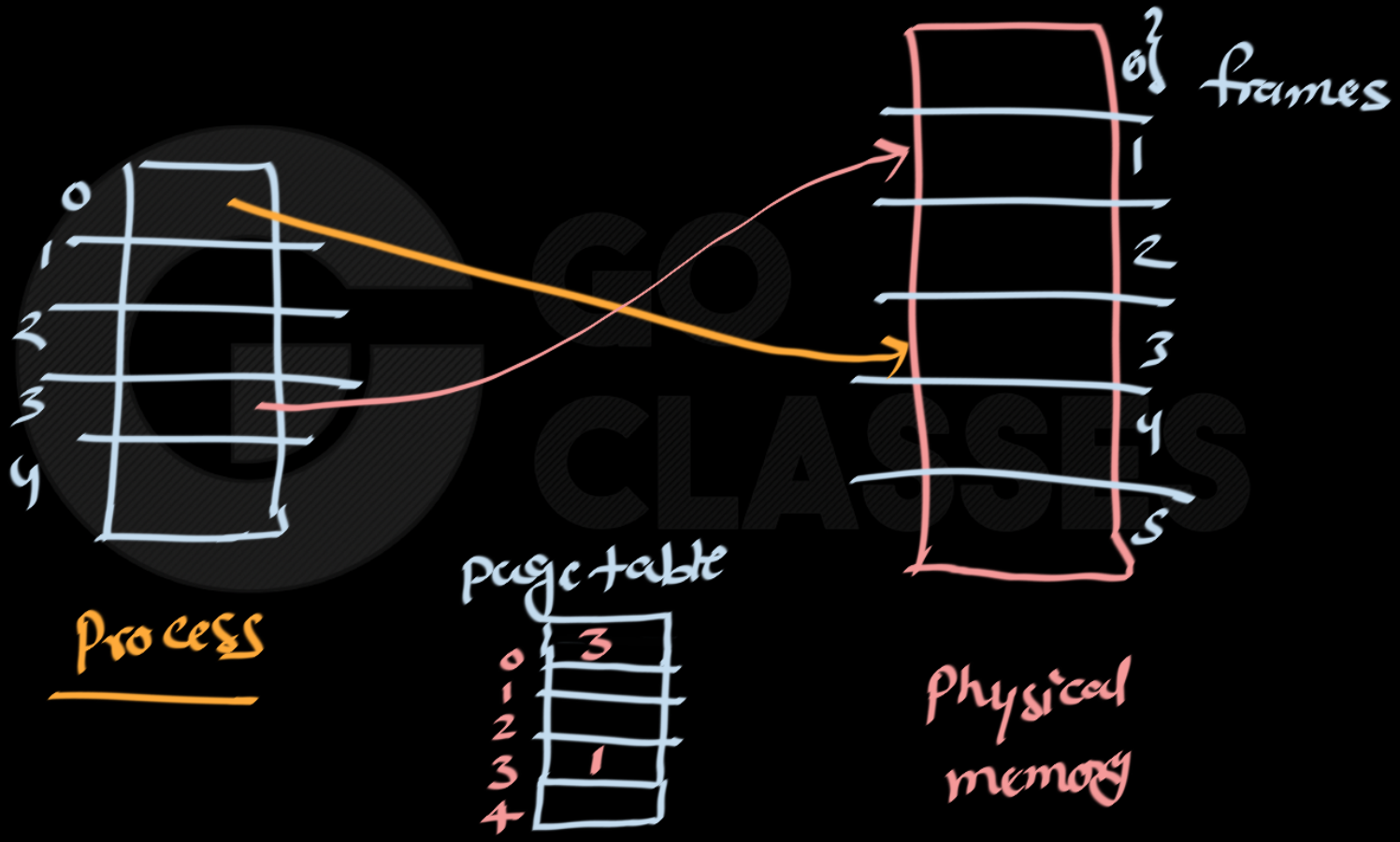
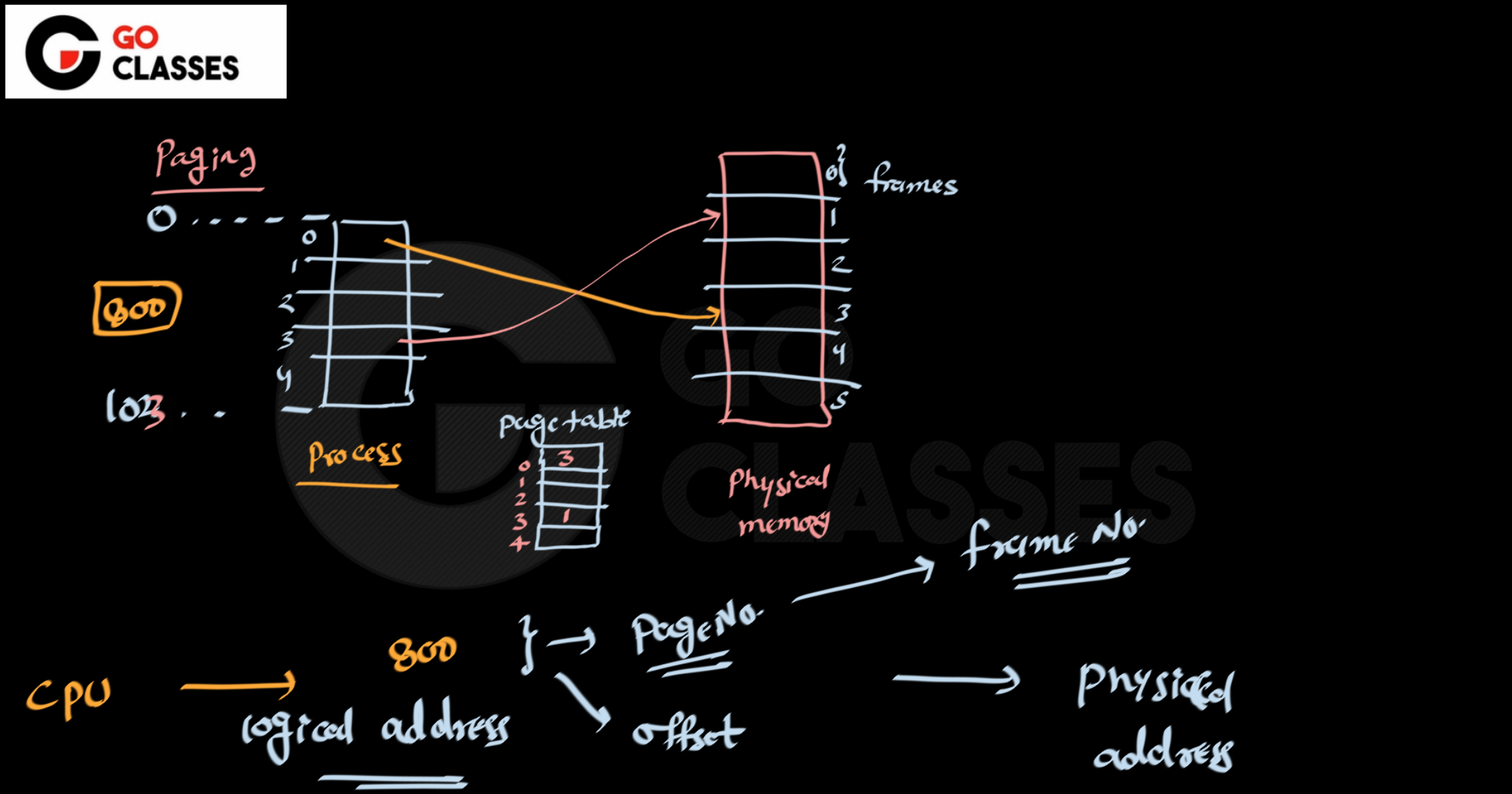
Lecture: 26

GO CLASSES

Paging



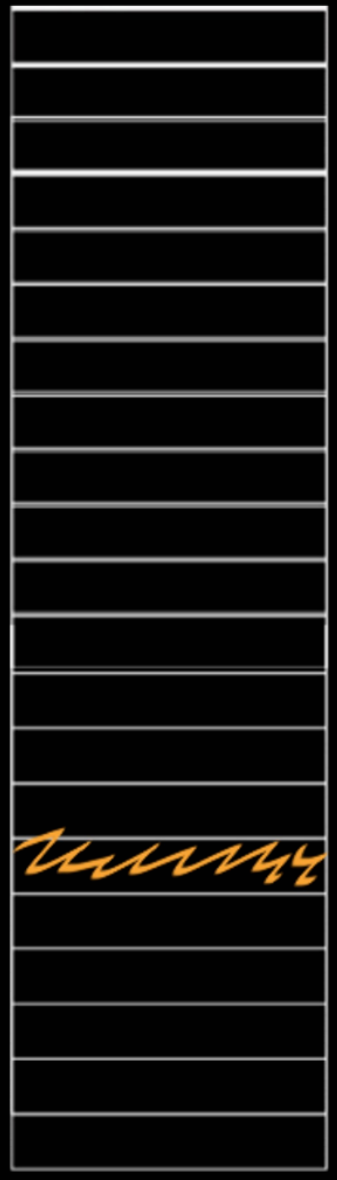

Paging
0 ....
900
1023 . .
0
1
2
3
4
Process
page table
0 3
1
2
3 1
4
Physical memory
0
1
2
3
4
5
frames
CPU
900
logical address
Page No.
offset
frame No.
Physical address

Each page contains = 8 Bytes
800 will be in 100th page number

Physical address of 803 ?

Pg No	Frame No
0	
1	
100	5
$2^7 - 1$	

2^7 Entries

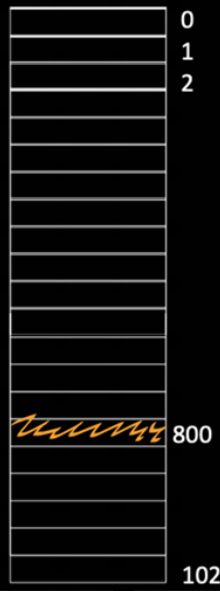


Each page contains = 8 Bytes
800 will be in 100th page number

Physical address of 803 ?

Pg No	Frame No
0	
1	
100	5
$2^7 - 1$	

2^7 Entries



$$\frac{803}{2^3} = 100.xx$$

2^3 → Page size

100 → Page No. ⇒ frame No. = 5

$$803 \,\%\, 2^3 = 3$$

Each page contains = 8 Bytes
800 will be in 100th page number

Physical address of 803 ?

Pg No	Frame No
0	
1	
100	5
$2^7 - 1$	

2^7 Entries

$$\frac{803}{2^3} = 100.xx$$

2^3 → Page size

Page No. ⇒ frame No. = 5

$803 \,\%\, 2^3 = 3$

Physical address = $8 \times 5 + 3 = 43$ ≈ physical addrs

$803 = 100 \times 2^3 + 3$

Page size

reminder

we will do division in binary

(Reason: because page size is in power of 2)

$803 = 100 \times 2^3 + 3$

100×2^3 → Page size

3 → reminder

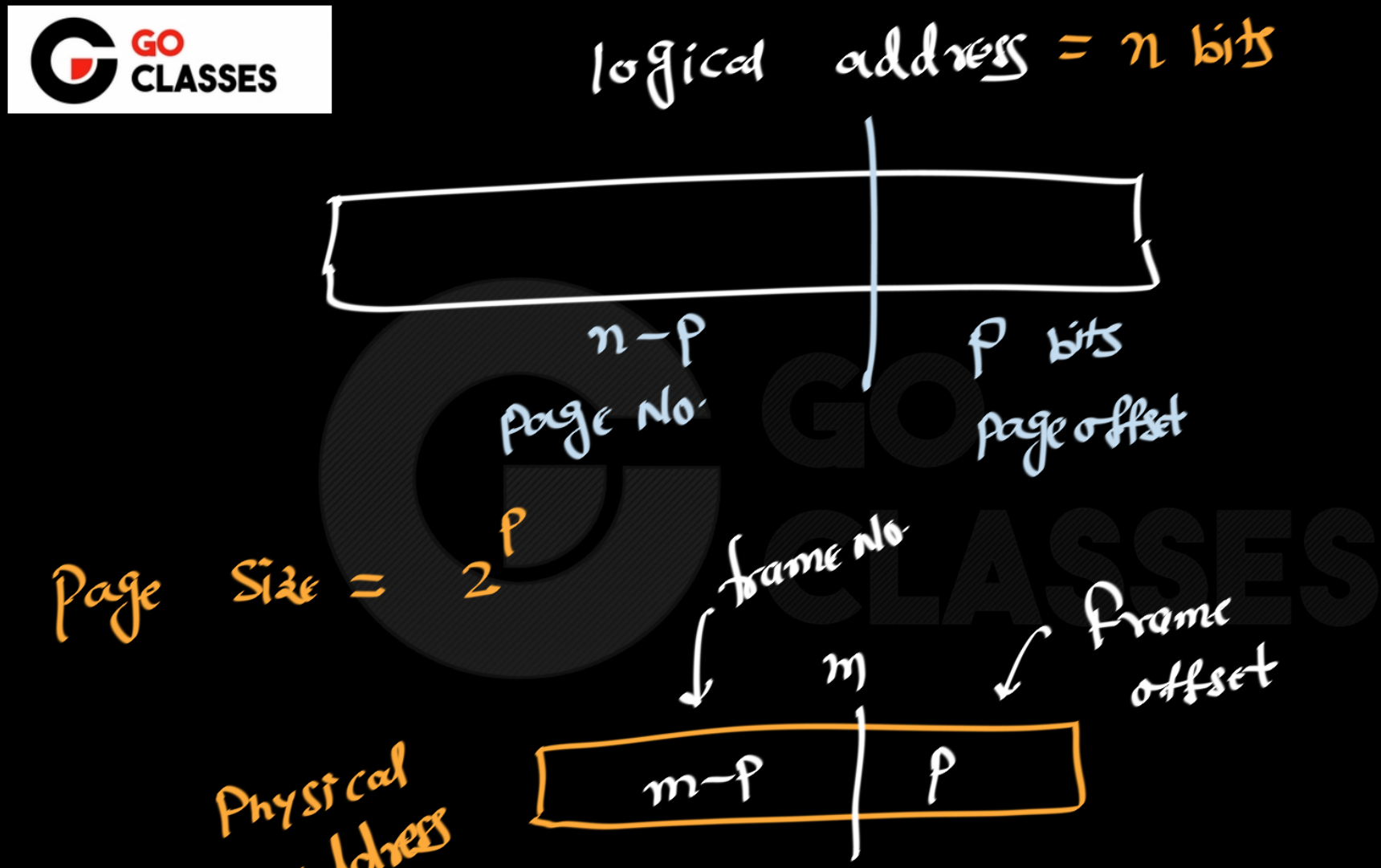
we will do division in binary

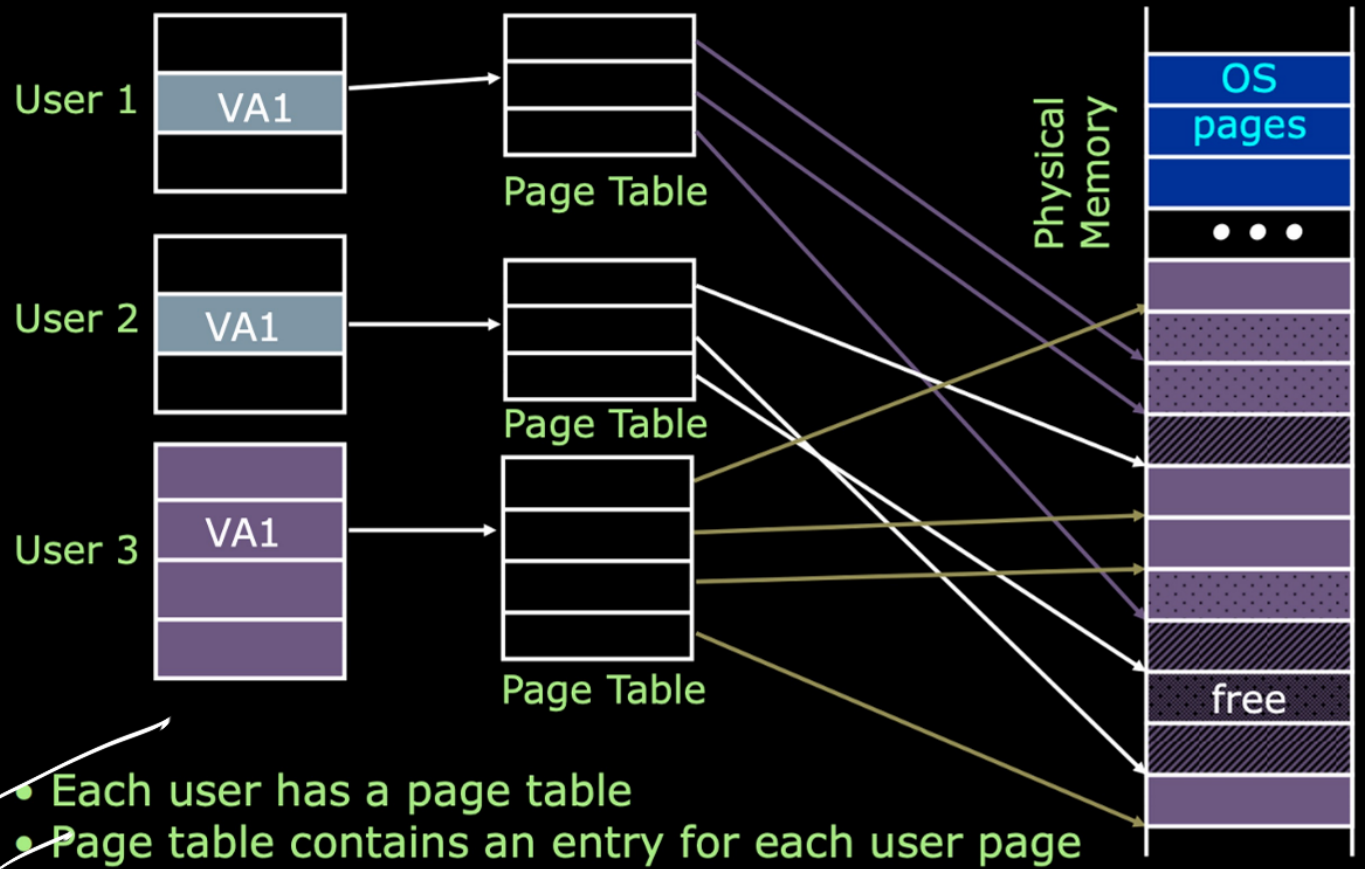
(Reason: because page size is in power of 2)

$803 \equiv$ 1100100 | 011

1100100 → Page No.

011 → offset


logical address = n bits
n-p
Page No.
p bits
page offset
Page Size = 2^p
frame no
m
frame offset
Physical address
m-p
p

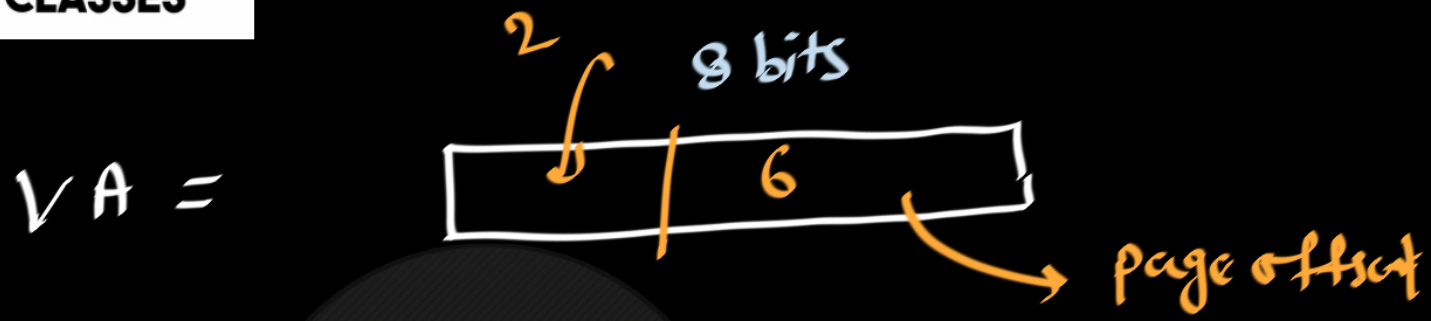


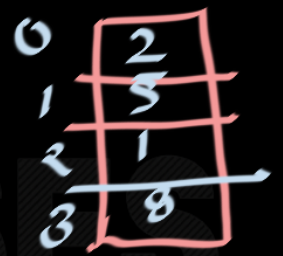
- Each user has a page table
- Page table contains an entry for each user page

http://csg.csail.mit.edu/6.823S15/lectures/L07.pdf

Page translation exercise

- 8-bit virtual address, 10-bit physical address, and each page is 64 bytes
  - How many virtual pages?
  - How many physical pages?
  - How many entries in page table?
  - Given page table = [2, 5, 1, 8], what's the physical address for virtual address 241?





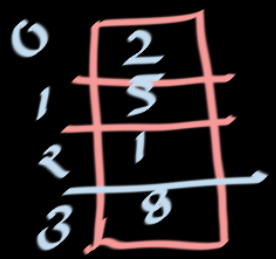
Page Size = 64 B = 2^6 B

One entry for each page

- How many virtual pages? → $2^2 = 4$
- How many physical pages? → $2^4 = 16$
- How many entries in page table? = 4
- Given page table = [2, 5, 1, 8], what's the physical address for virtual address 241?

GO CLASSES

VA 241 ⟹ PA ?



Decimal : $\frac{241}{64} = 3$

$241 = 3 \times 64 + 49$

3 → Page No.

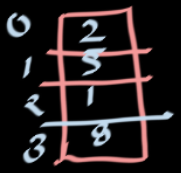
49 → offset

3 → 8

$8 \times 64 + 49 = 561$ → Physical address

GO CLASSES

VA 241 ⟹ PA ?



Decimal : $\frac{241}{64} = 3$

$241 = 3 \times 64 + 49$

3 → Page No. ; 49 → offset

$8 \times 64 + 49 = 561$ → physical address

Binary : $241 = 11 \mid 110001$

11 → Page No. ; 110001 → Page offset

Page No. = 3 ⟹ frame no. ⇒ 8

Binary : 241 = 11 | 110001

Page No

Page offset

Page No. = 3 ⟹ frame No. ⟹ 8

Physical address → 1000 | 110001 ⇒ 561

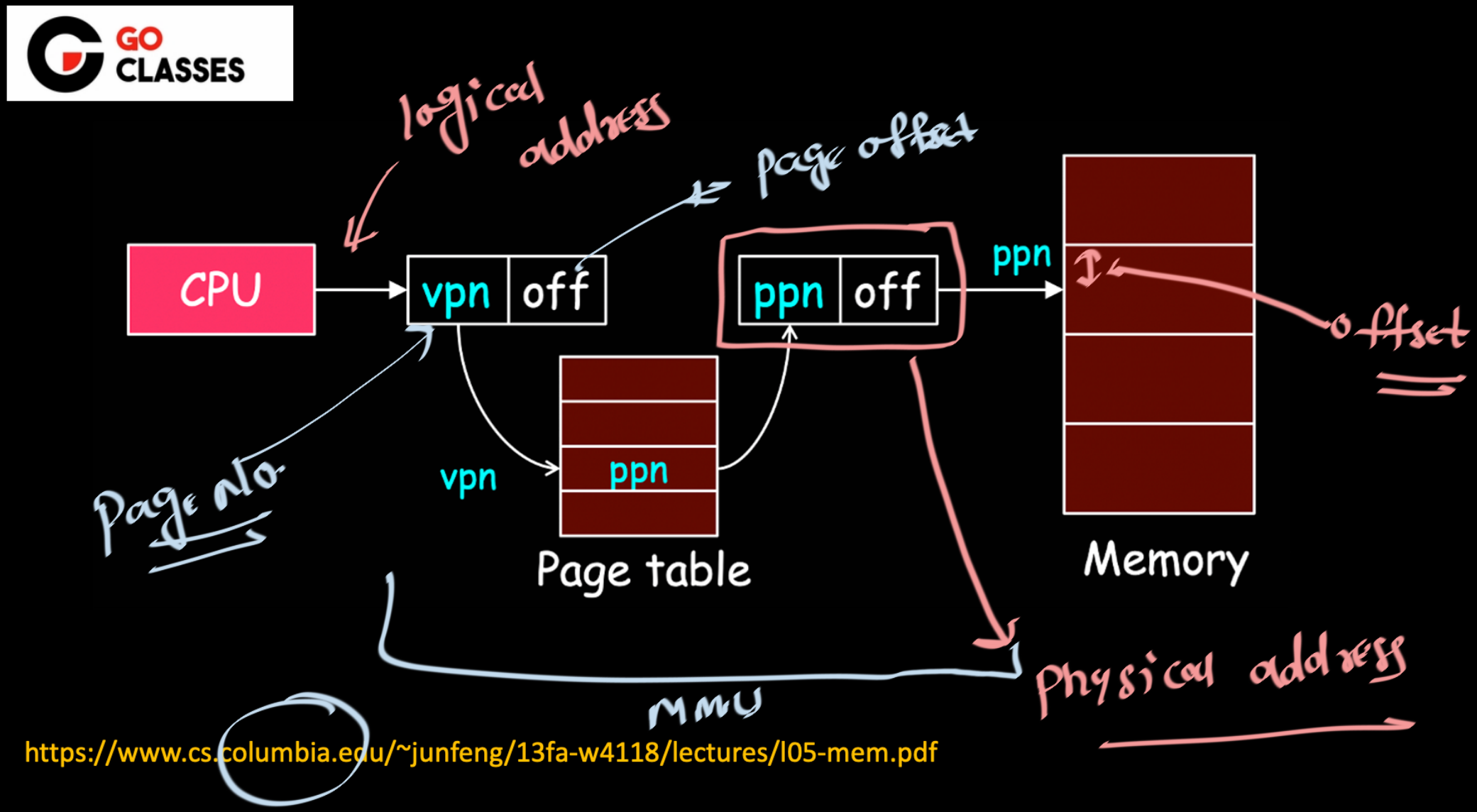
1000 → frame No.

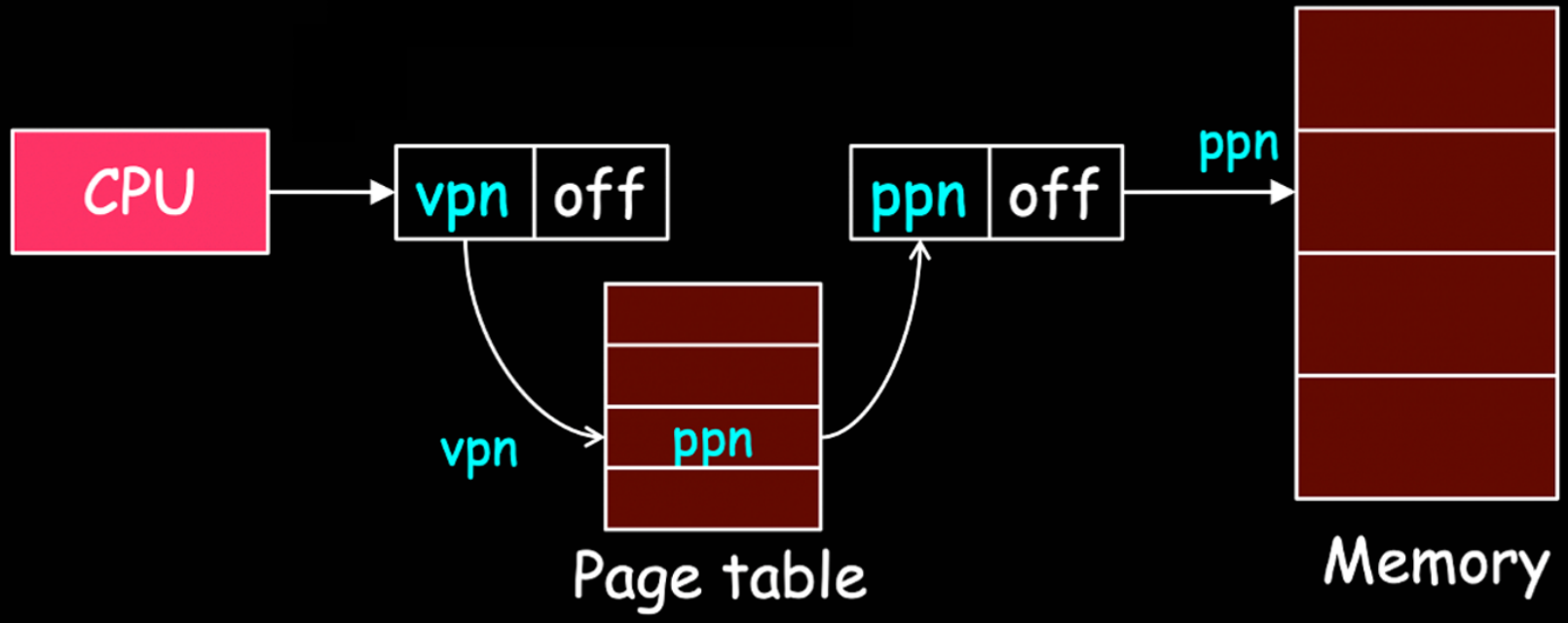
110001 → frame offset

GO CLASSES

Address translation in the hardware (MMU) for flat page table

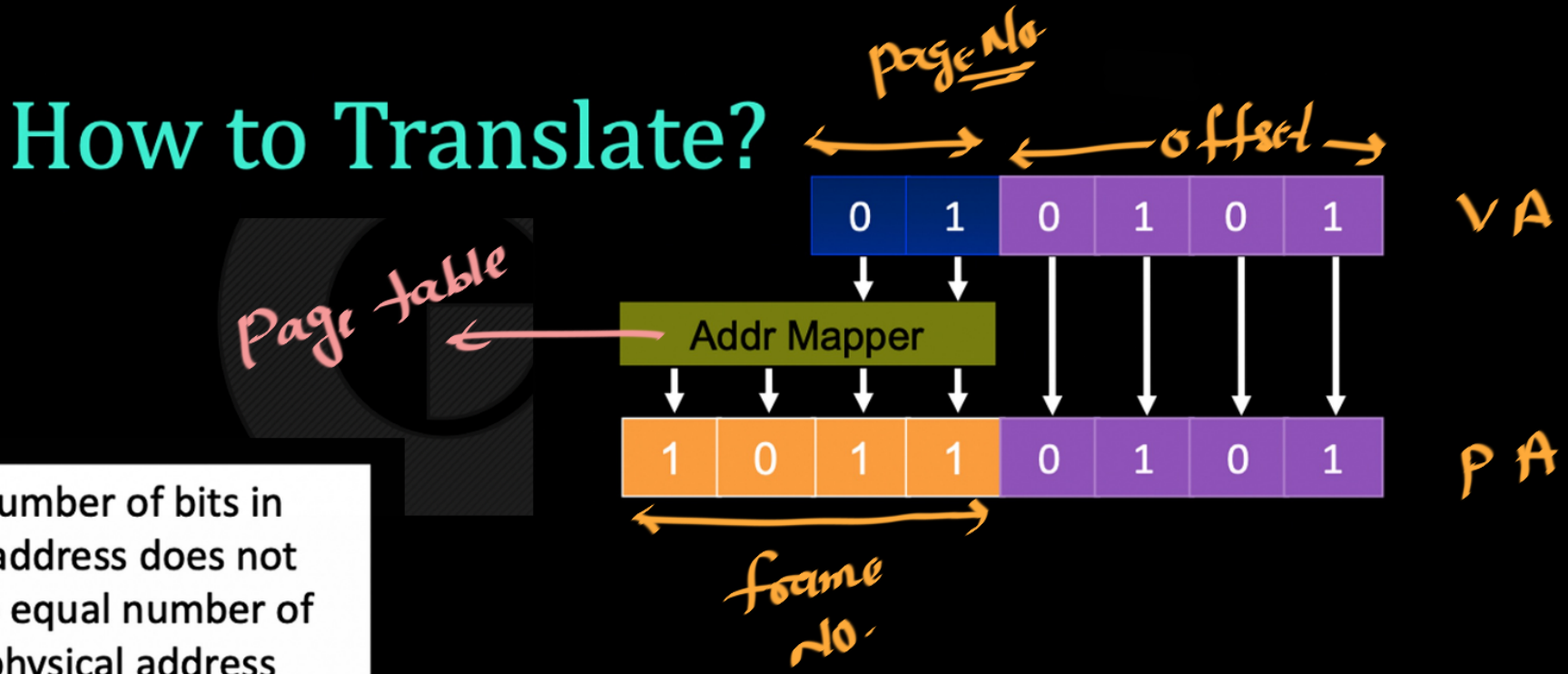
Memory management unit


GO CLASSES
logical address
page offset
CPU
vpn
off
ppn
off
ppn
offset
Page no
vpn
ppn
Page table
Memory
MMU
Physical address
https://www.cs.columbia.edu/~junfeng/13fa-w4118/lectures/l05-mem.pdf



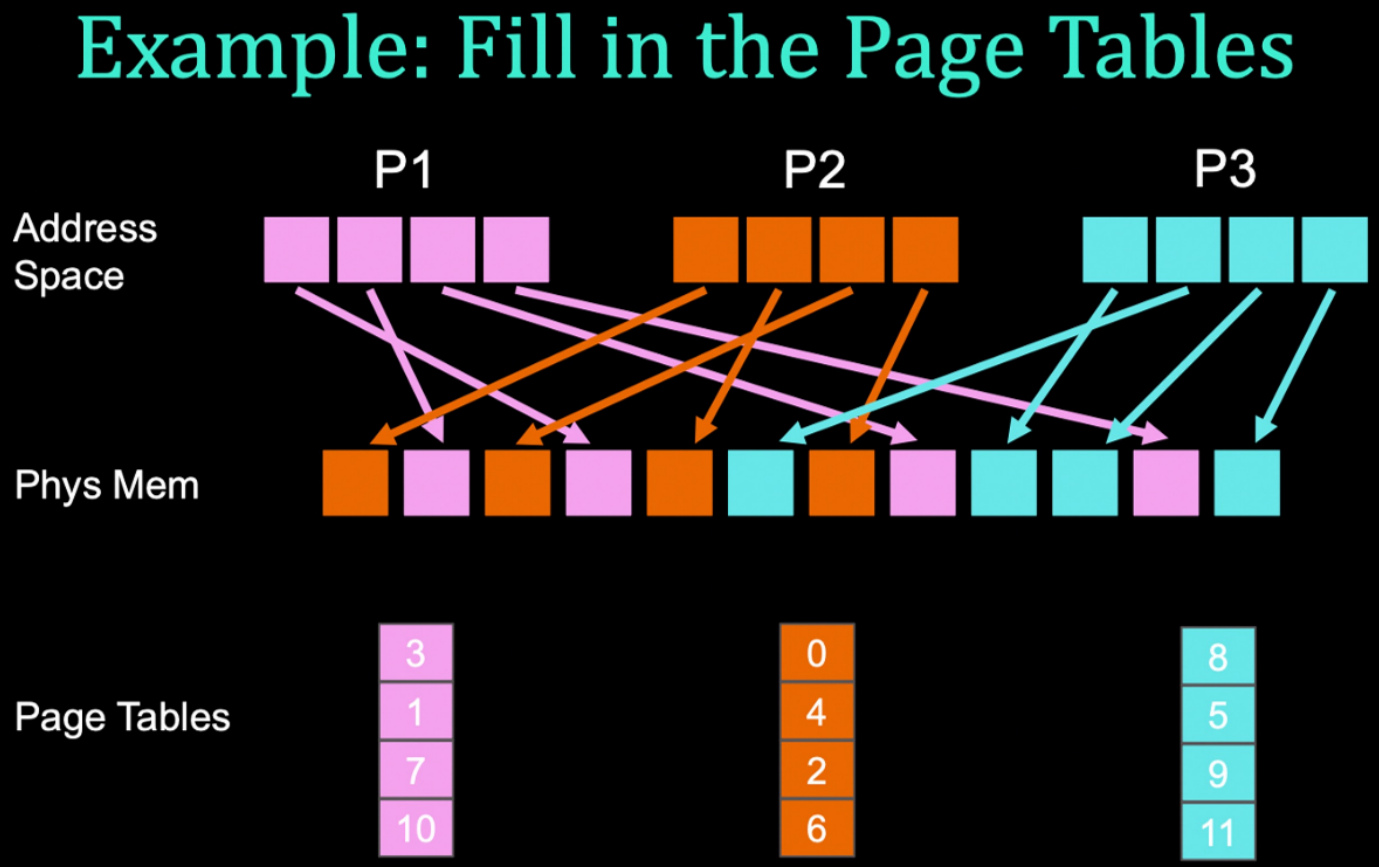
https://www.cs.columbia.edu/~junfeng/13fa-w4118/lectures/l05-mem.pdf

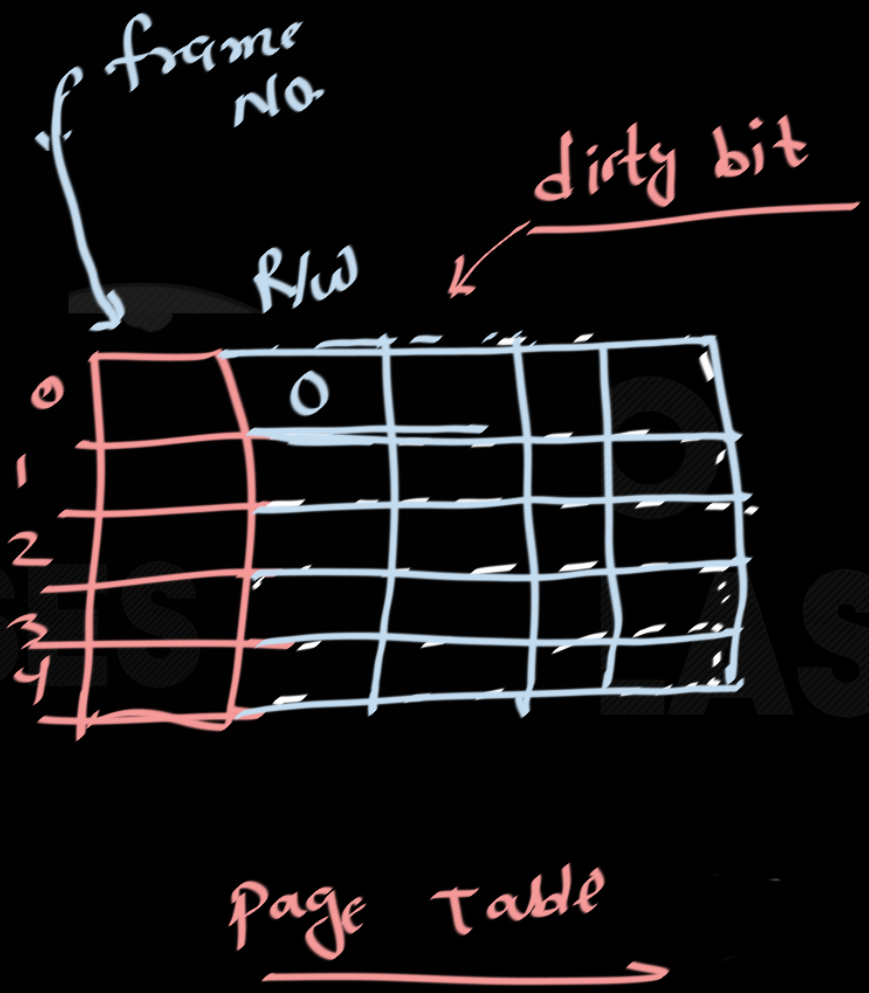
Optional Image



Note: number of bits in virtual address does not need to equal number of bits in physical address

Optional Image




frame No.
dirty bit
R/W
0
0
1
2
3
4
Page Table

GO CLASSES

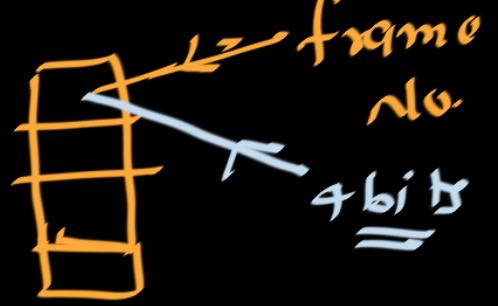
PA = | 4 | 6 |

LA = | 2 | 6 |

No. of entries in page table = $2^2 = 4$

Size of the page table :-

Size of one entry × No. of entries



GO CLASSES

Example -1

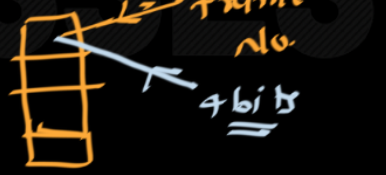
PA = | 4 | 6 |

LA = | 2 | 6 |

No. of entries in page table = $2^2 = 4$

Size of the page table :-

Size of one entry x No. of entries



4 x 4

= 16 bits = 2B

Example -2

PA = | 4 | 6 |

LA = | 2 | 6 |

No. of entries in page table = $2^2 = 4$

Size of the page table :-

Size of one entry × No. of entries

4 × 4

= 16 bits = 2B

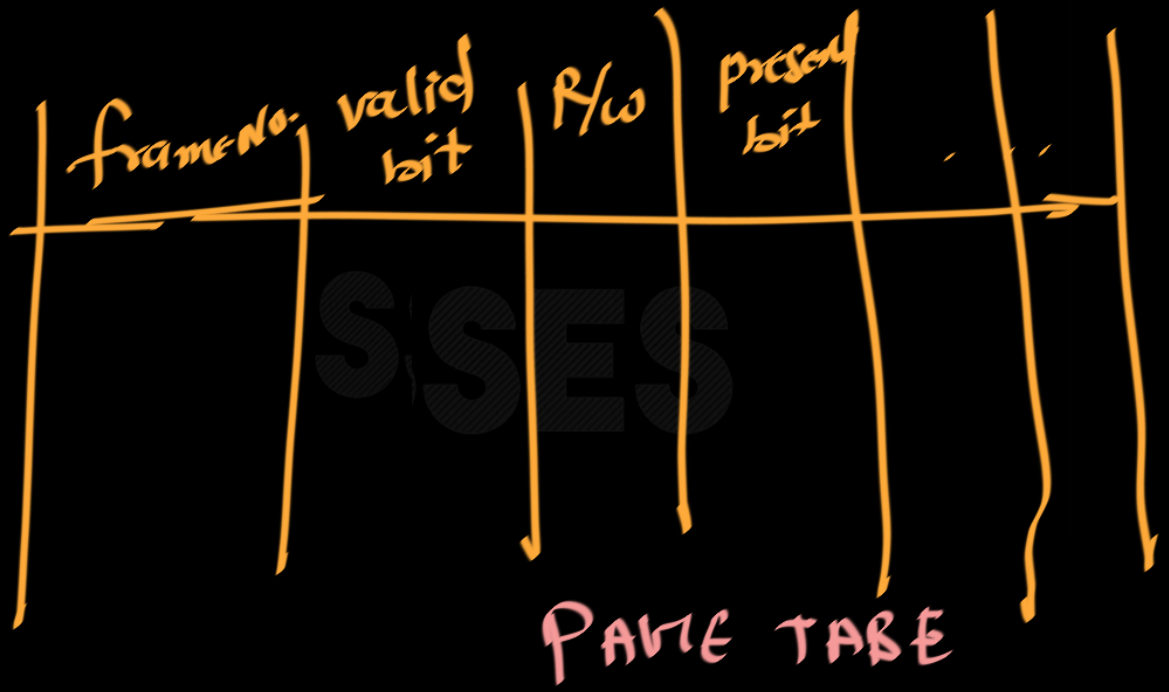
(PTE)

Page table entry size = 2 bytes (given)

page table Size = 2B × 4 = 8 bytes.

Other PTE Info

- What other info is in PTE besides PFN?
  - **Valid** bit
  - **Protection** bit
  - **Present** bit (needed later)
  - **Referenced** bit (needed later)
  - **Dirty** bit (needed later)



https://compas.cs.stonybrook.edu/~nhonarmand/courses/fa17/cse306/slides/06-paging.pdf

Common Flags Of Page Table Entry

- Valid Bit: Indicating whether the particular translation is valid.
- Protection Bit: Indicating whether the page could be read from, written to, or executed from
- Present Bit: Indicating whether this page is in physical memory or on disk(swapped out)
- Dirty Bit: Indicating whether the page has been modified since it was brought into memory
- Reference Bit(Accessed Bit): Indicating that a page has been accessed

Complete Page Table Entry (PTE)

Valid	Protection R/W/X	Ref	Dirty	Frame number

Synonyms:

- Valid bit == Present bit
- Dirty bit == Modified bit
- Referenced bit == Accessed bit

Question

- Logical address = 256KB
- Physical address = 64KB
- Page size = 4 KB

Calculate:

- Number of pages
- Number of bits in page offset
- Number of frames
- If each page table entry size is 2^p Bytes then size of page table

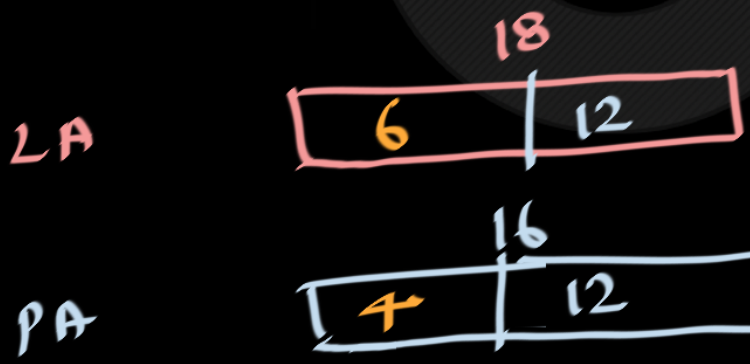
Question

- Logical address = 256KB → $2^8 \times 2^{10}$ B = 2^{18} B
- Physical address = 64KB → $2^6 \times 2^{10}$ B = 2^{16} B
- Page size = 4 KB → $2^2 \times 2^{10} = 2^{12}$ B

Calculate:

- Number of pages
- Number of bits in page offset → 12
- Number of frames → 24
- If each page table entry size is 2^p Bytes then size of page table → $2^6 \times 2^p = 2^{6+p}$

No. of pages = 2^6 = 64



Question

- Physical address = 128MB
- Page size = 4 KB
- Number of pages = 2^6

Calculate:

- Number of frames
- Page offset bits
- Logical address space size
- If each page table entry size is 2^p Bytes then size of page table

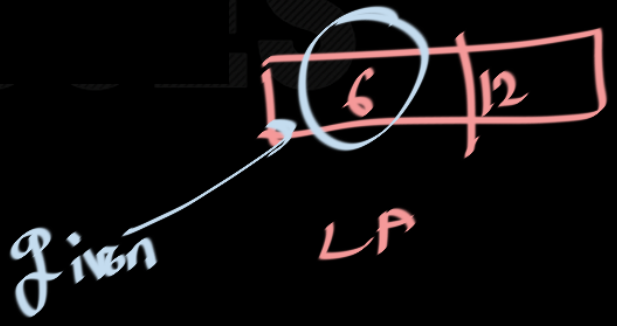
Question

- Physical address = 128MB → 2^{27} B
- Page size = 4 KB → 2^{12} B
- Number of pages = 2^6

Calculate:

- Number of frames → 2^{15}
- Page offset bits → 12
- Logical address space size → 2^{18}
- If each page table entry size is 2^p Bytes then size of page table

$2^6 \times 2^p = 2^{6+p}$



Question

- Logical address = 1GB
- Page size = 4 MB
- Number of frames = 2^{10}

Calculate:

- Physical address space size
- Page offset bits
- Logical address space size
- If each page table entry size is 2^p Bytes then size of page table

Question

- Logical address = 1TB
- Number of pages = 2^{12}
- Number of frames = 2^{10}

H.W

Calculate:

- Physical address space size
- Page offset bits
- If each page table entry size is 2^p Bytes then size of page table

Question

Problem 5

Consider the following page table used for address translations:
(for this portion of the question, all numbers are expressed using base 10).

Virtual page	Frame
0	3
1	10
2	9
3	2
4	0

2^{10} B

Assume the page size is 1024 bytes, convert the following virtual addresses into physical addresses (show your calculations):

a. **(2 marks)** 697

b. **(2 marks)** 1054

c. **(2 marks)** 1024

H.W.

d. **(4 marks)** If possible, convert the **physical address** 2075 into a **virtual address**. If not explain why it can not be done.

Virtual page	Frame
0	3
1	10
2	9
3	2
4	0

Page Size = 1024

697 = 0 × 1024 + 697

PageNo. = 0

697 → Page offset

frame No. = 3

PA = 3 × 1024 + 697 = 3769

d. **(4 marks)** If possible, convert the **physical address** 2075 into a **virtual address**. If not explain why it can not be done.

Virtual page	Frame
0	3
1	10
2	9
3	2
4	0

frame size = 1024

SrK to Everyone 8:38 PM

S

so most people don't know the connection between binary and decimal in addresses

Logical address

2075 PA = 2 × 1024 + 27

2 → frame No.

27 → offset

3 × 1024 + 27

= 3099

GO CLASSES

AIMT-1

Virtual page numbers and physical frame numbers are represented in decimal. Consider a machine with 32-bit virtual addresses and a page size of 512 bytes. During a program execution, the TLB contains the following valid entries (all in decimal).

Virtual Page Num	Physical Frame Num
591	100
5912	200
11548	300
2589	400
59125	500

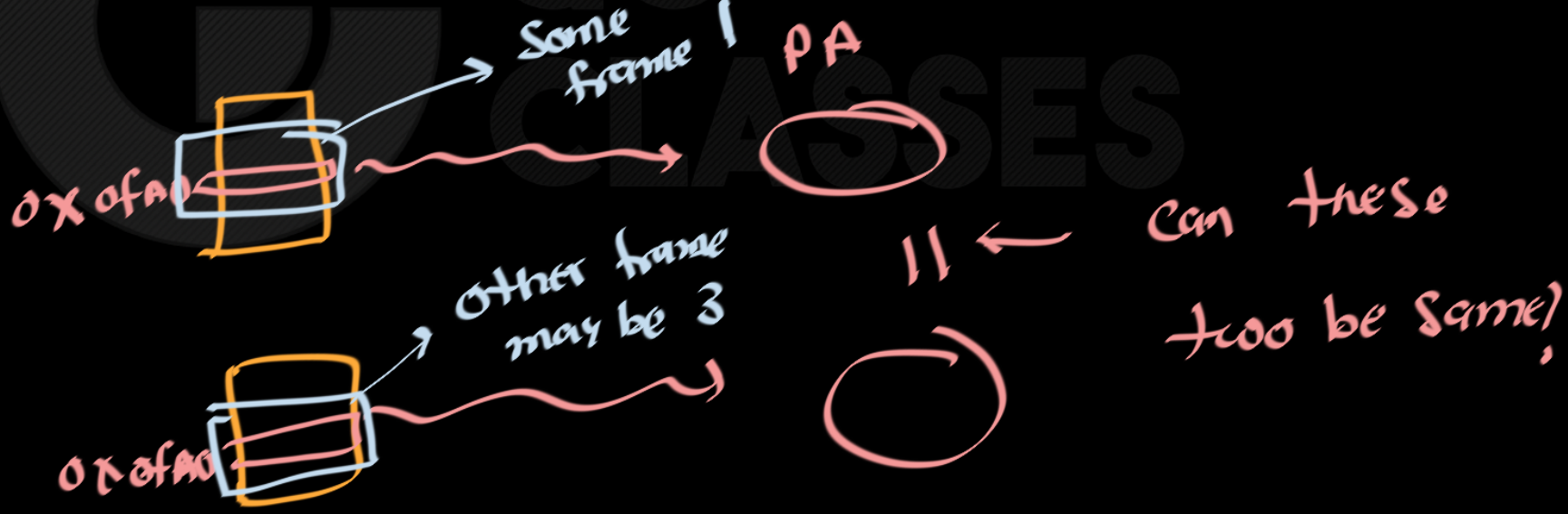
Translate the virtual address = 5912589 (in decimal) into a physical address (in decimal).

Suppose there are multiple processes in Paging System. Suppose no process is sharing any data with other process.

is it possible for same VA of two different Processes we get same frame No.?

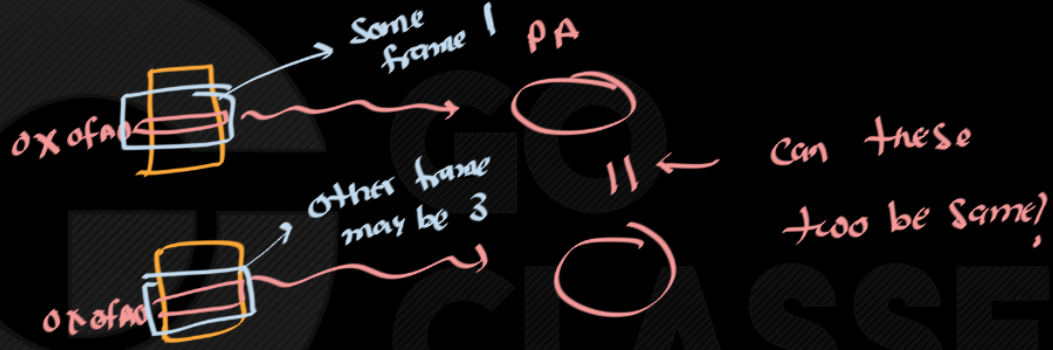
Suppose there are multiple processes in Paging System. Suppose no process is sharing any data with other process.

is it possible for same VA of two different Processes we get same frame No.?



Suppose there are multiple processes in Paging System. Suppose no process is sharing any data with other process.

is it possible for same VA of two different processes we get same frame No?



even though LA is same still PA is different because pages will be mapped to different frames.

a. **(2 marks)** 697

————————begin solution————————

697 = 0 * 1024 + 697 and so vpn = 3.
The real Address is 3 * 1024 + 697 = 3769

b. **(2 marks)** 1054

————————begin solution————————

1054 = 1024 + 30 and so is on page 1.
The real Address is 10*1024 + 30 = 10270;

————————end solution————————

c. **(2 marks)** 1024

————————begin solution————————

```
page 0 is virtual address 0 .. 1023.
So 1024 =  is vpn = 1  offset = 0
The real Address is 10 * 1024 = 10240
```

————————end solution————————

d. **(4 marks)** If possible, convert the **physical address** 2075 into a **virtual address**. If not explain why it can not be done.

————————begin solution————————

```
2075 is on physical frame 2 with offset 27.
From the page table we see that frame 2 is virtual page 3.
Virtual page 3 is 3 * 1024 = 3072 + the offset (27)
So the virtual address = 3099.
```

————————end solution————————

Question

Consider a single-level paging system with 12-bit virtual addresses, 24-bit physical addresses, and a 256 (2^8) byte page size.

What is the maximum number of entries in a page table in this system?

Solution

Consider a single-level paging system with 12-bit virtual addresses, 24-bit physical addresses, and a 256 (2^8) byte page size.

What is the maximum number of entries in a page table in this system?

$2^4 = 16$

Question

Consider a virtual memory system that uses paging. Virtual and physical addresses are both 32 bits long, and the page size is 4KB = 2^{12} bytes. A process P_1 has the following page table. Frame numbers are given in notation (recall that each hexadecimal digit represents 4 bits).

	Frame Number
0	0x00788
1	0x00249
2	0x0023f
3	0x00ace
4	0x00bcd

For each of the following virtual addresses, indicate the physical address to which it maps. If the virtual address is not part of the address space of P_1, write `NO TRANSLATION` instead. Use hexadecimal notation for the physical addresses.

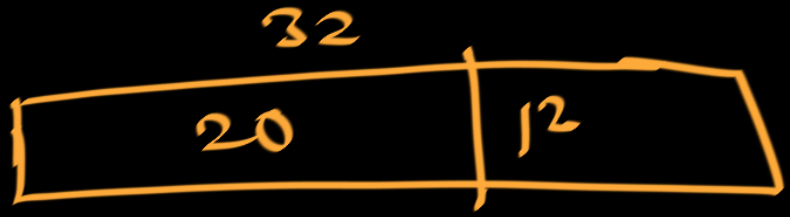
- `0x00001a60`
- `0x000051ff`
- `0x00000fb5`

https://student.cs.uwaterloo.ca/~cs350/common/old-exams/W11-midterm-sol.pdf

Question

Consider a virtual memory system that uses paging. Virtual and physical addresses are both 32 bits long, and the page size is 4KB = 2^{12} bytes. A process P_1 has the following page table. Frame numbers are given in notation (recall that each hexadecimal digit represents 4 bits).

	Frame Number
0	0x00788
1	0x00249
2	0x0023f
3	0x00ace
4	0x00bcd



For each of the following virtual addresses, indicate the physical address to which it maps. If the virtual address is not part of the address space of P_1, write `NO TRANSLATION` instead. Use hexadecimal notation for the physical addresses.

- 0x00001a60
- 0x000051ff
- 0x00000fb5
- 0x00005 1ff

0x00001 a60 → 0x00249 a60 } PA

- 0x00001a60

  0x00249a60

- 0x000051ff

  NO TRANSLATION

- 0x00000fb5

  0x00788fb5

Question

Consider a virtual memory system that uses paging. Virtual and physical addresses are both 32 bits long, and the page size is $4\text{KB} = 2^{12}$ bytes. A process P_1 has the following page table. Frame numbers are given in notation (recall that each hexadecimal digit represents 4 bits).

	Frame Number
0	0x00788
1	0x00249
2	0x0023f
3	0x00ace
4	0x00bcd

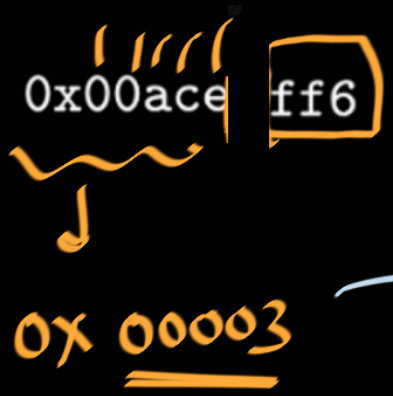
For each of the following physical addresses, indicate the virtual address that maps to it. If the physical address is not part of the physical memory assigned to P_1, write `NO TRANSLATION` instead. Use hexadecimal notation for the virtual addresses.

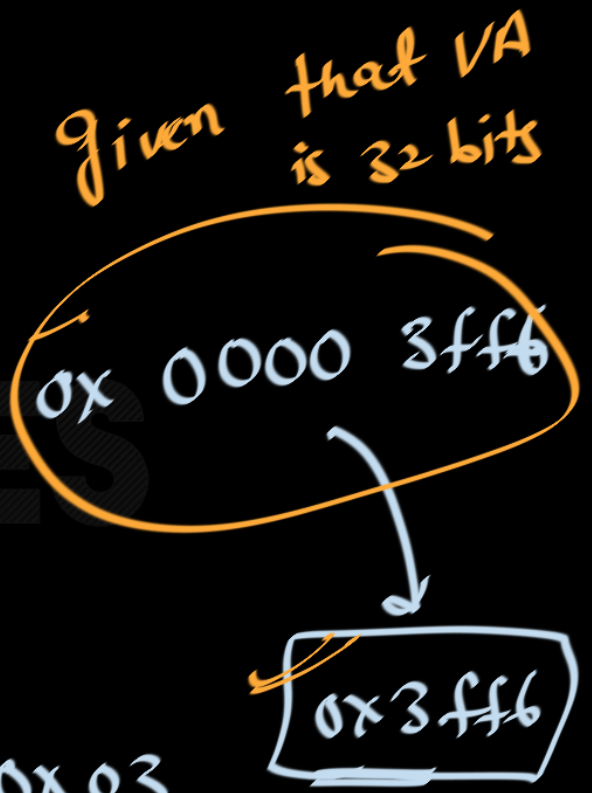
- `0x00aceff6`
- `0x00249000`
- `0x00000887`

	Frame Number
0	0x00788
1	0x00249
2	0x0023f
3	0x00ace
4	0x00bcd

For each of the following physical addresses, indicate the virtual address that maps to i physical address is not part of the physical memory assigned to P_1, write `NO TRANSLATION` Use hexadecimal notation for the virtual addresses.

- `0x00aceff6`
- `0x00249000`
- `0x00000887`





- 0x00aceff6

  0x00003ff6

- 0x00249000

  0x00001000

- 0x00000887

  NO TRANSLATION

Question

4.

Virtual Address	Physical Address
0x0008 0AF0	0x0010 1AF0
0x0000 2224	0x0008 3224
0x0007 3234	0x0018 3234

Suppose that while a process P is running, it uses the virtual addresses shown in the left column of the table above. For each of these virtual addresses, the corresponding physical address (after address translation) is also shown in the table. On the machine on which P is running, both virtual addresses and physical addresses are 32 bits long.

a. (3 marks)

Is it possible that the MMU used *paging*, with a page size of 64 KB (2^{16} bytes), to translate P's virtual addresses to physical addresses? If so, write "YES". If not, write "NO" and explain, briefly and clearly, how you know that paging with this page size was not used.

b. (2 marks)

Is it possible that the MMU used paging, with a page size of 4 KB (2^{12} bytes), to translate P's virtual addresses to physical addresses? If so, write "YES". If not, write "NO" and explain, briefly and clearly, how you know that paging with this page size was not used.

a. (3 marks)

Is it possible that the MMU used *paging*, with a page size of 64 KB (2^{16} bytes), to translate P's virtual addresses to physical addresses? If so, write "YES". If not, write "NO" and explain, briefly and clearly, how you know that paging with this page size was not used.

NO. If page sizes of 64 KB are used, virtual and physical addresses should have the same 16-bit offset. However, `0x0000 2224` and `0x0008 3224` have different offsets.

b. (2 marks)

Is it possible that the MMU used paging, with a page size of 4 KB (2^{12} bytes), to translate P's virtual addresses to physical addresses? If so, write "YES". If not, write "NO" and explain, briefly and clearly, how you know that paging with this page size was not used.

YES.

Question

14. A system uses paging to implement virtual memory. Specifically, it uses a simple linear page table. The virtual address space is of size 1 GB (30 bits); the page size is 1 KB; each page table entry holds only a valid bit and the resulting page-frame number (PFN); the system has a maximum of 2^{15} physical pages (32 MB of physical memory can be addressed at most).

How much memory is used for page tables, when there are 100 processes running in the system?

https://pages.cs.wisc.edu/~remzi/Classes/537/Fall2013/OldExams/11-fall-mid.pdf

Question

14. A system uses paging to implement virtual memory. Specifically, it uses a simple linear page table. The virtual address space is of size 1 GB (30 bits); the page size is 1 KB; each page table entry holds only a valid bit and the resulting page-frame number (PFN); the system has a maximum of 2^{15} physical pages (32 MB of physical memory can be addressed at most).

How much memory is used for page tables, when there are 100 processes running in the system?

Page bits = 2^{10}

VA = 30 bits

PTE = frame no | Valid bit

$2^{20} \times (15+1)$ bits

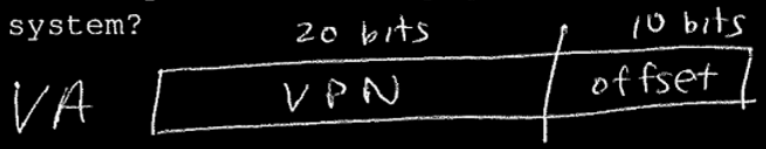
PA:

$= 2^{20} \times 2B$

2^{20} = Size of page table

14. A system uses paging to implement virtual memory. Specifically, it uses a simple linear page table. The virtual address space is of size 1 GB (30 bits); the page size is 1 KB; each page table entry holds only a valid bit and the resulting page-frame number (PFN); the system has a maximum of 2^{15} physical pages (32 MB of physical memory can be addressed at most).

How much memory is used for page tables, when there are 100 processes running in the system?



$2^{20} \times 2B$

$= 2MB \times 100 = 200\ MB$

GATE CSE 2001 | Question: 2.21

asked in **Operating System** Sep 15, 2014 • edited Jun 23, 2018 by **Pooja Khatri**

47,556 views

40

Consider a machine with 64 MB physical memory and a 32-bit virtual address space. If the page size s 4 KB, what is the approximate size of the page table?

A. 16 MB
B. 8 MB
C. 2 MB
D. 24 MB

https://gateoverflow.in/739/gate-cse-2001-question-2-21

58

Number of pages $= 2^{32}/4KB = 2^{20}$ as we need to map every possible virtual address.

So, we need 2^{20} entries in the page table. Physical memory being $64\ MB$, a physical address must be 26 bits and a page (of size $4KB$) address needs $26 - 12 = 14$ address bits. So, each page table entry must be at least 14 bits.

So, total size of page table $= 2^{20} \times 14\ bits \approx 2\ MB$ (assuming PTE is 2 bytes)

Best answer

Correct Answer: C

answered Jan 2, 2015 • edited May 22, 2019 by **Naveen Kumar 3**

edit flag hide comment Follow Pip Box

Delete with Reason Wrong Useful

share this

Arjun

GATE CSE 2016 Set 1 | Question: 47

asked in **Operating System** Feb 12, 2016 • edited Feb 17, 2016 by **makhdoom ghaya**

9,802 views

27

Consider a computer system with 40-bit virtual addressing and page size of sixteen kilobytes. If the computer system has a one-level page table per process and each page table entry requires 48 bits, then the size of the per-process page table is _____ megabytes.

gatecse-2016-set1 operating-system virtual-memory easy numerical-answers

https://gateoverflow.in/39690/gate-cse-2016-set-1-question-47

47

$$\text{No. of pages}(N) = 2^{26} = \text{No. of entries in Page Table}$$
$$\text{Page Table Entry Size}(E) = 6 \text{ bytes}$$

$$\text{So, Page Table Size} = n \times e = 2^{26} \times 6 \text{ bytes} = 384 \text{ MB}$$

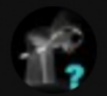
Best answer

answered Feb 19, 2016 • edited Jun 26, 2018 by **Arjun**

edit flag hide comment Follow Pip Box

Delete with Reason Wrong Useful

share this



G VENKATESWARLU

GATE CSE 2015 Set 2 | Question: 47

asked in **Operating System** Feb 13, 2015 • edited Jun 19, 2021 by **Lakshman Patel RJIT**

13,805 views

56

A computer system implements 8 kilobyte pages and a 32-bit physical address space. Each page table entry contains a valid bit, a dirty bit, three permission bits, and the translation. If the maximum size of the page table of a process is 24 megabytes, the length of the virtual address supported by the system is _____ bits.

gatecse-2015-set2 | operating-system | virtual-memory | normal | numerical-answers

https://gateoverflow.in/8247/gate-cse-2015-set-2-question-47

83

Best answer

8 KB pages means 13 offset bits.

For 32 bit physical address, $32 - 13 = 19$ page frame bits must be there in each PTE (Page Table Entry).

We also have 1 valid bit, 1 dirty bit and 3 permission bits.

So, total size of a PTE (Page Table Entry) $= 19 + 5 = 24$ bits $= 3$ bytes.

Given in question, maximum page table size $= 24$ MB

Page table size $=$ No. of PTEs $\times$ size of an entry

So, no. of PTEs $= 24$ MB$/3$ B $= 8$ M

Virtual address supported $=$ No. of PTEs $*$ Page size (As we need a PTE for each page and assuming single-level paging)

$= 8$ M $* 8$ KB

$= 64$ GB $= 2^{36}$ Bytes

So, length of virtual address supported $= 36$ bits (assuming byte addressing)

answered Feb 14, 2015 • edited Jun 19, 2021 by Lakshman Patel RJIT

edit flag hide comment Follow Pip Box

Delete with Reason Wrong Useful

share this

A Typical System has:

Page size = 4 KB, 32 bit LA

So, $\frac{2^{32}}{4\ KB} = \frac{2^{32}}{2^{12}} = 2^{20}\ pages$

1M

There will be 2^{20} page table entries,
if each page table entry is 4B then page table size = 4MB

Size of page table for 1 process

- One such page table for each process.

GO CLASSES

Example :

```
main()
{
    int a, *p;
    a = 5
    p = malloc(4);
    *p = a;
}
```

in it's entire life this process might be using

10B (code) + 1B (static) + 20B (stack) + 4B (heap)

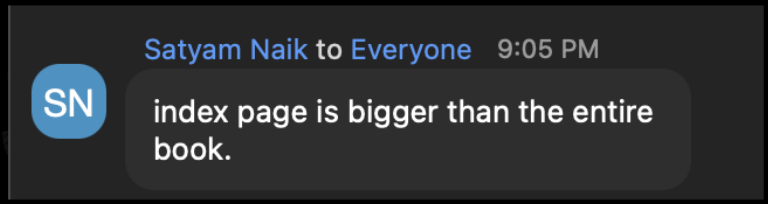
$\simeq$ 35B

Page size = 4 KB, 32 bit LA

So, $\frac{2^{32}}{4\ KB} = \frac{2^{32}}{2^{12}} = 2^{20}\ pages$

Page table size = 4 MB

Process : 35 B



A Typical System has:

Page size = 4 KB, 32 bit LA

So, $\frac{2^{32}}{4\ KB} = \frac{2^{32}}{2^{12}} = 2^{20}\ pages$

There will be 2^{20} page table entries,
if each page table entry is 4B then page size = 4MB

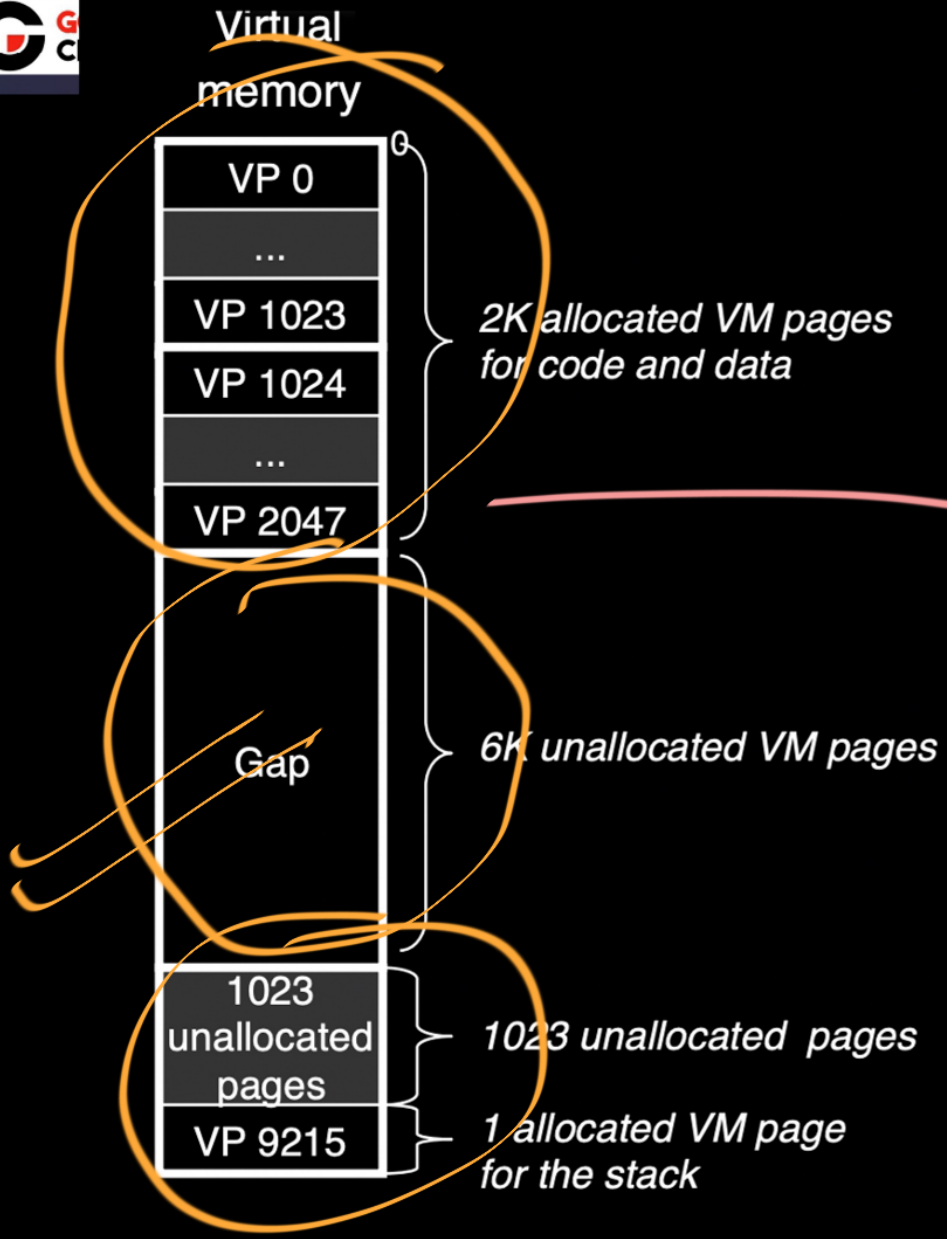
- One such page table for each process.

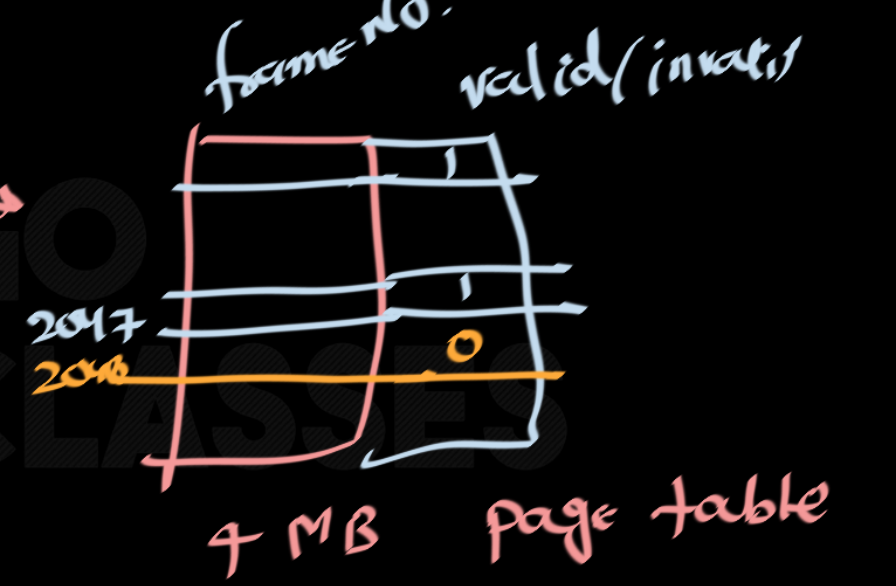
Observation

Usually there is lot of unallocated pages for a process in between heap and stack.

Process just usages few pages.

We can not afford this huge page table for every process


Virtual memory
0
VP 0
...
VP 1023
VP 1024
...
VP 2047
2K allocated VM pages for code and data
Gap
6K unallocated VM pages
1023 unallocated pages
1023 unallocated pages
VP 9215
1 allocated VM page for the stack


frame NO.
valid/invalid
1
1
2047
2046
0
4 MB page table


35B

GO CLASSES

Why are page tables so large? (even though the process is small)

↓

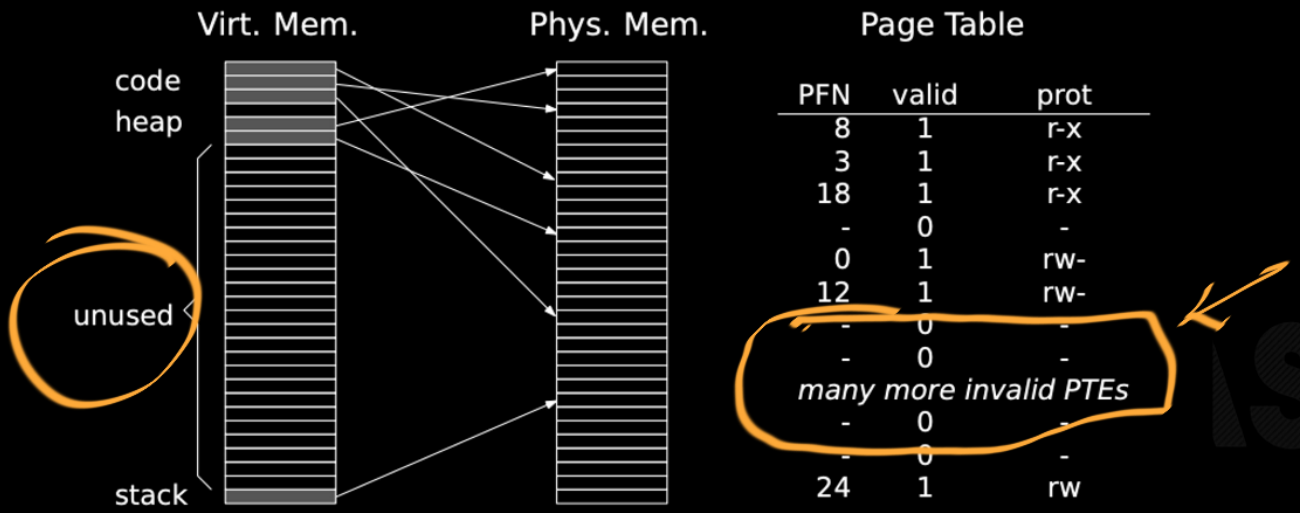
35 B

Page table size = 4MB

Because we have too many invalid pages to maintain.

Why are page tables so large?

Answer Page tables are full of invalid PTEs



Page Table

PFN	valid	prot
8	1	r-x
3	1	r-x
18	1	r-x
-	0	-
0	1	rw-
12	1	rw-
-	0	-
-	0	-
many more invalid PTEs		
-	0	-
-	0	-
24	1	rw

Question How to avoid storing invalid PTE?

https://web.fe.up.pt/~pfs/aulas/so2021/at/24vm_tables.pdf

now we have a challenge:

how to reduce the size of page tables?

the smart solution:

Multi-Level Page Tables

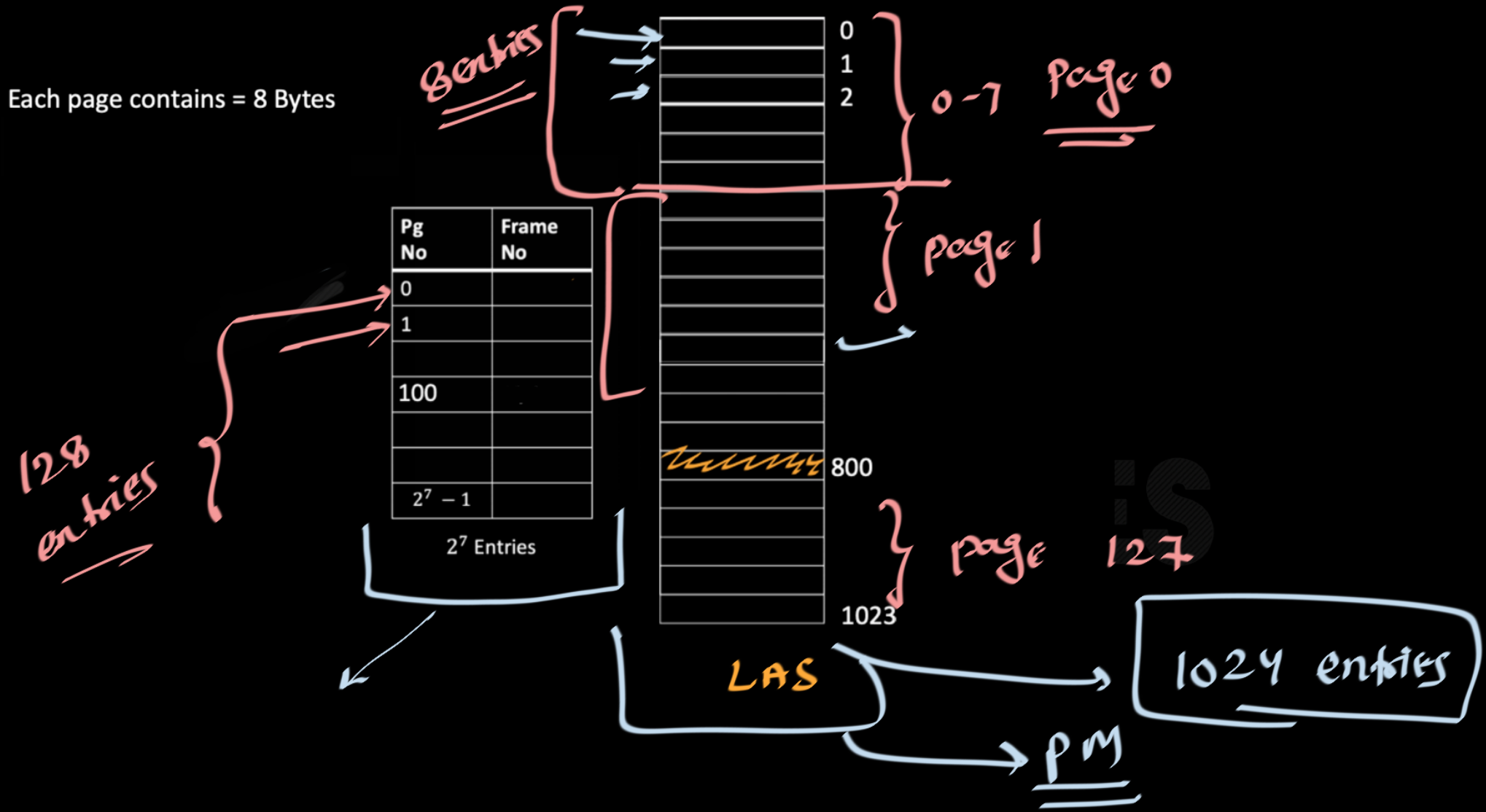
Multi Level Paging

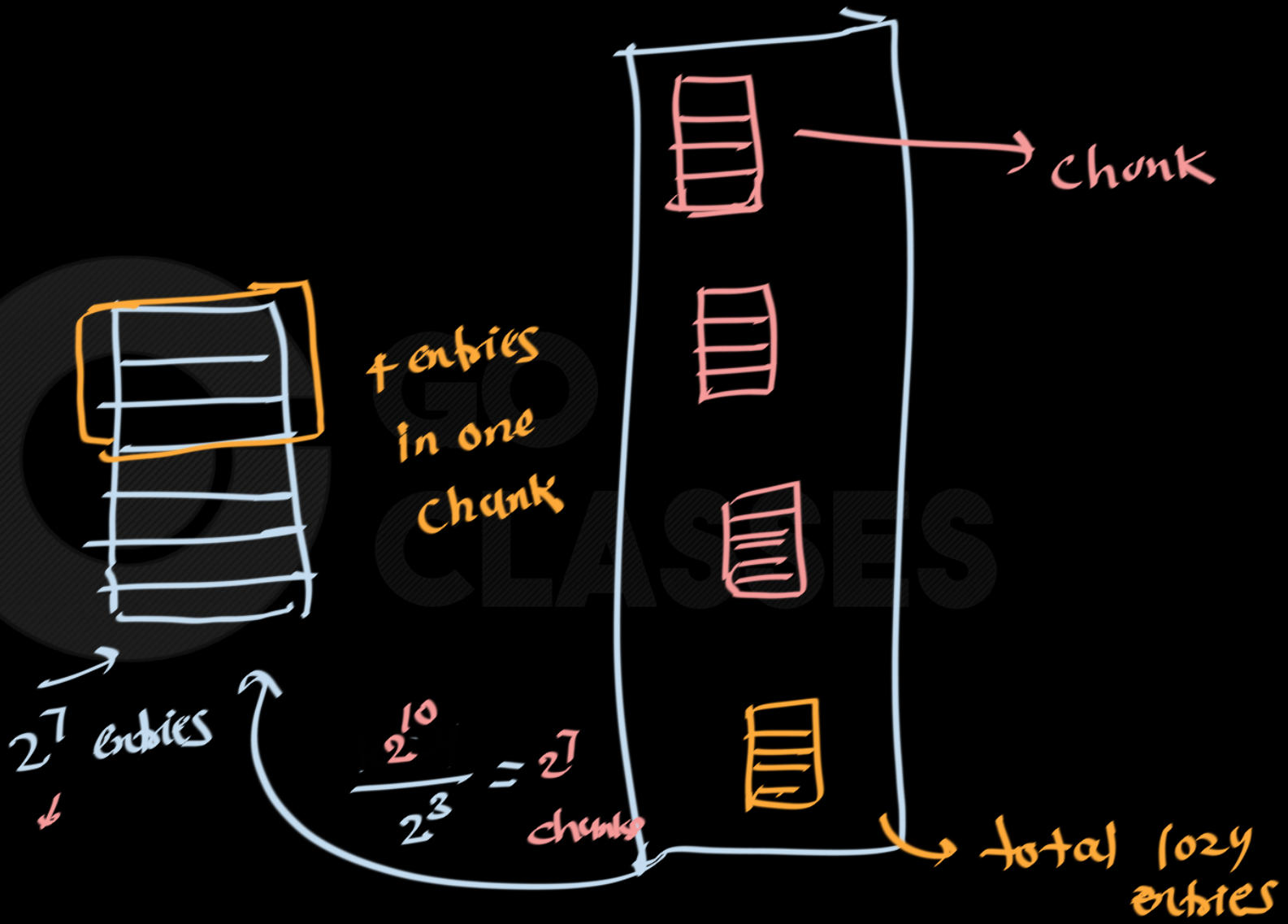
Why Multi-level paging ?

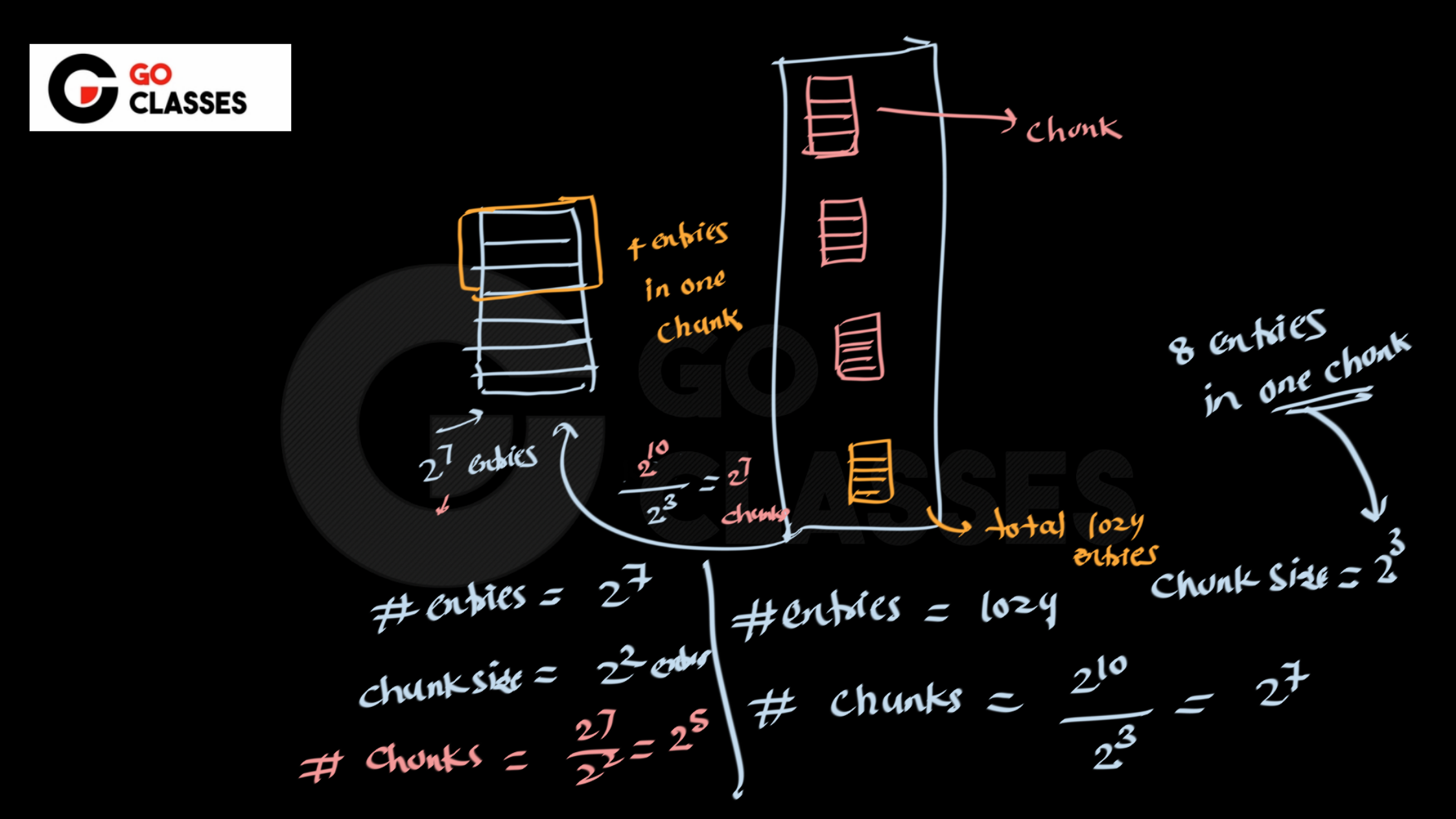
- We may not find continuous space to store page table in main memory.

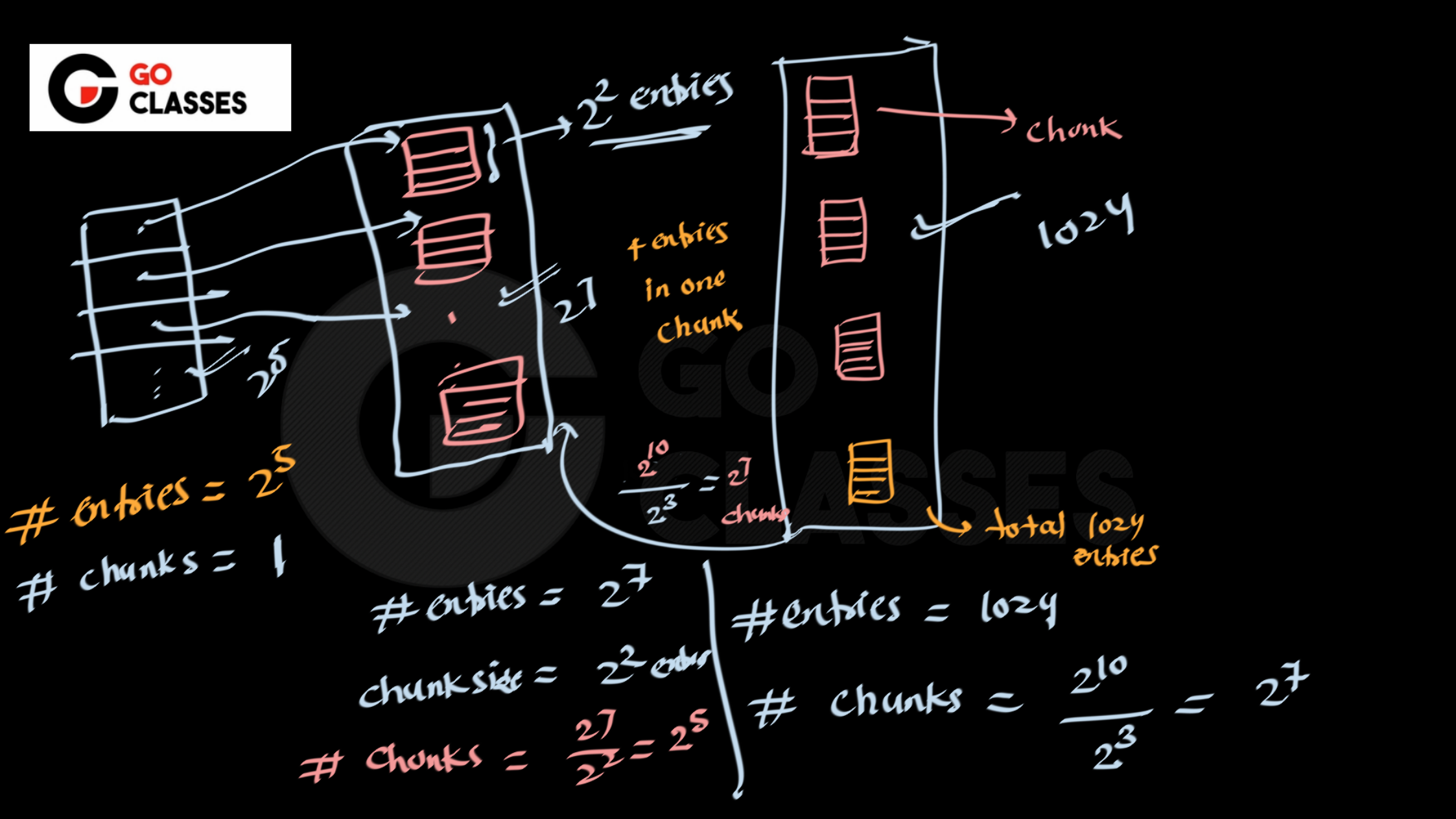
- We have seen most of the pages are empty.

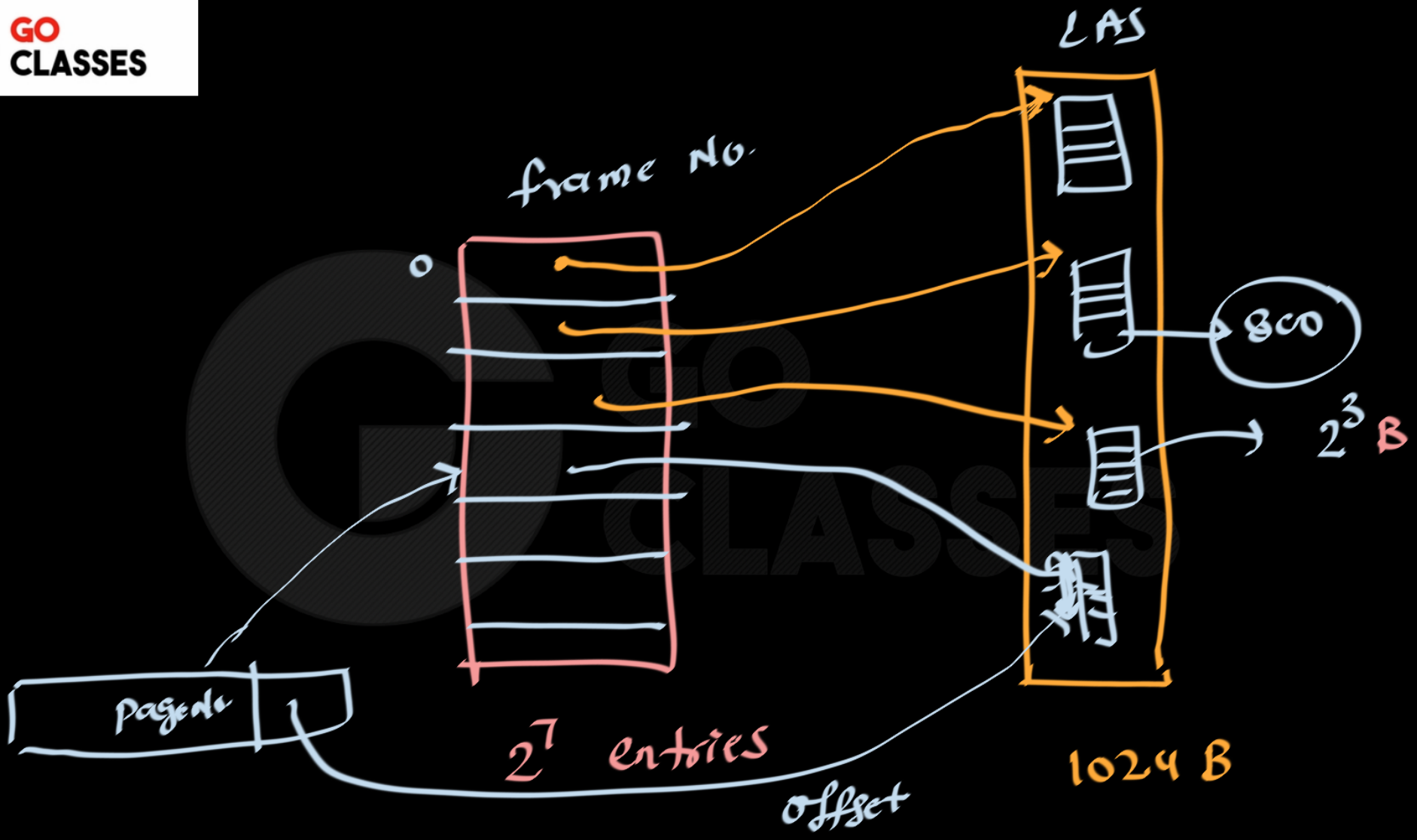
  Why to store entries corresponding to the pages which are empty ?

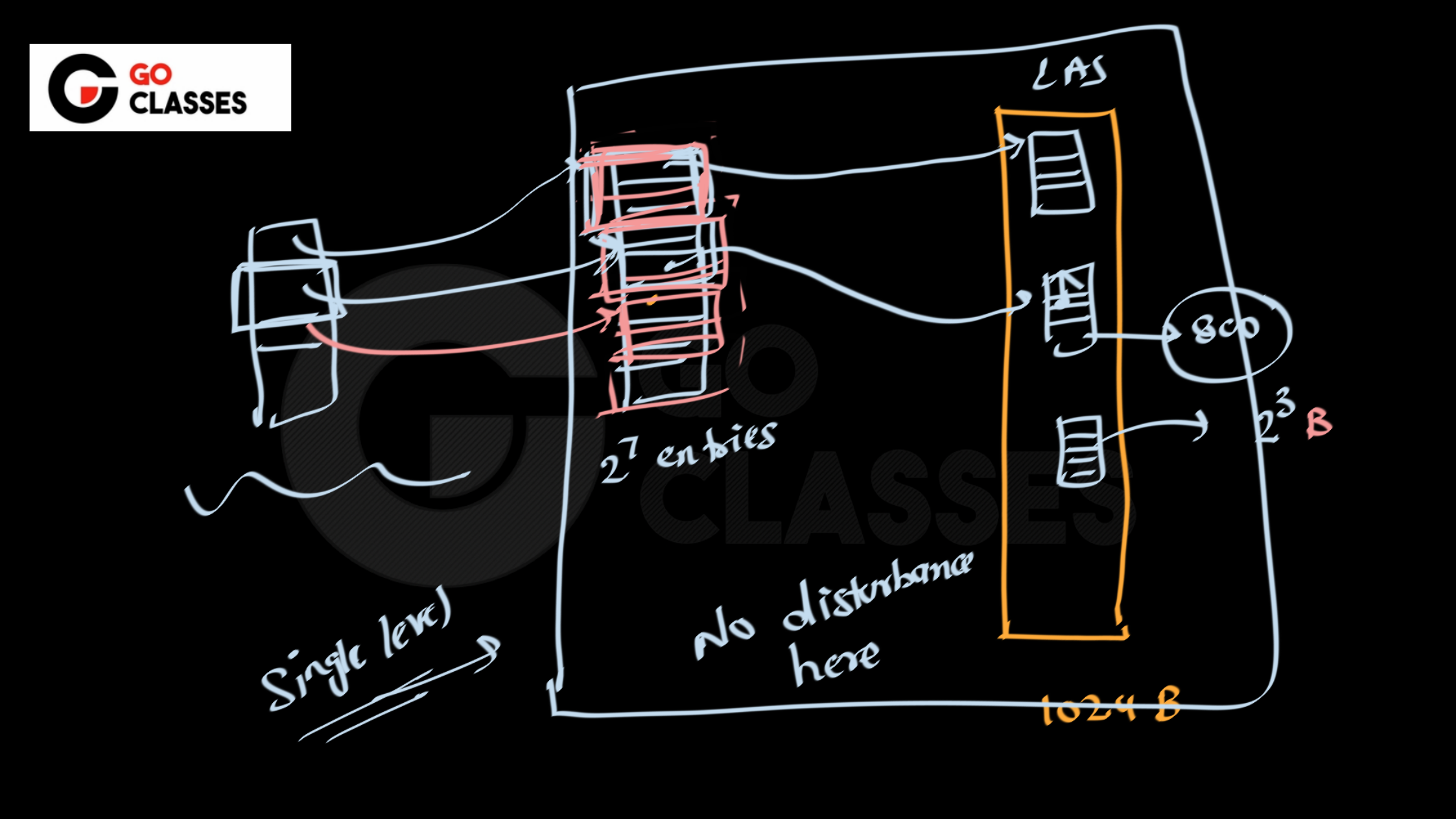

Each page contains = 8 Bytes
8 entries
0
1
2
0-7
Page 0
Page 1
Pg No
Frame No
0
1
100
$2^7 - 1$
2^7 Entries
128 entries
800
Page 127
1023
LAS
1024 entries
PM


GO CLASSES
Chunk
t entries in one Chunk
2^7 entries
$\frac{2^{10}}{2^3} = 2^7$ chunks
total lazy entries


GO CLASSES
Chunk
entries in one Chunk
2^7 entries
$\frac{2^{10}}{2^3} = 2^7$ chunks
total lozy entries
8 entries in one chunk
Chunk Size = 2^3
entries = 2^7
Chunk size = 2^2 entries
Chunks = $\frac{2^7}{2^2} = 2^5$
entries = lozy
Chunks = $\frac{2^{10}}{2^3} = 2^7$

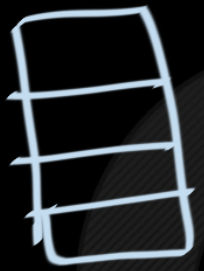

GO CLASSES
2^2 entries
chunk
1024
entries in one chunk
27
25
$\frac{2^{10}}{2^3} = 2^7$ chunks
total 1024 entries
entries = 2^5
chunks = 1
entries = 2^7
chunk size = 2^2 entries
Chunks = $\frac{2^7}{2^2} = 2^5$
entries = 1024
Chunks = $\frac{2^{10}}{2^3} = 2^7$


LAS
frame No.
0
800
2^3 B
Page No.
2^7 entries
1024 B
offset


GO CLASSES
LAS
2^7 entries
Sco
2^3 B
Single level
No disturbance here
1024 B

GO CLASSES

2 entries



2^2 entries

2^7 entries

chunk size = 2^1

chunks = $\frac{2^2}{2^1}$ =

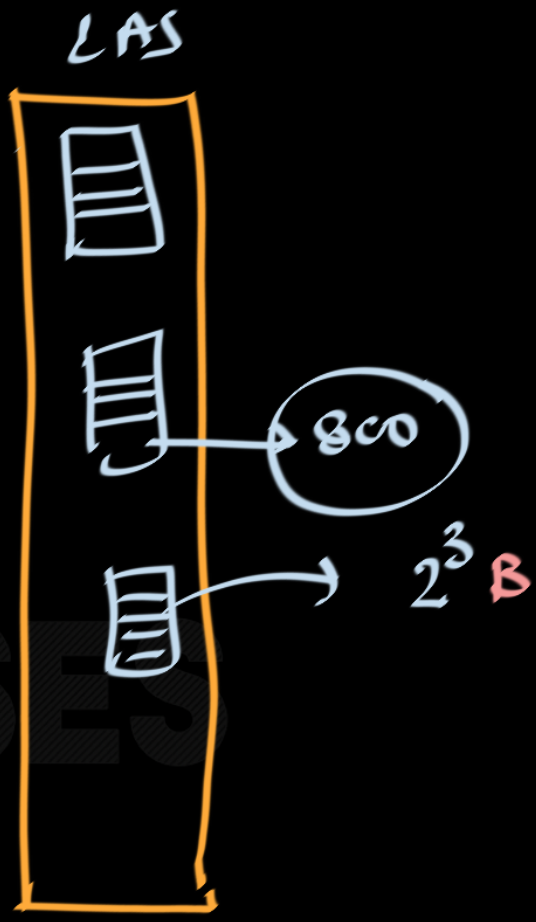
2

chunk size

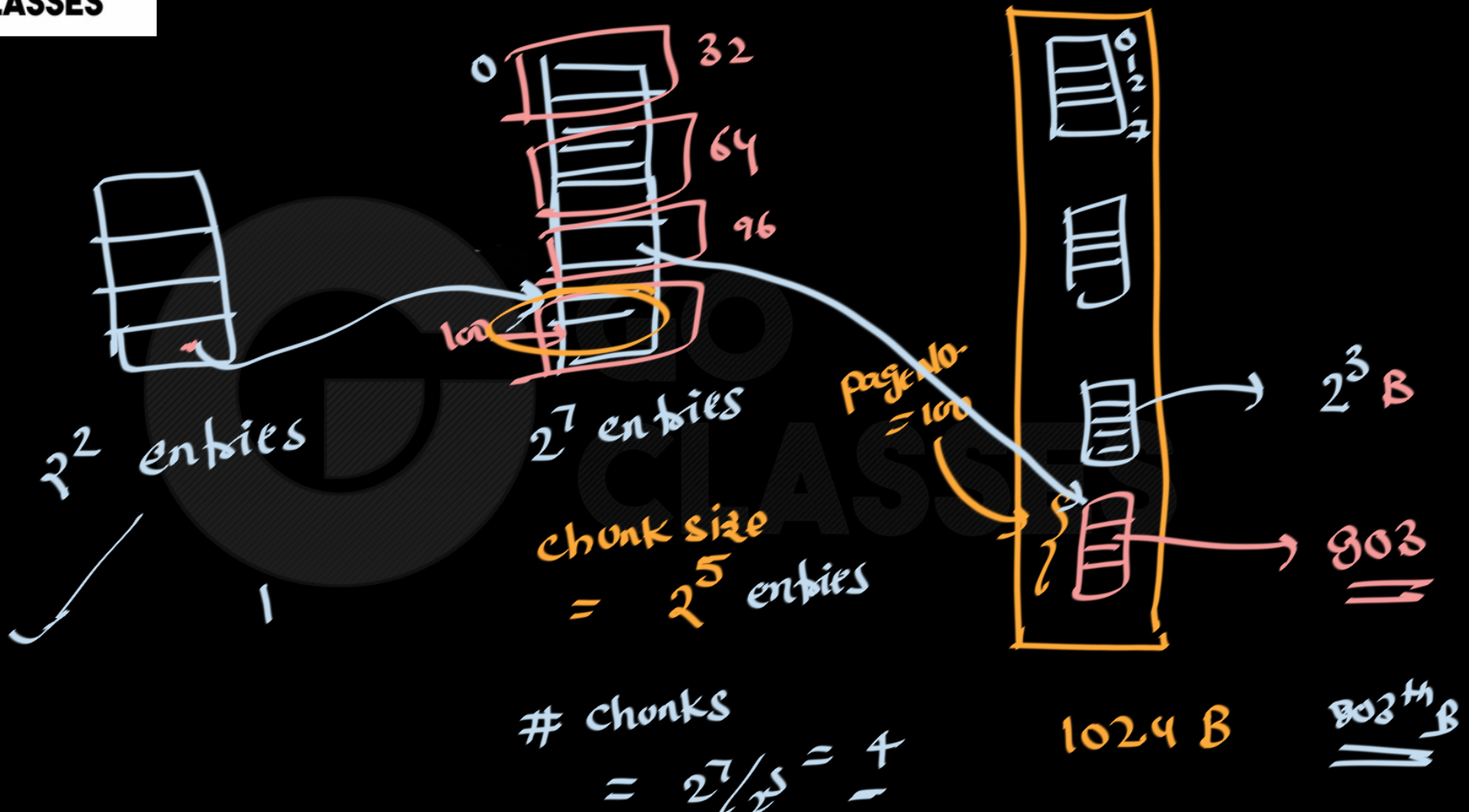
= 2^5 entries

chunks

= $2^{7/5}$ = 4



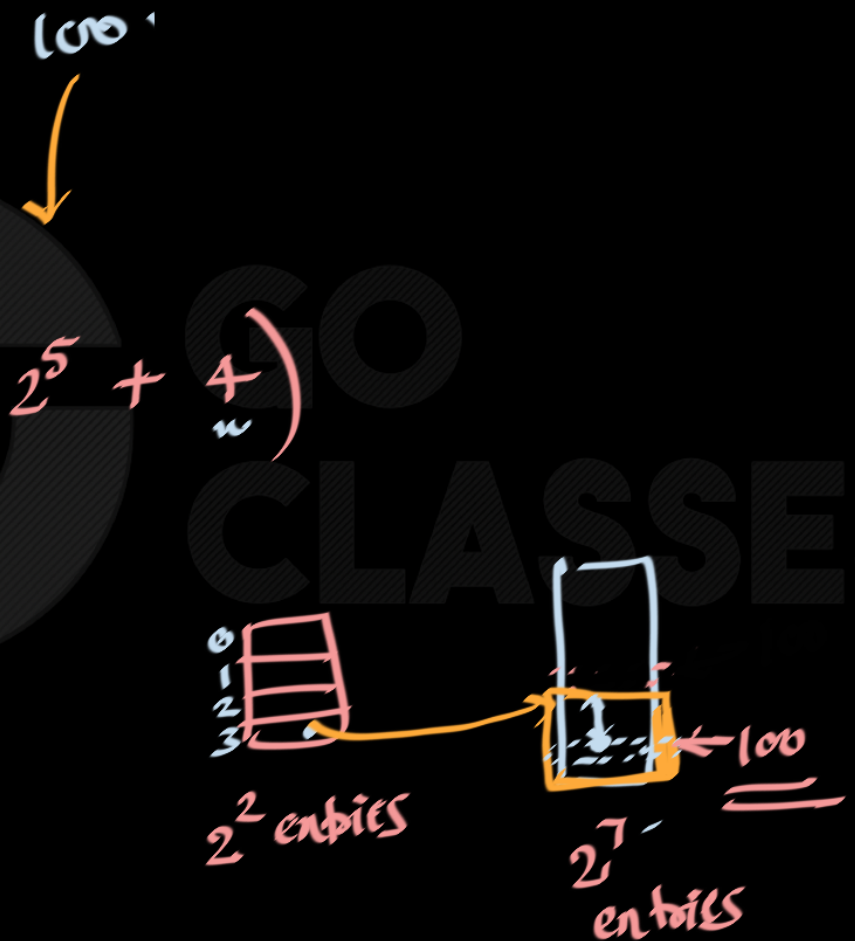
1024 B


GO CLASSES
0
32
64
96
100
2^2 entries
1
2^7 entries
chunk size
$= 2^5$ entries
chonks
$= 2^7/2^5 = 4$
page no. = 100
0
1
2
3
2^3 B
803
1024 B
803 th B

GO CLASSES

$803 = 100 \times 2^3 + 3$

$803 = (3 \times 2^5 + 4) \times 2^3 + 3$



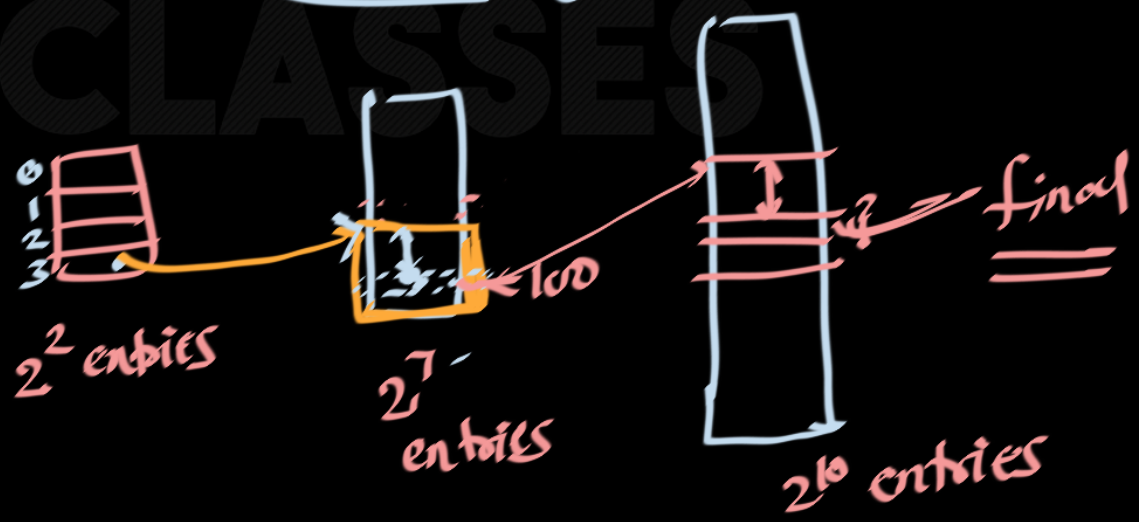
GO CLASSES

$$803 = 100 \times 2^3 + 3$$

$$803 = (3 \times 2^5 + 4) \times 2^3 + 3$$

Chunk size

Chunk size



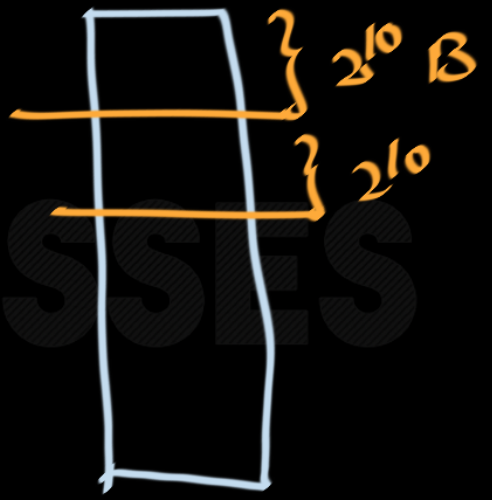
GO CLASSES

32 bit logical address

GO CLASSES

32 bit logical address

32 bit



$\frac{2^{32}}{2^{10}} = 2^{22}$ entries

2^{32} B LAS

GO CLASSES

32 bit logical address

32 bit

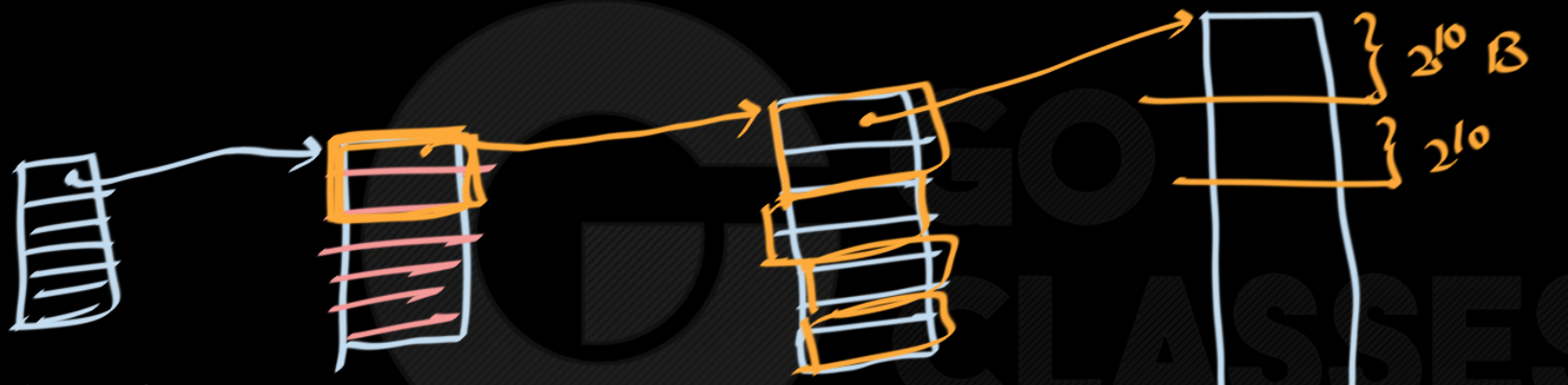
$\frac{2^{32}}{2^8} = 2^{14}$ entries $\quad \frac{2^{32}}{2^{10}} = 2^{22}$ entries $\quad 2^{32}$ B LAS

Chunk size
$= 2^8$ entries

GO CLASSES

32 bit logical address

32 bit



$\frac{2^{14}}{2^8} = 2^6$

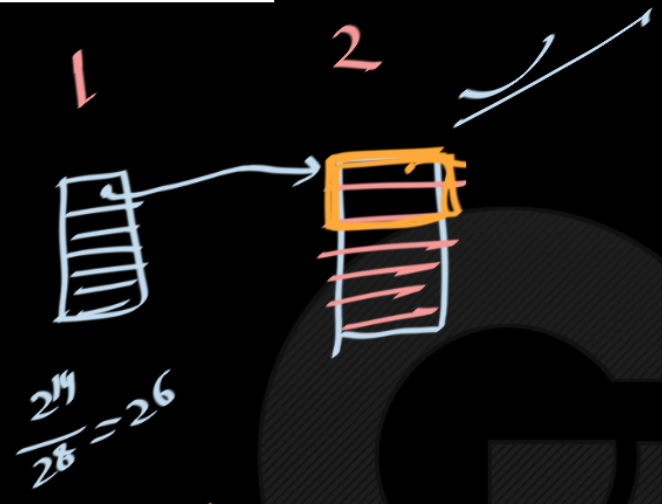
$\frac{2^{22}}{2^8} = 2^{14}$ entries

$\frac{2^{32}}{2^{10}} = 2^{22}$ entries

2^{32} B LAS

Chunk size = 2^8 entries

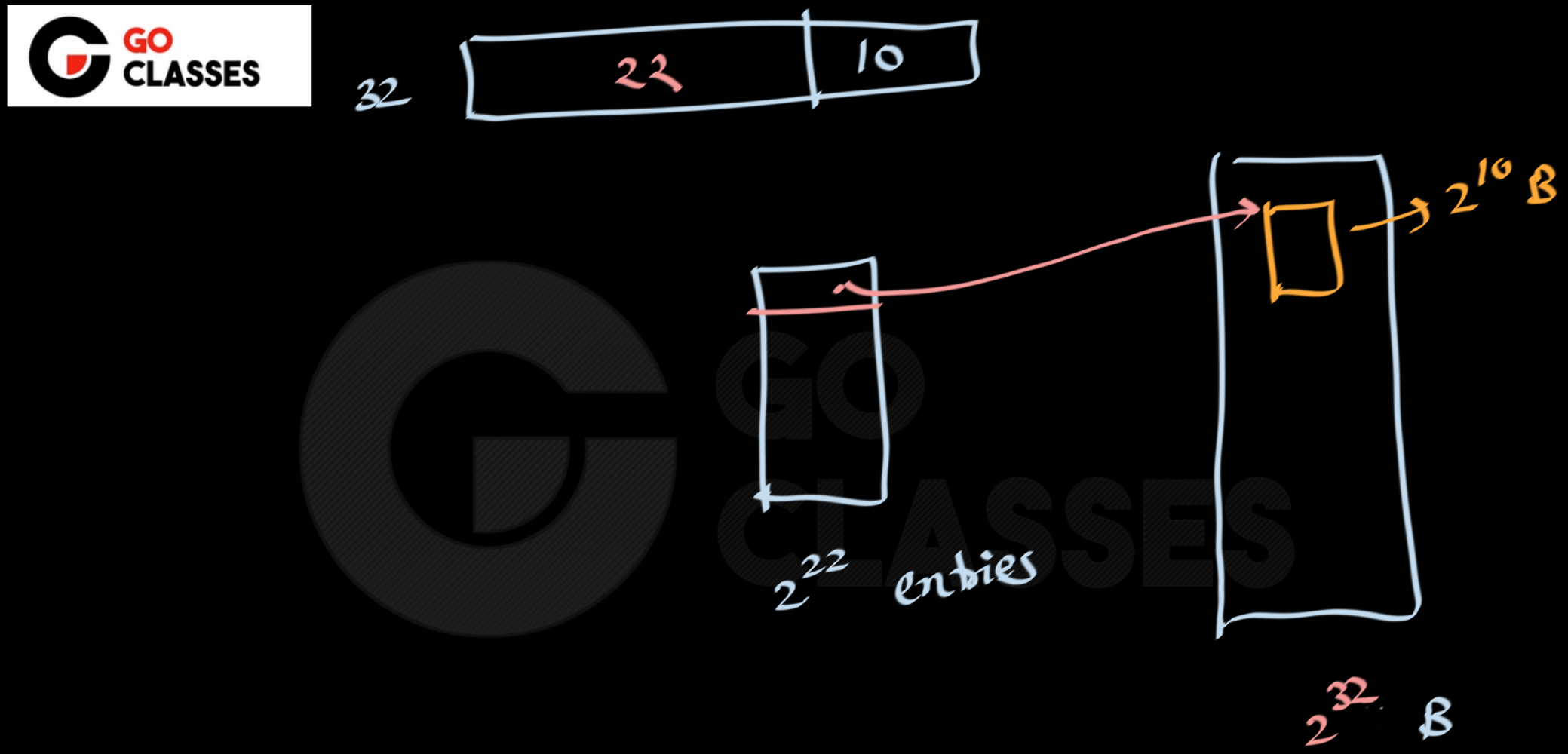
Chunk size = 2^8 entries

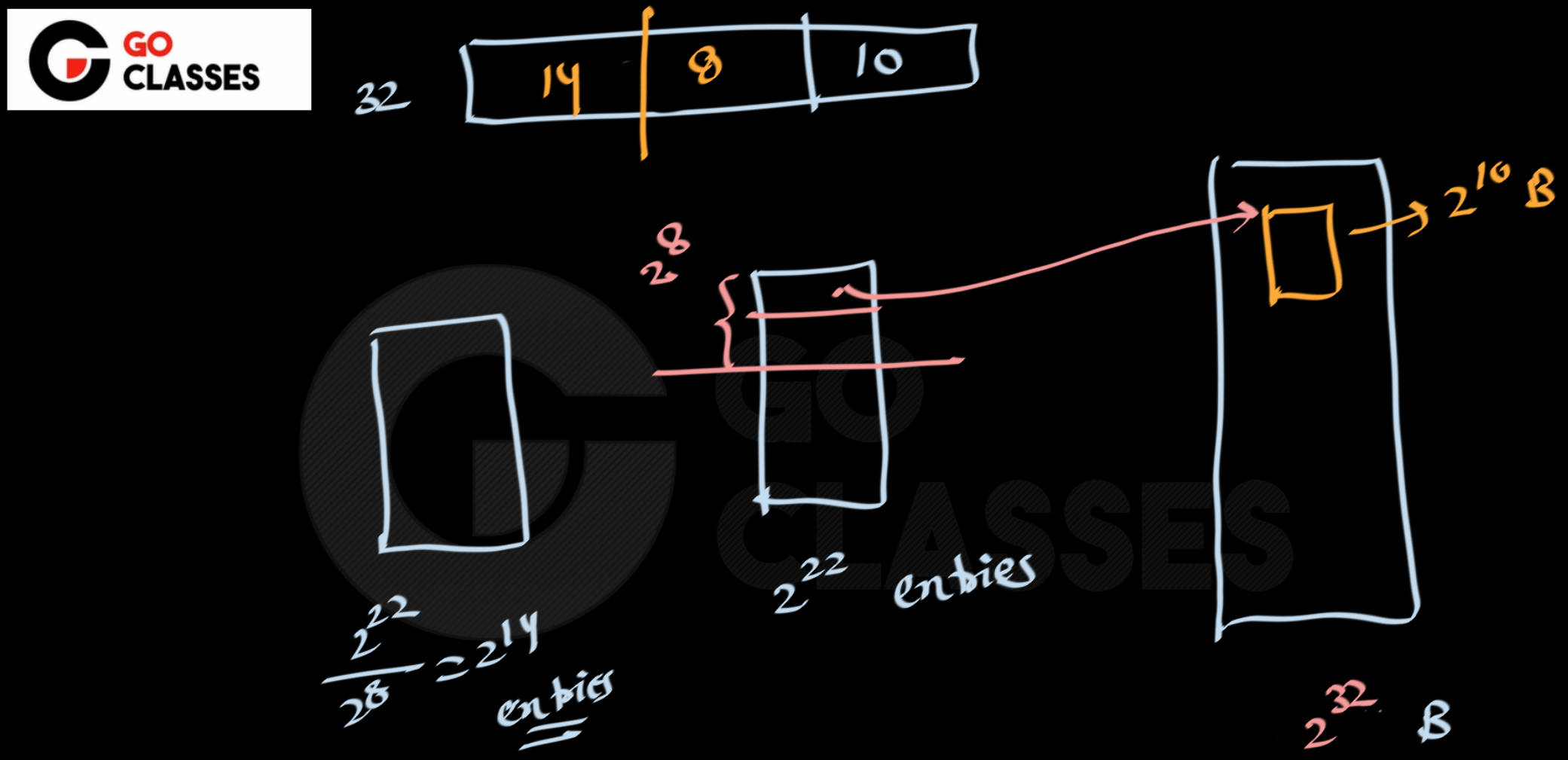


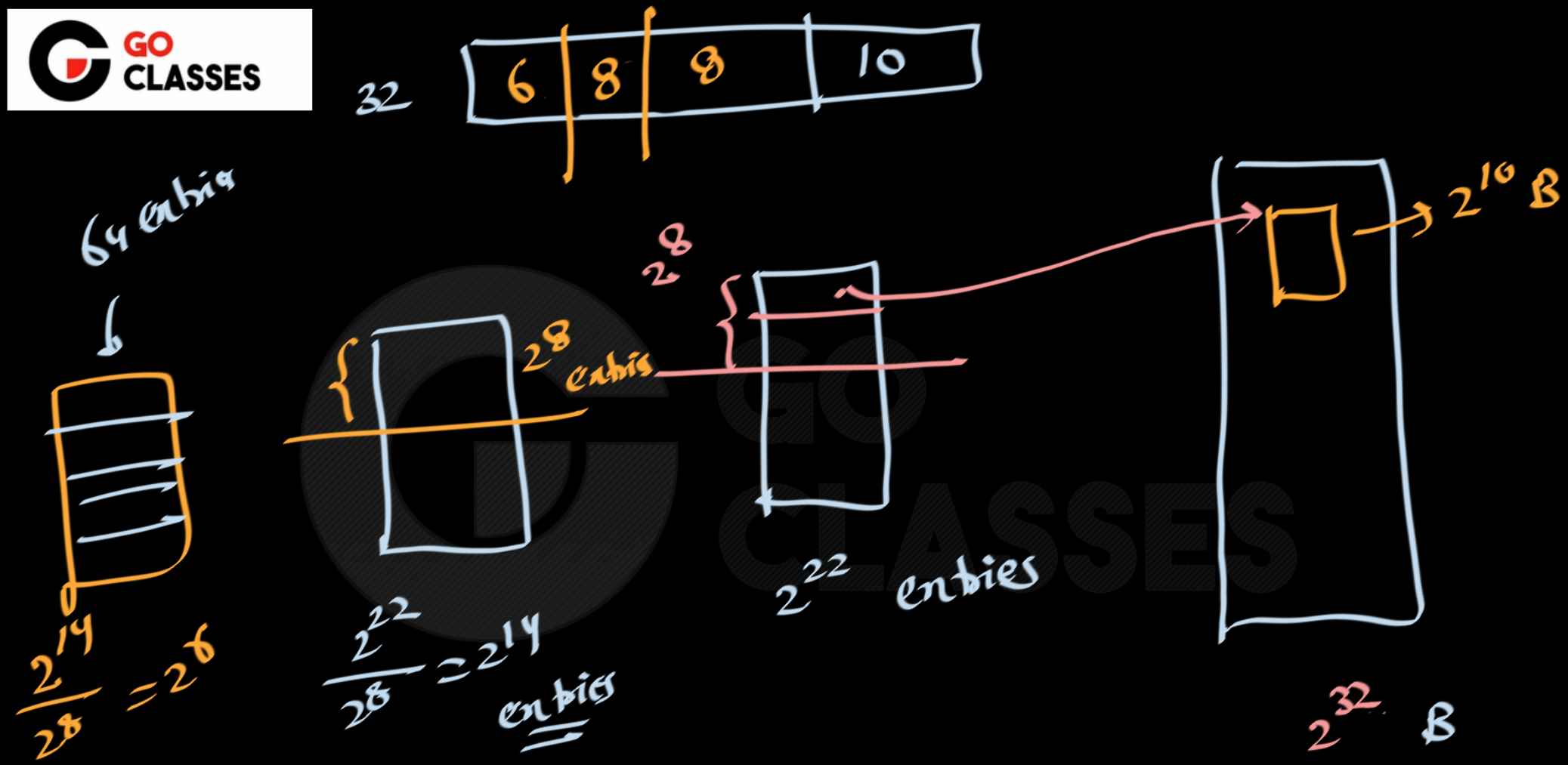
2^6 entries, every entry points to one chunk of size 2^8.

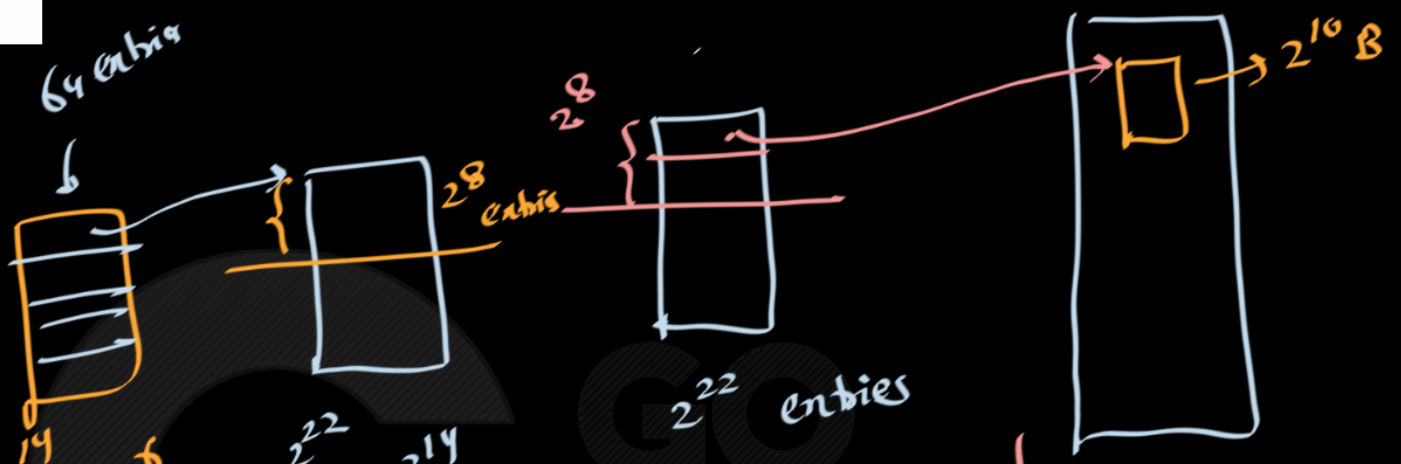
how many # chunks in level 2 : $2^6 \times 1 = 2^6$ chunks

how many # entries in level 2 : $2^6 \times 2^8 = 2^{14}$ entries


32
22
10
2^{10} B
2^{22} entries
2^{32} B


32
14
8
10
2^8
2^{16} B
2^{22} entries
$\frac{2^{22}}{2^8} = 2^{14}$ entries
2^{32} B

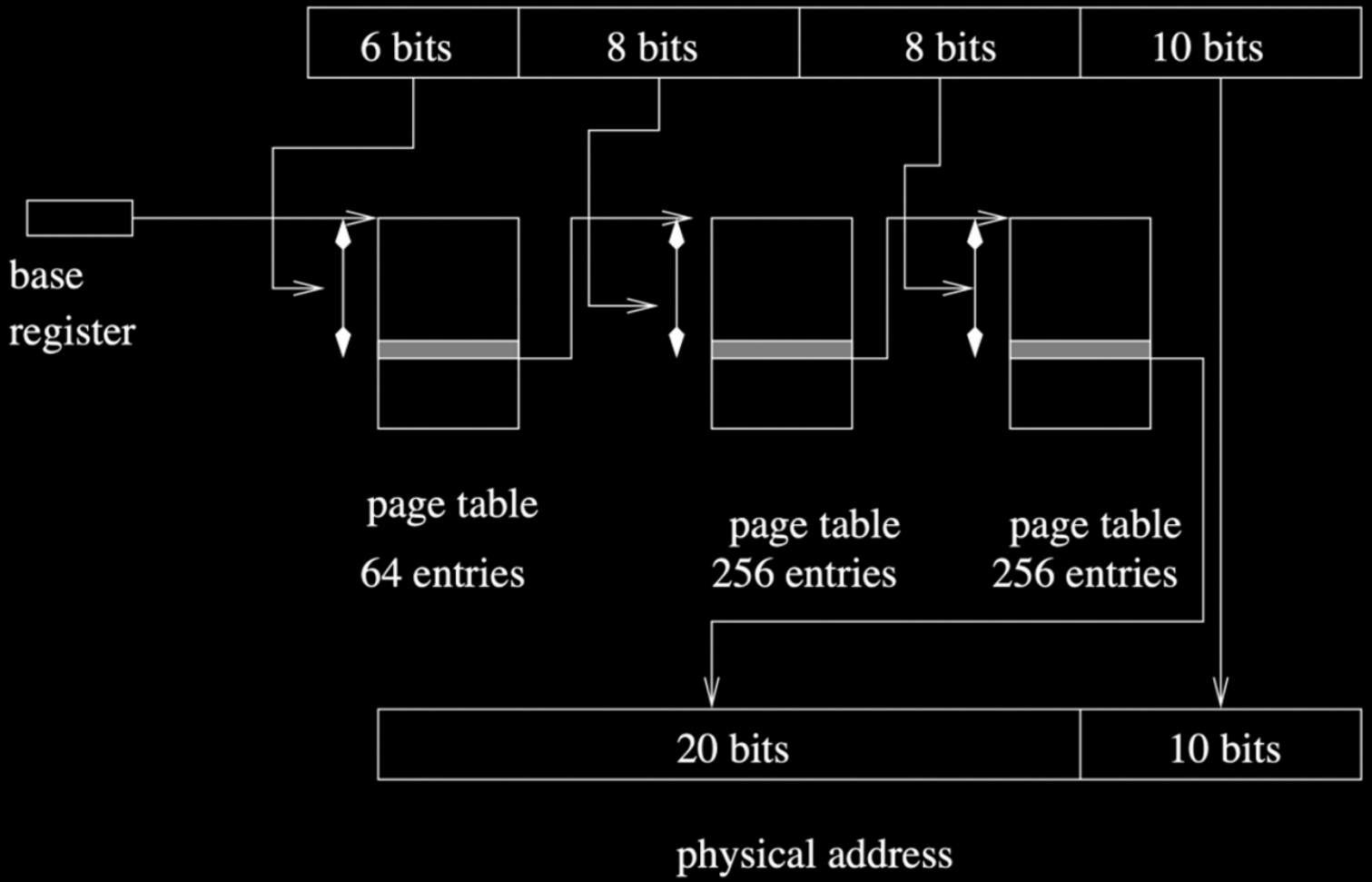

GO CLASSES
32
6
8
8
10
64 entries
$\frac{2^{14}}{2^6} = 2^8$
2^8
2^8 entries
$\frac{2^{22}}{2^8} = 2^{14}$ entries
2^{22} entries
2^{10} B
2^{32} B



~~Frame~~ Chunk

	$\frac{2^{14}}{2^8} = 2^6$ entries	$\frac{2^{22}}{2^8} = 2^{14}$ entries	2^{22} entries	2^{32} B
# of Chunks	1	2^6	2^{14}	2^{22}
Chunk Size	2^6	2^8	2^8	2^{10}
# of Entries	2^6	$2^6 \times 2^8 = 2^{14}$	$2^{14} \times 2^8 = 2^{22}$	$2^{22} \times 2^{10} = 2^{32}$ entries $= 2^{32}$ Bytes

6 bits	8 bits	8 bits	10 bits


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