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:

№ Бригады	Матрично-векторная процедура	Обозначения
4	kAx + y	A, B – квадратные матрицы x, y – векторы k, p – скаляры

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

Таблица 2

Времена выполнения функции с использованием SSE-функций и без

Объем массивов L	Без SSE, с	С SSE, с
400	0.435	0.216
4000	43.156	20.672

Результаты выполнения программы представлены в таблице 3.

Таблица 3

Результаты работы программы

Объем массивов L	Выполнение
4	Result: 27688.000000 The elapsed time is 0.000000 seconds Result: 27688.000000 The elapsed time is 0.000000 seconds
400	Result: 189375676416.000000 The elapsed time is 0.435000 seconds Result: 189374660608.000000 The elapsed time is 0.216000 seconds
4000	Result: 1795214322696192.000000 The elapsed time is 43.156000 seconds Result: 1802152573927424.000000 The elapsed time is 20.672000 seconds

Вывод: проанализировав результаты выполнения программы, можно сделать вывод, что ускорение вычислений при использовании функций SSE возрастает примерно в 2 раза, однако при этом возможна потеря точности из-за особенностей хранения типа данных float.

Листинг А

```
#include <stdio.h>
#include <xmmintrin.h>
#include <time.h>
#define N 8
#define M 4
#define L 4000
#define k (M+N)
// «обычная» функция
float inner1(float* x, float* y, float* A, int n)
{
    float s;
    int i;
    s = 0;
    for (i = 0; i < n; i++)
    {
        for (int j = 0; j < n; ++j)
        {
            s += k * A[i * L + j] * x[i] + y[i];
        }
    }
    return s;
}
// функция с использованием SSE
float inner2(float* x, float* y, float* A, int n)
{
    float sum;
    int i;
    __m128* xx, * yy, * AA;
    __m128 p, s, kk, tm1, tm2;
    xx = (__m128*)x;
    yy = (__m128*)y;
    AA = (__m128*)A;
    s = _mm_set_ps1(0);
    kk = _mm_set_ps1(k);
    for (i = 0; i < n / 4; i++)
    {
        for (int j = 0; j < n; j++)
        {
            tm1 = _mm_mul_ps(xx[i], AA[i * L + j]); // векторное умножение четырех
чисел
            tm2 = _mm_mul_ps(tm1, kk);
            p = _mm_add_ps(tm2, yy[i]);      // векторное сложение четырех чисел
            s = _mm_add_ps(s, p);
        }
    }
    p = _mm_movehl_ps(p, s);    // перемещение двух старших значений s в младшие p
    s = _mm_add_ps(s, p);               // векторное сложение
    p = _mm_shuffle_ps(s, s, 1);// перемещение второго значения в s в младшую
позицию в p
    s = _mm_add_ss(s, p);               // скалярное сложение
    _mm_store_ss(&sum, s);              // запись младшего значения в память;
    return sum;
}
```

```
int main()
{
    float* x, * y, * A, s;
    long t;
    int i;
    // выделение памяти с выравниванием
    x = (float*)_mm_malloc(L * sizeof(float), 16);
    y = (float*)_mm_malloc(L * sizeof(float), 16);
    A = (float*)_mm_malloc(L * L * sizeof(float), 16);
    for (i = 0; i < L; i++)
    {
        x[i] = N + i;
        y[i] = M + i;
        for (int j = 0; j < L; ++j)
        {
            A[i * L + j] = N + M + j + i;
        }
    }
    double time_spent = 0.0;
    clock_t begin = clock();
    // Using x87
    for (int itt1 = 0; itt1 < 1000; itt1++)
    {
        s = inner1(x, y, A, L);
    }
    printf("Result: %f\n", s);
    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);
    }
    printf("Result: %f\n", s);
    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);
    return 0;
}
```