|  |  |  |
| --- | --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования  Белорусский Государственный Университет Информатики и Радиоэлектроники | | |
| Факультет компьютерных систем и сетей | | |
| Кафедра электронных вычислительных средств | | |
| Дисциплина: Программирование компьютеров | | |
| Отчет по лабораторной работе №3  «Математический сопроцессор FPU»  Вариант №8 | | |
| Выполнили:  студенты гр. 250701  Ермакович В.А.,  Зубков Н.В. |  | Проверил:  Порхун М. И. |
| Минск 2025 | | |

# Цель работы

Цели лабораторной работы:

1) ознакомиться с архитектурой математического сопроцессора (FPU) и командами обработки чисел с плавающей запятой;

2) разработать ассемблерную вставку в высокоуровневый код для обработки данных с плавающей запятой.

# Задание

## 2.1 Задание 1

В данном задании необходимо разработать ассемблерную вставку с использованием команд математического сопроцессора FPU, которая выполняет в соответствии с индивидуальным вариантом вычисление арифметического выражения. Параметры x1 и x2 – числа типа float и должны вводиться с клавиатуры. Сравнить результат работы ассемблерной вставки с результатом вычислений в MATLAB/Си.

Выражение для первого задания представлено на рисунке 1

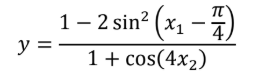


Рисунок 1 – Выражение для первого задания

## 2.2 Задание 2

В данном задании необходимо разработать ассемблерную вставку с использованием команд математического сопроцессора FPU, которая выполняет в соответствии с индивидуальным вариантом обработку массива. Элементы массива – числа типа float. Размер массива и его элементы должны вводиться с клавиатуры. Сравнить результат работы ассемблерной вставки с результатом вычислений в MATLAB/Си.

Разделить каждый элемент массива на значение косинуса первого элемента.

## 2.3 Задание 3

В данном задании необходимо разработать ассемблерную вставку с использованием команд математического сопроцессора FPU, которая выполняет в соответствии с индивидуальным вариантом один из алгоритмов ЦОС. Результат обработки записать в WAV-файл. Сравнить результат работы ассемблерной вставки с результатом вычислений в MATLAB/Си.

ЭЧМ сигнал. Длительность сигнала T = 3 секунды. Частота f0 = 500 Гц, а частота f1 = 6000 Гц. Частота дискретизации fs = 16 кГц. Амплитуда A = 10 Гц.

# Ход работы

В ходе выполнения лабораторной работы была составлена программа реализующие данные задания

Листинг программы приведён ниже:

#include "func.h"

#include <iostream>

#include <string>

#include <stdexcept>

#include <limits>

#define T 3.0 // Длительность сигнала (сек)

#define F0 500.0 // Начальная частота (Гц)

#define F1 6000.0 // Конечная частота (Гц)

#define FS 16000.0 // Частота дискретизации (Гц)

#define A 10.0 // Амплитуда (Гц)

#define PI 3.14159265358979323846

typedef struct {

char chunkID[4];

uint32\_t chunkSize;

char format[4];

char subchunk1ID[4];

uint32\_t subchunk1Size;

uint16\_t audioFormat;

uint16\_t numChannels;

uint32\_t sampleRate;

uint32\_t byteRate;

uint16\_t blockAlign;

uint16\_t bitsPerSample;

char subchunk2ID[4];

uint32\_t subchunk2Size;

} WavHeader;

using namespace std;

void clearScreen() {

system("cls || clear");

}

void writeWavFile(const char\* filename, const double\* samples, size\_t numSamples) {

WavHeader header;

memcpy(header.chunkID, "RIFF", 4);

memcpy(header.format, "WAVE", 4);

memcpy(header.subchunk1ID, "fmt ", 4);

header.subchunk1Size = 16;

header.audioFormat = 1; // PCM

header.numChannels = 1;

header.sampleRate = (uint32\_t)FS;

header.bitsPerSample = 16;

header.byteRate = header.sampleRate \* header.numChannels \* header.bitsPerSample / 8;

header.blockAlign = header.numChannels \* header.bitsPerSample / 8;

memcpy(header.subchunk2ID, "data", 4);

header.subchunk2Size = (uint32\_t)(numSamples \* header.bitsPerSample / 8);

header.chunkSize = 36 + header.subchunk2Size;

FILE\* file;

fopen\_s(&file, filename, "wb");

if (!file) {

perror("Ошибка создания файла");

return;

}

// Запись заголовка

fwrite(&header, sizeof(WavHeader), 1, file);

// Запись аудиоданных

for (size\_t i = 0; i < numSamples; i++) {

float sample = samples[i];

// Ограничение амплитуды

if (sample > 1.0f) sample = 1.0f;

if (sample < -1.0f) sample = -1.0f;

int16\_t intSample = (int16\_t)(sample \* 32767.0f);

fwrite(&intSample, sizeof(int16\_t), 1, file);

}

fclose(file);

}

int main() {

size\_t size = sizeof(int);

setlocale(LC\_ALL, "Russian");

int choice;

string input;

do {

clearScreen();

cout << "\n\t МЕНЮ ПРОГРАММЫ \n"

<< "\n\t 1. Задание 1 "

<< "\n\t 2. Задание 2 "

<< "\n\t 3. Задание 3 "

<< "\n\t 4. Дополнительное "

<< "\n\t 5. Выход "

<< "\n\nВыбор > ";

getline(cin, input);

try {

choice = stoi(input);

if (choice < 1 || choice > 5) {

throw out\_of\_range("");

}

switch (choice) {

case 1:

task1();

break;

case 2:

task2();

break;

case 3:

task3();

break;

case 4:

dop();

break;

case 5:

if (confirmExit()) return 0;

continue;

}

// Пауза после выполнения задания

cout << "\nОперация завершена. Нажмите Enter...";

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

catch (const invalid\_argument&) {

cout << "\nОшибка! Введите число от 1 до 5";

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

catch (const out\_of\_range&) {

cout << "\nНеверный пункт меню! Допустимые значения: 1-5";

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

} while (true);

}

bool confirmExit() {

clearScreen();

cout << "\nВы уверены? (y/n): ";

string answer;

getline(cin, answer);

return !answer.empty() && (tolower(answer[0]) == 'y');

}

void task1() {

clearScreen();

cout << "Посчитайте значение функции\nВведите значения для x1 и x2\n";

float x1 , x2 , y;

float pi = 3.141592653589793;

int one = 1, two = 2, four = 4;

std::cout << "Введите первое число (x1): ";

std::cin >> x1;

std::cout << "Введите второе число (x2): ";

std::cin >> x2;

//y = (1-2sin(x1-pi/4)^2)/(1+cos(4\*x2))

\_asm {

finit

fld x1 //st(0) = x1

FLDPI // st(0) = pi, st(1) = x1

fild four // st(0) = 4, st(1) = pi, st(2) = x1

fdivp ST(1), ST(0) //st(0) = pi/4, st(1) = x1

fsubp ST(1), ST(0) // ST(0) = x1 - pi/4

fsin //ST(0) =sin(x1-pi/4)

fmul ST(0), ST(0) //ST(0) =sin(x1-pi/4)^2

fild two //st(0) = 2, st(1) = sin(x1-pi/4)^2

fmulp ST(1), ST(0)//st(0) = 2\*sin(x1-pi/4)^2

fild one // st(0)=1, st(1) = 2\*sin(x1-pi/4)^2

fxch // st(0) = 2\*sin(x1-pi/4)^2, st(1)=1

fsubp ST(1), ST(0) //st(0)=1 - 2sin(x1 - pi / 4) ^ 2

fld x2 // st(0)=x2,st(1)=1 - 2sin(x1 - pi / 4) ^ 2

fild four // st(0)=4,st(1)=x2,st(2)=1 - 2sin(x1 - pi / 4) ^ 2

fmulp ST(1), ST(0) // st(0) = x2\*4, st(1)=1 - 2sin(x1 - pi / 4) ^ 2

fcos //st(0) = cos(x2\*4), st(1)=1 - 2sin(x1 - pi / 4) ^ 2

fild one //st(0)= 1,st(1) = cos(x2\*4), st(2)=1 - 2sin(x1 - pi / 4) ^ 2

faddp ST(1), ST(0) // st(0) = 1+cos(4\*x2), ST(1) = 1 - 2sin(x1 - pi / 4) ^ 2

fdivp ST(1), ST(0)

fstp y

}

cout << " Результат FPU y = " << y << endl;

float y\_Cpp = (1 - 2 \* pow(sin(x1 - pi / 4),2)) / (1 + cos(4 \* x2));

cout << " Результат C++ y = " << y\_Cpp << endl;

system("pause");

}

void task2() {

clearScreen();

int array\_size;

cout << "Разделить каждый элемент массива на значение косинуса первого элемента.\nВведите размер массива";

std::cout << "Размер массива: ";

std::cin >> array\_size;

float\* array = new float[array\_size];

float\* array\_cpp = new float[array\_size];

std::copy(array\_cpp, array\_cpp + array\_size, array);

for (short i = 0; i < array\_size; i++)

cin >> array[i];

for (short i = 0; i < array\_size; i++)

cout << "array[" << i << "] = " << array[i] << endl;

float cos\_val;

\_asm {

mov ECX, array\_size // Загружаем размер массива

test ECX, ECX // Проверяем на пустой массив

jz done // Выход если размер 0

mov EDI, array // заноси в EDI адрес массива

finit // Инициализация FPU

fld dword ptr[EDI] // st(0)=array[n]

fcos //st(0)=cos(array[0])

fstp dword ptr[cos\_val] //сохраняем значение косинуса

process\_loop:

fld dword ptr[EDI] // st(0)=array[n]

fdiv dword ptr[cos\_val] // st(0) = array[n]/cos(array[0])

fstp dword ptr[EDI] // Сохраняем результат обратно

add EDI, 4 // Переход к следующему элементу

loop process\_loop // Повторяем для всех элементов

ffree ST(0) // Очищаем регистры FPU

ffree ST(1)

done:

}

cout << "\nРезультат FPU:" << endl;

for (short i = 0; i < array\_size; i++)

cout << "array[" << i << "] = " << array[i] << endl;

for (short i = 0; i < array\_size; i++)

array\_cpp[i] = array\_cpp[i] / cos\_val;

cout << "\nРезультат C++:" << endl;

for (short i = 0; i < array\_size; i++)

cout << "array[" << i << "] = " << array[i] << endl;

delete[] array;

delete[] array\_cpp;

system("pause");

}

void task3() {

clearScreen();

double fs = 16000;

int chirpSamples = (int)(T \* fs);

float\* chirp = (float\*)malloc(chirpSamples \* sizeof(float));

double k = pow(F1 / F0, 1 / T);

double lnk = log(k);

const double fs\_inv = 1.0 / FS; // 1/fs для оптимизации

const double two\_pi = 2.0 \* PI; // 2\*PI

const double f0 = F0;

const double a = A;

double result = 0;

\_\_asm {

push ebx

push edi

push esi

mov edi, chirp

xor esi, esi

finit

p0 :

cmp esi, [chirpSamples] // Проверка завершения цикла

jge p1

mov ebx, esi

push ebx

fild dword ptr[esp] // st(0) = i

fld[fs\_inv] // st(0) = 1/fs, st(1) = i

fmulp st(1), st(0) // st(0) = t = i/fs

fld st(0)

fld[k]

fyl2x

fld1

fld st(1)

fprem

f2xm1

fadd

fscale

fxch st(1)

fstp st // st(0) = k^t

fld1

fsub

fdiv[lnk] // st(0) = (k ^ t - 1)/ ln(k)

fmul[f0] // st(0) = st(0) \* f0

fmul[two\_pi] // st(0) = st(0) \* 2 \* PI

fsin

fmul[a]

fstp dword ptr[edi + esi \* 4]

ffree st(0)

add esp, 4

inc esi

jmp p0

p1 :

fwait

pop esi

pop edi

pop ebx

}

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

cout << chirp[i] << ",";

}

system("pause");

}

void dop() {

clearScreen();

cout << "Посчитайте значение функции\nВведите значения для y и x\n";

float x, y, R;

std::cout << "Введите первое число (x): ";

std::cin >> x;

std::cout << "Введите второе число (y): ";

std::cin >> y;

// cos(x)\*sqrt(sin(y)\*x^y)) / (y \*log2(x+1))+exp(y)-tan(x/y)-2^(x-1)

\_asm{

finit

fld x // st(0) = x

fcos // st(0) = cos(x)

fld y // st(0) = y, st(1)=cos(x)

fld x // st(0) =x, st(1)= y, st(2) = cos(x)

FYL2X // ST(0) = y\*log2(x), ST(1) = cos(x)

fld st(0) // st(0) = y\*log2(x), st(1) = y\*log2(x), st(2)= cos(x)

frndint //st(0) = целая часть y\*log2(x), st(1) = y\*log2(x), st(2)= cos(x)

fsub ST(1), ST(0) //st(0) = целая часть y\*log2(x), st(1) = дробная y\*log2(x), st(2)= cos(x)

fxch // st(0) = дроб ная y\*log2(x), st(1) = целая часть y\*log2(x)

F2XM1 // ST(0) =дробная 2^(y\*log2(x)) -1, st(1) = целая часть y\*log2(x), st(2) = cos(x)

fld1 // st(0) = 1, ST(1) =дробная 2^(y\*log2(x)) -1, st(2) = целая часть y\*log2(x), st(3) = cos(x)

faddp ST(1), ST(0) // st(0) = дробная 2^(y\*log2(x)), st(1) = целая часть y\*log2(x), st(2) = cos(x)

fscale // st(0) = дробная 2^(y\*log2(x)) \* 2^(целая часть y\*log2(x)) -> x^y, st(1) = целая часть y\*log2(x), st(2) = cos(x)

fxch // st(0) = целая часть y\*log2(x), st(1) = x^y, st(2) = cos(x)

FSTP ST(0) // st(0) = x^y, st(1) = cos(x)

fld y // st(0) = y , st(1) = x^y, st(2) = cos(x)

fsin // st(0) = sin(y) , st(1) = x^y, st(2) = cos(x)

fmulp ST(1), ST(0) // st(0) = sin(y)\*x^y, st(1) = cos(x)

fsqrt // st(0) = sqrt(sin(y)\*x^y), st(1) = cos(x)

fmul ST(1), ST(0) // st(0) = cos(x)\*sqrt(sin(y)\*x^y)

fld y // st(0) = y, st(1) = cos(x)\*sqrt(sin(y)\*x^y)

fld x // st(0) = x, st(1) =y, st(2) = cos(x)\*sqrt(sin(y)\*x^y)

fld1 // st(0) =1, st(1) = x, st(2) =y, st(3) = cos(x)\*sqrt(sin(y)\*x^y)

faddp ST(1), ST(0) // st(0) =x+1, st(1) =y, st(2) = cos(x)\*sqrt(sin(y)\*x^y)

FYL2X // st(0) = y\*log2(x+1), st(1) = cos(x)\*sqrt(sin(y)\*x^y)

fdivp ST(1), ST(0) // st(0) = (cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1)

FLDL2E // st(0) = log2(e), st(1) = (cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1)

fld y //st(0) =y, st(1) = log2(e) , st(2) = (cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1)

fmulp ST(1), ST(0) // st(0) = y\*log2(e), st(1) =(cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1)

fld st(0) // st(0) = y\*log2(e), st(1) = y\*log2(e), st(2) =(cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1)

frndint // st(0) = целая часть y\*log2(e), st(1) = y\*log2(e), st(2) =(cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1)

fsub ST(1), ST(0) // st(0) = целая часть y\*log2(e), st(1) = дробная y\*log2(e), st(2) =(cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1)

fxch // st(0) = дробная часть y\*log2(e), st(1) = целая y\*log2(e), st(2) =(cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1)

F2XM1 // st(0) = дробная часть 2^(y\*log2(e)) -1, st(1) = целая y\*log2(e), st(2) =(cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1)

fld1 // st(0) =1, st(1) = дробная часть 2^(y\*log2(e)) -1, st(2) = целая y\*log2(e), st(3) =(cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1)

faddp ST(1), ST(0) // st(0) = дробная часть 2^(y\*log2(e)), st(1) = целая y\*log2(e), st(2) =(cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1)

fscale // st(0) = дробная часть 2^(y\*log2(e)) \* 2^(целая y\*log2(e)) -> e^y, st(1) = целая y\*log2(e), st(2) =(cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1)

fxch // st(0) = целая y\*log2(e), st(1) = e^y, st(2) = (cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1)

FSTP ST(0) // st(0) = e^y, st(1) = (cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1)

faddp ST(1), ST(0) // st(0) = (cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1) + e^y

fld x // st(0) =x , st(1) =(cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1) + e^y

fld y // st(0) =y, st(1) = x, st(2) =(cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1) + e^y

fdivp ST(1), ST(0) // st(0) = x/y, st(1) =(cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1) + e^y

fptan //st(0) =1, st(1) = tan(x/y), st(2) = (cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1) + e^y

fstp ST(0) // st(0) = tan(x/y) ,st(1) = (cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1) + e^y

fsubp ST(1), ST(0) // st(0) =(cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1) + e^y - tan(x/y)

fld x // st(0) = x, st(1) =(cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1) + e^y - tan(x/y)

fld1 // st(0) =1, st(1) = x, st(2) = (cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1) + e^y - tan(x/y)

fsubp ST(1), ST(0) // st(0) = x-1, st(1) = (cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1) + e^y - tan(x/y)

fld1 // st(0) = 1, st(1) = x-1, st(2) = (cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1) + e^y - tan(x/y)

fscale // st(0) = 2^(x-1), st(1) = x-1, st(2) = (cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1) + e^y - tan(x/y)

fxch // st(0) = x-1, st(1) = 2^(x-1), st(2) = (cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1) + e^y - tan(x/y)

FSTP ST(0) // st(0) = 2^(x-1), st(1) = (cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1) + e^y - tan(x/y)

fsubp ST(1), ST(0) // st(0)= (cos(x)\*sqrt(sin(y)\*x^y))/y\*log2(x+1) + e^y - tan(x/y) -2^(x-1)

fst R

}

cout << "Результат вычесления FPU: " << R << endl;

float R\_cpp = (cos(x) \* sqrt(sin(y)\*pow(x,y))) / (y \* log2(x + 1)) + exp(y) - tan(x / y) - exp2(x-1);

cout << "Результат вычесления C++: " << R\_cpp << endl;

system("pause");

}

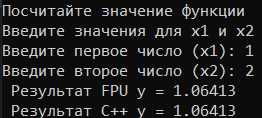


Рисунок 2 – Результат 1 задания

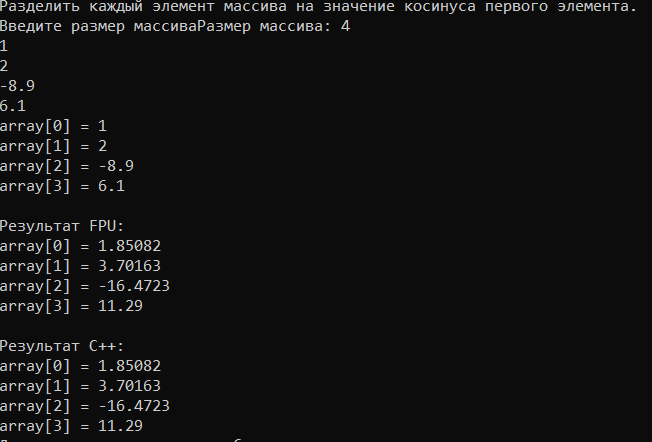


Рисунок 3 – Результат 2 задания

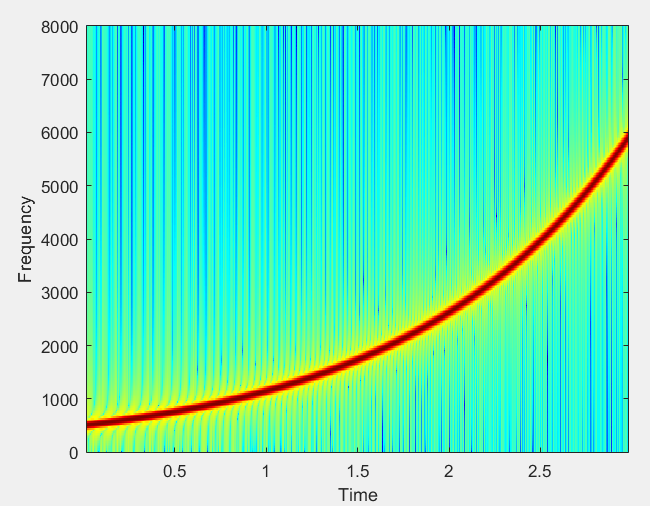


Рисунок 4 – Результат 3 задания

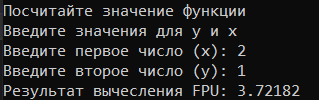


Рисунок 5 – Результат дополнительного задания

# Вывод

В ходе выполнения данной лабораторной работы научиться работать с архитектурой математического сопроцессора (FPU) и командами обработки чисел с плавающей запятой и разработал ассемблерную вставку в высокоуровневый код для обработки данных с плавающей запятой.