

Индивидуальное домашнее задание-2 (Гринченко Евгений, БПИ 236, вариант 16)

Небольшое предисловие перед началом - код программы я реализовывал сразу на оценку 8-9, поэтому не удалось соблюдать условия итеративности. Постараюсь отразить в этом отчёте выполняемость критериев на оценку 4-7. Также хотел бы добавить, что мне нужно было по заданию вычислить e^{-x} через степенной ряд, я использовал такой подход - я вычисляю e^x через степенной ряд, потом произвожу вычисление $1/e^x$, если введённый x изначально был отрицательным, то вычисление $1/e^x$ не производится

Финальный вариант программы, которую я реализовал, чтобы претендовать на оценку 10, был разбит на 3 ассемблерных файла:

1)Файл **data.s** в этой программе отвечает за объявление и инициализацию данных, используемых в процессе выполнения.

```
.data
number1:  .double 10.0          # Входное значение для вычисления  $e^{-x}$ 
number2:  .double 0
number3:  .double -1.0
number4:  .double 2.3
number5:  .double -2.3
number6:  .double 0.6
number7:  .double 0.8
accuracy: .double 0.0000000005   # Погрешность для ряда Тейлора
result:   .double 0.0           # Переменная для хранения результата  $e^{-x}$ 
start_game: .asciz "Вы хотите ввести данные с клавиатуры или запустить автоматическое тестирование(0-с
клавиатуры, 1 - автоматическое тестирование) "
input_number: .asciz "Введите число для вычислений: "
error_start: .asciz "Wrong number. Repeat input of variable, please!"
repeat: .asciz "Вы хотите закончить программу или заново пройти(0-закончить, 1 - заново пройти)"
test: .asciz "---NEW TEST---"
test_input: .asciz "Для x = "
msg_result: .asciz "Результат  $e^{-x}$ : "
ln : .asciz "\n"
.align 3
```

2)Файл **macros.inc** выполняет вспомогательные функции, которые упрощают код основной программы за счет использования макросов для повторяющихся операций.

```

.include "data.s"
# Макрос для возведения числа в целую степень
.macro pow(%number, %pow, %result)
    li t1, 1
    fcvt.d.w ft10, t1
    mv t0, %pow
    bnez t0, continue
    j end
continue:
    fmv.d ft11, %number
loop:
    beqz t0, end
    fmul.d ft10, ft10, ft11
    addi t0, t0, -1
    j loop
end:
    fmv.d %result, ft10
.end_macro

.macro print_str, (%str)
    la a0, %str    # Загрузить адрес строки
    li a7, 4       # Системный вызов для вывода строки
    ecall
.end_macro

.macro input_num
    li a7, 5       # Системный вызов для ввода числа
    ecall
.end_macro

# Макрос для нахождения факториала числа
.macro factorial(%number, %result)
    li t5, 1
    fcvt.d.w ft10, t5
    fcvt.d.w ft11, %number
    fcvt.w.d t6, ft11
    addi t6, t6, 1
loop:
    beq t6, t5, end
    addi t6, t6, -1
    fcvt.d.w ft11, t6
    fmul.d ft10, ft10, ft11
    j loop
end:
    fmv.d %result, ft10
.end_macro

```

```

# Макрос для вычисления e^x через ряд Тейлора
.macro e_x(%x, %accuracy, %result)
    li t3, 1
    fcvt.d.w ft8, t3
    fadd.d ft8, ft8, %accuracy
    li t3, 1
    fcvt.d.w ft7, t3
    li t3, 1
    fcvt.d.w ft2, t3
    fadd.d ft2, ft2, %x
    li t3, 2
loop:
    pow(%x, t3, ft3)
    factorial(t3, ft4)
    fdiv.d ft3, ft3, ft4
    fadd.d ft2, ft2, ft3
    fdiv.d ft6, ft2, ft7
    flt.d t4, ft8, ft6
    fmv.d ft7, ft2
    addi t3, t3, 1
    beq t4, zero, end
    j loop
end:
    fmv.d %result, ft2
.end_macro

.macro print_double(%reg, %addr)
    fld %reg, %addr, t0          # Загрузить значение из %addr в %reg
    li a7, 3                     # Системный вызов для вывода double
    ecall
.end_macro

```

3) Файл `text.s` содержит основную логику программы, которая обрабатывает ввод данных, выполнение автоматических тестов. Он также включает управление завершением программы и повторной работой.

```

.include "macros.inc"
.text
.globl main

main:
    print_str start_game
    input_num

```

```

    beqz a0, _start
    li, t1, 1
    beq a0, t1, auto_test
    print_str error_start
    print_str ln
    j main
auto_test:
    #test1
    print_str test
    print_str ln
    fld fa0, number1, t0
    fld fa1, accuracy, t0
    jal ra, e_func
    fsd fa2, result, t0          # Результат в памяти
    # Печать результата
    print_str test_input
    print_double fa0, number1
    print_str ln
    print_str msg_result
    print_double fa0, result
    print_str ln
    #test2
    print_str test
    print_str ln
    fld fa0, number2, t0
    fld fa1, accuracy, t0
    jal ra, e_func
    fsd fa2, result, t0          # Результат в памяти
    # Печать результата
    print_str test_input
    print_double fa0, number2
    print_str ln
    print_str msg_result
    print_double fa0, result
    print_str ln
    #test3
    print_str test
    print_str ln
    fld fa0, number3, t0
    fld fa1, accuracy, t0
    jal ra, e_func
    fsd fa2, result, t0          # Результат в памяти
    # Печать результата
    print_str test_input
    print_double fa0, number3
    print_str ln

```

```
print_str msg_result
print_double fa0,result
print_str ln
#test4
print_str test
print_str ln
fld fa0,number4,t0
fld fa1, accuracy, t0
jal ra, e_func
fsd fa2, result, t0      # Результат в памяти
# Печать результата
print_str test_input
print_double fa0, number4
print_str ln
print_str msg_result
print_double fa0,result
print_str ln
#test5
print_str test
print_str ln
fld fa0,number5,t0
fld fa1, accuracy, t0
jal ra, e_func
fsd fa2, result, t0      # Результат в памяти
# Печать результата
print_str test_input
print_double fa0, number5
print_str ln
print_str msg_result
print_double fa0,result
print_str ln
#test6
print_str test
print_str ln
fld fa0,number6,t0
fld fa1, accuracy, t0
jal ra, e_func
fsd fa2, result, t0      # Результат в памяти
# Печать результата
print_str test_input
print_double fa0, number6
print_str ln
print_str msg_result
print_double fa0,result
print_str ln
#test7
```

```

print_str test
print_str ln
fld fa0,number7,t0
fld fa1, accuracy, t0
jal ra, e_func
fsd fa2, result, t0          # Результат в памяти
# Печать результата
print_str test_input
print_double fa0, number7
print_str ln
print_str msg_result
print_double fa0,result
print_str ln
j repeat_game
# Основная программа
_start:
    print_str input_number
    li a7, 7
    ecall
    fld fa1, accuracy, t0      # Точность в fa1
    # Вызов подпрограммы e_func
    jal ra, e_func
    # Сохранение результата
    fsd fa2, result, t0       # Результат в памяти
    # Печать результата
    print_str msg_result
    print_double fa0,result
    print_str ln
    j repeat_game

repeat_game:
    print_str repeat
    input_num
    beqz a0, end
    li, t1,1
    beq a0, t1, main
    print_str error_start
    print_str ln
    j repeat_game
end:
    # Завершение программы
    li a7, 10                 # Системный вызов для завершения программы
    ecall

# Подпрограмма для вычисления e^(-x)
e_func:

```

```

# Инициализация регистров
fcvt.d.w fs10, zero      # fs10 = 0
fcvt.d.w fs11, zero      # fs11 = 0
fadd.d fs11, fs11, fa0    # fs11 = number
fabs.d fs10, fs11        # fs10 = abs(number)

# Проверка знака аргумента
feq.d s0, fs10, fs11      # s0 = 1, если number > 0; s0 = 0, если number < 0

# Вызов e_x для расчета e^x
e_x(fs10, fa1, ft5)      # e^x -> ft5
fmv.d ft0, ft5
beqz s0, e_end
fmv.d ft5, ft0
li t0, 1                  # t0 = 1 для конвертации в double
fcvt.d.w ft3, t0          # ft3 = 1.0
fdiv.d ft3, ft3, ft5      # ft3 = e^(-x)
fmv.d ft0, ft3

e_end:
fmv.d fa2, ft0            # Возвращаем результат в fa2
ret                      # Завершение подпрограммы

```

Вычисление e^x производится по формуле

$$e^x = \frac{x^n}{n!}$$

за счёт реализованных макросов.

1. Макросы для математических вычислений

`pow` — Возведение числа в степень

Этот макрос реализует простое возведение в степень для целого числа. Он используется для расчёта членов ряда Тейлора.

- **Аргументы:**
 - `%number` : основание степени, число, которое нужно возвести в степень.
 - `%pow` : показатель степени.
 - `%result` : переменная для хранения результата.

factorial — Нахождение факториала

Этот макрос вычисляет факториал целого числа, необходимый для знаменателя каждого члена ряда Тейлора.

- **Аргументы:**

- `%number` : число, факториал которого нужно найти.
- `%result` : переменная для хранения результата.

e_x — Вычисление экспоненты e^x

Этот макрос вычисляет e^x через сумму ряда Тейлора, добавляя члены до тех пор, пока текущая точность не достигнет заданного значения.

- **Аргументы:**

- `%x` : значение для возведения в степень.
- `%accuracy` : значение точности, до которой будет выполнено приближение.
- `%result` : переменная для хранения результата.

2. Макросы для работы с вводом и выводом данных

print_str — Вывод строки

Этот макрос отвечает за вывод строки на дисплей, показывая пользователю сообщения, такие как приглашение ввести данные или результат вычислений.

- **Аргументы:**

- `%str` : строка, которую нужно вывести.

input_num — Ввод числа

Этот макрос запрашивает у пользователя ввод числа, обрабатывая его, чтобы использовать далее в вычислениях.

print_double — Вывод вещественного числа

Макрос выводит на экран вещественное число, которое было вычислено программой.

Основная программа в этом коде организована следующим образом:

1. **Запуск и выбор режима работы:**

- При запуске программы пользователю отображается сообщение с выбором режима: ввести данные с клавиатуры или запустить автоматическое тестирование. Это

сообщение выводится с помощью макроса `print_str`, который показывает строку `start_game`.

- После выбора (ввод значения `0` или `1`) программа проверяет введённое значение и, в зависимости от него, переходит к одному из режимов: `_start` для ввода с клавиатуры или `auto_test` для тестирования. Если введено некорректное значение, то запрашивается повторный ввод данных.

2. Режим автоматического тестирования:

- Если выбран режим автотестирования (`auto_test`), программа последовательно выполняет вычисления e^{-x} для нескольких заранее определённых значений, таких как `number1`, `number2`, и так далее.
- Каждое значение загружается в регистр и передаётся в подпрограмму `e_func`, которая вычисляет e^{-x} .
- По завершении каждого вычисления результат сохраняется в сегменте `.data` в переменной `result`.
- Затем результат отображается на экране, используя макрос `print_double` для печати результата и `print_str` для вывода сообщения.

3. Режим ввода данных с клавиатуры:

- В этом режиме пользователь вручную вводит значение для вычислений. Программа запрашивает ввод числа, используя макрос `input_num`, а затем передаёт это значение вместе с точностью (значение `accuracy`) в подпрограмму `e_func`.
- Результат вычисления также сохраняется в переменной `result` и затем выводится на экран через `print_str` и `print_double`.

4. Повторное выполнение или завершение программы:

- После выполнения вычислений и вывода результата пользователю предлагается решить, завершить программу или выполнить её заново.

- Для этого выводится сообщение ``gereat``, после чего пользователь вводит значение ``0`` или ``1``. Если введено ``0``, программа завершает выполнение, вызывая системный вызов завершения (``li a7, 10; syscall``). Если введено ``1``, программа повторяет весь процесс, возвращаясь в начало (метка ``main``). В случае неправильного ввода отображается сообщение об ошибке ``error_start``, и запрос на ввод повторяется.

```

Вы хотите ввести данные с клавиатуры или запустить автоматическое тестирование(0-с клавиатуры, 1 - автоматическое тестирование) -1
Wrong number. Repeat input of variable, please!
Вы хотите ввести данные с клавиатуры или запустить автоматическое тестирование(0-с клавиатуры, 1 - автоматическое тестирование) 2
Wrong number. Repeat input of variable, please!
Вы хотите ввести данные с клавиатуры или запустить автоматическое тестирование(0-с клавиатуры, 1 - автоматическое тестирование) 3
Wrong number. Repeat input of variable, please!
Вы хотите ввести данные с клавиатуры или запустить автоматическое тестирование(0-с клавиатуры, 1 - автоматическое тестирование) 4
Wrong number. Repeat input of variable, please!
Вы хотите ввести данные с клавиатуры или запустить автоматическое тестирование(0-с клавиатуры, 1 - автоматическое тестирование) 0
Введите число для вычислений: -3
Результат e^(-x): 20.085536921517665
Вы хотите закончить программу или заново пройти(0-закончить, 1 - заново пройти)-1
Wrong number. Repeat input of variable, please!
Вы хотите закончить программу или заново пройти(0-закончить, 1 - заново пройти)2
Wrong number. Repeat input of variable, please!
Вы хотите закончить программу или заново пройти(0-закончить, 1 - заново пройти)1
Вы хотите ввести данные с клавиатуры или запустить автоматическое тестирование(0-с клавиатуры, 1 - автоматическое тестирование) 1
---NEW TEST---
Для x = 10.0
Результат e^(-x): 4.539992977005168E-5
---NEW TEST---
Для x = 0.0
Результат e^(-x): 1.0
---NEW TEST---
Для x = -1.0
Результат e^(-x): 2.7182818284467594
---NEW TEST---
Для x = 2.3
Результат e^(-x): 0.10025884372859327
---NEW TEST---
Для x = -2.3
Результат e^(-x): 9.974182454238752
---NEW TEST---
Для x = 0.6
Результат e^(-x): 0.5488116360954611
---NEW TEST---
Для x = 0.8
Результат e^(-x): 0.44932896411911155
Вы хотите закончить программу или заново пройти(0-закончить, 1 - заново пройти)0

-- program is finished running (0) --

Вы хотите ввести данные с клавиатуры или запустить автоматическое тестирование(0-с клавиатуры, 1 - автоматическое тестирование) 1
---NEW TEST---
Для x = 10.0
Результат e^(-x): 4.539992977005168E-5
---NEW TEST---
Для x = 0.0
Результат e^(-x): 1.0
---NEW TEST---
Для x = -1.0
Результат e^(-x): 2.7182818284467594
---NEW TEST---
Для x = 2.3
Результат e^(-x): 0.10025884372859327
---NEW TEST---
Для x = -2.3
Результат e^(-x): 9.974182454238752
---NEW TEST---
Для x = 0.6
Результат e^(-x): 0.5488116360954611
---NEW TEST---
Для x = 0.8
Результат e^(-x): 0.44932896411911155
Вы хотите закончить программу или заново пройти(0-закончить, 1 - заново пройти)1
Вы хотите ввести данные с клавиатуры или запустить автоматическое тестирование(0-с клавиатуры, 1 - автоматическое тестирование) 0
Введите число для вычислений: -3
Результат e^(-x): 20.085536921517665
Вы хотите закончить программу или заново пройти(0-закончить, 1 - заново пройти)0

-- program is finished running (0) --

```

Также было реализовано вычисление значений на питоне, для самопроверки.

main.py	Output
<pre>1 import math 2 3 number1 = 10.0 4 number2 = 0 5 number3 = -1.0 6 number4 = 2.3 7 number5 = -2.3 8 number6 = 0.6 9 number7 = 0.8 10 print(f"Точное значение e^{(-number1)} = {math.exp(-number1)}") 11 print(f"Точное значение e^{(-number2)} = {math.exp(-number2)}") 12 print(f"Точное значение e^{(-number3)} = {math.exp(-number3)}") 13 print(f"Точное значение e^{(-number4)} = {math.exp(-number4)}") 14 print(f"Точное значение e^{(-number5)} = {math.exp(-number5)}") 15 print(f"Точное значение e^{(-number6)} = {math.exp(-number6)}") 16 print(f"Точное значение e^{(-number7)} = {math.exp(-number7)}")</pre>	<pre>Точное значение e^{(-10.0)} = 4.5399929762484854e-05 Точное значение e^{(-0)} = 1.0 Точное значение e^{(-1.0)} = 2.718281828459045 Точное значение e^{(-2.3)} = 0.10025884372280375 Точное значение e^{(-2.3)} = 9.974182454814718 Точное значение e^{(-0.6)} = 0.5488116360940264 Точное значение e^{(-0.8)} = 0.44932896411722156 === Code Execution Successful ===</pre>

Тестовые данные в самой программе были взяты точно такие же.

```
number1:      .double 10.0           # Входное значение для вычисления e^{(-x)}
number2:      .double 0
number3:      .double -1.0
number4:      .double 2.3
number5:      .double -2.3
number6:      .double 0.6
number7:      .double 0.8

Вы хотите ввести данные с клавиатуры или запустить автоматическое тестирование (0 - с клавиатуры, 1 - автоматическое тестирование) 1
---NEW TEST---
Для x = 10.0
Результат e^{(-x)}: 4.539992977005168E-5
---NEW TEST---
Для x = 0.0
Результат e^{(-x)}: 1.0
---NEW TEST---
Для x = -1.0
Результат e^{(-x)}: 2.7182818284467594
---NEW TEST---
Для x = 2.3
Результат e^{(-x)}: 0.10025884372859327
---NEW TEST---
Для x = -2.3
Результат e^{(-x)}: 9.974182454238752
---NEW TEST---
Для x = 0.6
Результат e^{(-x)}: 0.5488116360954611
---NEW TEST---
Для x = 0.8
Результат e^{(-x)}: 0.44932896411911155
```

Как можем заметить выводы обеих программ практически идентичны.

Заключение.

Разработанный код на ассемблере RISC-V реализует автоматизированный и интерактивный режимы вычисления экспоненциальной функции e^{-x} с использованием ряда Тейлора. Структура программы построена на макросах, что позволяет переиспользовать и облегчить написание кода для выполнения математических операций, таких как возведение в степень, нахождение факториала, а также вычисление значений экспоненты. Макросы и подпрограммы делают код гибким и удобным для масштабирования.

Всем добра!

