

Отчет

Лабораторная работа 5 (0101 = 5)

Модули и функции. Вызов функций стандартной библиотеки C (libc и libm)

Операционная система ОС Windows, 64-разрядная ОС, соглашение о вызовах Microsoft x64 (x86-64).

Задание Л5.31.

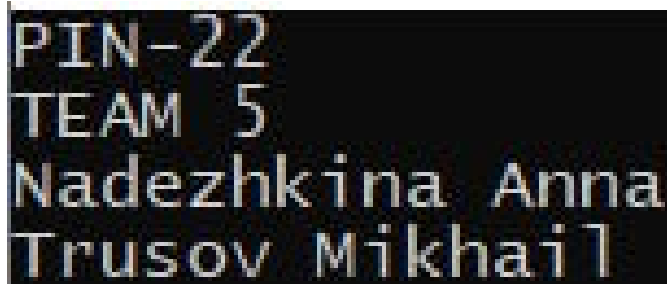
Разработайте программу, выводящую на стандартный вывод группу, номер и состав команды при помощи функции `puts()` библиотеки `libc` (аналогично заданию Л1.31).

Программный код:

```
.data
    msg:
        .string "PIN-22\nTEAM 5\nNadezhkina Anna\nTrusov Mikhail\n"
.text
.globl main

main:
    sub $8, %rsp                // выравнивание стека
    lea msg(%rip), %rcx         // загрузка адреса строки msg в регистр rcx
    sub $32, %rsp              // выделение 32 байт под аргументы функции puts на стеке
    call puts                  // вызов функции puts
    add $32, %rsp              // освобождение выделенных на стеке 32 байт после вызова puts
    add $8, %rsp               // восстановление значения rsp
    xor %eax, %eax             // устанавливает регистр eax в 0.
    ret
```

Вывод:



```
PIN-22
TEAM 5
Nadezhkina Anna
Trusov Mikhail
```

Задание Л5.з2.

Выделите в стеке *main()* место под переменные нескольких типов (по значению на каждый тип): – 16-битное целое; – 32-битное целое; – 64-битное целое; – 32-битное число с плавающей запятой; – 64-битное число с плавающей запятой. с учётом выравнивания — адрес переменной должен быть кратен как минимум её размеру. Введите в каждую из выделенных областей памяти по значению соответствующего типа при помощи *scanf()*. Напечатайте значения из памяти при помощи *printf()*. Обратите внимание, что *printf()* и *scanf()* имеют переменное число аргументов, что во многих соглашениях требует дополнительных действий

Программный код:

```
.data
short_format_in: .string "%hd"
short_format_out: .string "Vvedyonnyj short: %hd\n"
int_format_in: .string "%d"
int_format_out: .string "Vvedyonnyj int: %d\n"
long_format_in: .string "%ld"
long_format_out: .string "Vvedyonnyj long: %ld\n"
float_format_in: .string "%f"
float_format_out: .string "Vvedyonnyj float: %f\n"
double_format_in: .string "%lf"
double_format_out: .string "Vvedyonnyj double: %lf\n"
```

```
enter_short_msg: .string "Vvedite short: "
enter_int_msg: .string "\nVvedite int: "
enter_long_msg: .string "\nVvedite long: "
enter_float_msg: .string "\nVvedite float: "
enter_double_msg: .string "\nVvedite double: "
```

```
.text
```

```
.globl main
```

```
main:
```

```
sub $56, %rsp
```

```
// short
```

```
lea enter_short_msg(%rip), %rcx // загрузка адреса строки приглашения в регистр rcx
```

```
sub $32, %rsp
```

```
call puts // вызов функции puts для печати строки приглашения
```

```
add $32, %rsp
```

```
lea short_format_in(%rip), %rcx // загрузка адреса форматной строки для ввода short в регистр rcx
```

```
lea (%rsp), %rdx // загрузка адреса на вершине стека в регистр rdx (для хранения введенного значения)
```

```
sub $32, %rsp
```

```
call scanf // вызов функции scanf для ввода short и сохранения его на вершине стека
```

```
add $32, %rsp
```

```
lea short_format_out(%rip), %rcx // загрузка адреса форматной строки для вывода short в rcx
```

```
mov (%rsp), %edx // помещение значения short с вершины стека в регистр edx
```

```
sub $32, %rsp
```

```
call printf // вызов функции printf для печати значения short
```

```
add $32, %rsp
```

```
// int
lea enter_int_msg(%rip), %rcx
sub $32, %rsp
call puts
add $32, %rsp
```

```
lea int_format_in(%rip), %rcx
lea (%rsp), %rdx
sub $32, %rsp
call scanf
add $32, %rsp
```

```
lea int_format_out(%rip), %rcx
mov (%rsp), %edx
sub $32, %rsp
call printf
add $32, %rsp
```

```
// long
lea enter_long_msg(%rip), %rcx
sub $32, %rsp
call puts
add $32, %rsp
```

```
lea long_format_in(%rip), %rcx
lea (%rsp), %rdx
sub $32, %rsp
```

```
call scanf
```

```
add $32, %rsp
```

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

```
mov (%rsp), %edx
```

```
sub $32, %rsp
```

```
call printf
```

```
add $32, %rsp
```

```
// float
```

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

```
sub $32, %rsp
```

```
call puts
```

```
add $32, %rsp
```

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

```
lea (%rsp), %rdx
```

```
sub $32, %rsp
```

```
call scanf
```

```
add $32, %rsp
```

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

```
cvtss2sd (%rsp), %xmm1
```

```
movq %xmm1, %rdx
```

```
sub $32, %rsp
```

```
call printf
```

```
add $32, %rsp
```

```
// double  
lea enter_double_msg(%rip), %rcx  
sub $32, %rsp  
call puts  
add $32, %rsp
```

```
lea double_format_in(%rip), %rcx  
lea (%rsp), %rdx  
sub $32, %rsp  
call scanf  
add $32, %rsp
```

```
lea double_format_out(%rip), %rcx  
movq (%rsp), %rdx  
sub $32, %rsp  
call printf  
add $32, %rsp
```

```
xor %eax, %eax  
add $56, %rsp  
ret
```

Вывод:

```
Vvedite short:
12345
Vvedyonnyj short: 12345

Vvedite int:
123456
Vvedyonnyj int: 123456

Vvedite long:
1234567
Vvedyonnyj long: 1234567

Vvedite float:
1.23456
Vvedyonnyj float: 1.234560

Vvedite double:
1.2345678
Vvedyonnyj double: 1.234568
```

Задание Л5.з3.

Разработайте программу, вычисляющую по введённым значениям x и y с плавающей запятой двойной точности значение z (таблица Л5.1), вызывая функции `libm pow()/atan2()`.

Программный код:

```
.data
    double_format_in: .string "%lf %lf"
    double_format_out: .string "Result: %lf\n"
    enter_double_msg: .string "Vvedite dva double (osnovanie i stepen): "

.text

.globl main

main:
```

```
sub $56, %rsp
```

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

```
sub $32, %rsp
```

```
call puts
```

```
add $32, %rsp
```

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

```
lea 0(%rsp), %rdx // загрузка адреса первого double из стека в регистр rdx.
```

```
lea 8(%rsp), %r8 // загрузка адреса второго double на стеке в регистр r8
```

```
sub $32, %rsp
```

```
call scanf
```

```
add $32, %rsp
```

```
movsd 0(%rsp), %xmm0 // загрузка первого значения из памяти в регистр xmm0.
```

```
movsd 8(%rsp), %xmm1 // загрузка второго значения из памяти в регистр xmm1.
```

```
sub $32, %rsp
```

```
call pow // вызов функции pow
```

```
add $32, %rsp
```

```
movsd %xmm0, (%rsp) // сохранение результата функции
```

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

```
movq 0(%rsp), %rdx
```

```
sub $32, %rsp
```

```
call printf
```



```
add $32, %rsp
```

```
xor %eax, %eax
```

```
add $56, %rsp
```

```
ret
```

Вывод:

```
Vvedite dva double (osnovanie i stepen):  
2 5  
Result: 32.000000
```

```
Vvedite dva double (osnovanie i stepen):  
123.45 4.5  
Result: 2580536012.437853
```