**Вариант 17**

Разработать программу, вычисляющую **с помощью степенного ряда** с точностью не хуже 0,1% значение функции $ln(1-x)$ для входного параметра $x$.

**Описание метода решения задачи:**

Для вычисления значения функции $ln(1-x)$ в программе используется разложение функции в степенной ряд Тейлора в окрестности точки x = 0. Это позволяет аппроксимировать значение логарифма для значений x в диапазоне $-1 < x < 1$.

Функция $\ln(1-x)$ может быть разложена в бесконечный ряд:

$$\ln(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \frac{x^4}{4} - \frac{x^5}{5} - \cdots = -\sum_{n=1}^{\infty} \frac{x^n}{n}$$

Это разложение сходится для всех $x$ таких, что $|x| < 1$, что соответствует области определения нашей функции.

Начальное значение суммы $Sum = -x$.
Начальное значение текущего члена ряда $term = -x$.
Начальное значение степени $n = 1$.

Цикл вычисления ряда:
Увеличить степень $n$ на 1. Обновить текущий член ряда по формуле:

$$term = term \times (-x) \times \frac{1}{n}$$

Добавить текущий член ряда к сумме:

$$Sum = Sum + term$$

Проверить условие сходимости ряда:

Если абсолютное значение текущего члена $|term|$ меньше заданной точности $small\_value$, цикл завершается. (От $small\_value$ зависит точность числа и кол-во итераций. Чем меньше $small\_value$, тем точнее будет чсло и тем больше будет итераций!)

# Отчёт программы на 4–5 баллов:

Важно! Степенной ряд для функции $ln(1-x)$ сходится, если $\mathbf{|x| < 1}$. Таким образом, значение x может быть в интервале: $-1 < x < 1$.

- Решение задачи написано на ассемблере. Данные вводятся с клавиатуры, а вывод отображается на экране в файле main.asm. (Так как задание было сразу написано на оценку 10, есть main.asm и есть test_main.asm, если надо вводить данные с клавиатуры и получить вывод на экран, открываем файл main.asm, если же надо прогнать тесты и получить результаты на экран, то нужен файл test_main.asm)

- Код сопровождается комментариями, которые объясняют каждое выполняемое действие.

- В отчёте представлено полное тестовое покрытие, включающее результаты выполнения тестов, подтверждённые скриншотами, показывающими процесс работы программы.

**ТЕСТОВОЕ ПОКРЫТИЕ:**

**1. X = 1**

```
Функция ln(1 - x). Введите значение x (|x| < 1): 1
Ошибка! Недопустимое значение x. Введите x из интервала (-1, 1).
Функция ln(1 - x). Введите значение x (|x| < 1):
```

**Важно!** При вводе недопустимого значения $x$ программа повторно запрашивает ввод $x$.

**2. X = -1**

```
Функция ln(1 - x). Введите значение x (|x| < 1): -1
Ошибка! Недопустимое значение x. Введите x из интервала (-1, 1).
Функция ln(1 - x). Введите значение x (|x| < 1):
```

**3. X > 1**

```
Функция ln(1 - x). Введите значение x (|x| < 1): 100
Ошибка! Недопустимое значение x. Введите x из интервала (-1, 1).
Функция ln(1 - x). Введите значение x (|x| < 1):
```

**4. X < -1**

```
Функция ln(1 - x). Введите значение x (|x| < 1): -100
Ошибка! Недопустимое значение x. Введите x из интервала (-1, 1).
Функция ln(1 - x). Введите значение x (|x| < 1):
```

### 5. X = 0

```
Функция ln(1 - x). Введите значение x (|x| < 1): 0
ln(1 - x) = 0.0

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

### 6. X = 0.5

```
Функция ln(1 - x). Введите значение x (|x| < 1): 0.5
ln(1 - x) = -0.6931471805599005

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

### 7. X = -0.5

```
Функция ln(1 - x). Введите значение x (|x| < 1): -0.5
ln(1 - x) = 0.4054651081081568

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

### 8. X = -0.9

```
Функция ln(1 - x). Введите значение x (|x| < 1): -0.9
ln(1 - x) = 0.6418538861724222

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

### 9. X = 0.9

```
Функция ln(1 - x). Введите значение x (|x| < 1): 0.9
ln(1 - x) = -2.3025850929921035

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

### 10. X = -0.1

```
Функция ln(1 - x). Введите значение x (|x| < 1): -0.1
ln(1 - x) = 0.09531017980432552

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

### 11. X = 0.1

```
Функция ln(1 - x). Введите значение x (|x| < 1): 0.1
ln(1 - x) = -0.10536051565782553

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

# Отчёт программы на 6-7 баллов:

- В программе используется подпрограмма compute_ln, принимающая аргумент через регистр с плавающей точкой fa0 и возвращающая значение через регистр fa0, а также статус выполнения через регистр a0. В начале подпрограммы адрес возврата ra сохраняется в стек, чтобы в конце работы его можно было оттуда восстановить. Возможно повторно использовать подпрограммы с различными входными аргументами, включая применение в других программах.
- Для хранения локальных переменных в подпрограмме применяются временные регистры с плавающей точкой ft0 - ft11 и временные регистры общего назначения t0 - t6. Подпрограмма полностью отделена от вызывающего её кода. При необходимости сохранения данных используются соглашения вызова: регистры ra и s0 сохраняются на стеке в начале подпрограммы и восстанавливаются перед возвратом.
- В местах вызова функций добавлены подробные комментарии. В них подробно описывается, как передаются фактические параметры и как возвращаемые результаты передаются дальше. Также указывается, какая переменная или результат какого выражения относится к конкретному параметру функции.

# Отчёт программы на 8 баллов:

- Разработанная подпрограмма compute_ln поддерживает многократное использование с различными наборами исходных данных. Она принимает на вход значение x через регистр с плавающей точкой fa0 и возвращает результат ln(1 - x) в том же регистре, а также статус выполнения в регистре a0. Это позволяет вызывать подпрограмму с различными входными данными, включая возможность обработки различных исходных данных в других программах.
- Автоматизированное тестирование реализовано с помощью дополнительной тестовой программы в файле test_main.asm, которая прого-

няет подпрограмму compute_ln с различными тестовыми данными (вместо их ручного ввода). Для этого создан макрос test в файле macrolib.asm, который позволяет задавать различные значения x и соответствующие ожидаемые результаты.

- Для дополнительной проверки корректности вычислений осуществлены аналогичные тестовые прогоны с использованием существующих библиотек и Python. (см. скриншоты ниже)

**Код программы на Python:**

main.py

```python
import math
1 usage
def main():
    test_values = [
        0.0,
        -0.5,
        0.5,
        -0.9,
        0.9,
        0.1,
        -0.1,
        0.999,
        -0.999,
        0.001,
        -0.001,
    ]

    print("Вычисление ln(1 - x) в Python:\n")
    for x in test_values:
        try:
            result = math.log(1 - x)
            print(f"x = {x}")
            print(f"ln(1 - x) = {result}\n")
        except ValueError as e:
            print(f"x = {x}")
            print(f"Ошибка: {e}\n")


if __name__ == "__main__":
    main()
```

**Результаты тестирования на Python:**

```
Вычисление ln(1 - x) в Python:

x = 0.0
ln(1 - x) = 0.0

x = -0.5
ln(1 - x) = 0.4054651081081644

x = 0.5
ln(1 - x) = -0.6931471805599453

x = -0.9
ln(1 - x) = 0.6418538861723947

x = 0.9
ln(1 - x) = -2.302585092994046

x = 0.1
ln(1 - x) = -0.10536051565782628

x = -0.1
ln(1 - x) = 0.09531017980432493

x = 0.999
ln(1 - x) = -6.907755278982136

x = -0.999
ln(1 - x) = 0.6926470555182631

x = 0.001
ln(1 - x) = -0.0010005003335835344

x = -0.001
ln(1 - x) = 0.0009995003330834232
```

**Результаты тестирования в RARS:**

```
Тест 1
Ожидаемый результат:  0.0
Полученный результат: 0.0
Тест пройден.

Тест 2
Ожидаемый результат:  0.4054651081081644
Полученный результат: 0.4054651081081568
Тест пройден.

Тест 3
Ожидаемый результат:  -0.6931471805599453
Полученный результат: -0.6931471805599005
Тест пройден.

Тест 4
Ожидаемый результат:  0.6418538861723947
Полученный результат: 0.6418538861724222
Тест пройден.
```

```
Тест 5
Ожидаемый результат:  -2.302585092994046
Полученный результат: -2.3025850929921035
Тест пройден.

Тест 6
Ожидаемый результат:  -0.10536051565782628
Полученный результат: -0.10536051565782553
Тест пройден.

Тест 7
Ожидаемый результат:  0.09531017980432493
Полученный результат: 0.09531017980432552
Тест пройден.

Тест 8
Ожидаемый результат:  -6.907755278982136
Полученный результат: -6.907755278326664
Тест пройден.
```

```
Тест 9
Ожидаемый результат:  0.6926470555182631
Полученный результат: 0.6926470555183086
Тест пройден.

Тест 10
Ожидаемый результат:  -0.0010005003335835344
Полученный результат: -0.0010005003335835337
Тест пройден.

Тест 11
Ожидаемый результат:  9.995003330834232E-4
Полученный результат: 9.99500333083533E-4
Тест пройден.
```

```
Тест 12
Ошибка! Недопустимое значение x. Введите x из интервала (-1, 1).
Ожидаемый результат:  0.0
Полученный результат: 0.0
Тест пройден.

Тест 13
Ошибка! Недопустимое значение x. Введите x из интервала (-1, 1).
Ожидаемый результат:  0.0
Полученный результат: 0.0
Тест пройден.


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

**Важно!** По условию задачи допусима погрешность не более 0.1%. Как видно все тесты справляются с этим. Однако если нужны еще БОЛЕЕ ТОЧНЫЕ ответы, необходимо в файле compute_ln.asm изменить значение *small_value* на еще меньшее, но тогда увеличится количество итераций!

**Важно!** Также в конце видим что 12-й и 13-й тест выдают ошибку *"Ошибка! Недопустимое значение x. Введите x из интервала (-1, 1)."* Это происходит из-за того что в тестах $x$ принимает значение 1 и -1, которые не входят в допустимый диапазон. Соответсвенно результат не может быть посчитан, но в таком случае программа присваивает значение 0.0. Поэтому можно сделать тесты, в которых проверяется, что не может быть посчитан результат для $x >= 1$ и $x <= -1$.

# Отчёт программы на 9 баллов:

- В программе созданы макросы, позволяющие облегчить ввод и вывод данных, а также проведение тестирования. Макросы находятся в отдельном файле macrolib.asm и могут использоваться повторно с различными параметрами.

# Отчёт программы на 10 баллов:

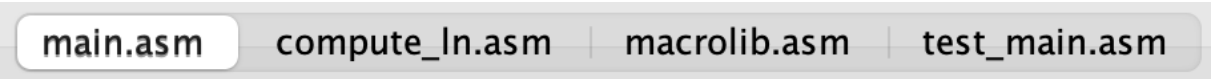
- Программа разделена на несколько единиц компиляции. Основные компоненты программы находятся в отдельных файлах: main.asm — основная программа, осуществляющая ввод исходных данных и вызов подпрограммы вычисления. compute_ln.asm — содержит подпрограмму compute_ln, выполняющую вычисление $\ln(1-x)$. macrolib.asm — содержит все макросы, выделенные в отдельную автономную библиотеку. test_main.asm — тестовая программа, осуществляющая автоматизированное тестирование подпрограммы compute_ln.

  Подпрограммы ввода и вывода данных представлены в виде унифицированных модулей. Макросы для ввода и вывода (print_string, read_double, print_double и другие) определены в файле macrolib.asm и используются как в основной программе main.asm, так и в тестовой программе test_main.asm.

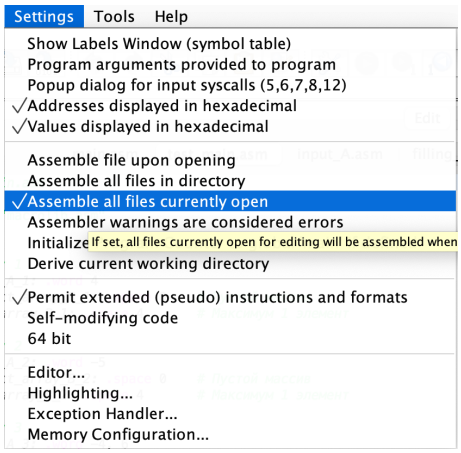
- Макросы выделены в отдельную автономную библиотеку macrolib.asm

# Обзор программы

Для начала надо открыть все файлы:




Следом выбрать: Settings -> Assemble all files currently open




Далее выбираем: хотим ли мы ввод с клавиатуры или прогнать тесты.

Если с вводом с клавиатуры, то выбираем файл main.asm, ассемблируем и нажимаем Run.

Если хотим прогнать тесты, то выбираем файл test_main.asm, ассемблируем и нажимаем Run.

Примеры ввода с клавиатуры можно найти в пункте "Отчёт программы на 4–5 баллов"в разделе "ТЕСТОВОЕ ПОКРЫТИЕ".

Прогон тестов можно найти в пункте "Отчёт программы на 8 баллов"в разделе "Результаты тестирования в RARS".

**СПАСИБО ЗА ВНИМАНИЕ!**