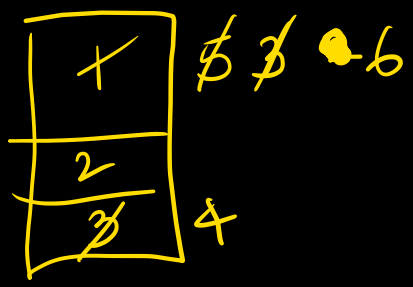
Gate 14)

1 2 3 4 2 1 5 3 2 4 6

PF=? opti, LRU, FIFO

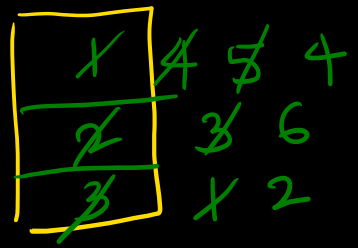
F=3

optimal



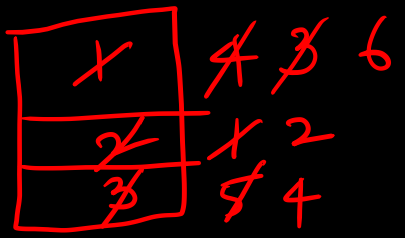
PF = 7

LRU:



PF = 10.

FIFO:



PF=10 1 2 3 4 1 3 2 4 6

Ex: Frames = 3 → PF = x

↓

Frames = 4 → PF = y

$x \geq y$

Frames = 5 → PF = z

$x \geq y \geq z$.

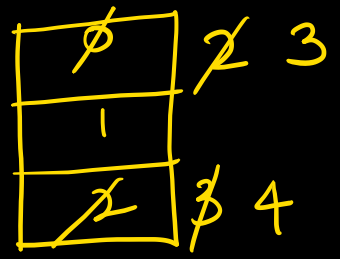
But in FIFO ⇒ for some examples

Frames ↑ PF ↑

Belady's anamoly.

Ex: 0 1 2 3 0 1 4 0 1 2 3 4

✓ Optimal: 3 Frames, 4 Frames

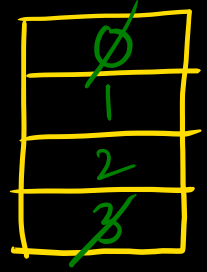


3

4

PF = 7

Frames ↑ PF ↓



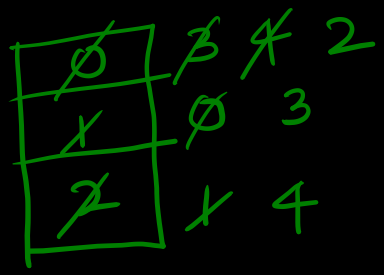
3

4

PF = 6.

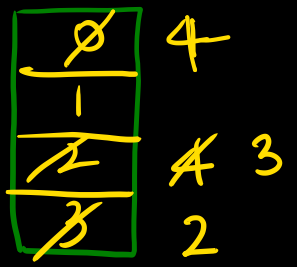
0 1 2 3 0 1 4 0 1 2 3 4

LRU: 3 frames, 4 frames



PF = 10.

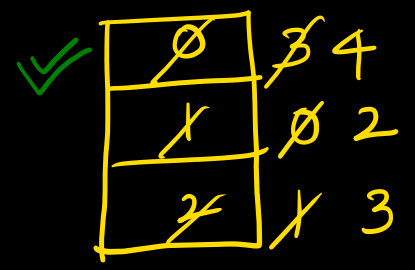
Frames ↑ PF ↓.



PF = 8.

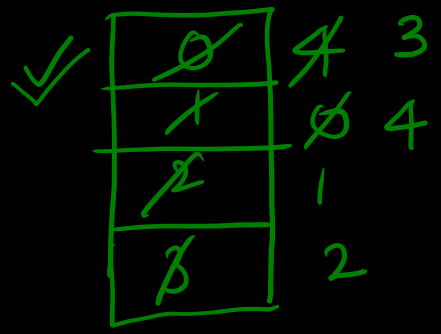
0 1 2 3 0 1 4 0 1 2 3 4

FIFO: 3 frames, 4 frames.



0 1 2 3 4 2 3

PF = 9



0 1 2 3 4 0 1 2 3 4

PF = 10.

FIFO

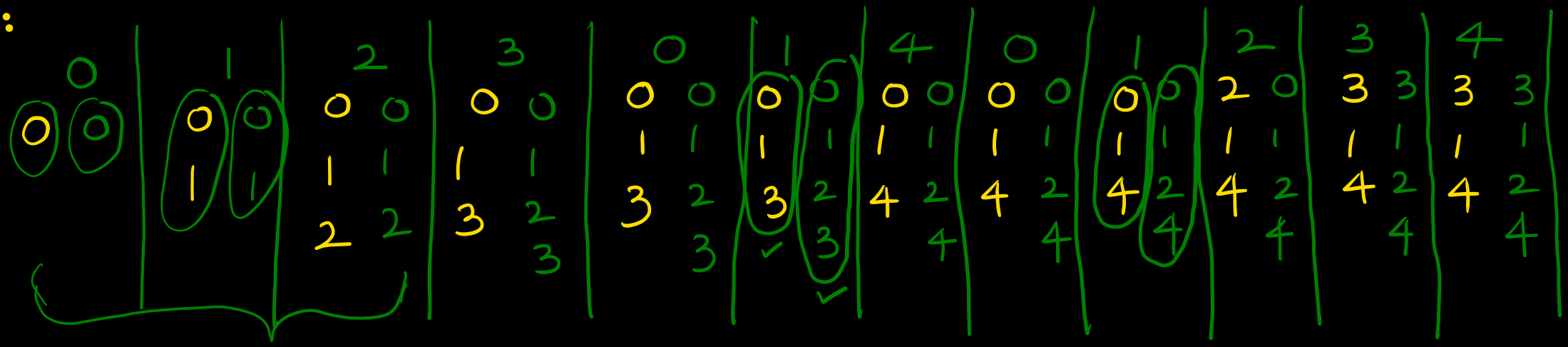
F↑ PF↑

Belady's anamoly.

↓

FIFO is not stack algo.

Optimal:



Pages (3F) ⊆ Pages (4F)

Stack property

Pages (mF) ⊆ Pages ((m+1)F)

Optimal follows stack property.

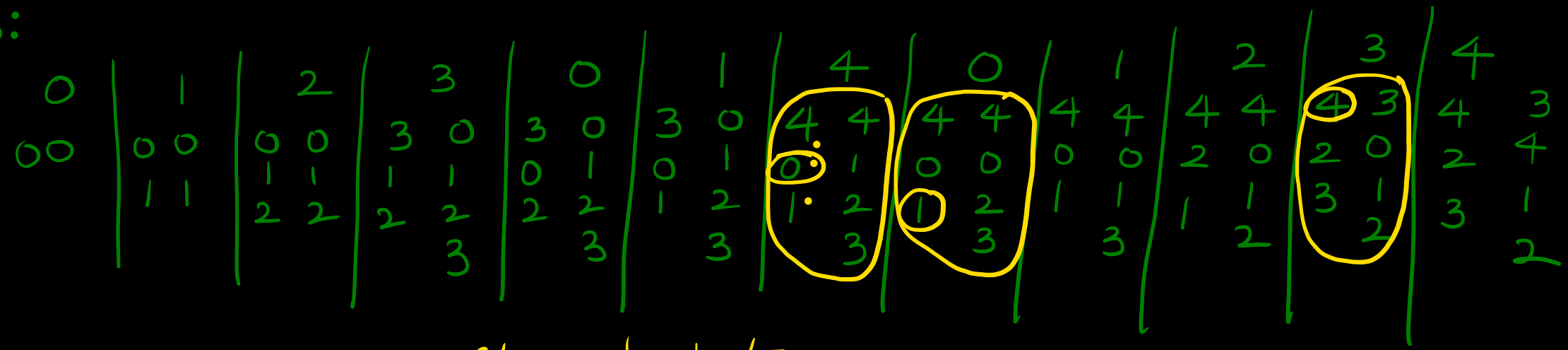
∴ optimal is stack algorithm.

LRU:

0	1	2	3	0	1	4	0	1	2	3	4
00	0 0 1 1	0 0 1 1 2 2	3 0 1 1 2 2 3	3 0 0 1 2 2 3	3 0 0 1 1 2 3	4 0 0 1 1 4 3	4 0 0 1 1 4 3	4 0 0 1 1 4 3	2 0 0 1 1 4 2	2 0 3 1 1 3 2	2 4 3 1 4 3 2

LRU follows stack property → no belady's anamoly.

FIFO:



No stack property ✓

↓

Belady's anamoly.

Gate) A numby page containing a heavily used variable that was initialized very early and is in constant use is removed when ___ is used

a) LRU ✗ b) FIFO ✓ c) LFU ✗ d) None

Gate:) ~~Gate~~ True / False

✓ P: Increasing the num of page frames sometimes increase Page faults in FIFO.

✓ Q: Some programs do not ~~exh~~ exhibit the locality of reference

- locality of reference → future reference will be from Recent pages.
- locality of reference → CO.
- Some programs → Ex: PL X

Gate: A system uses FIFO policy for page replacement it has 4 frames with no pages loaded to begin with. The system first accesses 100 distinct pages in some order then accesses the same 100 pages but now in reverse order. How many page faults will occur

a) 196 b) 192 c) 197 d) 195

0 1 2 3 4 5 6 99

99 98 97 96 95 0

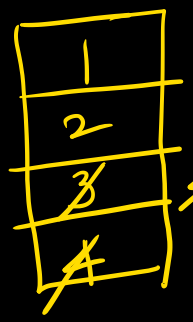
200

0	4	96
1	5	97
2	6	98
3	7	99

PF = 196

Some interesting behaviour of optimal page replacement:

1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6



∴ optimal is behaving like MRU.

optimal → best for all examples.

PF = 12

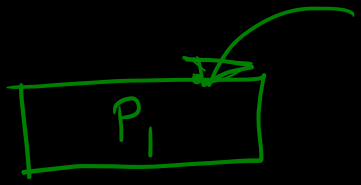
MRU: PF = 12 → for this example, it is best.

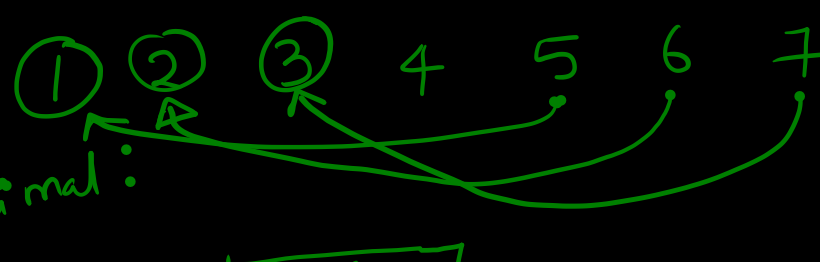
LRU: PF = 24

FIFO: PF = 24

optimal in best fit all examples

but it cannot be practically implemented.





~~1~~	5
~~2~~	2
~~3~~	7
4	

∴ In this example optimal is behaving like LRU.

MRU gives worst PF's.

LRU = FIFO = optimal. ✓

optimal.

LRU ✓ MRU ✓ FIFO

based on examples.

How do we decide, how many frames are to be given to a process? 3F 4F 10F 100F .... ?

working set algorithm:

1 2 3 1 2 4 1 4 2 1 5 1 2 4 2 1

A = 4

$w = \{1\}$ ①

$w = \{1,2\}$ ②

$w = \{1,2,3\}$ ③

$w = \{1,2,3\}$ ③

$w = \{1,2,3\}$ ③

$w = \{1,2,3,4\}$ ④

$w = \{1,2,4\}$ ③

$w = \{1,2,4\}$ ③

$w = \{1,2,4\}$ ③.

$w = \{1,2,4\}$ ③.

$w = \{1,2,4,5\}$ ④.

$w = \{1,5,2\}$ ③.

$w = \{1,5,2\}$ ③.

$w = \{1,2,4,5\}$ ④.

$w = \{1,4,2\}$ ③.

$w = \{1,4,2\}$ ③.

$\frac{48}{16} = 3$

3 frames are required.

Dynamically Frames.

No gate questions

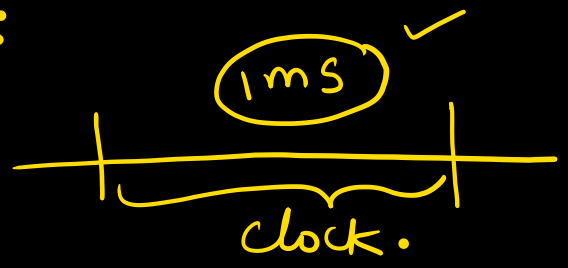
never asked

But they may ask.

Implementations of Page replacement algorithms:

optimal → Cannot be implemented → Predict the future.

Least Recently used:



Reference bit

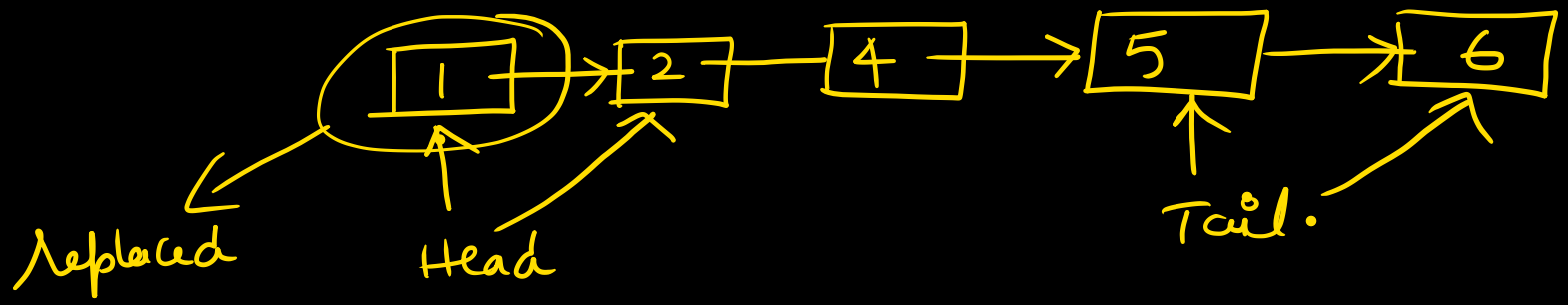
2	0
3	0

→ one of them Replaced

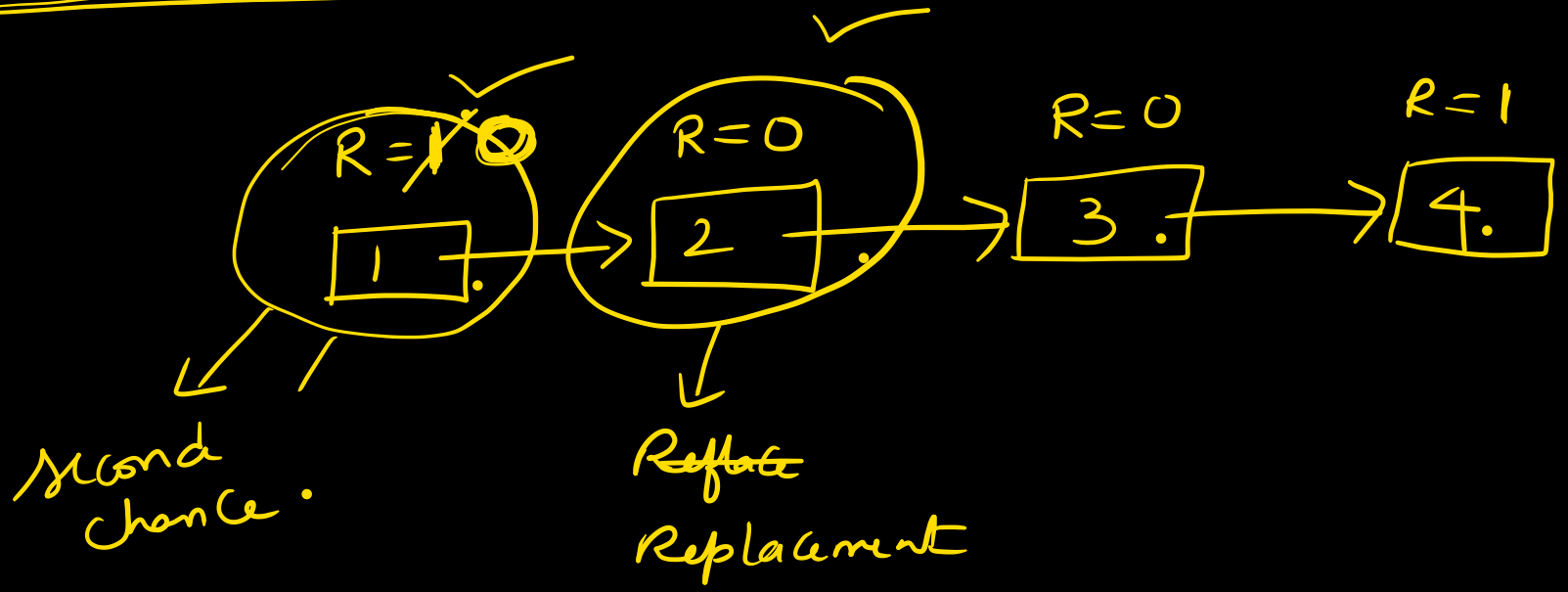
1 1

4 1

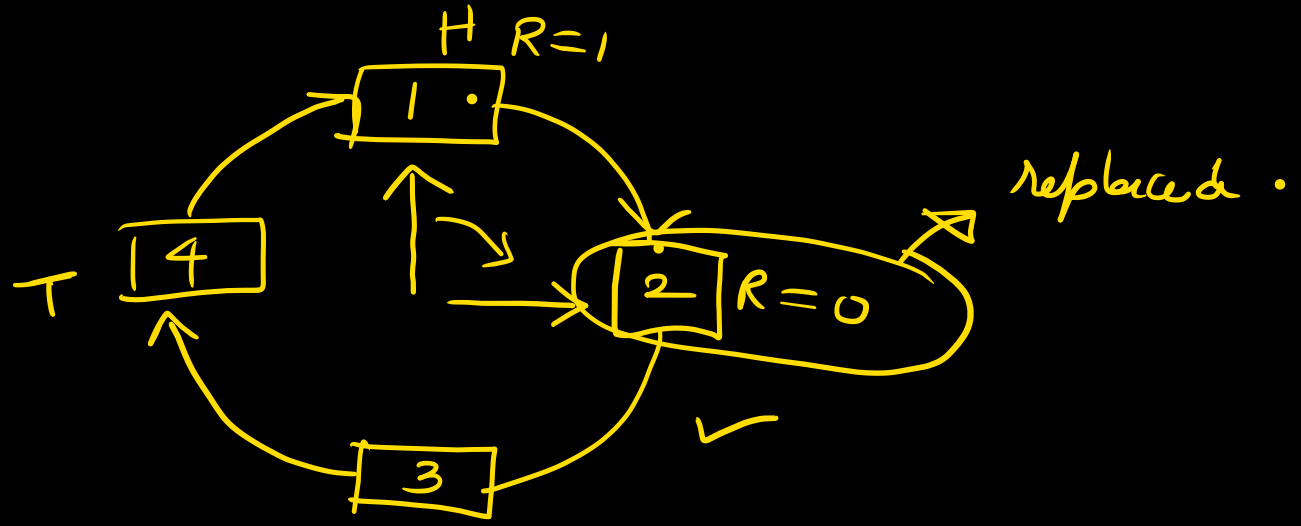
FIFO: maintain a queue (Linked ~~broke~~ list)



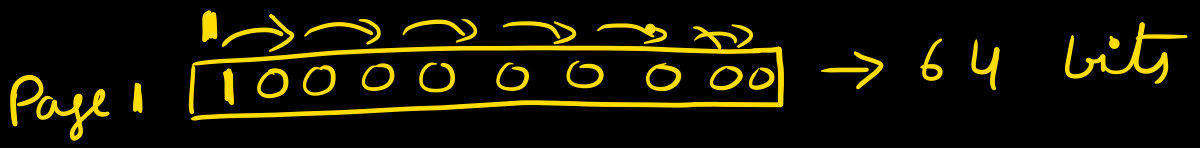
Second chance in FIFO:



Clock page replacement :



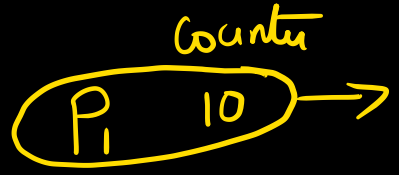
LRU :



1 0 1 0 1 0 0 1 1 → highest value means most recently used

Least value → LRU → Replacement.

LFU :



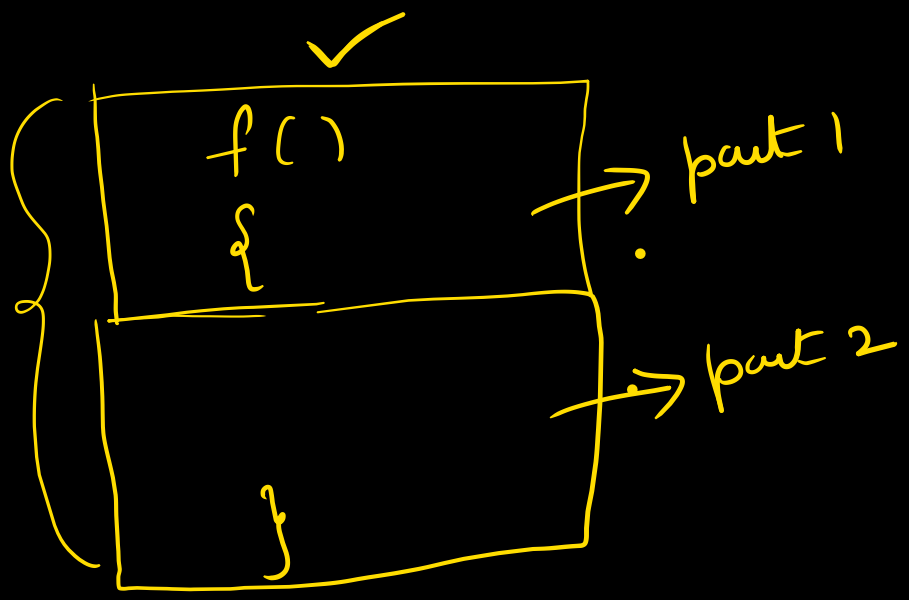
→ replaced.

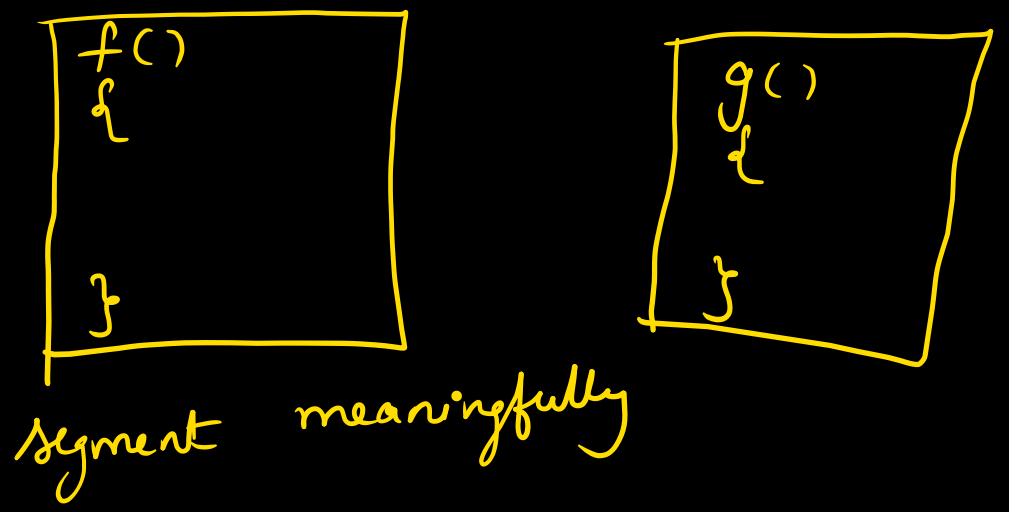
galvin, Tanenbaum, Stallings, Ohamdhure.

↓ memory mangt

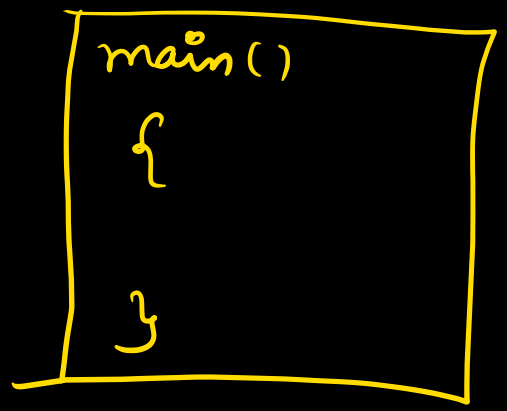
↓ Synchronization

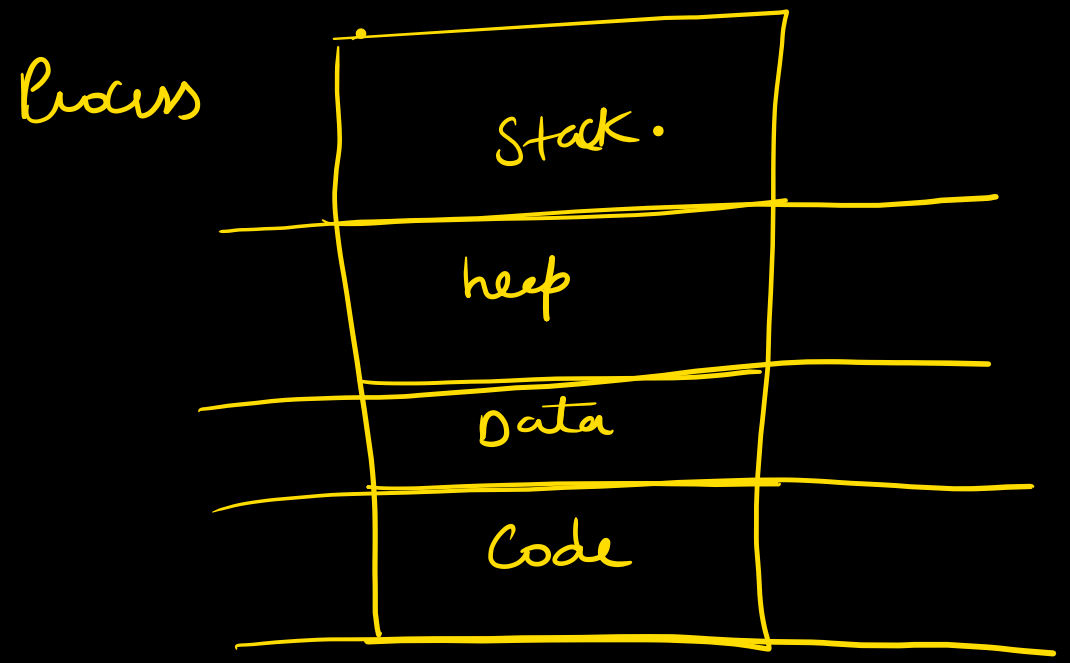
Segmentation:

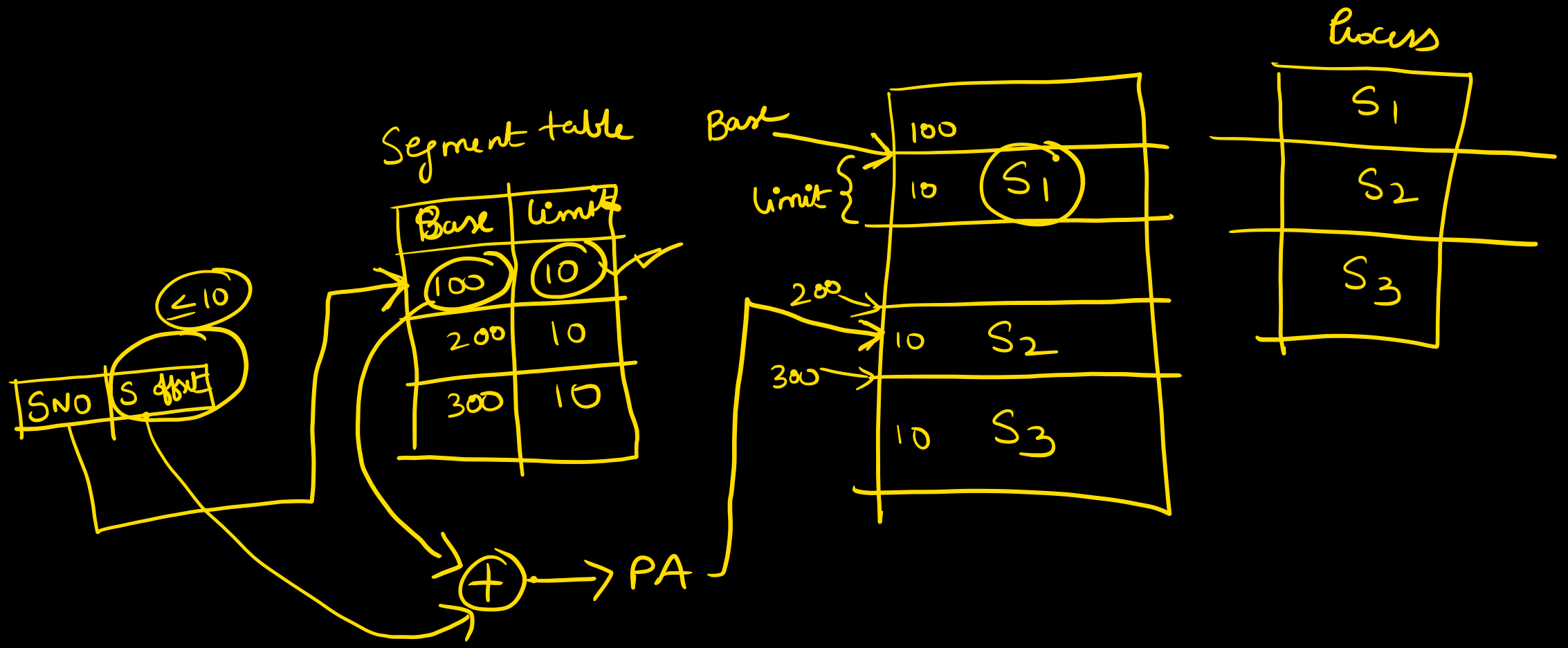




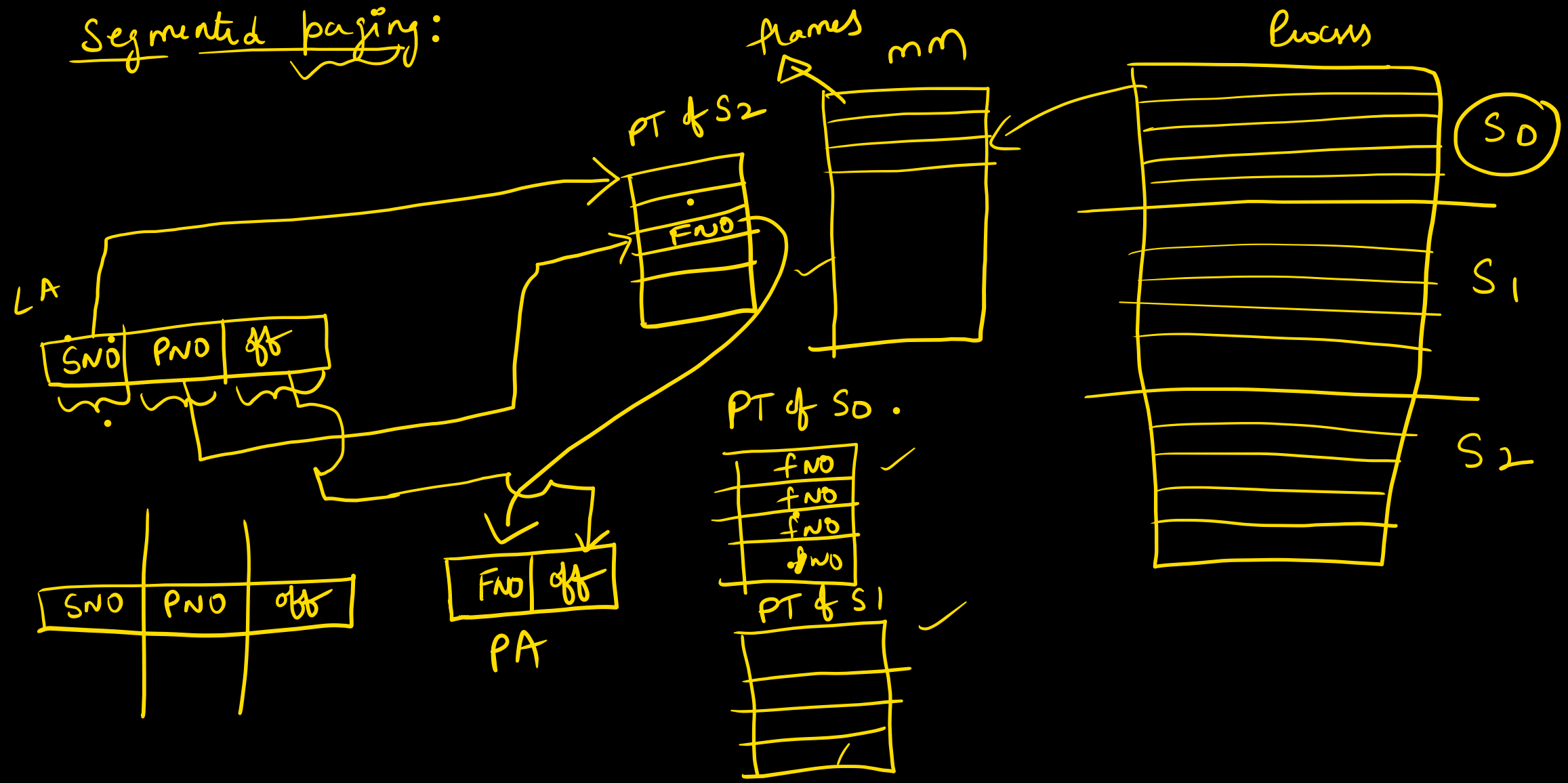
segment meaningfully




Process
Stack.
heap
Data
Code

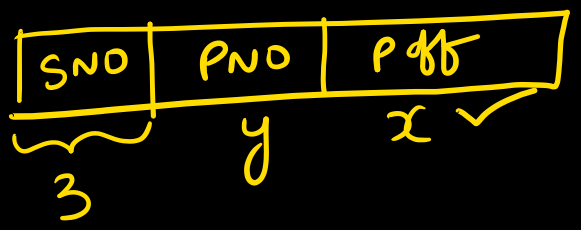

Process
S1
S2
S3
Segment table
Base
Limit
100
10
200
10
300
10
≤10
SNO
S
Offset
Base
Limit
100
10
S1
200
10
S2
300
10
S3
+
PA

Segmented paging:



Gate) PAS = VAS = 2^{16}. Byte addressable. VAS is divided into 8 equal sized segments. The MMU has a segment table, each entry of which contains PA of PT for the segment. PT are stored in MM and consists of 2 B PTE. What is the minimum Page size in Bytes so that Page table for a segment requires atmost one page to store it.

Sol:



$3 + x + y = 16$

$x + y = 13$.

PTS = No of entries x PTE

$2^x = 2^y \times 2$

$x = y + 1$

$x = 7$

$y = 6$

Page size = 2^7 = 128