Лабораторная работа 2

Комментарии

1. В отчет включены:
   1. Уточненная формулировка задачи
   2. Форматы входных и выходных данных (с примерами)
   3. Словесное описание алгоритма или идеи
2. Все процедуры описаны в файлах Cpp.cpp и Architecture2\_mmx.cpp проекта.

# Подсчет суммы элементов ряда для функции

## Уточненная постановка задачи

Разработать процедуру вычисления функции путем подсчета суммы первых N членов ряда Тэйлора.

C:\Users\Михаил_2\Desktop\Безымянный5.png

## Вход

Значения Х и N. Пример

0.2 100

## Выход

2.60718

## Алгоритм

Из формулы общего члена ряда видно, что xn и n! Могут быть вычислены рекуррентно.

В цикле от 0 до N

1. Вычислить cos(i\*x)
2. Разделить cos(i\*x) на i!
3. Добавить результат к сумме
4. Рекуррентно вычислить I, i\*x и i! путем прибавления соответственно 1, х и i

# Подсчет суммы элементов ряда для функции

## Уточненная постановка задачи

Разработать процедуру вычисления значения функции с заданной точностью epsilon.

C:\Users\Михаил_2\Desktop\Безымянный5.png

## Вход

Значения Х и Eps. Пример

0.2 0.001

## Выход

2.61222

## Алгоритм

Из формулы общего члена ряда видно, что xn и n! Могут быть вычислены рекуррентно.

В цикле

1. Вычислить cos(i\*x)
2. Разделить cos(i\*x) на i!
3. Добавить результат к сумме
4. Вычислить разность текущего и предыдущего элементов ряда Тэйлора
5. Если модуль разности <= epsilon, выход из цикла
6. Рекуррентно вычислить I, i\*x и i! путем прибавления соответственно 1, х и i

# Операции с массивами с использованием MMX

## Уточненная постановка задачи

Разработать процедуры для сложения, вычитания, деления, умножения и сравнения для массивов произвольной длины и типов данных.

## Вход

Количество элементов в массивах N, сами элементы массивов через пробел, степень двойки. Пример

10  
1 2 3 4 5 6 7 8 9 10

10 9 8 7 6 5 4 3 2 1

2

## Выход

Add:

11 11 11 11 11 11 11 11 11 11 11

Add with saturation:

11 11 11 11 11 11 11 11 11 11 11

Sub:

-9 -7 -5 -3 -1 1 3 5 7 9

Sub with saturation:

-9 -7 -5 -3 -1 1 3 5 7 9

Mul first:

4 8 12 16 20 24 28 32 36 40

Div first

0 0 0 1 1 1 1 2 2 2

Is equivalent?

0

Is greater?

0

## Алгоритм

С помощью перегрузки функций реализована работа почти со всеми возможными типами массивов для рассматриваемых операций.

Имеется универсальный шаблон (см. след. Стр.), с помощию которого реализована обработка массивов различных типов.

## Используемые инструкции ММХ

Для каждой из инструкций (кроме пересылки, упаковки и emms) имеется собственная перегруженная процедура.

1. Подготовка swr: emms
2. Пересылка: movq, movb
3. Упаковка: packswb, pascssd
4. Сложение: paddb, paddw, paddd, paddsb, paddsw, paddusb, paddusw
5. Вычитание: psubb, psubsw, psubd, pusbsb, psubsw, psubusb, psubusw
6. Умножение: psllw, pslld
7. Деление: psraw, psrad, psrlw, psrld
8. Сравнение на равенство: pcmpeqb, pcmpeqw, pcmpeqd
9. Сравнение «больше»: pcmpgtb, pcmpgtw, pcmpgtd

Юнит-тесты находятся в исходном коде

**Шаблон функции** для обработки массива. Бирюзовым помечены операторы и константы, зависимые от размера элементов массива и желаемой функциональности.

|  |  |
| --- | --- |
| Мнемоника | Комментарий |
| mov ecx, len  shr ecx, 1  jz lenLes8 | Загрузка длины массива  Вычисление количества проходов с использованием ММХ\*  Если элементов слишком мало для ММХ, переход на цепочки |
| mov esi, [sec]  mov edi, [fir]  cyc:  movq mm0, qword ptr[esi]  movq mm1, qword ptr[edi]  paddd mm0, mm1  movq qword ptr[edi], mm0  add esi, 8  add edi, 8  loop cyc | Загрузка указателей на массивы  Загрузка данных из массивов в регистры ММХ  Вычисление (инструкция зависит от задачи обработки массива) Выгрузка элементов в память (в первый операнд)  Сдвиг указателей на следующие 8 байтов массива |
| lenles8:  mov ecx, len  and ecx, 1  jz ext  cyc2:  lodsd  add eax, [edi]  stosd  loop cyc2 | Загрузка длины массива  Вычисление количества остаточных цепочечных проходов\*\*  Если все элементы уже обработаны ММХ – выход  Загрузка элемента в EAX, AX или AL  Вычисление  Выгрузка в память |
| ext: emms | Инициализация SWR |

\*Числовое значение непосредственной константы зависит от типа. Для char = 3, short = 2, int =1.

\*\*Числовое значение – битовая маска - также зависит от типа. Для char = 7, для short = 3, для int = 1.

**Пример функции** для арифметического сложения массивов fir и sec типа int.

|  |
| --- |
| void Add(signed int \*fir, signed int \*sec, size\_t len) {  \_asm  {  mov ecx, len  shr ecx, 1  jz lenLes8  mov esi, [sec]  mov edi, [fir]  cyc: movq mm0, qword ptr[esi]  movq mm1, qword ptr[edi]  paddd mm0, mm1  movq qword ptr[edi], mm0  add esi, 8  add edi, 8  loop cyc  lenles8: mov ecx, len  and ecx, 1  jz ext  cyc2: lodsd  add eax, [edi]  stosd  loop cyc2  ext: emms  }} |