

Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Хрисанова Ксения Олеговна

Содержание

1	Цель работы	1
2	Задание.....	1
3	Теоретическое введение	1
4	Выполнение лабораторной работы.....	2
4.1	Реализация переходов в NASM	2
4.2	Изучение структуры файла листинга.....	5
4.3	Задания для самостоятельной работы	7
5	Выводы	11
	Список литературы	12

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлов листинга
3. Выполнение заданий для самостоятельной работы

3 Теоретическое введение

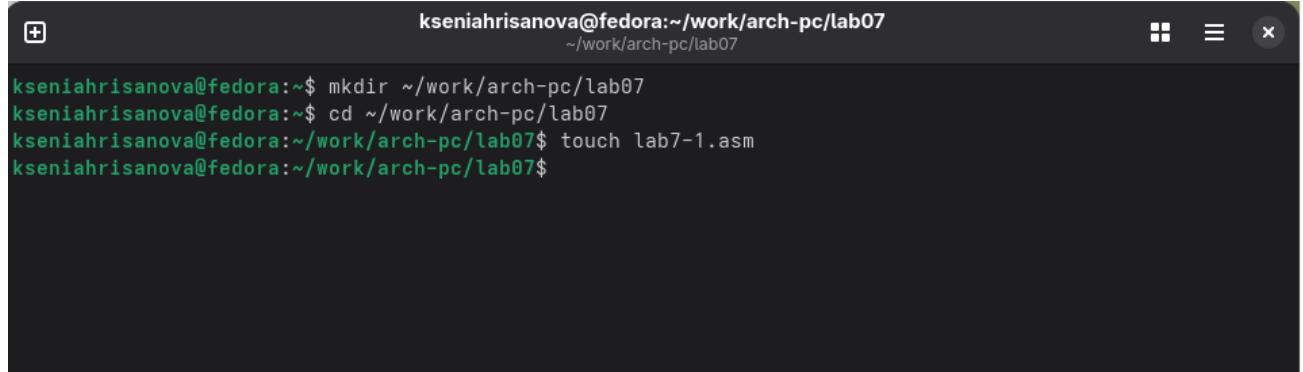
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы №7 (рис. 1).

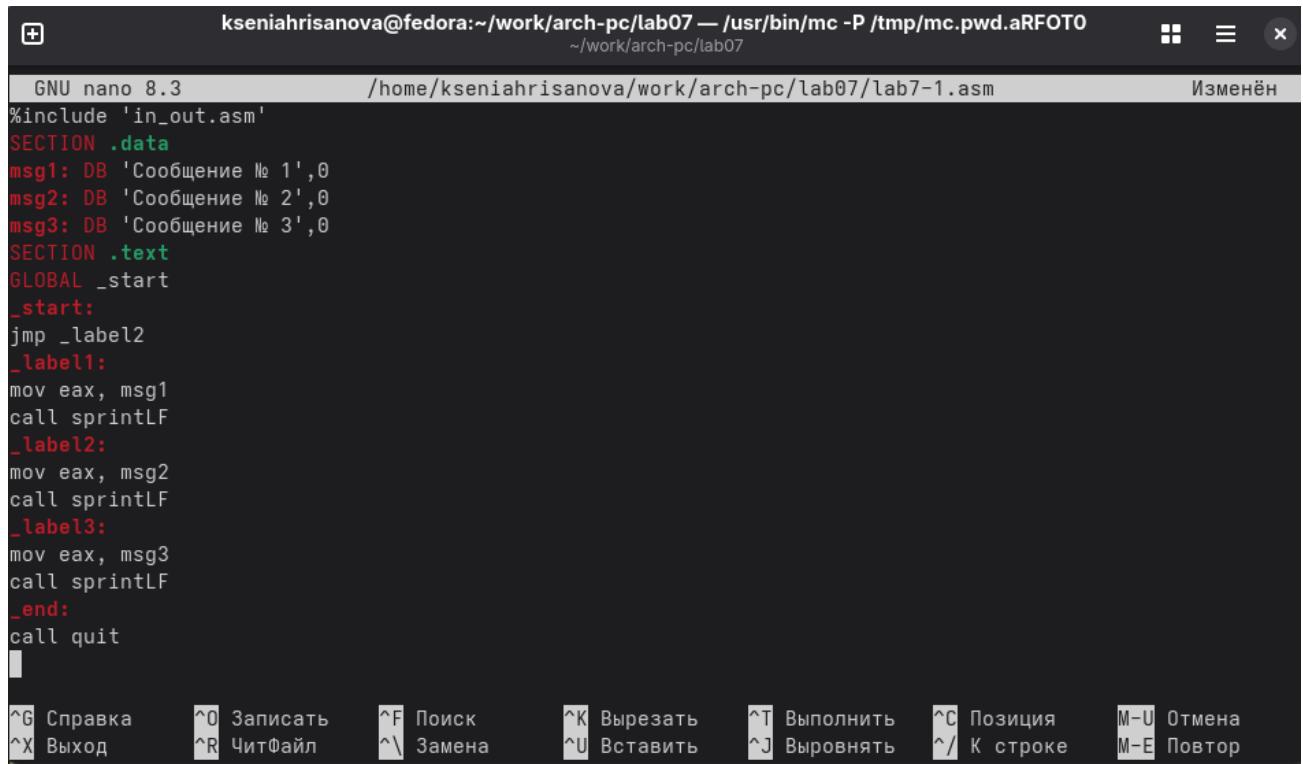


```
kseniahrisanova@fedora:~/work/arch-pc/lab07
~/work/arch-pc/lab07

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

Рис. 1: Создание каталога и файла для программы

Копирую код из листинга в файл будущей программы. (рис. 2).



```
GNU nano 8.3          /home/kseniahrisanova/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.pwd.aRFOTO      Изменён
kseniahrisanova@fedora:~/work/arch-pc/lab07

%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
[

^G Справка    ^O Записать    ^F Поиск    ^K Вырезать    ^T Выполнить    ^C Позиция    M-U Отмена
^X Выход    ^R ЧитФайл    ^\ Замена    ^U Вставить    ^J Выровнять    ^/ К строке    M-E Повтор
```

Рис. 2: Сохранение программы

При запуске программы я убедилась в том, что безусловный переход изменяет порядок выполнения инструкций (рис. 3).

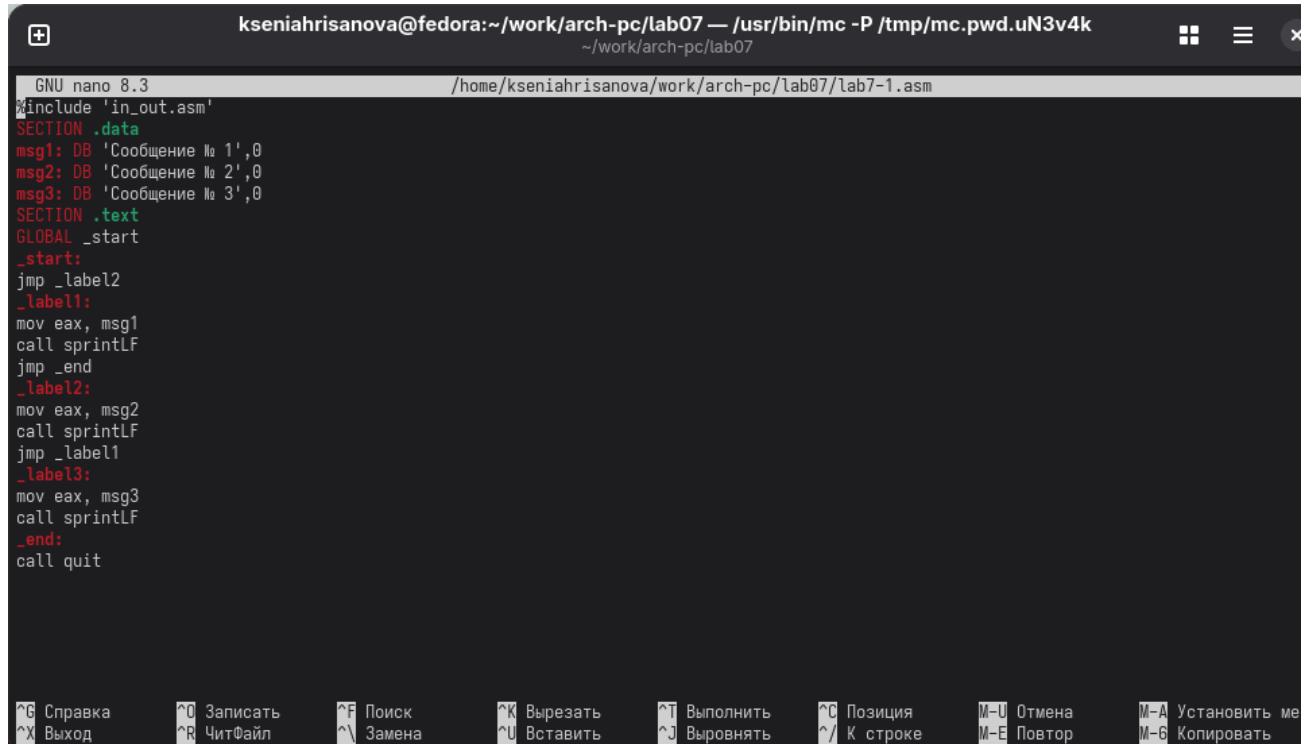
```

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

```

Рис. 3: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения функций (рис. 4).



```

GNU nano 8.3                               /home/kseniahrisanova/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.pwd.uN3v4k
+  kseniahrisanova@fedora:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.pwd.uN3v4k
~ /work/arch-pc/lab07

SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
_label1:
    jmp _label2
_label2:
    mov eax, msg1
    call sprintLF
    jmp _end
_label3:
    mov eax, msg2
    call sprintLF
    jmp _label1
_label4:
    mov eax, msg3
    call sprintLF
_end:
    call quit

^G Справка      ^O Записать      ^F Поиск      ^K Вырезать      ^T Выполнить      ^C Позиция      M-U Отмена      M-A Установить ме
^X Выход      ^R Читать файл      ^\ Замена      ^U Вставить      ^J Выровнять      ^/ К строке      M-E Повтор      M-G Копировать

```

Рис. 4: Изменение программы

Запускаю программу и проверяю, что примененные изменения верны (рис. 5).

```

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

```

Рис. 5: Запуск измененной программы

Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. 6).

The screenshot shows a terminal window titled "kseniahrisanova@fedora:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.pwd.kBjcYn". The file path is ~/work/arch-pc/lab07/lab7-1.asm. The code in the editor is:

```
GNU nano 8.3
#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 _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
```

The status bar at the bottom shows keyboard shortcuts for various operations like Help (^G), Save (^O), Find (^F), Cut (^K), Paste (^U), Run (^T), Position (^C), Undo (^M-U), Redo (^M-E), and Copy (^M-A).

Рис. 6: Изменение программы

Программа работает корректно (рис. 7).

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

Рис. 7: Проверка изменений

Создаю новый рабочий файл и вставляю в него код из листинга 7.3 (рис. 8).

The screenshot shows a terminal window with the following content:

```
GNU nano 8.3
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",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
    cmp ecx,[C]
    jg check_B
    mov ecx,[C]
    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

```

At the bottom of the terminal window, there is a menu bar with Russian labels and keyboard shortcuts:

- Справка (F1)
- Записать (Ctrl+O)
- Поиск (Ctrl+F)
- Вырезать (Ctrl+K)
- Выполнить (Ctrl+T)
- Позиция (Ctrl+C)
- Отмена (M-U)
- Повтор (M-E)
- Установить скобку (M-L)
- На скобку (M-B)
- Предыдущий (M-F)
- Выход (Alt+F4)
- ЧитФайл (Ctrl+R)
- Замена (Ctrl+H)
- Вставить (Ctrl+U)
- Выровнять (Ctrl+J)
- К строке (Ctrl+/)
- Повтор (M-E)
- Копировать (M-B)
- Обр. поиск (M-V)
- Следующий (M-F)

Рис. 8: Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными (рис. 9).

The screenshot shows a terminal window with the following command-line session:

```
kseniahrisanova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
kseniahrisanova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
kseniahrisanova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 20
Наибольшее число: 50
kseniahrisanova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
kseniahrisanova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 70
Наибольшее число: 70
kseniahrisanova@fedora:~/work/arch-pc/lab07$
```

рис. 9: Проверка программы из листинга

4.2 Изучение структуры файла листинга

Создаю файл листинга с помощью флага -l команды nasm и открываю его с помощью текстового редактора mousepad (рис. 10).

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

lab7-2.lst      [---] 0 L:[ 1+ 0  1/217] *(0  /12937b) 0032 0x020
1           %include 'in_out.asm'
1           ;----- slen -----
2           ; Функция вычисления длины сообщения
3           slen:
4 00000000 53          push    ebx
5 00000001 89C3        mov     ebx, eax
6           .....
7           nextchar:
8 00000003 803800      cmp     byte [eax], 0
9 00000006 7403        jz      finished
10 00000008 40         inc     eax
11 00000009 EBF8        jmp     nextchar
12           .....
13           finished:
14 0000000B 29D8        sub     eax, ebx
15 0000000D 5B         pop     ebx
16 0000000E C3         ret
17           .....
18           .....
19           ;----- sprint -----
20           ; Функция печати сообщения
21           ; входные данные: mov eax,<message>
22           sprint:
23 0000000F 52          push    edx
24 00000010 51          push    ecx
25 00000011 53          push    ebx
26 00000012 50          push    eax
27 00000013 E8E8FFFFFF  call    slen
28           .....
29 00000018 89C2        mov     edx, eax
30 0000001A 58          pop     eax
31           .....
32 0000001B 89C1        mov     ecx, eax
33 0000001D B801000000  mov     ebx, 1
34 00000022 B804000000  mov     eax, 4
35 00000027 CD80        int    80h
36           .....
37 00000029 5B          pop     ebx
38 0000002A 59          pop     ecx
39 0000002B 5A          pop     edx
40 0000002C C3         ret
41           .....
42           .....
43           ;----- sprintLF -----
44           ; Функция печати сообщения с переводом строки
```

Рис. 10: Проверка файла листинга

Файл листинга NASM представляет собой расширенный вариант исходного текста программы, в котором для каждой строки указаны номер строки, смещение машинного кода, байты сгенерированных инструкций и исходная команда. Листинг позволяет анализировать соответствие команд ассемблера и результата их трансляции.

Удаляю один operand из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. 11).

```
GNU nano 8.3
%include 'in_out.asm'
section .data
msg1 db "Введите В: ",0h
msg2 db "Наибольшее число: ",0h
A dd '28'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
    call sprint
    mov edx,10
    call sread
    mov eax,B
    call atoi
    mov [B],eax
    mov ecx,[A]
    mov [max],ecx
    cmp ecx,[C]
    jg check_B
    mov ecx,[C]
    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 iprintf
    call quit
```

[Записано 39 строк]

Справка ⌘F Выход ⌘Q Записать ⌘F Запись ⌘F Поиск ⌘F Замена ⌘X Вырезать ⌘V Вставить ⌘I Выполнить ⌘C Позиция ⌘K Строке ⌘U Отмена ⌘E Повтор ⌘A Установить мем ⌘M На скобку ⌘B Копировать ⌘V Обр. поиск ⌘B Предыдущий ⌘F Следующий ⌘N Назад ⌘P Вперед ⌘P ПредСлово ⌘N СледСлово

Рис. 11: Удаление операнда из программы

Создаётся только файл листинга lab7-2.lst, объектный файл не формируется из-за ошибки. В листинге у строки с испорченной инструкцией появляется сообщение об ошибке ассемблера (текст еrror: ... после исходной строки).

(рис. 12).

```
kseniahrisanova@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:13: error: invalid combination of opcode and operands
kseniahrisanova@fedora:~/work/arch-pc/lab07$
```

Рис. 12: Просмотр ошибки в файле листинга

4.3 Задания для самостоятельной работы

Возвращаю operand к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 13).

GNU nano 8.3
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '17'
C dd '45'
SECTION .bss
min 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 [min], ecx

cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [min], ecx

check_B:
mov eax, min
call atoi
mov [min], eax

mov ecx, [min]
cmp ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx

fin:
mov eax, msg2
call sprint
mov eax, [min]
call sprintLF
call quit

Справка Выход Записать Читать Помощь Поиск Замена Вырезать Вставить Выполнить Выровнять Позиция Отмена Повтор Установить курсор На скобку Копировать Обр. поиск Предыдущий Следующий Назад Вперед ПредСлово СледСлово

Рис. 13: программа самостоятельной работы

Текст первой программы:

```
%include 'in_out.asm'  
SECTION .data  
msg1 db 'Введите B: ', 0h  
msg2 db 'Наименьшее число: ', 0h  
A dd '17'  
C dd '45'  
SECTION .bss  
min 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 [min], ecx
cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [min], ecx
check_B:
mov eax, min
call atoi
mov [min], eax
mov ecx, [min]
cmp ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx
fin:
mov eax, msg2
call sprint
mov eax, [min]
call iprintLF
```

Проверяю работу первой программы (рис. 14).

```
kseニアhrisanova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
kseニアhrisanova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
kseニアhrisanova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 23
Наименьшее число: 17
kseニアhrisanova@fedora:~/work/arch-pc/lab07$
```

Рис. 14: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатурых переменных a и x (рис. 15).

```
GNU nano 8.3
kseniahrisanova@fedora:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.pwd.jsb1Pp
~/work/arch-pc/lab07

SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
msg_f: DB 'Результат f(x): ', 0

SECTION .bss
x: RESB 80
a: RESB 80

SECTION .text
GLOBAL _start

_start:
    mov eax, msg_x
    call sprint

    mov ecx, x
    mov edx, 80
    call sread

    mov eax, x
    call atoi
    mov edi, eax

    mov eax, msg_a
    call sprint

    mov ecx, a
    mov edx, 80
    call sread

    mov eax, a
    call atoi
    mov esi, eax

    cmp edi, esi
    jl less_than_a

    mov eax, 8
    jmp print_answer

less_than_a:
    mov eax, esi
    shl eax, 1
    sub eax, edi

print_answer:
    mov ebx, eax

    mov eax, msg_f
    call sprint

    Справка Выход Записать Читайфайл Поиск Замена Вырезать Вставить Выполнить Позиция Отмена К Повтор Установить На скобку Предыдущий Следующий Обр. поиск Назад Вперед ПредСлово СледСлово
```

Рис. 15: Вторая программа самостоятельной работы

Текст второй программы:

```
%include 'in_out.asm'

SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
msg_f: DB 'Результат f(x): ', 0

SECTION .bss
x: RESB 80
a: RESB 80

SECTION .text
GLOBAL _start

_start:
    mov eax, msg_x
    call sprint

    mov ecx, x
    mov edx, 80
    call sread

    mov eax, x
    call atoi
    mov edi, eax
```

```
mov eax, msg_a
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax
cmp edi, esi
jl less_than_a
mov eax, 8
jmp print_answer
less_than_a:
mov eax, esi
shl eax, 1
sub eax, edi
print_answer:
mov ebx, eax
mov eax, msg_f
call sprint
```

Транслирую и компоную файл, запускаю и проверяю работу программы для значений а и х, которые даны в задании (рис. 16).

```
kseニアhrisanova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
kseニアhrisanova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
kseニアhrisanova@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 1
Введите значение переменной a: 2
Результат f(x): 3
kseニアhrisanova@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 2
Введите значение переменной a: 1
Результат f(x): 8
kseニアhrisanova@fedora:~/work/arch-pc/lab07$
```

Рис. 16: Проверка работы второй программы

5 Выводы

При выполнении лабораторной работы я изучила команды условных и безусловных переходов, приобрела навыки написания программ с использованием переходов и познакомилась с назначением и структурой файлов листинга.

Список литературы

1. Курс на ТУИС
2. Лабораторная работа №7