

Отчёт по лабораторной работе №7

Выполнил студент НКАбд-02-25

Арина Андреевна Дрекина

Содержание

1 Цель работы	3
2 Порядок выполнения лабораторной работы	4
3 Реализация переходов в NASM	5
4 Изучение структуры файлы листинга.	16
5 Задание для самостоятельной работы.	19
6 Вывод	24

1 Цель работы

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

2 Порядок выполнения лабораторной работы

3 Реализация переходов в NASM

Первым делом я создала каталог для лабораторной работы №7.(Рисунок 3.1)



```
aadrekina@fedora:~/work/arch--pc/lab07$ mkdir ~/work/arch--pc/lab07
aadrekina@fedora:~/work/arch--pc/lab07$ cd ~/work/arch--pc/lab07
aadrekina@fedora:~/work/arch--pc/lab07$
```

Рисунок 3.1: Создание каталога для лабораторной работы.

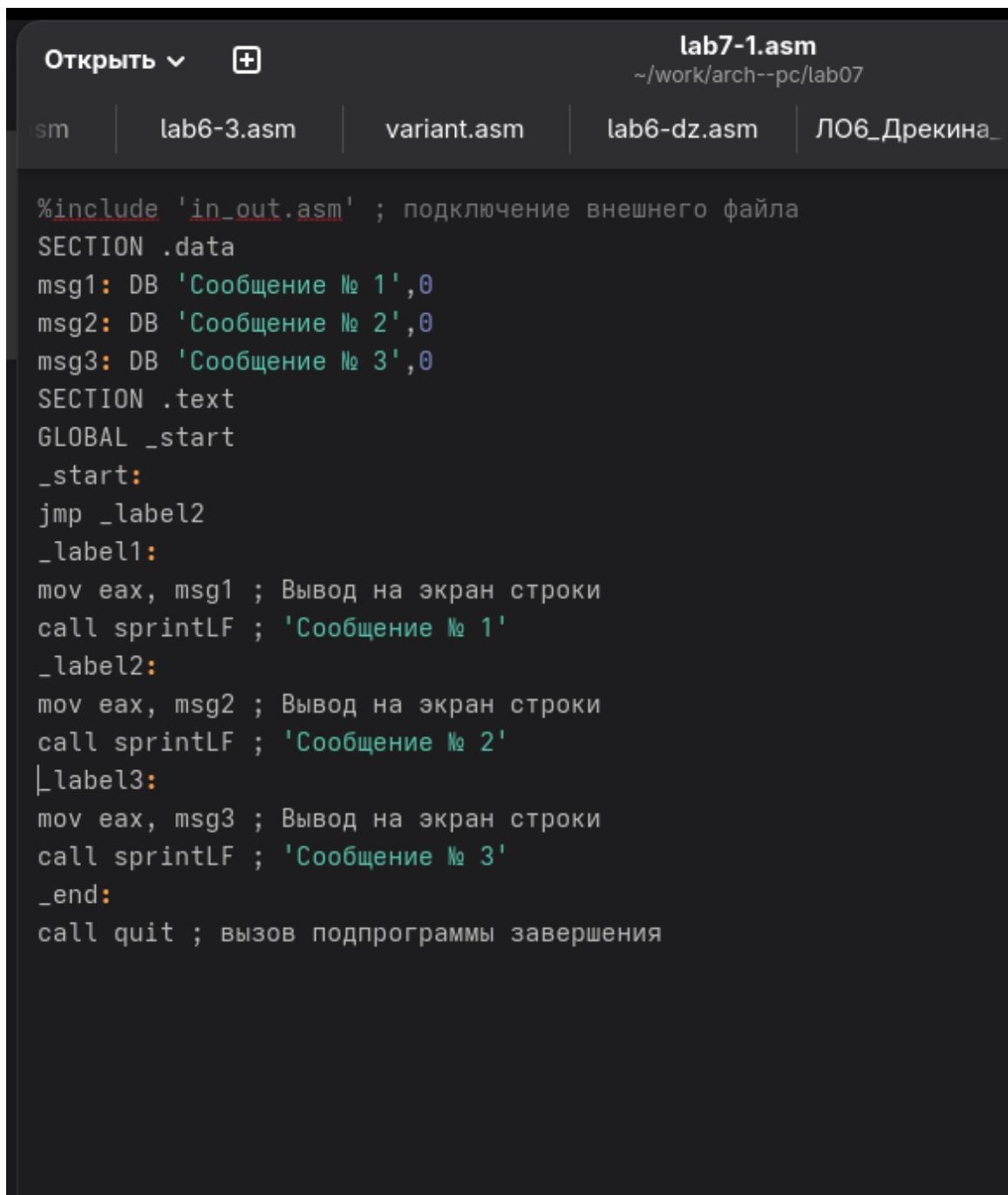
Затем я создала текстовый файл для работы. (Рисунок 3.2)



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

Рисунок 3.2: Создание текстового файла для лабораторной работы.

Затем в созданный файл я ввела текст Листинга 7.1(Рисунок 3.3)



lab7-1.asm
~/work/arch--pc/lab07

Открыть +

asm | lab6-3.asm | variant.asm | lab6-dz.asm | ЛО6_Дрекина_

```
%include 'in_out.asm' ; подключение внешнего файла
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 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рисунок 3.3: Вставка текста в файл.

Листинг 7.1:

```
%include 'in_out.asm' ; подключение внешнего файла

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 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'

_label2:

mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'

_label3:

mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'

_end:

call quit ; вызов подпрограммы завершения
```

Затем я создала исполняемый файл и запустила его. (Рисунок 3.4)

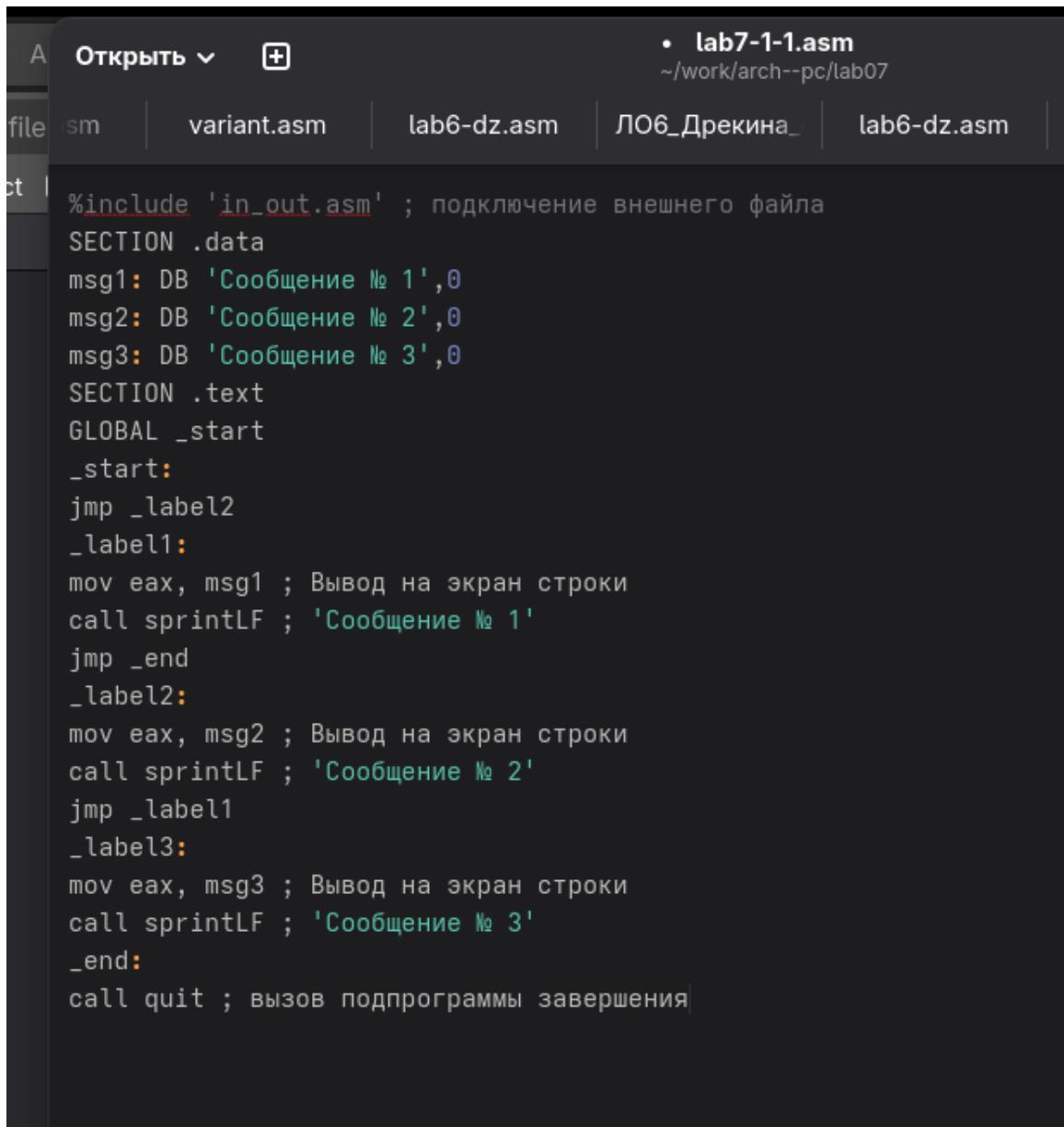
```
aadrekina@fedora:~/work/arch--pc/lab07$ nasm -f elf lab7-1.asm
aadrekina@fedora:~/work/arch--pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
aadrekina@fedora:~/work/arch--pc/lab07$ ./lab7-1
толь
ет вы
воля Сообщение № 2
Сообщение № 3
aadrekina@fedora:~/work/arch--pc/lab07$
```

Рисунок 3.4: Исполняемый файл и запуск.

Результат, который вывел мне терминал совпадает с тем, что написано в лекции,

значит я сделала все правильно.

Затем я изменила текст программы так, чтобы сначала выводилось „Сообщение № 2“, потом „Сообщение № 1“ и работа завершалась. Текст программы я взяла из Листинга 7.2.(Рисунок 3.5)



The screenshot shows a Windows Notepad window with the following assembly code:

```
%include 'in_out.asm' ; подключение внешнего файла
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 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рисунок 3.5: Изменения в программе.

Затем я сделала файл исполняемым и проверила его работу. (Рисунок 3.6)

```

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

```

Рисунок 3.6: Проверка работы команды.

На выводе вышла правильная последовательность, значит я внесла правильные изменения в программу.

Листинг 7.2:

```
%include 'in_out.asm' ; подключение внешнего файла

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 ; Вывод на экран строки
    call sprintLF ; 'Сообщение № 1'

    jmp _end

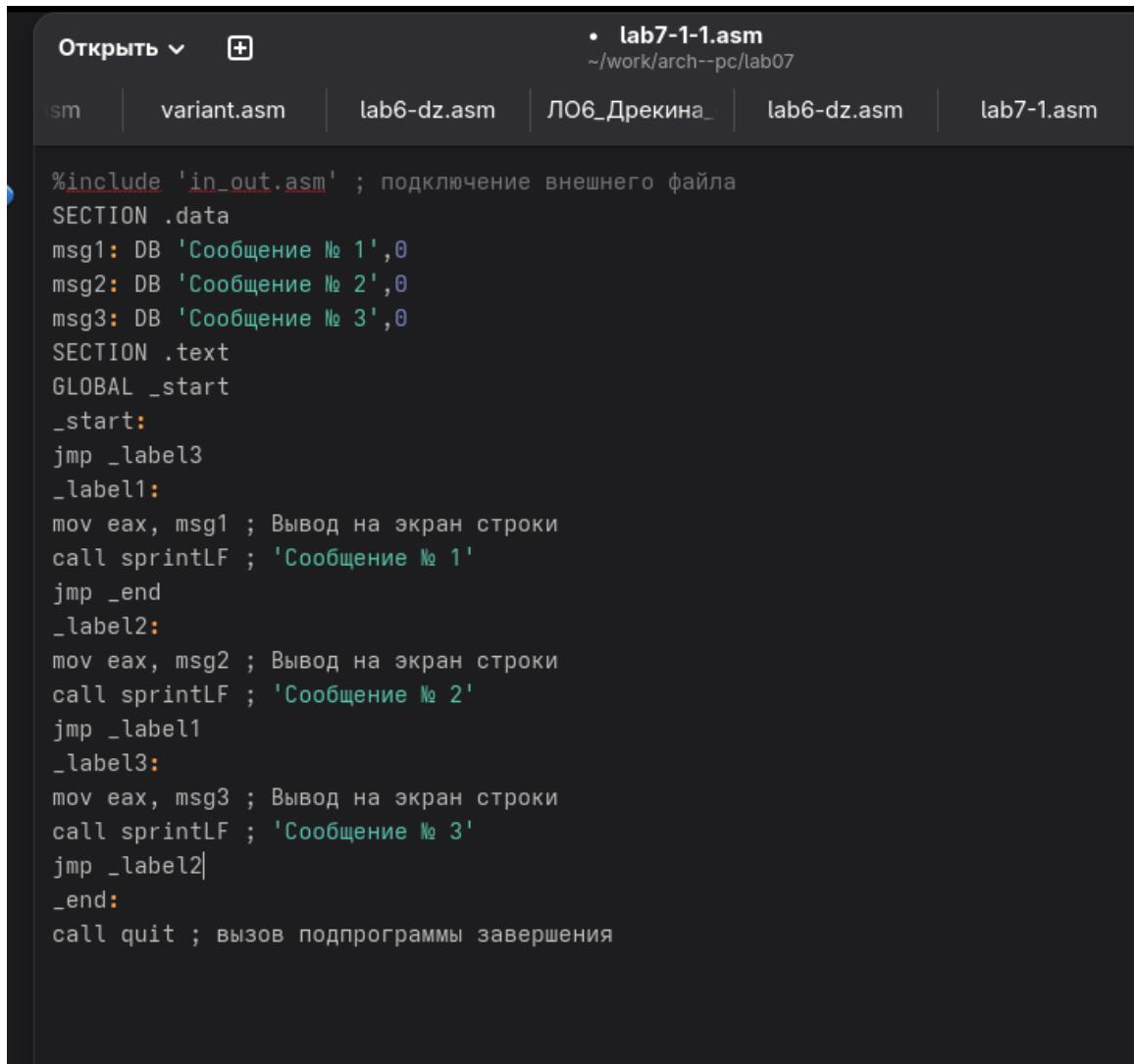
_label2:
    mov eax, msg2 ; Вывод на экран строки
    call sprintLF ; 'Сообщение № 2'

    jmp _label1

_label3:
    mov eax, msg3 ; Вывод на экран строки
```

```
call sprintLF ; 'Сообщение № 3'  
_end:  
call quit ; вызов подпрограммы завершения
```

Далее я изменила программу так, чтобы программа выводила сначала «Сообщение № 3», потом «Сообщение № 2» и в конце «Сообщение № 1». (Рисунок 3.7)



```
Открыть +  
• lab7-1-1.asm  
~/work/arch--pc/lab07  
sm variant.asm lab6-dz.asm ЛО6_Дрекина_ lab6-dz.asm lab7-1.asm  
  
%include 'in_out.asm' ; подключение внешнего файла  
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 ; Вывод на экран строки  
call sprintLF ; 'Сообщение № 1'  
jmp _end  
_label2:  
mov eax, msg2 ; Вывод на экран строки  
call sprintLF ; 'Сообщение № 2'  
jmp _label1  
_label3:  
mov eax, msg3 ; Вывод на экран строки  
call sprintLF ; 'Сообщение № 3'  
jmp _label2  
_end:  
call quit ; вызов подпрограммы завершения
```

Рисунок 3.7: Корректировки кода.

Затем я запустила полученную программу (Рисунок 3.8)

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

Рисунок 3.8: Запуск программы.

У меня вывелаась правильная последовательность, значит я правильно внесла изменения и поняла как работает программа.

Далее я создала еще один файл для работы. (Рисунок 3.9)

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

Рисунок 3.9: Создание файла.

В созданный файл я ввела текст из Листинга 7.3. (Рисунок 3.10)

Открыть lab7-2.asm
~/work/arch--pc/lab07

sm | lab6-dz.asm | ЛО6_Дрекина_ | lab6-dz.asm | lab7-1.asm | lab7-1

```
%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:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рисунок 3.10: Ввод программы.

Листинг 7.3:

```
%include 'in_out.asm'

section .data
msg1 db 'Введите В: ',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:
; ----- Вывод сообщения 'Введите В: '
mov eax,msg1
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'В' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'В'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
```

```

cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход

```

Затем я создала исполняемый файл и запустила программу. (Рисунок 3.11)

```
наибольшее число: 50
aadrekina@fedora:~/work/arch--pc/lab07$ ./lab7-2
Введите В: 100
Наибольшее число: 100
aadrekina@fedora:~/work/arch--pc/lab07$ ./lab7-2
Введите В: 5
Наибольшее число: 50
aadrekina@fedora:~/work/arch--pc/lab07$
```

Рисунок 3.11: Запуск программы.

Я ввела несколько разных значений В, чтобы убедиться в корректности работы программы.

4 Изучение структуры файлы листинга.

Теперь я создала файл листинга для программы из файла «lab7-2.asm», с помощью ключа -l. (Рисунок 4.1)

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

Рисунок 4.1: Создание файла листинга для программы.

Затем я открыла созданный файл с помощью редактора mcedit. (Рисунок 4.2)

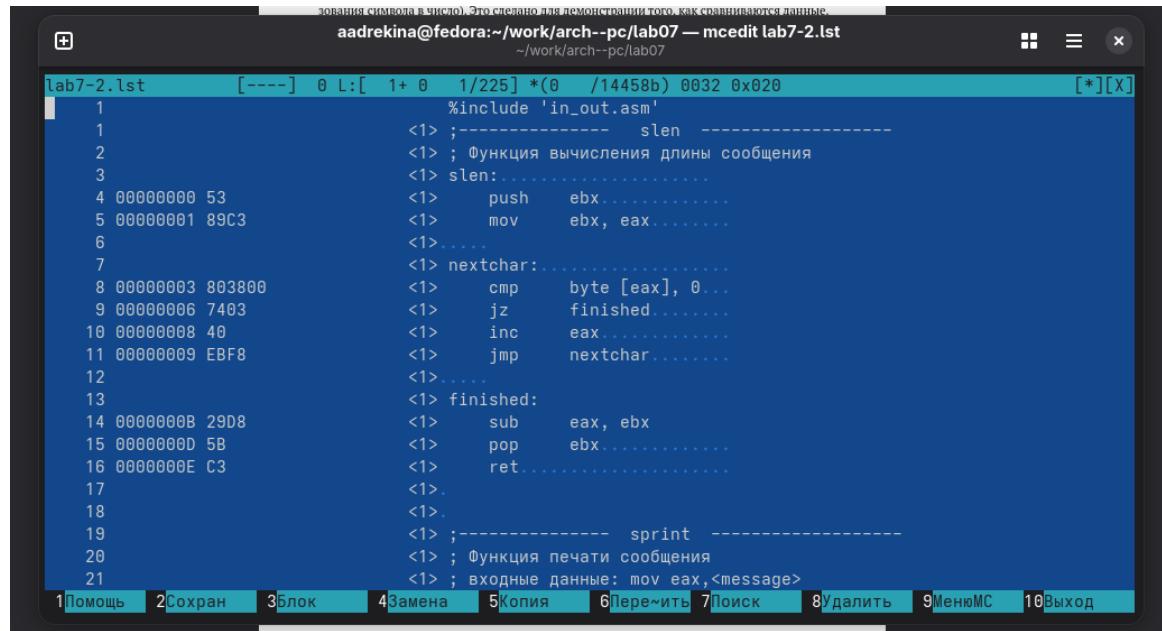


Рисунок 4.2: Открытие файла.

Я ознакомилась с содержимым. Я выбрала три строки, чтобы описать содержимое:

Первая строка:

6 - «00000039 35300000 C dd „50“», эта инструкция хранит в переменной С значение «50» в виде символьных байт. dd записывает в секцию указанное значение в виде 32-битного слова. Кавычки около 50 означают, что это не число, а символьное выражение. 00000039 - адрес где в памяти лежит С. 35300000 - значение, которое ассемблер записал в память по этому адресу.

Вторая строка:

14 - «000000E8 B8[00000000] mov eax,msg1», эта инструкция кладет в регистр eax адрес строки msg1. [00000000] - это место, куда подставится реальный адрес метки. 000000E8 - адрес команды в памяти. B8 - код операции mov в регистре eax.

Третья строка:

23 «0000010B A3[0A000000] mov [B],eax», mov это команда перемещения данных в x86, [B] обращение к памяти по адресу B, eax регистор в котором хранится результат функции atoi. Если смотреть в общем, то это строка копирует число из регистра eax и переносит его в переменную B.

Далее я открыла файл lab7-2.asm и в инструкции с двумя операндами удалила один.
(Рисунок 4.3)

```
а в числе | mov [B],eax ; запись преобразованного числа в B
и символа в ч ; ----- Записываем 'A' в переменную 'max'
сла в max    mov ecx; 'ecx = A'
()           mov [max],ecx ; 'max = A'
              ; ----- Сравниваем 'A' и 'C' (как символы)
```

Рисунок 4.3: Удаление одного операнда.

Затем я попыталась сделать файл исполняемым, но вышла ошибка. Все потому что инструкция ожидает на вход два операнда. (Рисунок 4.4)

```
aadrekina@fedora:~/work/arch--pc/lab07$ nasm -f elf lab7-2.asm  
lab7-2.asm:25: error: invalid combination of opcode and operands  
aadrekina@fedora:~/work/arch--pc/lab07$
```

Рисунок 4.4: Запуск программы.

5 Задание для самостоятельной работы.

Далее я создала файл для самостоятельно работы. И на основе Листингов из лекции составила программу. Значения переменных я взяла из 9 варианта.(Рисунок 5.1)

Открыть lab7-1dz.asm
~/work/arch--pc/lab07

asm | lab5-1-2.asm | lab6-1.asm | ЛО6_Дрекина_ | lab7-1dz.asm | lab7-2.lst

```
%include 'in_out.asm'
section .data
msg2 db "Наименьшее число: ",0h
A dd '24'
C dd '98'
B dd '15'
section .bss
min resb 10
section .text
global _start
_start:
    mov eax,B
    call atoi
    mov [B],eax
    mov ecx,[A]
    mov [min],ecx
    cmp ecx,[C] ;
    jl check_B ;
    mov ecx,[C] ;
    mov [min],ecx
check_B:
    mov eax,min
    call atoi|
    mov [min],eax
    mov ecx,[min]
    cmp ecx,[B]
    jl fin ;
    mov ecx,[B] ;
    mov [min],ecx
fin:
    mov eax, msg2
    call sprint
    mov eax,[min]
    call iprintLF
    call quit
```

Рисунок 5.1: Создание программы.

Затем я запустила программу. (Рисунок 5.2)

```
aadrekina@fedora:~/work/arch--pc/lab07$ nasm -f elf lab7-1dz.asm
aadrekina@fedora:~/work/arch--pc/lab07$ ld -m elf_i386 -o lab7-1dz lab7-1dz.o
aadrekina@fedora:~/work/arch--pc/lab07$ ./lab7-1dz
Наименьшее число: 15
aadrekina@fedora:~/work/arch--pc/lab07$ █
```

Рисунок 5.2: Запуск программы.

На выходе у меня получилось, что самое минимальное число это 15, что соответствует действительности.

Теперь я создала еще один текстовый файл. Написала туда другую программу, так же основываясь на Листингах из лекции. (Рисунок 5.3)

The screenshot shows a terminal window with the title 'Виктория-88-55'. The window contains assembly code for a program named 'lab7-2dz.asm' located at '~/work/arch--pc/lab07'. The code includes directives like %include, section .data, and section .text, along with various mov, call, cmp, and jmp instructions. It handles user input for variables x and a, performs a calculation, and prints the result.

```
1 %include 'in_out.asm'
2 section .data
3 msgX db 'Введите x: ',0h
4 msgA db 'Введите a: ',0h
5 msgR db 'Результат f(x): ',0h
6 section .bss
7 x resb 10
8 a resb 10
9 res resd 10
10 section .text
11 global _start
12 _start:
13 mov eax,msgX
14 call sprint
15 mov ecx,x
16 mov edx,10
17 call sread
18 mov eax,x
19 call atoi
20 mov [x],eax
21 mov eax,msgA
22 call sprint
23 mov ecx,a
24 mov edx,10
25 call sread
26 mov eax,a
27 call atoi
28 mov [a],eax
29 mov eax,[x]
30 mov ebx,[a]
31 cmp eax,ebx
32 jl check_B
33 mov [res],ebx
34 jmp fin
35 check_B:
36 add eax,ebx
37 mov [res], eax
38 fin:
39 mov eax,msgR
40 call sprint
41 mov eax,[res]
42 call iprintLF
43 call quit
```

Рисунок 5.3: Текст программы.

Затем я запустила программу и ввела данные из варианта №9. (Рисунок 5.4)

```
введите x: 5
введите a: 7
результат f(x): 12
aadrekina@fedora:~/work/arch--pc/lab07$ ./lab7-2dz
введите x: 6
введите a: 4
результат f(x): 4
aadrekina@fedora:~/work/arch--pc/lab07$ █
```

Рисунок 5.4: Запуск программы.

После запуска, я проверила вручную. Вывод программы совпал с моими вычислениями. Значит, программа написана правильно.

6 Вывод

Я изучила команды условного и безусловного переходов. Приобрела навыки написания программ с использованием переходов. Ознакомилась с назначением и структурой файла листинга. И все полученные знания применила на практике.