Compilers Must Produce Working Assembly Code

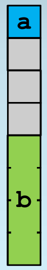
Even **ISAs that**
- **do not require aligned accesses**
- **execute unaligned accesses slowly** (sometimes as much as ~100× slower).

Compilers must produce working code.

Thus **compilers align**
- **fields to their size** (for primitive types),
- and **structures to the maximum alignment needed by any field**.

A Padding Example

Consider:
```
struct one_t {
    int8_t a;
    int32_t b;
};
```

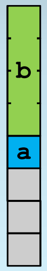


A `one_t` **must be 4-byte aligned** because of **`b`**.

After `a`, a compiler
- inserts **3 bytes of padding**
- so that **`b`** is aligned properly.

Changing Order May or May Not Affect Size

Consider:
```
struct one_t {
    int32_t b;
    int8_t a;
};
```



What if we change the order?

Same result: 8 bytes.

(Arrays of `one_t`s must have proper alignment, too.)

Field Access Operator . Accesses a Structure's Fields

The **C operator for field access is**

. (a period).

For example, given
```
struct book_t book;
```
we can write
```
book.author   // the author field
book.title    // the title field
```