

Национальный исследовательский университет
«Высшая школа экономики»

ОТЧЕТ ПО ЗАДАНИЮ №3
«Арифметика с плавающей точкой»
Вариант 3

Выполнил:
студент 213 группы
Фам Ч. Х.

Преподаватель:
Виденин С. А.

Москва
2022

1. Условие задачи

Разработать программу, вычисляющую с помощью степенного ряда с точностью не хуже 0,1% значение функции $\cos(x)$ для заданного параметра x .

2. Код на языке Си

```
1  #include <stdio.h>
2  #include <math.h>
3
4  void add_one(int *i) {
5      int tmp = 1;
6      *i += tmp;
7  }
8
9  int main() {
10     double x, an = 1, S = 1;
11     scanf( format: "%lf", &x);
12     printf( format: "cos(%lf) = ", x);
13     x = x - (int) (x / 2 / M_PI) * 2 * M_PI;
14     for (int i = 1; fabs( X: an / S) > 0.001; add_one( i: &i)) {
15         an *= -x * x / (2 * i * (2 * i - 1));
16         S += an;
17     }
18     printf( format: "%lf\n", S);
19     return 0;
20 }
```

3. Используемые опции

- O0 убирает любую оптимизацию
- Wall задает режим предупреждения сомнительных конструкциях в исходном коде
- masm=intel устанавливает синтаксиса Intel
- fno-asynchronous-unwind-tables одна из опций отладки
- fcf-protection=none убирает технологию CET из кода программы

```
rel@rel-VirtualBox:~$ gcc -O0 -Wall -masm=intel -fno-asynchronous-unwind-tables  
-fcf-protection=none cosx.c -o cosx -save-temps
```

Ассемблерная программа откомпилирована без оптимизирующих и отладочных опций, макросы отсутствуют.

4. Результаты тестов

```
rel@rel-VirtualBox:~$ ./cosx
3.14
cos(3.140000) = -1.000003
rel@rel-VirtualBox:~$ ./cosx_old
3.14
cos(3.140000) = -1.000003
rel@rel-VirtualBox:~$ ./cosx
0
cos(0.000000) = 1.000000
rel@rel-VirtualBox:~$ ./cosx_old
0
cos(0.000000) = 1.000000
rel@rel-VirtualBox:~$ ./cosx
1.57
cos(1.570000) = 0.000796
rel@rel-VirtualBox:~$ ./cosx_old
1.57
cos(1.570000) = 0.000796
rel@rel-VirtualBox:~$ ./cosx
18
cos(18.000000) = 0.660329
rel@rel-VirtualBox:~$ ./cosx_old
18
cos(18.000000) = 0.660329
rel@rel-VirtualBox:~$ ./cosx
6.28
cos(6.280000) = 0.999973
rel@rel-VirtualBox:~$ ./cosx_old
6.28
cos(6.280000) = 0.999973
rel@rel-VirtualBox:~$ ./cosx
10
cos(10.000000) = -0.839069
rel@rel-VirtualBox:~$ ./cosx_old
10
cos(10.000000) = -0.839069
```

Обе программы выдают одинаковый результат на одних и тех же входных данных.

5. Ассемблерный код с комментариями

```
.file "cosx.c"
.intel_syntax noprefix
.text
.globl      add_one
.type add_one, @function
add_one:
    push    rbp
    mov     rbp, rsp
```

```

    mov     QWORD PTR -24[rbp], rdi #int *i
    mov     DWORD PTR -4[rbp], 1 #local int tmp = 1
    mov     rax, QWORD PTR -24[rbp]
    mov     edx, DWORD PTR [rax] #edx = i
    mov     eax, DWORD PTR -4[rbp] #eax = tmp
    add     edx, eax #i += tmp
    mov     rax, QWORD PTR -24[rbp]
    mov     DWORD PTR [rax], edx #new i
    nop     #void function
    pop     rbp
    ret
    .size add_one, .-add_one
    .section .rodata
.LC1:
    .string "%lf"
.LC2:
    .string "cos(%lf) = "
.LC8:
    .string "%lf\n"
    .text
    .globl  main
    .type main, @function
main:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 48
    mov     rax, QWORD PTR fs:40
    mov     QWORD PTR -8[rbp], rax #canary
    xor     eax, eax
    movsd   xmm0, QWORD PTR .LC0[rip]
    movsd   QWORD PTR -24[rbp], xmm0 #double an = 1
    movsd   xmm0, QWORD PTR .LC0[rip]
    movsd   QWORD PTR -16[rbp], xmm0 #double S = 1
    lea     rax, -32[rbp] #double x
    mov     rsi, rax
    lea     rax, .LC1[rip] #get format string
    mov     rdi, rax
    mov     eax, 0
    call    __isoc99_scanf@PLT #scan x
    mov     rax, QWORD PTR -32[rbp] #get x
    movq    xmm0, rax
    lea     rax, .LC2[rip] #format string
    mov     rdi, rax
    mov     eax, 1
    call    printf@PLT #print x
    movsd   xmm0, QWORD PTR -32[rbp] #xmm0 = x
    movsd   xmm1, QWORD PTR -32[rbp] #xmm1 = x
    movsd   xmm2, QWORD PTR .LC3[rip] #2
    divsd   xmm1, xmm2 #xmm1 / 2
    movsd   xmm2, QWORD PTR .LC4[rip] #Pi
    divsd   xmm1, xmm2 #xmm1 / Pi
    cvtsd2si eax, xmm1 #convert to int 32 and mo to eax
    add     eax, eax # eax *= 2
    pxor    xmm2, xmm2
    cvtsi2sd xmm2, eax #convert to double and mov to xmm2
    movsd   xmm1, QWORD PTR .LC4[rip] #Pi

```

```

    mulsd xmm1, xmm2 #xmm2 * Pi
    subsd xmm0, xmm1 #x - xmm1
    movsd QWORD PTR -32[rbp], xmm0 #mov new x to x
    mov     DWORD PTR -36[rbp], 1 #for int i = 1
    jmp     .L3
.L4:
    movsd xmm0, QWORD PTR -32[rbp] #xmm0 = x
    movq   xmm1, QWORD PTR .LC5[rip] #-1
    xorpd  xmm1, xmm0 #-1 * x
    movsd xmm0, QWORD PTR -32[rbp] #xmm0 = x
    mulsd  xmm0, xmm1 #-x * x
    mov    eax, DWORD PTR -36[rbp] #eax = i
    add    eax, eax #2 * i
    lea    edx, -1[rax] #2i - 1
    mov    eax, DWORD PTR -36[rbp] #eax = i
    imul   eax, edx #i * (2i - 1)
    add    eax, eax #i * 2 * (2i - 1)
    pxor   xmm1, xmm1
    cvtsi2sd  xmm1, eax #convert to double
    divsd  xmm0, xmm1 #-x * x / ...
    movsd  xmm1, QWORD PTR -24[rbp] #an
    mulsd  xmm0, xmm1 #an * ...
    movsd  QWORD PTR -24[rbp], xmm0 #new an
    movsd  xmm0, QWORD PTR -16[rbp] #get S
    addsd  xmm0, QWORD PTR -24[rbp] #S += an
    movsd  QWORD PTR -16[rbp], xmm0 #new S
    lea    rax, -36[rbp] #get i
    mov    rdi, rax
    call   add_one #++i
.L3:
    movsd  xmm0, QWORD PTR -24[rbp] #xmm0 = an
    divsd  xmm0, QWORD PTR -16[rbp] #xmm0 / S
    movq   xmm1, QWORD PTR .LC6[rip] #get const
    andpd  xmm0, xmm1
    comisd  xmm0, QWORD PTR .LC7[rip] #get eflags
    ja     .L4
    mov    rax, QWORD PTR -16[rbp] #get S
    movq   xmm0, rax
    lea    rax, .LC8[rip] #format string
    mov    rdi, rax
    mov    eax, 1
    call   printf@PLT #print S
    mov    eax, 0
    mov    rdx, QWORD PTR -8[rbp]
    sub    rdx, QWORD PTR fs:40 #canary
    je     .L6
    call   __stack_chk_fail@PLT
.L6:
    leave
    ret
    .size   main, .-main
    .section .rodata
    .align 8
.LC0:

```

```

        .long      0
        .long      1072693248
        .align 8
.LC3:
        .long      0
        .long      1073741824
        .align 8
.LC4:
        .long      1413754136
        .long      1074340347
        .align 16
.LC5:
        .long      0
        .long      -2147483648
        .long      0
        .long      0
        .align 16
.LC6:
        .long      -1
        .long      2147483647
        .long      0
        .long      0
        .align 8
.LC7:
        .long      -755914244
        .long      1062232653
        .ident      "GCC: (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0"
        .section    .note.GNU-stack,"",@progbits

```

6. Ассемблерный код компилятора языка C

```

.file "cosx.c"
.text
.globl      add_one
.type add_one, @function
add_one:
.LFB0:
        .cfi_startproc
        endbr64
        pushq %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq %rsp, %rbp
        .cfi_def_cfa_register 6
        movq %rdi, -24(%rbp)
        movl $1, -4(%rbp)
        movq -24(%rbp), %rax
        movl (%rax), %edx
        movl -4(%rbp), %eax
        addl %eax, %edx
        movq -24(%rbp), %rax
        movl %edx, (%rax)
        nop

```

```

    popq %rbp
    .cfi_def_cfa 7, 8
    ret
.cfi_endproc
.LFE0:
    .size add_one, .-add_one
    .section .rodata
.LC1:
    .string "%lf"
.LC2:
    .string "cos(%lf) = "
.LC8:
    .string "%lf\n"
    .text
    .globl main
    .type main, @function
main:
.LFB1:
    .cfi_startproc
    endbr64
    pushq %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq %rsp, %rbp
    .cfi_def_cfa_register 6
    subq $48, %rsp
    movq %fs:40, %rax
    movq %rax, -8(%rbp)
    xorl %eax, %eax
    movsd .LC0(%rip), %xmm0
    movsd %xmm0, -24(%rbp)
    movsd .LC0(%rip), %xmm0
    movsd %xmm0, -16(%rbp)
    leaq -32(%rbp), %rax
    movq %rax, %rsi
    leaq .LC1(%rip), %rax
    movq %rax, %rdi
    movl $0, %eax
    call __isoc99_scanf@PLT
    movq -32(%rbp), %rax
    movq %rax, %xmm0
    leaq .LC2(%rip), %rax
    movq %rax, %rdi
    movl $1, %eax
    call printf@PLT
    movsd -32(%rbp), %xmm0
    movsd -32(%rbp), %xmm1
    movsd .LC3(%rip), %xmm2
    divsd %xmm2, %xmm1
    movsd .LC4(%rip), %xmm2
    divsd %xmm2, %xmm1
    cvttss2sd %xmm1, %eax
    addl %eax, %eax
    pxor %xmm2, %xmm2
    cvtsi2sdl %eax, %xmm2
    movsd .LC4(%rip), %xmm1

```

```

    mulsd %xmm2, %xmm1
    subsd %xmm1, %xmm0
    movsd %xmm0, -32(%rbp)
    movl $1, -36(%rbp)
    jmp .L3
.L4:
    movsd -32(%rbp), %xmm0
    movq .LC5(%rip), %xmm1
    xorpd %xmm0, %xmm1
    movsd -32(%rbp), %xmm0
    mulsd %xmm1, %xmm0
    movl -36(%rbp), %eax
    addl %eax, %eax
    leal -1(%rax), %edx
    movl -36(%rbp), %eax
    imull %edx, %eax
    addl %eax, %eax
    pxor %xmm1, %xmm1
    cvtsi2sdl %eax, %xmm1
    divsd %xmm1, %xmm0
    movsd -24(%rbp), %xmm1
    mulsd %xmm1, %xmm0
    movsd %xmm0, -24(%rbp)
    movsd -16(%rbp), %xmm0
    addsd -24(%rbp), %xmm0
    movsd %xmm0, -16(%rbp)
    leaq -36(%rbp), %rax
    movq %rax, %rdi
    call add_one
.L3:
    movsd -24(%rbp), %xmm0
    divsd -16(%rbp), %xmm0
    movq .LC6(%rip), %xmm1
    andpd %xmm1, %xmm0
    comisd .LC7(%rip), %xmm0
    ja .L4
    movq -16(%rbp), %rax
    movq %rax, %xmm0
    leaq .LC8(%rip), %rax
    movq %rax, %rdi
    movl $1, %eax
    call printf@PLT
    movl $0, %eax
    movq -8(%rbp), %rdx
    subq %fs:40, %rdx
    je .L6
    call __stack_chk_fail@PLT
.L6:
    leave
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE1:
    .size main, .-main
    .section .rodata
    .align 8

```



```

.LC0:
    .long 0
    .long 1072693248
    .align 8
.LC3:
    .long 0
    .long 1073741824
    .align 8
.LC4:
    .long 1413754136
    .long 1074340347
    .align 16
.LC5:
    .long 0
    .long -2147483648
    .long 0
    .long 0
    .align 16
.LC6:
    .long -1
    .long 2147483647
    .long 0
    .long 0
    .align 8
.LC7:
    .long -755914244
    .long 1062232653
    .ident      "GCC: (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0"
    .section    .note.GNU-stack,"",@progbits
    .section    .note.gnu.property,"a"
    .align 8
    .long 1f - 0f
    .long 4f - 1f
    .long 5
0:
    .string     "GNU"
1:
    .align 8
    .long 0xc0000002
    .long 3f - 2f
2:
    .long 0x3
3:
    .align 8
4:

```

7. Дополнение на 5 баллов

В реализованной программе используется функция **add_one** с передачей параметров через параметры.

Используется локальная переменная в **int tmp**, которая отображается на стек в функции.

В ассемблерный код добавлены комментарии для формальных параметров, описывающие связь между параметрами языка Си и регистрами (пункт 5 содержит необходимый ассемблерный код).

8. Дополнение на 6 баллов

Была проведена модификация ассемблерного кода, целью которой было уменьшить количество обращений к памяти за счет использования регистров.

Скриншоты изменений кода:

```
6 add_one:
7 push rbp
8 mov rbp, rsp

9 mov DWORD PTR -4[rbp], 1 #local int tmp = 1
10 mov rax, rdi
11 mov edx, DWORD PTR [rax] #edx = i
12 mov eax, 1 #eax = tmp
13 add edx, eax #i += tmp
14 mov rax, rdi
15 mov DWORD PTR [rax], edx #new i
16 nop #void function
17 pop rbp
18 ret

38 movsd QWORD PTR -16[rbp], xmm0 #double S = 1
39 lea rsi, -32[rbp] #double x
40 lea rdi, LC1[rip] #get format string
41 xor eax, eax

42 call __isoc99_scanf@PLT #scan x
43 mov rax, QWORD PTR -32[rbp] #get x
44 movq xmm0, rax
45 lea rdi, LC2[rip] #format string

46 mov eax, 1
61 movsd QWORD PTR -32[rbp], xmm0 #mov new x to x
62 mov ebx, 1 #for int i = 1
63 mov DWORD PTR -36[rbp], ebx
64 jmp .L3

70 mulsd xmm0, xmm1 #-x * x
71 mov eax, ebx #eax = i
72 add eax, eax #2 * i
73 lea edx, -1[rax] #2i - 1
74 mov eax, ebx #eax = i
75 imul eax, edx #i * (2i - 1)
84 addsd xmm0, QWORD PTR -24[rbp] #S += an
85 movsd QWORD PTR -16[rbp], xmm0 #new S
86 lea rdi, -36[rbp] #get i

87 call add_one #++i
88 mov ebx, DWORD PTR -36[rbp]

89 .L3:
94 comisd xmm0, QWORD PTR LC7[rip] #get eflags
95 ja .L4
96 movsd xmm0, QWORD PTR -16[rbp] #get S
97 lea rdi, LC8[rip] #format string

98 mov eax, 1
99 call printf@PLT #print S
```

```
6 add_one:
7 push rbp
8 mov rbp, rsp
9 mov QWORD PTR -24[rbp], rdi #int i
10 mov DWORD PTR -4[rbp], 1 #local int tmp = 1
11 mov rax, QWORD PTR -24[rbp]
12 mov edx, DWORD PTR [rax] #edx = i
13 mov eax, DWORD PTR -4[rbp] #eax = tmp
14 add edx, eax #i += tmp
15 mov rax, QWORD PTR -24[rbp]
16 mov DWORD PTR [rax], edx #new i
17 nop #void function
18 pop rbp
19 ret

41 movsd QWORD PTR -16[rbp], xmm0 #double S = 1
42 lea rax, -32[rbp] #double x
43 mov rsi, rax
44 lea rax, LC1[rip] #get format string
45 mov rdi, rax
46 mov eax, 0

47 call __isoc99_scanf@PLT #scan x
48 mov rax, QWORD PTR -32[rbp] #get x
49 movq xmm0, rax
50 lea rax, LC2[rip] #format string
51 mov rdi, rax
52 mov eax, 1
62 movsd QWORD PTR -32[rbp], xmm0 #mov new x to x
63 mov DWORD PTR -36[rbp], 1 #for int i = 1

69 jmp .L3

75 mulsd xmm0, xmm1 #-x * x
76 mov eax, DWORD PTR -36[rbp] #eax = i
77 add eax, eax #2 * i
78 lea edx, -1[rax] #2i - 1
79 mov eax, DWORD PTR -36[rbp] #eax = i
80 imul eax, edx #i * (2i - 1)
89 addsd xmm0, QWORD PTR -24[rbp] #S += an
90 movsd QWORD PTR -16[rbp], xmm0 #new S
91 lea rax, -36[rbp] #get i
92 mov rdi, rax
93 call add_one #++i

94 .L3:
99 comisd xmm0, QWORD PTR LC7[rip] #get eflags
100 ja .L4
101 mov rax, QWORD PTR -16[rbp] #get S
102 movq xmm0, rax
103 lea rax, LC8[rip] #format string
104 mov rdi, rax
105 mov eax, 1
106 call printf@PLT #print S
```

В функции **add_one** были убраны лишние обращения к памяти и лишние действия. Локальная переменная в цикле **for** была заменена на регистр **ebx** вместо стека. Так же были убраны лишние действия, связанные с ненужным использованием команды **mov**. Размер объектных файлов исходной и модифицированной поменялся (Исходная 2,5 кБ, модифицированная 1,9 кБ). Были так же получены размеры бинарных файлов(Без рефакторинга 12,1 кБ, после 11,5 кБ).

```
rel@rel-VirtualBox:~$ ./cosx_old
3.14
cos(3.140000) = -1.000003
rel@rel-VirtualBox:~$ ./cosx_ref
3.14
cos(3.140000) = -1.000003
rel@rel-VirtualBox:~$ ./cosx_old
1.57
cos(1.570000) = 0.000796
rel@rel-VirtualBox:~$ ./cosx_ref
1.57
cos(1.570000) = 0.000796
rel@rel-VirtualBox:~$ ./cosx_old
0
cos(0.000000) = 1.000000
rel@rel-VirtualBox:~$ ./cosx_ref
0
cos(0.000000) = 1.000000
rel@rel-VirtualBox:~$ ./cosx_old
6.28
cos(6.280000) = 0.999973
rel@rel-VirtualBox:~$ ./cosx_ref
6.28
cos(6.280000) = 0.999973
rel@rel-VirtualBox:~$ ./cosx_old
1.04
cos(1.040000) = 0.506221
rel@rel-VirtualBox:~$ ./cosx_ref
1.04
cos(1.040000) = 0.506221
rel@rel-VirtualBox:~$
```

Результаты выполнения программы после рефакторинга ассемблерного кода не изменились.