1 AVX zapiski

Avx slang:

- upper half: 127-255
- lower half: 0-127

1.1 Printing avx register

One can print avx register like this:

```
_m256d reg=_mm256_set_pd(1,2,3,4); // sets reg[255-192]=1 and reg[0-63]=4
double *p = (double*)&reg
std::cout << p[0]<<' ' << p[1]<<' ' << p[2]<<' ' << p[3] << std::endl;
double arr[4];
_mm_storeu_pd(arr, reg);
std::cout << arr[0]<<' '<< arr[1]<<' '<< arr[2]<<' '<< arr[3] << std::endl;
```

This will print 1 2 3 4. Thus, it starts at 0 -> at the lower part of register! The same happens in the second case - where we print array called *arr*.

1.2 Fast sign change and absolute value of float register

SOURCE
Changing the sign of float values using SSE code The IEEE 754 floating point format defines the memory layout for the C++ float datatype. It consists of a one bit sign, the 8 bit exponent and 23 bits that store the fractional part of the value. float x = [sign (1 bit) | exponent (8bit) | fraction (23bit)] We can use this knowledge about the memory-layout in order to change the sign of floating point values without the need for floating point arithmetic. For example, calculating the absolute value of a floating point number is equivalent to setting the sign bit to zero. In SSE we can do this for four float values simultaneously by using a binary mask and logical operations:

```
static const __m128 SIGNMASK =
               _mm_castsi128_ps(_mm_set1_epi32(0x80000000));

//absolute value
__m128 val = /* some value */;
__m128 absval = _mm_andnot_ps(SIGNMASK, val); // absval = abs(val)


// change sign
__m128 val = /* some value */;
__m128 minusval = _mm_xor_ps(val, SIGNMASK); // minusval = -val
```

2 Mask encoding

Commonly used masks are imm8 masks. The easiest way to write them is to use binary encoding. It start with *0b* and is followed by eight numbers (0 or 1). The first bit (designated as 0 in Intel intrinsics manual) is the most right one. The bits are then read on from the right side towards the left side. Right side sets the lower part of the register, left side sets the higher part of the register.

Example: reverse __m128 register:

```
/*
first two bits are 11 - which means we put the last element into the
first place of the register (designeated by index 0)

third and fourth two bits are 0 and 1 - together 10 in binary means
3- third element is put in the second place of the register .... and
so on.
*/
  _mm256_shuffle_ps(register, 0b00011011)
```