Министерство образования Республики Беларусь

Учреждение Образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа №6

по дисциплине «Программирование на языке ассемблера»

на тему «Интерфейс с языками высокого уровня. Работа с математическим сопроцессором»

вариант №4

|  |  |  |
| --- | --- | --- |
| Выполнил  студент гр. 250502  Грибовская А.А. |  | Проверил  Туровец Н.О. |

Минск 2023

**Цель работы:** Ознакомиться с вариантами внедрения ассемблерной процедуры в программу, написанную на языке программирования C\C++, изучить архитектуру математического сопроцессора и команды работы с ним.

**Теоретические сведения**

Написание программы полностью на языке ассемблера допустимо только для небольших программ. На практике используют совмещенные варианты создания программ, которые требуют сочетания ассемблера и более высоких языков программирования:

-- основная часть программы пишется на языке высокого уровня, а на ассемблере пишутся отдельные процедуры, которые должны осуществлять управление нижнего уровня и(или) иметь высокую производительность;

-- ассемблерная программа использует библиотечные средства языков высокого уровня.

Для выполнения работы требуется рассмотреть следующие элементы языка ассемблера и операционной системы:

1. Соглашения об объединении программных модулей. Связь ассемблерных модулей с языками высокого уровня требует следующих соглашений, которые сильно зависят от применяемых компиляторов и операционной системы:

-- Согласование вызовов. Вызов процедуры и возврат из нее в головную программу должны быть согласованы друг с другом.

-- Согласование имен. Согласование имен требуется для того, чтобы компоновщик мог собрать исполняемый модуль.

-- Согласование параметров.

Язык С использует следующие правила формирования параметров:

-- параметры помещаются в стек в порядке обратном их записи в списке параметров;

-- удаление параметров из стека выполняет вызывающая программа.

2. Встроенный ассемблер.

Встроенный ассемблер – вставка ассемблерного кода непосредственно в

код программы на языке высокого уровня. Использование встроенного ассемблера позволяет создавать программы более быстро, используя небольшие фрагменты кода без выполнения выше изложенных требований по сборке проекта.

Любую ассемблерную команду можно записать в виде:

аsm код\_операции операнды ;

-- asm – оператор встроенной команды ассемблера (для компиляторов C++ от Microsoft используется ключевое слово \_asm);

-- код\_операции – команду языка ассемблера (например, mov); -- операнды – операнды команды (например, ax, bx). Если с помощью одного слова asm необходимо задать много ассемблерных команд, то они заключаются в фигурные скобки. Комментарии можно записывать только в форме, принятой в языке С++. В программе на языке С++, использующей ассемблерные команды, иногда необходимо задать директиву #pragma inline – эта директива сообщает компилятору, что программа содержит внутренний ассемблерный код, что важно при оптимизации программы.

3. Работа с математическим сопроцессором. В процессорах Intel операции с плавающей запятой выполняет специальный математический сопроцессор (FPU), который имеет собственные регистры и собственный набор команд.

Сопроцессор выполняет все вычисления в 80-ти битном расширенном формате, а 32-х и 64-х битные числа используются для обмена данными с основным процессором и памятью.

В математическом сопроцессоре есть следующие регистры:

-- регистры данных (R0 – R7) – не доступны по именам, а рассматриваются как стек, вершина которого называется ST(0) или просто ST, а следующие элементы – ST(1), ST(2) и т.д. до ST(7).

-- регистр состояний SR.

Важные флаги регистра состояний:

- С3 – С0 – результат выполнения предыдущей команды, используются для условных переходов;

- ТОР – номер регистра данных, который в настоящий момент является вершиной стека.

- ES – общий флаг ошибки;

- SF – ошибка стека;

- UE – флаг антипереполнения;

- ОЕ – флаг переполнения;

- ZE – флаг деления на ноль;

- IE – флаг недопустимой операции.

-- регистр управления CR

- RC – управление округлением

- PC – управление точностью результатов команд (FADD, FSUB, FSUBR, FMUL, FDIV, FDIVR и FSQRT);

-- регистр тегов TW – описывает текущее состояние каждого регистра данных;

-- регистры FIP и FDP содержат адрес последней выполненной команды и адрес ее операнда соответственно.

Команды математического сопроцессора делят на следующие группы:

-- команды управления сопроцессором;

-- команды пересылки данных;

-- команды базовой арифметики;

-- команды сравнения (основные);

-- трансцендентные операции.

**Код программы (.cpp)**

#include "stdio.h"

#include "windows.h"

#define SIZE 10

float array[SIZE];

void inputArray();

void outputArray();

void asmAlgorithm();

int main() {

inputArray();

printf("Input array: \n");

outputArray();

asmAlgorithm();

printf("\nResult array: \n");

outputArray();

return 0;

}

void inputArray() {

int res;

printf("Input 10 elements: \n");

for (int i = 0; i < SIZE; ++i) {

do {

printf("%d) ", i);

res = scanf\_s("%f", &array[i]);

while (getchar() != '\n');

if (res != 1 || array[i] == 0) printf("Invalid input\n");

} while (res != 1 || array[i] == 0);

}

}

void outputArray() {

for (int i = 0; i < SIZE; ++i) {

printf("%d) %.5f\n", i, array[i]);

}

}

void asmAlgorithm() {

\_\_asm {

finit ; Инициализация сопроцессора FPU.

xor ecx, ecx ; Обнуление регистра ecx (используется для управления циклом).

mov ecx, SIZE ; Инициализация ecx размером массива.

lea ebx, array ; Загрузка адреса массива array в регистр ebx.

calculate\_loop:

fld1 ; Загрузка значения 1.0 в стек FPU.

fdiv [ebx] ; Выполнение деления 1.0 на x (значение в массиве).

fstp [ebx] ; Сохранение результата обратно в массив.

add ebx, 4 ; Сдвиг адреса на следующий элемент массива (размер элемента float).

loop calculate\_loop ; Повторение цикла SIZE раз, уменьшая ecx на каждой итерации.

fwait ; Дожидаемся завершения всех операций FPU.

}

}

**Вывод программы**

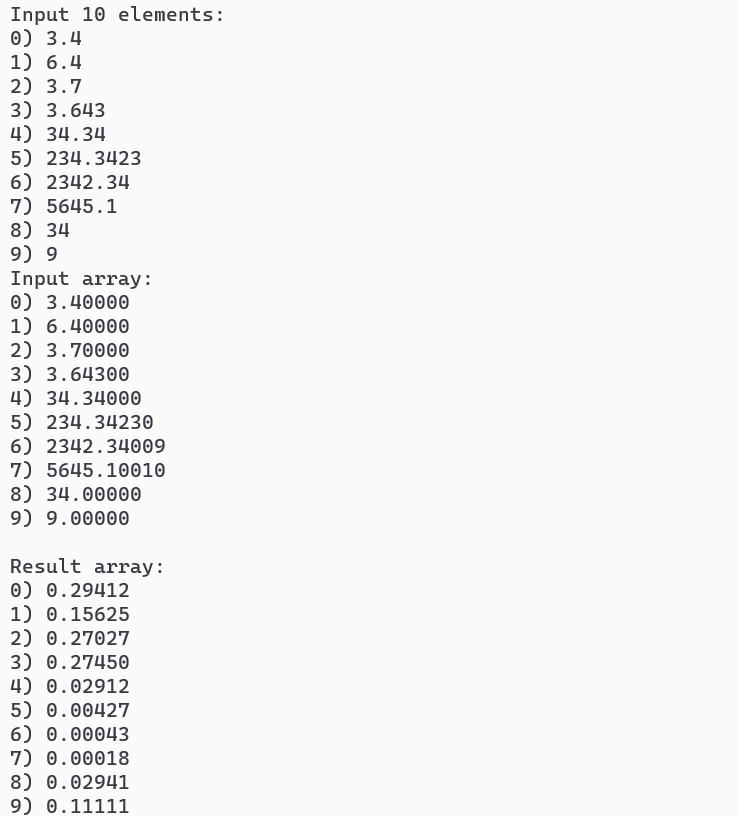


Рисунок 1 – Результат работы программы