

Правительство Российской Федерации

**Федеральное государственное автономное образовательное
учреждение высшего образования «Национальный исследовательский
университет «Высшая школа экономики»**

Факультет компьютерных наук
Департамент программной инженерии

**Отчет к домашнему заданию
По дисциплине
«Архитектура вычислительных систем»**

Работу выполнил:
Студент группы БПИ-194 Романюк А.С

Москва 2020

Задание

Разработать программу, вычисляющую с помощью степенного ряда с точностью не хуже 0,05% значение функции $\arccos(x)$ для заданного параметра x (использовать FPU).

Решение

Функция $\arccos(x)$ может быть разложена в степенной ряд Тейлора согласно [1]:

$$\arccos(x) = \frac{\pi}{2} - \arcsin(x) = \frac{\pi}{2} - \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1}$$

Для получения общего члена ряда следует разделить выражение под знаком суммы для n итерации на выражение под знаком суммы для $n-1$ итерации:

$$\frac{(2n)! x^{2n+1}}{4^n (n!)^2 (2n+1)} \cdot \frac{4^{(n-1)} ((n-1)!)^2 (2(n-1)+1)}{(2(n-1))! x^{2(n-1)+1}} = \frac{x^2 (2n-1)(2n)(2(n-1)+1)}{4n^2 (2n+1)}$$

Для проверки полученной суммы степенного ряда следует вычислить точное значение $\arccos(x)$. Но поскольку в наборе команд FPU отсутствует такая команда, но присутствует команда FPATAN для вычисления арктангенса, для вычисления $\arccos(x)$ можно воспользоваться формулой, приведенной в [2]:

$$\arccos(x) = \frac{\pi}{2} - \arcsin(x) = \frac{\pi}{2} - \operatorname{arctg} \frac{x}{\sqrt{1-x^2}}$$

Напишем сначала программу, вычисляющую с помощью степенного ряда значение функции $\arccos(x)$ на языке C++, чтобы «обкатать» алгоритм.

Текст программы приведен ниже:

```
#define _USE_MATH_DEFINES

#include <math.h>
#include <iostream>

using namespace std;

int main() {
    double _x = 0.5;
    double eps = 0.0005; // Точность
    double s = 0;
    int n = 0;
    double a = _x;
    do {
        s += a;
        a = a * _x * _x * (2 * n - 1) * (2 * n) / 4.0 / (n * n) * (2 * (n - 1) + 1) / (2
* n + 1);
    } while (fabs(a) >= eps);

    cout << M_PI / 2 - s << endl;
```

```
cout << acos(_x) << endl;

system("pause");
return 0;}
```

Итерационный цикл завершается, когда очередной член суммы по модулю будет меньше заданной точности. Точность 0,05% соответствует абсолютной величине 0,0005.

Теперь, когда алгоритм отлажен, и результат совпадает с точным значением (в пределах заданной погрешности), перепишем данную программу на ассемблере для компилятора FASM.

Для ввода исходных данных воспользуемся функциями стандартной библиотеки си:

```
char *gets(char *str)
```

Функция `gets()` считывает символы из `stdin` и помещает их в массив символов, на который указывает `str`. Символы считываются до тех, пока не встретится новая строка или EOF. Символ «новая строка» не делается частью строки, а транслируется в нулевой символ, завершающий строку.

Чтобы преобразовать строку в действительное значение, применим функцию `sscanf`, которая распознает и считывает данные по заданному шаблону из строки.

```
int sscanf(char *buf, const char *format, arg-list);
```

Она имеет следующие аргументы:

- `buf` — указатель на символьный буфер, подлежащий считыванию;
- `format` — указатель на C-строку, содержащую формат результата;
- остальные аргументы — данные, подлежащие форматированию.

Данная функция возвращает целое значение, которое представляет собой количество корректно распознанных значений из `format`, что позволяет реализовать проверку корректности ввода.

Так как у данной функции количество аргументов не постоянно, то она использует соглашение вызова `cdecl`, которое отличается от `stdcall` тем, что очистку стека от аргументов выполняет вызывающая функция, а не вызываемая.

Для вывода результата воспользуемся функцией `printf`, которая форматирует данные по заданному шаблону и выводит их на консоль.

```
int printf (const char * format, ... );
```

Она имеет следующие аргументы:

- `format` — указатель на C-строку, содержащую формат результата;
- остальные аргументы — данные, подлежащие форматированию.

Так как у данной функции количество аргументов не постоянно, то она использует соглашение вызова `cdecl`, которое отличается от `stdcall` тем, что очистку стека от аргументов выполняет вызывающая функция, а не вызываемая.

Для завершения работы программы выполним вызов функции `ExitProcess`, которая имеет один аргумент — код завершения работы программы.

```
VOID ExitProcess(  
    UINT uExitCode // код выхода для всех потоков  
);
```

Входной параметр `x` считывается из консоли (клавиатуры). Результат работы выводится на консоль (экран). Программу можно разбить на следующие функции:

Главная функция программы. В ней выполняет ввод исходных данных с проверкой их корректности, вызываются функции `myarccos` и `arccos` и выводятся на консоль результаты выполнения данных функций.

`double myarccos(double x, double eps);`

`myarccos` — функция вычисления $\arccos(x)$ с помощью степенного ряда с заданной точностью `eps`. Вызывается по соглашению `cdecl`. Имеет локальные переменные:

`t` — временная переменная

`a` — очередное слагаемое ряда

Результат возвращается в `ST(0)`.

`double arccos(double x);`

`arccos` — функция точного вычисления $\arccos(x)$. Вызывается по соглашению `cdecl`.

Результат возвращается в `ST(0)`.

Текст программы приведен ниже:

```
format PE Console  
entry start  
  
include '../..//INCLUDE/win32a.inc'  
  
section '.data' data readable writeable  
x1      dq ? ; Введённое пользователем значение:  
eps1    dd 0.0005 ; Точность 0.05%  
; Константы:  
c2      dq 2.0  
c4      dq 4.0  
msg1    db 'Enter x (|x|<=1): ',0  
msg2    db 'Wrong number.',13,10,0  
fmt1    db '%lf',0  
msg3    db 'Teylor row = %lg',13,10,0  
msg4    db 'Calculated arccos = %lg',13,10,0  
buf     db 256 dup(0)
```

```

section '.code' code readable executable
start:
    ccall [printf],msg1          ; Выводим сообщение в консоль
    ccall [gets],buf             ; Вводим с консоли значение
    ccall [scanf],buf,fmt1,x1    ; Парсим введенную строку в число

    ; Если преобразование удалось, то продолжить:
    cmp eax,1
    jz m1

    ; Иначе: выводим сообщение об ошибке и начинаем заново.
    ccall [printf],msg2
    jmp start

m1:    fld [x1]                  ; Введенное значение
        fabs                    ; Модуль введенного значения
        fld1                    ; 1
        fcompp                  ; Сравниваем 1 с модулем введенного числа
        fstsw ax                ; Записать флаги сопроцессора в ax
        sahf                    ; Переносим их в флаги процессора
        jb start                ; 1 < x, начать заново
        fld [eps1]              ; Точность вычисления
        sub esp, 8              ; Выделяем в стеке место под double
        fstp qword [esp]        ; Записать в стек double число
        fld qword [x1]          ; Введенное значение
        sub esp, 8              ; Выделить в стеке место под double
        fstp qword [esp]        ; Записать в стек double число
        call myarccos           ; Вычислить myarccos(x,eps)
        add esp, 16             ; Удалить переданные параметры

        sub esp, 8              ; Передать сумму ряда
        fstp qword [esp]        ; Функции через стек
        push msg3               ; Формат сообщения
        call [printf]           ; Сформировать результат
        add esp, 12             ; Коррекция стека

        fld qword [x1]          ; Введенное значение
        sub esp, 8              ; Выделить в стеке место под double
        fstp qword [esp]        ; Записать в стек double число
        call arc                ; Вычислить arccos(x,eps)
        add esp,                ; Удалить переданные параметры

        sub esp, 8              ; Передать точное значение arccos
        fstp qword [esp]        ; Функции через стек
        push                   ; Формат сообщения
        call [printf]           ; Сформировать результат
        add esp, 12             ; Коррекция стека

        ccall [_getch]          ; Ожидание нажатия любой клавиши

ex:    stdcall [ExitProcess], 0 ; Выход
; -----
; Функция для вычисления arccos(x) с точность eps.
; Соглашение вызова через cdecl
; Вход: double x, double eps
; Вывод: double
myarccos:
    push ebp                    ; Создать кадр стека
    mov ebp,esp
    sub esp,0ch                 ; Создание локальных переменных
;Локальные переменные:

```

```

t      equ ebp-0ch      ; Временная переменная
a      equ ebp-8h      ; Очередное слагаемое ряда

;Переданные функции параметры:
x      equ ebp+8h
eps    equ ebp+10h

;Вычисленное значение
      fld qword [x]      ;Загрузить x
      fstp qword [a]     ;a = x
      fldpi              ;pi
      fdiv [c2]          ;pi/2
      fldz               ;s=0
      mov ecx,0          ;//n=0

m11:   fadd qword [a]     ;s += a;
      inc ecx            ;n++;
      fld qword [a]      ;a
      fmul qword [x]     ;a*x
      fmul qword [x]     ;a*x*x
      lea eax,[2*ecx-1]   ;2n-1
      mov [t],eax        ;t=2n-1
      fimul dword [t]    ;a*x*x*(2n-1)
      lea eax,[2*ecx]    ;2n
      mov [t],eax        ;t=2n
      fimul dword [t]    ;a*x*x*(2n-1)*2n
      fdiv [c4]          ;a*x*x*(2n-1)*2n/2
      mov [t],ecx        ;n
      fidiv dword [t]    ;a*x*x*(2n-1)*2n/2/(n*n)
      fidiv dword [t]    ;n-1
      lea edx,[ecx-1]    ;(2 * (n - 1) + 1)
      lea eax,[2*edx+1]  ;t=(2 * (n - 1) + 1)
      mov [t],eax        ;a*x*x*(2n-1)*2n/2/(n*n)*(2 * (n - 1) + 1)
      fimul dword [t]    ;(2 * n + 1)
      lea eax,[2*ecx+1]  ;t=(2 * n + 1)
      mov [t],eax        ;a*x*x*(2n-1)*2n/2/(n*n)*(2 * (n - 1) + 1)/
                          ;(2 * n + 1)
      fst qword [a]      ;a = a*x*x*(2 * n - 1)*(2 * n) / 4.0 / (n*n)*(2 * (n-
                          ;1) + 1) / (2 * n + 1);
      fabs               ;|a|
      fcomp qword [eps]  ; сравнить |a| с eps
      fstsw ax;          ; перенести флаги сравнения в ax
      sahf;              ; занести ah в флаги процессора
      jnb m11;           ; Если |a|>=e, продолжить цикл
      fsubp st1,st       ;pi/2-полученная сумма
      leave
      ret

; -----
; Функция для точного вычисления arccos(double x)
; Соглашение вызова через cdecl
; Вход: double x
; Вывод: double
arccos:
      push ebp           ; Создать кадр стека
      mov ebp,esp
      fldpi              ;pi
      fdiv [c2]          ;pi/2
      fld qword [ebp+8];x
      fld1               ;1
      fld qword [ebp+8];x
      fmul st,st         ; x^2

```

```

        fsubp st1,st          ; 1-x^2
        fsqrt                ; sqrt(1-x^2)
        fpatan               ; arctg(x/sqrt(1-x^2))
        fsubp st1,st          ; pi/2-arctg(x/sqrt(1-x^2))
        pop ebp               ; Эпилог функции
        ret

; -----
section '.idata' import data readable

library kernel,'kernel32.dll',\
        user,'user32.dll',\
        msvcrt,'msvcrt.dll'

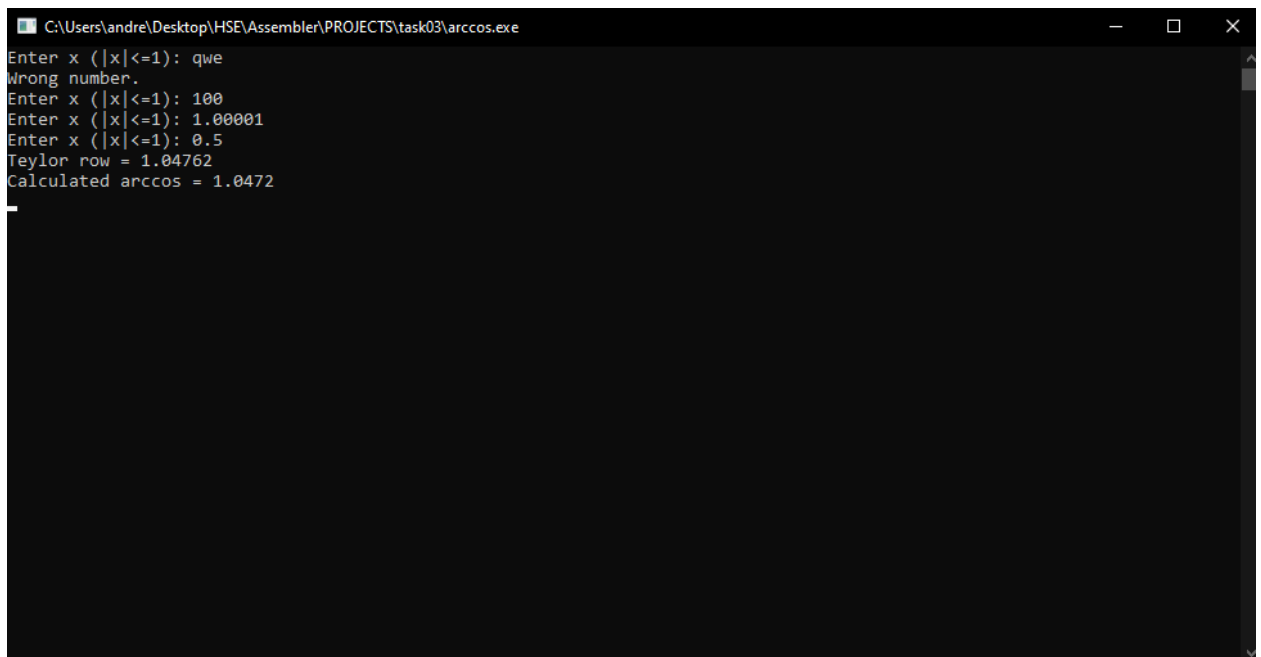
import kernel,\
        ExitProcess,'ExitProcess'

import msvcrt,\
        sscanff,'sscanf',\
        gets,'gets',\
        _getch,'_getch',\
        printf,'printf'

```

Тестирование

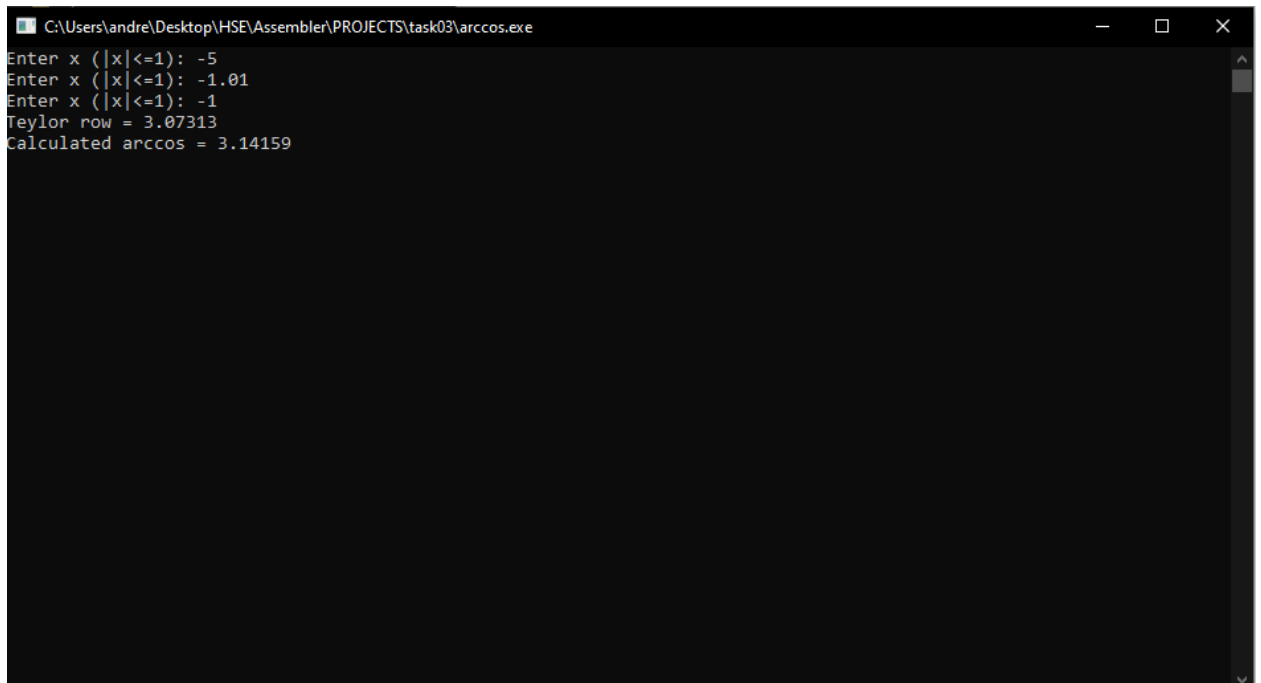
Результат выполнения программы приведен на рисунке 1.



```
C:\Users\andre\Desktop\HSE\Assembler\PROJECTS\task03\arccos.exe
Enter x (|x|<=1): qwe
Wrong number.
Enter x (|x|<=1): 100
Enter x (|x|<=1): 1.00001
Enter x (|x|<=1): 0.5
Taylor row = 1.04762
Calculated arccos = 1.0472
```

Рисунок 1 – Результат выполнения программы

Как видно из рисунка 1, программа обрабатывает некорректный ввод, погрешность вычисления соответствует заданной.



```
C:\Users\andre\Desktop\HSE\Assembler\PROJECTS\task03\arccos.exe
Enter x (|x|<=1): -5
Enter x (|x|<=1): -1.01
Enter x (|x|<=1): -1
Taylor row = 3.07313
Calculated arccos = 3.14159
```

Рисунок 2 – Результат выполнения программы для отрицательных значений

Как видно из рисунка 2, программа обрабатывает некорректный отрицательный ввод, а также работает с отрицательными числами.

Список используемых источников

1. Википедия (2020) «Ряд Тейлора» (https://ru.wikipedia.org/wiki/Ряд_Тейлора)
Просмотрено: 15.10.2020
2. Википедия (2020) «Обратные тригонометрические функции»
(https://ru.wikipedia.org/wiki/Обратные_тригонометрические_функции)
Просмотрено 15.10.2020
3. Легалов А.И.(2020) «Разработка программ на ассемблере. Использование макроопределений» (<http://softcraft.ru/edu/comparch/practice/asm86/04-macro/>)
Просмотрено: 18.10.2020
4. Легалов А.И.(2020) «Разработка программ на ассемблере. Использование сопроцессора с плавающей точкой»
(<http://softcraft.ru/edu/comparch/practice/asm86/05-fpu/>) Просмотрено: 18.10.2020
5. YouTube “Яша добрый хакер” (2018) «Канал Яша добрый хакер»
(<https://www.youtube.com/user/yashechka85/videos>) Просмотрено: 20.10.2020
6. YouTube “CryptoFun [IT]” (2019) «Канал CryptoFun [IT]»
(<https://www.youtube.com/c/CryptFunIT/videos>) Просмотрено: 20.10.2020
7. Кип Р.И. Язык ассемблера для процессоров Intel. М.: Издательский дом “Вильямс”, 2005. – 912с.