

федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
«Московский институт электронной техники»

Направление подготовки 09.03.04 «Программная инженерия»
Дисциплина «Электротехника»

Лабораторная РАБОТА №8
Модули и функции. Вызов функций libc и libm.

Работу выполнили студенты группы ПИН-24 Баранов Д.А. и Демочкина А.В.
Работу проверил ассистент Института СПИНТех Фомин Р.А.

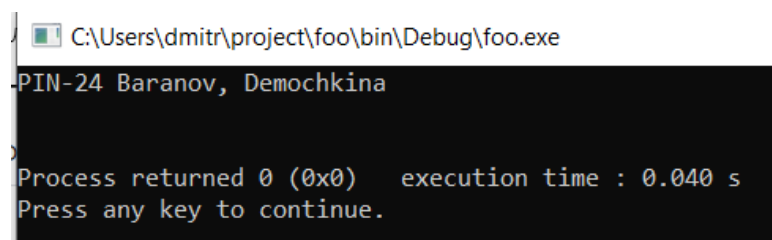
Цель работы: изучить процесс компиляции программы на C++; научиться включать в проекты на языке C++ ассемблерные модули; изучить стандартные соглашения о вызовах и их соответствие платформам.

Задание Л8.31. Разработайте программу, выводящую на стандартный вывод группу, номер и состав команды при помощи функции *puts()* библиотеки *libc* (аналогично заданию Л4).

```
1  .data
2      msg:
3      .string "PIN-24 Baranov, Demochkina\n"
4  .text
5  .globl foo
6      foo:
7      sub $8, %rsp
8      lea msg(%rip), %rcx
9      sub $32, %rsp
10     call puts
11     add $32, %rsp
12     add $8, %rsp
13     xor %eax, %eax
14
15     ret
```

```
.data
    msg:
    .string "PIN-24 Baranov, Demochkina\n"
.text
.globl foo
foo:
    sub $8, %rsp
    lea msg(%rip), %rcx
    sub $32, %rsp
    call puts
    add $32, %rsp
    add $8, %rsp
    xor %eax, %eax

    ret
```



```
C:\Users\dmitr\project\foo\bin\Debug\foo.exe
PIN-24 Baranov, Demochkina
Process returned 0 (0x0) execution time : 0.040 s
Press any key to continue.
```

Задание Л8.32. Разработайте программу, реализующую ввод и последующий вывод (эхо) двух значений x и y следующих типов:

- 16-битное целое;
- 32-битное целое;
- 64-битное целое;
- 32-битное число с плавающей запятой;
- 64-битное число с плавающей запятой

с использованием библиотеки `libc` (в частности, функций `puts()`/`printf()`/`scanf()`).

```
1      .data
2      fmt_in_short: .string "%hd"
3      fmt_out_short: .string "Short: %hd\n"
4      fmt_in_integer: .string "%d"
5      fmt_out_integer: .string "Integer: %d\n"
6      fmt_in_long: .string "%ld"
7      fmt_out_long: .string "Long: %ld\n"
8      fmt_in_float: .string "%f"
9      fmt_out_float: .string "Float: %f\n"
10     fmt_in_double: .string "%lf"
11     fmt_out_double: .string "Double: %lf\n"
12
13     msg_prompt1: .string "Enter the short: "
14     msg_prompt2: .string "Enter the integer: "
15     msg_prompt3: .string "Enter the long: "
16     msg_prompt4: .string "Enter the float: "
17     msg_prompt5: .string "Enter the double: "
18
19     .text
20     .globl foo2
21     foo2:
22         sub $40, %rsp
23
24         //short
25         lea msg_prompt1(%rip), %rcx
26         call puts
27
28         lea fmt_in_short(%rip), %rcx
29         lea (%rsp), %rdx
30         call scanf
31
32         lea fmt_out_short(%rip), %rcx
33         mov (%rsp), %edx
34         call printf
```

```

35      //integer
36      lea msg_prompt2(%rip), %rcx
37      call puts
38
39      lea fmt_in_integer(%rip), %rcx
40      lea (%rsp), %rdx
41      call scanf
42
43      lea fmt_out_integer(%rip), %rcx
44      mov (%rsp), %edx
45      call printf
46
47      //long
48      lea msg_prompt3(%rip), %rcx
49      call puts
50
51      lea fmt_in_long(%rip), %rcx
52      lea (%rsp), %rdx
53      call scanf
54
55      lea fmt_out_long(%rip), %rcx
56      mov (%rsp), %edx
57      call printf
58
59
60      //float
61      lea msg_prompt4(%rip), %rcx
62      call puts
63
64      lea fmt_in_float(%rip), %rcx
65      lea (%rsp), %rdx
66      call scanf
67
68      lea fmt_out_float(%rip), %rcx
69      cvtss2sd (%rsp), %xmm1
70      movq %xmm1, %rdx
71      call printf
72
73
74      //double
75      lea msg_prompt5(%rip), %rcx
76      call puts
77
78      lea fmt_in_double(%rip), %rcx
79      lea (%rsp), %rdx
80      call scanf
81
82      lea fmt_out_double(%rip), %rcx
83      movq (%rsp), %rdx
84      call printf
85
86
87      xor %eax, %eax
88      add $40, %rsp
89      ret

```

.data

fmt_in_short: .string "%hd"

```

fmt_out_short: .string "Short: %hd\n"
fmt_in_integer: .string "%d"
fmt_out_integer: .string "Integer: %d\n"
fmt_in_long: .string "%ld"
fmt_out_long: .string "Long: %ld\n"
fmt_in_float: .string "%f"
fmt_out_float: .string "Float: %f\n"
fmt_in_double: .string "%lf"
fmt_out_double: .string "Double: %lf\n"

msg_prompt1: .string "Enter the short: "
msg_prompt2: .string "Enter the integer: "
msg_prompt3: .string "Enter the long: "
msg_prompt4: .string "Enter the float: "
msg_prompt5: .string "Enter the double: "
.text
.globl foo2
foo2:
sub $40, %rsp

//short
lea msg_prompt1(%rip), %rcx
call puts

lea fmt_in_short(%rip), %rcx
lea (%rsp), %rdx
call scanf

lea fmt_out_short(%rip), %rcx
mov (%rsp), %edx
call printf

//integer
lea msg_prompt2(%rip), %rcx
call puts

lea fmt_in_integer(%rip), %rcx
lea (%rsp), %rdx
call scanf

lea fmt_out_integer(%rip), %rcx
mov (%rsp), %edx
call printf

//long
lea msg_prompt3(%rip), %rcx
call puts

lea fmt_in_long(%rip), %rcx
lea (%rsp), %rdx

```

```
call scanf
```

```
lea fmt_out_long(%rip), %rcx  
mov (%rsp), %edx  
call printf
```

```
//float
```

```
lea msg_prompt4(%rip), %rcx  
call puts
```

```
lea fmt_in_float(%rip), %rcx  
lea (%rsp), %rdx  
call scanf
```

```
lea fmt_out_float(%rip), %rcx  
cvtss2sd (%rsp), %xmm1  
movq %xmm1, %rdx  
call printf
```

```
//double
```

```
lea msg_prompt5(%rip), %rcx  
call puts
```

```
lea fmt_in_double(%rip), %rcx  
lea (%rsp), %rdx  
call scanf
```

```
lea fmt_out_double(%rip), %rcx  
movq (%rsp), %rdx  
call printf
```

```
xor %eax, %eax  
add $40, %rsp  
ret
```

Результат:

```
Enter the short:
32767
Short: 32767
Enter the integer:
2147483647
Integer: 2147483647
Enter the long:
2147483647
Long: 2147483647
Enter the float:
0.3434234
Float: 0.343423
Enter the double:
0.923535333535121
Double: 0.923535
```

Переполнение:

[illegible]

Задание Л8.33. Разработайте программу, вычисляющую (вызывая функции `libc/libm`) по введённым значениям x и y с плавающей запятой двойной точности значение z (таблица Л8.1):

Вариант 2.

2	$z = \text{atan2}(x, y)$, угол между вектором (x, y) и осью абсцисс
---	--

Первый вариант:

```
1  .data
2      fmt_in_double: .string "%lf %lf"
3      fmt_out_double: .string "Double: %lf\n\n"
4      msg_prompt: .string "Enter the double: "
5  .text
6
7  .globl foo3
8      foo3:
9          sub $40, %rsp
10
11      //double
12      lea msg_prompt(%rip), %rcx
13      call puts
14
15      lea fmt_in_double(%rip), %rcx
16      lea 0(%rsp), %rdx
17      lea 8(%rsp), %r8d
18      call scanf
19
20      movq 8(%rsp), %xmm0
21      movq 0(%rsp), %xmm1
22      call atan2
23
24      lea fmt_out_double(%rip), %rcx
25      movq %xmm0, %rdx
26      call printf
27
28      add $40, %rsp
29      xor %eax, %eax
30      ret
```

```
.data
    fmt_in_double: .string "%lf %lf"
    fmt_out_double: .string "Double: %lf\n\n"
    msg_prompt: .string "Enter the double: "
.text
```

```
.globl foo3
    foo3:
        sub $40, %rsp

        //double
        lea msg_prompt(%rip), %rcx
        call puts

        lea fmt_in_double(%rip), %rcx
```



```
lea 0(%rsp), %rdx
lea 8(%rsp), %r8d
call scanf
```

```
movq 8(%rsp), %xmm0
movq 0(%rsp), %xmm1
call atan2
```

```
lea fmt_out_double(%rip), %rcx
movq %xmm0, %rdx
call printf
```

```
add $40, %rsp
xor %eax, %eax
ret
```

```
Enter the double:
-1 -1
Double: -2.356194
```

```
Enter the double:
0 1
Double: 1.570796
```

```
Enter the double:
2.5 2.5
Double: 0.785398
```

Второй вариант:

```
2  .data
3      fmt_in_double: .string "%lf %lf"
4      fmt_out_double: .string "Double: %lf\n\n"
5      msg_prompt: .string "Enter the double: "
6  .text
7  .globl foo3_2
8      foo3_2:
9          sub $40, %rsp
10
11      //double
12      lea msg_prompt(%rip), %rcx
13      call puts
14
15      lea fmt_in_double(%rip), %rcx
16      lea 0(%rsp), %rdx
17      lea 8(%rsp), %r8
18      call scanf
19
20      fldl 8(%rsp) // y в стеке
21      fldl 0(%rsp) // x, y в стеке
22      fpatan
23      fstpl 0(%rsp)
24
25
26      lea fmt_out_double(%rip), %rcx
27      movq (%rsp), %rdx
28      call printf
29
30      xor %eax, %eax
31      add $40, %rsp
32      ret
```

```
.data
    fmt_in_double: .string "%lf %lf"
    fmt_out_double: .string "Double: %lf\n\n"
    msg_prompt: .string "Enter the double: "
.text
.globl foo3_2
foo3_2:
    sub $40, %rsp

    //double
    lea msg_prompt(%rip), %rcx
    call puts

    lea fmt_in_double(%rip), %rcx
```

```
lea 0(%rsp), %rdx
lea 8(%rsp), %r8
call scanf
```

```
fldl 8(%rsp) // y в стеке
fldl 0(%rsp) // x, y в стеке
fpatan
fstpl 0(%rsp)
```

```
lea fmt_out_double(%rip), %rcx
movq (%rsp), %rdx
call printf
```

```
xor %eax, %eax
add $40, %rsp
ret
```

```
Enter the double:
1 -1
Double: -0.785398
```

```
Enter the double:
-1 0
Double: 3.141593
```

```
Enter the double:
23.1 21.1
Double: 0.740180
```

Задание Л8.34. Задайте с клавиатуры N и x_0 и напечатайте первые N членов целочисленной последовательности: $x_{i+1} = \begin{cases} x_i/2, & x_i\%2 = 0 \\ 3x_i + 1, & x_i\%2 \neq 0 \end{cases}$.

```

1  .data
2      fmt_in_integer: .string "%d %u"
3      fmt_out_integer: .string "Value: %d      Step: %u\n\n"
4
5      msg_prompt2: .string "Enter the integer: "
6  .text
7  .globl foo4_2
8      foo4_2:
9      sub $40, %rsp
10
11     //integer
12     lea msg_prompt2(%rip), %rcx
13     call puts
14
15     //ввод значения числа, и количества членов последовательности
16     lea fmt_in_integer(%rip), %rcx
17     lea 0(%rsp), %rdx
18     lea 8(%rsp), %r8
19     call scanf
20
21
22     movl 8(%rsp), %esi
23     movl 0(%rsp), %ebx
24     begin:
25         cmpl $0, %esi
26         je end_loop
27
28         // счётчик цикла
29         subl $1, %esi

```

```

31      //определение остатка от деления на 2
32      movl %ebx, %edi
33      and $1, %edi
34      cmpl $1, %edi
35      je setting_1
36      ine setting_2
37
38      setting_1:
39          // умножение на 3 и прибавление 1
40          imul $3, %ebx
41          add $1, %ebx
42          jmp end_check
43      setting_2:
44          // целове от деления на 2
45          sar $1, %ebx
46          jmp end_check
47      end_check:
48          nop
49
50      // вывод значения и номер шага
51      lea fmt_out_integer(%rip), %rcx
52      movl %ebx, 0(%rsp)
53      movl %ebx, 0(%rsp)
54      movq 0(%rsp), %rdx
55      movq 8(%rsp), %r8
56      call printf
57
58      //повтор цикла
59      jmp end
60      end:
61      jmp begin
62      end_loop:
63      nop
64
65      xor %eax, %eax
66      add $40, %rsp
67      ret

```

.data

fmt_in_integer: .string "%d %u"

fmt_out_integer: .string "Value: %d Step: %u\n\n"

msg_prompt2: .string "Enter the integer: "

.text

.globl foo4_2

foo4_2:

sub \$40, %rsp

```

//integer
lea msg_prompt2(%rip), %rcx
call puts

//ввод значения числа, и количества членов последовательности
lea fmt_in_integer(%rip), %rcx
lea 0(%rsp), %rdx
lea 8(%rsp), %r8
call scanf

movl 8(%rsp), %esi
movl 0(%rsp), %ebx
begin:
    cmpl $0, %esi
    je end_loop

    // счётчик цикла
    subl $1, %esi

    //определение остатка от деления на 2
    movl %ebx, %edi
    and $1, %edi
    cmpl $1, %edi
    je setting_1
    jne setting_2

setting_1:
    // умножение на 3 и прибавление 1
    imul $3, %ebx
    add $1, %ebx
    jmp end_check
setting_2:
    // целове от деления на 2
    sar $1, %ebx
    jmp end_check
end_check:
    nop

    // вывод значения и номер шага
    lea fmt_out_integer(%rip), %rcx
    movl %ebx, 0(%rsp)
    movl %ebx, 0(%rsp)
    movq 0(%rsp), %rdx
    movq 8(%rsp), %r8
    call printf

    //повтор цикла
    jmp end

```

```

end:
    jmp begin
end_loop:
    nop

xor %eax, %eax
add $40, %rsp
ret

```

```

Enter the integer:
33 16
Value: 100      Step: 16
Value: 50       Step: 100
Value: 25       Step: 50
Value: 76       Step: 25
Value: 38       Step: 76
Value: 19       Step: 38
Value: 58       Step: 19
Value: 29       Step: 58
Value: 88       Step: 29
Value: 44       Step: 88
Value: 22       Step: 44
Value: 11       Step: 22
Value: 34       Step: 11
Value: 17       Step: 34
Value: 52       Step: 17
Value: 26       Step: 52

```

```

Enter the integer:
16 6
Value: 8      Step: 6
Value: 4      Step: 8
Value: 2      Step: 4
Value: 1      Step: 2
Value: 4      Step: 1
Value: 2      Step: 4

```