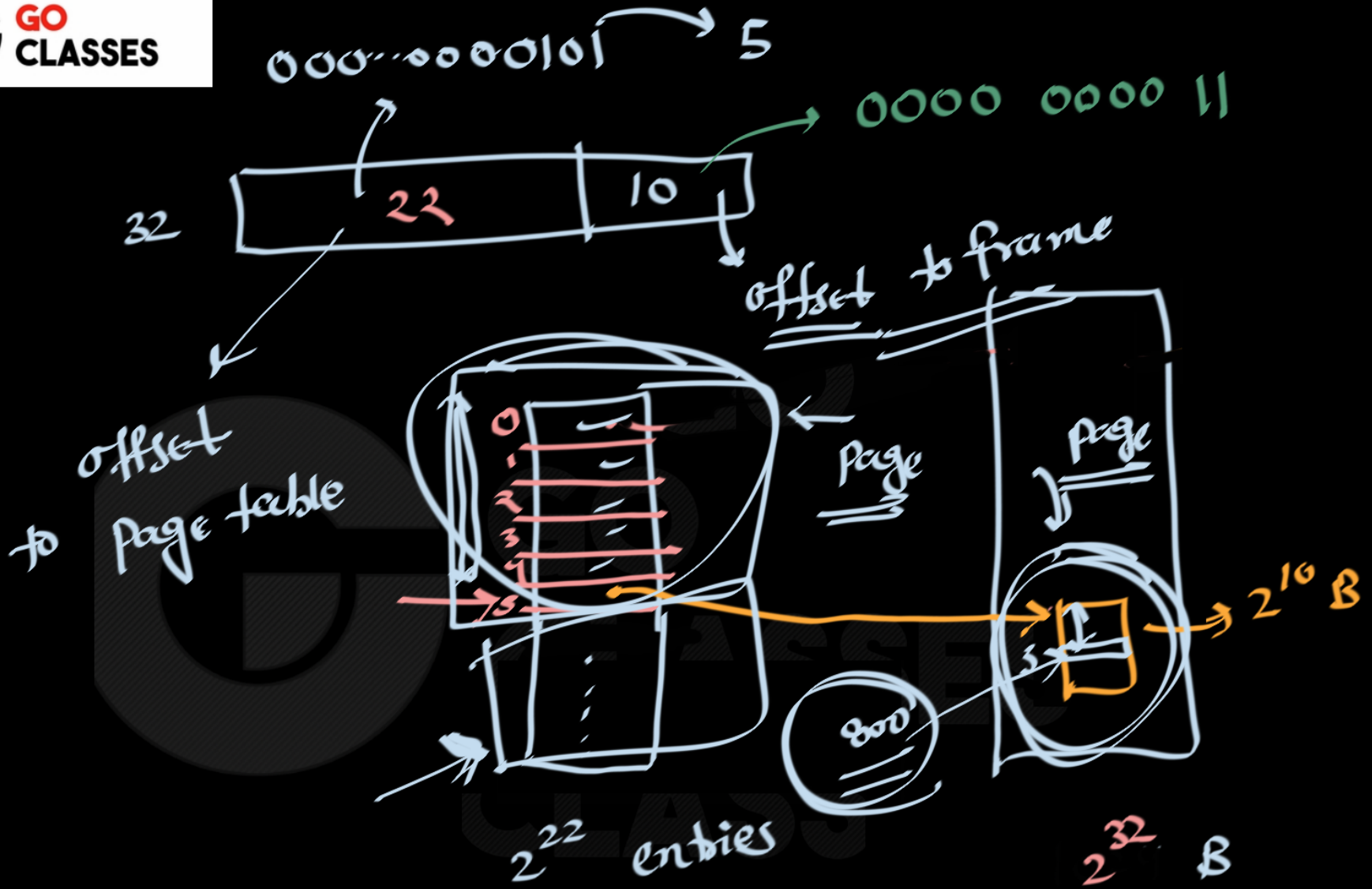
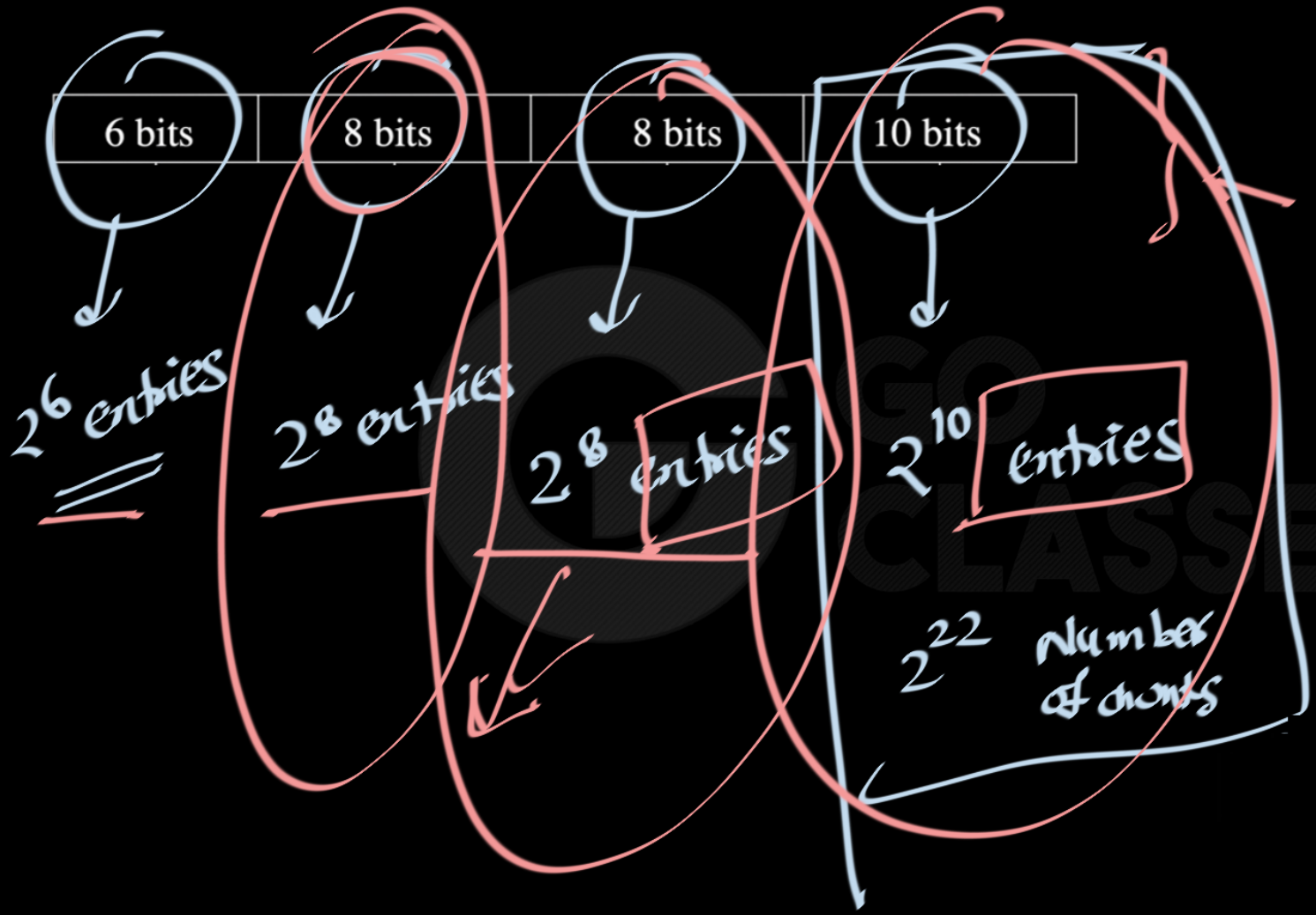
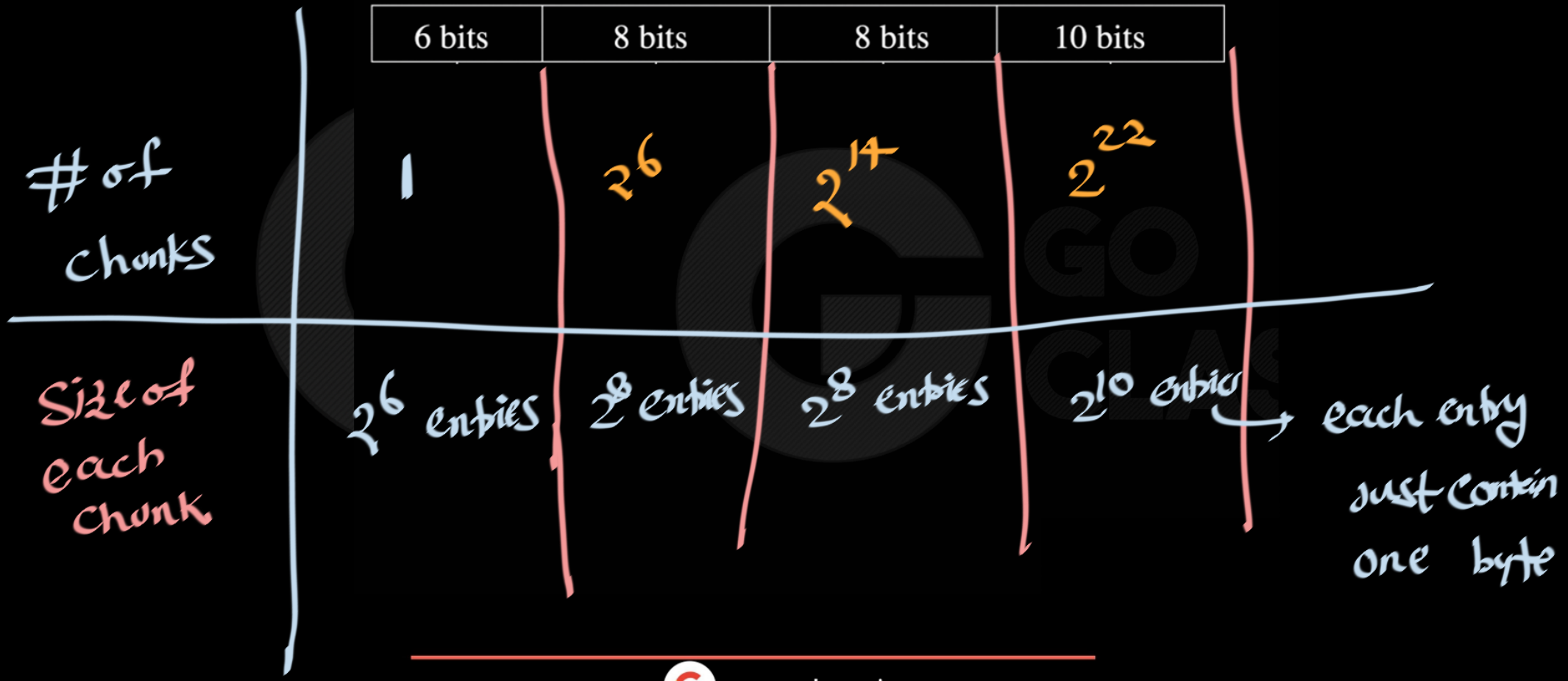
# Lecture: 27

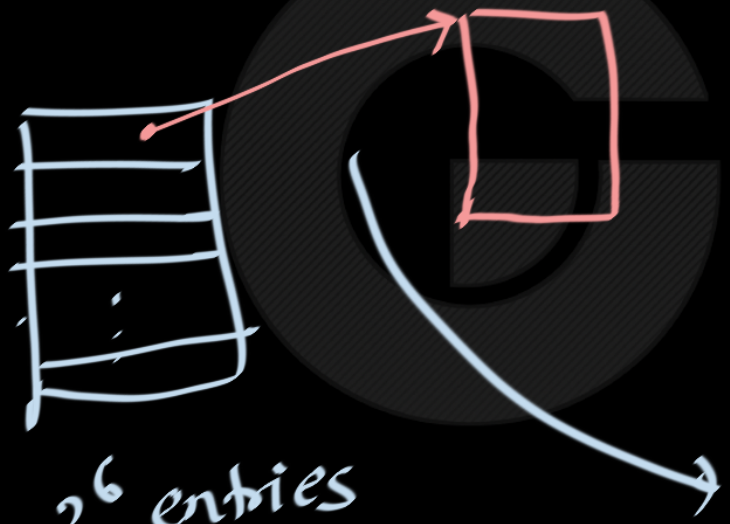

GO CLASSES
000...00000101
5
0000 0000 11
32
22
10
offset to frame
offset
to Page table
Page
Page
0
1
2
3
4
15
800
$2^{16}$ B
$2^{22}$ entries
$2^{32}$ B




address Split in multilevel page table.


6 bits
8 bits
8 bits
10 bits
# of Chunks
1
$2^6$
$2^{14}$
$2^{22}$
Size of each Chunk
$2^6$ entries
$2^8$ entries
$2^8$ entries
$2^{10}$ entries
each entry just contain one byte

| 6 bits | 8 bits | 8 bits | 10 bits |
|---|---|---|---|



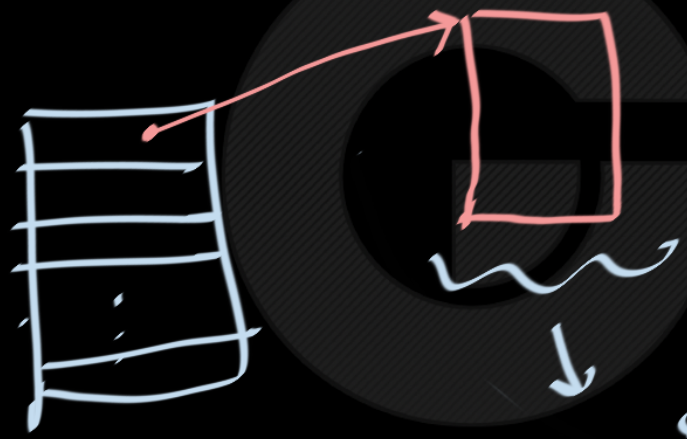
One Entry Points to one chunk of next level

$2^6$ entries

One Chunk

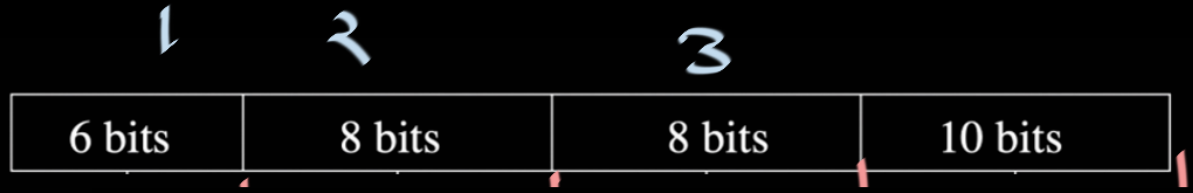
Points to Some Chunk of next level

| 6 bits | 8 bits | 8 bits | 10 bits |
|---|---|---|---|



$2^6$ entries
One Chonk

$2^8$ entries
is Size of Chonk




$2^8$

$2^6$ entries

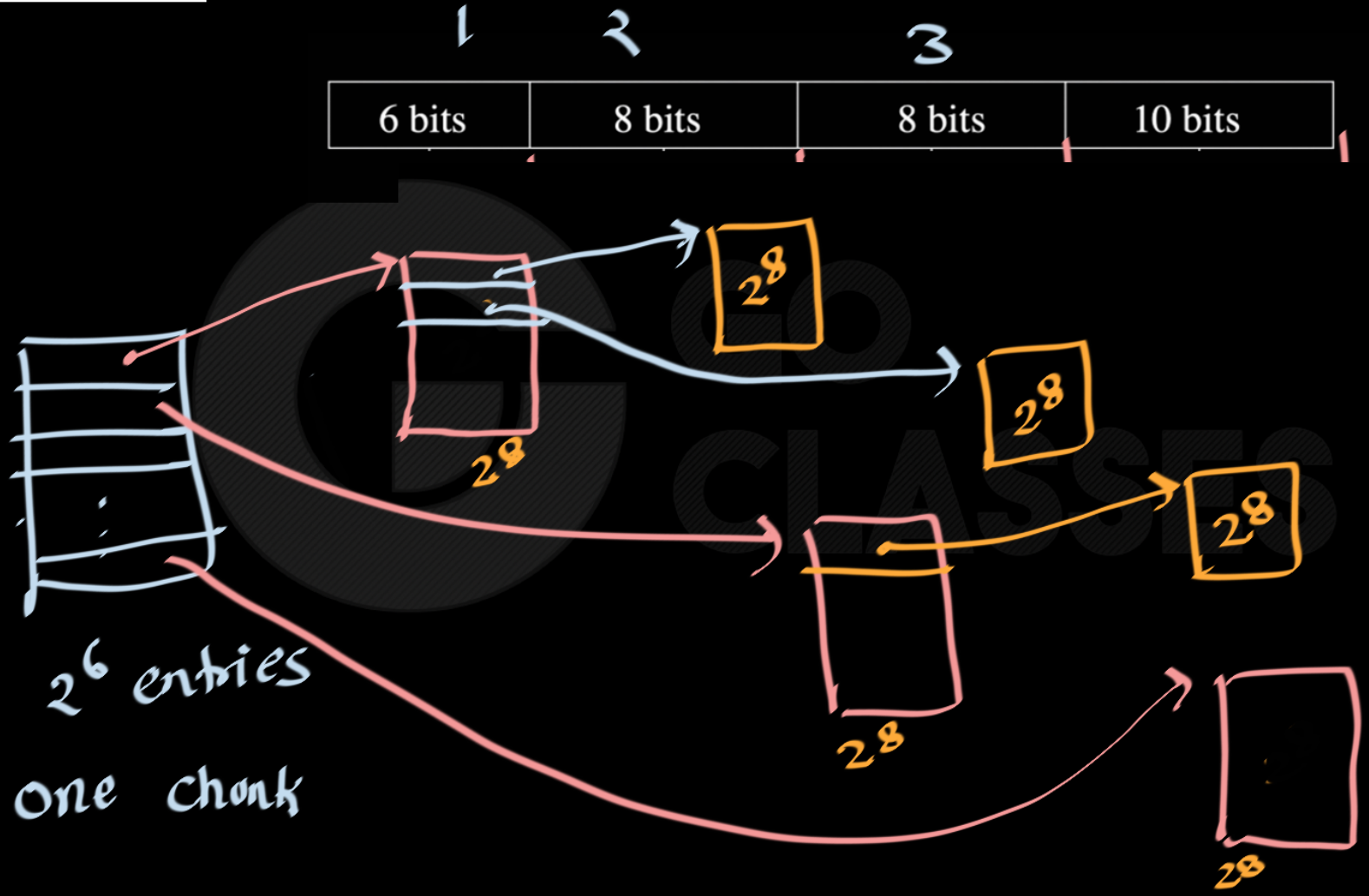
one chunk

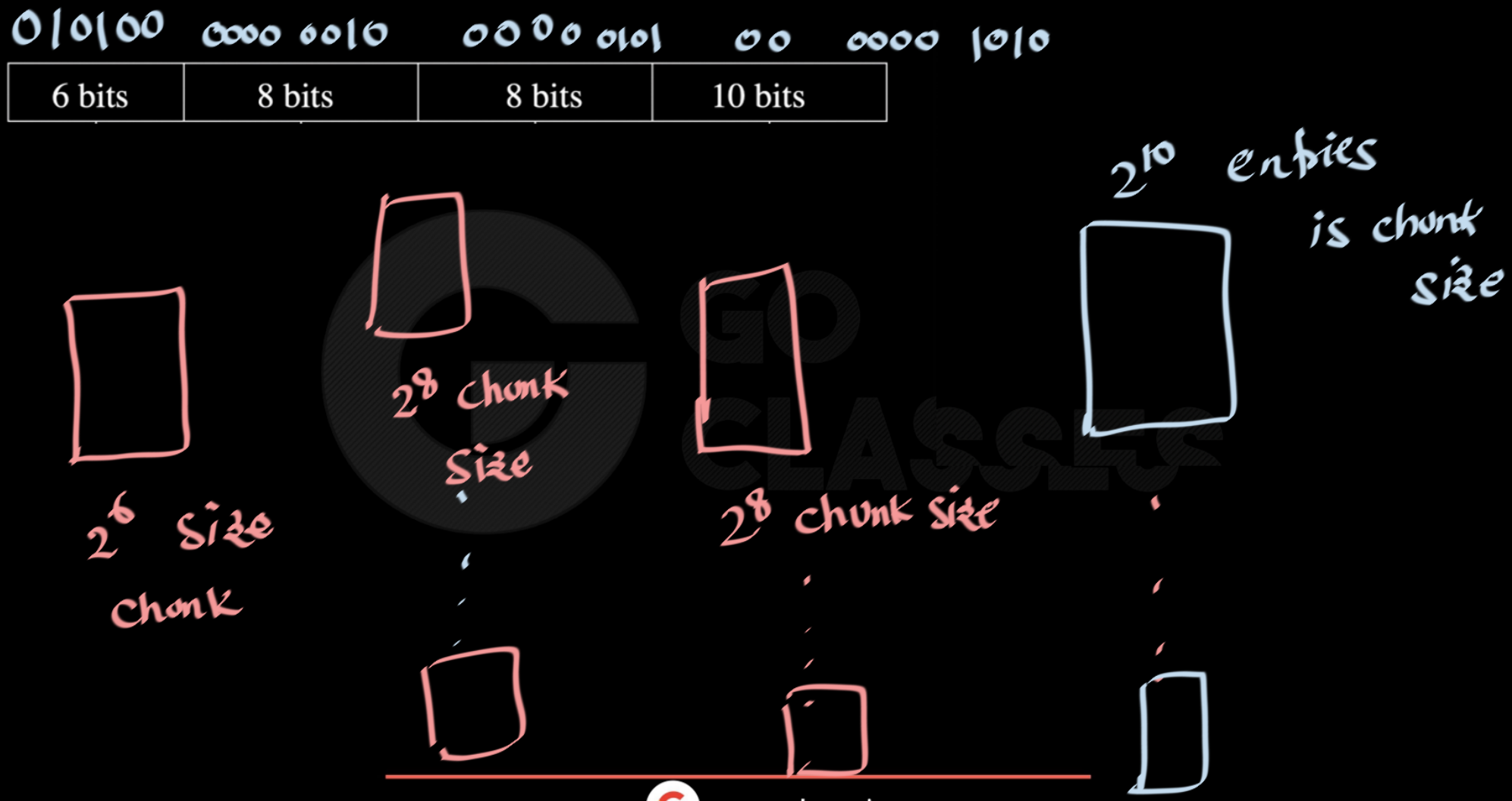
$2^6$ chunks in level 2

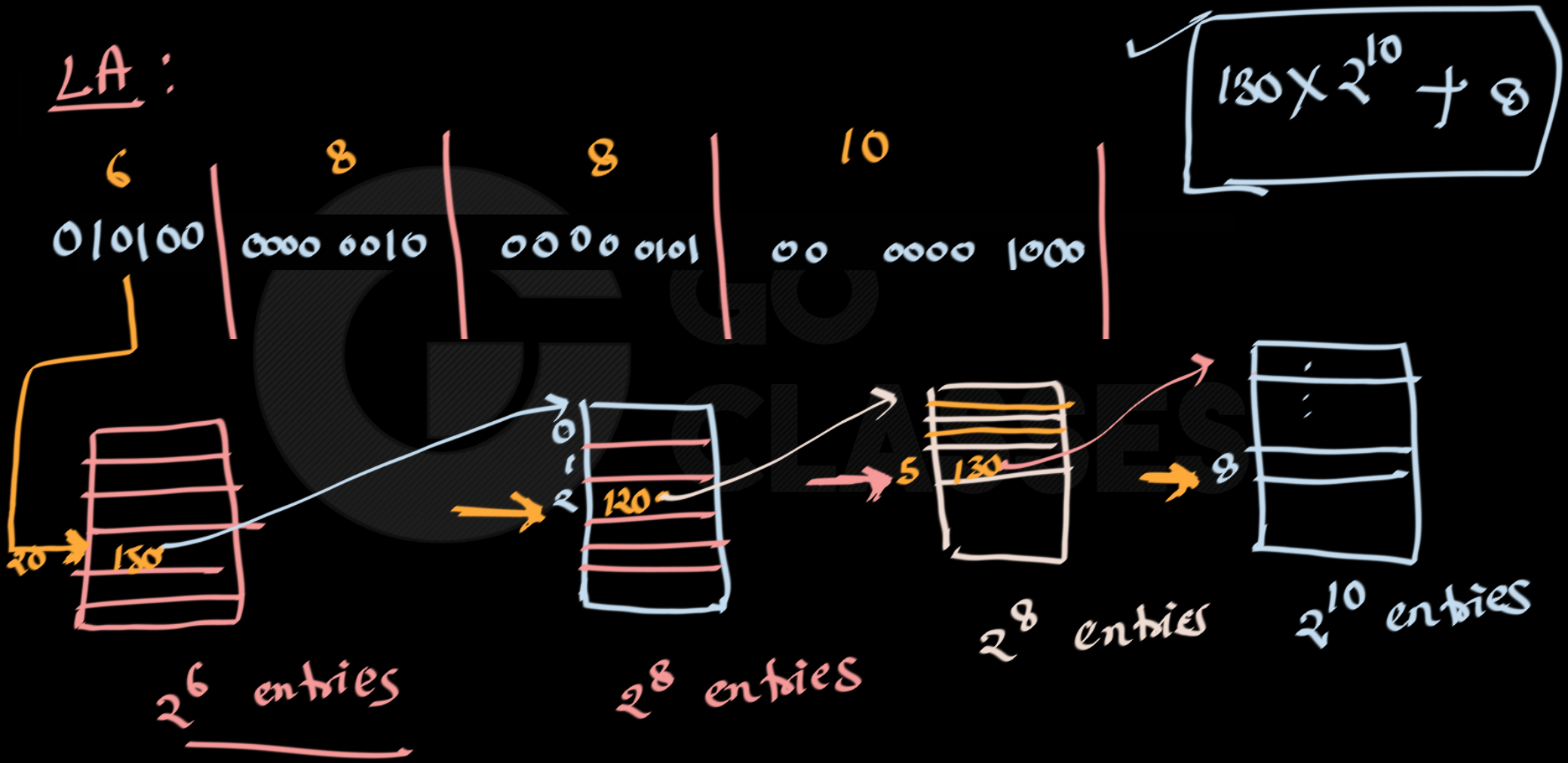
No. of entries = $2^8 \times 2^6$

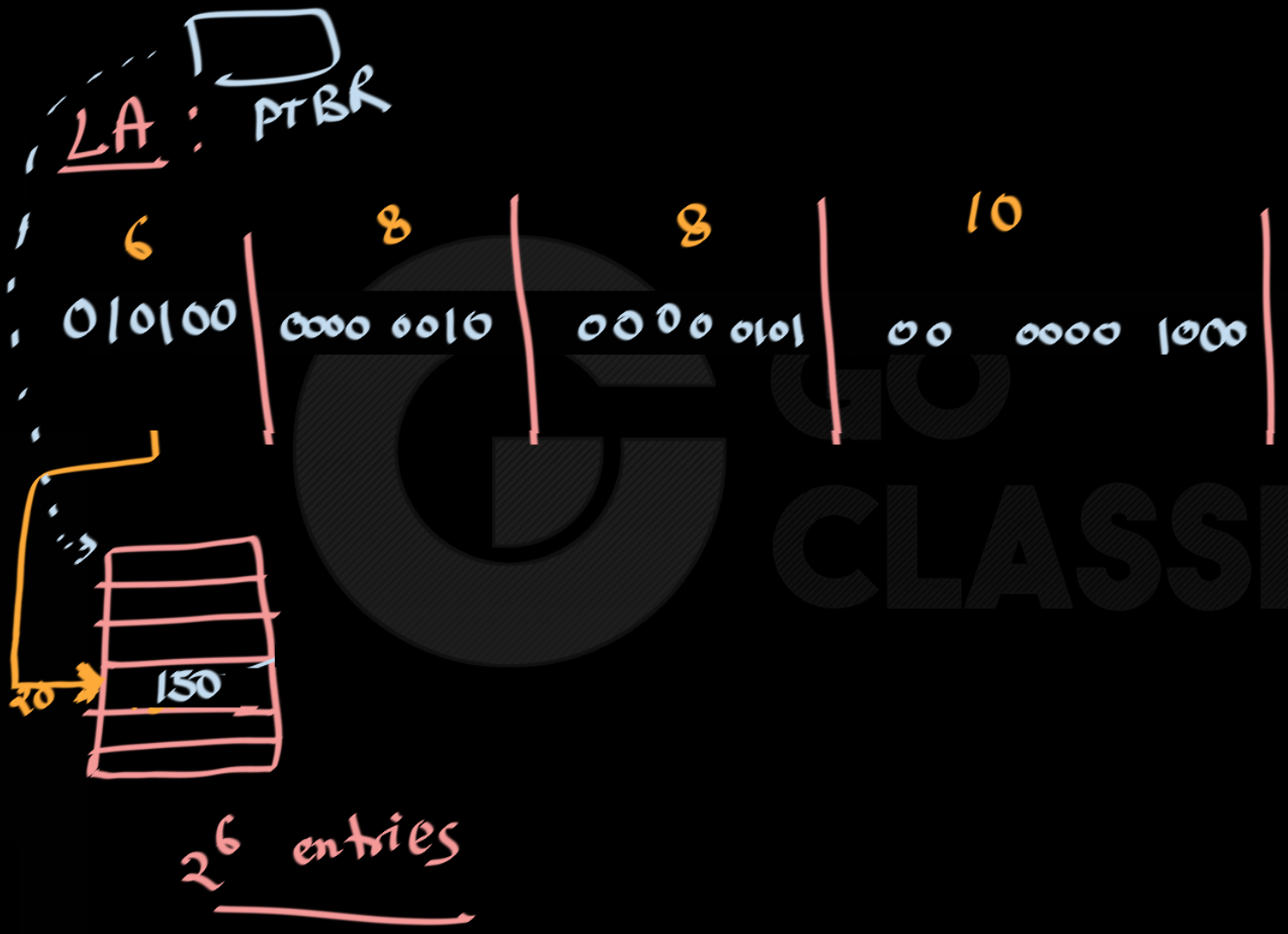
$= 2^{14}$ entry

$2^8$


1
2
3
6 bits
8 bits
8 bits
10 bits
$2^8$
$2^8$
$2^8$
$2^8$
$2^8$
$2^8$
$2^6$ entries
One Chonk

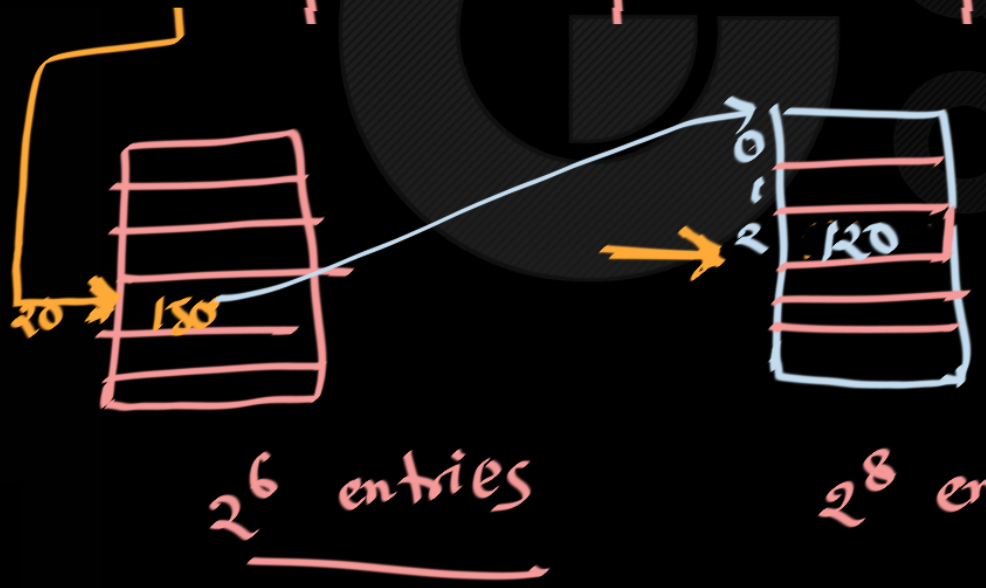

010100 0000 0010 0000 0101 00 0000 1010
6 bits
8 bits
8 bits
10 bits
2^10 entries is chunk size
2^8 chunk Size
2^6 Size Chunk
2^8 chunk size


LA :
6
8
8
10
0 1 0 1 0 0
0000 0010
0000 0101
00 0000 1000
$130 \times 2^{10} + 8$
20
130
0
1
2
120
5
130
8
$2^6$ entries
$2^8$ entries
$2^8$ entries
$2^{10}$ entries

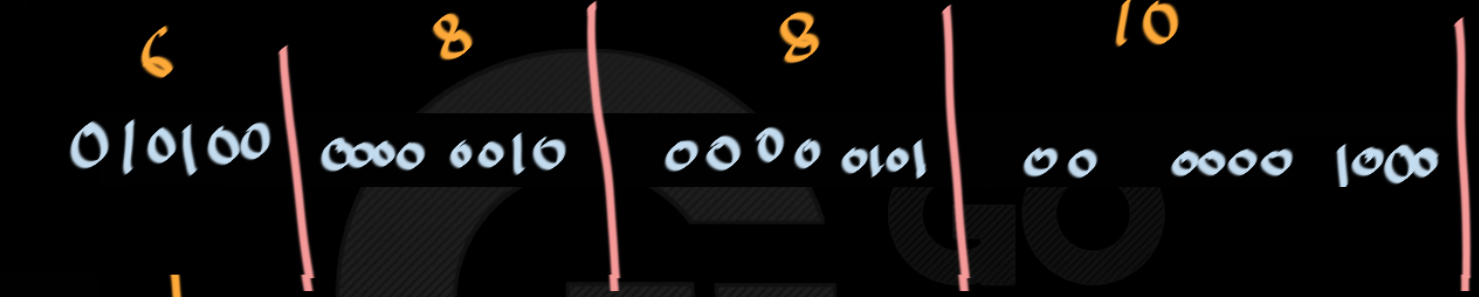

LA :
PTBR
6
8
8
10
010100
0000 0010
0000 0101
00 0000 1000
20
150
$2^6$ entries

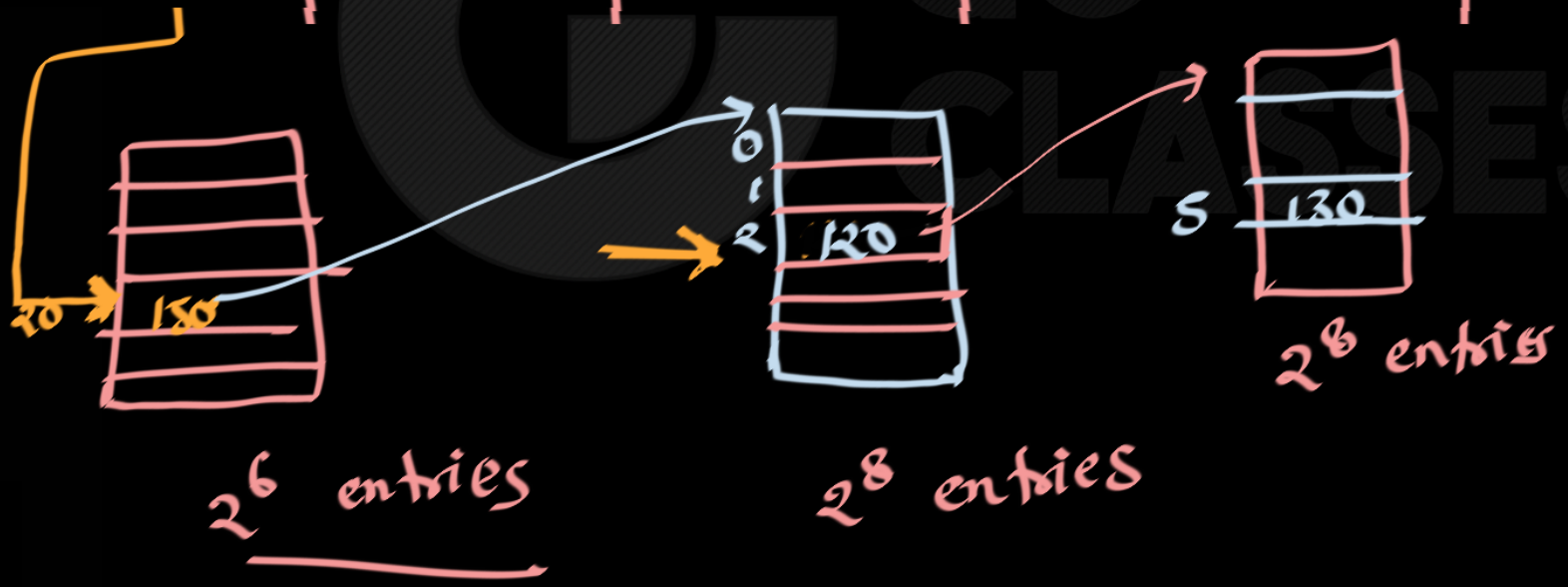
LA :

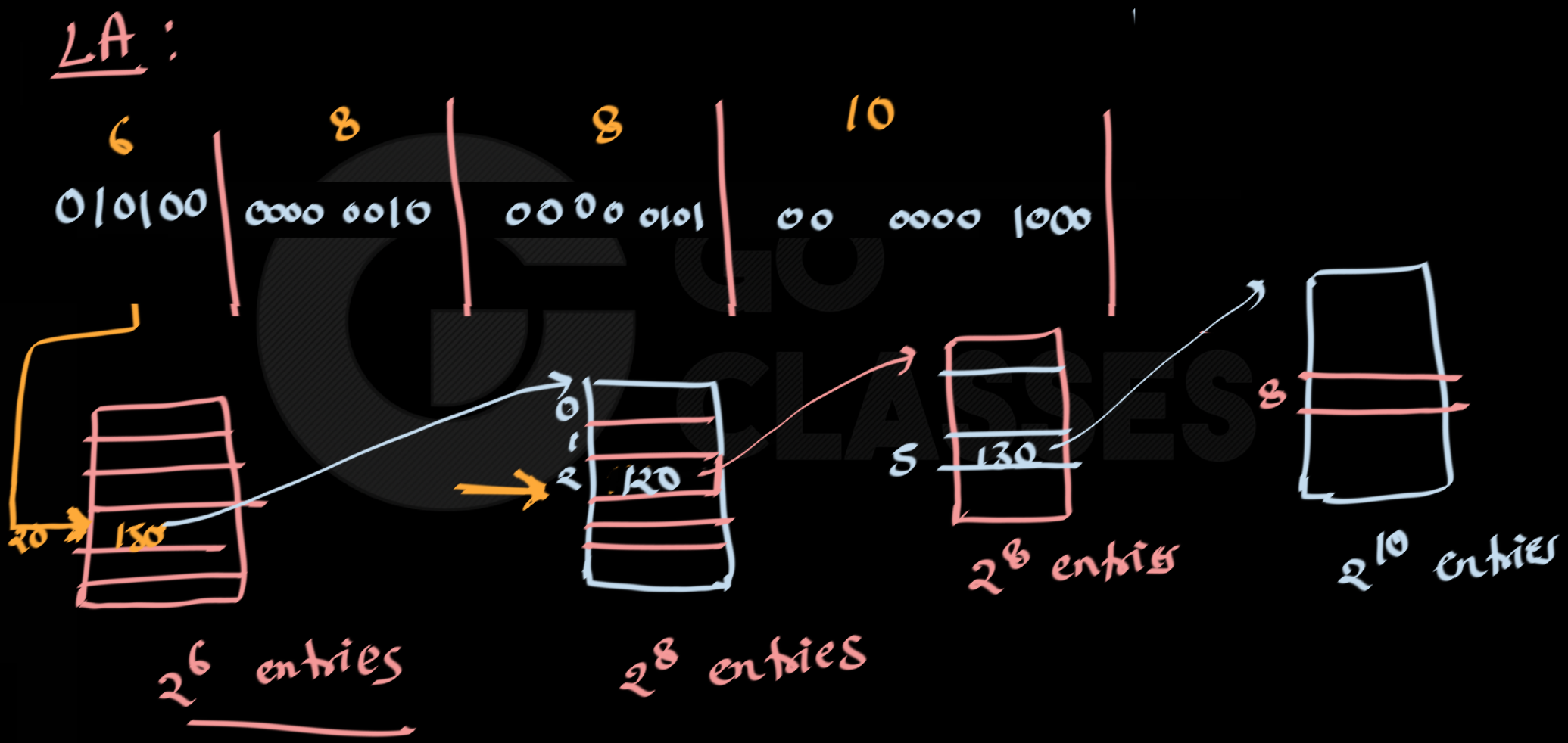

6
8
8
10
010100
0000 0010
0000 0101
00 0000 1000


20
130
0
1
2
120
$2^6$ entries
$2^8$ entries

LA :


6
8
8
10
010100
0000 0010
0000 0101
00 0000 1000


20
130
0
1
2
130
5
130
$2^6$ entries
$2^8$ entries
$2^8$ entries


LA :
6
8
8
10
0 1 0 1 0 0
0000 0010
0000 0101
00 0000 1000
20
130
0
1
2
130
5
130
8
$2^6$ entries
$2^8$ entries
$2^8$ entries
$2^{10}$ entries

GO CLASSES

## Question :

Suppose we have 32 VA and value of address is : 0x12345678, and we are using 5 level page tables. at the innermost page table the frame No is 0x0134 then calculate PA if size of page is $2^8$ bytes.

GO CLASSES

Question :

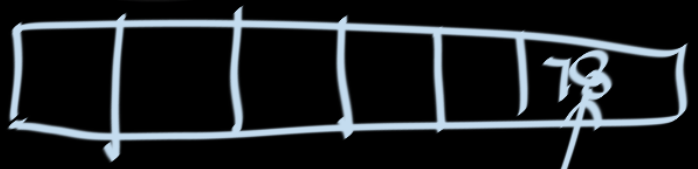
Suppose we have 32 VA and value of address is : 0x12345678, and we are using 5 level page tables. at the innermost page table the frame no is 0x0134 then calculate PA if size of page is $2^8$ bytes.

8 bits

0x012478

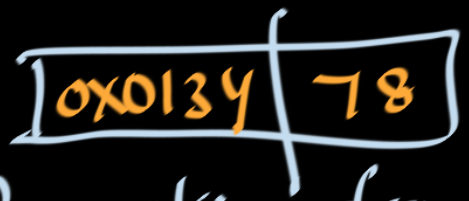
PA

Sol? :



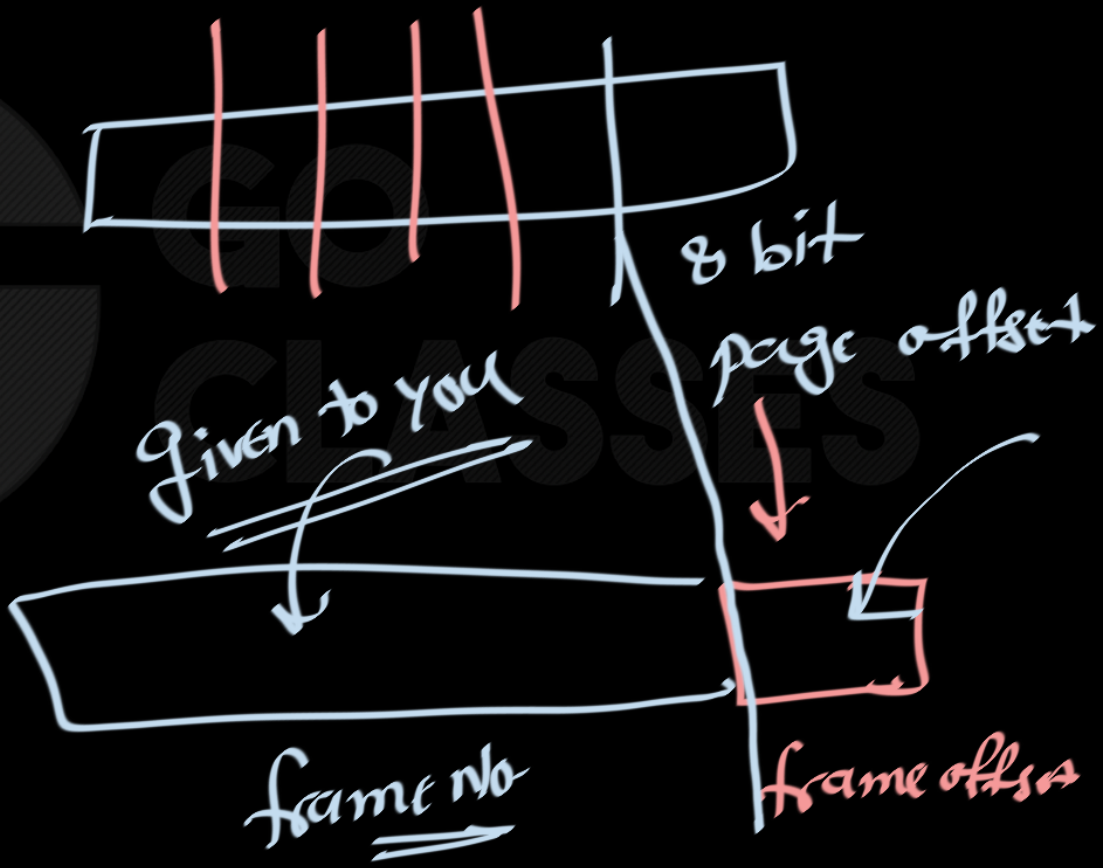

LA

8 bits

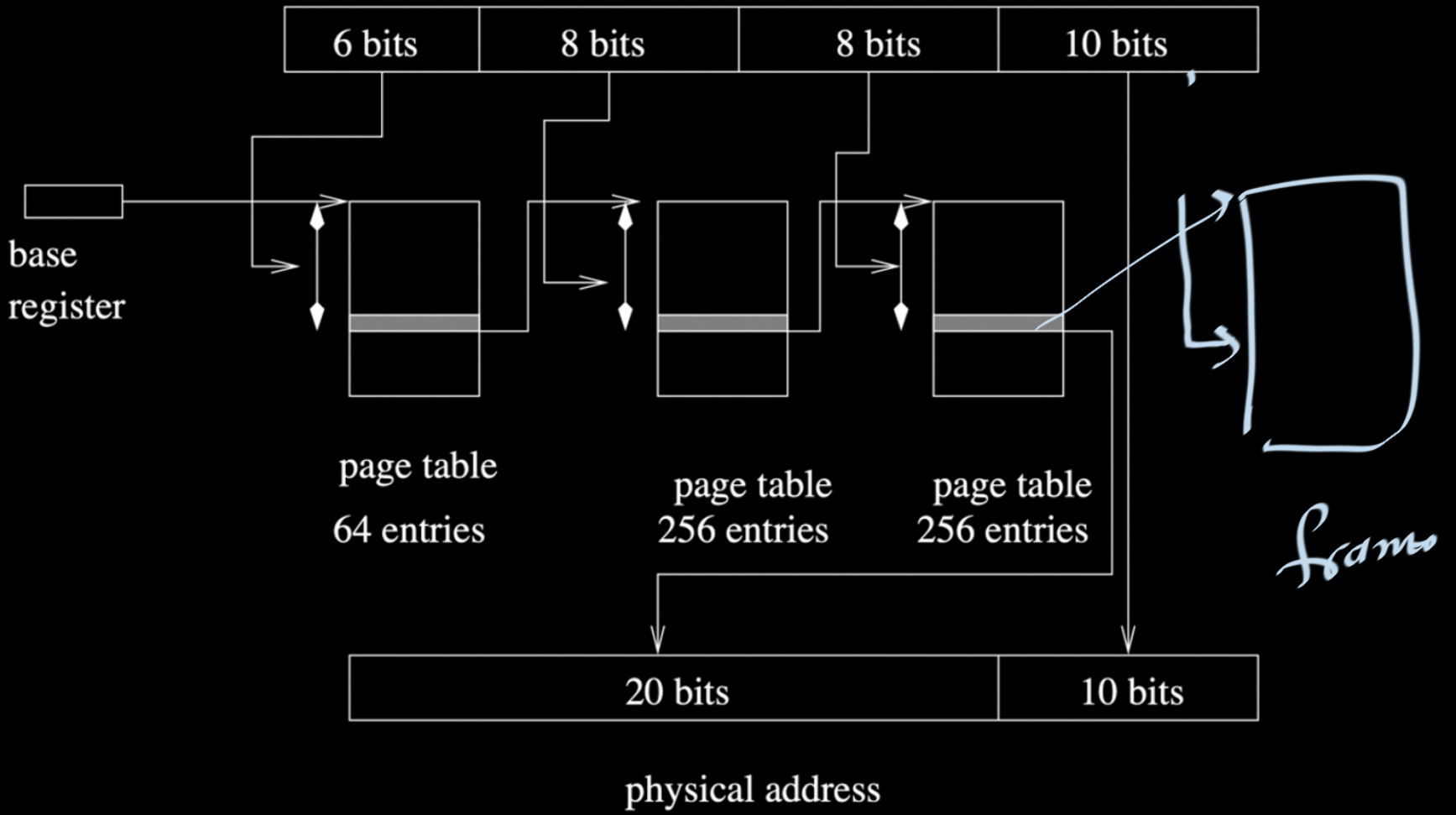

frame no

frame offset


8 bit
page offset
given to you
frame no
frame offset

| 6 bits | 8 bits | 8 bits | 10 bits |
|---|---|---|---|

address split in multilevel page table.


6 bits
8 bits
8 bits
10 bits
base
register
page table
64 entries
page table
256 entries
page table
256 entries
frame
20 bits
10 bits
physical address

# GATE CSE 2002 | Question: 19

asked in **Operating System** Sep 16, 2014 • edited Jul 13, 2018 by **kenzou** 10,436 views

32

A computer uses $32 - bit$ virtual address, and $32 - bit$ physical address. The physical memory is byte addressable, and the page size is $4\,\text{Kbytes}$. It is decided to use two level page tables to translate from virtual address to physical address. Equal number of bits should be used for indexing first level and second level page table, and the size of each table entry is $4$ bytes.

A. Give a diagram showing how a virtual address would be translated to a physical address.
B. What is the number of page table entries that can be contained in each page?
C. How many bits are available for storing protection and other information in each page table entry?

gatecse-2002 operating-system virtual-memory normal descriptive

https://gateoverflow.in/872/gate-cse-2002-question-19

# GATE CSE 2002 | Question: 19

asked in Operating System Sep 16, 2014 • edited Jul 13, 2018 by kenzou

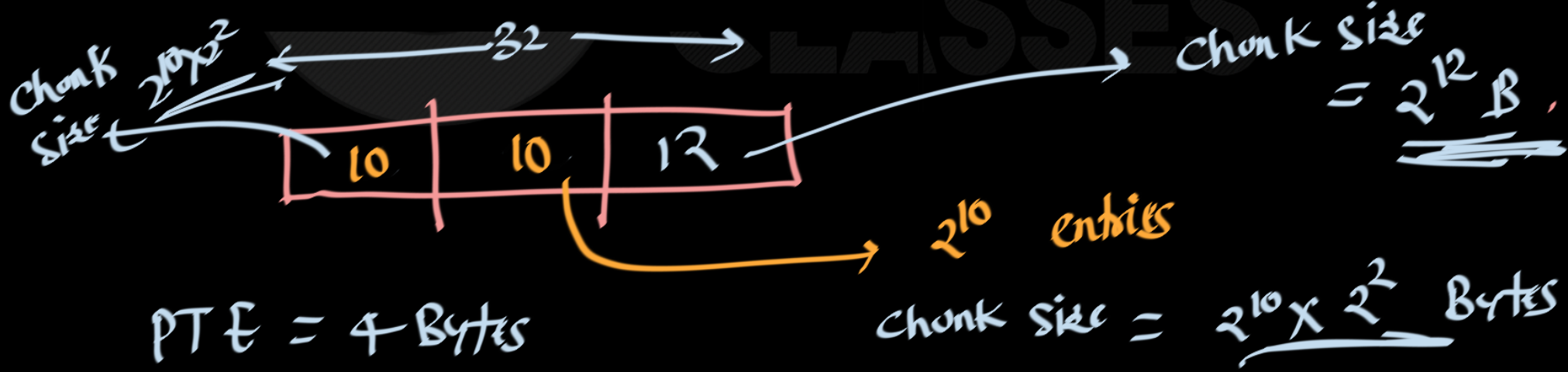
10,436 views

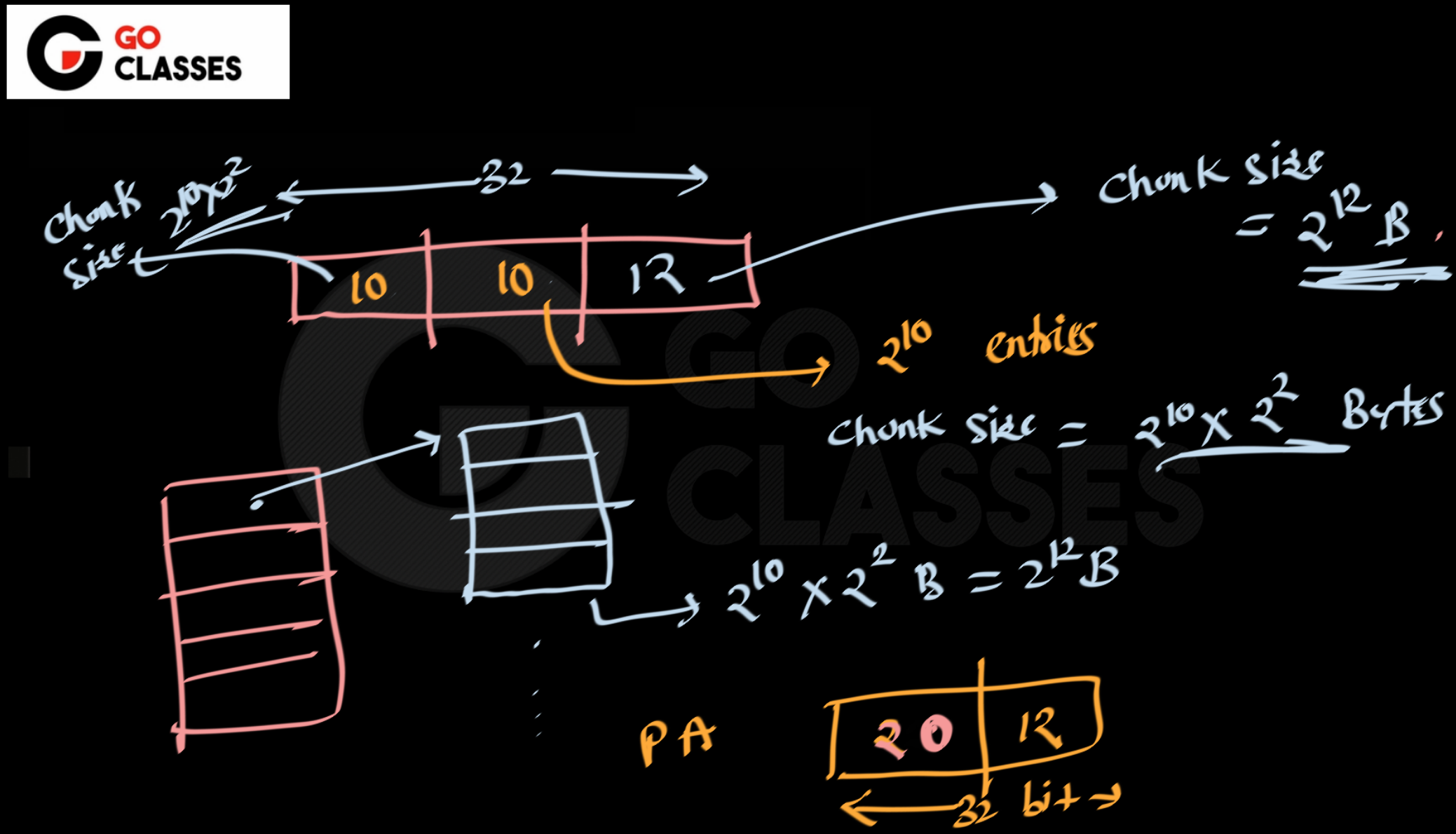
32

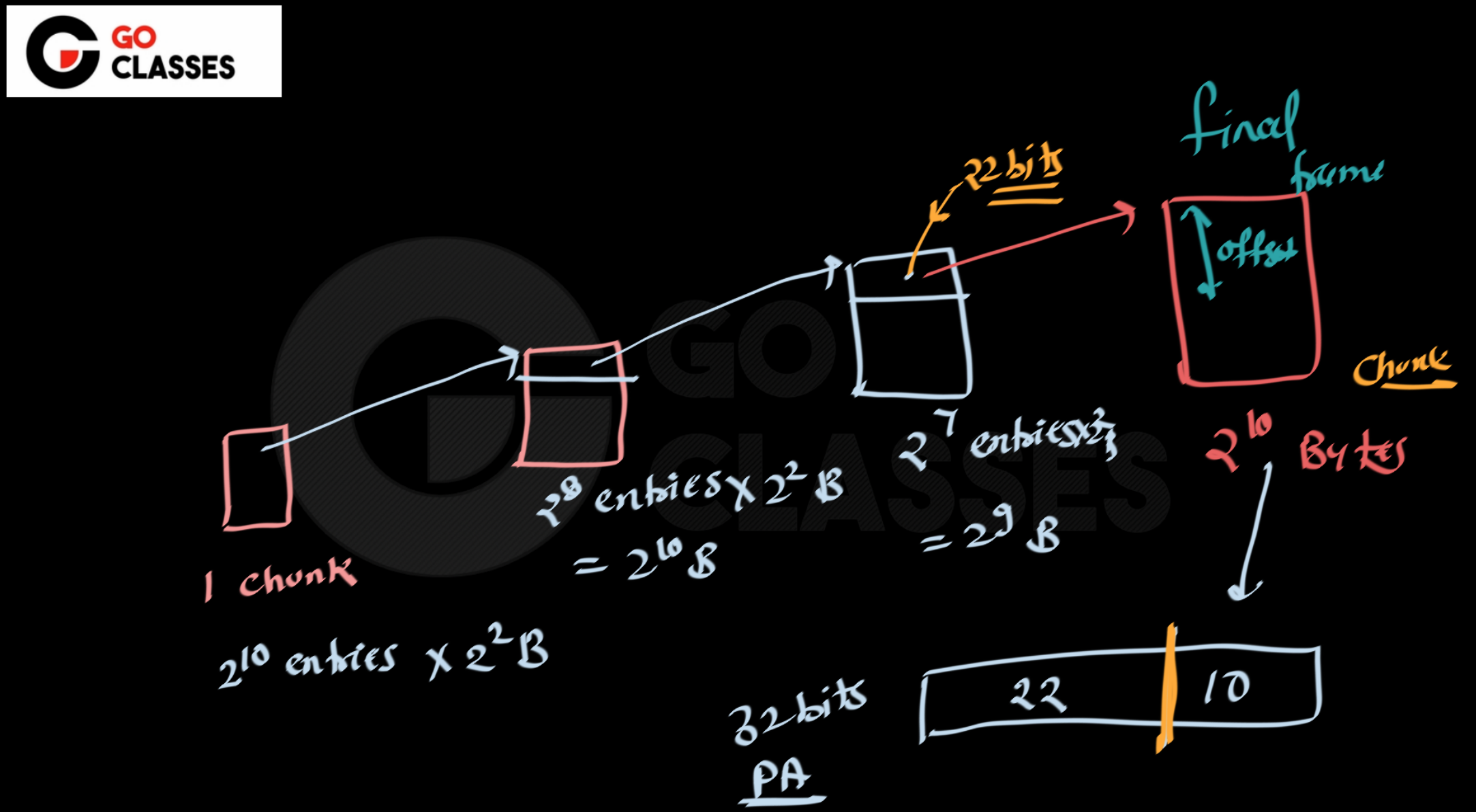
A computer uses $32 - bit$ virtual address, and $32 - bit$ physical address. The physical memory is byte addressable, and the page size is $4$ Kbytes. It is decided to use two level page tables to translate from virtual address to physical address. Equal number of bits should be used for indexing first level and second level page table, and the size of each table entry is $4$ bytes.

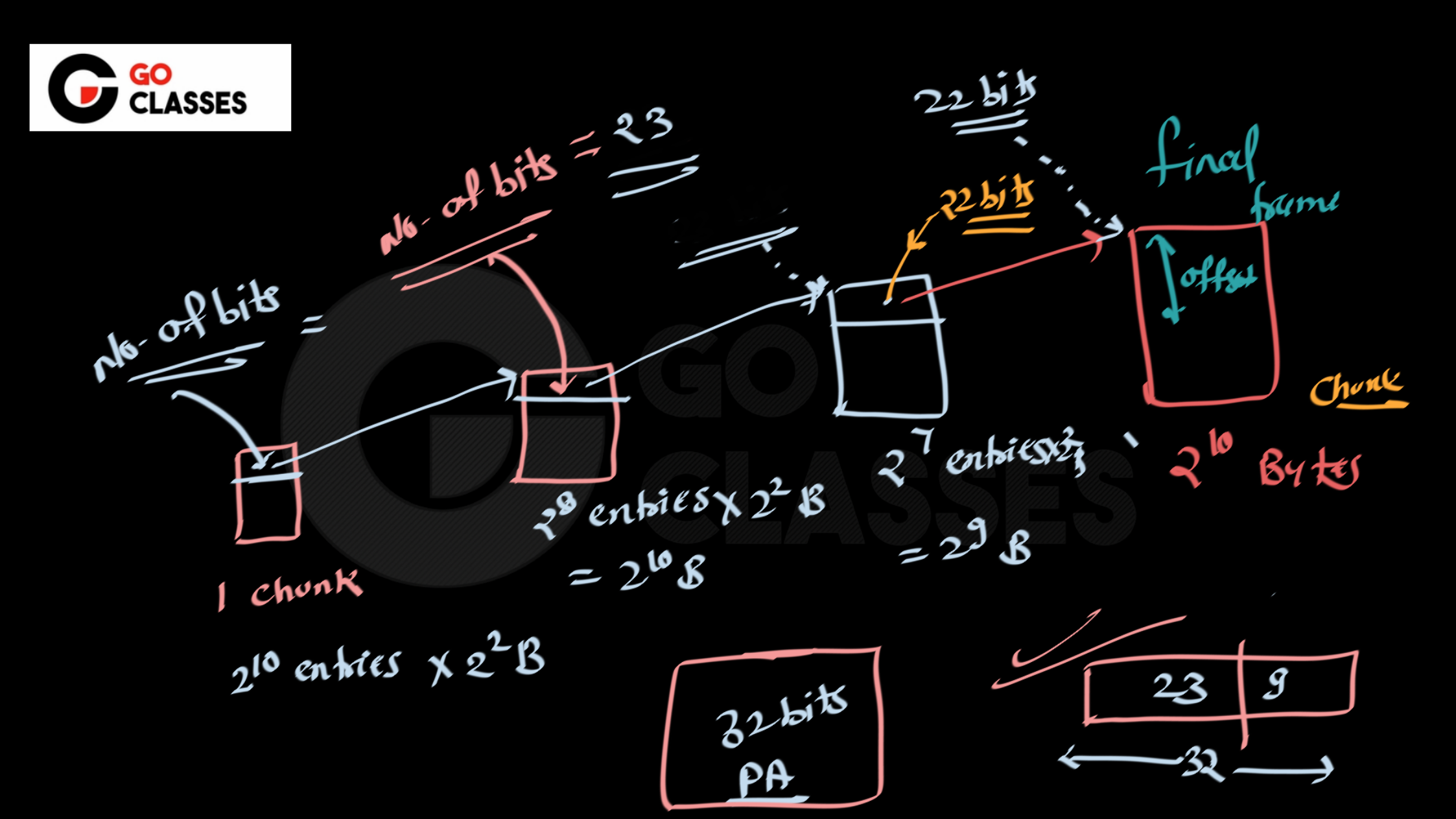
A. Give a diagram showing how a virtual address would be translated to a physical address.
B. What is the number of page table entries that can be contained in each page?
C. How many bits are available for storing protection and other information in each page table entry?

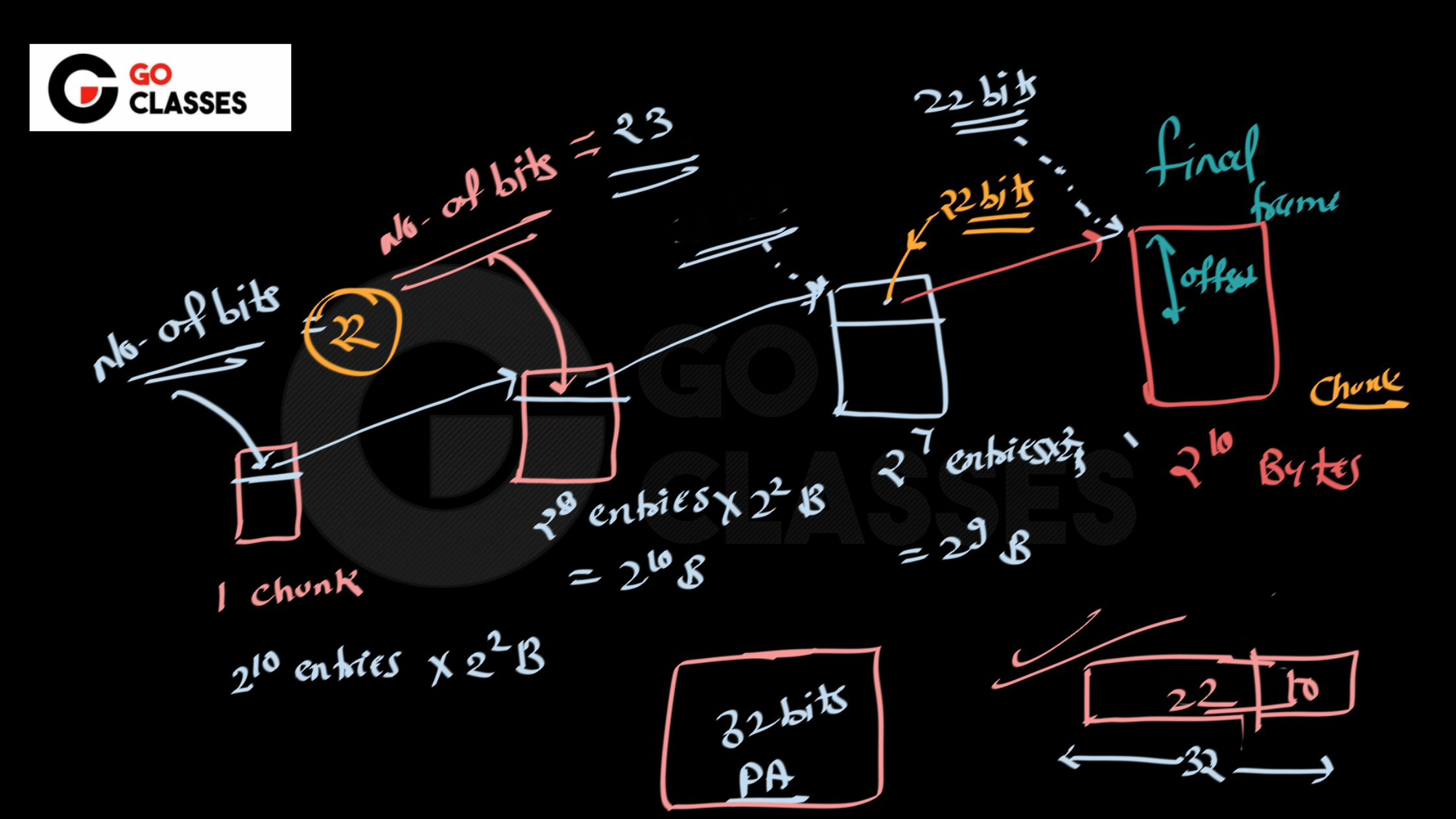
gatecse-2002 operating-system virtual-memory normal descriptive

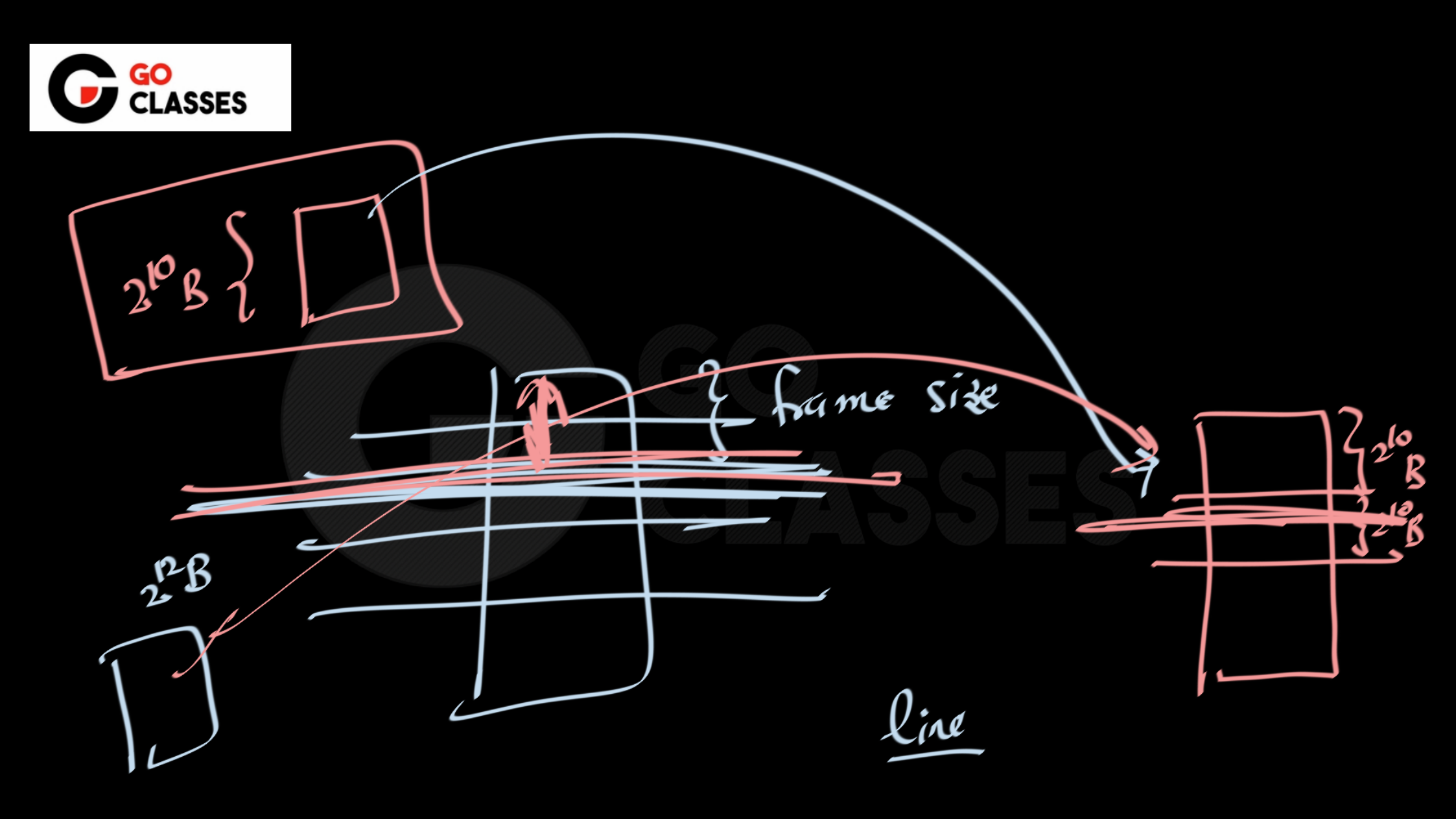



Chunk size $2^{10} \times 2^2$
32
10
10
12
Chunk size
$= 2^{12}$ B
$2^{10}$ entries
Chunk size $= 2^{10} \times 2^2$ Bytes
$2^{10} \times 2^2$ B $= 2^{12}$ B
PA
20
12
32 bit


GO CLASSES
22 bits
final frame
offset
Chunk
$2^{10}$ Bytes
1 Chunk
$2^{10}$ entries $\times 2^2$ B
$2^8$ entries $\times 2^2$ B
$= 2^{10}$ B
$2^7$ entries $\times 2^2$ B
$= 2^9$ B
32 bits
PA
22
10

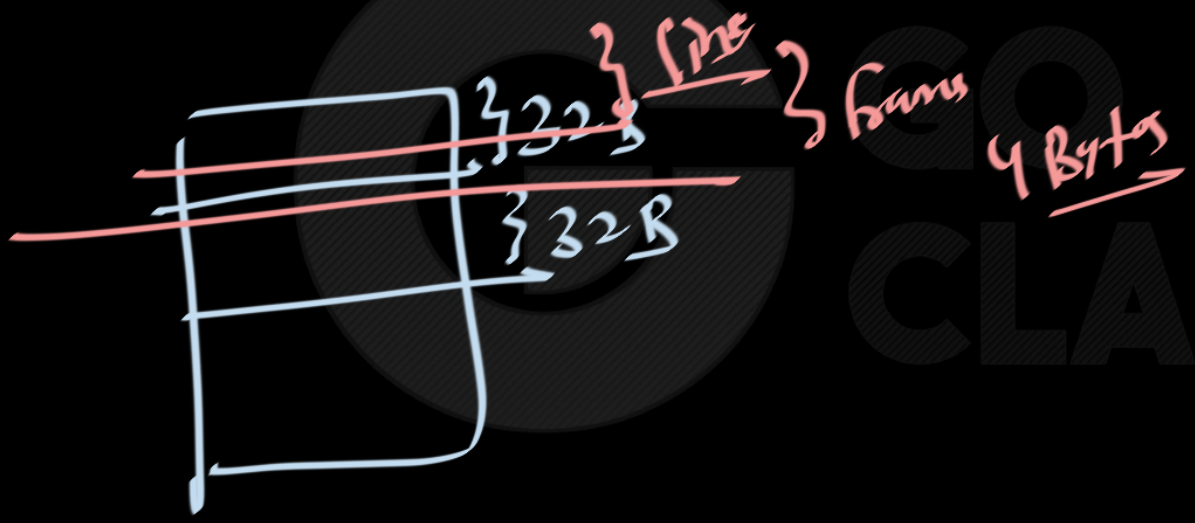

GO CLASSES
No. of bits = 23
No. of bits =
22 bits
22 bits
final frame
offset
Chunk
1 Chunk
$2^{10}$ entries $\times 2^2$ B
$2^8$ entries $\times 2^2$ B
$= 2^{10}$ B
$2^7$ entries
$= 2^9$ B
$2^{10}$ Bytes
32 bits
PA
23
9
32

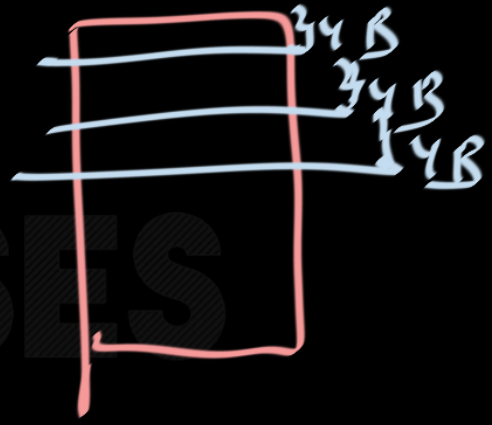

GO CLASSES
No. of bits
No. of bits = 23
22
22 bits
22 bits
final frame
offset
Chunk
1 Chunk
$2^{10}$ entries $\times 2^2$ B
$2^8$ entries $\times 2^2$ B
$= 2^{10}$ B
$2^7$ entries
$= 2^9$ B
$2^{10}$ Bytes
32 bits
PA
22
10
32

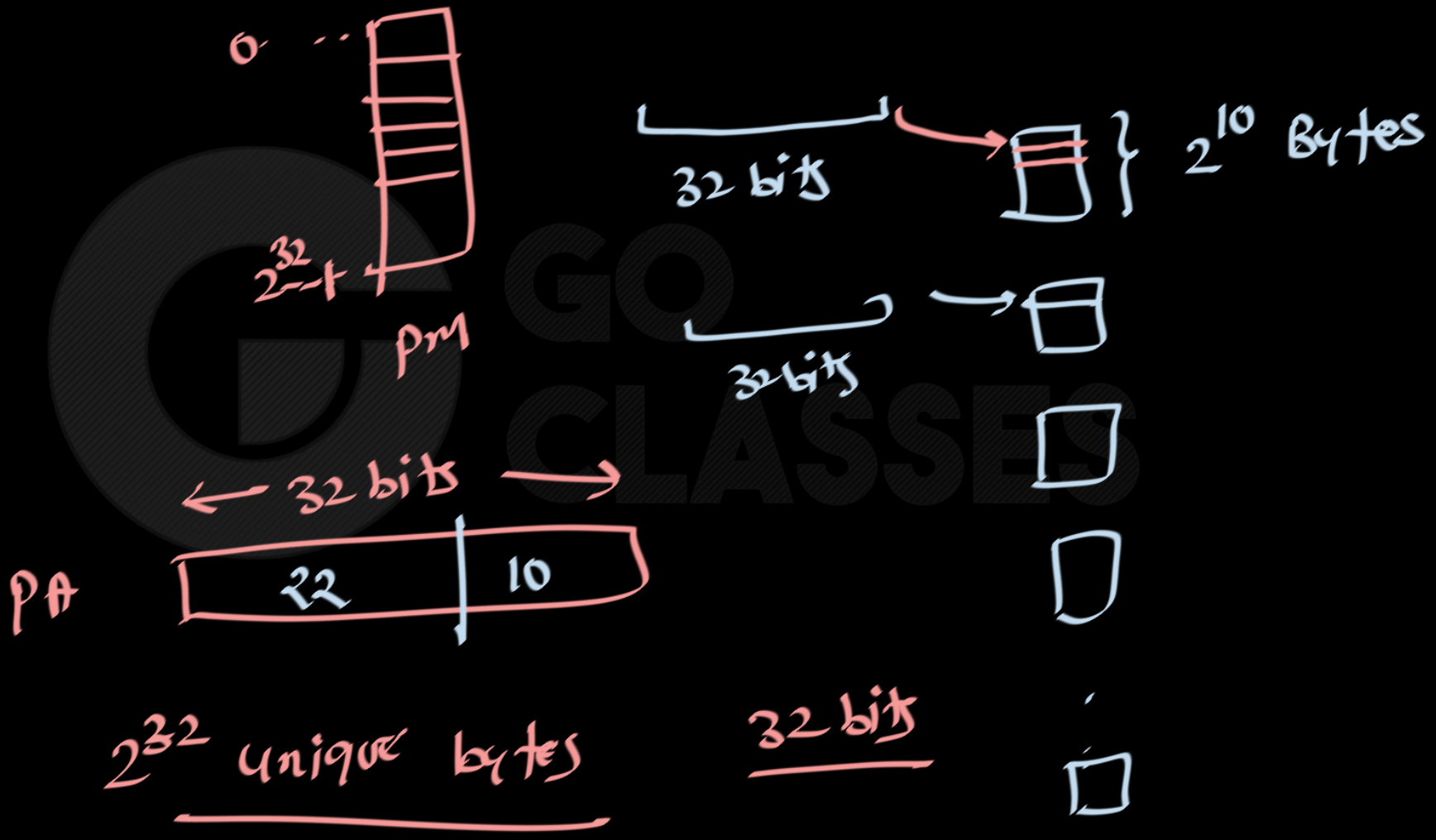

GO CLASSES
$2^{10}$ B
frame size
$2^{12}$ B
$2^{10}$ B
$2^{10}$ B
line

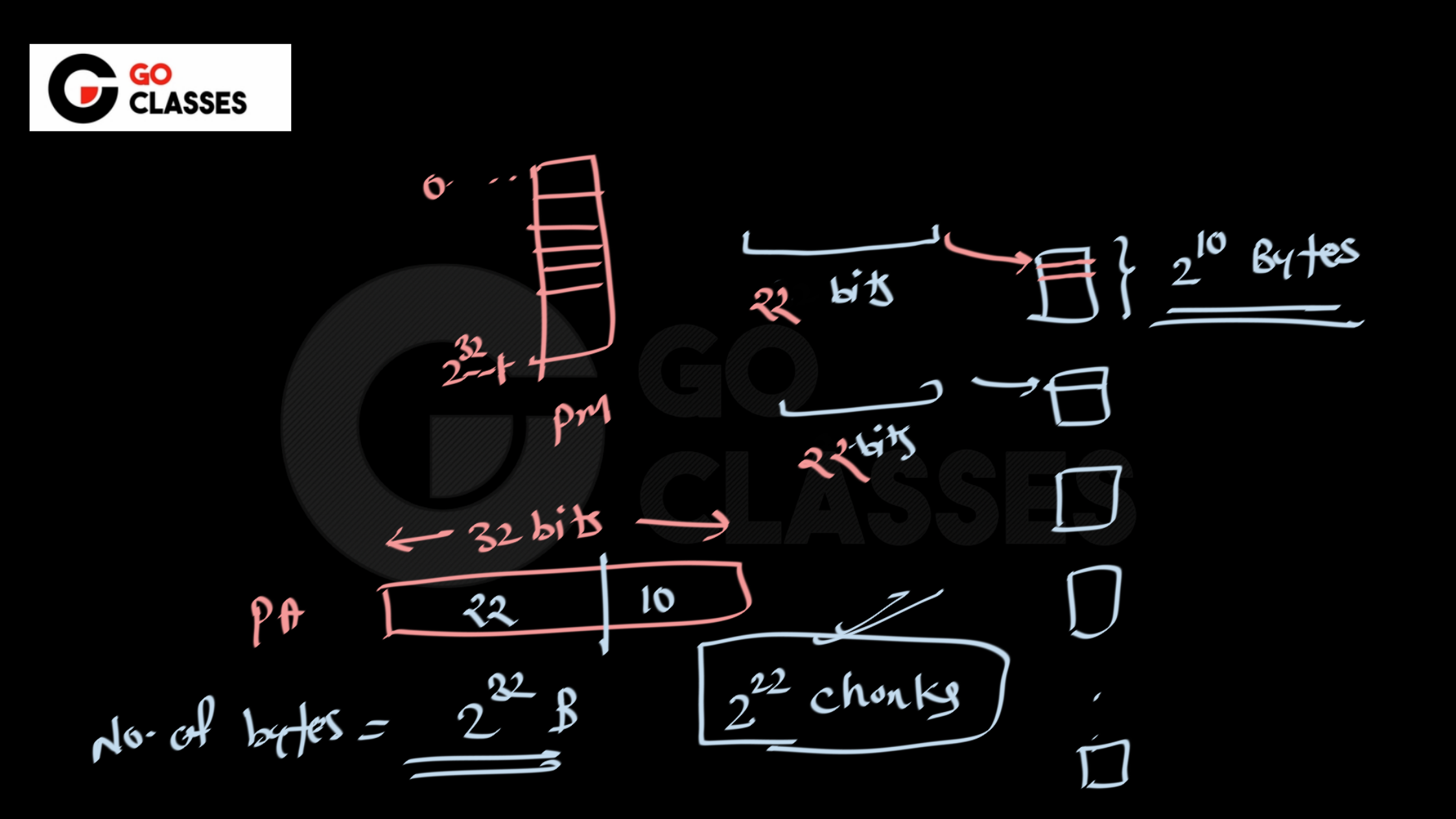
GO CLASSES

Struct ← 32 B



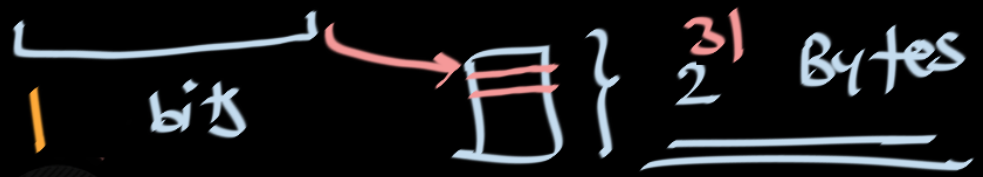




0
$2^{32}-1$
PM
32 bits
$2^{10}$ Bytes
32 bits
32 bits
PA
22
10
$2^{32}$ unique bytes
32 bits


GO CLASSES
0 ..
$2^{32}-1$
PM
22 bits
$2^{10}$ Bytes
22-bits
32 bits
PA
22
10
$2^{22}$ chunks
No. of bytes = $2^{32}$ B


17 bits
$2^{15}$ Bytes
17 bits
32
17
15


PA





maximum = 32 bits to recognise a byte

minimum = 1 bit

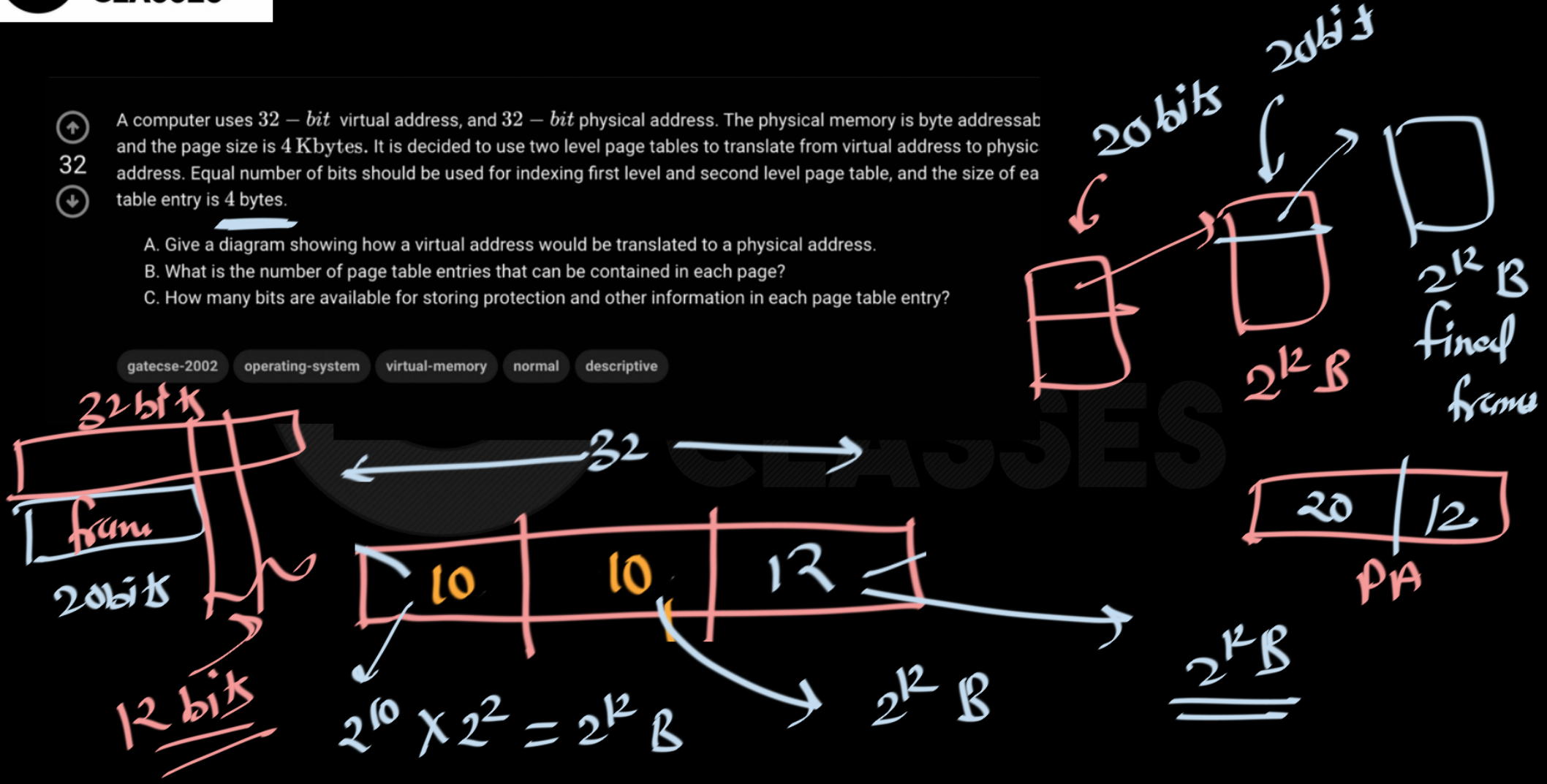
GO CLASSES

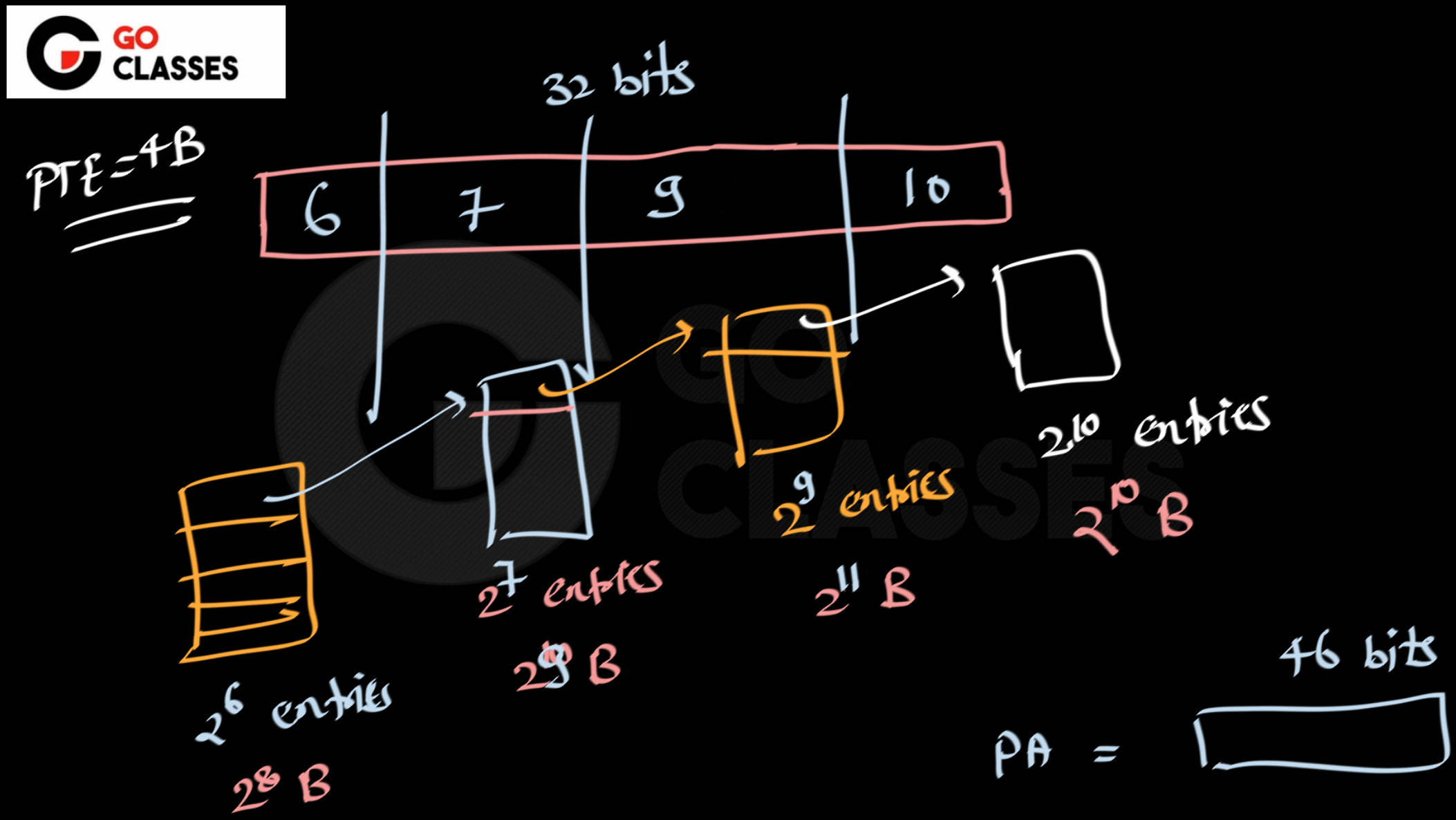
32

A computer uses $32 - bit$ virtual address, and $32 - bit$ physical address. The physical memory is byte addressab and the page size is $4\,\text{Kbytes}$. It is decided to use two level page tables to translate from virtual address to physic address. Equal number of bits should be used for indexing first level and second level page table, and the size of ea table entry is $4$ bytes.

A. Give a diagram showing how a virtual address would be translated to a physical address.
B. What is the number of page table entries that can be contained in each page?
C. How many bits are available for storing protection and other information in each page table entry?

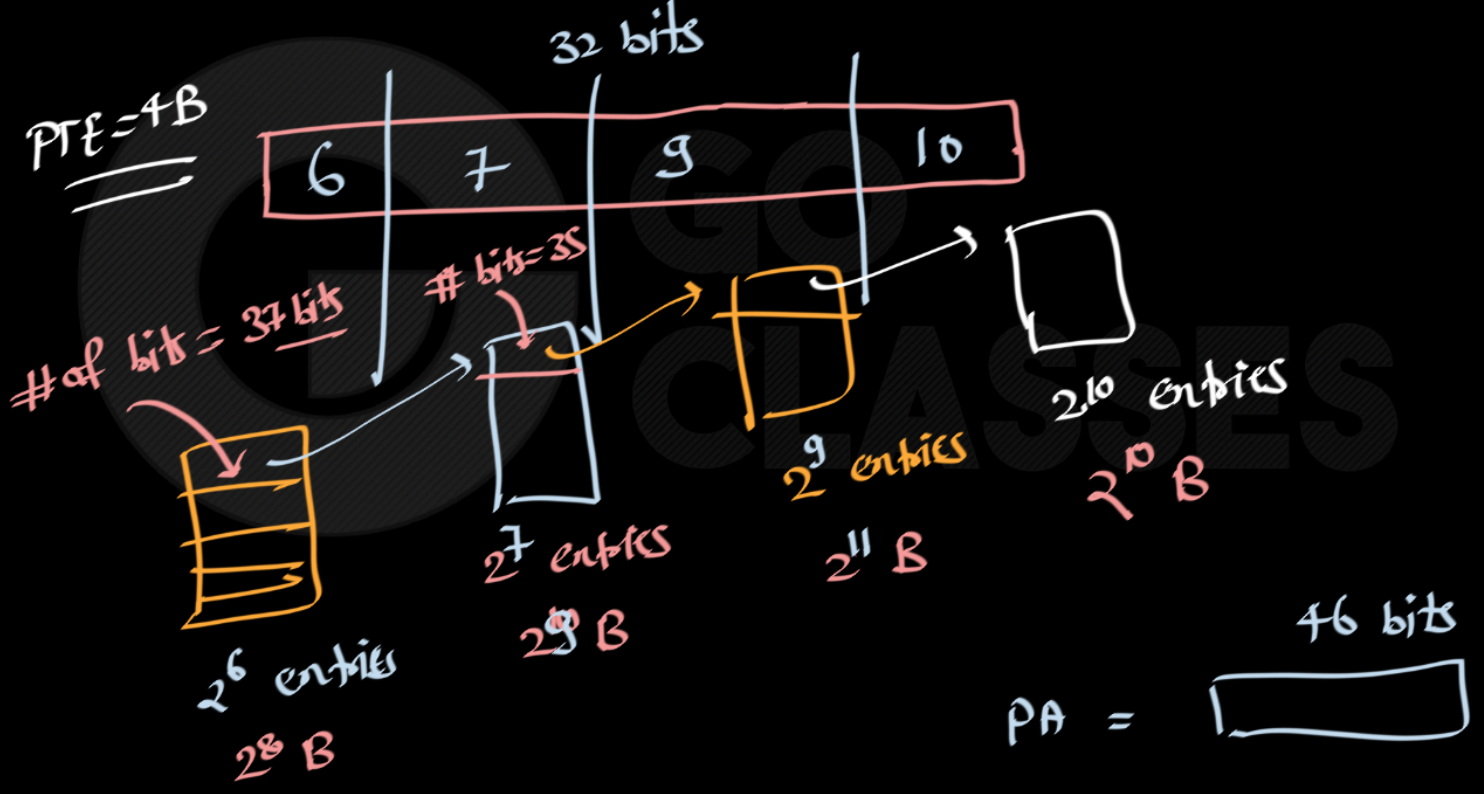
gatecse-2002 operating-system virtual-memory normal descriptive




GO CLASSES
PTE = 4B
32 bits
6
7
9
10
$2^6$ entries
$2^8$ B
$2^7$ entries
$2^9$ B
$2^9$ entries
$2^{11}$ B
$2^{10}$ entries
$2^{10}$ B
PA =
46 bits

GO CLASSES

# Summary



45

- $VA = 32\ bits$
- $PA = 32\ bits$
- Page size $= 4\ KB = 2^{12} B$
- PTE $= 4\ B$

Since page size is $4$ KB we need $\lg 4K = 12$ bits as offset bits.

Best answer

**(A)** It is given that equal number of bits should be used for indexing first level and second level page table. So, out of the remaining $32 - 12 = 20$ bits $10$ bits each must be used for indexing into first level and second level page tables as follows:

**(B)** Since $10$ bits are used for indexing to a page table, number of page table entries possible $= 2^{10} = 1024$. This is same for both first level as well as second level page tables.

**(C)**

Frame no $= 32$ bit (Physical Address) $-12$ (Offset) $= 20$

No. of bits available for Storing Protection and other information in second level page table

$= 4 \times 8 - 20$

$= 32 - 20 = 12\ bits$

No. of bits in first level page table to address a second level page table is $\log_2$ of

$$\frac{\text{Physical memory size}}{\#\text{Entries in a Second level page table} \times \text{PTE size}}$$

$$= \log_2 \left[ \frac{2^{32}}{2^{10} \times 4} \right]$$

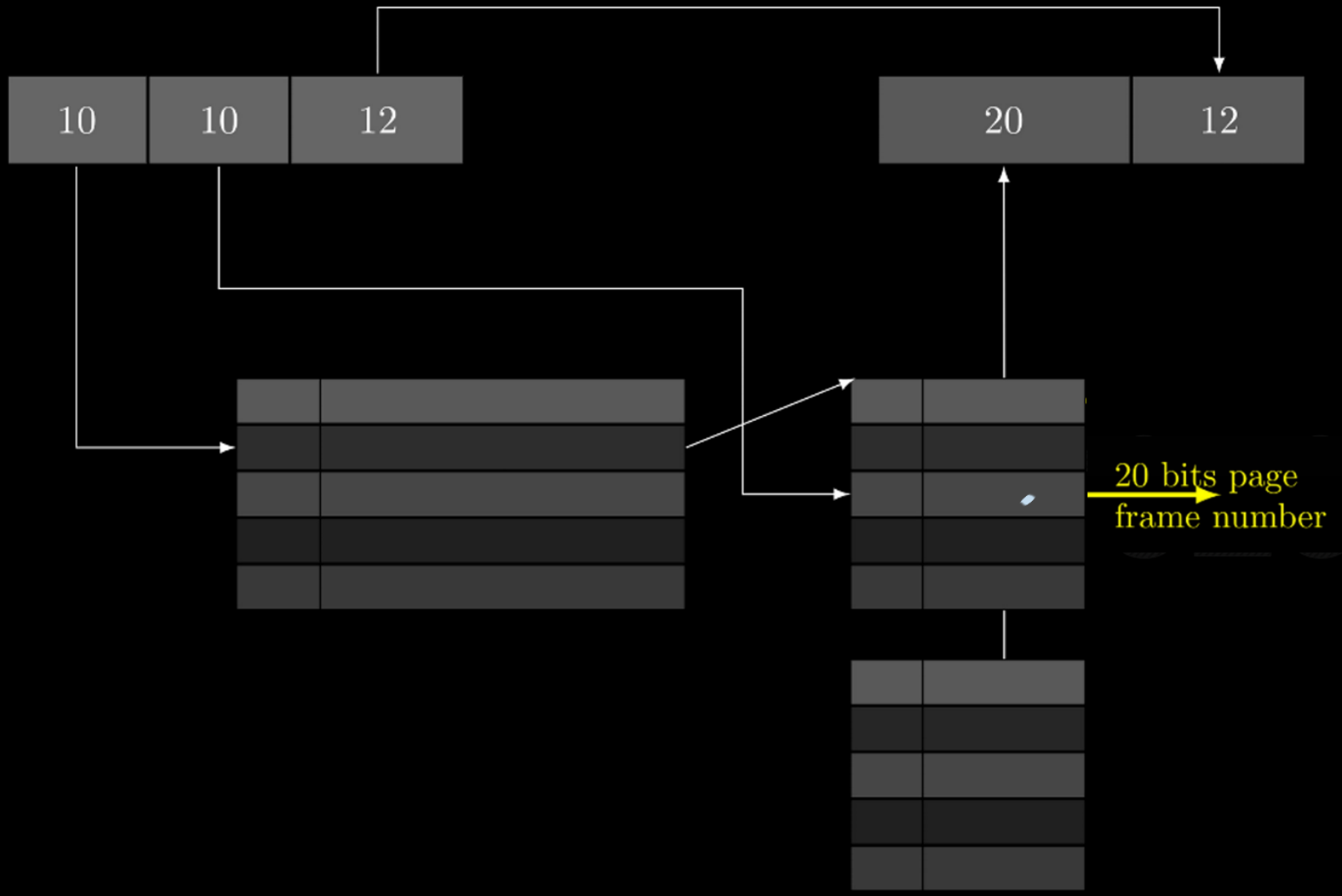
$$= \log_2 \left(2^{20}\right)$$

$$= 20\ bits.$$

So here also, the no. of bits available for storing protection and other information $= 32 - 20 = 12\ bits.$

# Operating Systems





**Q2** (10 pts) The following table shows some parameters of a virtual memory system:

| Virtual Address | Physical Address | Physical Page Size | Page Table Entry |
|---|---|---|---|
| 42 bits | 40 bits | 64 KB | 8 bytes |

**a)** (3 pts) For a single-level page table, how many page table entries are needed? How much memory is needed for storing the page table?

**b)** (4 pts) In general, the virtual address space of a process is mostly empty. To reduce the size of the page table, a multilevel page table is used for address translation. How many levels of page tables will be needed for address translation, given that the size of a page table at any level is the size of a physical page? Draw a virtual address showing the number of bits for the page table index at each level and the page offset.

H.W.

Q2 Solution:

a) Page size = 64 KB
Therefore, Page offset = 16 bits
Virtual page number = 42 bits – 16 bits = 26 bits
Number of page table entries = $2^{26}$
Page Table Size = $2^{26}$ × 8 bytes = $2^{29}$ bytes = 512 MB

b) Size of a page table = Size of a physical page = 64 KB
Each page table entry is 8 bytes
Therefore, number of entries per page table = $2^{16}$ / 8 = $2^{13}$ entries
13 bits are needed to index a page table at any level
Virtual page number = 26 bits

# GATE CSE 2003 | Question: 26

25

In a system with $32$ bit virtual addresses and $1$ KB page size, use of one-level page tables for virtual to physical address translation is not practical because of

A. the large amount of internal fragmentation
B. the large amount of external fragmentation
C. the large memory overhead in maintaining page tables
D. the large computation overhead in the translation process

gatecse-2003 | operating-system | virtual-memory | normal

https://gateoverflow.in/916/gate-cse-2003-question-26

43

A. Internal fragmentation exists only in the last level of paging.

B. There is no External fragmentation in the paging.

C. $\frac{2^{32}}{2^{10}} = 2^{22} = 4M$ entries in the page table which is very large. (**Answer**)

D. Not much relevant.

Best answer

Abhishek Singhal

# Question

Consider a 3-level page table on a sytem where pages are 256 bytes, page table entries are 2 bytes, and

- first level page tables contain 16 entries
- second level page tables contains 128 entries
- third level page tables contain 128 entries

What is size of VAS ?


$\Rightarrow 4+7+7+8$

$12+14 = 26$

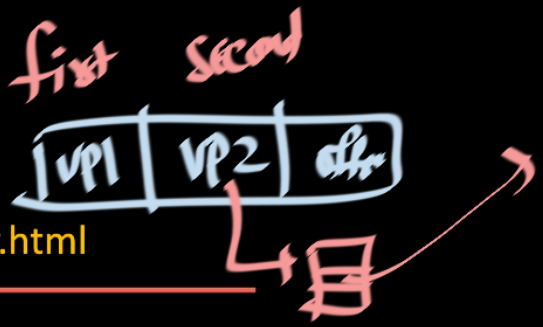
# Question

Lets assume there is a two-level page table in your system.

The virtual address is split as <vpn part1, vpn part2, page offset>.

Which of the following statement is true?

A. vpn part 1 is used to index the physical page that contains data
B. vpn part 1 is used to get the base address of the first-level table
C. page offset is used to index the TLB
D. vpn part 2 is used to index the PTE in the second-level page table




https://www.cs.virginia.edu/~cr4bd/3330/F2020/files/f2018quizzes-key.html

# Question

Lets assume there is a two-level page table in your system.

The virtual address is split as <vpn part1, vpn part2, page offset>.

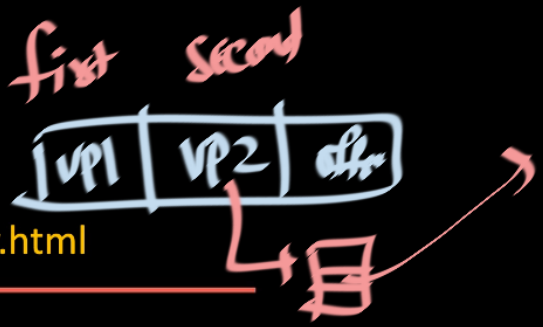
Which of the following statement is true?

A. vpn part 1 is used to index the physical page that contains data

B. vpn part 1 is used to get the base address of the first-level table

C. page offset is used to index the TLB

D. vpn part 2 is used to index the PTE in the second-level page table




https://www.cs.virginia.edu/~cr4bd/3330/F2020/files/f2018quizzes-key.html

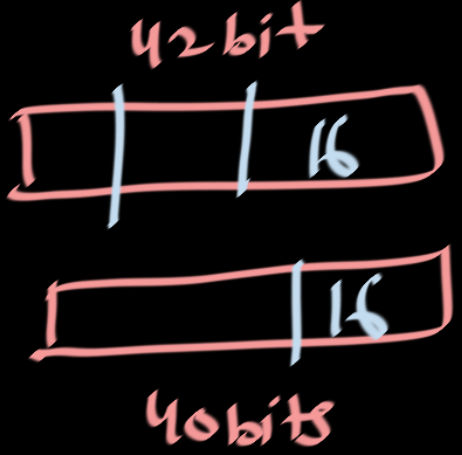
# Question

**Information for questions 13–17**

Consider a system with:

- two-level page tables
- 42-bit virtual addresses
- 40-bit physical addresses
- 8 byte page table entries
- 64KB ($2^{16}$ byte) pages
- 64KB ($2^{16}$ byte) page tables at each level

Chant size




**Question 17 [2 pt]:** (see above) Which of the following addresses use the same **first-level** page table entry as `0x00 123 456 789`? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A ☐ `0x00 123 444 444`

B ☐ `0x00 0A9 876 789`

C ☐ `0x00 136 345 678`

D ☐ `0x00 000 006 789`

https://www.cs.virginia.edu/~cr4bd/3330/F2020/files/s2018key3H.pdf

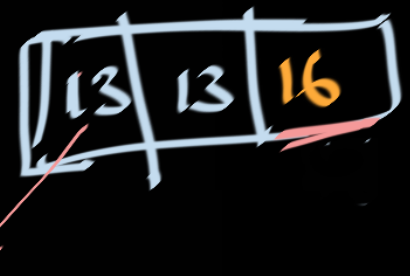
**Question 17 [2 pt]:** (see above) Which of the following addresses use the same **first-level** page table entry as `0x00 123 456 789`? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A ☐ `0x00 123 444 444`

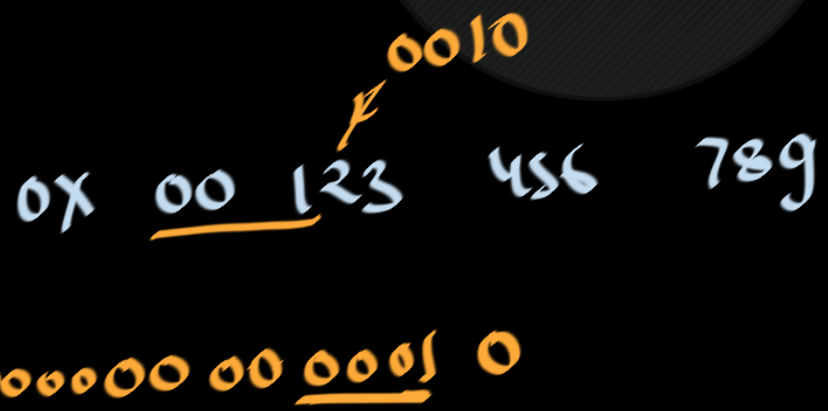
B ☐ `0x00 0A9 876 789`

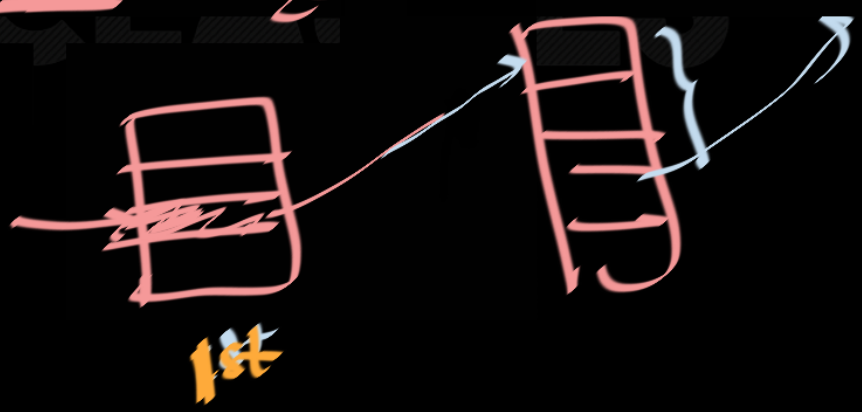
C ☐ `0x00 136 345 678`

D ☐ `0x00 000 006 789`




$2^{16}$ bytes $= \frac{2^{16}}{2^3} = 2^{13}$ entries



**Question 17 [2 pt]:** (see above) Which of the following addresses use the same **first-level** page table entry as `0x00 123 456 789`? **Place a ✓in each box corresponding to a correct answer and leave other boxes blank.**

A [✓] `0x00 123 444 444`

B [ ] `0x00 0A9 876 789`

C [✓] `0x00 136 345 678`

D [ ] `0x00 000 006 789`

# Question

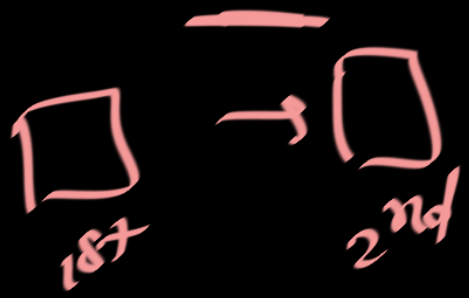
**Information for questions 7–11**

Suppose a processor has:

- 8KB pages
- 32-bit virtual addresses
- 28-bit physical addresses
- a two-level page table, with a 1KB page table at the first level, and 8KB page tables at the second level
- 4-byte page table entries

**Question 11 [2 pt]:** (see above) Which of the following virtual addresses share the same second-level page table as `0x12345678`? **Select all that apply.**

**A** `0x123FD000`
**B** `0x00046678`
**C** `0x12345FFF`
**D** `0x99345678`




https://www.cs.virginia.edu/~cr4bd/3330/F2020/files/s2017finalkeyG.pdf

**Question 11 [2 pt]:** (see above) Which of the following virtual addresses share the same second-level page table as `0x12345678`? **Select all that apply.**

**A** `0x123FD000`
**B** `0x00046678`
**C** `0x12345FFF`
**D** `0x99345678`

Answer: A C

# Question

4. (27 points total) Two-Level Page-based virtual Addressing. Consider a 32-bit machine with a multi-level virtual memory system with 32-bit pointers and 4096-byte pages that supports two-levels of page tables. All Page Table Entries (PTEs) are 4 bytes.

   a. (6 points) Show the complete format of a virtual address.

   b. (6 points) Explain the steps the hardware takes in translating a virtual address to a physical address for this scheme (do not worry about supporting paging to disk).

   c. (3 points) How many memory operations are required to read or write a single 32-bit word?

   d. (4 points) List the fields of a Page Table Entry (PTE).

   e. (6 points) How much physical memory is needed for a process with one page of virtual memory?

   f. (2 points) What happens in the virtual memory subsystem on a context switch?

https://hkn.eecs.berkeley.edu/examfiles/cs162_sp03_mt1_sol.pdf

4. (27 points total) Two-Level Page-based Virtual Addressing. Consider a 32-bit machine with a multi-level virtual memory system with 32-bit pointers and 4096-byte pages that supports two-levels of page tables. All Page Table Entries (PTEs) are 4 bytes.

   a. (6 points) Show the *complete* format of a virtual address.

   *Each address is broken up into three parts:*

| *Page Level 1 – 10 bits* | *Page Level 2 – 10 bits* | *Offset – 12 bits* |
|---|---|---|

   *1 point for each field, 1 point for correct bit length of each field.*

   *This is a multi-level scheme using two sets of page tables for* ***each*** *process: an "outer" one and an "inner" one. PL 1 is an index into the outer page table, containing PTE's that contain pointers to the inner page table (which can be paged). In other words, the inner page table is broken up into pages that do not have to be stored contiguously.*

   *The PTE at outer [PL 1] contains a pointer to the PL $1^{th}$ page in the inner page table. PL 2 is an index into the PL $1^{th}$ page in the inner page table. PTE's are 4 bytes, so there are 1,024 PTE's per page. Thus, PL 2 must be exactly 10 bits. The index at PL 2 points to the page that holds virtual addresses starting with PL 1 PL 2. The particular byte is the $d^{th}$ byte in that page. Since pages are 4K bytes, d must be exactly 12 bits. Since d is 12 bits and PL 2 is 10 bits, PL 1 must be 10 bits.*

b. (6 points) Explain the steps the hardware takes in translating a virtual address to a physical address for this scheme (do not worry about supporting paging to disk).

*See Problem 3a for the picture of the steps that are taken. The PTBR is used to locate the outer page table* ***in physical memory****. PL 1 is used to index into the outer page table. As above, the PTE at this index points to a page in the inner page table. PL 2 is used to index in this inner page table, and the PTE in the inner page table points to the physical page. Offset is a location on the physical page. If you didn't mention the PTBR, we subtracted one point.*

*Note that each PTE should have a valid bit and several protection bits – e.g., read, write, execute, valid. The memory operation (load, store, load for execution) must agree with these bits in both sets of PTE's (inner and outer). Otherwise the hardware will generate an interrupt. If you didn't mention the role of the valid bit or the protection bits, we subtracted one point for each missing role.*

*If you missed a level, we subtracted one point for each level missed. If you included an incorrect step, we subtracted one point.*

c. (3 points) How many memory operations are required to read or write a single 32-bit word?

*Without extra hardware, performing a memory operation takes 3 actual memory operations: two page table lookups in addition to the desired memory operation. We did not award partial credit for this problem.*

d. (4 points) List the fields of a Page Table Entry (PTE).

*Each PTE will have a pointer to the proper page (two points) plus several bits – read, write, execute, (1 point for protection bits), and valid (one point). This information can all fit into 4 bytes, since if physical memory is $2^{32}$ bytes, then 20 bits will be needed to point to the proper page, leaving ample space (12 bits) for the information bits.*

*If you didn't mention the valid and protection bits, then you lost 2 points. If you added incorrect fields, we subtracted one point for each incorrect field.*

e. (6 points) How much physical memory is needed for a process with one page of virtual memory?

*Three pages are needed: one for the outer page table, one for one page of the inner page table, and one for the process' single page.*

*Note that the inner page table does not need 4M ($2^{10}$ * 4K), because the outer page table enables you to only have inner page table pages for those pages that are part of the process's virtual address space.*

*For partially correct answers, we subtracted three points for the wrong total and one point for each incorrect page level. Answers that stated that only enough memory was needed for a PTE at the outer and inner levels (instead of an entire page), were penalized at least three points.*

f. (2 points) What happens *in the virtual memory subsystem* on a context switch?

*On a context switch, the PTBR of the new process must be loaded.*
*Note that it is not necessary to save the PTBR of the outgoing process as that does not change on a context switch, as it is already stored in the process' control block. We did not give partial credit for answers that only saved the PTBR.*

# Question

You are designing a paged virtual memory system on a system with 32-bit virtual addresses and 48-bit physical addresses. Each page is 4KB ($2^{12}$ bytes), and each page table entry is 64 bits (8 bytes, $2^3$ bytes).

**a. (2 marks)**

In a single-level paged system, how many bits of a virtual memory address would refer to the page number and how many to the offset? How many bits of a physical address would refer to the frame number and how many to the offset?

**b. (2 marks)**

How many bytes would a single-level page table require?

**c. (2 marks)**

If a page table entry contains a frame number, a valid bit, a writeable bit, and a single bit for tracking page useage, how many bits per page table entry are unused?

**d. (2 marks)**

How many page table entries fit onto a single page?

**e. (2 marks)**

What is the minimum number of levels necessary to implement a multi-level paged system if each page table at each level must fit into a single page?



https://student.cs.uwaterloo.ca/~cs350/common/old-exams/F18-midterm-sol.pdf

# Question

Consider a system, Starting with a 48-bit virtual addresses, it uses 4 levels of page table to translate the address to a 52-bit physical address.

Each page table entry is 8 bytes long.

If the page size is increased, the number of levels of page table can be reduced. How large must pages be in order to translate 48-bit virtual addresses with only a 2 level page table?

Notice first that the page size is 2^12 in Figure 1. The page table size is also 2^12 (2^9 * 8 bytes per page table entry, for a 64 bit processor).

For a two level page table, say page size is 2^n. The page table size is also 2^n bytes in this multi-level page table. So the number of PTE in a page table is 2^(n-3).

So the virtual address is broken into (n-3), (n-3), and n bits.

(n-3) + (n-3) + n = 48

n = 18, so page size is 2^18 = 256KB.

The 48-bit virtual address is divided into 15, 15, 18 bits. The first 15 bits index into the first page table, the second 15 bits index in the second page table, and the rest of the 18 bits is the page offset.

# GATE CSE 2008 | Question: 67

237

A processor uses $36$ bit physical address and $32$ bit virtual addresses, with a page frame size of $4$ Kbytes. Each page table entry is of size $4$ bytes. A three level page table is used for virtual to physical address translation, where the virtual address is used as follows:

- Bits $30 - 31$ are used to index into the first level page table.
- Bits $21 - 29$ are used to index into the 2nd level page table.
- Bits $12 - 20$ are used to index into the 3rd level page table.
- Bits $0 - 11$ are used as offset within the page.

The number of bits required for addressing the next level page table(or page frame) in the page table entry of the first, second and third level page tables are respectively

A. $20,20,20$
B. $24,24,24$
C. $24,24,20$
D. $25,25,24$

gatecse-2008 operating-system virtual-memory normal

## GATE CSE 2008 | Question: 67

237

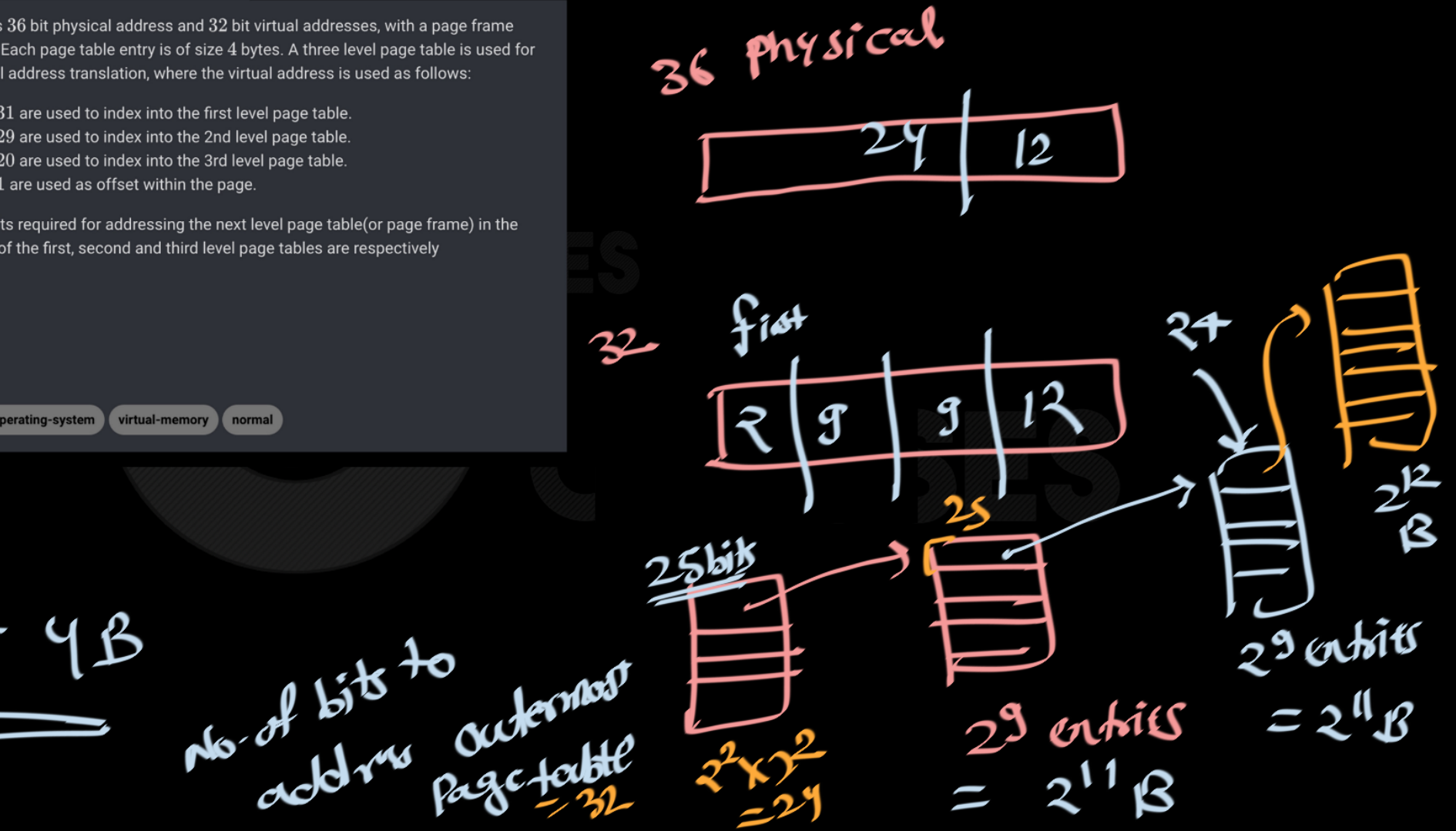
A processor uses $36$ bit physical address and $32$ bit virtual addresses, with a page frame size of $4$ Kbytes. Each page table entry is of size $4$ bytes. A three level page table is used for virtual to physical address translation, where the virtual address is used as follows:

- Bits $30 - 31$ are used to index into the first level page table.
- Bits $21 - 29$ are used to index into the 2nd level page table.
- Bits $12 - 20$ are used to index into the 3rd level page table.
- Bits $0 - 11$ are used as offset within the page.

The number of bits required for addressing the next level page table(or page frame) in the page table entry of the first, second and third level page tables are respectively

A. $20,20,20$
B. $24,24,24$
C. $24,24,20$
D. $25,25,24$

gatecse-2008 | operating-system | virtual-memory | normal



285

Best answer

Physical address is $36$ bits. So, number of bits to represent a page frame $= 36 - 12 = 24\ bits$ ($12$ offset bits as given in question to address $4$ KB assuming byte addressing). So, each entry in a third level page table must have $24$ bits for addressing the page frames.

A page in logical address space corresponds to a page frame in physical address space. So, in logical address space also we need $12$ bits as offset bits. From the logical address which is of $32$ bits, we are now left with $32 - 12 = 20\ bits$; these $20\ bits$ will be divided into three partitions (as given in the question) so that each partition represents 'which entry' in the $i^{th}$ level page table we are referring to.

- An entry in level $i$ page table determines 'which page table' at $(i+1)^{th}$ level is being referred.

Now, there is only 1 first level page table. But there can be many second level and third level page tables and "how many" of these exist depends on the physical memory capacity. (In actual the no. of such page tables depend on the memory usage of a given process, but for addressing we need to consider the worst case scenario). The simple formula for getting the number of page tables possible at a level is to divide the available physical memory size by the size of a given level page table.

$$\text{Number of third level page tables possible} = \frac{\text{Physical memory size}}{\text{Size of a third level page table}}$$
$$= \frac{2^{36}}{\text{Number of entries in a single third level page table} \times \text{ Size of an entry}}$$
$$= \frac{2^{36}}{2^9 \times 4} \because \text{(bits 12-20 gives 9 bits)}$$
$$= \frac{2^{36}}{2^{11}}$$
$$= 2^{25}$$

PS: No. of third level page tables possible means the no. of distinct addresses a page table can have. At any given time, no. of page tables at level $j$ is equal to the no. of entries in the level $j-1$, but here we are considering the **possible** page table addresses.

http://www.cs.utexas.edu/~lorenzo/corsi/cs372/06F/hw/3sol.html See Problem 3, second part solution - It clearly says that we should not assume that page tables are page aligned (page table size need not be same as page size unless told so in the question and different level page tables can have different sizes).
So, we need $25$ bits in second level page table for addressing the third level page tables.

Similarly we need to find the no. of possible second level page tables and we need to address each of them in first level page table.

Now,

$$\text{Number of second level page tables possible} = \frac{\text{Physical memory size}}{\text{Size of a second level page table}}$$
$$= \frac{2^{36}}{\text{Number of entries in a single second level page table} \times \text{ Size of an entry}}$$
$$= \frac{2^{36}}{2^9 \times 4} \because \text{(bits 21-29 gives 9 bits)}$$
$$= \frac{2^{36}}{2^{11}}$$
$$= 2^{25}$$

So, we need $25$ bits for addressing the second level page tables as well.

So, answer is (**D**).

Video Explanation for Multi-level Paging: https://youtu.be/bArypfVmPb8

(Edit:-
There is nothing to edit for such awesome explanation but just adding one of my comment if it is useful - comment. However if anyone finds something to add (or correct) then feel free to do that in my comment.)

answered Nov 14, 2014 • edited Jul 13, 2018 by **kenzou**

edit flag hide comment Unfollow
Pip Box Delete with Reason Wrong Useful


Arjun

# Why Multi-level paging ?

- We may not find continuous space to store page table in main memory.

- We have seen most of the pages are empty.

  Why to store entries corresponding to the pages which are empty ?

# But where are the savings ?

It seems we have added one more page table and not saving any space.

Where are savings that we promised earlier ?