

Отчет

Индивидуальное задание №2. Арифметический сопроцессор

Вариант № 6

Выполнила: Захарова Наталья Владимировна

Группа: БПИ223

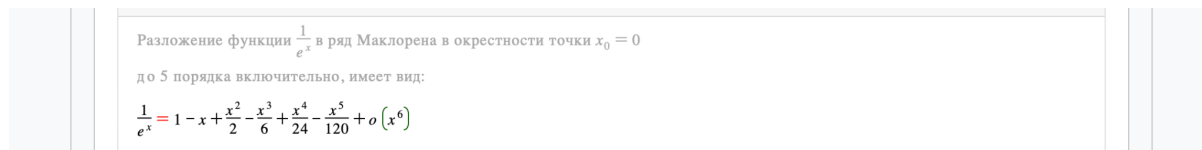
Условие:

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:

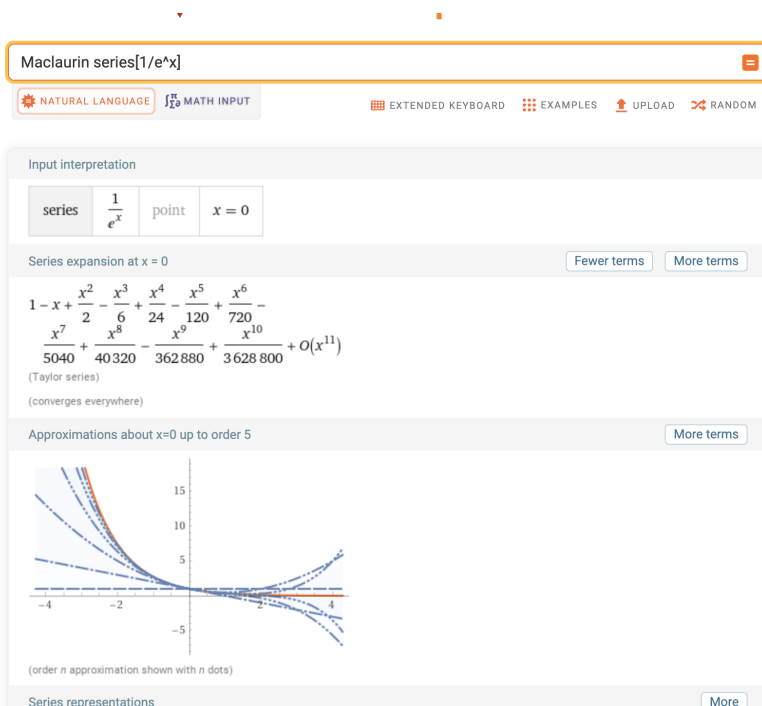
<http://mathdep.ifmo.ru/wp-content/uploads/2020/12/%D0%A0%D0%B0%D0%B7%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5-%D1%8D%D0%B%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%BD%D1%8B%D1%85-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B9-%D0%B2-%D1%80%D1%8F%D0%B4-%D0%9C%D0%B0%D0%BA%D0%BB%D0%BE%D1%80%D0%B5%D0%BD%D0%B0.pdf>

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



The screenshot shows a test runner window with a 'Messages' tab and a 'Run I/O' button. On the left is a 'Clear' button. The main area displays the following text:

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

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



The screenshot shows a test runner window with a 'Messages' tab and a 'Run I/O' button. On the left is a 'Clear' button. The main area displays the following text:

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

тест для целого положительного числа



The screenshot shows a test runner window with a 'Messages' tab and a 'Run I/O' button. On the left is a 'Clear' button. The main area displays the following text:

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

тест для дробного числа



The screenshot shows a test runner window with a 'Messages' tab and a 'Run I/O' button. On the left is a 'Clear' button. The main area displays the following text:

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

тест с отрицательным дробным числом



The screenshot shows a test runner window with a 'Messages' tab and a 'Run I/O' button. On the left is a 'Clear' button. The main area displays the following text:

```
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)

```

Messages    Run I/O
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: 0.006737946999086907
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) --
Clear

```

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)

```

CMakeLists.txt  main.cpp
#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(std::exp(-x) - result) > accuracy);

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

    return 0;
}

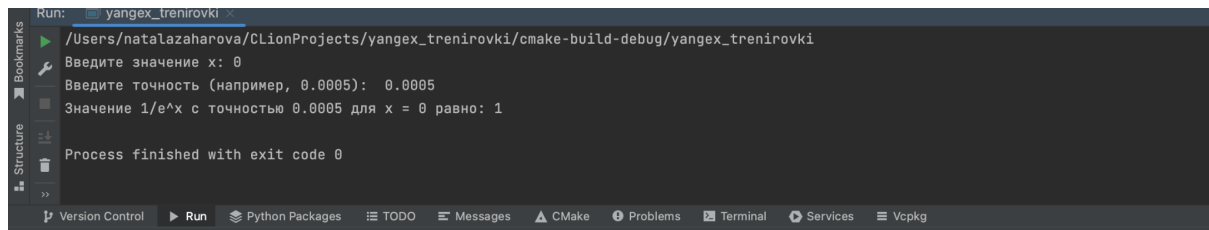
```

```

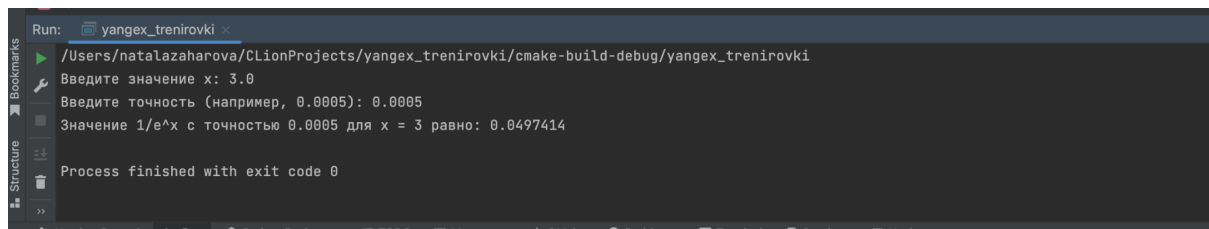
Run: yangex_trenirovki
/Users/natalazaharova/CLionProjects/yangex_trenirovki/cmake-build-debug/yangex_trenirovki
Введите значение x: 1.2
Введите точность (например, 0.0005): 0.0005
Значение 1/e^x с точностью 0.0005 для x = 1.2 равно: 0.301207
Process finished with exit code 0

```

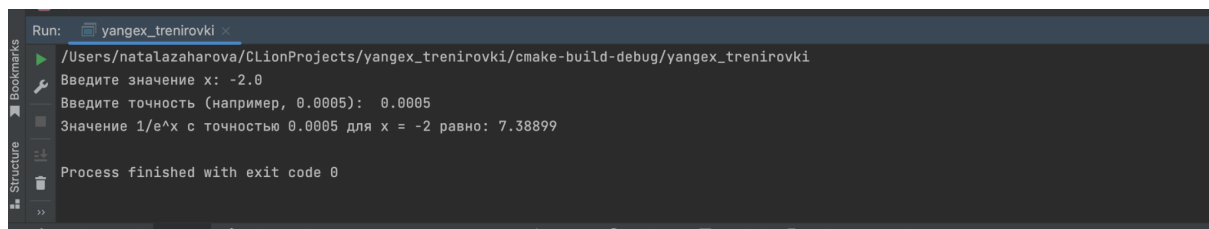
Скрины для проверки работы тестера



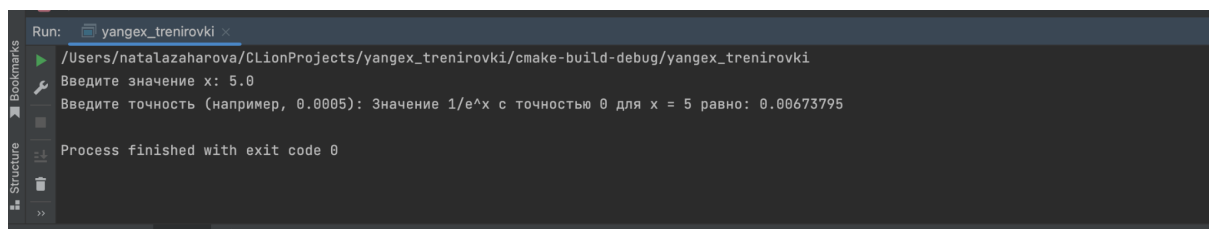
```
Run: yangex_trenirovki x
/Users/natalazaharova/CLionProjects/yangex_trenirovki/cmake-build-debug/yangex_trenirovki
Введите значение x: 0
Введите точность (например, 0.0005): 0.0005
Значение 1/e^x с точностью 0.0005 для x = 0 равно: 1
Process finished with exit code 0
```



```
Run: yangex_trenirovki x
/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 x
/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 x
/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
```



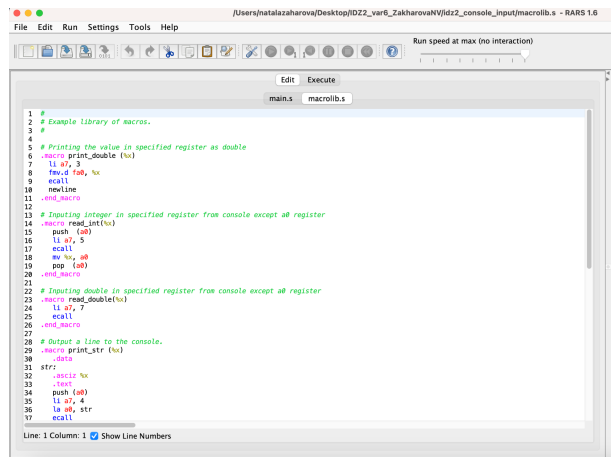
```
Run: yangex_trenirovki x
/Users/natalazaharova/CLionProjects/yangex_trenirovki/cmake-build-debug/yangex_trenirovki
Введите значение x: 1.2
Введите точность (например, 0.0005): 0.0005
Значение 1/e^x с точностью 0.0005 для x = 1.2 равно: 0.301207
Process finished with exit code 0
```

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

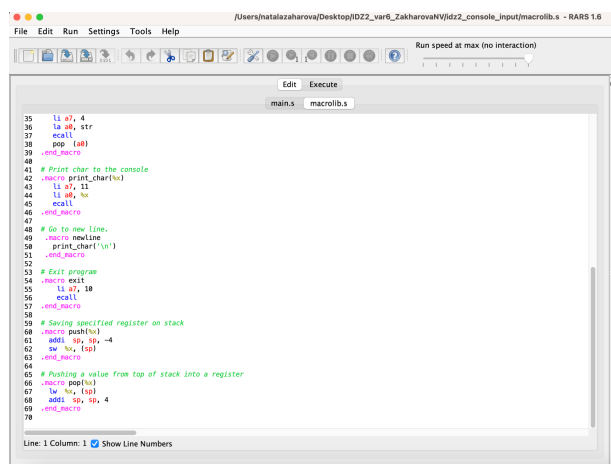
Скрины кода программы

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

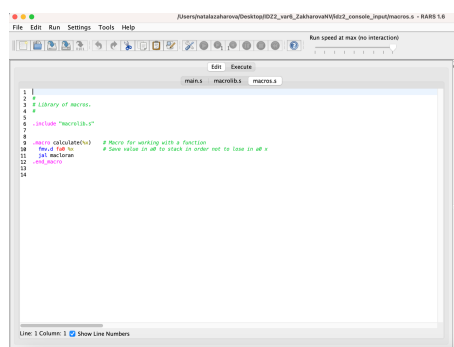


```
1 #
2 # Example library of macros.
3 #
4
5 # Printing the value in specified register as double
6 .macro print_double(%r)
7     li %r, 3
8     fmv.d %r0, %r
9     ecall
10    newline
11 .end_macro
12
13 # Inputting integer in specified register from console except a0 register
14 .macro read_int(%r)
15     push(%a0)
16     li %r, 5
17     ecall
18     mv %r, %a0
19     pop(%a0)
20 .end_macro
21
22 # Inputting double in specified register from console except a0 register
23 .macro read_double(%r)
24     li %r, 7
25     ecall
26 .end_macro
27
28 # Output a line to the console.
29 .macro print_str(%r)
30     .data
31     str:
32     .ascii %r
33     .len
34     push(%a0)
35     li %r, 4
36     la %r, str
37     ecall
38 .end_macro
```



```
35     li %r, 4
36     la %r, str
37     ecall
38     pop(%a0)
39 .end_macro
40
41 # Print char to the console
42 .macro print_char(%r)
43     li %r, 11
44     li %r, %r
45     ecall
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 %r, 10
56     ecall
57 .end_macro
58
59 # Saving specified register on stack
60 .macro push(%r)
61     addi %r, %r, -4
62     sw %r, [%r]
63 .end_macro
64
65 # Pushing a value from top of stack into a register
66 .macro pop(%r)
67     lw %r, [%r]
68     addi %r, %r, 4
69 .end_macro
70
```

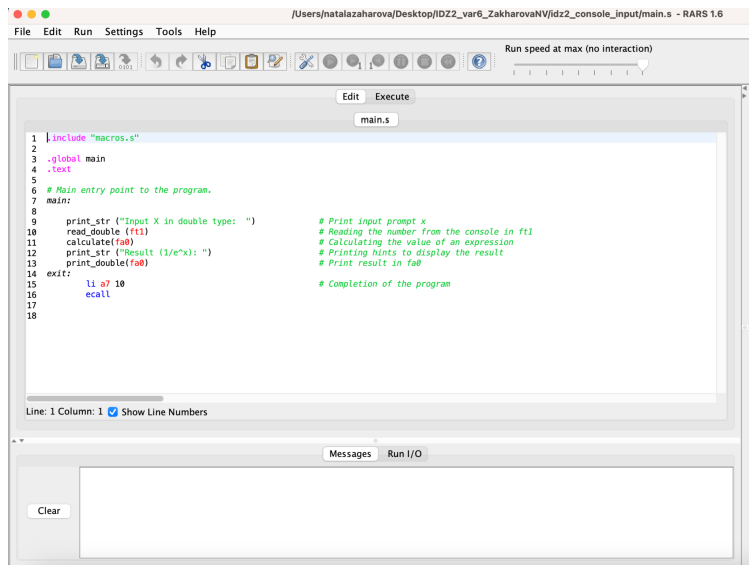
macros.s



```
1 #
2 # Library of macros.
3 #
4
5 # include "macrolib.s"
6
7 # Macro for working with a function
8 .macro calculate(%r)
9     # Save value of %r to stack in order not to lose it in all
10    # subroutines
11    .data
12    str:
13    .ascii %r
14    .len
15    push(%a0)
16    li %r, 5
17    ecall
18    mv %r, %a0
19    pop(%a0)
20 .end_macro
```

subprogram.s

main для программы с консольным вводом

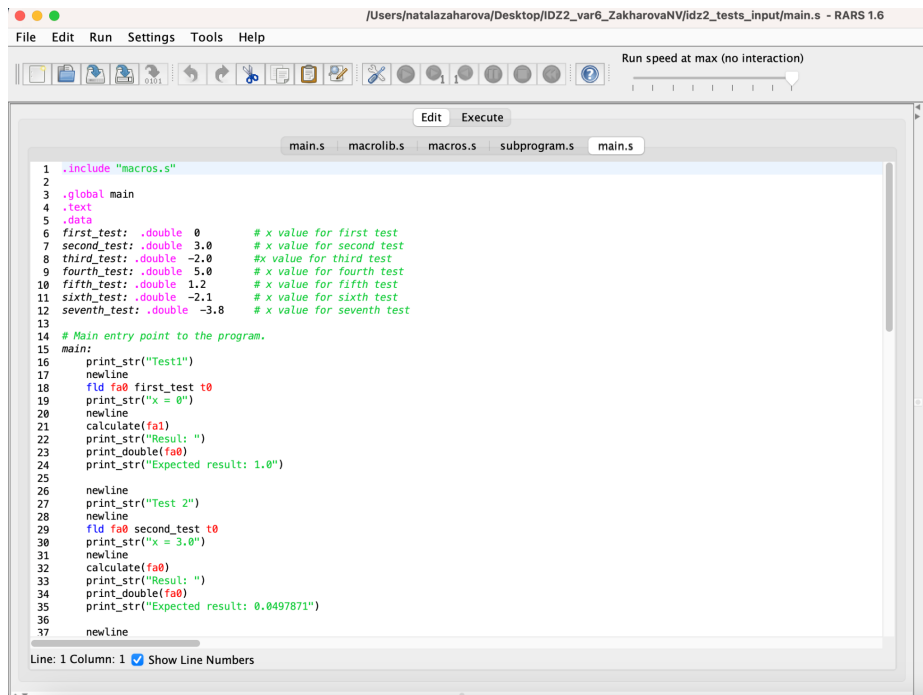


The screenshot shows the RARS 1.6 IDE with the file `main.s` open. The code is as follows:

```
1 |.include "macros.s"
2
3 |.global main
4 |.text
5
6 |# Main entry point to the program.
7 |main:
8
9 |    print_str("Input X in double type: ")      # Print input prompt x
10 |    read_double(fa0)                          # Reading the number from the console in fa0
11 |    calculate(fa0)                             # Calculating the value of an expression
12 |    print_str("Result (1/e^x): ")              # Printing hints to display the result
13 |    print_double(fa0)                         # Print result in fa0
14 |    exit:                                       # Completion of the program
15 |        li a7 10
16 |        ecall
17
18
```

The status bar at the bottom indicates "Line: 1 Column: 1" and "Show Line Numbers" is checked.

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



The screenshot shows the RARS 1.6 IDE with the file `main.s` open. The 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 5.0   # 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 fa0 first_test t0
19 |    print_str("x = 0")
20 |    newline
21 |    calculate(fa0)
22 |    print_str("Resul: ")
23 |    print_double(fa0)
24 |    print_str("Expected result: 1.0")
25 |
26 |    newline
27 |    print_str("Test 2")
28 |    newline
29 |    fld fa0 second_test t0
30 |    print_str("x = 3.0")
31 |    newline
32 |    calculate(fa0)
33 |    print_str("Resul: ")
34 |    print_double(fa0)
35 |    print_str("Expected result: 0.0497871")
36 |
37 |    newline
```

The status bar at the bottom indicates "Line: 1 Column: 1" and "Show Line Numbers" is checked.