http://stackoverflow.com/questions/5473189/what-is-a-packed-structure-in-c

When structures are defined, the compiler is allowed to add paddings (spaces without actual data) so that members fall in address boundaries that are easier to access for the CPU.
For example, on a 32-bit CPU, 32-bit members should start at addresses that are multiple of 4 bytes in order to be efficiently accessed (read and written). The following structure definition adds a 16-bit padding between both members, so that the second member falls in a proper address boundary:

```
struct S {
    int16_t member1;
    int32_t member2;
};
```

The structure in memory of the above structure in a 32-bit architecture is (**~** = padding):

```
+---------+---------+
| m1 |~~~~|   m2    |
+---------+---------+
```

When a structure is packed, these paddings are not inserted. The compiler has to generate more code (which runs slower) to extract the non-aligned data members, and also to write to them.
The same structure, when packed, will appear in memory as something like:

```
+---------+---------+
| m1 |   m2    |~~~~
+---------+---------+
```