SSE/AVX

SSE/AVX

- SSE (Streaming SIMD Extension) - набор инструкций процессора, разработанный Intel для Pentium 3.
- SIMD - Single Instruction Multiple Data
- AVX (Advanced Vector Extensions) - новый набор инструкций вышедший в 2008 году.
- Данные пакеты позволяют проводить векторные операции над данными в регистрах.

Регистры

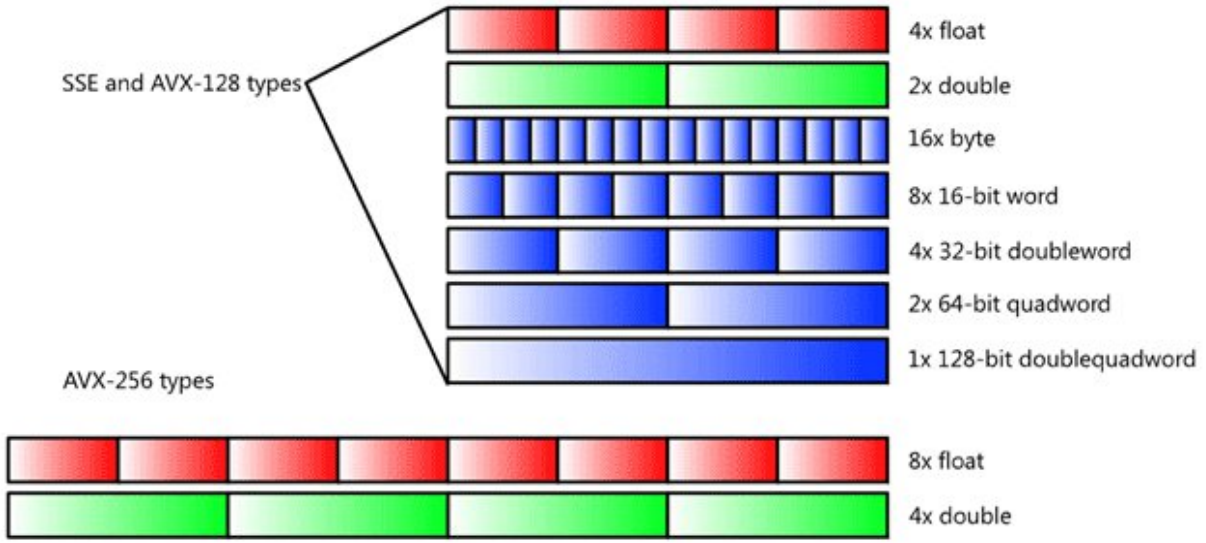
- SSE добавил 16 регистров XMM размером в 128 бит.
- AVX увеличил эти регистры XMM до YMM в два раза.
- AVX-512 Добавил еще 16 регистров, а так же увеличил их все в два раза. Название самых больших регистров ZMM.
- В нашем курсе мы будем писать на 256 битном AVX, потому что 512 мой процессор не поддерживает

x64 AVX-512 register scheme as extension from the x64 AVX (YMM0-YMM15) and x64 SSE (XMM0-XMM15) registers

511 256	255 128	127 0
ZMM0	YMM0	XMM0
ZMM1	YMM1	XMM1
ZMM2	YMM2	XMM2
ZMM3	YMM3	XMM3
ZMM4	YMM4	XMM4
ZMM5	YMM5	XMM5
ZMM6	YMM6	XMM6
ZMM7	YMM7	XMM7
ZMM8	YMM8	XMM8
ZMM9	YMM9	XMM9
ZMM10	YMM10	XMM10
ZMM11	YMM11	XMM11
ZMM12	YMM12	XMM12
ZMM13	YMM13	XMM13
ZMM14	YMM14	XMM14
ZMM15	YMM15	XMM15
ZMM16	YMM16	XMM16
ZMM17	YMM17	XMM17
ZMM18	YMM18	XMM18
ZMM19	YMM19	XMM19
ZMM20	YMM20	XMM20
ZMM21	YMM21	XMM21
ZMM22	YMM22	XMM22
ZMM23	YMM23	XMM23
ZMM24	YMM24	XMM24
ZMM25	YMM25	XMM25
ZMM26	YMM26	XMM26
ZMM27	YMM27	XMM27
ZMM28	YMM28	XMM28
ZMM29	YMM29	XMM29
ZMM30	YMM30	XMM30
ZMM31	YMM31	XMM31

Типы данных

- __m256 - packed(вектор из) float
- __m256d - packed double
- __m256i - packed int
- Так же можно использовать аналогичные типы меньшей размерности (__m128 например)



Команды AVX

- Шаблон команды: [TYPE]_[OP]_[SURFIX]
- TYPE может быть _m256 или _mm (для 128)
- OP - операция (SUB, ADD, MUL, BROADCAST и т.д.)
- SURFIX - PS (packed single-percision), PD, SS (scalar single-percision), SD, EPIN (Extended packed N-bit signed integer)
- _m256_mul_ps(__m256 A, __m256 B) - умножает каждый 32 битный float в A с соответствующим float в B и возвращает полученный результат.
- Часть команд имеет аналогичную ASM команду, часть выполняется при помощи нескольких ASM операций.
- Что бы использовать эти функции, нужно подключить x86intrin.h

Примеры команд

- _mm256_setzero_ps (void) - возвращает заполненный нулями __mm256.
- _mm256_broadcast_ss (float const *__X) - возвращает заполненный скалярным значением X __mm256.
- _mm256_load_ps (float const *__P) - копирует 8 float по указателю в __mm256 и возвращает его.
- _mm256_store_ps (float *__P, __m256 __A) - кладет по указателю P 8 float из A. Ячейки в массиве P должны быть 32-битными.
- Все команды можно найти здесь: https://software.intel.com/sites/landingpage/IntrinsicsGuide

А у меня есть AVX?

Для того что бы проверить, есть ли у Вас AVX Вы можете воспользоваться следующими командами:

- cat /proc/cpuinfo | egrep "(flags|model name|vendor)" | sort | uniq -c - выводит данные о Вашем CPU и доступные пакеты команд (там должен быть avx)
- gcc -march=native -dM -E - < /dev/null | egrep "SSE|AVX" | sort - возможности gcc. (Должен быть #define __AVX__ 1)
- uname -a - Ваша версия ядра должна быть больше 2.6.30

Автоматическая векторизация

- GCC может автоматически векторизировать некоторые циклы. Для этого необходимо указать флаг -ftree-vectorize. Он входит в флаг -O3.
- Для того что бы узнать какой цикл был векторизован можно указать флаги:
  - -fopt-info-vec-optimized - выводит какие циклы были векторизованы
  - -fopt-info-vec-all - так же выводит информацию о циклах которые не были векторизованы и почему.

Автоматическая векторизация

- Не все циклы хорошо поддаются автовекторизации:
  - Циклы с вызовами функций для которых нет векторизированного анлога.
  - Циклы с условиями.
  - Циклы с зависимостью данных (x[i] = x[i-1]).
  - Вложенные циклы.

Table 1. Functions for which the compiler has a vectorized version.

acos	ceil	fabs	round
acosh	Cos	floor	sin
asin	Cosh	fmax	sinh
asinh	erf	fmin	sqrt
atan	Erfc	log	tan
atan2	Erfinv	log10	tanh
atanh	Exp	log2	trunc
cbrt	exp2	pow	

Автоматическая векторизация

- Компилятору можно дать указания по автовекторизации:
  - #pragma loop_count(n) - подсказка компилятору, что этот цикл чаще всего выполняется n раз. Можно указывать минимальное, максимальное и среднее количество итераций.
  - #pragma vector [un]alligned - цикл должен быть векторизированым для [не]выровненных данных.
  - #pragma ivdep - игнорировать зависимость данных (мол ее здесь нет, gcc показалось)
  - #pragma vector always - gcc будет использовать векторизацию любого уровня эффективности.