Let’s solve it

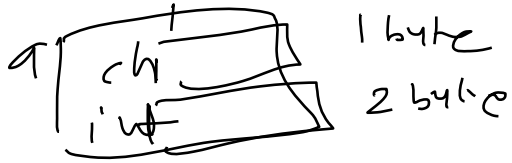
36

int x; sizeof(int) // sizeof(x);

```
struct XYZ
{
    char ch;
    int i;
};
```

struct XYZ a;

sizeof(a) ?



⇒ 1+2 ⇒ 3 bytes

sizeof(struct XYZ)

WORD size (computer world)

example – set of bytes.

– i.e 32 bit computer team "WORD" 4 byte (32/8)

64 bit computer team "WORD" 8 byte (64/8)

what is the purpose of "WORD" size?

— a standard

— When you define any structure, compiler will allocate memory bytes as near as possible to a multiple of word size

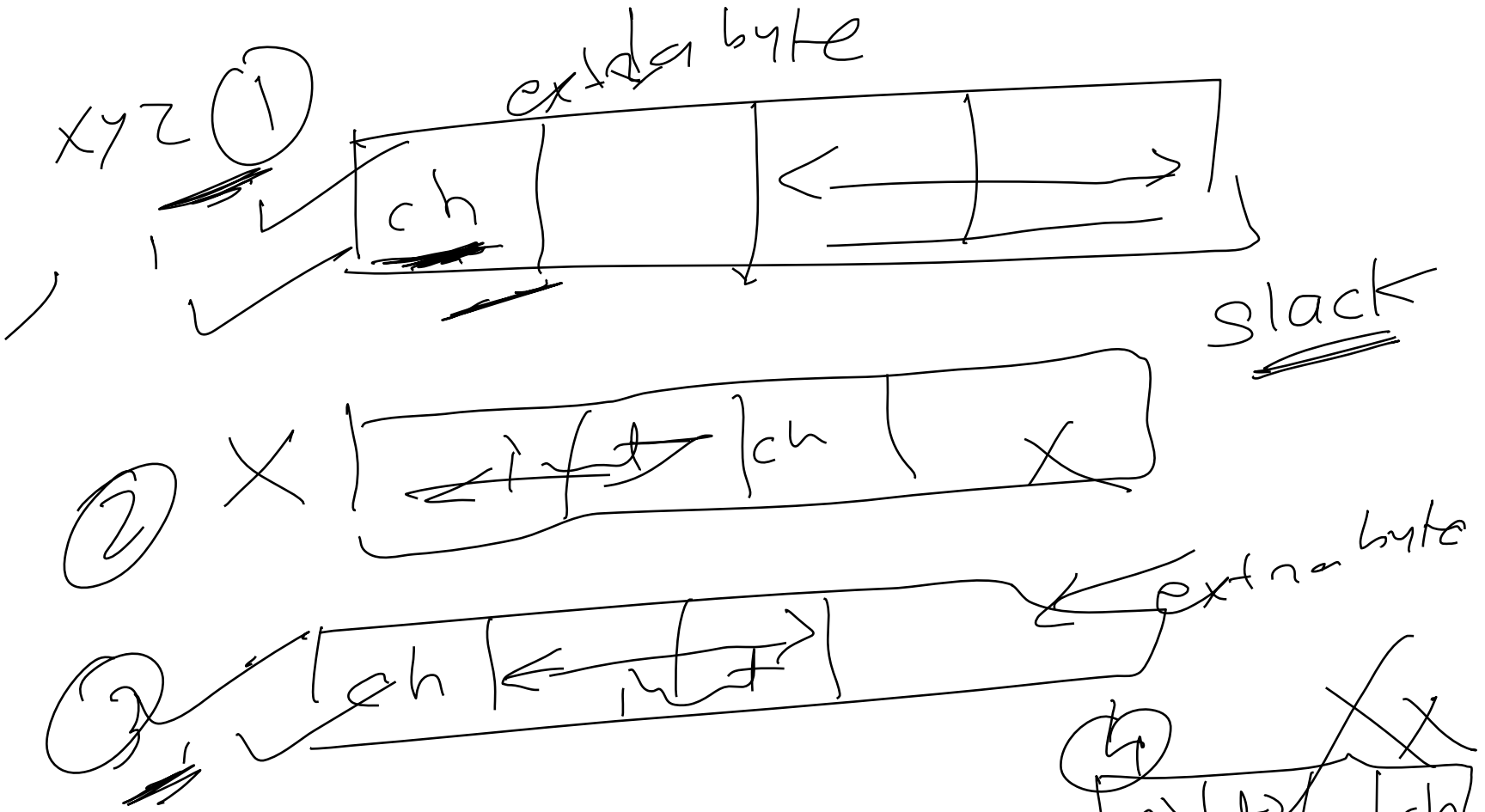
i.e. struct XYZ of 3 bytes actually. will ⇒ 4 bytes will be allocated.

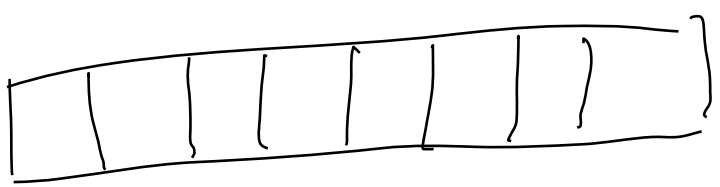
Amazon courier box size
small medium large

why they may not have boxes for every damn size?

It's not feasible.

It will not help in arranging into warehouse courier vehicle





st - pqr
{
char ch[3];
int i[2];
}

3 bytes + 4 bytes
3 x 1 2 x 2

3 + 4 => 7 bytes

i.e 32 bit 4 byte WORD size

slack bytes will be ~~organ~~ ~~initialized to~~ garbage because they are uninitialized.

```
s1 = s2; // memcpy
```

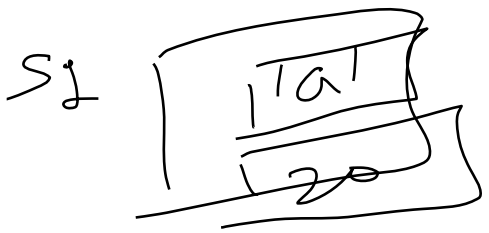
Assignment is supported

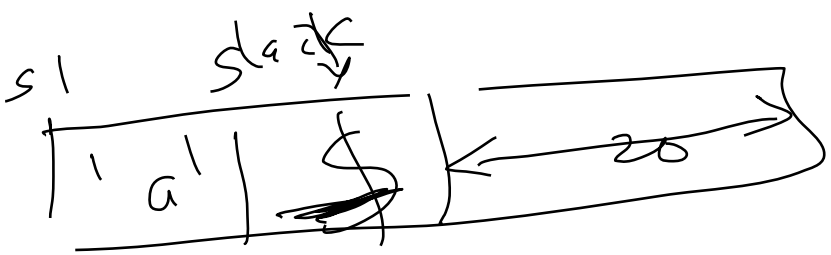
```
Struct XYZ  s1 = {'a', 20};
Struct XYZ  s2 = {'a', 20};
```

Comparison equality

s1 == s2

~~not~~ supported.





< = > <

tell me S1 and S2 are same or not ?

compare one member by member

```
if ((S1.id == S2.id) &&
    (strcmp(S1.name, S2.name) == 0) &&
    (S1.marks == S2.marks))
{
    printf("Both are same");
}
else
{
    printf("Both are not same");
}
```

for(i=0, j=0; i<n && j<m; i++, j++)
{
}

Union

- enables you to create user defined datatype
- Differs to 'structures' w.r.t. to memory allocation

```
struct xyz
{
    char ch;
    int i;
    float f;
} x;
```

sizeof(x) ⇒ 7 bytes (ch: 1, i: 2, f: 4)

```
union PQR
{
    char ch;
    int i;
    float f;
} p;
```

p → union variable

will allocate memory only for the maximum number of bytes required by any individual member

⇒ 4 bytes

union demo

{

char ch1;

char ch2;

int i;

} d;

2 bytes

can I store ch1 as well ch2?

NO

int store;

=0 || ch1

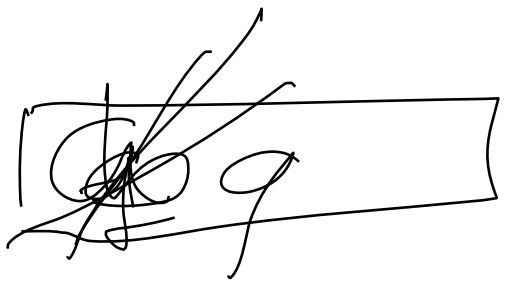
=1 || ch2

=2 || i

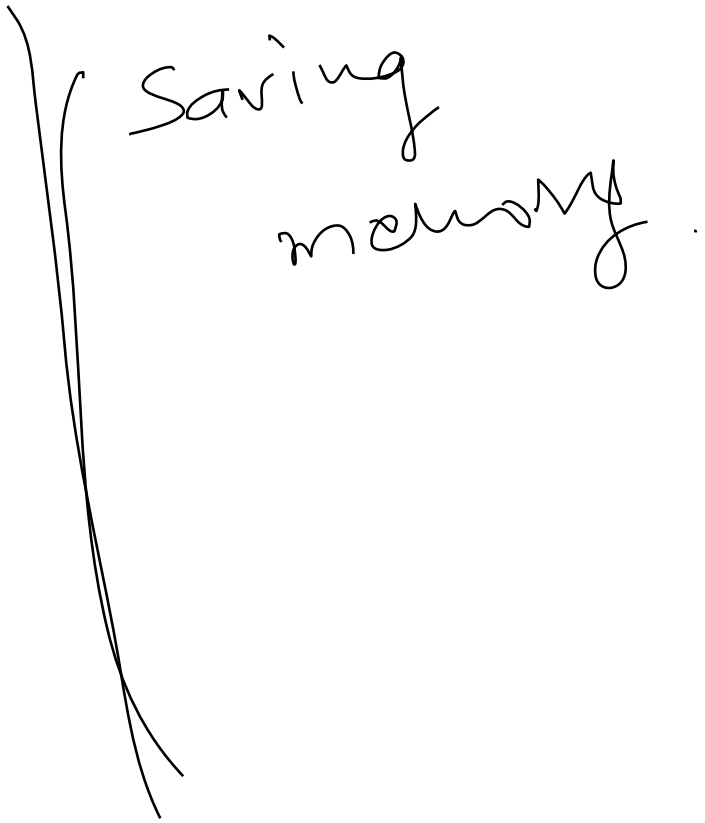
d.ch1 = 'a';

d.ch2 = 'b';

d.i = 19;



```
if (store == 0)
    printf("%c", ch1);
else if (store == 1)
    printf("%c", ch2);
else
    p__ ("%d", i1);
```



Heterogeneous Array

Hint:

✓ You can use struct & union