ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»

Факультет безопасности информационных технологий

Дисциплина:

«Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

Поразрядные и логические операции.

Выполнил:

Студент группы N3153

Синюта А.А.

Проверил:

Грозов В.А.

Санкт-Петербург

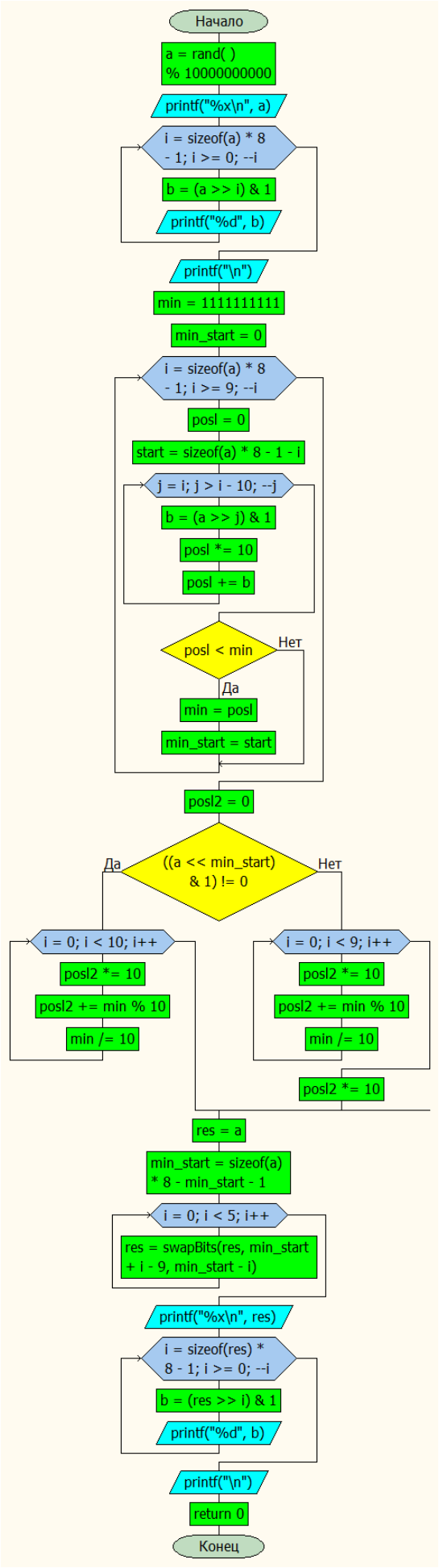
2021г.

Вариант 27.

Задание: написать программу, которая получает случайное целое типа int, выводит его двоичное представление на экран (необходимо выводить все разряды числа), выполняет преобразование в соответствии с вариантом (Табл. 1), затем выводит на экран двоичное представление результата преобразования.

27. Найти в числе непрерывную последовательность из 10 битов, имеющую наименьшее значение, и изменить в ней порядок следования битов на обратный. 0xDEADBEEF → 0xDFB53EEF.

Блок-схема алгоритма преобразования для программы на C:


Начало
a = rand( ) % 10000000000
printf("%x\n", a)
i = sizeof(a) * 8 - 1; i >= 0; --i
b = (a >> i) & 1
printf("%d", b)
printf("\n")
min = 1111111111
min_start = 0
i = sizeof(a) * 8 - 1; i >= 9; --i
posl = 0
start = sizeof(a) * 8 - 1 - i
j = i; j > i - 10; --j
b = (a >> j) & 1
posl *= 10
posl += b
posl < min
Нет
Да
min = posl
min_start = start
posl2 = 0
((a << min_start) & 1) != 0
Да
Нет
i = 0; i < 10; i++
posl2 *= 10
posl2 += min % 10
min /= 10
i = 0; i < 9; i++
posl2 *= 10
posl2 += min % 10
min /= 10
posl2 *= 10
res = a
min_start = sizeof(a) * 8 - min_start - 1
i = 0; i < 5; i++
res = swapBits(res, min_start + i - 9, min_start - i)
printf("%x\n", res)
i = sizeof(res) * 8 - 1; i >= 0; --i
b = (res >> i) & 1
printf("%d", b)
printf("\n")
return 0
Конец

Текст программы на С с комментариями:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

//функция обмена битов в числе
int swapBits(unsigned int n, unsigned int p1, unsigned int p2)
{
    unsigned int bit1 = (n >> p1) & 1;
    unsigned int bit2 = (n >> p2) & 1;
    unsigned int x = (bit1 ^ bit2);
    x = (x << p1) | (x << p2);
    unsigned int result = n ^ x;
    return result;
}

int main()
{
    int a = rand() % 10000000000;
    //int a = 0xDEADBEEF;
    printf("%x\n", a);

    //вывод двоичного представления числа
    for (int i = sizeof(a) * 8 - 1; i >= 0; --i)
    {
        int b = (a >> i) & 1;
        printf("%d", b);
    }
    printf("\n");

    //определение последовательности из 10 битов с наименьшим значением
    int min = 1111111111;
    int min_start = 0;
    for (int i = sizeof(a) * 8 - 1; i >= 9; --i)
    {
        int posl = 0;
        int start = sizeof(a) * 8 - 1 - i;
        for (int j = i; j > i - 10; --j)
        {
            int b = (a >> j) & 1;
            posl *= 10;
            posl += b;
        }
        if (posl < min)
        {
            min = posl;
            min_start = start;
        }
    }
    int posl2 = 0;
    if (((a << min_start) & 1) != 0)
    {
        for (int i = 0; i < 10; i++)
        {
            posl2 *= 10;
            posl2 += min % 10;
            min /= 10;
        }
    }
    else
    {
        for (int i = 0; i < 9; i++)
        {
            posl2 *= 10;
```

```
            posl2 += min % 10;
            min /= 10;
        }
        posl2 *= 10;
    }
    int res = a;

    //min_start – номер бита, с которого начинается наименьшая последовательность
    min_start = sizeof(a) * 8 - min_start - 1;

    //изменение порядка следования битов в этой последовательности на обратный
    for (int i = 0; i < 5; i++)
    {
        res = swapBits(res, min_start + i - 9, min_start - i);
    }
    printf("%x\n", res);

    //вывод двоичного представления получившегося числа
    for (int i = sizeof(res) * 8 - 1; i >= 0; --i)
    {
        int b = (res >> i) & 1;
        printf("%d", b);
    }
    printf("\n");
    return 0;
}
```

Текст программы на ассемблере NASM с комментариями:

```
section .data
section .bss
        num resd 1
        c resb 0
global _start
section .text
output:
        mov ecx, 32
        xor r8, r8
        mov ebx, 1
        shl ebx, 31
        lp1:
        mov esi, eax
        and esi, ebx
        shr esi, 31
        add esi, 48
        mov [edi], esi
        shl eax, 1
        inc edi
        cmp ecx, 32
        jle if
        mov esi, 32
        mov [edi], esi
        inc edi
        inc r8
        if:
        inc r8
        dec ecx
        cmp ecx, 0
        jg lp1
        mov byte [edi], 10
        inc r8
        mov eax, 1
        mov edi, 1
        mov esi, c
        mov rdx, r8
```

```
        syscall
ret
search_min:
        mov ebx, 0x3FF
        mov cl, 0
        mov r8w, 0x3FF
        lp2:
                mov edx, r12d
                and edx, ebx
                shr edx, cl
                shl ebx, 1
                cmp edx, r8d
                jg else
                        mov r8d, edx
                        mov r11b, cl
                else:
                inc cl
                cmp cl, 23
                jl lp2
ret
reverse:
        mov cl, 0
        mov ebx, 1
        mov r10d, 0
        lp3:
                mov r9d, r8d
                and r9d, ebx
                shr r9d, cl
                or r10d, r9d
                shl r10d, 1
                shl ebx, 1
                inc cl
                cmp cl, 9
                jl lp3
                mov r8d, r10d
ret
assembling:
        mov ebx, 0xFFFFFC00
        mov cl, r11b
        rol ebx, cl
        and r12d, ebx
        shl r8d, cl
        or r8d, r12d
ret
_start:
   mov rax, 0
   rdtsc
        mov r12d, eax
        mov edi, c
        call output
        call search_min
        call reverse
        call assembling
        mov eax, r8d
        mov edi, c
        call output
mov eax, 60
xor edi, edi
syscall
```

Дизассемблерный листинг существенных частей программы на C с добавленными комментариями или пояснениями:

```
	.file	"laba.c"
	.text
	.globl	swapBits
	.type	swapBits, @function
swapBits:
.LFB6:
	.cfi_startproc
	endbr64
	pushq	%rbp
	.cfi_def_cfa_offset 16
	.cfi_offset 6, -16
	movq	%rsp, %rbp
	.cfi_def_cfa_register 6
	movl	%edi, -20(%rbp)
	movl	%esi, -24(%rbp)
	movl	%edx, -28(%rbp)
	movl	-24(%rbp), %eax
	movl	-20(%rbp), %edx
	movl	%eax, %ecx
	shrl	%cl, %edx
	movl	%edx, %eax
	andl	$1, %eax
	movl	%eax, -16(%rbp)
	movl	-28(%rbp), %eax
	movl	-20(%rbp), %edx
	movl	%eax, %ecx
	shrl	%cl, %edx
	movl	%edx, %eax
	andl	$1, %eax
	movl	%eax, -12(%rbp)
	movl	-16(%rbp), %eax
	xorl	-12(%rbp), %eax
	movl	%eax, -8(%rbp)
	movl	-24(%rbp), %eax
	movl	-8(%rbp), %edx
	movl	%edx, %esi
	movl	%eax, %ecx
	sall	%cl, %esi
	movl	-28(%rbp), %eax
	movl	-8(%rbp), %edx
	movl	%eax, %ecx
	sall	%cl, %edx
	movl	%edx, %eax
	orl	%esi, %eax
	movl	%eax, -8(%rbp)
	movl	-20(%rbp), %eax
	xorl	-8(%rbp), %eax
	movl	%eax, -4(%rbp)
	movl	-4(%rbp), %eax
	popq	%rbp
	.cfi_def_cfa 7, 8
	ret
	.cfi_endproc
.LFE6:
	.size	swapBits, .-swapBits
	.section	.rodata
.LC0:
	.string	"%x\n"
.LC1:
	.string	"%d"
	.text
	.globl	main
	.type	main, @function
main:
.LFB7:
	.cfi_startproc
```

```
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $80, %rsp
        call    rand@PLT
        movslq  %eax, %rcx
        movabsq $247588007857076055, %rdx
        movq    %rcx, %rax
        imulq   %rdx
        movq    %rdx, %rax
        sarq    $27, %rax
        movq    %rcx, %rdx
        sarq    $63, %rdx
        subq    %rdx, %rax
        movabsq $10000000000, %rdx
        imulq   %rax, %rdx
        movq    %rcx, %rax
        subq    %rdx, %rax
        movl    %eax, -20(%rbp)
        movl    -20(%rbp), %eax
        movl    %eax, %esi
        leaq    .LC0(%rip), %rax
        movq    %rax, %rdi
        movl    $0, %eax
        call    printf@PLT
        movl    $31, -68(%rbp)
        jmp     .L4
.L5:
        movl    -68(%rbp), %eax
        movl    -20(%rbp), %edx
        movl    %eax, %ecx
        sarl    %cl, %edx
        movl    %edx, %eax
        andl    $1, %eax
        movl    %eax, -4(%rbp)
        movl    -4(%rbp), %eax
        movl    %eax, %esi
        leaq    .LC1(%rip), %rax
        movq    %rax, %rdi
        movl    $0, %eax
        call    printf@PLT
        subl    $1, -68(%rbp)
.L4:
        cmpl    $0, -68(%rbp)
        jns     .L5
        movl    $10, %edi
        call    putchar@PLT
        movl    $1111111111, -64(%rbp)
        movl    $0, -60(%rbp)
        movl    $31, -56(%rbp)
        jmp     .L6
.L10:
        movl    $0, -52(%rbp)
        movl    -56(%rbp), %edx
        movl    $31, %eax
        subl    %edx, %eax
        movl    %eax, -12(%rbp)
        movl    -56(%rbp), %eax
        movl    %eax, -48(%rbp)
        jmp     .L7
.L8:
        movl    -48(%rbp), %eax
        movl    -20(%rbp), %edx
```

```
        movl    %eax, %ecx
        sarl    %cl, %edx
        movl    %edx, %eax
        andl    $1, %eax
        movl    %eax, -8(%rbp)
        movl    -52(%rbp), %edx
        movl    %edx, %eax
        sall    $2, %eax
        addl    %edx, %eax
        addl    %eax, %eax
        movl    %eax, -52(%rbp)
        movl    -8(%rbp), %eax
        addl    %eax, -52(%rbp)
        subl    $1, -48(%rbp)
.L7:
        movl    -56(%rbp), %eax
        subl    $9, %eax
        cmpl    %eax, -48(%rbp)
        jge     .L8
        movl    -52(%rbp), %eax
        cmpl    -64(%rbp), %eax
        jge     .L9
        movl    -52(%rbp), %eax
        movl    %eax, -64(%rbp)
        movl    -12(%rbp), %eax
        movl    %eax, -60(%rbp)
.L9:
        subl    $1, -56(%rbp)
.L6:
        cmpl    $8, -56(%rbp)
        jg      .L10
        movl    $0, -44(%rbp)
        movl    -60(%rbp), %eax
        movl    -20(%rbp), %edx
        movl    %eax, %ecx
        sall    %cl, %edx
        movl    %edx, %eax
        andl    $1, %eax
        testl   %eax, %eax
        je      .L11
        movl    $0, -40(%rbp)
        jmp     .L12
.L13:
        movl    -44(%rbp), %edx
        movl    %edx, %eax
        sall    $2, %eax
        addl    %edx, %eax
        addl    %eax, %eax
        movl    %eax, -44(%rbp)
        movl    -64(%rbp), %ecx
        movslq  %ecx, %rax
        imulq   $1717986919, %rax, %rax
        shrq    $32, %rax
        sarl    $2, %eax
        movl    %ecx, %esi
        sarl    $31, %esi
        subl    %esi, %eax
        movl    %eax, %edx
        movl    %edx, %eax
        sall    $2, %eax
        addl    %edx, %eax
        addl    %eax, %eax
        subl    %eax, %ecx
        movl    %ecx, %edx
        addl    %edx, -44(%rbp)
        movl    -64(%rbp), %eax
```

```
        movslq  %eax, %rdx
        imulq   $1717986919, %rdx, %rdx
        shrq    $32, %rdx
        sarl    $2, %edx
        sarl    $31, %eax
        movl    %eax, %ecx
        movl    %edx, %eax
        subl    %ecx, %eax
        movl    %eax, -64(%rbp)
        addl    $1, -40(%rbp)
.L12:
        cmpl    $9, -40(%rbp)
        jle     .L13
        jmp     .L14
.L11:
        movl    $0, -36(%rbp)
        jmp     .L15
.L16:
        movl    -44(%rbp), %edx
        movl    %edx, %eax
        sall    $2, %eax
        addl    %edx, %eax
        addl    %eax, %eax
        movl    %eax, -44(%rbp)
        movl    -64(%rbp), %ecx
        movslq  %ecx, %rax
        imulq   $1717986919, %rax, %rax
        shrq    $32, %rax
        sarl    $2, %eax
        movl    %ecx, %esi
        sarl    $31, %esi
        subl    %esi, %eax
        movl    %eax, %edx
        movl    %edx, %eax
        sall    $2, %eax
        addl    %edx, %eax
        addl    %eax, %eax
        subl    %eax, %ecx
        movl    %ecx, %edx
        addl    %edx, -44(%rbp)
        movl    -64(%rbp), %eax
        movslq  %eax, %rdx
        imulq   $1717986919, %rdx, %rdx
        shrq    $32, %rdx
        sarl    $2, %edx
        sarl    $31, %eax
        movl    %eax, %ecx
        movl    %edx, %eax
        subl    %ecx, %eax
        movl    %eax, -64(%rbp)
        addl    $1, -36(%rbp)
.L15:
        cmpl    $8, -36(%rbp)
        jle     .L16
        movl    -44(%rbp), %edx
        movl    %edx, %eax
        sall    $2, %eax
        addl    %edx, %eax
        addl    %eax, %eax
        movl    %eax, -44(%rbp)
.L14:
        movl    -20(%rbp), %eax
        movl    %eax, -32(%rbp)
        movl    -60(%rbp), %edx
        movl    $31, %eax
        subl    %edx, %eax
```

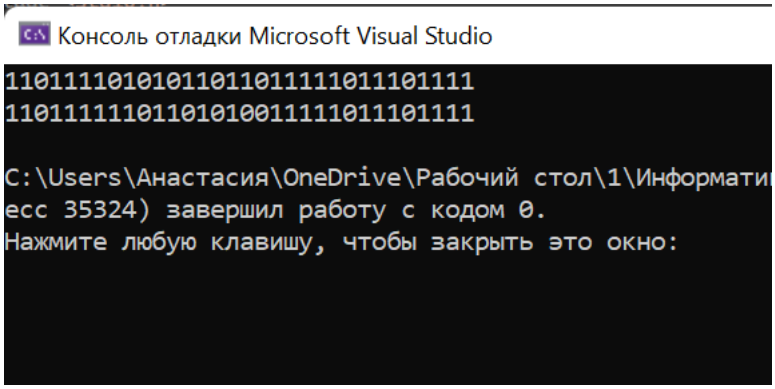
```
        movl    %eax, -60(%rbp)
        movl    $0, -28(%rbp)
        jmp     .L17
.L18:
        movl    -60(%rbp), %eax
        subl    -28(%rbp), %eax
        movl    %eax, %edx
        movl    -60(%rbp), %ecx
        movl    -28(%rbp), %eax
        addl    %ecx, %eax
        subl    $9, %eax
        movl    %eax, %ecx
        movl    -32(%rbp), %eax
        movl    %ecx, %esi
        movl    %eax, %edi
        call    swapBits
        movl    %eax, -32(%rbp)
        addl    $1, -28(%rbp)
.L17:
        cmpl    $4, -28(%rbp)
        jle     .L18
        movl    -32(%rbp), %eax
        movl    %eax, %esi
        leaq    .LC0(%rip), %rax
        movq    %rax, %rdi
        movl    $0, %eax
        call    printf@PLT
        movl    $31, -24(%rbp)
        jmp     .L19
.L20:
        movl    -24(%rbp), %eax
        movl    -32(%rbp), %edx
        movl    %eax, %ecx
        sarl    %cl, %edx
        movl    %edx, %eax
        andl    $1, %eax
        movl    %eax, -16(%rbp)
        movl    -16(%rbp), %eax
        movl    %eax, %esi
        leaq    .LC1(%rip), %rax
        movq    %rax, %rdi
        movl    $0, %eax
        call    printf@PLT
        subl    $1, -24(%rbp)
.L19:
        cmpl    $0, -24(%rbp)
        jns     .L20
        movl    $10, %edi
        call    putchar@PLT
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE7:
        .size   main, .-main
        .ident  "GCC: (Ubuntu 11.2.0-7ubuntu2) 11.2.0"
        .section .note.GNU-stack,"",@progbits
        .section .note.gnu.property,"a"
        .align 8
        .long   1f - 0f
        .long   4f - 1f
        .long   5
0:
        .string "GNU"
1:
```

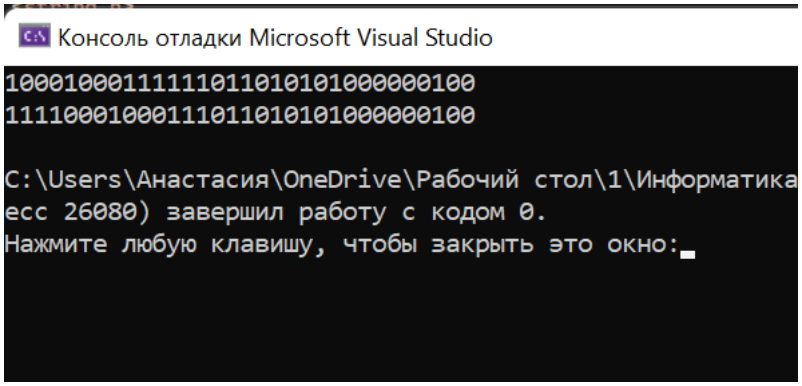
```
        .align 8
        .long   0xc0000002
        .long   3f - 2f
2:
        .long   0x3
3:
        .align 8
4:
```

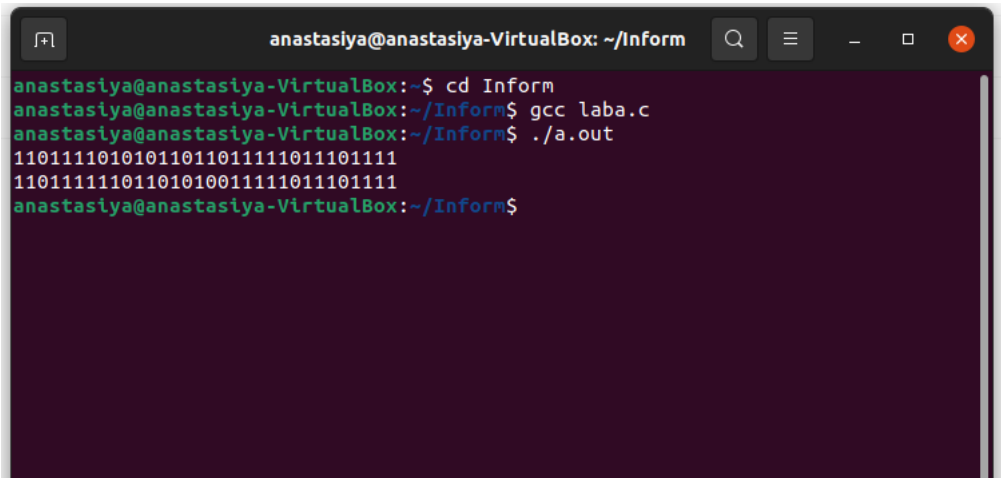
Краткий анализ по результатам сравнения программы на ассемблере и дизассемблированной программы на С:

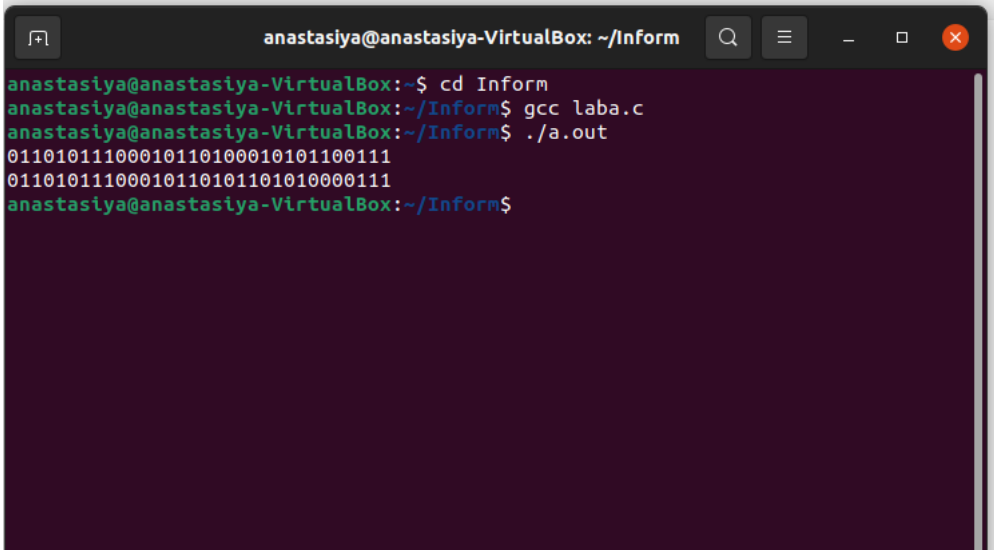
В дизассемблированном листинге присутствуют команды, которые я не использовала. И перечень этих команд намного шире, чем те, которые использовала я. Также в дизассемблированном листинге присутствуют указатели, которых нет в моей программе на nasm. И в моей программе на nasm примерно в полтора раза меньше строк, чем в листинге. Также в листинге присутствуют команды из библиотеки языка Си.

Скриншоты прогонов программ на различных исходных данных:






anastasiya@anastasiya-VirtualBox: ~/Inform
anastasiya@anastasiya-VirtualBox:~$ cd Inform
anastasiya@anastasiya-VirtualBox:~/Inform$ gcc laba.c
anastasiya@anastasiya-VirtualBox:~/Inform$ ./a.out
11011110101011011011111011101111
11011111101101010011111011101111
anastasiya@anastasiya-VirtualBox:~/Inform$


anastasiya@anastasiya-VirtualBox: ~/Inform
anastasiya@anastasiya-VirtualBox:~$ cd Inform
anastasiya@anastasiya-VirtualBox:~/Inform$ gcc laba.c
anastasiya@anastasiya-VirtualBox:~/Inform$ ./a.out
01101011100010110100010101100111
01101011100010110101101010000111
anastasiya@anastasiya-VirtualBox:~/Inform$