CS & IT ENGINEERING

Programming in C

Arrays and Pointers

Lec- 04

By- Pankaj Sharma sir

Arrays and Pointers-4

int a[2][3][2] = {1,2,3,4,5,6,7,8,9,10,11,12};

= **&a[0]

= *a[0]

⇓

*&a[0][0]

⇓

a[0][0]

⇓

&a[0][0][0]

Delaration and initialization

1) `int a[];` Invalid
2) `int a[3];` ✓
3) `int a[] = {1,2,3};` ✓
4) `int a[3] = {1};` ✓
5) `int a[3] = {1,2,3};` ✓

① If only declaration is there without any initialization

⇒ It is mandatory to provide size of each dimension.

② If we are also initializing an array, then there is flexibility to omit the size of 1st dimension.

This flexibility is only for 1st dimension not for any other dimension.

(i) `int a[];`

(ii) `int a[][];`

(iii) `int a[2][];`

(iv) `int a[][3];`

} All are invalid

(v) `int a[3];` ✓

(vi) `int a[2][3];` ✓

(vii) `int a[][2][3];` X

(viii) `int a[2][][3];` X

(ix) `int a[2][3][];` X

(x) `int a[2][3][2];` ✓

(i) `int a[3] = {1,2,3};`

(ii) `int a[] = {1,2,3};` // 1st dim ⇒ omit ✓

(iii) `int a[][] = {1,2,3,4,5,6};` ✗

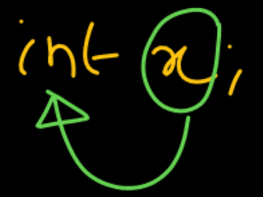
(iv) `int a[2][] = {1,2,3,4,5,6};` ✗

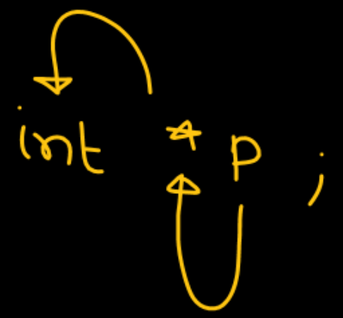
(v) `int a[][3] = {1,2,3,4,5,6};` ✓

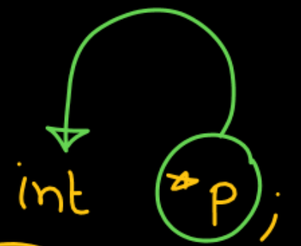
(vi) `int a[2][3] = {1,2,3,4,5,6};` ✓

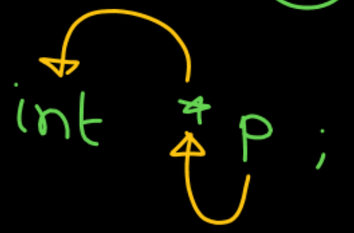
Pointer

(1) Pointer is a **special variable** which is used to hold address of other variable









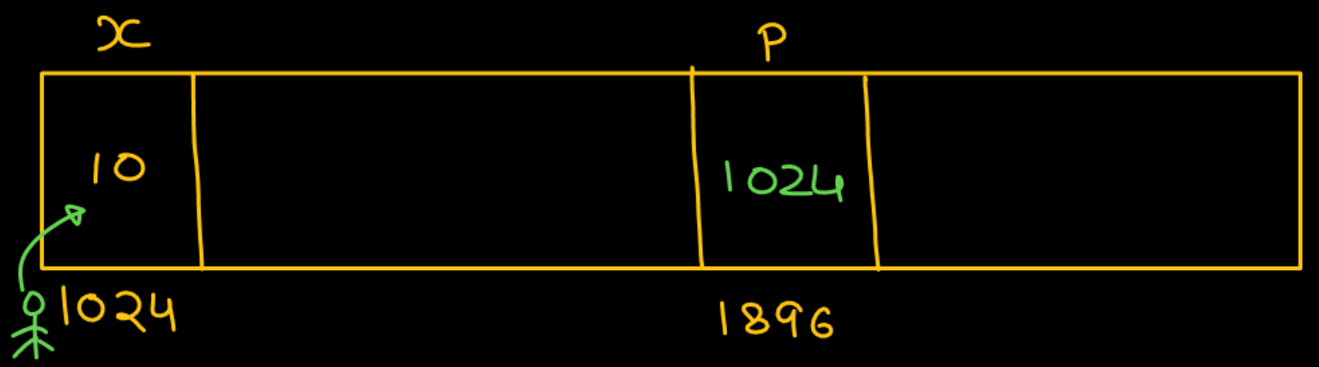
P is a pointer to integer

P can hold address of some integer variable.

```
int x = 10;
int *p;
p = &x;
```

1024

```
printf("%u", p);   1024
printf("%u", *p);  10
```

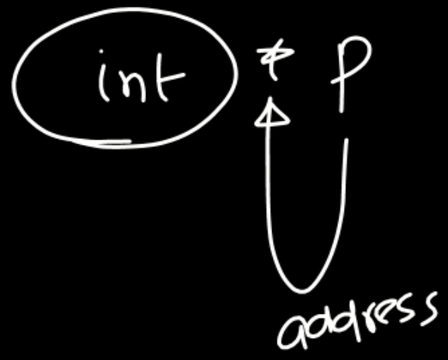


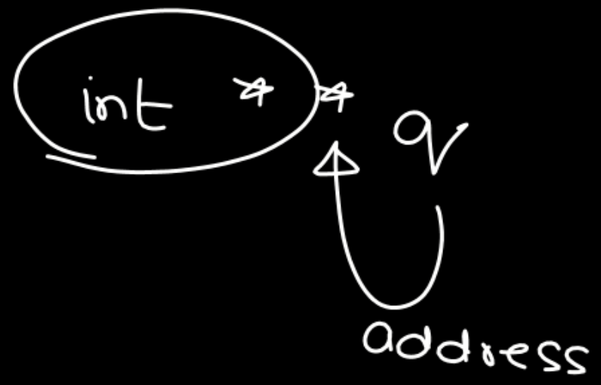
x : 10

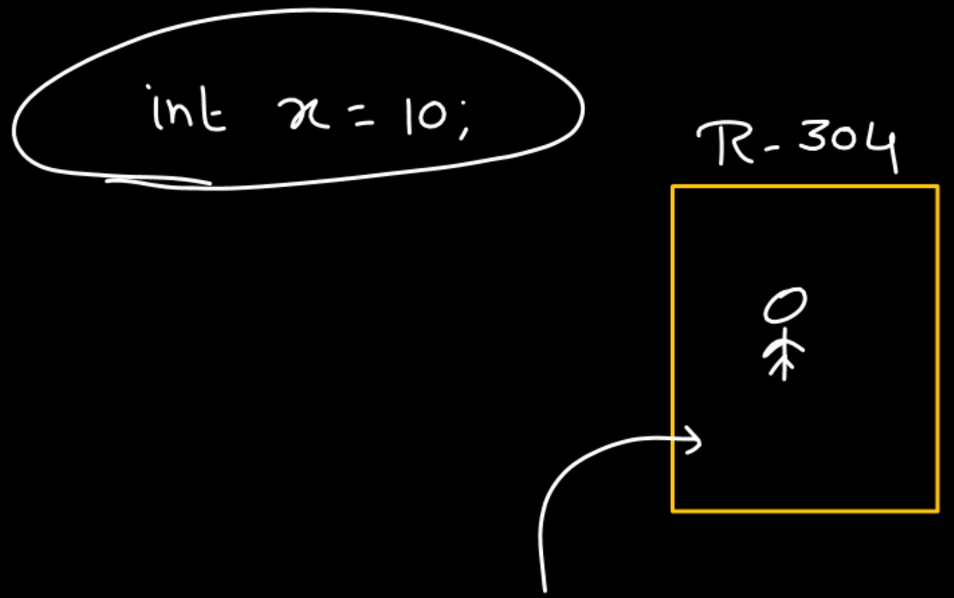
p : Memory location 1024

*p = value at (Memory location 1024)

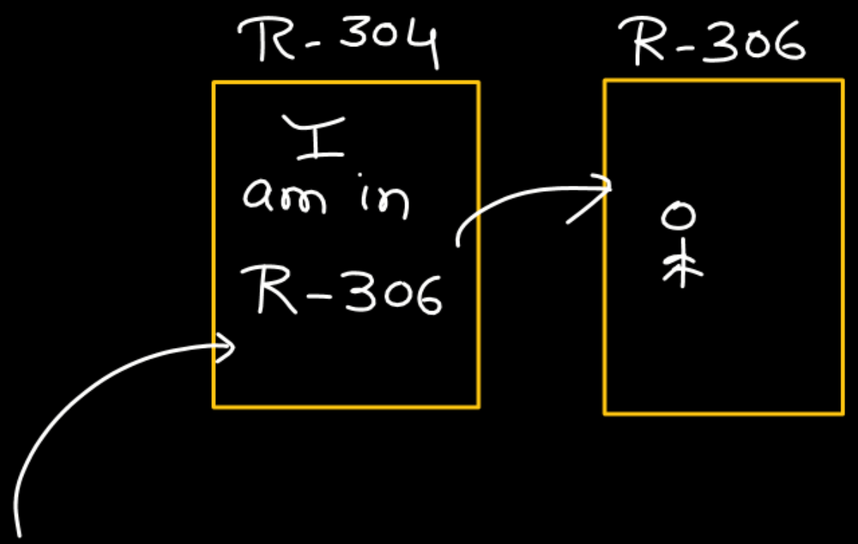
```
int x = 10;
int *p;
int **q;
```



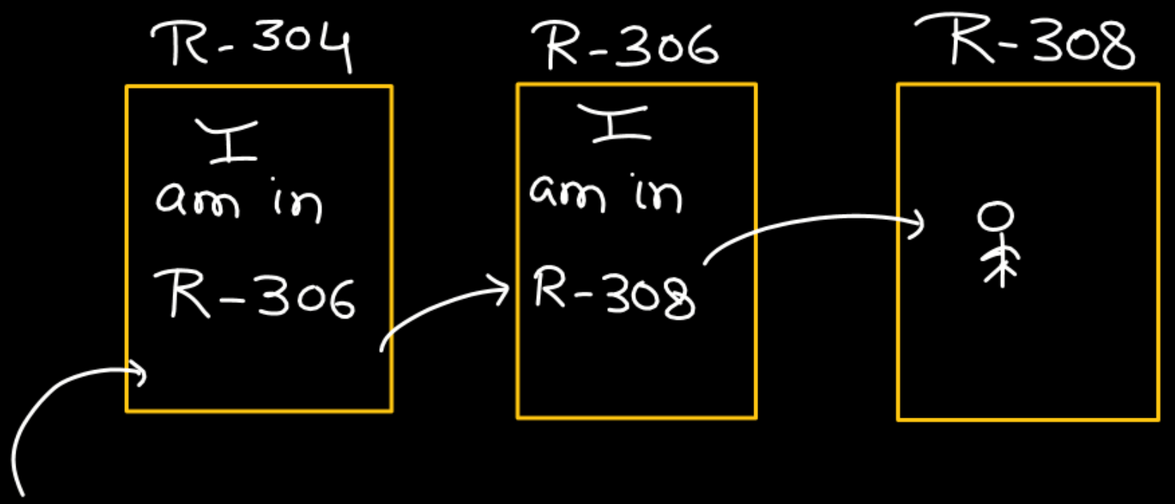


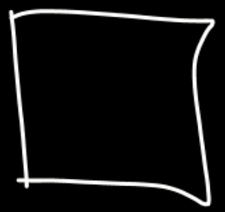

int x = 10;
R-304

```
int x=10
int *p;
```



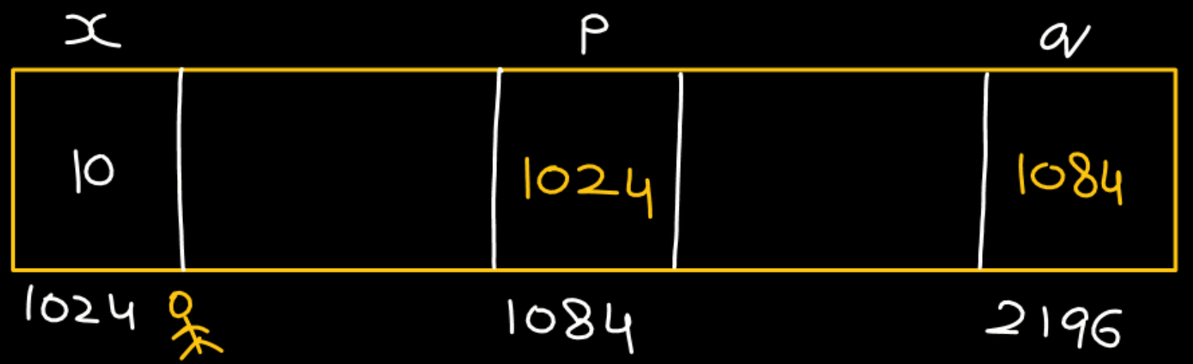
int **q;



q ⇒ Address of 

```
int x = 10
int *p;
int **q;
p = &x ;
q = &p ;
```

```
pf("%u", p);   1024
pf("%d", *p);  10
pf("%u", q);   1084
```



x : 10

p : Memory location 1024

q : Memory location 1084

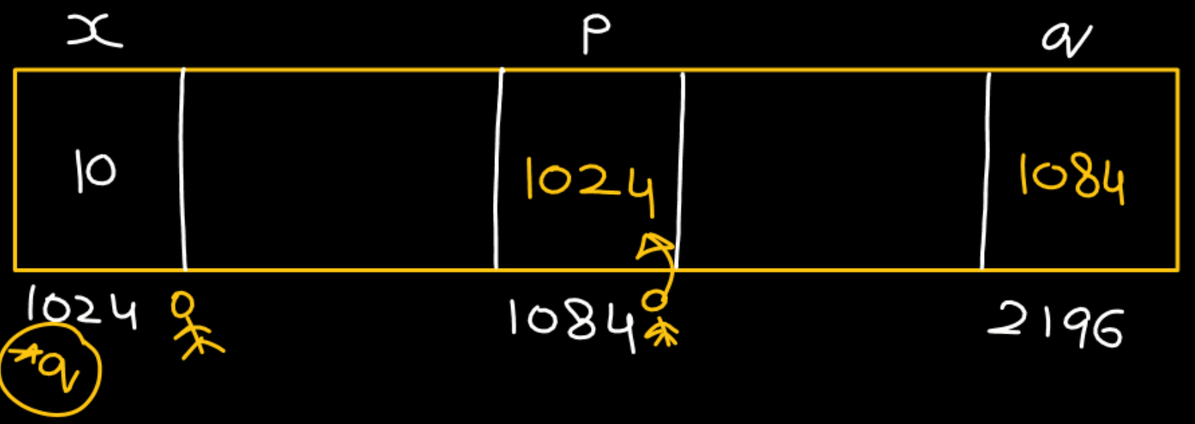
p = Mem. location 1024

*p = value at (Memory location 1024) = 10

q : Memory location 1084

```
int x = 10
int *p;
int **q;
p = &x ;  ✓
q = &p ;  ✓
```

```
pf("%u", p);   1024
pf("%d", *p);  10
pf("%u", q);   1084
pf("%u", *q);  1024
pf("%u", **q); 10
```



q : Memory location 1084

*q : valueat (Memory location 1084)

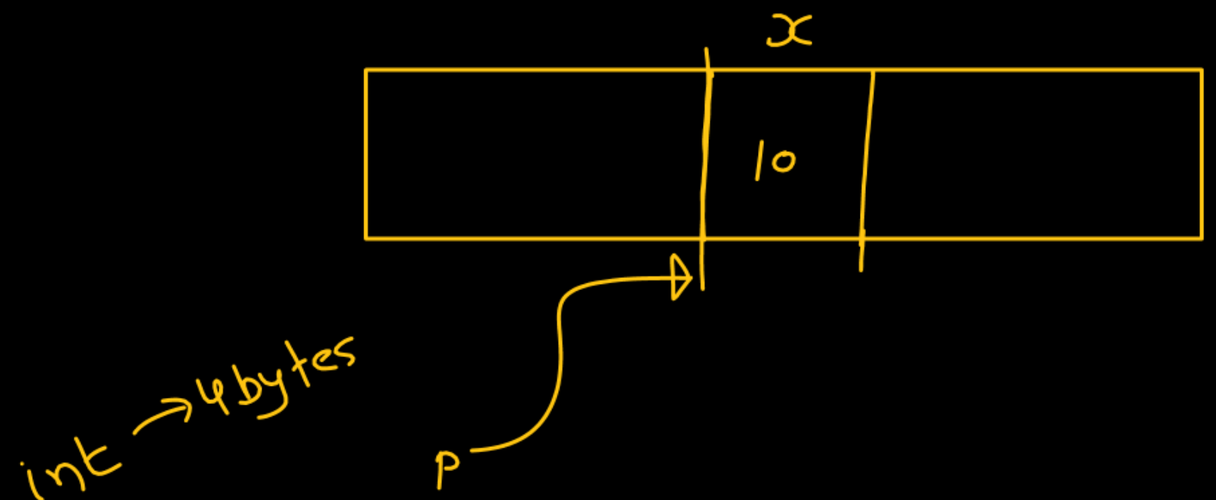
1024 ⇒ is a memory location

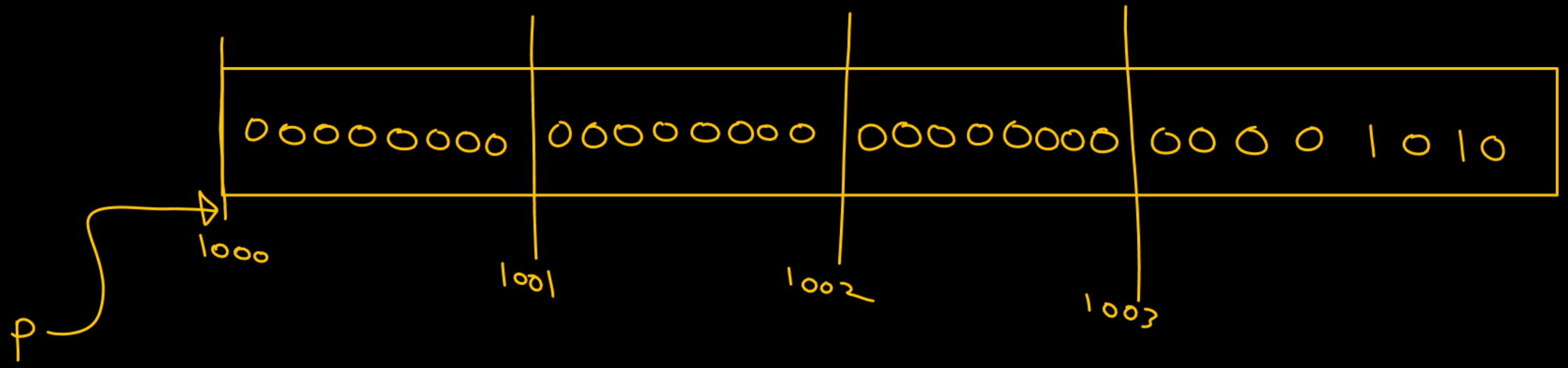
*q : Memory location 1024

**q : valueat (Memory location 1024)

= 10

```
int x = 10;
int *p;
p = &x;
```





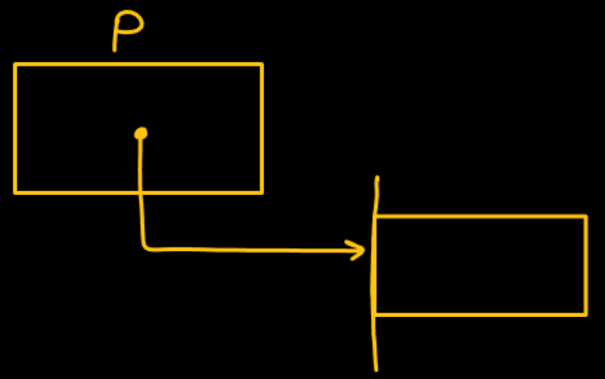
assume: int : 2 byte
float : 4 byte

2 BHK

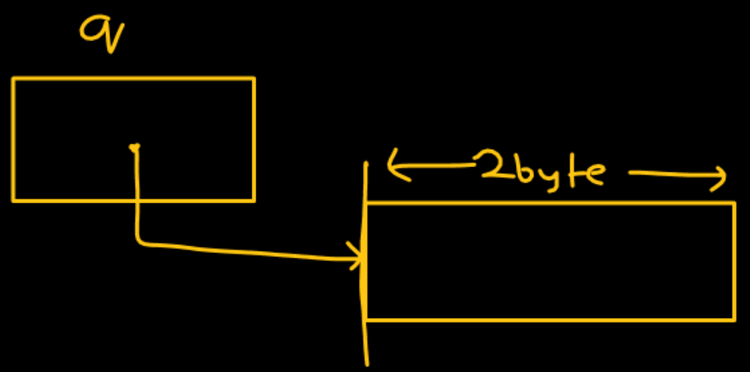
5 BHK

10 BHK

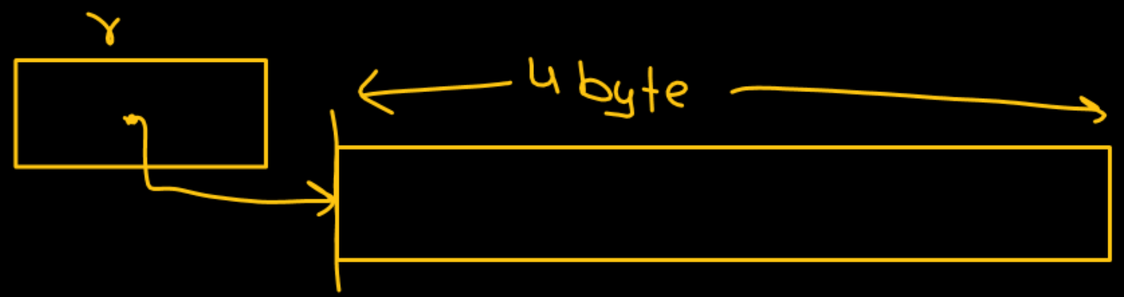
char *p ;



int * q



float * r



2.

```
int x=10;
int *p;
p= &x
```

*p

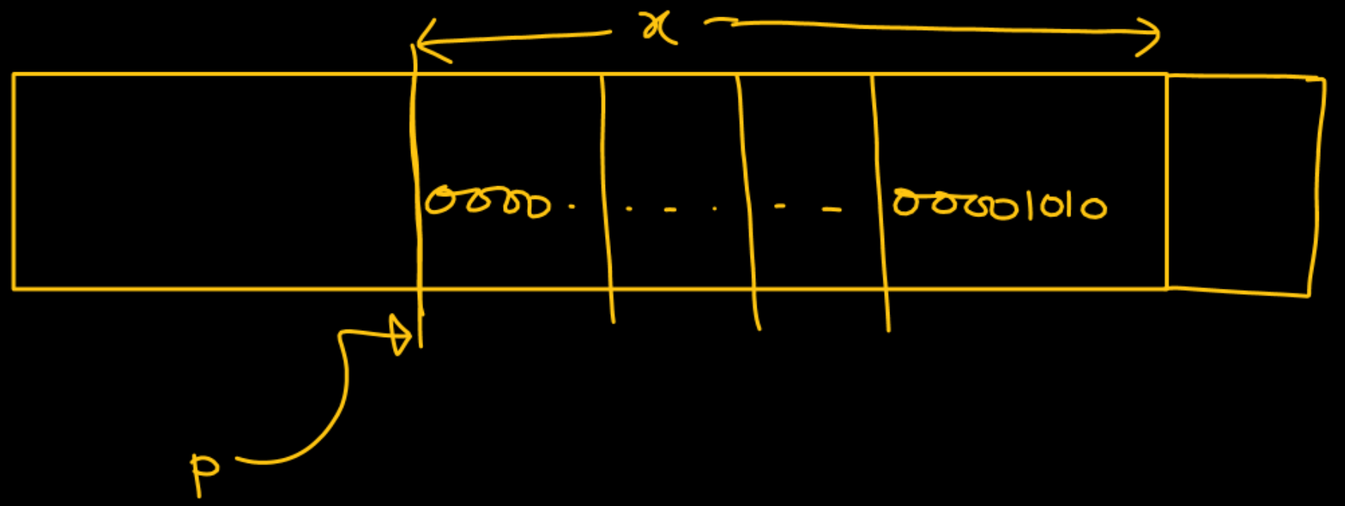
⇓

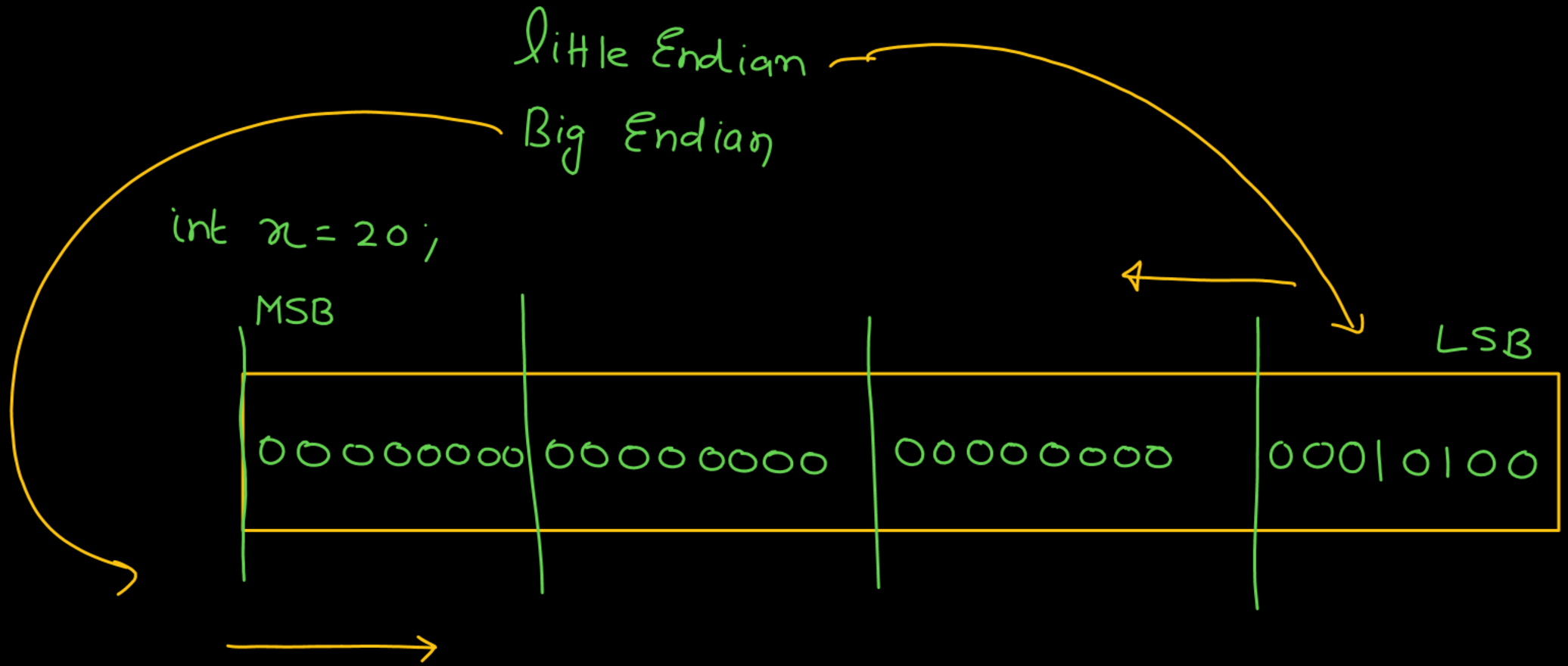
dereferencing

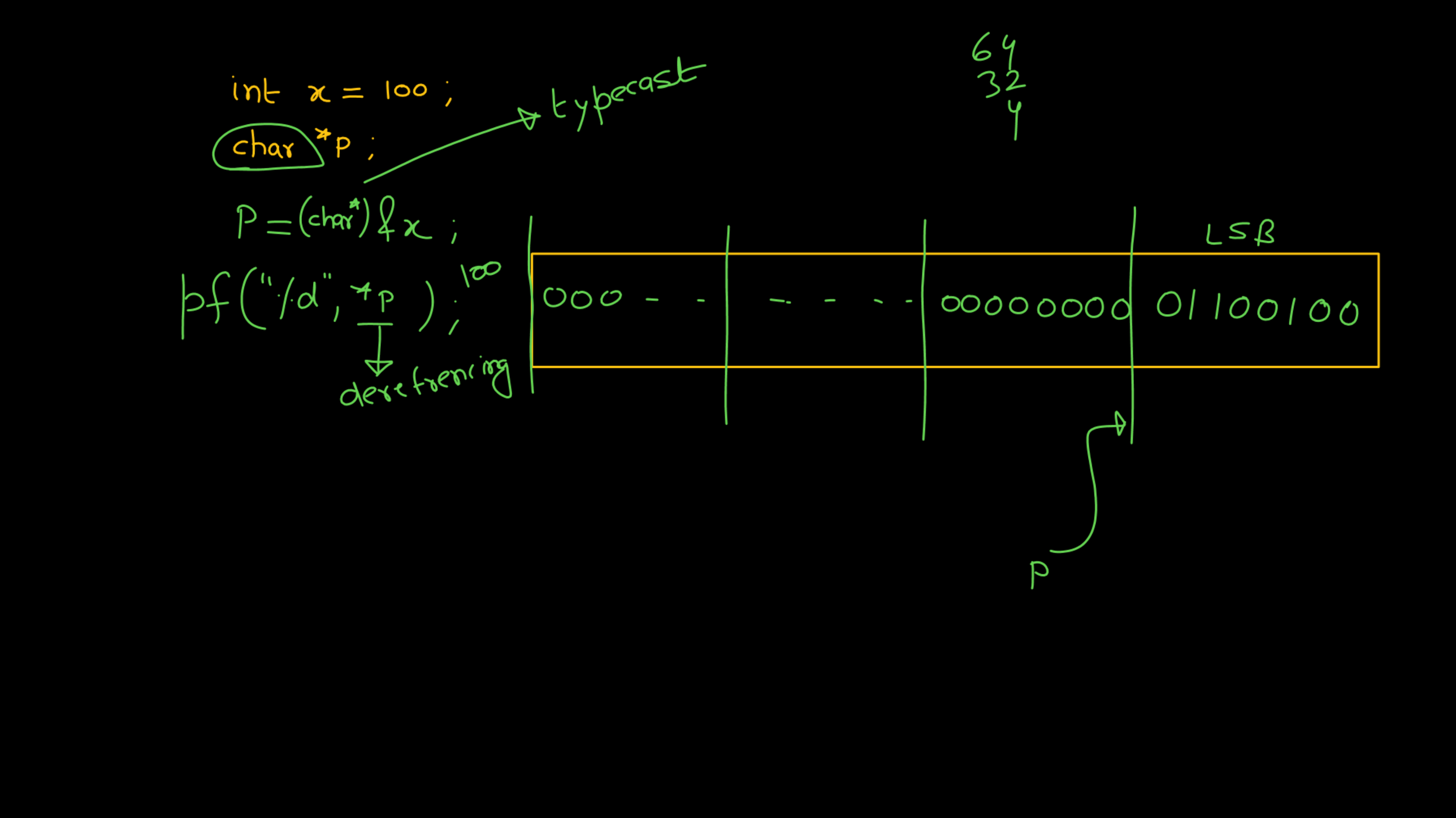
value fetch

```
pf("%d", *p)
```

4 byte



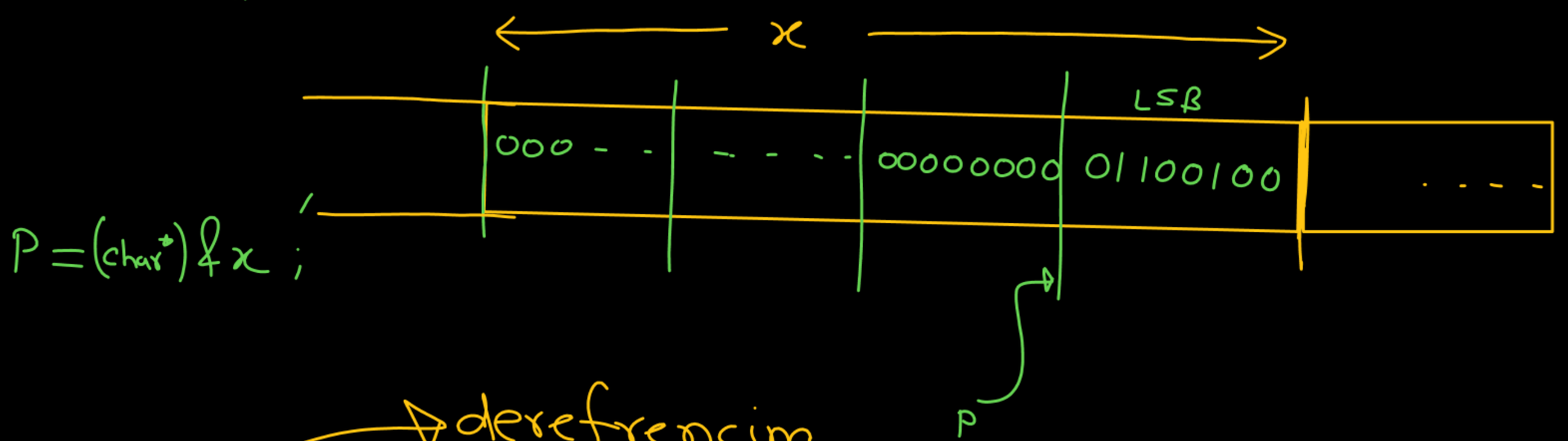

little Endian
Big Endian
int x = 20;
MSB
LSB
00000000
00000000
00000000
00010100


int x = 100 ;
char *p ;
typecast
64
32
4
p = (char*) &x ;
pf("%d", *p) ;
100
dereferencing
LSB
000 - -
- - - -
00000000
01100100
p

int x = 100;

char *p; // p को 32/64 bit char का addr.

dereferencing => 1 byte



p = (char*) &x;

pf("%d", *p);

dereferencing

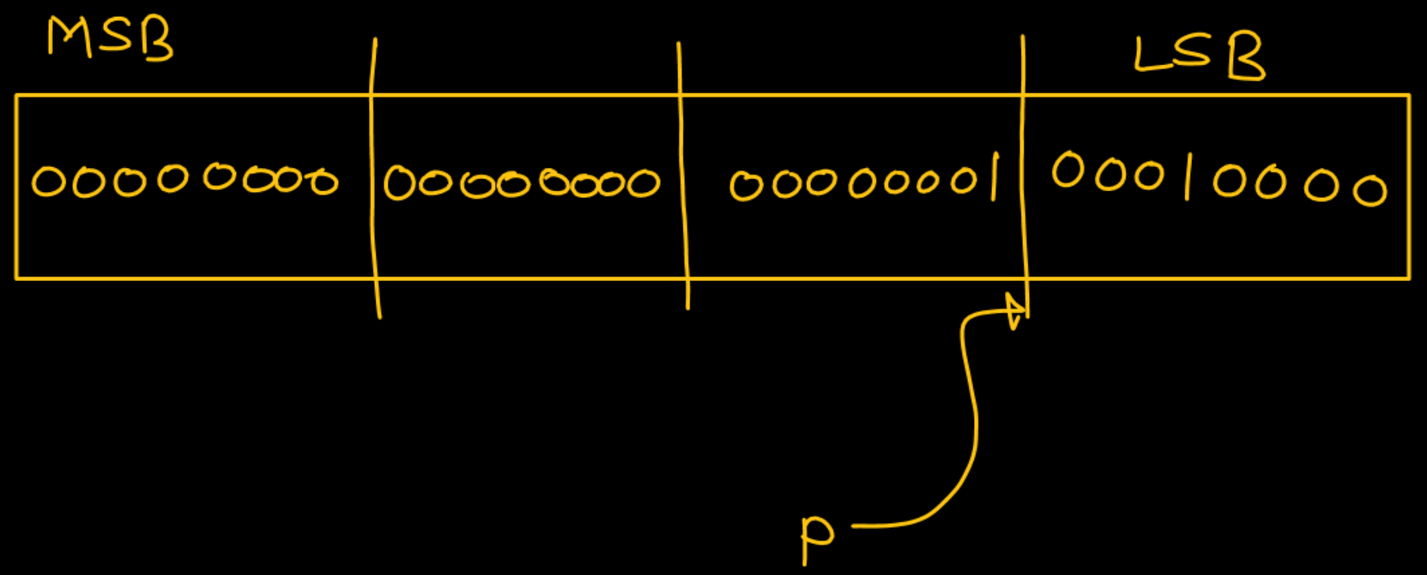
char *p

1 byte की value fetch करेगा

```
int x = 272;
char *p;
P= (char*) &x;

printf("%d", *p);
```

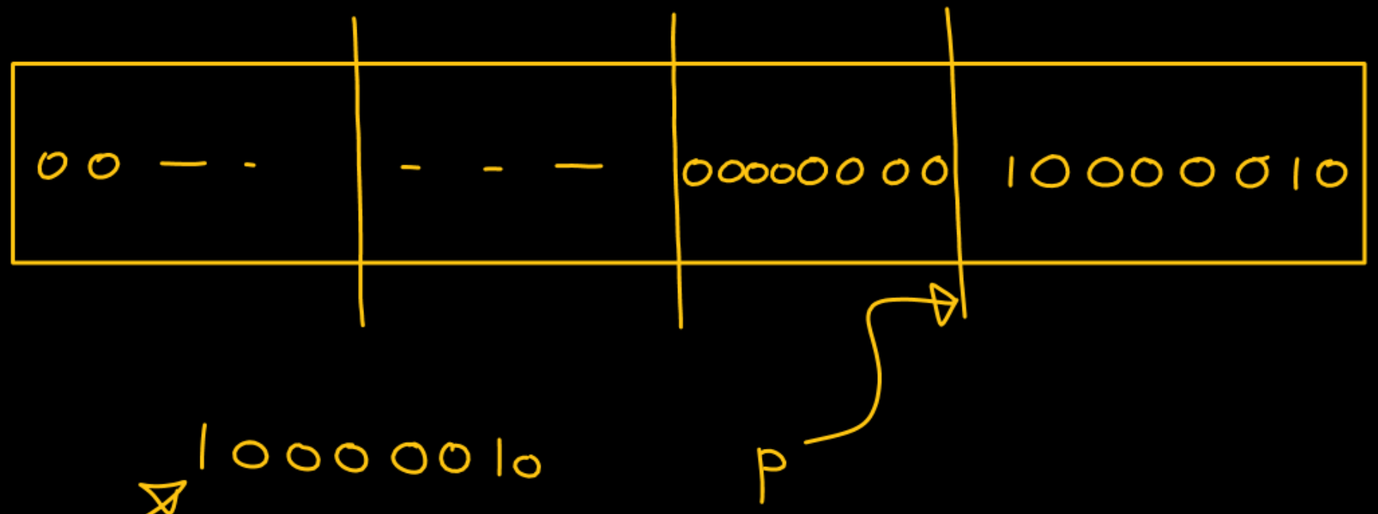
→ 256 + 16



```
int x = 130;
char *p;
p = (char*)&x;
printf("%d", *p);
```

-ve

2's complementation



10000010

-ve

2's complement

last day → 2's comp. → 10000010

C lang.

last day

binary ⇒ 1 पे focus

Cyclic property

10000010
$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$= -2^6 - 2^5 - 2^4 - 2^3 - 2^2 - 2^0 - 1$ ← Extra

$= -64 - 32 - 16 - 8 - 4 - 1 - 1$

$= -96 - 24 - 6$

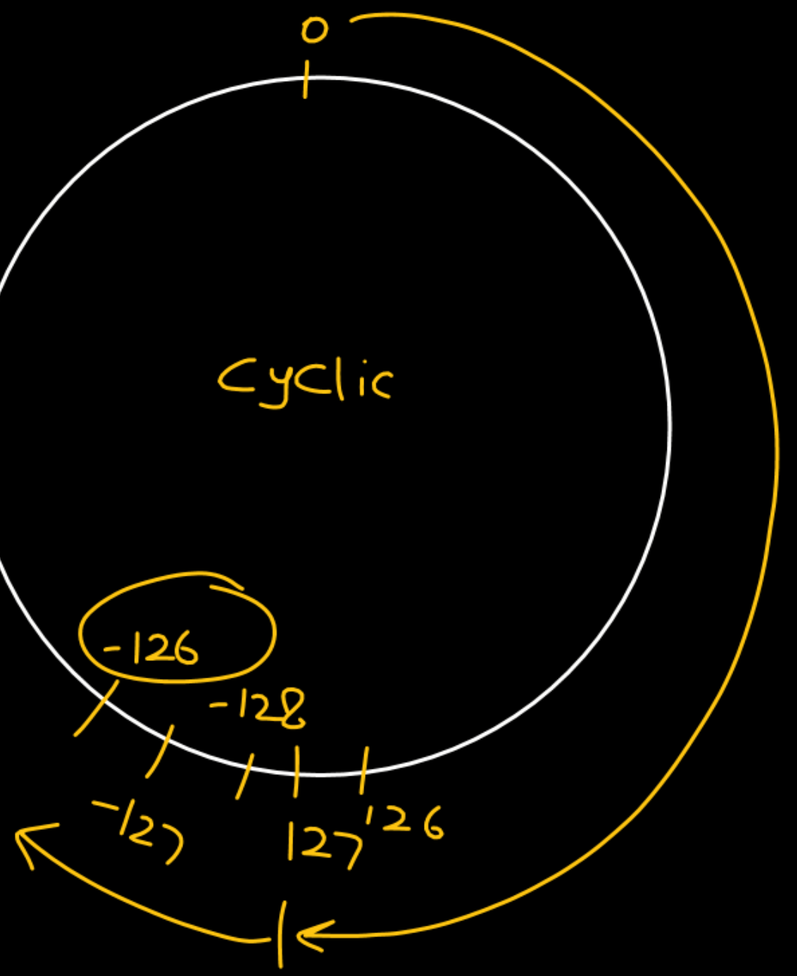
$= -126$

Dennis Ritchie

480 % 256

= 144 − 256

= −112




THANK YOU GW
SOLDIERS !