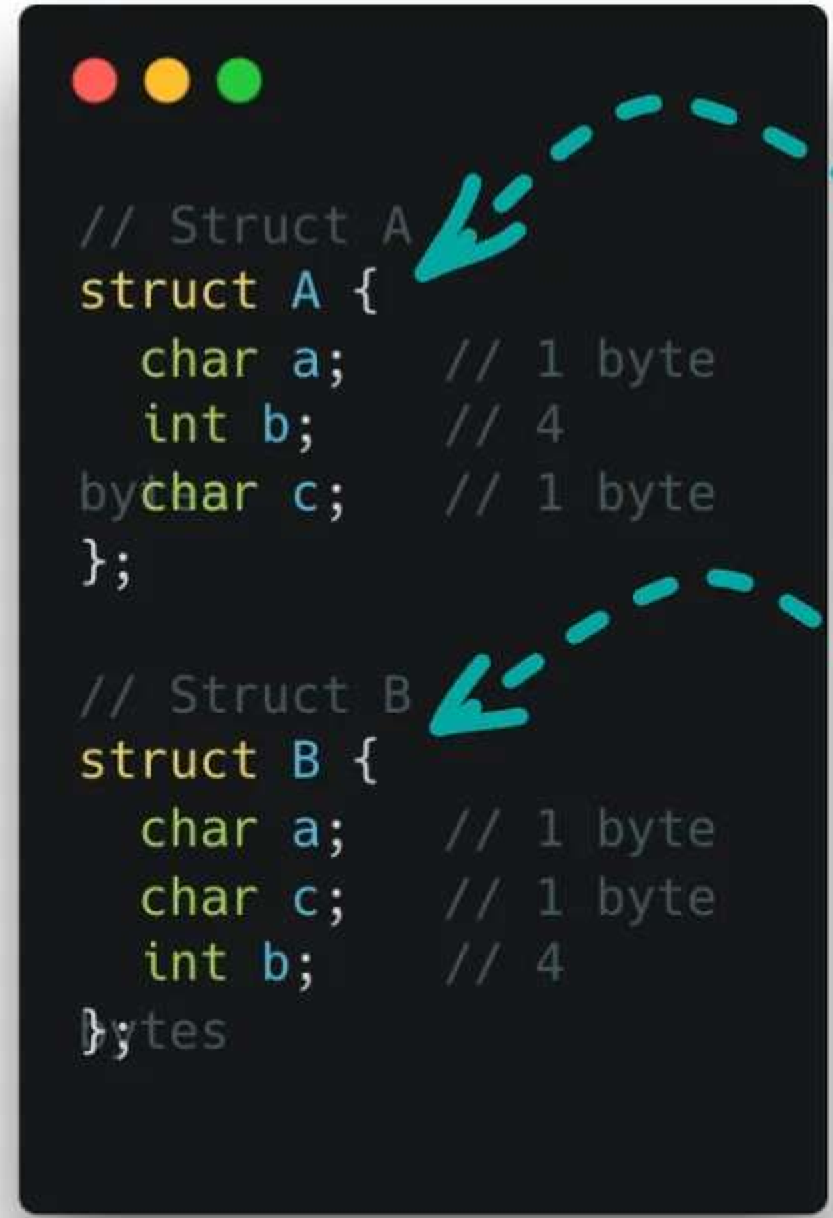
Why does order change the size? 🤯



```c
// Struct A
struct A {
  char a;    // 1 byte
  int b;     // 4
  char c;    // 1 byte
};

// Struct B
struct B {
  char a;    // 1 byte
  char c;    // 1 byte
  int b;     // 4
};
```

size = 12 bytes

size = 8 bytes

swipe $\longrightarrow$

What's Happening Here?

🧠On a 32-bit CPU,
Memory is typically accessed in 4-byte chunks (called word alignment).

That means:

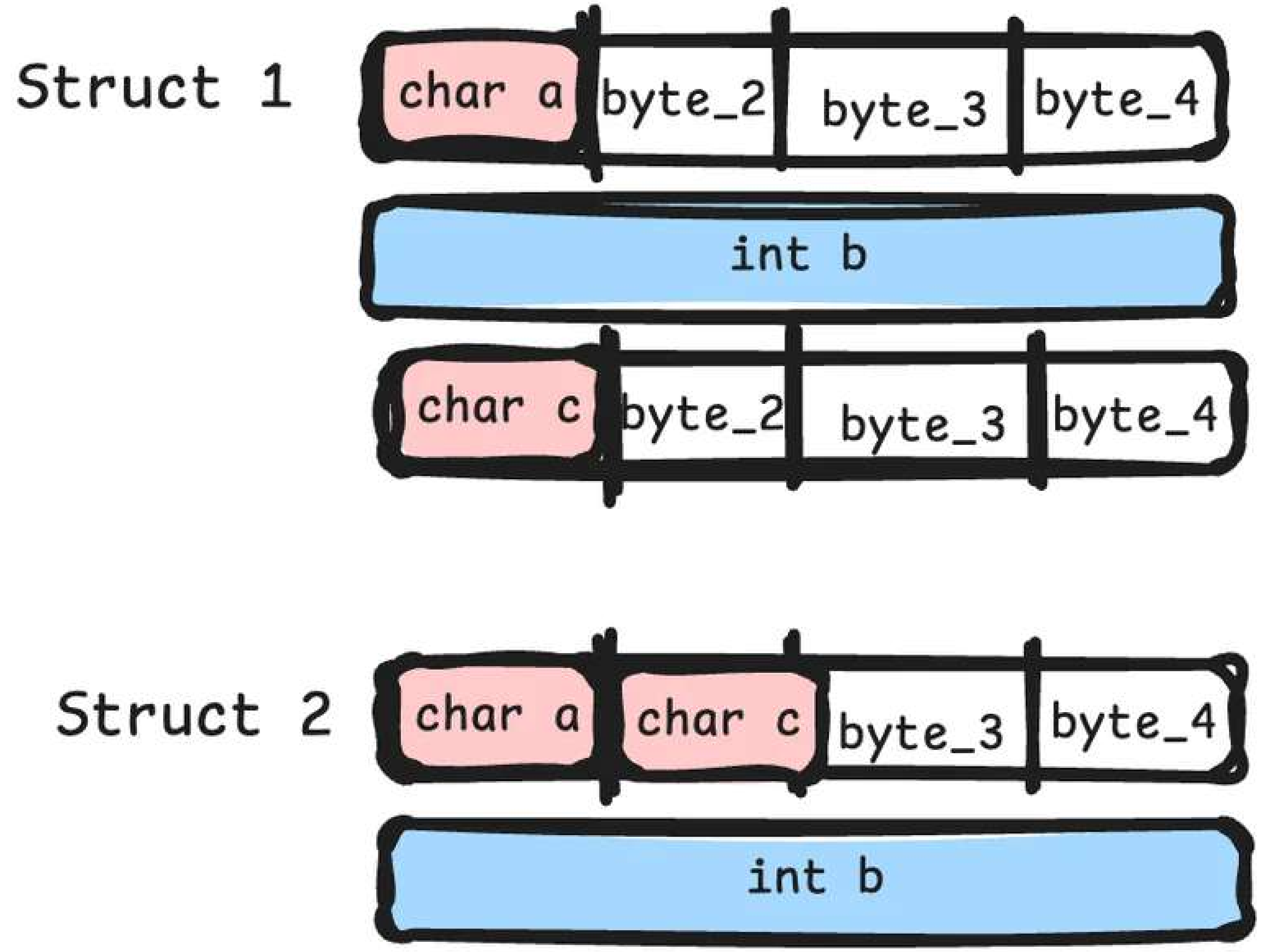
int (4 bytes) must start at an address that's a multiple of 4.

If it doesn't, the CPU runs slower.

To avoid this, the compiler adds
padding bytes between struct members.
Also known as Struct Padding.

swipe →

Lets break it down



swipe →

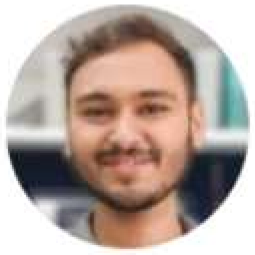
So Remember

✅ Always check your struct layout.
✅ Order members from largest to smallest.

few bytes saved in stuct =
kilobytes saved in memory.

EWskills

Practice struct padding at EWskills.com
Link in description.



Follow me for more insights!