L2/18-186

Feedback related to memory alignment and Unicode encoding schemes

Henri Sivonen (hsivonen@mozilla.com)

2018-05-02

When discussing the distinction between Encoding Forms and Encoding Schemes, Unicode 10.0 discusses byte order on pages 40 and 41 in section *2.6 Encoding Schemes* and on page 134 in section *3.10 Unicode Encoding Schemes*. However, neither section mentions memory alignment.

The text of the Standard is suggestive of byte order being the main issue when converting from a Unicode Encoding Scheme to the corresponding Unicode Encoding Form. This ignores that in addition to the Encoding Form being in the native-endian byte order of the computer that is processing text, the code units of the Encoding Form also need to be aligned according to the requirements of the programming language. (Even on CPU architectures that do not restrict the alignment of scalar loads and stores, e.g. the C programming language imposes alignment requirements, whose practical relevance can become visible via autovectorization on the assumption that alignment requirements were adhered to and the CPU architecture enforcing alignment for vector loads and stores.)

Therefore, when ingesting data in a non-UTF-8 Unicode Encoding Scheme using a byte-oriented IO interface, not only byte order but also memory alignment is an obstacle to interpreting a byte buffer as a buffer consisting of code units of the Unicode Encoding Form corresponding to the Unicode encoding Scheme being ingested.

It would be worthwhile to informatively point this out in the Standard in order to give readers a more comprehensive view of the information they should be aware or when evaluating the merits of the different Unicode Encoding Schemes for storage or interchange. That is, UTF-8 not only avoids the byte order issue but it also avoids the alignment issue.