Expand universal intrinsics to cover AVX-2, AVX-512 etc.

- Author: Vadim Pisarevsky
- Link: The feature request
- Status: **Draft**
- Platforms: **Intel** (and maybe other platforms with SIMD registers wider than 128 bits)
- Complexity: A few man-weeks

Introduction and Rationale

Currently OpenCV includes very convenient universal intrinsics that cover different 128-bit SIMD extensions on different platforms, such as `SSE2` (or higher) on IA, `NEON` on ARM and `VSX` on PPC64. The intrinsics implement the concept "write once - run everywhere". In addition to that, they do not require runtime dispatching, because those basic instruction sets (e.g. `SSE2` on IA) are considered as "always-available" feature.

For `AVX2` and `AVX-512` dispatching on Intel we currently use dynamic dispatcher, which is very convenient for users (no need to have a separate build of OpenCV for each platform). At the same time this technology ("one binary fits all") does not provide the best performance and the number of those dynamically dispatched code branches is rather small (and those branches need special hardware to test).

So, it would be nice to be able to build specialized versions of OpenCV where `AVX2` or `AVX-512` (or other similar instruction set) is enabled by default and correspondingly all the universal intrinsics are expanded to those actual intrinsics instead of the baseline `SSE2` etc.

Proposed solution

- It's suggested to rename all the vector types used in universal intrinsics to something size-agnostic. Currently we use `v_uint8x16` etc. types for the vector types. They can be renamed to `v_uint8xn` (or simply to `v_uint8`).
- The intrinsics themselves can stay the same.
- There are already `v_uint8x16::nlanes` etc. enumeration constants, they just need to be defined properly, depending on the actual data type, e.g. `v_uint8xn::nlanes == 16` in the case of `SSE2` and `v_uint8xn::nlanes == 32` in the case of `AVX-2`.
- The vectorized loops should be modified accordingly to increment the pointers by this `...::nlanes` instead of literal `16` etc.
- Those expanded universal intrinsics can be used together with the dynamic dispatcher, just like before - the actual expansion of universal intrinsics is defined by the `baseline` instruction set. For example, OpenCV can be built as AVX-2 library, then the universal intrinsics will be expanded as AVX-2 intrinsics. At the same time, AVX-512 branches can be dispatched dynamically if the actual hardware supports it.

Impact on existing code, compatibility

Some people may already use those SIMD128 universal intrinsics. For them we could retain the previous defines as aliases (and report a compile-time error if they are used with improper default instruction set, e.g. `-mavx2` etc.)

Possible alternatives

1. leave things as-is
2. mini-Halide, embedded into OpenCV, can be a viable alternative for regular kernels that fit Halide. For complex custom loops that need to be optimized manually Halide solution will not work.

References

1. universal intrinsics
2. dynamic CPU dispatch guide
3. AVX2 implementation of universal intrinsics