

Отчет
Индивидуальное задание №2. Арифметический сопроцессор
Вариант № 6

Выполнила: Захарова Наталья Владимировна

Группа: БПИ223

Условие:

Разработать программу на языке Ассемблера процесса RISC-V, с использованием команд арифметического сопроцессора, выполняемые в симуляторе RARS. Разработанные программы должны принимать числа в допустимом диапазоне. Например, нужно учитывать области определения и допустимых значений, если это связано с условием задачи.

Разработать программу, вычисляющую с помощью степенного ряда с точностью не хуже 0,05% значение функции $1/e^x$ для заданного параметра x .

Метод решения задачи:

Для поиска значения функции использовался степенной ряд как и просилось в условии задачи. Ниже приведена формула с помощью которой вычисляется значение выражения

Разложение функции $\frac{1}{e^x}$ в ряд Маклорена в окрестности точки $x_0 = 0$

до 5 порядка включительно, имеет вид:

$$\frac{1}{e^x} = 1 - x + \frac{x^2}{2} - \frac{x^3}{6} + \frac{x^4}{24} - \frac{x^5}{120} + o(x^6)$$

Для решения данной задачи я использовала разложение в ряд маклорена.

Более подробное описание алгоритма

Для вычисления значения функции $1/e^x$ с заданной точностью с помощью степенного ряда можно использовать следующий метод:

1. Инициализация переменных: x (заданный параметр), n (начальное значение степени), sum (начальное значение суммы).
2. Вычислить значение члена степенного ряда: $term = x^n / n!$.
3. Добавить значение члена ряда к сумме: $sum += term$ (учитывая знак четности)

4. Увеличить значение степени на единицу: $n++$;
5. Повторение 3 и 4 до тех пор, пока значение члена ряда не станет меньше заданной точности;
6. Полученная сумма будет приближенным значением функции $1/e^x$ с заданной точностью.

Ссылки на источник информации с описанием метода решения задачи:

Теоретическая часть:

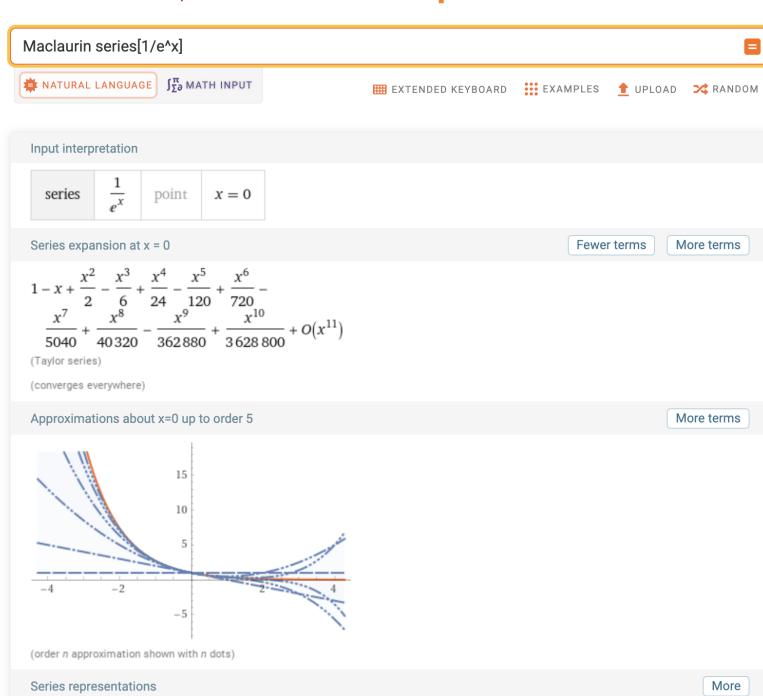
<http://mathdep.ifmo.ru/wp-content/uploads/2020/12/%D0%A0%D0%B0%D0%B7%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5-%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%BD%D1%8B%D1%85-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BD%D0%BC%D0%BD%D0%BA%D0%BB%D0%BE%D1%80%D0%B5%D0%BD%D0%B0.pdf>

Разложение в ряд маклорена с использованием сайта:

<https://mathforyou.net/online/calculus/series/maclaurin/>

Разложение и просмотр графика

<https://www.wolframalpha.com/input/?i=Maclaurin+series%5B1%2Fe%5Ex%5D>



Соответствие программы критериям и проведенные изменения

На 4–5 баллов

- Приведено решение задачи на ассемблере. Ввод данных осуществляется с клавиатуры. Вывод данных осуществляется на дисплей.
- В программе присутствуют комментарии, поясняющие выполняемые действия.
- В отчете представлено полное тестовое покрытие. Приведены результаты тестовых прогонов.

На 5-6 баллов

- В программе используются необходимые подпрограммы с передачей аргументов через параметры, что обеспечивает их повторное использование с различными входными аргументами. При нехватке регистров, используемых для передачи параметров, оставшиеся параметры передаются через стек.
- Внутри подпрограмм используются необходимые локальные переменные. При нехватке временных регистров обеспечивается сохранение данных на стеке в соответствии с соглашениями, принятыми для процессора.
- В местах вызова функции добавлены комментарии, описывающие передачу фактических параметров и перенос возвращаемого результата. При этом отмечена, какая переменная или результат какого выражения соответствует тому или иному фактическому параметру.

На 8 баллов

- Разработанные подпрограммы поддерживают многократное использование с различными наборами исходных данных, включая возможность подключения различных исходных и результирующих массивов.
- Реализовано автоматизированное тестирование за счет создания дополнительной тестовой программы, осуществляющей прогон подпрограмм, осуществляющих вычисления для различных тестовых данных (вместо их ввода). Осуществлен прогон тестов обеспечивающих покрытие различных ситуаций.
- Для дополнительной проверки корректности вычислений осуществлены аналогичные тестовые прогоны с использованием существующих библиотек и одного из языков программирования высокого уровня по выбору: C, C++, Python. Мною был выбран C++.

На 9 баллов

- Добавлено в программу использование макросов для реализации ввода и вывода данных. Макросы поддерживают повторное использование с различными массивами и другими параметрами.

На 10 баллов

- Программа разбита на несколько единиц компиляции. При этом подпрограммы ввода–вывода составляют унифицированные модули, используемые повторно как в программе, осуществляющей ввод исходных данных, так и в программе, осуществляющей тестовое покрытие.
- Макросы выделены в отдельную автономную библиотеку

Отчет расширен и дополнен новыми данными.

Скрины тестового покрытия

Описание тестовых прогонов с представлением информации о результатах тестирования.

Для программы с консольным вводом

тест с отрицательным числом



```
Input X: -3.0
Result (1/e^x): 20.08553692318766
-- program is finished running (0) --
```

$$x=0; e^0 = 1 \Rightarrow 1/1 = 1$$



```
-- program is finished running (0) --
Input X: 0
Result (1/e^x): 1.0
-- program is finished running (0) --
```

тест для целого положительного числа



```
-- program is finished running (0) --
Input X: 2.0
Result (1/e^x): 0.13533528323661276
-- program is finished running (0) --
```

тест для дробного числа



```
-- program is finished running (0) --
Input X: 1.6
Result (1/e^x): 0.20189651799465544
-- program is finished running (0) --
```

тест с отрицательным дробным числом



```
-- program is finished running (0) --
Input X: -3.1
Result (1/e^x): 22.197951281441632
-- program is finished running (0) --
```

Скрин работы дополнительной тестовой программы, осуществляющей прогон подпрограмм, осуществляющих вычисления для различных тестовых данных (вместо их ввода). Осуществлен прогон тестов обеспечивающих покрытие различных ситуаций (нулевое значение x, положительные и отрицательные целые значения и дробные)

```
Test1
x = 0
Result: 1.0
Expected result: 1.0
Test 2
x = 3.0
Result: 0.049787068367863986
Expected result: 0.0497871
Test 3
x = -2.0
Result: 7.389056098930649
Expected result: 7.38906
Test 4
x = 5.0
Result: 0.00673794699986907
Expected result: 0.006737946999
Test 5
x = 1.2
Result: 0.3011942119122021
Expected result: 0.301207
Test 6
x = -2.1
Result: 8.166169912567652
Expected result: 8.16606
Test 7
x = -3.8
Result: 44.70118449330082
Expected result: 44.7011
-- program is finished running (0) --
```

Для дополнительной проверки корректности вычислений осуществлены аналогичные тестовые прогоны с использованием существующих библиотек и одного из языков программирования высокого уровня по выбору: C++ (подобная реализация алгоритма)

```
#include <iostream>
#include <cmath>

int main() {
    double x, accuracy, result = 1.0, term = 1.0;

    // Введите значение x и желаемую точность
    std::cout << "Введите значение x: ";
    std::cin >> x;

    std::cout << "Введите точность (например, 0.0005): ";
    std::cin >> accuracy;

    // Используем степенной ряд для вычисления 1/e^x
    int n = 1;
    do {
        term *= (-x) / n;
        result += term;
        n++;
    } while (std::abs(1.0 / term) > accuracy);

    // Выводим результат
    std::cout << "Значение 1/e^x с точностью " << accuracy << " для x = " << x << " равно: " << result << std::endl;
}

main
yangex_trenirovki
/Users/natalzaharova/ClionProjects/yangex_trenirovki/cmake-build-debug/yangex_trenirovki
Введите значение x: 1.2
Введите точность (например, 0.0005): 0.0005
Значение 1/e^x с точностью 0.0005 для x = 1.2 равно: 0.301207

Process finished with exit code 0
```

Скрины для проверки работы тестера

```
Run: yangex_trenirovki
/Users/natalazaharova/CLionProjects/yangex_trenirovki/cmake-build-debug/yangex_trenirovki
Введите значение x: 0
Введите точность (например, 0.0005): 0.0005
Значение 1/e^x с точностью 0.0005 для x = 0 равно: 1

Process finished with exit code 0
```

```
Run: yangex_trenirovki
/Users/natalazaharova/CLionProjects/yangex_trenirovki/cmake-build-debug/yangex_trenirovki
Введите значение x: 3.0
Введите точность (например, 0.0005): 0.0005
Значение 1/e^x с точностью 0.0005 для x = 3 равно: 0.0497414

Process finished with exit code 0
```

```
Run: yangex_trenirovki
/Users/natalazaharova/CLionProjects/yangex_trenirovki/cmake-build-debug/yangex_trenirovki
Введите значение x: -2.0
Введите точность (например, 0.0005): 0.0005
Значение 1/e^x с точностью 0.0005 для x = -2 равно: 7.38899

Process finished with exit code 0
```

```
Run: yangex_trenirovki
/Users/natalazaharova/CLionProjects/yangex_trenirovki/cmake-build-debug/yangex_trenirovki
Введите значение x: 5.0
Введите точность (например, 0.0005): Значение 1/e^x с точностью 0 для x = 5 равно: 0.00673795

Process finished with exit code 0
```

```
Run: yangex_trenirovki
/Users/natalazaharova/CLionProjects/yangex_trenirovki/cmake-build-debug/yangex_trenirovki
Введите значение x: 1.2
Введите точность (например, 0.0005): 0.0005
Значение 1/e^x с точностью 0.0005 для x = 1.2 равно: 0.301207

Process finished with exit code 0
```

Значения программ сходятся (отражено в самой программе тестера) => алгоритм верный. **Проверка корректности пройдена.**

Скрины кода программы

Стоит отметить, что программа разбита на несколько единиц компиляции. При этом подпрограммы ввода–вывода составляют унифицированные модули, используемые повторно как в программе, осуществляющей ввод исходных данных, так и в программе, осуществляющей тестовое покрытие.

macrolib.s

```
1 # Example library of macros.
2 #
3 #
4 # Printing the value in specified register as double
5 .macro print_double(%x)
6     fmovd %x, %a0
7     fmovd %a0, %x
8     movl %x, %a0
9     movl %a0, %x
10    .end_macro
11
12
13 # Inputting integer in specified register from console except a0 register
14 .macro read_int(%x)
15     pushl %a0
16     li %a7, 5
17     calll _scanf
18     movl %x, %a0
19     popl %a0
20     .end_macro
21
22 # Instantiating double in specified register from console except a0 register
23 .macro read_double(%x)
24     pushl %a7, 7
25     calll _scanf
26     .end_macro
27
28 # Output a Line to the console.
29 .macro print_str(%x)
30
31     strl %x, %a0
32     movl %a0, %x
33     .text
34     pushl %a0
35     li %a7, 4
36     la %a0, str
37     calll _printf
38
39 .end_macro
40
41 # Print char to the console
42 .macro print_char(%x)
43     li %a7, 11
44     la %a0, %x
45     calll _printf
46     .end_macro
47
48 # Go to new line.
49 .macro newline
50     print_char("\n")
51     .end_macro
52
53 # Exit program
54 .macro exit
55     li %a7, 18
56     calll _exit
57     .end_macro
58
59 # Saving specified register on stack
60 .macro push(%x)
61     subl %sp, %sp, -4
62     sw %x, (%sp)
63     .end_macro
64
65 # Pushing a value from top of stack into a register
66 .macro pop(%x)
67     lw %x, (%sp)
68     addl %sp, %sp, 4
69     .end_macro
70
```

```
50     .end_macro
51
52     .macro _scanf(%x)
53     pushl %a0
54     movl %a0, %x
55     .text
56     pushl %a0
57     li %a7, 10
58     calll _scanf
59     .end_macro
60
61     .macro _printf(%x)
62     pushl %a0
63     movl %a0, %x
64     .text
65     pushl %a0
66     li %a7, 19
67     calll _printf
68     .end_macro
69
70     .end_macro
71
```

macros.s

```
1 #
2 # Library of macros.
3 #
4 section:macrolib-2
5
6 .macro calculate(%x)
7     movl %a0, %a1
8     addl %a1, %a0
9     movl %a0, %a1
10    .end_macro
11
12    # Macro for working with a function
13    # Save value of %x to stack in order not to lose it in a x
14    # Call function
15    # Load result
16    # End macro
17
18 .end_macro
19
```

subprogram.s

main для программы с консольным вводом

The screenshot shows the RARS 1.6 IDE interface. The main window displays assembly code for a program named 'main.s'. The code includes instructions for reading input from the console, calculating the result of the expression $1/e^x$, and printing the result. The assembly code is as follows:

```
1 .include "macros.s"
2
3 .global main
4 .text
5
6 # Main entry point to the program.
7 main:
8     .int3f    #<input X in double type: "
9     read_double(%t1)          # Reading the number from the console in ft1
10    calculate(%t0)           # Calculating the value of an expression
11    print_str("Result (1/e^x): ")
12    print_double(%t0)         # Printing hints to display the result
13    print_double(%t0)         # Print result in fo0
14    exit:                   # Completion of the program
15        li $v7 $0
16        ecall
17
18
```

The bottom panel shows the 'Messages' tab, which is currently empty.

main для программы с тестером (весь код приведен в папке)

The screenshot shows the RARS 1.6 IDE interface. The main window displays assembly code for a program named 'main.s'. The code includes a section for data definitions and multiple test cases (first_test through seventh_test) involving floating-point arithmetic and string output. The assembly code is as follows:

```
1 .include "macros.s"
2
3 .global main
4 .text
5 .data
6 first_test: .double 0          # x value for first test
7 second_test: .double 3.0       # x value for second test
8 third_test: .double -2.0      # x value for third test
9 fourth_test: .double 0.0       # x value for fourth test
10 fifth_test: .double 1.2       # x value for fifth test
11 sixth_test: .double -2.1      # x value for sixth test
12 seventh_test: .double -3.8     # x value for seventh test
13
14 # Main entry point to the program.
15 main:
16     print_str("Test1")
17     newline
18     fld %t0 first_test %t0
19     print_str("x = 0")
20     newline
21     calculate(%t0)
22     print_str("Result: ")
23     print_double(%t0)
24     print_str("Expected result: 1.0")
25
26     newline
27     print_str("Test 2")
28     newline
29     fld %t0 second_test %t0
30     print_str("x = 3.0")
31     newline
32     calculate(%t0)
33     print_str("Result: ")
34     print_double(%t0)
35     print_str("Expected result: 0.0497871")
36
37     newline
```

The bottom panel shows the 'Messages' tab, which is currently empty.