**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

**Проект по дисциплине «Архитектура вычислительных систем»**

**Программа для вычисления значения функции гиперболического синуса от заданного значения на языке ассемблера FASM**

Исполнитель

студент группы БПИ196-1

Махнач Ф. О.

01.11.2020 г.

**Оглавление**

[1. Текст задания 2](#_Toc55067661)

[2. Теоретическая база 2](#_Toc55067662)

[3. Алгоритм вычисления 2](#_Toc55067663)

[3.1. Код на языке ассемблера FASM 3](#_Toc55067664)

[4. Ограничения, крайние случаи 4](#_Toc55067665)

[5. Особенности программы 7](#_Toc55067666)

[6. Тестирование 9](#_Toc55067667)

[7. Исходный код программы на языке ассемблера FASM 12](#_Toc55067668)

[8. Используемые источники 15](#_Toc55067669)

# Текст задания

Разработать программу, вычисляющую с помощью степенного ряда с точностью не хуже 0,1% значение функции гиперболического синуса для заданного параметра (использовать FPU).

Вывод данных следует осуществлять в консоль.

# Теоретическая база

Приближённое значение гиперболического синуса в точке можно получить с помощью степенного ряда, который в свою очередь выводится аналитически из степенного ряда функции [1]. Итого имеем ряд:

# Алгоритм вычисления

Заметим, что слагаемое ряда можно получить из путём домножения на (т. е. ).   
Разумно будет вычислять значение суммы степенного ряда в цикле. Тогда на каждой итерации цикла нам понадобится:  
 1) непосредственно число (а точнее, можно сразу вычислить );   
 2) промежуточное значение суммы (res);   
 3) значение предыдущего слагаемого (term);  
 4) значение – последнее число, на которое мы делили слагаемое.

Перед первой итерацией мы вычисляем значение xSqr = , а также устанавливаем значения res = , term = , = 1.   
На каждой итерации цикла вычисляем новое слагаемое как   
term \*= xSqr /   
и добавляем к сумме  
res += term  
После этого нам необходимо проверить условие выхода из цикла. В тексте задачи указано, что точность должна быть не менее 0,1%, причём определяется она через разность полученного значения со значением предыдущей частичной суммы: (раз речь о процентах, мы по видимому должны разделить на текущее значение). Нетрудно догадаться, что разность есть просто слагаемое:   
Причём модули также могут быть опущены исходя из того, что и имеют один знак.

Псевдокод для приведённого алгоритма:  
x = readFloat()  
xSqr = x \* x, res = x, term = x, m = 1  
while term / res > 0.001:  
 term \*= xSqr  
 term /= (m + 1) \* (m + 2)  
 m += 2   
 res += term

На языке ассемблера FASM данные вычисления будут выполняться при помощи FPU (Floating Point Unit). В частности, здесь используется сравнение, сложение, деление и умножение вещественных чисел. Помимо этого, вместо выполнения вычисления   
(m + 1) \* (m + 2) использована команда inc и деление на m дважды.

## Код на языке ассемблера FASM

Ниже приведён фрагмент кода на FASM, аналогичный псевдокоду выше. Вопрос ввода, вывода, обработки ошибок пока опущен.

**start:** FINIT  
 ; Ввод числа x  
**calculation:** ; Movf – макрос, перемещающий fp значение из одного адреса в другой  
 Movf res, x ; res = x  
 Movf term, x ; term = x   
 Movf xSqr, x ; xSqr = x \* x  
 fld [xSqr]  
 fmul [x]  
 fstp [xSqr]  
 mov [lastDenomN], 1 ; lastDenomN = 1   
**calculation\_loop:**  
 ; Вычисляем слагаемое  
 fld [term]  
 fmul [xSqr] ; term \*= xSqr  
 inc [lastDenomN] ; lastDenomN++  
 fidiv dword[lastDenomN] ; term /= lastDenomN  
 inc [lastDenomN] ; lastDenomN++  
 fidiv dword[lastDenomN] ; term /= lastDenomN  
 fstp [term]   
  
 fld [res]  
 fadd [term] ; res += term  
 fstp [res]

; Сравниваем term / res с accuracy (= 0.001)  
 fld [res]  
 fld [term]  
 fdivrp st1, st0  
 fld [accuracy]  
 fcompp  
 fstsw ax  
 sahf  
 jb calculation\_loop ; Пока term / res > accuracy, продолжаем вычисления  
  
; Вывод результата, повтор решения

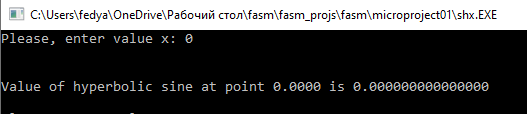
# Ограничения, крайние случаи

На вход программе подаётся действительное число. Дополнительных ограничений на это число не налагается, так как функция гиперболического синуса определена для всех действительных чисел.

В программе отдельно обработаны следующие случаи:

1) Ввод числа 0 (обычный алгоритм приведёт к делению на ноль при обработке условия выхода из цикла). Проверяем перед началом цикла:

fld [x]  
fldz  
fcompp  
fstsw ax  
sahf  
je output



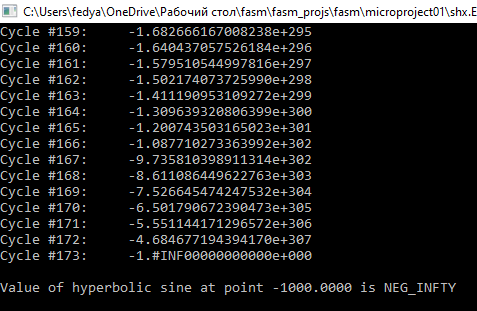
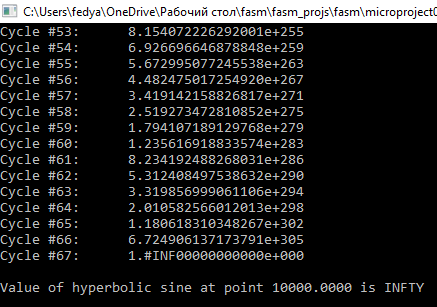
Рисунок

2) Ввод слишком большого (по модулю) числа, что приводит к переполнению при вычислении значения гиперболического синуса. Такая ситуация возникает (приблизительно) при достижении значения (значение гиперболического синуса превышает 1e308 по модулю). Проверка на бесконечность производится после каждой итерации вычисления суммы с помощью проверки соответствующего флага. Обнаружение бесконечности в процессе вычисления приводит к выходу из цикла:

fstsw ax  
and ax, 1000b  
cmp eax, 0  
jg infty\_case

В infty\_case расположены инструкции, проверяющие знак бесконечности (проверяем, сравнивая x с нулём):

infty\_case:  
 fld [x]  
 fldz  
 fcompp  
 fstsw ax  
 sahf  
 jbe sv\_inf ; Если значение больше нуля, то это пол. беск.  
 invoke printf, outMsgStr, dword[x], dword[x+4], negInfty  
 jmp endprog  
sv\_inf:  
 invoke printf, outMsgStr, dword[x], dword[x+4], infty  
 jmp endprog



Рисунок

Рисунок 3

Технически можно было бы просто проверить входное значение на принадлежность диапазону (допустим, ), но по какой-то причине я решил проверять переполнение на месте.  
Последующие тесты показали, что по загадочным причинам при вводе числа, изначально выходящего за границы FP значений двойной точности (напр. 1e310) данный подход не работает и программа «ломается». Для решения этой проблемы было решено всё-таки прикрутить проверку на входного значение. Границей допустимого я решил использовать upperBound = 1e10, который также использую для определения типа вывода.

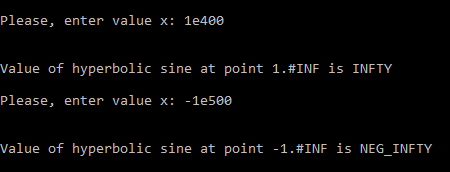


Рисунок 4

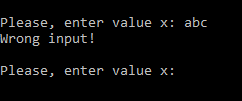
3) Ввод символов, не представляющих вещественное число (напр. “abc”). В этом случае выводится сообщение «Wrong input!» и программа предлагает ввести значение снова. Это достигается проверкой значения регистра eax после вызова scanf.

Рисунок 5

Однако здесь возникает проблема: scanf(“%lf”, &x) буферизирует ввод, т. е. при вводе «1.5abc» он считывает 1.5, сохраняет “abc” и после вычисления значения для 1.5 пытается прочитать FP значение из “abc”. Также (например) ввод “2.2.2.2.2” будет прочитан как последовательность “2.2”, “.2”, “.2”, “.2” и обработан соответственно.

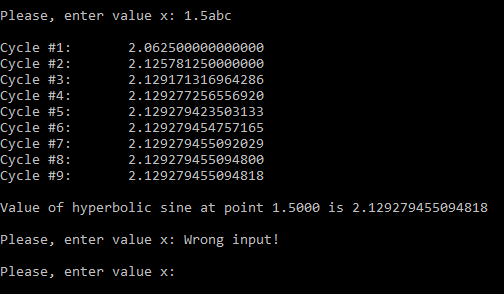


Рисунок 6

После ряда попыток исправить данную ситуацию было принято решение оставить всё в текущем состоянии. Испробованные варианты решения (попытки очистить буфер, считать всю оставшуюся строку) либо потерпели неудачу, либо повлекли более неприятные побочные эффекты. Другие не испробованные на практике способы (посимвольный перевод строки в вещественное число) видятся мне чересчурсложными в контексте отсутствия острой необходимости такой жёсткой обработки ввода.

# Особенности программы

1. Принято решение «зациклить» программу. После каждой итерации (запрос ввода => вычисление результата => вывод) программа возвращается в исходное состояние и снова запрашивает у пользователя ввод. Для выхода из программы можно воспользоваться стандартными средствами операционной системы (нажать на крестик, сочетание Ctrl+C).

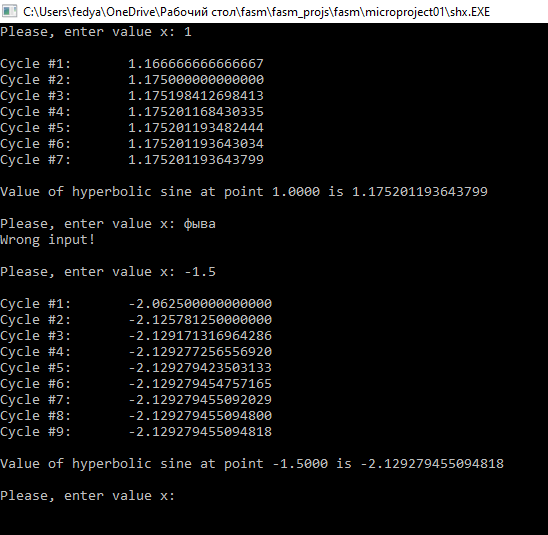


Рисунок 7

2. Программа выводит все промежуточные значения при вычислении суммы ряда. Этот функционал был необходим при разработке, в финальной версии я решил его не убирать.

3. Точность я увеличил с 0,1% до , так как для больших входных данных точность 0,1% даёт довольно неточные значения.

4. В программе используется небольшой макрос Movf, который копирует значение второго аргумента в первый, где оба аргумента представляют floating-point значения.

macro Movf dstf, srcf {  
 mov edx, dword[srcf]  
 mov dword[dstf], edx  
 mov edx, dword[srcf+4]  
 mov dword[dstf+4], edx  
}

5. Как указано в пункте 4.3, использование scanf для считывания ввода приводит к тому, что при введении нескольких «слов» каждое «слово» будет обработано отдельно.   
Так, ввод «1abc 2.2 rb1» приведет к обработке частей «1», «abc», «2.2», «rb1», как представлено на рисунке 8.

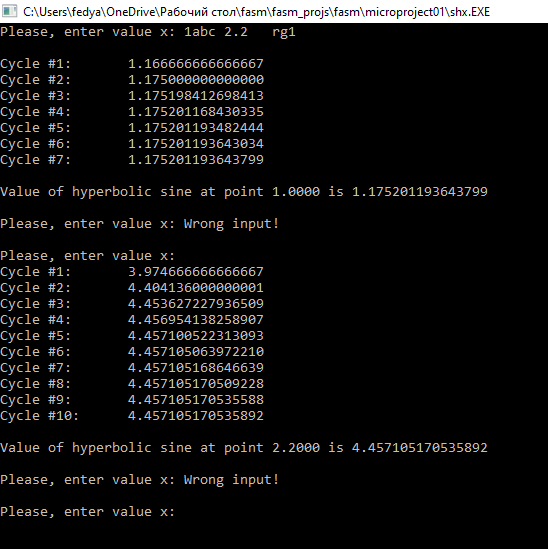


Рисунок 8

6. При достаточно больших промежуточных значениях и значениях результата программа выводит эти значения в экспоненциальном виде (рис. 9), что реализовано в «функциях» PrintMidResult и PrintResult.

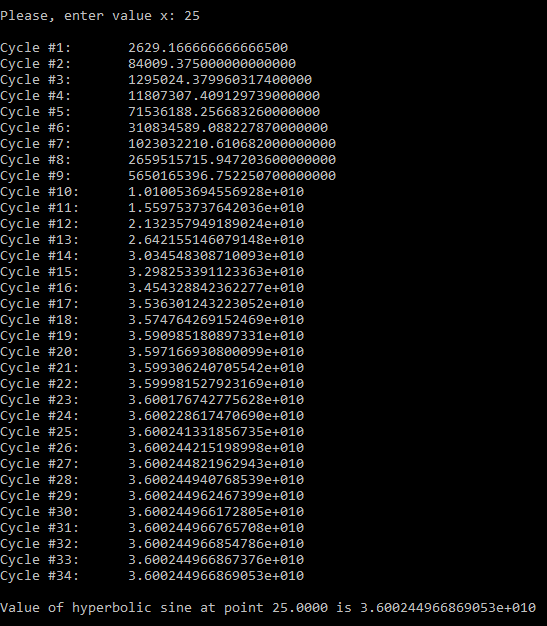
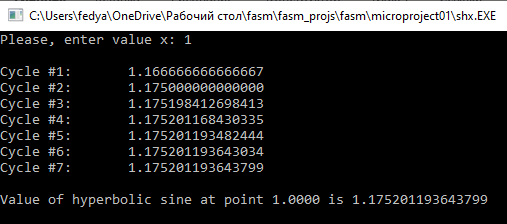


Рисунок 9

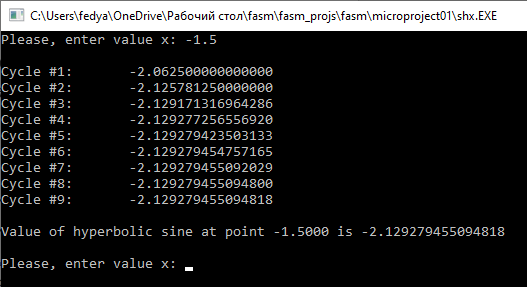
# Тестирование

Ниже приведены скриншоты, демонстрирующие вывод программы при различных входных данных. Также предоставлены значения гиперболического синуса, полученные в онлайн калькуляторе [5].  
P.S. Я отключил вывод значения на каждой итерации в некоторых тестах для компактности

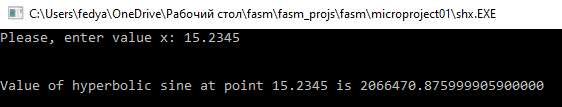
1. x = 1,   
   Значение в онлайн калькуляторе: 1.1752011936438014  
   Значение, полученное в программе: 1.175201193643799



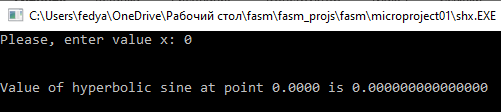
1. x = -1.5,  
   Значение в онлайн калькуляторе: -2.1292794550948173  
   Значение, полученное в программе: -2.129279455094818



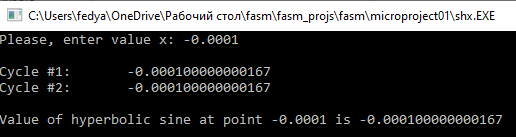
1. x = 15.2345,  
   Значение в онлайн калькуляторе: 2066470.8760000272  
   Значение, полученное в программе: 2066470.875999905900000



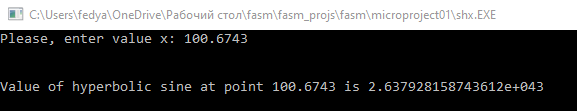
1. x = 0,  
   Значение в онлайн калькуляторе: 0  
   Значение, полученное в программе: 0



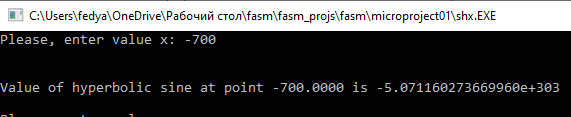
1. x = -0.0001,  
   Значение в онлайн калькуляторе: -0.00010000000016668897  
   Значение, полученное в программе: -0.000100000000167



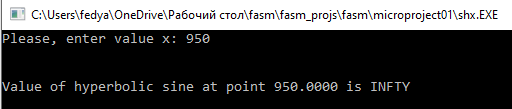
1. x = 100.6743,  
   Значение в онлайн калькуляторе: 2.6379281587439964e+43  
   Значение, полученное в программе: 2.637928158743612e+043



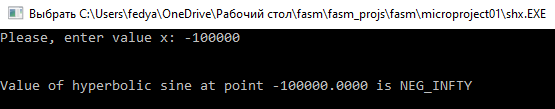
1. x = -700,  
   Значение в онлайн калькуляторе: -5.0711602736748336e+303  
   Значение, полученное в программе: -5.071160273669960e+303



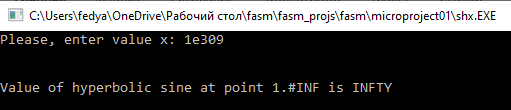
1. x = 950,  
   Значение в онлайн калькуляторе: **∞**  
   Значение, полученное в программе: INFTY



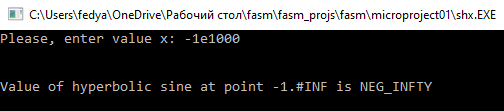
1. x = -100000,  
   Значение в онлайн калькуляторе: -  
   Значение, полученное в программе: NEG\_INFTY



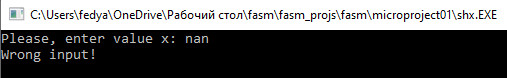
1. x = 1e309,  
   Значение в онлайн калькуляторе: **∞**  
   Значение, полученное в программе: INFTY



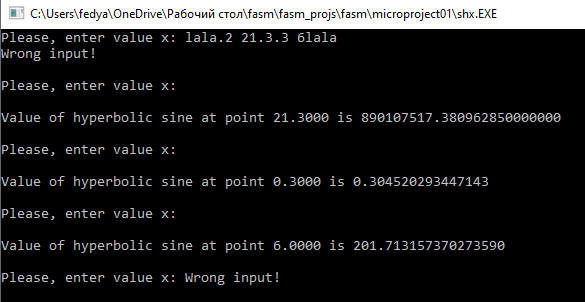
1. x = -1e1000,  
   Значение в онлайн калькуляторе: -  
   Значение, полученное в программе: NEG\_INFTY



1. x = nan,  
   Значение в онлайн калькуляторе: «Ожидается число.»  
   Значение, полученное в программе: «Wrong input!»



1. x = lala.2 21.3.3 6lala  
   Значения, полученные в программе:   
    «lala.2» => «Wrong input!»   
    «21.3» => 890107517.380962850000000  
    «.3» => 0.304520293447143  
    «6» => 201.713157370273590  
    «lala» => «Wrong input!»



# Исходный код программы на языке ассемблера FASM

; Махнач Федор, БПИ 196  
; Вариант 15  
format PE console  
  
entry start  
  
include 'win32a.inc'  
  
; Перемещает fp значение из одного адреса в другой через регистр edx  
macro Movf dstf, srcf {  
 mov edx, dword[srcf]  
 mov dword[dstf], edx  
 mov edx, dword[srcf+4]  
 mov dword[dstf+4], edx  
}  
;--------------------------------------------------------------------------  
section '.data' data readable writable

inputRequest db 'Please, enter value x: ', 0  
wrongInput db 'Wrong input!', 13, 10, 0  
calcMsg db 'Cycle #%d:', 9, '%.15lf', 13, 10, 0  
calcMsgExp db 'Cycle #%d:', 9, '%.15e', 13, 10, 0  
outMsg db 'Value of hyperbolic sine at point %.4lf is %.15lf', 13,10,0  
outMsgExp db 'Value of hyperbolic sine at point %.4lf is %.15e', 13, 10, 0  
outMsgStr db 13,10,'Value of hyperbolic sine at point %.4lf is %s',13,10,0  
fmtFlt db '%lf', 0  
fmts db '%s', 0  
newLine db 13, 10, 0  
infty db 'INFTY', 0  
negInfty db 'NEG\_INFTY', 0  
holder db 0  
  
x dq 1 ; Вводимое пользователем значение  
res dq 1 ; Результат  
xSqr dq 1 ; Квадрат значения x (чтобы не вычислять каждый раз)  
term dq 1 ; Предыдущее слагаемое суммы степенного ряда  
lastDenomN dd 1 ; Последнее число, на которое мы делили   
  
accuracy dq 1e-12 ; "с точностью не хуже 0,1%" <=> можно и точнее. upperBound dq 1e10 ; Значение, после которого мы начинаем выводить в экспоненциальном формате. Также это граница допустимого ввода (по модулю)

section '.code' code readable executable  
;--------------------------------------------------------------------------  
start:  
 FINIT  
 invoke printf, inputRequest  
 invoke scanf, fmtFlt, x, holder  
 cmp eax, 0  
 jne calculation  
 invoke scanf, fmts, holder ; Считываем строку до конца, освобождаясь от буфера (чтобы сканф сработал на следующей итерации)  
 invoke printf, wrongInput ; Сообщаем о неверном вводе  
 jmp endprog

calculation:  
 invoke printf, newline  
 Movf res, x ; Записываем в результат число х  
 fld [x]  
 fldz  
 fcomp ; Проверка на ноль  
 fstsw ax  
 sahf  
 je output  
  
 fabs  
 fld [upperBound]  
 fcompp ; Проверка на слишком большое значение  
 fstsw ax  
 sahf  
 jb infty\_case  
  
correct\_input\_val:  
 Movf term, x ; Первое слагаемое -- само число х  
 Movf xSqr, x ; Записываем в xSqr значение x^2 (не считать его каждый раз)  
 fld [xSqr]  
 fmul [x]  
 fstp [xSqr]  
 mov [lastDenomN], 1  
 mov ebx, 1 ; Счётчик  
  
calculation\_loop:  
 ; Вычисляем очередное слагаемое, умножая предыдущее на x^2 / (n\*(n + 1))  
 fld [term]  
 fmul [xSqr]  
 inc [lastDenomN]  
 fidiv dword[lastDenomN]  
 inc [lastDenomN]  
 fidiv dword[lastDenomN]  
 fstp [term]  
 ; Добавляем слагаемое к результату  
 fld [res]  
 fadd [term]  
 fstp [res]  
 ; Выводим сообщение, содержащее значение промежуточного результата  
 call PrintMidResult  
 inc ebx  
 ; Проверяем, является ли значение нулём или бесконечностью  
 fstsw ax  
 and ax, 1000b  
 cmp eax, 0  
 jg infty\_case  
 ; Необходимо вычислить отклонение: |res\_i - res\_{i - 1}| / |res\_i|  
 ; Значение под модулем это просто term, term имеет тот же знак, что и res, поэтому просто сравниваем term/res с accuracy  
 fld [res]  
 fld [term]  
 fdivrp st1, st0  
 fld [accuracy]  
 fcompp  
 fstsw ax  
 sahf  
 ; Если accuracy больше, чем term/res -- продолжаем вычислять сумму  
 jb calculation\_loop

output:  
 ; Выводим результат  
 invoke printf, newline  
 call PrintResult

endprog:  
 invoke printf, newline  
 jmp start ; Бесконечный цикл, да

infty\_case:  
 fld [x]  
 fldz  
 fcompp  
 fstsw ax  
 sahf  
 jbe sv\_inf ; Если значение больше нуля, то это полож. беск.  
 invoke printf, outMsgStr, dword[x], dword[x+4], negInfty  
 jmp endprog

sv\_inf:  
 invoke printf, outMsgStr, dword[x], dword[x+4], infty  
 jmp endprog

;--------------------------------------------------------------------------  
; При большом значении (>upperBound) мы выводим значение в экспоненциальной форме  
PrintMidResult:   
 fld [res]  
 fabs  
 fld [upperBound]  
 fcompp  
 fstsw ax  
 sahf  
 jb PrintMidResult\_exp\_output

invoke printf, calcMsg, ebx, dword[res], dword[res+4]  
 add esp, 16  
ret  
PrintMidResult\_exp\_output:  
 invoke printf, calcMsgExp, ebx, dword[res], dword[res+4]  
 add esp, 16  
ret  
;--------------------------------------------------------------------------  
; При большом значении (>upperBound) мы выводим значение в экспоненциальной форме  
PrintResult:  
 fld [res]  
 fabs  
 fld [upperBound]  
 fcompp  
 fstsw ax  
 sahf  
 jb PrintResult\_exp\_output  
 invoke printf, outMsg, dword[x], dword[x+4], dword[res], dword[res+4]  
 add esp, 20  
ret

PrintResult\_exp\_output:  
 invoke printf, outMsgExp, dword[x], dword[x+4], dword[res], dword[res+4]  
 add esp, 20  
ret  
;---------------------------------------------------------------------  
section '.idata' import data readable

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

import kernel,\  
 ExitProcess, 'ExitProcess'

import msvcrt,\  
 printf, 'printf',\  
 getch, '\_getch',\  
 scanf, 'scanf'

# Используемые источники

1. Степенной ряд функции гиперболического синуса: <https://scask.ru/m_book_gf.php?id=10>
2. Презентация с семинара про FPU:   
   <http://softcraft.ru/edu/comparch/practice/asm86/05-fpu/fpu.pdf>
3. Команды сравнения FPU: <http://osinavi.ru/asm/FPUexpansion/5.html>
4. Инструкции FPU: <http://flatassembler.narod.ru/fasm.htm#2-1-13>
5. Онлайн калькулятор гиперболических функций: <https://planetcalc.ru/1116/>
6. Команды форматирования для функций printf, scanf:   
   <https://prog-cpp.ru/c-input-output/>
7. Бесчисленное множество сайтов/форумов по разным вопросам