АВС ИДЗ 2, Вариант 3

Власов Николай Алексеевич, БПИ229

https://github.com/qwatro2/computing-systems-architecture-ihw2

Условие

Разработать программу, вычисляющую с помощью степенного ряда с точностью не хуже $0,1\%$ значение функции $\cos x$ для заданного параметра x.

Описание метода решения задачи

Чтобы уменьшить количество вычислений, будем запоминать предыдущее слагаемое ряда и домножать его на $\frac{-x^2}{2k(2k-1)}$. Тогда имеем:

$$\frac{(-1)^{k-1}x^{2k-2}}{(2k-2)!} * \frac{-x^2}{2k(2k-1)} = \frac{(-1)^k x^{2k}}{(2k)!}$$

- следующее слагаемое ряда.
Ряд мы складываем в одной переменной и смотрим на отношение текущего слагаемого ко всей сумме ряда. Если оно соответствует заданной точности $0.1\% = 0.001$, останавливаем алгоритм.

Тесты, демонстрирующие проверку разработанных программ.

Листинг 1: Код, генерирующий тесты

```
from math import cos, pi
from random import random, seed
seed(10)
# random returns float from [0; 1) => pi_random returns float from [-pi, pi)
pi_random = lambda: (2 * random() - 1) * pi

def solve(x: float):
    cos_x = 1
    prev = 1
    cur = 0
    k = 1
    eps = 0.001

    while True:
        m = - x * x / ((2*k - 1)*(2*k))
        cur = prev * m

        if abs(cur / cos_x) < eps:
            cos_x += cur
            break

        cos_x += cur
        prev = cur
        k += 1


    return cos_x
```

```
28 def generate():
29     n = 10
30     dmn = [pi_random() for _ in range(n)]
31     rng = [solve(x) for x in dmn]
32 
33     true = [cos(x) for x in dmn]
34 
35     for x, our, py in zip(dmn, rng, true):
36         print(f"x = {x}, therefore our cos(x) = {our}.\tPython cos(x) = {py}")
37 
38 if __name__ == "__main__":
39     generate()
```

```
x = 0.44863573385016364, therefore our cos(x) = 0.9010396323447458.     Python cos(x) = 0.901039672957227
x = -0.4468032468449707, therefore our cos(x) = 0.9018329371885391.     Python cos(x) = 0.9018329764934756
x = 0.49066211590664083, therefore our cos(x) = 0.8820209732443182.     Python cos(x) = 0.8820210563400134
x = -1.8466392695751883, therefore our cos(x) = -0.2723613319386085.    Python cos(x) = -0.2723581097577735
x = 1.9686554829554352, therefore our cos(x) = -0.3874525299837752.     Python cos(x) = -0.3874456036669677
x = 2.033168849468953, therefore our cos(x) = -0.44608294483717587.     Python cos(x) = -0.44607276037750276
x = 0.9642963700634886, therefore our cos(x) = 0.5699953437677615.      Python cos(x) = 0.5699951535339216
x = -2.1348406582948445, therefore our cos(x) = -0.5346079651821817.    Python cos(x) = -0.5346084249148925
x = 0.1298694167983847, therefore our cos(x) = 0.9915788199636711.      Python cos(x) = 0.9915788133020788
x = -1.0821353395129152, therefore our cos(x) = 0.46944461316083647.    Python cos(x) = 0.46944401169953215
```

Результаты тестовых прогонов для различных исходных данных.

```
Enter 0 to exit, 1 to run the program or 2 to run the tests: 2
Simple tests:
x = 0.0, therefore cos(x) = 1.0
x = 0.5235987, therefore cos(x) = 0.8660253
x = 0.7853981, therefore cos(x) = 0.7071033
x = 1.0471975, therefore cos(x) = 0.50000054
x = 1.5707963, therefore cos(x) = 9.709748E-8
x = 2.0943952, therefore cos(x) = -0.5000146

Random tests:
x = 0.44863573, therefore cos(x) = 0.9010396
x = -0.44680324, therefore cos(x) = 0.901833
x = 0.49066213, therefore cos(x) = 0.88202095
x = -1.8466393, therefore cos(x) = -0.27236134
x = 1.9686555, therefore cos(x) = -0.38745254
x = 2.0331688, therefore cos(x) = -0.44608286
x = 0.96429634, therefore cos(x) = 0.56999534
x = -2.1348407, therefore cos(x) = -0.53460795
x = 0.12986942, therefore cos(x) = 0.9915788
x = -1.0821353, therefore cos(x) = 0.46944466
Enter 0 to exit, 1 to run the program or 2 to run the tests: 0
```

```
Enter 0 to exit, 1 to run the program or 2 to run the tests: 1
Enter x to calculate cos(x): 1
x = 1.0, therefore cos(x) = 0.5403026
Enter 0 to exit, 1 to run the program or 2 to run the tests: 1
Enter x to calculate cos(x): 12
x = 12.0, therefore cos(x) = 0.8437766
Enter 0 to exit, 1 to run the program or 2 to run the tests: 1
Enter x to calculate cos(x): 4.221
x = 4.221, therefore cos(x) = -0.47185206
Enter 0 to exit, 1 to run the program or 2 to run the tests: 1
Enter x to calculate cos(x): 3.14
x = 3.14, therefore cos(x) = -1.0000027
Enter 0 to exit, 1 to run the program or 2 to run the tests: 1
Enter x to calculate cos(x): 2.6179938779914944
x = 2.6179938, therefore cos(x) = -0.86601764
Enter 0 to exit, 1 to run the program or 2 to run the tests: 0

-- program is finished running (0) --
```

Как видно, все тесты программа проходит успешно

Тексты программы на языке ассемблера.

Листинг 2: main.asm

```
.include "macrolib.asm"

.text
    .globl main
    main:
        print_str("Enter 0 to exit, 1 to run the program or 2 to run the tests: ")
        input_int(t2)  # input command

        li t0 1
        li t1 2

        beqz t2 jal_exit  # if t2 == 0 go to exit
        beq t2 t0 jal_user  # if t2 == 1 go to user input
        beq t2 t1 jal_test  # if t2 == 2 go to running tests

        print_str("Wrong choice. You should enter number 0, 1 or 2.\n")  # else wrong command

        j main  # repeat solution

    jal_exit:
        j exit  # go to exit

    jal_user:
        jal run_user  # call function for user input
        j main  # repeat solution

    jal_test:
        jal run_test  # call function for running tests
        j main  # repeat solution

    exit:  # exit
        li a7 10
        ecall
```

Листинг 3: user.asm

```
.include "macrolib.asm"

.text
.globl run_user
run_user:
        stack_push(ra)  # write ra because we will use jal later

    jal input_x  # call function for reading x (return: x in fa0)
    jal solve  # call function for solving problem (params: x in fa0; return: result in fa1)
    jal output_result  # call function for printing result (params: x in fa0, cos(x) in fa1)

        stack_pop(ra)  # recover ra
    ret
```

Листинг 4: macrolib.asm

```
.macro print_str (%x)  # macro for printing screen in-place
   .data
                str: .asciz %x
   .text
                li a7, 4
                la a0, str
                ecall
   .end_macro

.macro print_float(%reg)  # macro for printing float from register %reg
    fmv.s fa0 %reg
    li a7 2
    ecall
.end_macro

.macro input_float(%reg)  # macro for reading float and writing to register %reg
    li a7 6
    ecall
    fmv.s %reg fa0
.end_macro

.macro input_int(%reg)  # macro for reading int and writing to register %reg
        li a7 5
        ecall
        mv %reg a0
.end_macro

.macro stack_push(%reg)  # macro for push value from register reg on stack
        addi sp sp -4
        sw %reg (sp)
.end_macro

.macro stack_pop(%reg)  # macro for pop address from stack to register reg
        lw %reg (sp)
        addi sp sp 4
.end_macro
```

Листинг 5: output.asm

```
.include "macrolib.asm"

.text
.globl output_result
output_result:  # params: x in fa0, cos(x) in fa1
    print_str("x = ")  # print promt
        print_float(fa0)
        print_str(", therefore cos(x) = ")
        print_float(fa1)
    print_str("\n")
    ret
```

Листинг 6: input.asm

```
.include "macrolib.asm"

.text
.globl input_x
input_x:  # return: x in fa0
    print_str("Enter x to calculate cos(x): ")  # print promt
    input_float(fa0)  # write x to fa0
    ret
```

Листинг 7: solution.asm

```
.include "macrolib.asm"
.data
        eps: .float 0.001

.text
.globl solve
solve:  # params: x in fa0; return: result in fa1
        flw ft0 eps t0 # epsilon
        li t1 1
        li t2 -1
        fcvt.s.w ft7 t2  # const -1
        fcvt.s.w fa1 t1  # result
        fcvt.s.w ft1 t1  # previous term
        fcvt.s.w ft2 zero  # current term

        li t1 1  # iteration variable

        while:
                add t2 t1 t1
                addi t3 t2 -1
                mul t2 t2 t3
                fcvt.s.w ft3 t2  # denumerator 2k*(2k - 1)
                fmul.s ft4 fa0 fa0
                fmul.s ft4 ft4 ft7  # numerator -x^2
                fdiv.s ft5 ft4 ft3  # m

                fmul.s ft2 ft1 ft5  # cur = prev * m

                fdiv.s ft6 ft2 fa1  # ft6 = cur / res
                fabs.s ft6 ft6  # ft6 = abs(cur / res)
                flt.s t2 ft6 ft0
                bnez t2 break  # t2 = 1 <=> ft6 < ft0 <=> abs(cur / res) < eps

                fadd.s fa1 fa1 ft2  # res += ft2
                fmv.s ft1 ft2  # prev = cur
                addi t1 t1 1  # k += 1
                j while

        break:
                fadd.s fa1 fa1 ft2  # res += ft2

                ret
```

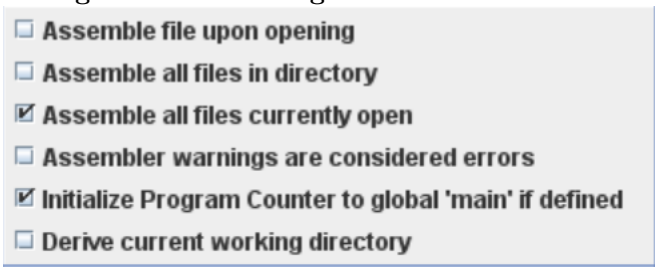
Листинг 8: test.asm

```
.include "macrolib.asm"

# this macro is using for simplify writing tests, I deliberately did not place it in the mac
.macro test(%x)
        .data
                x: .float %x
        .text
                flw fa0 x t0
                jal solve  # call function for solving problem (params: x in fa0; return: res
                jal output_result  # call function for printing result (params: x in fa0, cos
.end_macro

.text
.globl run_test
run_test:
        stack_push(ra) # write ra because we will use jal later

        print_str("Simple tests:\n")
        test(0)
        test(0.5235987)  # pi / 6
        test(0.7853981)  # pi / 4
        test(1.0471975)  # pi / 3
        test(1.5707963)  # pi / 2
        test(2.0943951)  # 2pi / 3

        print_str("\nRandom tests:\n")
        test(0.44863573385016364)
        test(-0.4468032468449707)
        test(0.49066211590664083)
        test(-1.8466392695751883)
        test(1.9686554829554352)
        test(2.033168849468953)
        test(0.9642963700634886)
        test(-2.1348406582948445)
        test(0.1298694167983847)
        test(-1.0821353395129152)

        stack_pop(ra)  # recover ra
        ret
```

Дополнительная информация, подтверждающая выполнение задания в соответствие требованиям на предполагаемую оценку.

- Приведено решение задачи на ассемблере. Ввод данных осуществляется с клавиатуры. Вывод данных осуществляется на дисплей. ✓
- В программе должны присутствовать комментарии, поясняющие выполняемые действия. ✓
- Допускается использование требуемых подпрограмм без параметров и локальных переменных. ✓
- В отчете должно быть представлено полное тестовое покрытие. Приведены результаты тестовых прогонов. Например, с использованием скриншотов. ✓
- В программе необходимо использовать подпрограммы с передачей аргументов через параметры, что обеспечивает их повторное использование с различными входными аргументами. При нехватке регистров, используемых для передачи параметров, оставшиеся параметры передавать через стек. ✓
- Внутри подпрограмм необходимо использовать локальные переменные. При нехватке временных регистров обеспечить сохранение данных на стеке в соответствии с соглашениями, принятыми для процессора. **- не потребовалось (кроме записи ra)** ✓
- В местах вызова функции добавить комментарии, описывающие передачу фактических параметров и перенос возвращаемого результата. При этом необходимо отметить, какая переменная или результат какого выражения соответствует тому или иному фактическому параметру. ✓
- Разработанные подпрограммы должны поддерживать многократное использование с различными наборами исходных данных, включая возможность подключения различных исходных и результирующих массивов. **- функция main позволяет поддерживать многократное использование программы** ✓
- Реализовать автоматизированное тестирование за счет создания дополнительной тестовой программы, осуществляющей прогон подпрограммы обработки массивов с различными тестовыми данными (вместо ввода данных). Осуществить прогон тестов обеспечивающих покрытие различных ситуаций. Тестовые данные можно формировать в различных исходных массивах. ✓
- Для дополнительной проверки корректности вычислений осуществить аналогичные тестовые прогоны с использованием существующих библиотек и одного из языков программирования высокого уровня по выбору: C, C++, Python. ✓
- Добавить в программу использование макросов для реализации ввода и вывода данных. Макросы должны поддерживать повторное использование с различными массивами и другими параметрами. ✓
- Программа должна быть разбита на несколько единиц компиляции. При этом подпрограммы ввода–вывода должны составлять унифицированные модули, используемые повторно как в программе, осуществляющей ввод исходных данных, так и в программе, осуществляющей тестовое покрытие. ✓
- Макросы должны быть выделены в отдельную автономную библиотеку. **- macrolib.asm** ✓

ВАЖНО!!! В эмуляторе RARS включить компляцию всех открытых файлов и "Initialize Program Counter to global 'main' if defined"



В репозитории на гитхабе лежит еще и реализация на double, не знал, какая требуется :)

Ссылки на источники информации с описанием метода решения

Виленкин Н.Я., Куницкая Е.С., Мордкович А.Г. Ряды. Математический анализ. Учебное пособие для студентов-заочников III курса физико-математических факультетов педагогических институтов. - М.: Просвещение, 1982. - 160 с. - 31 с., (5.8)

$$\cos x = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k}}{(2k)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots + \frac{(-1)^k x^{2k}}{(2k)!} + \ldots . \quad (5.8)$$