ИДЗ-2_Хадзакос_НА_236

Вариант 1

Условие

Разработать программу, вычисляющую **с помощью степенного ряда** с точностью не хуже *0,05%* значение функции $\sqrt{1+x}$ для заданного параметра x.

Решение на 7 баллов

Файл: https://github.com/khadzakos/asm_course/blob/main/ihw2/ihw2.asm

```
.data
null: .float 0.0
one: .float 1.0
pow: .float -0.5
epsilon: .float 0.0005
.align 2
prompt: .asciz "Данная программа ищет корень 1 + x, используя степенной
ряд\n"
prompt_elem: .asciz "Введите элемент(неотрицательное число): "
result_msg: .asciz "Результат sqrt(1 + x) с точностью ошибки 0.0005%: "
warning_ltz: .asciz "Число должно быть неотрицательным!\n"
endl: .asciz "\n"

.text
.globl main
main:
    la a0, prompt
    li a7, 4
    ecall
    la t1, null
    flw f0, 0(t1)

enter_number: # Ввод числа до момента корректного ввода
    la a0, prompt_elem
    li a7, 4
    ecall
```

```
    li a7, 6
    ecall
    fmv.s fs0, fa0 # fs0 = x
    flt.s t1, fs0, f0
    beqz t1, end_enter_number

    la a0, warning_ltz # Вывод ошибки
    li a7, 4
    ecall
    j enter_number
end_enter_number:

    fmv.s fa0, fs0 # В функцию передается только один параметр x
    jal sqrt_x
    # Результат лежит в fa0, сразу выводим
    la a0, result_msg # Вывод сообщения
    li a7, 4
    ecall
    li a7, 2 # Вывод ответа
    ecall

    li a7, 10
    ecall

sqrt_x:
    addi sp, sp, -16
    fsw fs0, 8(sp)    # Сохраняем float регистры
    fsw fs1, 4(sp)
    fsw fs2, 0(sp)

    flw f0, one, t1
    flt.s t5, fa0, f0 # Проверка, что число меньше или больше либо равно
1, чтобы установить значение fs0 = x или fs0 = 1/x соответственно
    beqz t5, x_greater
x_less:
    fmv.s fs0, fa0 # fs0 = x
    j x_next
x_greater:
    # Вычисляем fs0 = 1/x
    flw f0, one, t1
    fdiv.s fs0, f0, fa0 # fs0 = 1/x
    j x_next
```

```
x_next:

    flw fs1, one, t1  # fs1 = 1 - назовем его term
    flw fs2, one, t1  # fs2 = 1 - назовем его result
    li t0, 1 # счетчик - назовем n

loop:
    # Вычисляем (2*n - 3) работает корректно до o(x^6)(этого достаточно
для нащей точности)
    add t1, t0, t0  # t1 = 2*n
    addi t1, t1, -3 # t1 = 2*n - 3
    fcvt.s.w f1, t1 # Конвертируем в float

    # Умножаем на -0.5
    flw f2, pow, t1
    fmul.s f1, f1, f2

    # Умножаем на (1/x), если x >= 1 или на x, если x < 1
    fmul.s f1, f1, fs0

    # Делим на n
    fcvt.s.w f2, t0  # Конвертируем n в float
    fdiv.s f1, f1, f2

    # Умножаем текущий term
    fmul.s fs1, fs1, f1 # Новый term

    # Добавляем к result
    fadd.s fs2, fs2, fs1



    # Проверяем условие выхода |term| > epsilon * result(проверяем,
текущий член изменяет сильнее, чем на 0.005%)
    fabs.s f1, fs1 # |term|
    flw f2, epsilon, t1 # загружаем epsilon
    fmul.s f3, f2, fs2 # epsilon * result
    fle.s t1, f3, f1 # if |term| >= epsilon * result

    # Увеличиваем n
    addi t0, t0, 1

    bnez t1, loop
```

```
    beqz t5, result_greater # Проверка, что число меньше или больше либо
равно 1(t5 мы не меняли, в нем лежит результат прошлой проверки flt.s
t5, fa0, f0)
result_less:
    fmv.s fa0, fs2
    j result_next
result_greater:
    # Умножаем на sqrt(x) - этого требует степенная форма, где x >= 1
    fsqrt.s f0, fa0
    fmul.s fa0, f0, fs2
    j result_next
result_next:
    # Эпилог
    j done

done:
    # Восстанавливаем сохраненные регистры
    flw fs0, 8(sp)
    flw fs1, 4(sp)
    flw fs2, 0(sp)
    addi sp, sp, 16
    ret
```

Концепция решения:

Так как необходимо найти $\sqrt{1+x}$ с помощью степенного ряда, то выпишем ряд Тейлора

$$\sqrt{1+x} = 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{x^3}{16} - \frac{5x^4}{128} + \frac{7x^4}{256} + o(x^6)$$

Мы воспользуемся нестандартной записью, которую предлагает Wolfram:

$$\sqrt{1+x} = \sum_{n=0}^{\infty} \binom{\frac{1}{2}}{n} x^n$$

Покажем, что оно действительно так:

n	1	2	3	4	5
$\binom{\frac{1}{2}}{n}$	$\frac{1}{2}$	$-\frac{1}{8}$	$\frac{1}{16}$	$-\frac{5}{128}$	$\frac{7}{256}$
approximation	0.5	−0.125	0.0625	−0.0390625	0.0273438

Мы будем пользоваться такой записью для вычислений.
Важно заметить, что данная формула работает при x в окрестностях 0. Чтобы решить задачу для x >= 1, необходимо найти приблизительный ответ для $\sqrt{1+\frac{1}{x}}$ и домножить на $\sqrt{x}$. Для мы вынуждены воспользоваться встроенной командой *fsqrt.s*.

Результаты тестирования

Тест 1

```
Данная программа ищет корень 1 + x, используя степенной ряд
Введите элемент(неотрицательное число): -4
Число должно быть неотрицательным!
Введите элемент(неотрицательное число): 3
Результат sqrt(1 + x) с точностью ошибки 0.0005%: 1.9998438
-- program is finished running (0) --
```

Тест 2

```
Данная программа ищет корень 1 + x, используя степенной ряд
Введите элемент(неотрицательное число): 0
Результат sqrt(1 + x) с точностью ошибки 0.0005%: 1.0
-- program is finished running (0) --
```

Тест 3

```
Данная программа ищет корень 1 + x, используя степенной ряд
Введите элемент(неотрицательное число): 1
Результат sqrt(1 + x) с точностью ошибки 0.0005%: 1.4145569
-- program is finished running (0) --
```

Тест 4

```
Данная программа ищет корень 1 + x, используя степенной ряд
Введите элемент(неотрицательное число): 100
Результат sqrt(1 + x) с точностью ошибки 0.0005%: 10.049875
-- program is finished running (0) --
```

Тест 5

```
Данная программа ищет корень 1 + x, используя степенной ряд
Введите элемент(неотрицательное число): 0.2
Результат sqrt(1 + x) с точностью ошибки 0.0005%: 1.0955
-- program is finished running (0) --
```

Тест 6

```
Данная программа ищет корень 1 + x, используя степенной ряд
Введите элемент(неотрицательное число): 15
Результат sqrt(1 + x) с точностью ошибки 0.0005%: 4.000003
-- program is finished running (0) --
```

Тест 7

```
Данная программа ищет корень 1 + x, используя степенной ряд
Введите элемент(неотрицательное число): 52
Результат sqrt(1 + x) с точностью ошибки 0.0005%: 7.2801065
-- program is finished running (0) --
```

Решение на 8 баллов

Файл: https://github.com/khadzakos/asm_course/blob/main/ihw2/ihw2_test.asm

```
.macro endl() # Макрос перевода строки
    .data
        endl: .asciz "\n"
    .text
        li a7, 4
        la a0, endl
        ecall
.end_macro

.macro print_str(%x) # Макрос вывода строки
    .data
       str: .asciz %x
   .text
       li a7, 4
       la a0, str
       ecall
.end_macro
```

```
.macro print_float(%x) # Макрос вывода float
    addi sp, sp, -4
    fsw fa0, 0(sp) # Сохраняем

    fmv.s fa0, %x
    li a7, 2
    ecall

    flw fa0, 0(sp) # Восстанавливаем
    addi sp, sp, 4
.end_macro

.macro print_int(%x) # Макрос вывода int
    mv a0, %x
    li a7, 1
    ecall
.end_macro



.data
null: .float 0.0
one: .float 1.0
pow: .float -0.5

tests: .float 0.2, 15.0, 52.0, 993.0, 81.0, 120.0 # массив X-ов
eps: .float 0.0005, 0.0005, 0.001, 0.001, 0.0001, 0.0002 # массив
точностей
n: .word 6

.text
.global test_main
test_main:
    li s0, 0 # счетчик
    lw s1, n # ограничитель
    la s2, tests # указатель на tests
    la s3, eps # указатель на eps
test_loop:
    # Вывод номера теста
    print_str("Тест ")
    addi s0, s0, 1
    print_int(s0)
    addi s0, s0,  -1
    endl()
```

```
    # Передача значений в функцию
    flw fa0, 0(s2) # tests[i]
    flw fa1, 0(s3) # (ИЗМЕНЕНИЕ, передача нового параметра) eps[i]

        # Вывод переданных значений
    print_str("X = ")
    print_float(fa0)
    endl()
    print_str("eps = ")
    print_float(fa1)
    endl()

    jal sqrt_x # Вызов

        # Вывод результата
    print_str("Результат: ")
    print_float(fa0)
    endl()

    addi s0, s0, 1
    addi s2, s2, 4
    addi s3, s3, 4
    blt s0, s1, test_loop

    li a7, 10
    ecall


sqrt_x:
    addi sp, sp, -16
    fsw fs0, 8(sp)    # Сохраняем float регистры
    fsw fs1, 4(sp)
    fsw fs2, 0(sp)

    flw f0, one, t1
    flt.s t5, fa0, f0 # Проверка, что число меньше или больше либо равно
1, чтобы установить значение fs0 = x или fs0 = 1/x соответственно
    beqz t5, x_greater
x_less:
    fmv.s fs0, fa0 # fs0 = x
    j x_next
x_greater:
```

```
    # Вычисляем fs0 = 1/x
    flw f0, one, t1
    fdiv.s fs0, f0, fa0 # fs0 = 1/x
    j x_next
x_next:

    flw fs1, one, t1  # fs1 = 1 - назовем его term
    flw fs2, one, t1  # fs2 = 1 - назовем его result
    li t0, 1 # счетчик - назовем n

loop:
    # Вычисляем (2*n - 3) работает корректно до o(x^6)(этого достаточно
для нащей точности)
    add t1, t0, t0  # t1 = 2*n
    addi t1, t1, -3 # t1 = 2*n - 3
    fcvt.s.w f1, t1 # Конвертируем в float

    # Умножаем на -0.5
    flw f2, pow, t1
    fmul.s f1, f1, f2

    # Умножаем на (1/x), если x >= 1 или на x, если x < 1
    fmul.s f1, f1, fs0

    # Делим на n
    fcvt.s.w f2, t0  # Конвертируем n в float
    fdiv.s f1, f1, f2

    # Умножаем текущий term
    fmul.s fs1, fs1, f1 # Новый term

    # Добавляем к result
    fadd.s fs2, fs2, fs1


    # Проверяем условие выхода |term| > epsilon * result(проверяем,
текущий член изменяет сильнее, чем на 0.005%)
    fabs.s f1, fs1 # |term|
    fmv.s f2, fa1 # загружаем epsilon(ИЗМЕНЕНИЕ, теперь он лежит в fa1)
    fmul.s f3, f2, fs2 # epsilon * result
    fle.s t1, f3, f1 # if |term| >= epsilon * result

    # Увеличиваем n
```

```
    addi t0, t0, 1

    bnez t1, loop

    beqz t5, result_greater # Проверка, что число меньше или больше либо
равно 1(t5 мы не меняли, в нем лежит результат прошлой проверки flt.s
t5, fa0, f0)
result_less:
    fmv.s fa0, fs2
    j result_next
result_greater:
    # Умножаем на sqrt(x) — этого требует степенная форма, где x >= 1
    fsqrt.s f0, fa0
    fmul.s fa0, f0, fs2
    j result_next
result_next:
    # Эпилог
    j done

done:
    # Восстанавливаем сохраненные регистры
    flw fs0, 8(sp)
    flw fs1, 4(sp)
    flw fs2, 0(sp)
    addi sp, sp, 16
    ret
```

Пояснение:

В данном решении функция sqrt_x перенесена из первого решения. Добавлена передача кастомной точности и оперирования с ней. Добавлены макросы на перевод строки, вывод строки, вывод числа с плавающей точкой и вывод целого числа.
Для тестирования заведен массив tests, в котором хранятся тестовые случаи. Тесты прогоняются, используя цикл, который кастомно выводит номер теста, тестовый случай и результат работы.

Результаты тестирования

Рассматриваются только корректные тесты

```
Тест 1
X = 0.2
eps = 5.0E-4
Результат: 1.0955
Тест 2
X = 15.0
eps = 5.0E-4
Результат: 4.000003
Тест 3
X = 52.0
eps = 0.001
Результат: 7.2801065
Тест 4
X = 993.0
eps = 0.001
Результат: 31.52777
Тест 5
X = 81.0
eps = 1.0E-4
Результат: 9.055385
Тест 6
X = 120.0
eps = 2.0E-4
Результат: 11.0

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

Прогон тестов через функцию sqrt стандартной библиотеки C++

```
Test 1
Actual value of sqrt(1 + x): 1.09545
Test 2
Actual value of sqrt(1 + x): 4
Test 3
Actual value of sqrt(1 + x): 7.28011
Test 4
Actual value of sqrt(1 + x): 31.5278
Test 5
Actual value of sqrt(1 + x): 9.05539
Test 6
Actual value of sqrt(1 + x): 11
```

Решение на 9 баллов

Файл: https://github.com/khadzakos/asm_course/blob/main/ihw2/ihw_2_macros.asm

```
.macro endl() # Макрос перевода строки
    .data
        endl: .asciz "\n"
    .text
        li a7, 4
        la a0, endl
        ecall
.end_macro

.macro push(%x)
    addi sp, sp, -4
    fsw %x, 0(sp)
.end_macro

.macro pop(%x)
    flw %x, 0(sp)
    addi sp, sp, 4
.end_macro

.macro print_str(%x) # Макрос вывода строки
    .data
       str: .asciz %x
```

```
   .text
       li a7, 4
       la a0, str
       ecall
.end_macro

.macro read_float(%x) # Макрос вывода float
    push(fa0) # Сохраняем

    li a7, 6
    ecall
    fmv.s %x, fa0

    pop(fa0) # Восстанавливаем
.end_macro

.macro print_float(%x) # Макрос вывода float
    addi sp, sp, -4
    fsw fa0, 0(sp) # Сохраняем

    fmv.s fa0, %x
    li a7, 2
    ecall

    flw fa0, 0(sp) # Восстанавливаем
    addi sp, sp, 4
.end_macro

.macro sqrt(%x) # Макрос вызова функции
     fmv.s fa0, %x # В функцию передается только один параметр x
     jal sqrt_x
.end_macro

.data
null: .float 0.0
one: .float 1.0
pow: .float -0.5
epsilon: .float 0.0005

.text
.globl main
main:
    print_str("Данная программа ищет корень 1 + x, используя степенной
```

```
ряд\n")
    flw f0, null, t1

enter_number: # Ввод числа до момента корректного ввода
    print_str("Введите элемент(неотрицательное число): ")

    read_float(fs0)
    flt.s t1, fs0, f0
    beqz t1, end_enter_number

    print_str("Число должно быть неотрицательным!\n")
    j enter_number
end_enter_number:

    sqrt(fs0) # Вызов функции с ЛЮБЫМИ входнвми
    # Результат лежит в fa0, сразу выводим
    print_str("Результат sqrt(1 + x) с точностью ошибки 0.0005%: ")
    print_float(fa0)

    li a7, 10
    ecall

sqrt_x:
    addi sp, sp, -16
    fsw fs0, 8(sp)    # Сохраняем float регистры
    fsw fs1, 4(sp)
    fsw fs2, 0(sp)

    flw f0, one, t1
    flt.s t5, fa0, f0 # Проверка, что число меньше или больше либо равно
1, чтобы установить значение fs0 = x или fs0 = 1/x соответственно
    beqz t5, x_greater
x_less:
    fmv.s fs0, fa0 # fs0 = x
    j x_next
x_greater:
    # Вычисляем fs0 = 1/x
    flw f0, one, t1
    fdiv.s fs0, f0, fa0 # fs0 = 1/x
    j x_next
x_next:

    flw fs1, one, t1  # fs1 = 1 - назовем его term
```

```
    flw fs2, one, t1  # fs2 = 1 - назовем его result
    li t0, 1 # счетчик - назовем n

loop:
    # Вычисляем (2*n - 3) работает корректно до o(x^6)(этого достаточно
для нащей точности)
    add t1, t0, t0  # t1 = 2*n
    addi t1, t1, -3 # t1 = 2*n - 3
    fcvt.s.w f1, t1 # Конвертируем в float

    # Умножаем на -0.5
    flw f2, pow, t1
    fmul.s f1, f1, f2

    # Умножаем на (1/x), если x >= 1 или на x, если x < 1
    fmul.s f1, f1, fs0

    # Делим на n
    fcvt.s.w f2, t0  # Конвертируем n в float
    fdiv.s f1, f1, f2

    # Умножаем текущий term
    fmul.s fs1, fs1, f1 # Новый term

    # Добавляем к result
    fadd.s fs2, fs2, fs1



    # Проверяем условие выхода |term| > epsilon * result(проверяем,
текущий член изменяет сильнее, чем на 0.005%)
    fabs.s f1, fs1 # |term|
    flw f2, epsilon, t1 # загружаем epsilon
    fmul.s f3, f2, fs2 # epsilon * result
    fle.s t1, f3, f1 # if |term| >= epsilon * result

    # Увеличиваем n
    addi t0, t0, 1

    bnez t1, loop

    beqz t5, result_greater # Проверка, что число меньше или больше либо
равно 1(t5 мы не меняли, в нем лежит результат прошлой проверки flt.s
t5, fa0, f0)
```

```
result_less:
    fmv.s fa0, fs2
    j result_next
result_greater:
    # Умножаем на sqrt(x) — этого требует степенная форма, где x >= 1
    fsqrt.s f0, fa0
    fmul.s fa0, f0, fs2
    j result_next
result_next:
    # Эпилог
    j done

done:
    # Восстанавливаем сохраненные регистры
    flw fs0, 8(sp)
    flw fs1, 4(sp)
    flw fs2, 0(sp)
    addi sp, sp, 16
    ret
```

Пояснение

Добавлены макросы **добавления на стек и удаления со стека, ввода и вывода float, вывода строки и вызова функции**. В main все вводы, выводы и вызовы были заменены на макросов. Кроме добавления.

Результат работы команды

```
Данная программа ищет корень 1 + x, используя степенной ряд
Введите элемент(неотрицательное число): 993
Результат sqrt(1 + x) с точностью ошибки 0.0005%: 31.527765
-- program is finished running (0) --
```