

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

дисциплина: *Архитектура компьютера*

Студент: Сухова Арина

Группа: НПИбд-02-25

МОСКВА

2025 г.

1. Цель работы

Освоить условный и безусловный переход. Ознакомиться с назначением и структурой файла листинга.

2. Задание

Написать программы для решения системы выражений.

3. Выполнение лабораторной работы

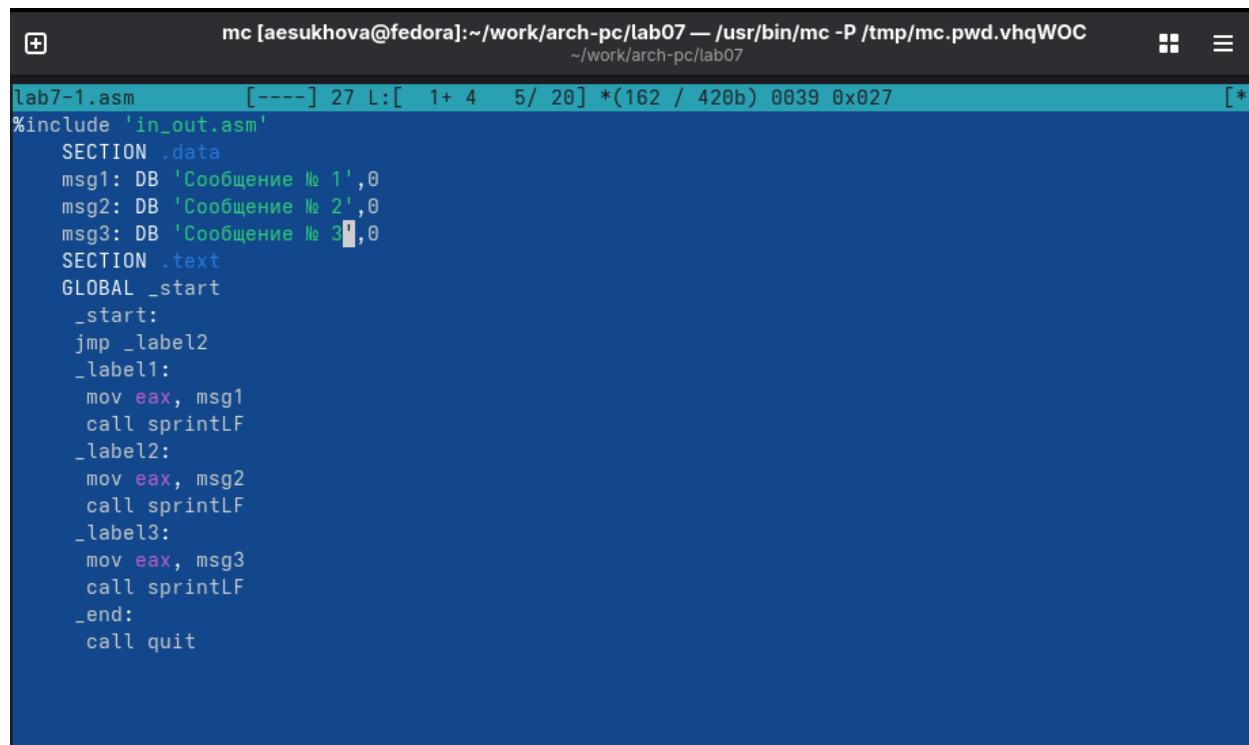
3.1 Реализация переходов в NASM

Создаем каталог для программ ЛБ7 и в нем создаем файл (рис.3.1)

```
aesukhova@fedora:~$ mkdir ~/work/arch-pc/lab07
aesukhova@fedora:~$ cd ~/work/arch-pc/lab07
aesukhova@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рис. 3.1 Создаем каталог с помощью команды mkdir и файл с помощью команды touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.1 (рис. 3.2)



```
mc [aesukhova@fedora]:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.pwd.vhqWOC
~/work/arch-pc/lab07

lab7-1.asm      [---] 27 L:[ 1+ 4   5/ 20 ] *(162 / 420b) 0039 0x027
[*
%include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
    jmp _label2
_label1:
    mov eax, msg1
    call sprintLF
_label2:
    mov eax, msg2
    call sprintLF
_label3:
    mov eax, msg3
    call sprintLF
_end:
    call quit
```

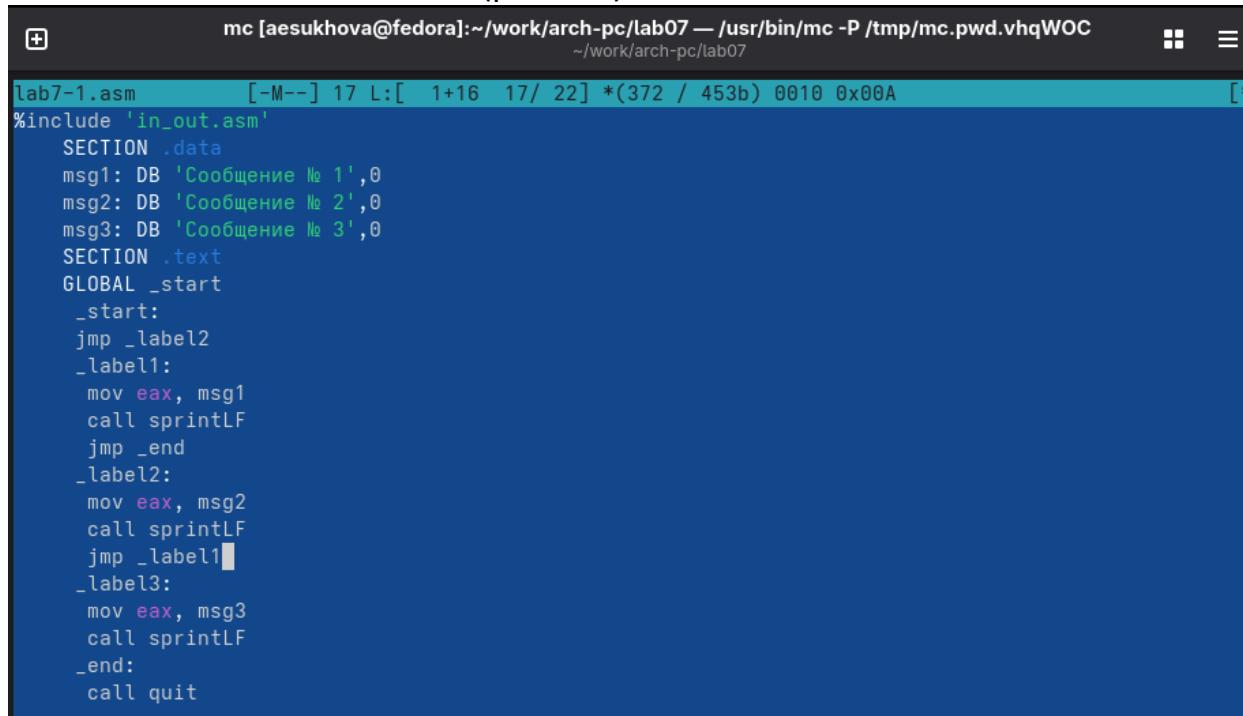
Рис. 3.2 Заполняем файл

Создаем исполняемый файл и запускаем его (рис. 3.3)

```
aesukhova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aesukhova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
aesukhova@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
aesukhova@fedora:~/work/arch-pc/lab07$
```

Рис. 3.3 Запускаем файл и смотрим на его работу

Снова открываем файл для редактирования и изменяем его в соответствии с листингом 7.2 (рис. 3.4)



The screenshot shows the assembly code for `lab7-1.asm` in a terminal window. The code defines three messages in the `.data` section and prints them in the `.text` section using `sprintLF`. The assembly code is as follows:

```
lab7-1.asm      [-M--] 17 L:[ 1+16 17/ 22] *(372 / 453b) 0010 0x00A
%include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
    jmp _label2
_label1:
    mov eax, msg1
    call sprintLF
    jmp _end
_label2:
    mov eax, msg2
    call sprintLF
    jmp _label1
_label3:
    mov eax, msg3
    call sprintLF
_end:
    call quit
```

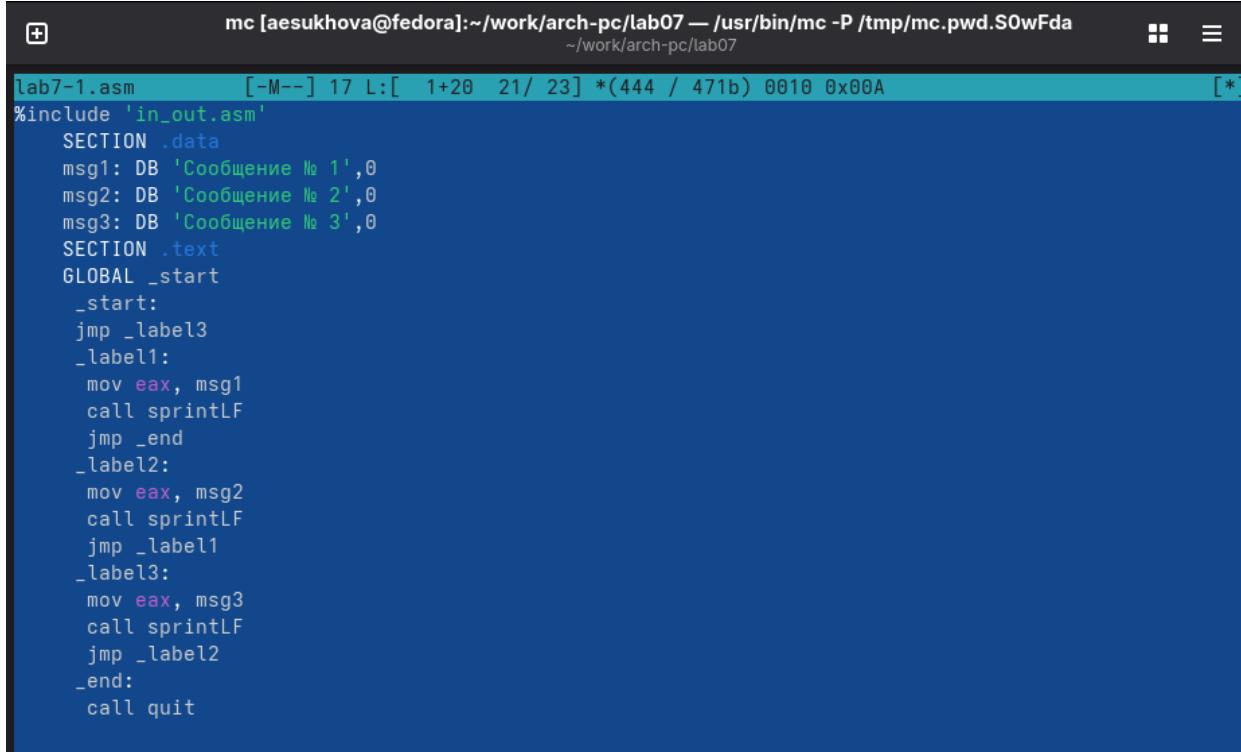
Рис. 3.4 Изменяю файл

Создаем исполняемый файл и запускаем его (рис. 3.5).

```
aesukhova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aesukhova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
aesukhova@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
aesukhova@fedora:~/work/arch-pc/lab07$
```

Рис. 3.5 Запускаем файл и смотрим на его работу

Снова открываем файл для редактирования и изменяем его чтобы произошел данный вывод (рис. 3.6)



```
mc [aesukhova@fedora]:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.pwd.S0wFda
~/work/arch-pc/lab07

lab7-1.asm      [-M--] 17 L:[ 1+20 21/ 23] *(444 / 471b) 0010 0x00A
[*]

%include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
_start:
jmp _label3
_label1:
    mov eax, msg1
    call sprintLF
    jmp _end
_label2:
    mov eax, msg2
    call sprintLF
    jmp _label1
_label3:
    mov eax, msg3
    call sprintLF
    jmp _label2
_end:
    call quit
```

Рис. 3.6 Редактируем файл

Создаем исполняемый файл и запускаем его (рис. 3.7).

```
aesukhova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aesukhova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
aesukhova@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
aesukhova@fedora:~/work/arch-pc/lab07$
```

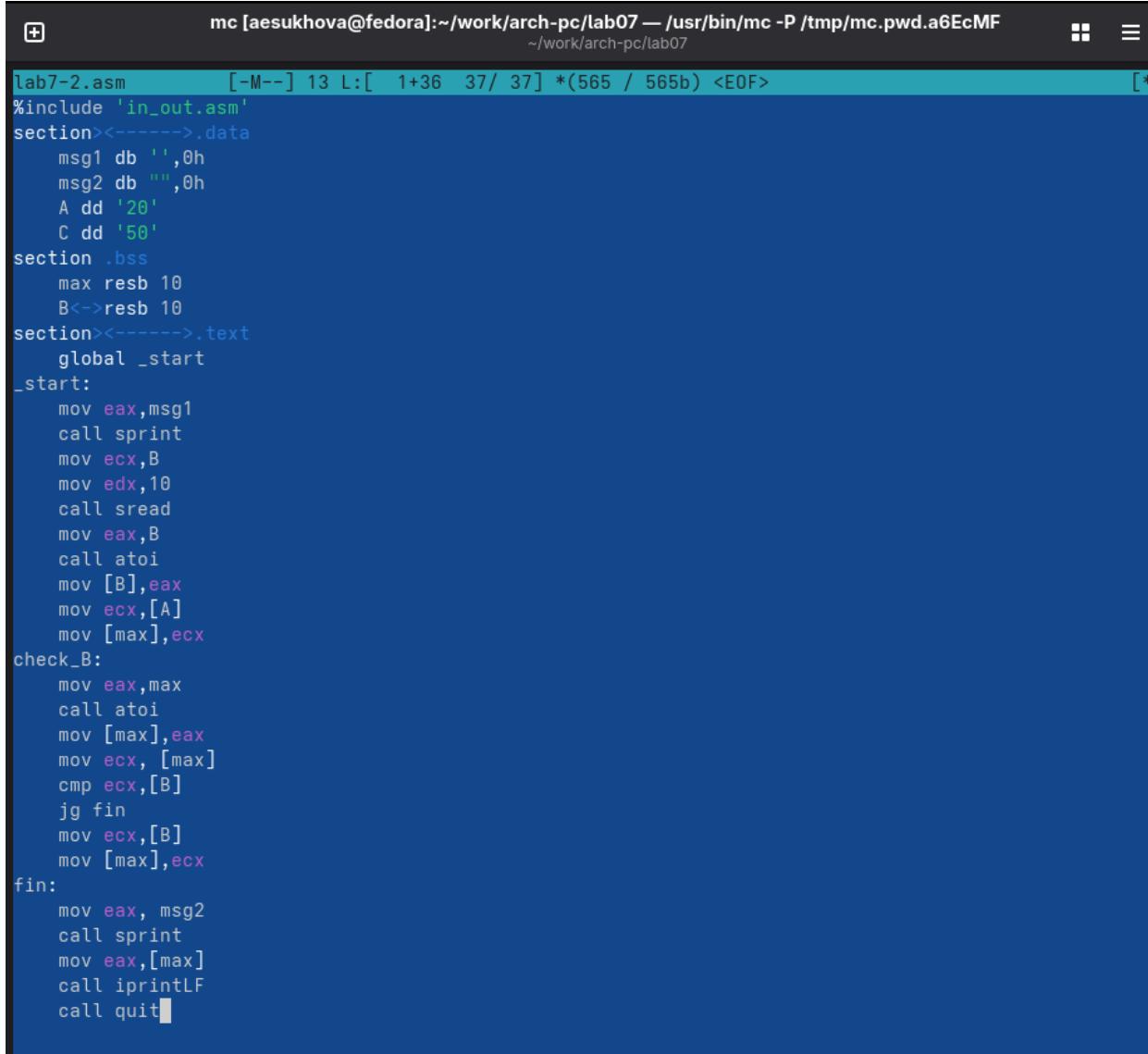
Рис. 3.7 Проверяем, сошелся ли наш вывод с данным в условии выводом

Создаем новый файл (рис. 3.8)

```
aesukhova@fedora:~/work/arch-pc/lab07$ touch lab7-2.asm
aesukhova@fedora:~/work/arch-pc/lab07$
```

Рис. 3.8 Создаем файл командой touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.3 (рис. 3.9)



```
mc [aesukhova@fedora]:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.pwd.a6EcMF ~/work/arch-pc/lab07

lab7-2.asm      [-M--] 13 L:[ 1+36 37/ 37] *(565 / 565b) <EOF>
%include 'in_out.asm'
section><----->.data
    msg1 db ' ',0h
    msg2 db "",0h
    A dd '20'
    C dd '50'
section .bss
    max resb 10
    B<->resb 10
section><----->.text
    global _start
_start:
    mov eax,msg1
    call sprint
    mov ecx,B
    mov edx,10
    call sread
    mov eax,B
    call atoi
    mov [B],eax
    mov ecx,[A]
    mov [max],ecx
check_B:
    mov eax,max
    call atoi
    mov [max],eax
    mov ecx, [max]
    cmp ecx,[B]
    jg fin
    mov ecx,[B]
    mov [max],ecx
fin:
    mov eax, msg2
    call sprint
    mov eax,[max]
    call iprintLF
    call quit
```

Рис. 3.9 Заполняем файл

Создаем исполняемый файл и проверяем его работу, вводя разные значения В (рис. 3.10).

```
aesukhova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aesukhova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
aesukhova@fedora:~/work/arch-pc/lab07$ ./lab7-2
5
20
aesukhova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aesukhova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
aesukhova@fedora:~/work/arch-pc/lab07$ ./lab7-2
10
20
aesukhova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aesukhova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
aesukhova@fedora:~/work/arch-pc/lab07$ ./lab7-2
1
20
aesukhova@fedora:~/work/arch-pc/lab07$
```

Рис. 3.10 Смотрим на работу программ

3.2 Изучение структуры файла листинга

Создаем файл листинга для программы lab7-2.asm (рис. 3.11)

```
aesukhova@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm  
aesukhova@fedora:~/work/arch-pc/lab07$
```

Рис.3.11 Создаем файл листинга

Открываем файл листинга с помощью команды mcedit и изучаем его (рис. 3.12)

```
mc [aesukhova@fedora:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.pwd.dv4neF ~/work/arch-pc/lab07]  
/home/aesukhova/work/arch-pc/lab07/lab7-2.lst 2899/12620  
1 %include 'in_out.asm'  
1 <1> ;----- slen -----  
2 <1> ; Функция вычисления длины сообщения  
3 <1> slen:  
4 00000000 53 <1> push    ebx  
5 00000001 89C3 <1> mov     ebx,  eax  
6 <1>  
7 <1> nextchar:  
8 00000003 803800 <1> cmp     byte [eax], 0  
9 00000006 7403 <1> jz      finished  
10 00000008 40 <1> inc    eax  
11 00000009 EBF8 <1> jmp    nextchar  
12 <1>  
13 <1> finished:  
14 0000000B 29D8 <1> sub    eax,  ebx  
15 0000000D 5B <1> pop    ebx  
16 0000000E C3 <1> ret  
17 <1>  
18 <1>  
19 <1> ;----- sprint -----  
20 <1> ; Функция печати сообщения  
21 <1> ; входные данные: mov eax,<message>  
22 <1> sprint:  
23 0000000F 52 <1> push   edx  
24 00000010 51 <1> push   ecx  
25 00000011 53 <1> push   ebx  
26 00000012 50 <1> push   eax  
27 00000013 E8E8FFFFF <1> call   slen  
28 <1>  
29 00000018 89C2 <1> mov    edx,  eax  
30 0000001A 58 <1> pop    eax  
31 <1>  
32 0000001B 89C1 <1> mov    ecx,  eax  
33 0000001D BB01000000 <1> mov    ebx,  1  
34 00000022 B804000000 <1> mov    eax,  4  
35 00000027 CD80 <1> int    80h  
36 <1>  
37 00000029 5B <1> pop    ebx  
38 0000002A 59 <1> pop    ecx  
39 0000002B 5A <1> pop    edx  
40 0000002C C3 <1> ret  
41 <1>  
42 <1>  
43 <1> ;----- sprintLF -----  
44 <1> ; Функция печати сообщения с переводом строки  
45 <1> ; входные данные: mov eax,<message>
```

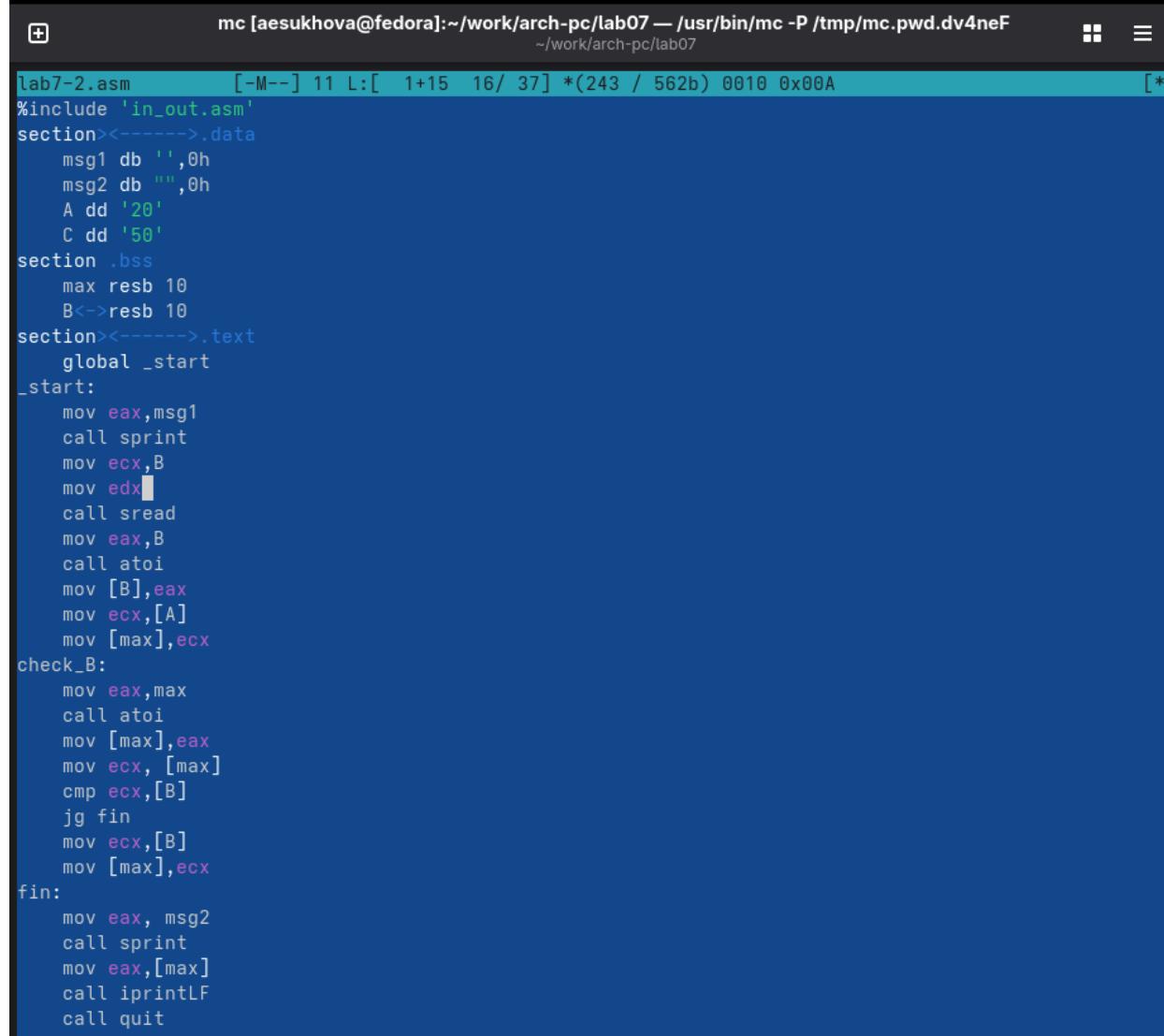
Рис. 3.12 Изучаем файл

Строка 33: 0000001D-адрес в сегменте кода,BB01000000-машинный код,мов ebx,1-присвоение переменной ecx значения 1.

Строка 34: 00000022-адрес в сегменте кода,B804000000-машинный код,мов eax,4-присвоение переменной eax значения 4.

Строка 35 00000027-адрес в сегменте кода, CD80-машинный код, int 80h-вызов ядра.

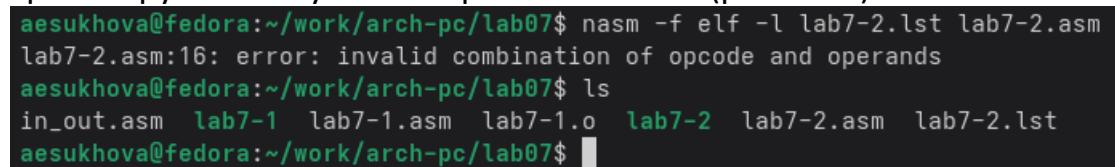
Открываем файл и удаляем один операндум (рис.3.13).



```
lab7-2.asm      [-M--] 11 L:[ 1+15 16/ 37] *(243 / 562b) 0010 0x00A
%include 'in_out.asm'
section><----->.data
    msg1 db ' ',0h
    msg2 db "",0h
    A dd '20'
    C dd '50'
section .bss
    max resb 10
    B<->resb 10
section><----->.text
    global _start
_start:
    mov eax, msg1
    call sprint
    mov ecx,B
    mov edx[ ]
    call sread
    mov eax,B
    call atoi
    mov [B],eax
    mov ecx,[A]
    mov [max],ecx
check_B:
    mov eax,max
    call atoi
    mov [max],eax
    mov ecx, [max]
    cmp ecx,[B]
    jg fin
    mov ecx,[B]
    mov [max],ecx
fin:
    mov eax, msg2
    call sprint
    mov eax,[max]
    call iprintLF
    call quit
```

Рис. 3.13 Удаляем операндум из файла

Транслируем с получением файла листинга (рис. 3.14).



```
aesukhova@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:16: error: invalid combination of opcode and operands
aesukhova@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2  lab7-2.asm  lab7-2.lst
aesukhova@fedora:~/work/arch-pc/lab07$
```

Рис. 3.14 Транслируем файл

При трансляции файла выдается ошибка, но создаются исполнительный файл lab7-2 и lab7-2.lst

Снова открываем файл листинга и изучаем его (рис. 3.15).

```
aesukhova@fedora:~/work/arch-pc/lab07 — mcedit lab7-2.lst
~/work/arch-pc/lab07

lab7-2.lst      [---]  0 L:[ 1+ 0   1/209] *(0  /12708b) 0032 0x020
1           %include 'in_out.asm'
1           <1> ;-----  slen -----
2           <1> ; Функция вычисления длины сообщения
3           <1> slen:.....
4 00000000 53
5 00000001 89C3
6
7           <1> nextchar:.....
8 00000003 803800
9 00000006 7403
10 00000008 40
11 00000009 EBF8
12           <1>.....
13           <1> finished:
14 0000000B 29D8
15 0000000D 5B
16 0000000E C3
17           <1>.
18           <1>.
19           <1> ;----- sprint -----
20           <1> ; Функция печати сообщения
21           <1> ; входные данные: mov eax,<message>
22           <1> sprint:
23 0000000F 52
24 00000010 51
25 00000011 53
26 00000012 50
27 00000013 E8E8FFFF
28           <1> call slen
29 00000018 89C2
30 0000001A 58
31           <1>.....
32 0000001B 89C1
33 0000001D B801000000
34 00000022 B804000000
35 00000027 CD80
36           <1>.
37 00000029 5B
38 0000002A 59
39 0000002B 5A
40 0000002C C3
41           <1>.
42           <1>.
43           <1> ;----- sprintLF -----
44           <1> ; Функция печати сообщения с переводом строки
45           <1> ; входные данные: mov eax,<message>
```

Рис. 3.15 Анализируем файл с ошибкой

Создается ТОЛЬКО файл листинга `lab7-2.lst`. Файл объектного кода lab7-2.o НЕ создается, потому что ассемблер обнаружил ошибку и остановил компиляцию.

При наличии синтаксических ошибок в исходном коде:

- Объектный файл (.o) не создается - компиляция прерывается
- Листинг создается, но содержит сообщения об ошибках - это помогает локализовать проблему
- Ассемблер указывает точное место и характер ошибки - в данном случае "ожидалась инструкция" из-за неполной команды

3.3 Задание для самостоятельной работы ВАРИАНТ-5

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c . Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу.

Создаем новый файл (рис. 3.16)

```
aesukhova@fedora:~/work/arch-pc/lab07$ touch lab7-3.1.asm
aesukhova@fedora:~/work/arch-pc/lab07$ mcedit lab7-3.1.asm
```

Рис. 3.16 Создаем файл командой touch и редактируем в Midnight Commander

Открываем его и пишем программу, которая выберет наименьшее число из трех (2 числа уже в программе, третье вводится с консоли)(рис. 3.17)

```

aesukhova@fedora:~/work/arch-pc/lab07 — mcedit lab7-3.1.asm
~/work/arch-pc/lab07

lab7-3.1.asm      [---] 13 L:[ 1+39  40/ 40] *(845 / 845b) <EOF>
[*]

%include 'in_out.asm'
section .data
    msg1 db 'Введите В: ',0h
    msg2 db "Наименьшее число: ",0h
section .bss
    B resb 10
section .text
    global _start
_start:
    ; Вывод приглашения
    mov eax, msg1
    call sprint
    ...
    ; Ввод В
    mov ecx, B
    mov edx, 10
    call sread
    ...
    ; Преобразование В в число
    mov eax, B
    call atoi
    mov ebx, eax      ; ebx = В
    ...
    ; Сравниваем с А=54
    cmp ebx, 54
    jl compare_with_c
    mov ebx, 54      ; если 54 меньше, то ebx = 54
    ...
compare_with_c:
    ; Сравниваем с С=87
    cmp ebx, 87
    jl output_result
    mov ebx, 87      ; если 87 меньше, то ebx = 87
    ...
output_result:
    mov eax, msg2
    call sprint
    mov eax, ebx
    call iprintLF
    call quit

```

Рис. 3.17 Прописываем программу

Транслируем файл и смотрим на работу программы (рис. 3.18)

```

aesukhova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.1.asm
aesukhova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3.1 lab7-3.1.o
aesukhova@fedora:~/work/arch-pc/lab07$ ./lab7-3.1
Введите В: 62
Наименьшее число: 54

```

Рис. 3.18 Смотрим на работу выполненного файла (всё верно)

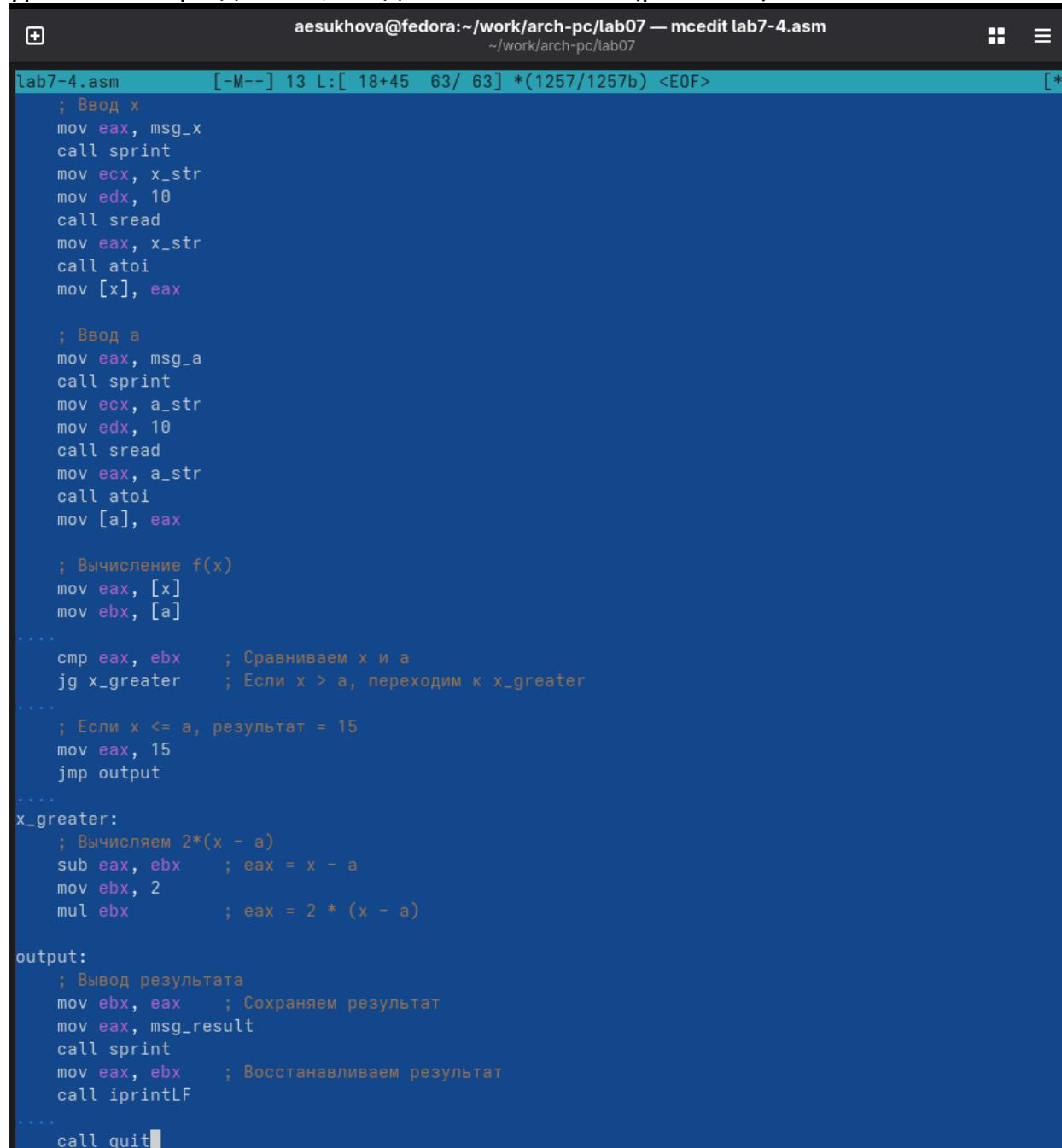
2. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6.

Создаем новый файл (рис. 3.19).

```
aesukhova@fedora:~/work/arch-pc/lab07$ touch lab7-4.asm
```

Рис. 3.19 Создаем файл командой touch

Открываем его и пишем программу, которая решит систему уравнений при данных, введенных в консоль (рис. 3.20)



```
aesukhova@fedora:~/work/arch-pc/lab07 — mcedit lab7-4.asm
~/work/arch-pc/lab07

lab7-4.asm      [-M--] 13 L:[ 18+45  63/ 63] *(1257/1257b) <EOF>
[*]

; Ввод x
mov eax, msg_x
call sprint
mov ecx, x_str
mov edx, 10
call sread
mov eax, x_str
call atoi
mov [x], eax

; Ввод a
mov eax, msg_a
call sprint
mov ecx, a_str
mov edx, 10
call sread
mov eax, a_str
call atoi
mov [a], eax

; Вычисление f(x)
mov eax, [x]
mov ebx, [a]

.....
cmp eax, ebx      ; Сравниваем x и a
jg x_greater      ; Если x > a, переходим к x_greater

.....
; Если x <= a, результат = 15
mov eax, 15
jmp output

.....
x_greater:
; Вычисляем 2*(x - a)
sub eax, ebx      ; eax = x - a
mov ebx, 2
mul ebx          ; eax = 2 * (x - a)

output:
; Вывод результата
mov ebx, eax      ; Сохраняем результат
mov eax, msg_result
call sprint
mov eax, ebx      ; Восстанавливаем результат
call iprintLF
.....
call quit
```

Рис. 3.20 Прописываем программу

Транслируем файл и проверяем его работу при $x = 1$ и $a = 2$ (рис. 3.21).

```
aesukhova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
aesukhova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
aesukhova@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 1
Введите a: 2
Результат f(x): 15
```

Рис 3.21 Проверяем работу программы (работает верно)

Транслируем файл и проверяем его работу при $x = 2$ и $a = 1$ (рис 3.22)

```
aesukhova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
aesukhova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
aesukhova@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 2
Введите a: 1
Результат f(x): 2
```

Рис. 3.22 Проверяем работу программы (работает верно)

4. Выводы

Мы научились решать программы с использованием циклов и обработкой аргументов командной строки.