Федеральноегосударственное бюджетное образовательное учреждение высшего образования. «Национально-исследовательский университет «Московскийэнергетический институт»

Кафедра ВМСС

Лабораторная работа №5 по курсу

«Вычислительные системы»

Тема: Исследование SIMD-расширений процессора

Выполнил: студент группы А-08-19 Балашов С.А.

Проверил: Карпов А.В.

Москва, 2023 г.

**Цель работы:** изучение SIMD-расширений архитектуры x86 и их применение в программах на языке C.

**Домашняя подготовка**:

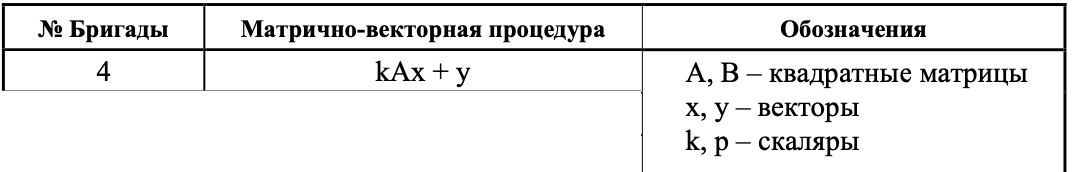
Написать программу скалярного произведения векторов с использованием встроенных функций SSE на языке C и без их использования (вектор x заполнить по правилу N+i, где N – номер группы, а вектор y – по правилу M+i, где M – номер по журналу). Сравнить время выполнения получившихся программ.

Проанализировать результаты и сделать вывод

Исходные данные представлены в таблице 1.

Таблица 1

Исходные данные



Листинг А содержит код программы. Результаты выполнения программы представлены в таблице 2.

Таблица 2

Результаты работы программы

|  |  |
| --- | --- |
| Объем массивов L | Без SSE сверху, с SSE внизу |
| 40 |  |
| 400 |  |
| 4000 |  |
| 40000 |  |
| 400000 |  |

Вывод: проанализировав результаты выполнения программы, можно сделать вывод, что ускорение вычислений при использовании функций SSE возрастает примерно в 2 раза, это достигается за счёт распараллеливания данных в регистрах и одновременного выполнения векторных и скалярных операций.

**Лабораторное задание:**

В соответствии с вариантом задания (см. Таблицу 1) написать программу на  
языке C, реализующую матрично-векторную процедуру с использованием  
расширений SSE. Размерность матриц и векторов считать кратным четырем.

Листинг Б содержит код программы. Результаты выполнения программы представлены в таблице 3.

Таблица 3

Результаты работы программы

|  |  |
| --- | --- |
| Объем массивов L | Без SSE сверху, с SSE внизу |
| 40 |  |
| 400 |  |
| 4000 |  |

Вывод: проанализировав результаты выполнения программы, можно сделать вывод, что ускорение комплексных вычислений при использовании функций SSE становится заметным при объемах около 400 и возрастает примерно в 2 раза по сравнению с обычным методом вычислений.

**Листинг А**

**#include <stdio.h>**

**// SSE (V2)**

**#include <emmintrin.h>**

**#include <time.h>**

**#define ITERATIONS 1000**

**#define N 8**

**#define M 4**

**#define k (M+N)**

**// функция без использования SSE**

**double calc\_standart(double\* x, double\* y, int n)**

**{**

**double s = 0;**

**for (int i = 0; i < n; i++)**

**{**

**s += x[i] \* y[i];**

**}**

**return s;**

**}**

**// функция с использованием SSE**

**double calc\_sse(double\* x, double\* y, int n)**

**{**

**double sum;**

**\_\_m128d\* xx, \* yy;**

**\_\_m128d p, s;**

**xx = (\_\_m128d\*)x;**

**yy = (\_\_m128d\*)y;**

**s = \_mm\_set\_pd(0, 0);**

**for (int i = 0; i < n / 2; i++)**

**{**

**p = \_mm\_mul\_pd(xx[i], yy[i]); // векторное умножение четырех чисел**

**s = \_mm\_add\_pd(s, p); // векторное сложение четырех чисел**

**}**

**p = \_mm\_shuffle\_pd(s, s, 1); // перемещение второго значения в s в младшую позицию в p**

**s = \_mm\_add\_pd(s, p); // скалярное сложение**

**\_mm\_store\_sd(&sum, s); // запись младшего значения в память;**

**return sum;**

**}**

**int main()**

**{**

**double s;**

**const int sizes[] = { 40, 400, 4000, 40000, 400000 };**

**for (int w = 0; w < sizeof(sizes) / sizeof(int); w++) {**

**int L = sizes[w];**

**printf("L:%d\n", L);**

**// выделение памяти с выравниванием**

**double\* x = (double\*)\_mm\_malloc(L \* sizeof(double), 16);**

**double\* y = (double\*)\_mm\_malloc(L \* sizeof(double), 16);**

**if (x && y) {**

**for (int i = 0; i < L; i++)**

**{**

**x[i] = N + i;**

**y[i] = M + i;**

**}**

**}**

**double time\_spent = 0.0;**

**printf("Size of arrays %d\n\n", L);**

**clock\_t begin = clock();**

**for (int itt1 = 0; itt1 < ITERATIONS; itt1++)**

**{**

**s = calc\_standart(x, y, L);**

**}**

**printf("\tResult: %.0f\n", s);**

**clock\_t end = clock();**

**time\_spent += (double)(end - begin) / CLOCKS\_PER\_SEC;**

**printf("\tThe elapsed time is %f seconds\n", time\_spent);**

**time\_spent = 0.0;**

**begin = clock();**

**// Using SSE**

**for (int itt2 = 0; itt2 < ITERATIONS; itt2++)**

**{**

**s = calc\_sse(x, y, L);**

**}**

**printf("\tResult: %.0f\n", s);**

**end = clock();**

**time\_spent += (double)(end - begin) / CLOCKS\_PER\_SEC;**

**printf("\tThe elapsed time is %f seconds\n\n", time\_spent);**

**\_mm\_free(x);**

**\_mm\_free(y);**

**}**

**return 0;**

**}**

**Листинг Б**

**#include <stdio.h>**

**#include <emmintrin.h>**

**#include <time.h>**

**#define N 8**

**#define M 4**

**#define k (M+N)**

**// «обычная» функция**

**double\* inner1(double\* x, double\* y, double\*\* A, int n)**

**{**

**double\* s;**

**int i;**

**s = (double\*)calloc(n, sizeof(double));**

**for (i = 0; i < n; ++i)**

**{**

**for (int j = 0; j < n; ++j)**

**{**

**s[i] += k \* A[i][j] \* x[i] + y[i];**

**}**

**}**

**return s;**

**}**

**// функция с использованием SSE**

**double\* inner2(double\* x, double\* y, double\*\* A, int n)**

**{**

**double\* sum;**

**sum = (double\*)calloc(n, sizeof(double));**

**int i, j;**

**\_\_m128d \*xx, \*yy, \*\*AA, p, s, kk, tm1, tm2;**

**xx = (\_\_m128d\*)x;**

**yy = (\_\_m128d\*)y;**

**AA = (\_\_m128d\*\*)A;**

**s = \_mm\_set1\_pd(0);**

**p = \_mm\_set1\_pd(0);**

**kk = \_mm\_set1\_pd(k);**

**for (i = 0; i < n; i++)**

**{**

**tm1 = \_mm\_set1\_pd(0);**

**tm2 = \_mm\_set1\_pd(0);**

**for (j = 0; j < n/2; j++)**

**{**

**p = \_mm\_mul\_pd(AA[i][j], xx[j]);**

**tm1 = \_mm\_add\_pd(tm1, p);**

**tm2 = \_mm\_add\_pd(tm2, yy[j]);**

**}**

**tm1 = \_mm\_mul\_pd(tm1, kk);**

**s = \_mm\_add\_pd(tm1, tm2);**

**p = \_mm\_shuffle\_pd(s, s, 1);**

**s = \_mm\_add\_pd(s, p);**

**\_mm\_store\_sd(&sum[i], s);**

**}**

**return sum;**

**}**

**int main()**

**{**

**const int sizes[] = { 40, 400, 4000};**

**for (int w = 0; w < 5; w++) {**

**int L = sizes[w];**

**double\* x, \* y, \*\*A, \*s, result;**

**int i, j;**

**// выделение памяти с выравниванием**

**x = (double\*)\_mm\_malloc(L \* sizeof(double), 16);**

**y = (double\*)\_mm\_malloc(L \* sizeof(double), 16);**

**A = (double\*\*)malloc(L \* sizeof(double\*));**

**for (i = 0; i < L; i++) // цикл по строкам**

**{**

**A[i] = (double\*)\_mm\_malloc(L \* sizeof(double), 16);**

**}**

**s = (double\*)calloc(L, sizeof(double));**

**if (x && y && A) {**

**for (i = 0; i < L; i++)**

**{**

**x[i] = (double)(N + i);**

**y[i] = (double)(M + i);**

**if (A[i]) {**

**for (j = 0; j < L; j++)**

**{**

**A[i][j] = (double)(N + M + i + j);**

**}**

**}**

**}**

**}**

**double time\_spent = 0.0;**

**printf("Size of arrays %d\n\n", L);**

**clock\_t begin = clock();**

**// Using x87**

**for (int itt1 = 0; itt1 < 1000; itt1++)**

**{**

**s = inner1(x, y, A, L);**

**}**

**result = 0;**

**printf("STANDART FUNCTION\n");**

**printf("\tResult:\n\t");**

**for (i = 0; i < L; i++)**

**{**

**result += s[i];**

**}**

**printf("%.0f\n", result);**

**clock\_t end = clock();**

**time\_spent += (double)(end - begin) / CLOCKS\_PER\_SEC;**

**printf("The elapsed time is %f seconds\n", time\_spent);**

**time\_spent = 0.0;**

**begin = clock();**

**// Using SSE**

**for (int itt2 = 0; itt2 < 1000; itt2++)**

**{**

**s = inner2(x, y, A, L);**

**}**

**result = 0;**

**printf("\nFUNCTION WITH SSE2\n");**

**printf("\tResult:\n\t");**

**for (i = 0; i < L; i++)**

**{**

**result += s[i];**

**}**

**printf("%.0f\n", result);**

**end = clock();**

**time\_spent += (double)(end - begin) / CLOCKS\_PER\_SEC;**

**printf("The elapsed time is %f seconds\n", time\_spent);**

**\_mm\_free(x);**

**\_mm\_free(y);**

**for (i = 0; i < L; i++)**

**{**

**\_mm\_free(A[i]);**

**}**

**free(A);**

**free(s);**

**}**

**return 0;**

**}**

**Характеристики системы**

**Компьютер:**

**Тип компьютера System manufacturer System Product Name**

**Операционная система Microsoft Windows 10 Pro**

**Пакет обновления ОС -**

**Internet Explorer 11.1.22621.0**

**Edge 44.22621.1555.0**

**DirectX DirectX 12.0**

**Имя компьютера WORLD**

**Системная плата:**

**Тип ЦП HexaCore AMD Ryzen 5 3600X, 4058 MHz (41 x 99)**

**Системная плата Asus ROG Strix X370-F Gaming (3 PCI-E x1, 3 PCI-E x16, 1 M.2, 4 DDR4 DIMM, Audio, Video, Gigabit LAN)**

**Чипсет системной платы AMD X370, AMD K17.7 FCH, AMD K17.7 IMC**

**Системная память 16315 МБ**

**DIMM3: A-Data 8 ГБ DDR4-2666 DDR4 SDRAM**

**DIMM4: A-Data 8 ГБ DDR4-2666 DDR4 SDRAM**

**Тип BIOS AMI (07/28/2020)**

**Дисплей:**

**Видеоадаптер NVIDIA GeForce RTX 2060 (6 ГБ)**

**Видеоадаптер NVIDIA GeForce RTX 2060 (6 ГБ)**

**Видеоадаптер NVIDIA GeForce RTX 2060 (6 ГБ)**

**Видеоадаптер NVIDIA GeForce RTX 2060 (6 ГБ)**

**3D-акселератор nVIDIA GeForce RTX 2060**

**Монитор Generic Monitor (SAMSUNG) [NoDB] (30361)**