# OPERATING SYSTEMS (MC-601)

## ASSIGNMENT-3

NAME : Oindreela Bhowmick

ROLL NO: 2K17/MC/070

## MEMORY MANAGEMENT

Que=1 Consider the following segment table:

| Segment | Base | Length |
|---|---|---|
| 0 | 219 | 600 |
| 1 | 2300 | 14 |
| 2 | 90 | 100 |
| 3 | 1327 | 580 |
| 4 | 1952 | 96 |

What are the physical addresses for the following logical addresses?

A) 0,430

$430 \leq 600$

$\therefore$ physical address = $219 + 430 = 649$

B) 1,10

$10 \leq 14$

$\therefore$ physical address = $2300 + 10 = 2310$

C) 2,500

$500 \nleq 100$

Illegal reference, traps to operating system

D) 3,400

$400 \leq 580$

$\therefore$ physical address = $1327 + 400 = 1727$

E) 4,112

$112 \nleq 96$

Illegal reference, traps to operating system

Que=2 Consider a logical address space of 256 pages with 4KB page size, mapped onto a physical memory of 64 frames.

A) How many bits are required in the logical address?

page size = 4KB = $2^{12}$ bytes

page offset (d) needs 12 bits.

256 pages = $2^8$ pages $\therefore p = 8$ bits.

Logical address = $p + d = 8 + 12 = 20$ bits.

(b) How many bits are required in the physical address?

64 frames = $2^6$ frames $\therefore f = 6$ bits

physical address = $f + d = 6 + 12 = 18$ bits

Que=3 Consider a paging system with the page table stored in memory.

a) If a memory reference takes 50 nanoseconds, how long does a paged memory reference take?

$2 \times 50 = 100$ ns, 50 ns to access the page table plus 50 ns to access the word in memory.

b) If we add TLBs, and 75% of all page-table references are found in TLBs, what is the effective memory reference time? (Assume that finding a page-table entry in the TLBs takes 2 ns if entry is present).

$75\%(50+2) + 25\%(100+2) = 64.5$ ns

Que=4 Given six memory partitions of 300KB, 600KB, 350KB, 200KB, 750KB, and 125KB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 115KB, 500KB, 358KB, 200KB, and 375KB (in order)? Rank the algorithms in terms of how efficiently they use memory.

let $P_1 = 115KB$ $P_2 = 500KB$ $P_3 = 358KB$ $P_4 = 200KB$ $P_5 = 375KB$

and $M_1 = 300KB$ $M_2 = 600KB$ $M_3 = 350KB$ $M_4 = 200KB$ $M_5 = 750KB$ $M_6 = 125KB$

first-fit:

| | | | | | | |
|---|---|---|---|---|---|---|
| $P_1 = 115$ → | $M_1 = 300KB$ | $M_2 = 600$ | $M_3 = 350$ | $M_4 = 200$ | $M_5 = 750$ | $M_6 = 125$ |
| $P_2 = 500$ | $M_1 = 185$ | $M_2 = 600$ | $M_3 = 350$ | $M_4 = 200$ | $M_5 = 750$ | $M_6 = 125$ |
| $P_3 = 358$ | $M_1 = 185$ | $M_2 = 100$ | $M_3 = 350$ | $M_4 = 200$ | $M_5 = 750$ | $M_6 = 125$ |
| $P_4 = 200$ | $M_1 = 185$ | $M_2 = 100$ | $M_3 = 350$ | $M_4 = 200$ | $M_5 = 392$ | $M_6 = 125$ |
| $P_5 = 375$ | $M_1 = 185$ | $M_2 = 100$ | $M_3 = 150$ | $M_4 = 200$ | $M_5 = 392$ | $M_6 = 125$ |

Best fit

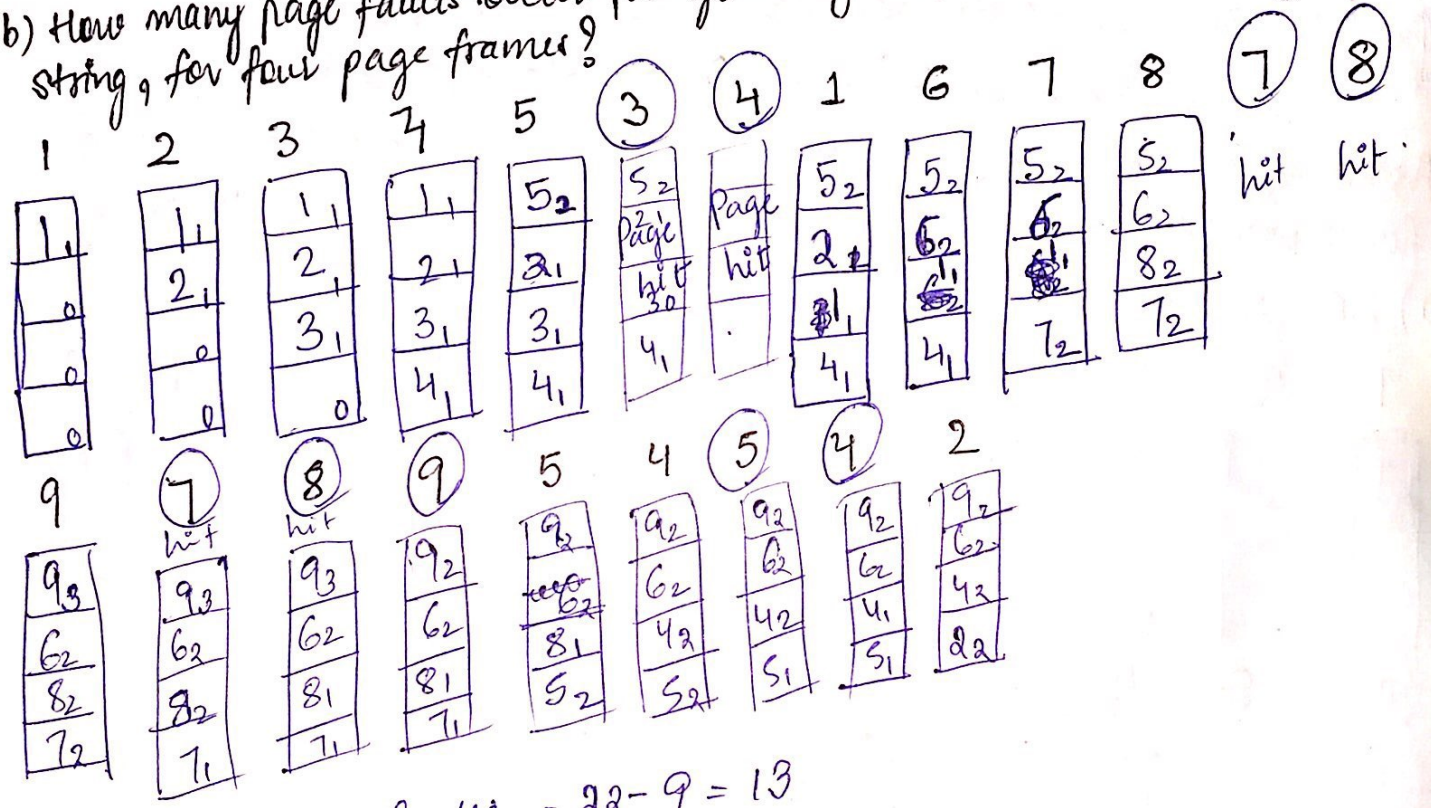
| | | | | | | |
|---|---|---|---|---|---|---|
| $P_1 = 115$ | $M_1 = 300$ | $M_2 = 600$ | $M_3 = 350$ | $M_4 = 200$ | $M_5 = 750$ | $M_6 = 125$ |
| $P_2 = 500$ | $M_1 = 300$ | $M_2 = 600$ | $M_3 = 350$ | $M_4 = 200$ | $M_5 = 750$ | $M_6 = 10$ |
| $P_3 = 358$ | $M_1 = 300$ | $M_2 = 100$ | $M_3 = 350$ | $M_4 = 200$ | $M_5 = 750$ | $M_6 = 10$ |
| $P_4 = 200$ | $M_1 = 300$ | $M_2 = 100$ | $M_3 = 350$ | $M_4 = 200$ | $M_5 = 392$ | $M_6 = 10$ |
| $P_5 = 375$ | $M_1 = 300$ | $M_2 = 100$ | $M_3 = 350$ | $M_4 = 0$ | $M_5 = 392$ | $M_6 = 10$ |

1st-fit:

| | | | | | | |
|---|---|---|---|---|---|---|
| =115 | $M_1 = 300$ | $M_2 = 600$ | $M_3 = 350$ | $M_4 = 200$ | $M_5 = 750$ | $M_6 = 125$ |
| $_2 = 500$ | $M_1 = 300$ | $M_2 = 600$ | $M_3 = 350$ | $M_4 = 200$ | $M_5 = 685$ | $M_6 = 125$ |
| $P_3 = 358$ | $M_1 = 300$ | $M_2 = 600$ | $M_3 = 350$ | $M_4 = 200$ | $M_5 = 135$ | $M_6 = 125$ |
| $P_4 = 200$ | $M_1 = 300$ | $M_2 = 242$ | $M_3 = 350$ | $M_4 = 200$ | $M_5 = 135$ | $M_6 = 125$ |
| $P_5 = 875$ | $M_1 = 300$ | $M_2 = 242$ | $M_3 = 150$ | $M_4 = 200$ | $M_5 = 135$ | $M_6 = 125$ |

Qu: 5(a) 1) Initial value of the counters 0
2) counters are increased whenever a new page is associated with that frame.
3) Counters are decreased whenever one of the pages associated with that frame is no longer required.
4) How the page to be replaced is selected? find a frame with the smallest counter. Use FIFO for breaking ties.

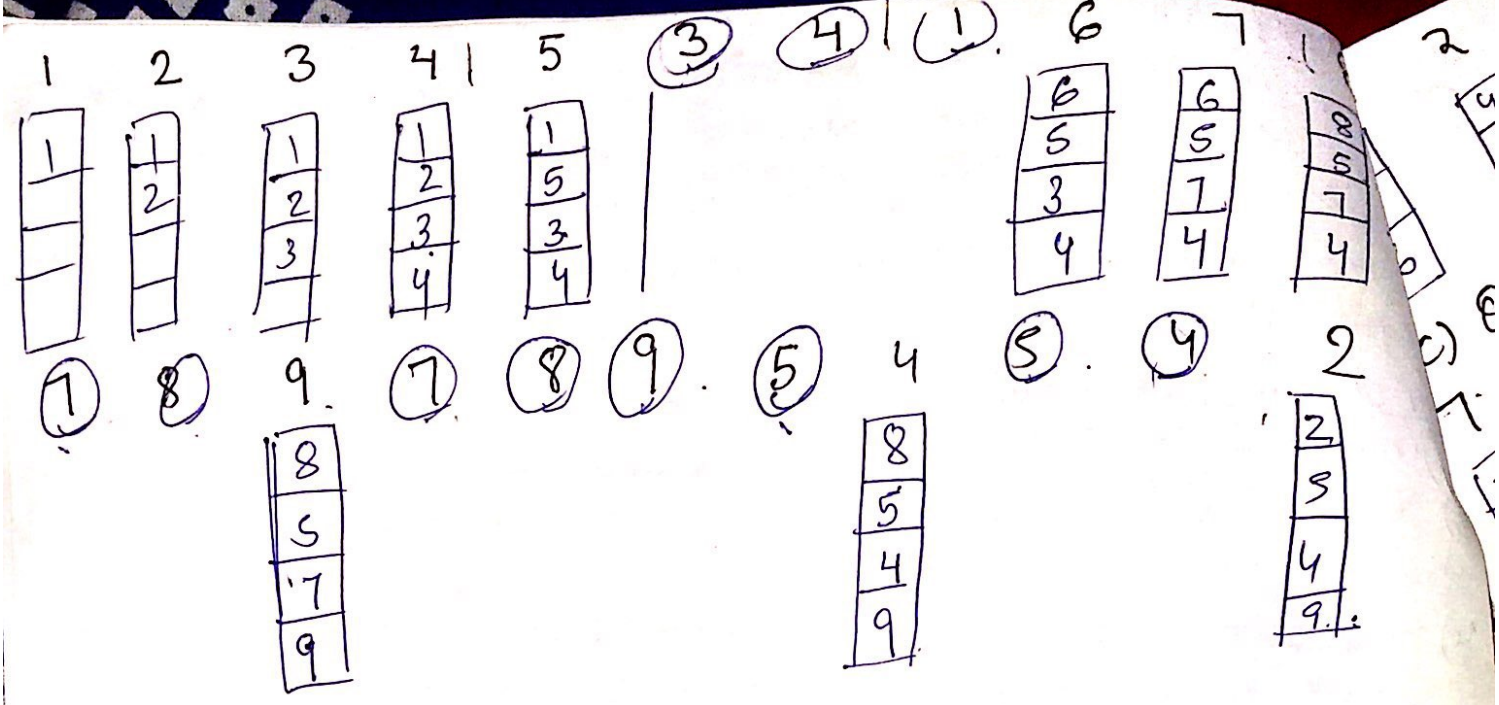
b) How many page faults occur for your algorithm for the following reference string, for four page frames?




No. of page faults = 22 - 9 = 13

c) what is the minimum no. of page faults for an optimal page replacement strategy for the reference string in part b with four page frames?

On next page.



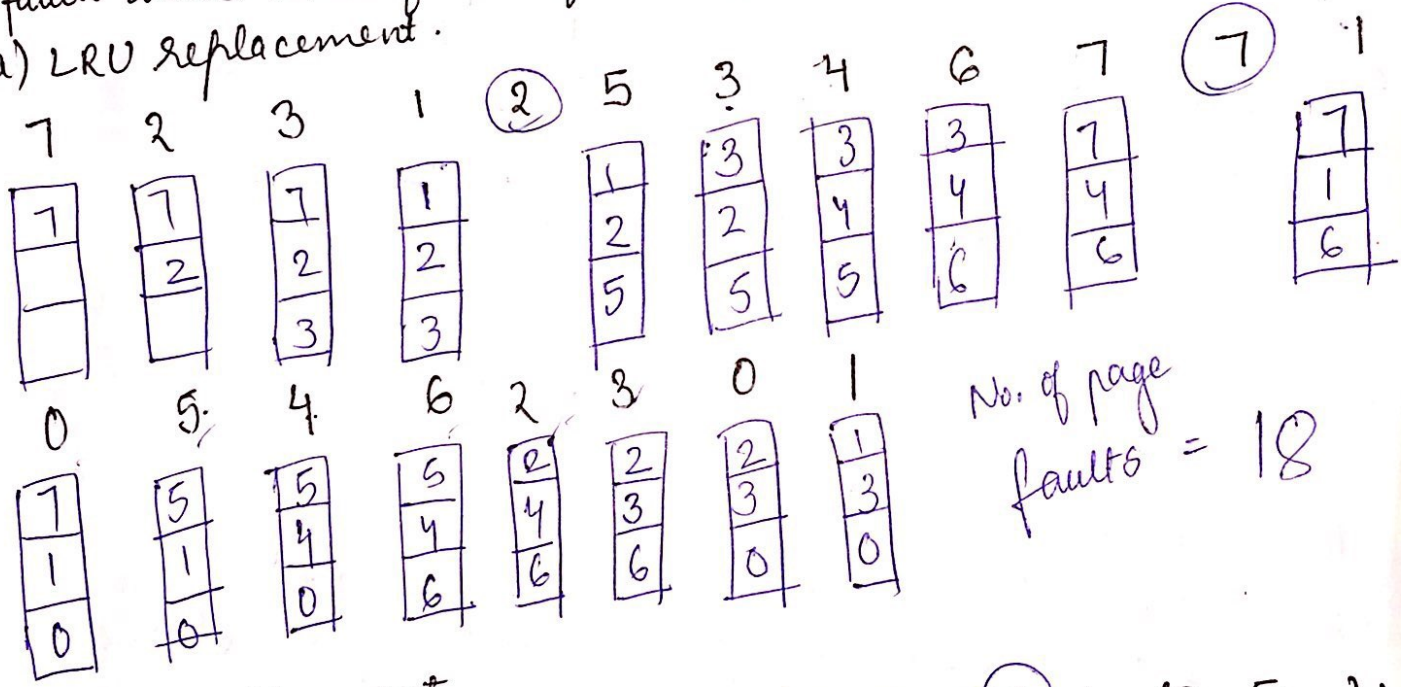

No. of page faults = 22 - 11 = 11

Que-6 Consider the following page reference string:

7, 2, 3, 1, 2, 5, 3, 4, 6, 7, 7, 1, 0, 5, 4, 6, 2, 3, 0, 1

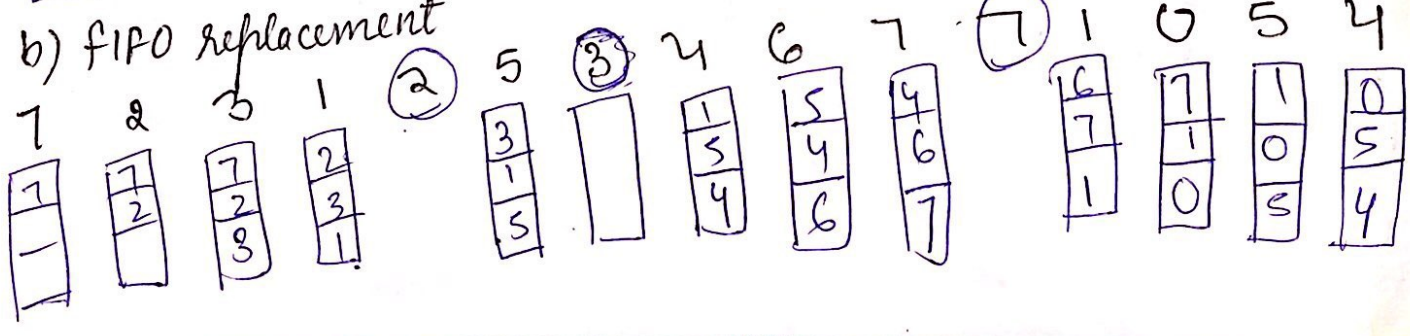
Assuming demand paging with three frames, how many page faults would occur for the following replacement algorithms?

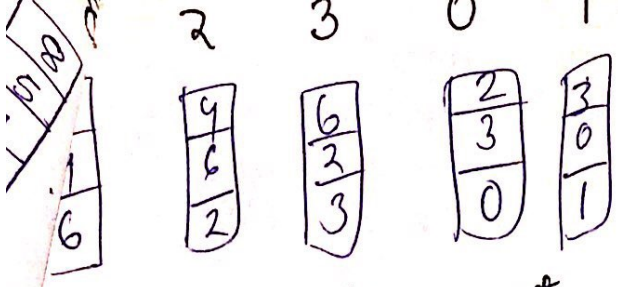
a) LRU replacement.



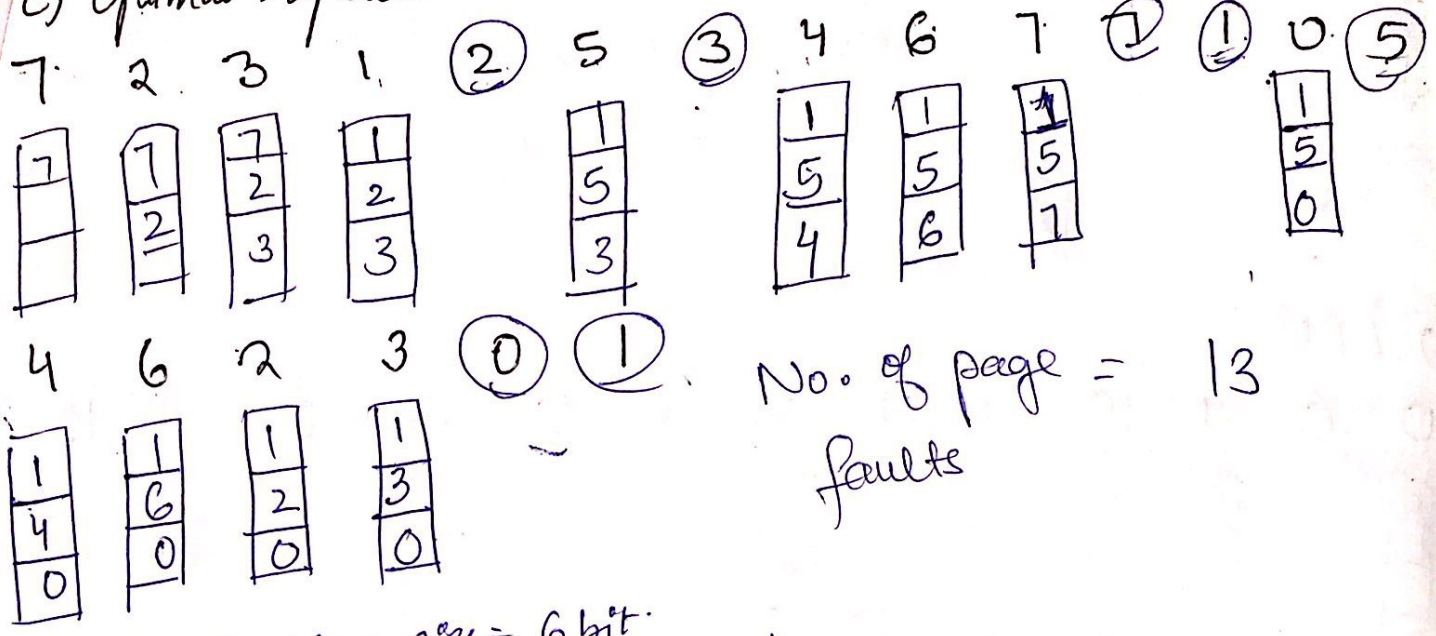

No. of page faults = 18

b) FIFO replacement






c) Optimal replacement







Q1 Virtual address size = 6 bit.
Logical page size = 16B = $2^4$B — 4bits
offset size = 4 bits
page bit = 6 - 4 = 2 bit. ∴ most significant 2bits represent page no.

Page no.s

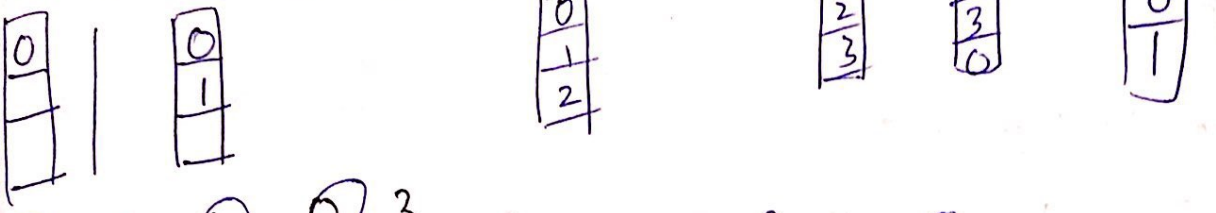
0 → 000000 → 0
1 → 000001 → 0
20 → 010100 → 1
2 → 000010 → 0
20 → 010100 → 1
21 → 010101 → 1
32 → 100000 → 2
31 → 011111 → 1
0 → 000000 → 0
60 → 111100 → 3

0 → 000000 → 0
0 → 000000 → 0
16 → 010000 → 1
1 → 000001 → 0
17 → 010001 → 1
18 → 010010 → 1
32 → 100000 → 2
31 → 011111 → 1
0 → 000000 → 0
61 → 111101 → 3

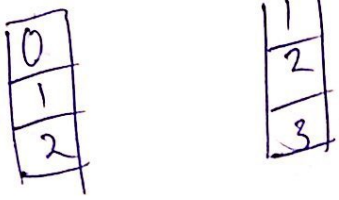
a) Reference string = 0 0 1 0 1 1 2 1 0 3 0 0 1 0 1 1 2 1 0 3

b) fifo

0 (0) 1 (0) (1) (1) 2 (1) (0) 3 0 (0) 1



(1) 2 (1) (0) 3



No. of page faults = 8

c) LRU

0 (0) 1 (0) (1) (1) 2 (1) (0) 3 (0) (0) (1)

(0) (1) (0) 2 (1) (0) 3

No. of page faults = 6

d) Optimal

0 (0) 1 (0) (1) (1) 2 (1) (0) 3 (0) (0) (1) (0) (1) (1)

2 (1) (0) 3

No. of page faults = 6

Q8 a) physical memory size = 8GB = $2^{33}$ B

physical frame size = 4KB = $2^{12}$ B

No. of physical frames = $\frac{2^{33}}{2^{12}} = 2^{21}$

① 1 page table entry has frame no. (21 bits) & flags 10 bits ≈ 4 bytes (31 bits). ~~The total no. of pages per process~~ =

Virtual address space = 4GB = $2^{32}$ B

Page size = 4KB = $2^{12}$ B

no. of pages = $\frac{2^{32}}{2^{12}} = 2^{20}$

Total size of inner page table = $2^{20} \times 4 = 4$ MB

Each page can hold $\frac{2^{12}}{4} = 2^{10}$ page table entries, so we need $\frac{2^{20}}{2^{10}} = 2^{10}$ page table entries to point to inner page tables, which will fit in a single outer page table. So, the total size of page tables of one process is 4MB + 4kB. for 1K processes, total memory consumed by page tables = 4GB + 4MB

Que=9 In a virtual memory system, size of virtual address is 32-bit, size of physical address is 30 bit, page size = 4KB, size of each page table entry = 32 bit. The main memory is byte addressable. Which of the following is the max. no. of bits that can be used for storing protection & other info. in each page table entry?

(A) 2

(B) 10

(C) 12

(D) ✓ 14

Virtual memory = $2^{32}$ bytes

Physical " = $2^{30}$ "

Page size = 4KB = $2^{12}$ B

No. of frames = $\frac{2^{30}}{2^{12}} = 2^{18}$

no. of bits for frame = 18

PTE = 32 bits.

For other info = 32 - 18 = 14 bits

(D) is the correct option.

Que-10 Write a short note on the following:

a) Copy on write is a resource-management technique used in computer programming to efficiently implement a "duplicate" or "copy" operation on modifiable resources. If a resource is duplicated but not modified, it is not necessary to create a new resource; the resource can be shared b/w the copy & the original. Modifications must still create a copy, hence the technique: the copy operation is deferred to the first-write. By sharing resources in this way, it is possible to significantly reduce the resource consumption of unmodified copies, while adding a small overhead to resource-modifying operations.

b) Demand Paging: The process of loading the page into memory on demand (whenever page fault occurs) is known as demand paging. The process includes the following steps:

1) If CPU try to refer a page that is currently not available in main memory, it generates an interrupt indicating memory access fault.
2) The OS puts the interrupted process in blocking state. for the execution to proceed the OS must bring required page into memory.
3) The OS will search for the reqd. page in logical address space.
4) The reqd. page will be brought from logical address space to physical address space. The page replacement algorithms are used for decision making of replacing the page in physical address space.
5) The page table will be updated accordingly.
6) The signal will be sent to the CPU to continue the program execution & it will place the process back into ready state.

Hence, whenever a page fault occurs these steps are followed by the operating system and the required page is brought into the memory.

(b) Virtual memory size = 4GB = $2^{32}$ B @ 32bit.

~~Offset size = $\log_2 t$~~. Page size = 4KB = $2^{12}$ B

Offset size = 12 bits

Page no. bit = 32 - 12 = 20 bit.

Inverted PTE contains page no. + pid

no. of bits = 20 + 10 = 30 bits.

$\approx$ 4 byte

Inverted page table memory size = 4byte x no. of frames.

$= 2^2 \times 2^{21} = 2^{23}$ B

$= 8$ MB

RAM size = 8GB = $2^{33}$ B

no. of frames = $\frac{2^{33}}{2^{12}} = 2^{21}$ (frame size)

Q10 (c) Logical v/s Physical Address Space

| PARAMETER | LOGICAL ADDRESS | PHYSICAL ADDRESS |
|---|---|---|
| Basic | Generated by CPU | ~~location~~ located in a memory unit. |
| Address Space | Logical Address Space is set of all logical addresses generated by CPU in reference to a program | Physical Address Space is set of all physical addresses mapped to the corresponding logical addresses |
| Visibility | User can views the logical address of a program. | User cannot view the physical address of program |
| ~~Generation~~ Access | The user can use the logical address to access the physical address. | The user can indirectly access physical address but not directly. |

# OPERATING SYSTEMS (MCOD)

## ASSIGNMENT-3

NAME: Oindreela Bhowmick

ROLL NO: 2K17/MC/070

## PROCESS SYNCHRONISATION

Que=1

```
husband                          wife
{ withdraw(amount)               { deposit(amount)
    Read balance                   read balance
  { balance = balance              { balance = balance + amount
      - amount                     } write balance
  } write balance.                 }
}                                }
```

Let balance initially be ₹250 which is shared by husband & wife.

Let husband withdraw amount ₹50 & wife deposit ₹100

Suppose husband process executes & the balance is deducted & local balance is ₹200 & then context switch occurs & wife process executes & new balance ₹350 is written, then control returns back to husband & now local balance ₹200 is written. If we do this again similarly but with wife process executing first. We will get a different balance.

This is called race condition

To prevent this, we use Peterson's Algorithm or Semaphores, semaphores can be implemented by using wait(s) at the beginning of withdraw() & deposit() & using signal(s) at the end of withdraw() & deposit() and set s=1. This way, we can prevent 1 process from executing while other is being executed.

Q2

```
f
{ 1) x = x - 50        g
  2) y = y + 50        { 3) a = a + x
}                        4) a = a + y
                       }
```

x = 100  y = 200  a = 0

not possible for a?

~~a)~~ 300

~~b)~~ 250

~~c)~~ 350

d) 200

I: f completes then g completes. (1 2 3 4)

$x = x - 50$ $\quad a = 0 + 50$

$x = 50$ $\quad = 50$.

$y = 250$ $\quad a = 50 + 250 = 300$

(a) is not ~~possible~~. correct.

II: g completes then f completes. (3 4 1 2)

$a = 0 + 100$ $\quad x = 100 - 50$

$= 100$ $\quad = 50$

$a = 100 + 200$ $\quad y = 200 + 50$

$= 300$ $\quad = 250$.

III: 1, 3, 2, 4

$x = 100 - 50$, $a = 0 + 50$, $y = 200 + 50$, $a = 50 + 250$

$= 50$, $= 50$, $= 250$, $= 300$.

IV 1, 3, 4, 2

$x = 50$, $a = 50$, ~~$y = 200$~~ $a = 50 + 200$, $y = 200 + 50$

$= 250$ $\quad = 250$

(b) is not correct

V 3, 1, 2, 4

$a = 0 + 100 = 100$, $x = 50$, $y = 250$, $a = 100 + 250 = 350$.

(c) not correct.

VI 3, 1, 4, 2

$a = 100$, $x = 50$, $a = 100 + 200$, $y = 250$

$= 300$

~~(a) is~~ (d) is the correct option as 200 cannot be generated by any of the combinations.

Qu=3 Each Process $P_i$, $i = 1, 2, \ldots, 9$ is coded as follows:

```
repeat
P(mutex)
{ critical section
} V(mutex)
forever
```

The code for $P_{10}$ is identical except it uses V(mutex) in place of P(mutex). What is the largest number of processes that can be inside the critical section at any moment?

Suppose $P_1$ starts execution then due to mutex ($P_2, \ldots, P_9$) cannot start execution, then in between context switch occurs & $P_{10}$ starts executing. Then due to V(mutex), another process can start executing. This ~~goes on~~ can go on and finally all the 10 processes can enter critical section at the same time.

∴ Max no. of processes in critical section = 10

Q4 Deadlock situation will arise when:

P[0] has acquired m[0], is waiting for m[1]
P[1] " " m[1] " " m[2]
P[2] " " m[2] " " m[3]
P[3] " " m[3] " " m[0]

~~P[4] " " m[4] " " m[0]~~

Q5 Let W = P(S), X = V(T), Y = P(T), Z = V(S)

and S = 1, T = 0

In this way, we ensure that when Process P starts execution. Process Q cannot (bcz T = 0) Thus when Process P completes execution ('0' is printed) T is set to 1 (due to V(T)).
These values ensure alternate execution of Process P & Q.
& string 00110011... is printed.

Q6 This is a producer consumer problem so,

(D) option is corrected.

P(empty) checks the overflow condition that if no cell is empty (empty = 0) then element cannot be added to buffer

V(full) increases no. of full blocks by 1 after it has been added to the buffer.

P(full) checks underflow condition that if all the cells are empty & no cell is full (full = 0) then element cannot be removed from buffer.

V(empty) ~~decreases~~ increases no. of empty blocks by 1 after it has been removed ~~by~~ from the buffer.

Q7

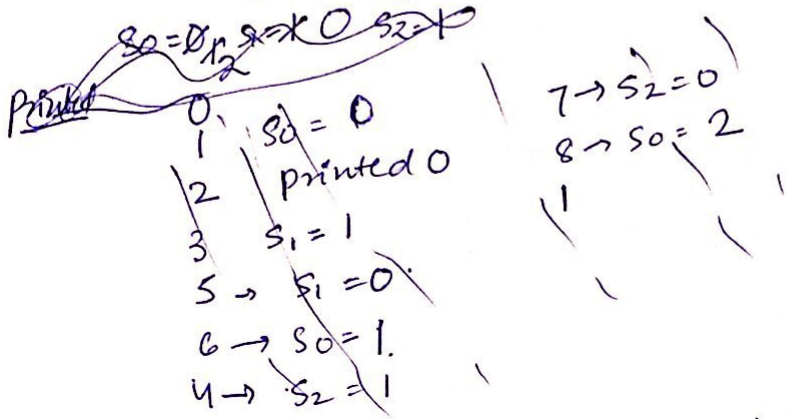
| $P_0$ | $P_1$ | $P_2$ |
|---|---|---|
| while true | 5 wait($S_1$) | 7 wait($S_2$) |
| { 1 wait($S_0$) | 6 release($S_0$) | 8 release($S_0$) |
| 2 print 0 | | |
| 3 release($S_1$) | | |
| 4 release($S_2$) | | |
| } | | |

a.

$S_0 = 1 \quad S_1 = 0 \quad S_2 = 0$.

Case I:

1 2 3 5 6 4 7 8




Initially only $P_0$ can go inside the while loop as $S_0 = 1, S_1 = 0, S_2 = 0$. $P_0$ first prints '0' then, after releasing $S_1$ and $S_2$, either $P_1$ or $P_2$ will execute and release $S_0$. So 0 is printed again.

∴ 0 is printed atleast twice

Q8

```
P1()                    P2
{1 C = B - 1            {3 D = 2 * B
2 B = 2 * C             4 B = D - 1
}                       }
```

B = shared variable
= 2 (initial)

No. of distinct values of B { }

Case I: 1, 2, 3, 4

C = 1, B = 2×1 = 2, D = 2×2 = 4, B = 4 − 1 = 3

B = 3

Case II: 1, 3, 2, 4

C = 1, D = 4, B = 2, B = 3

Case III: 3, 4, 1, 2

D = 4, B = 3, C = 2, B = 4

B = 4

Case IV: 3 1 4 2

D = 4, C = 1, B = 3, B = 2

B = 2

Case V  1 3 4 2

C = 1, D = 4, B = 3, B = 2

Case VI  3 1 2 4

D = 4, C = 1, B = 2, B = 3

∴ 3 distinct values of B are possible.

Q9

| $P_1$ | $P_2$ |
|---|---|
| { 1 Read x | { 4 Read x |
| 2 x = x + 3 | 5 x = x + x + 2 |
| 3 write x | 6 write x |
| } | } |

initial value of x = 1

set of possible values of x {

Case I: 1 2 3 4 5 6

x = 1, x = 4, x = 10 — x = 10

Case II: 4 5 6 1 2 3

x = 1, x = 4, x = 7 — x = 7

Case III: 1 2 4 5 6 3

x = 4, x = 4, — x = 4

Case IV: 1 4 5 6 | 2 3

$P_1$ — x = 1 | $P_2$ x = 4 write | x = 4 —

∴ The set of possible values of x = {4, 7, 10}

Q10 (C) is the correct option.