|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования FPMI_ngtu_neti_rgb_polya«Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Практическое задание № 4 | | |
| по дисциплине «Языки программирования и методы трансляции» | | |
| **Программирование сопроцессора** | | |
|  | | |
|  | Группа ПМ-04 | Плешкова Дарья |
| Бригада 11 | Сальников дмитрий |
|  |  |
|  |  |
|  |  |
|  |  |
| Преподаватель | сивак мария алексеевна |
|  | петров роман владимирович |
| Новосибирск, 2022 | | |

**1. Задание**

Вычислить значение функции в произвольной точке x.

**2. Алгоритм**

1. Получаем адрес точки, в которой необходимо посчитать значение функции.
2. Вычисляем значение функции в точке.

2.1. Если полученное значение больше “условной бесконечности”, то вывести на экран сообщение о том что значение функции равно бесконечности.

2.2. Иначе вернуть вычисленное значение из процедуры.

1. Выходим из подпрограммы, переходя к адресу возврата и выводим вычисленное значение.

**3. Программные средства**

* PUSH – команда «положить в стек»
* CALL – команда вызова внешней функции
* MOV – команда, пересылающая байт, слово или двойное слово между регистром и ячейкой памяти или между двумя регистрами
* POP – команда «взять из стека»
* RET – команда завершения процедуры
* FLD1 **–** команда загрузки еденицы в регистр ST(0)
* FLD - Загрузка вещественного числа из области памяти или из другого регистра

сопроцессора на вершину стека сопроцессора

* FSIN **-** команда вычисляет синус угла, находящегося в вершине стека сопроцессора. Результат возвращается в регистр ST(0).
* FDIVP **–** команда вычисляет отношение регистров ST(1) и ST(0). Результат помещает в ST(0).
* FSTP **–** команда, осуществляющая выталкивание вещественного числа из стека ST(0) и последующего его сохранение в памяти.
* FCOM **–** команда, сравнивающая значения числа в вершине стека ST(0) и источника.

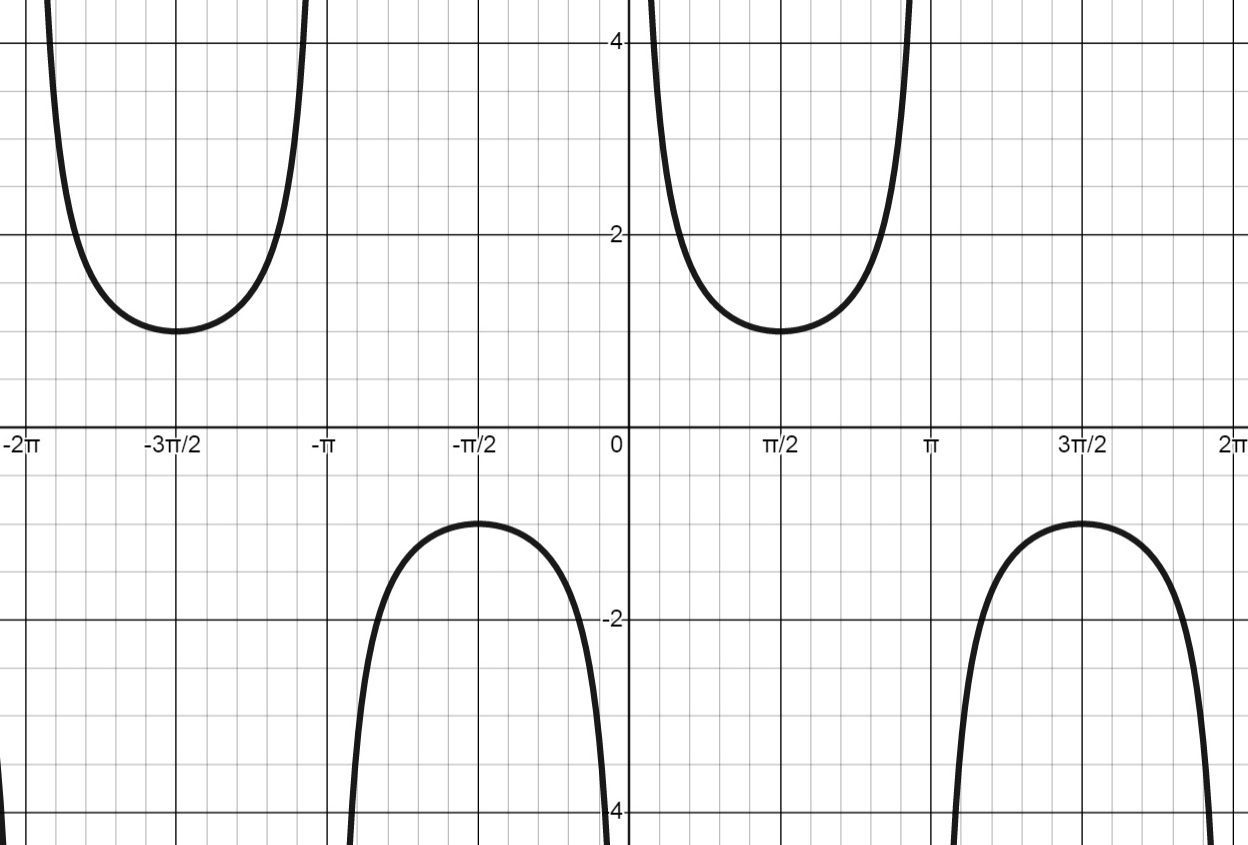
Переменной может быть вещественная переменная либо регистр ST(i), если источник не указан то подразумевается ST(1).

* FSTSW **-** команда, которая позволяет запомнить значение регистра состояния сопроцессора в регистре АХ или ячейке памяти.
* SAHF **–** команда, записывает содержимое регистра AH в регистр флагов EFLAGS.
* FXCH **–** команда обмена вершины регистрового стека ST(0) с любым другим регистром стека сопроцессора.

**4. Тесты**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Исходные данные** | **Результат** | **Комментарий** |
| **1.** | 5 | -1.0428352 | Произвольная точка |
| **2.** | 11.2831853 | -1.0428351 | Переодичность (x = 5 + 2π) |
| **3.** | 0 | Функция не определена | Функция не определена |
| **4.** | 6.26 | -43.1350555 | Значение переодической функции мало |
| **5.** | -5 | 1.0428352 | Переодичность с отрицательным числом |
| 6 | 3.1415926535 | -11438666.0000000 | Точка ≈ π |

**4.График функции**



**5.Текст программы**

**Source.cpp**

#include <locale>

#include <iostream>

#include <iomanip>

using namespace std;

extern "C" void \_\_stdcall newton(float y, float\* func);

int main()

{

setlocale(0, "Ru");

float x, y;

cout << "Введите точку, в которой посчитать функцию, х = ";

cin >> x;

newton(x, &y);

if (y <= 1e+7)

cout << "Значение функции f(x) = " << fixed << setprecision(7) << y << endl;

}

**asm.asm**

*.386*

*.MODEL* FLAT, C

*; Укажем некоторые внешние функции:*

*EXTRN* *WriteConsoleA@20*:*PROC* *; вывод текста*

*EXTRN* *GetStdHandle@4*:*PROC* *; стандартный дескриптор ввода-вывода*

*EXTRN* *CharToOemA@8*:*PROC* *; перекодировка*

*EXTRN* *lstrlenA@4*:*PROC* *; определение длинны строки*

*.DATA*

dout *DD* ? *; Переменная для дескриптора вывода*

x *DD* ? *; Заданное значение х*

adress *DD* ? *; Адрес возвращаемого значения*

e *DD* 10000000.0 *; Условное значение бесконечности*

message *DB* "Функция не определена", 13, 10, 0

*; Строка, выводимая при превышении условного значения бесконечности*

*.CODE*

*; Вывод сообщения о том что функция в этой точке не определена*

*PrintError* *PROC*

*PUSH* -11

*CALL* *GetStdHandle@4*

*MOV* dout, EAX *; Получаем дескриптор вывода*

*PUSH* *OFFSET* message

*PUSH* *OFFSET* message

*CALL* *CharToOemA@8* *; Перекодируем строку*

*PUSH* *OFFSET* message

*CALL* *lstrlenA@4* *; Получаем длину строки*

*PUSH* 0 *; 5-й параметр (Конец аргументов)*

*PUSH* EBX *; 4-й параметр (Место куда поместить кол-во выведенных символов)*

*PUSH* EAX *; 3-й параметр (Длина выводимой строки)*

*PUSH* *OFFSET* message *; 2-й параметр (Адрес выводимой строки)*

*PUSH* dout *; 1-й параметр (Дескриптор вывода)*

*CALL* *WriteConsoleA@20* *; Выводим строку*

*Ret*

PrintError *ENDP*

*; Процедура вычисления функции*

*newton@8* *PROC*

*; Стек: [Старое значение EBP 4][Адрес возврата 4][Заданное значение х 4][Адрес возвращаемого значения 4]*

*PUSH* EBP *; Сохраняем старое значение EBP*

*MOV* EBP, ESP

*MOV* EAX, [EBP] + 8 *; Сдвигаемся на заданное значение*

*MOV* x, EAX *; Кладем его в x*

*MOV* EAX, [EBP] + 12 *; Сдвигаемся на адрес возвращаемого значения*

*MOV* adress, EAX *; Кладем его в adress (туда мы положим значение функции от х)*

*FLD1* *; ST(0) = 1*

*FLD* x *; ST(1) = 1, ST(0) = x*

*FSIN* *; ST(1) = 1, ST(0) = sin(x)*

*FDIVP* *; ST(0) = 1 / sin(x)*

*FSTP* x *; Загружаем в х найденное значение функции*

*FLD* x *; ST(0) = x*

*FLD* e *; ST(1) = x, ST(0) = e*

*FCOM* *; Сравниваем ST(0) и ST(1)*

*FSTSW* AX *; Сохраняем флаги сравнения в АХ*

*SAHF* *; Загружаем флаги в регистр флагов EFLAGS*

*JNC* *not\_inf* *; Если ST(0) > ST(1) (e > x), переходим на метку not\_inf (сообщение об х = inf не выводится)*

*FXCH* ST(1) *; Иначе значения ST(0) и ST(1) меняются местами*

*CALL* *PrintError*

*not\_inf*:

*MOV* ECX, adress *; Кладем адрес на ячейку, в которой будет лежать f(x)*

*MOV* EDX , x *; Кладем в регистр значение f(x)*

*MOV* [ECX] , EDX *; Кладем в ячейку памяти по адресу возвращаемого значения значение X*

*POP* EBP *; Возвращаем старое значение регистра*

*RET* 8 *; Выход из процедуры*

newton@8 *ENDP*

*END*