ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук

Департамент программной инженерии

Микропроект №1
Тема:
Программа для вычисления числа Пи с точностью не хуже 0.1% посредством дзета-функции Римана на языке ассемблер

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Исполнитель
Студент группы БПИ 199
_____/Кочик Д.А./
«___»_____2020 г.

Москва

2020

Содержание

1. Постановка задачи

Необходимо разработать программу для вычисления числа **Пи** с точностью не хуже 0.1%, посредством дзета-функции Римана (использовать FPU). Программа должна быть разработана на языке ассемблер.

2. Применяемые расчетные методы

Для вычисления числа Пи использовалась дзета-функция Римана[1] при s = 2. Таким образом программа вычисляет значение суммы ряда по ф. 1. Указанный в формуле ряд называется рядом обратных квадратов. Значение его суммы было посчитано [2] и равняется $\frac{\pi^2}{6}$.

$$\zeta(2) = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} \dots$$

Формула 1. Ряд обратных квадратов

Таким образом, алгоритм программы можно разделить на следующие этапы:

1) Вычисление суммы ряда по ф. 1 до тех пор, пока отношение текущей суммы ряда к сумме ряда на предыдущей итерации не становится меньше или равно чем 0.001 т.е. как пока точность вычисления не достигает 0.1%. В результате мы получаем значение, близкое к $\frac{\pi^2}{6}$.
2) Умножение полученной суммы на 6. В результате мы получаем значение, близкое к π^2.
3) Взятие положительного квадратного корня из значения, полученного на шаге 2. В результате мы получаем значение, близкое к π.
4) Подсчет ошибки через вычитание полученного приближенного значения π и уже известного точного.
5) Вывод известной информации.

3. Входные и выходные данные

Входные и выходные данные не требуется. Этим же обусловлено отсутствие тестовых примеров.

4. Использованные источники

1) Дзета-функция Римана [Электронный ресурс]//URL: https://ru.wikipedia.org/wiki/%D0%94%D0%B7%D0%B5%D1%82%D0%B0-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_%D0%A0%D0%B8%D0%BC%D0%B0%D0%BD%D0%B0 (дата обращения 20.10.2020).
2) Ряд обратных квадратов [Электронный ресурс]//URL: https://ru.wikipedia.org/wiki/%D0%A0%D1%8F%D0%B4_%D0%BE%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D1%8B%D1%85_%D0%BA%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D0%BE%D0%B2 (дата обращения 20.10.2020).

5. Приложение 1

Текст программы

```
format PE console
;-----------------------------------------------------------------------------------------------------------
;Кочик Дмитрий Алексеевич
;БПИ199
;Вариант 11
;-----------------------------------------------------------------------------------------------------------
entry start
include 'win32a.inc'



section '.data' data readable writable
        sum                 dq 0.0
        curr_num            dq 0.0
        curr_num2           dq ?
        answer_val          dq ?
        coef                dq 0.001
        coef_pi_counting    dq 6.0
        pos                 dd ?
        counter             dd 1
        sth                 dd 1000000

        raw_result          db "PI^2/6 is: %f",10 ,0
        raw_result2         db "PI^2 is: %f",10 ,0
        res                 db "The result is: %f",10 ,0
        answer              db "Correct answer is: %f", 10, 0
        inaccuracy          db "Inaccuracy is: %f*10^-6", 10, 0

        inf db "sth: %f", 10, 0

section '.code' code readable executable
    start:
```

```
        FINIT

        call countSum       ; Вычисляем примерное значение дзета-функции Римана для s=2.
        call countPiVal     ; Зная примерное значение ф-ии, вычисляем примерное значение числа пи.
        call getAnswer      ; Выводим ответ.

        invoke getch
        invoke ExitProcess  ; Завершаем выполнение программы.
;-------------------------------------------------------------------------------------------------------------
    countSum:
        mov [pos], esp      ; Запоминаем позиция функции в стеке при вызове.

    startCycle:             ; Начинаем цикл для подсчета дзета-функции Римана.
        FLD [sum]
        FSTP [curr_num2]    ; Запоминаем старое значение суммы.


        FILD [counter]
        FST [curr_num]
        FMUL [curr_num]
        FSTP [curr_num]     ; Вычисляем i-й элемент суммы.

        FLD1
        FDIV [curr_num]     ; i-й элемент равен 1/(i^2).
        FADD [sum]
        FSTP [sum]          ; Обновляем значение суммы.

        mov eax, [counter]
        inc eax
        mov [counter], eax  ; увеличиваем счетчик итераций цикла.

        FLD [curr_num2]
        FDIV [sum]
```

```
    FCOMP [coef]        ; Если отношение суммы предыдущей итерации и текущей больше чем 0.1%
    jge startCycle      ; Продолжаем выполнение цикла.

  finish:
    mov esp, [pos]      ; Иначе прекращаем выполнение.
    ret
;-------------------------------------------------------------------------------------------------------------
  countPiVal:
    mov [pos], esp      ; Запоминаем позиция функции в стеке при вызове.

    invoke printf, raw_result, dword[sum], dword[sum+4]  ; Выводим информацию о вычисленном значении дзета-функции Римана при s=2.
                        ; sum = Pi^2/6
    FLD [sum]
    FMUL [coef_pi_counting]  ; Умножаем сумму на 6.
    FST [sum]
    invoke printf, raw_result2, dword[sum], dword[sum+4] ; Выводим информацию о вычисленном значении дзета-функции Римана при s=2, умноженной на 6.
    FSQRT
    FSTP [sum]          ; Вычисляем корень из sum*6. Это и есть число пи.

    mov esp, [pos]
    ret
;-------------------------------------------------------------------------------------------------------------
  getAnswer:
    mov [pos], esp      ; Запоминаем позиция функции в стеке при вызове.

    invoke printf, res, dword[sum], dword[sum+4]                ; Выводим информацию о том, что нам удалось вычислить.

    FLDPI
    FST [answer_val]
    invoke printf, answer, dword[answer_val], dword[answer_val+4]     ; Выводим правильный ответ для сравнения.
```

```
        FLD [sum]
        FSUB [answer_val]
        FIMUL [sth]
        FSTP [answer_val]

        invoke printf, inaccuracy, dword[answer_val], dword[answer_val+4]  ; Выводим полученную погрешность.

        mov esp, [pos]
        ret
;--------------------------------------------------------------------------------------------------------------
section '.idata' import data readable

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

include 'api\user32.inc'
include 'api\kernel32.inc'
    import kernel,\
           ExitProcess, 'ExitProcess',\
           HeapCreate,'HeapCreate',\
           HeapAlloc,'HeapAlloc'
    import msvcrt,\
           printf, 'printf',\
           scanf, 'scanf',\
           getch, '_getch'
```