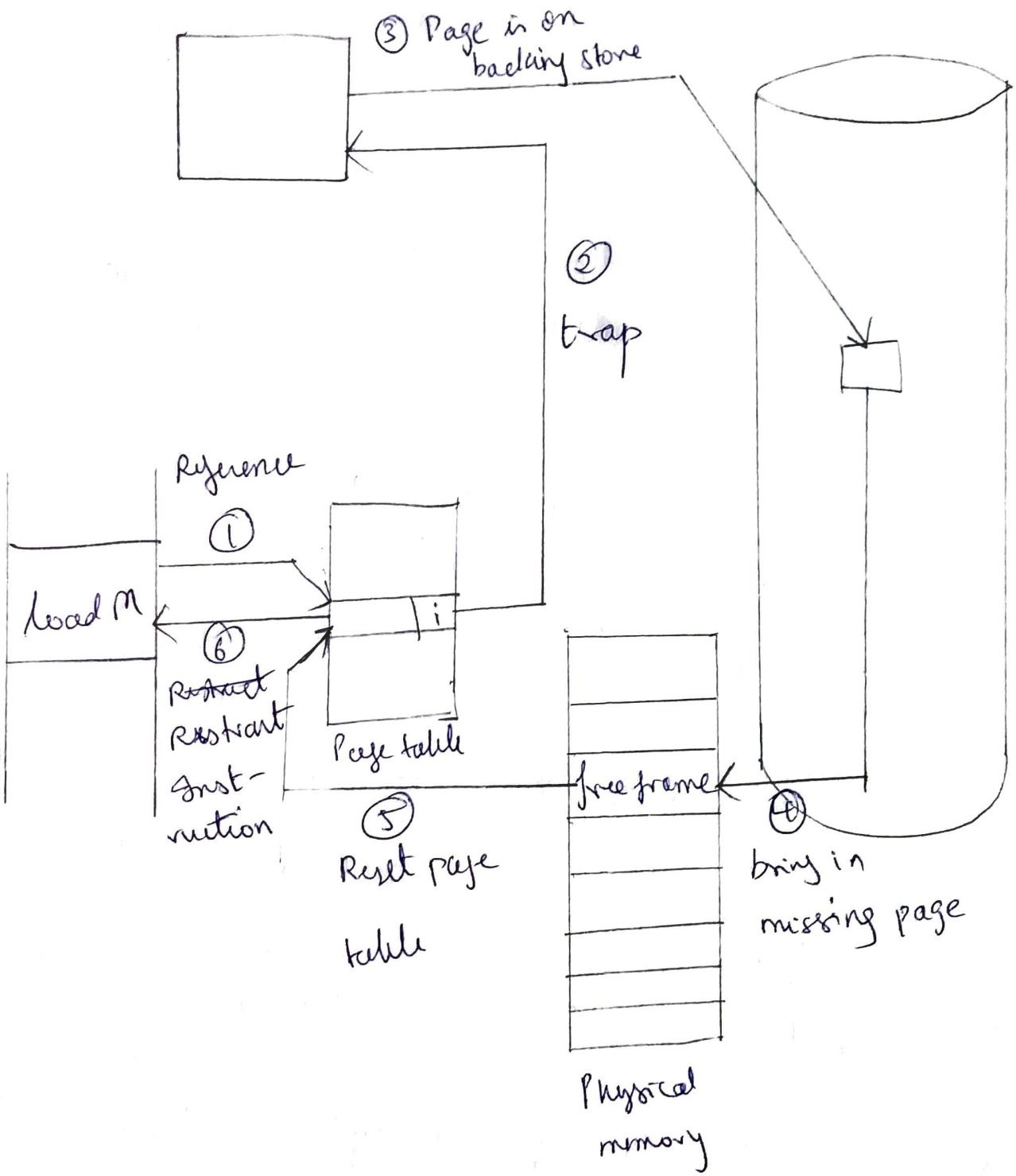
(iv) LFU : Least Freq. used

→ Replace the least freq. used page.

(v) MFU : Most Freq. used

→ Replace the most freq. used page.

Steps in handling a page fault ~~are~~ are:



3911037J

PART-A

(1) These are the procs that can affect / are affected by other processes running on the system

(2) It is a synchronisation object that controls the access by multiple processes to a common resource.

types:

(i) Binary Semaphore

(ii) Counting "

(iii) Mutex "

(3) The request for any resource will be granted if the resulting state of the system doesn't cause deadlock in the system. is called deadlock avoidance

(4) It is a type of interrupt, raised by the H/W when a running proj. accesses a memory page.

3911037 3 (P)

(P)	Internal Fragmentation	Control fragmentation
(i)	But fit Block search is the solution	Compaction is the solution
(ii)	occurs when paging is employed.	occurs when segmentation is employed