Правительство Российской Федерации

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

Факультет компьютерных наук

Отчет к домашней работе №3

По дисциплине

“Архитектура вычислительных систем”

Работу выполнил

Студент группы БПИ-197 _____ А. А. Синенко

подпись, дата

Работу проверил _____ А. И. Легалов

подпись, дата

Москва 2020

Содержание

Постановка задачи

1) Разработать программу, вычисляющую с помощью ряда Эйлера с точностью не хуже 0,1% значение числа e^1 (с использованием команд сопроцессора).

Уточнение задачи

1) Написать программу, на Assembler в интегрированной среде разработке FASM, которая вычисляет значение числа e^1 с отклонением не более 0,1% от табличного.
2) На экран в консольную программу выведется вычисленное значение числа e^1.

[1] e - основание натурального логарифма, математическая константа, иррациональное и трансцендентное число. Приблизительно равно 2,71828. Иногда число e называют числом Эйлера или числом Непера. Обозначается строчной латинской буквой «e». [1, стр.1]

Математическое описание

Число e может быть определено несколькими способами [1, стр.1]:

Как сумма ряда:

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

Каждую итерацию цикла суммирования вычисляются:

1)Факториал от n.

2)Результат деления единицы на ранее вычисленный факториал от n.

Реализация программы

- Для работы с вещественными числами были использованы сопроцессорные команды.
- Функция `CalcE` осуществляет вычисление числа Эйлера. Работает она следующим образом:
    - Обнуляется и инкрементируется регистр ecx, представляющий собой индексатор.
    - Запускается цикл суммирования, в котором каждую итерацию:
        - Проверяется равен ли регистр ecx значению по адресу N, где хранится количество итераций, необходимых для предельно точного вычисления значения числа e. Если равен, то цикл завершается.
        - В регистр сопроцессора ST(0) загружается значение по адресу A, где для удобства хранится номер итерации в вещественном формате.
        - Регистр ST(0) инкрементируется, дублируя результат по адресу A.
        - Регистр ST(0) умножается на последнее вычисленное значение факториала из прошлой итерации.
        - Результат записывается по адресу factorial. Регистр ST(0) стирается (выталкивается).
        - В регистр сопроцессора ST(0) загружается вещественное значения константного инкремента (всегда равное единице) и делится на значение по адресу factorial.
        - К тому же регистру прибавляется значение общей суммы. Результат записывается (стирая регистр ST(0)) по адресу result.
        - Регистр ecx инкрементируется.
    - Итоговое значение числа e сохранено по адресу result.
- Для вывода полученного числа e используется c-подобная функция printf.
- В процедуре `start`, находящейся в секции ‘.code’, реализован последовательный вызов процедур, а так же вызывается процедура `finish`, которая производит корректный выход из программы.

Тестирование программы

Продемонстрируем работу процедуры **CalcE**:

- Программа запущена, и выведен результат вычислений с точностью до 10 знаков после запятой. (рис.1)

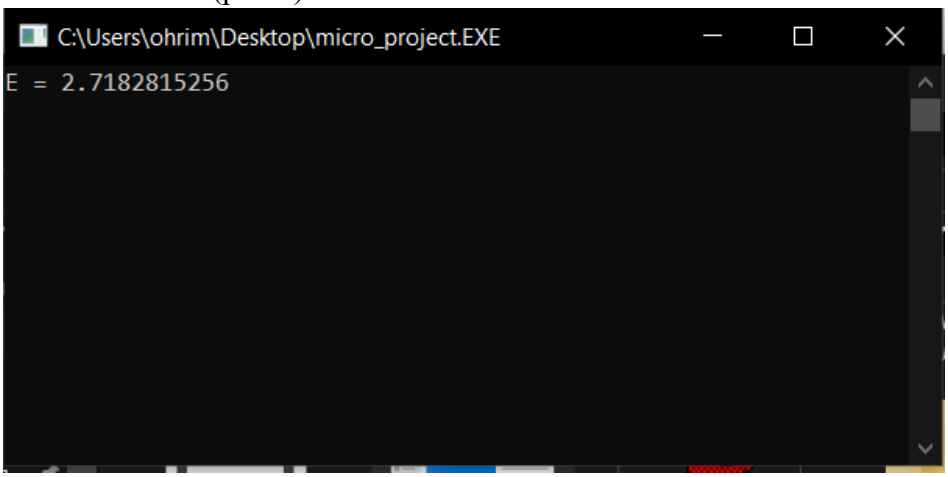

Рисунок 1 – демонстрация работы программы.

Корректность работы **CalcE** проверим при помощи стандартных арифметических операций в Wolfram Mathematica (рис. 2):

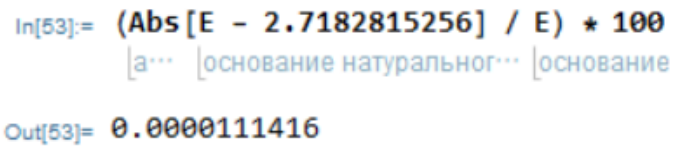

Рисунок 2 – демонстрация вычисления погрешности в Wolfram Mathematica.

Как видно из Рисунка 6, отклонение от табличного значения числа e состовляет менее 0,1% от самого числа e. Результат вычислений соответсвует заданию.

Использованная литература

1. e (число): Википедия (интернет источник) / https://ru.wikipedia.org/wiki/E_(%D1%87%D0%B8%D1%81%D0%BB%D0%BE)

Приложение

```
format PE console

entry start

include 'win32a.inc'

section '.data' data readable writable
        ;Строка форматирования для вывода результата вычислений
        strResult   db "E = %1.10f", 13, 10, 0

            A           dq  0.0     ;Вещественный индексатор
            result      dq  1.0     ;Результат вычислений (вещественный)
            N           dd  10      ;Количество итераций
            factorial   dq  1.0     ;Текущее значение факториала
            increment   dq  1.0     ;Вещественный константный инкремент

    section '.code' code readable executable
    ;Входная точка программы
    start:
            FINIT
            call CalcE

            invoke printf, strResult, dword[result], dword[result + 4]
    finish:
            call [getch]
            push 0
            call [ExitProcess]

    CalcE:
            xor ecx, ecx            ;Обнуление регистра индексатора
            inc ecx                 ;+1 т.к. счет результата начинается с 1
    startLoop:
            cmp ecx, [N]            ;Проверка на завершение цикла
            je endLoop
            FLD [A]             ;Загрузка вещественного индексатора в ST(0)
            FADD [increment]        ;Инкрементируем ST(0)
            FST [A]                 ;Результат сохраняем в A
```

```
38.             FMUL [factorial]        ;Умножаем индексатор на прошлое
    значение факториала
39.             FSTP [factorial]        ;Сохраняем новое значение факториала,
    очищ. ST(0)
40.
41.             FLD [increment]         ;Загружаем константный вещественный
    инкремент в ST(0)
42.             FDIV [factorial]        ;Делим его на новое значение факториала
43.             FADD [result]           ;Добавляем туда общий результат суммы
44.             FSTP [result]      ;Сохраняем новое значение суммы, очищ. ST(0)
45.
46.             inc ecx                 ;Инкрементируем регистр ecx
47.             jmp startLoop           ;Возвращаемся на начало цикла
48.     endLoop:
49.             ret  ;Основной стек никак не задействован, поэтому нет смысла
    его восстанавливать
50.
51.     section '.idata' import data readable
52.         library kernel, 'kernel32.dll',\
53.                 msvcrt, 'msvcrt.dll'
54.
55.         import kernel,\
56.                ExitProcess, 'ExitProcess'
57.         import msvcrt,\
58.                printf, 'printf',\
59.                sprintf, 'sprintf',\
60.                scanf, 'scanf',\
61.                getch, '_getch'
```