

# Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера  
Бородин Дмитрий Алексеевич

## 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.....	10
4.3	Задания для самостоятельной работы .....	13
5	Выводы.....	19
	Список литературы.....	19

## 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. Самостоятельное написание программ по материалам лабораторной работы

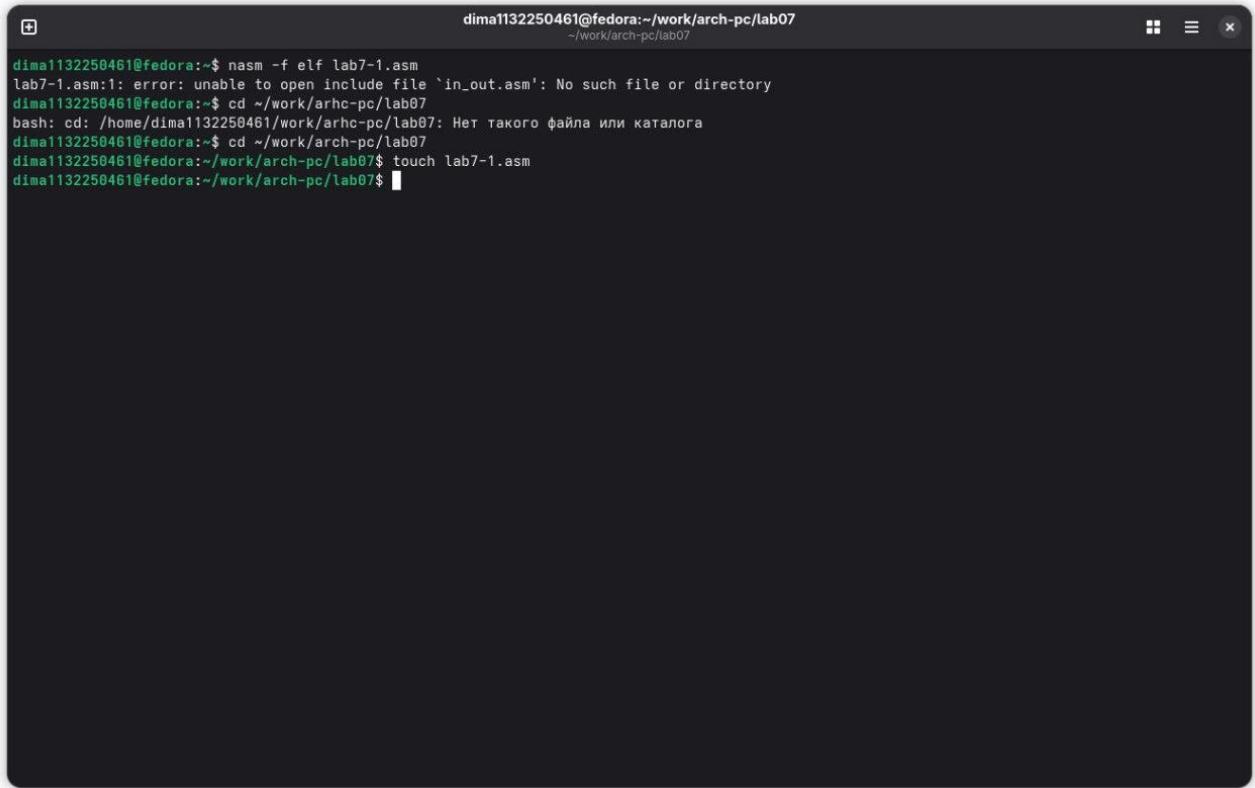
## 3 Теоретическое введение

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).



```
dima1132250461@fedora:~/work/arch-pc/lab07
~/work/arch-pc/lab07

dima1132250461@fedora:~$ nasm -f elf lab7-1.asm
lab7-1.asm:1: error: unable to open include file `in_out.asm': No such file or directory
dima1132250461@fedora:~$ cd ~/work/arch-pc/lab07
bash: cd: /home/dima1132250461/work/arch-pc/lab07: Нет такого файла или каталога
dima1132250461@fedora:~$ cd ~/work/arch-pc/lab07
dima1132250461@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
dima1132250461@fedora:~/work/arch-pc/lab07$
```

*Рис. 1: Создание каталога и файла для программы*

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

The screenshot shows a terminal window with the following content:

```
*~/work/arch-pc/lab07/lab7-1.asm - Mousepad
Файл Правка Поиск Просмотр Документ Помощь
%include 'in_out.asm'

SECTION .data
msg1: DB 'Cookiemine № 1', 0
msg2: DB 'Cookiemine № 2', 0
msg3: DB 'Cookiemine № 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. 2: Сохранение программы*

При запуске программы я убедился в том, что неусловный переход действительно изменяет порядок выполнения инструкций (рис. 3).

The screenshot shows a terminal window with a dark background and light-colored text. At the top, it displays the command line information: `dima1132250461@fedora:~/work/arch-pc/lab07`. Below this, the terminal shows the following sequence of commands and output:

```
dima1132250461@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
dima1132250461@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
dima1132250461@fedora:~/work/arch-pc/lab07$ ./lab7-1
Cookiemine № 2
Cookiemine № 3
dima1132250461@fedora:~/work/arch-pc/lab07$
```

*Рис. 3: Запуск программы*

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

```
*~/work/arch-pc/lab07/lab7-1.asm - Mousepad
Файл Правка Поиск Просмотр Документ Помощь
%include 'in_out.asm'

SECTION .data
msg1: DB 'CooGueHue № 1', 0
msg2: DB 'CooGueHue № 2', 0
msg3: DB 'CooGueHue № 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. 4: Изменение программы*

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

The screenshot shows a terminal window with a dark background and light-colored text. The title bar indicates the session is running on a Fedora system with the command-line interface. The terminal displays the following sequence of commands and output:

```
dima1132250461@fedora:~/work/arch-pc/lab07
~/work/arch-pc/lab07
dima1132250461@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
dima1132250461@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
dima1132250461@fedora:~/work/arch-pc/lab07$ ./lab7-1
CooGueHue № 2
CooGueHue № 1
dima1132250461@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).

```
*~/work/arch-pc/lab07/lab7-1.asm - Mousepad
Файл Правка Поиск Просмотр Документ Помощь
%include 'in_out.asm'

SECTION .data
msg1: DB 'CooGuehke № 1', 0
msg2: DB 'CooGuehke № 2', 0
msg3: DB 'CooGuehke № 3', 0

SECTION .text
GLOBAL _start

_start:
    jmp _label3

_label1:
    mov eax, msg1
    call sprintLF
    jmp _end

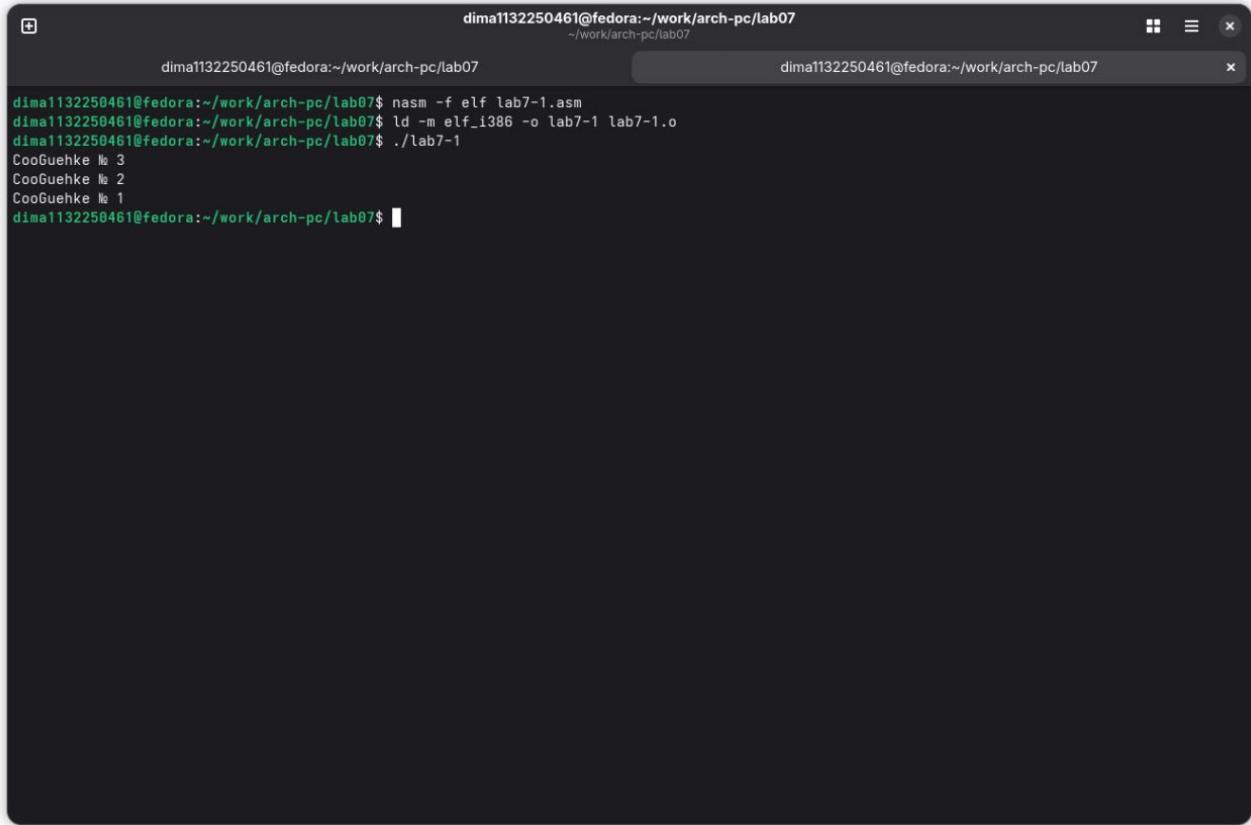
_label2:
    mov eax, msg2
    call sprintLF
    jmp _label1

_label3:
    mov eax, msg3
    call sprintLF
    jmp _label2

_end:
    call quit|
```

*Рис. 6: Изменение программы*

Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. 7).



The screenshot shows a terminal window with a dark background and light-colored text. The title bar indicates the session is running on a Fedora system. The terminal displays the following command-line session:

```
dima1132250461@fedora:~/work/arch-pc/lab07
~/work/arch-pc/lab07
dima1132250461@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
dima1132250461@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
dima1132250461@fedora:~/work/arch-pc/lab07$ ./lab7-1
CooGuehke № 3
CooGuehke № 2
CooGuehke № 1
dima1132250461@fedora:~/work/arch-pc/lab07$
```

Рис. 7: Проверка изменений

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

Файл Правка Поиск Просмотр Документ Помощь

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:

```
    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]
    mov ecx, eax
```

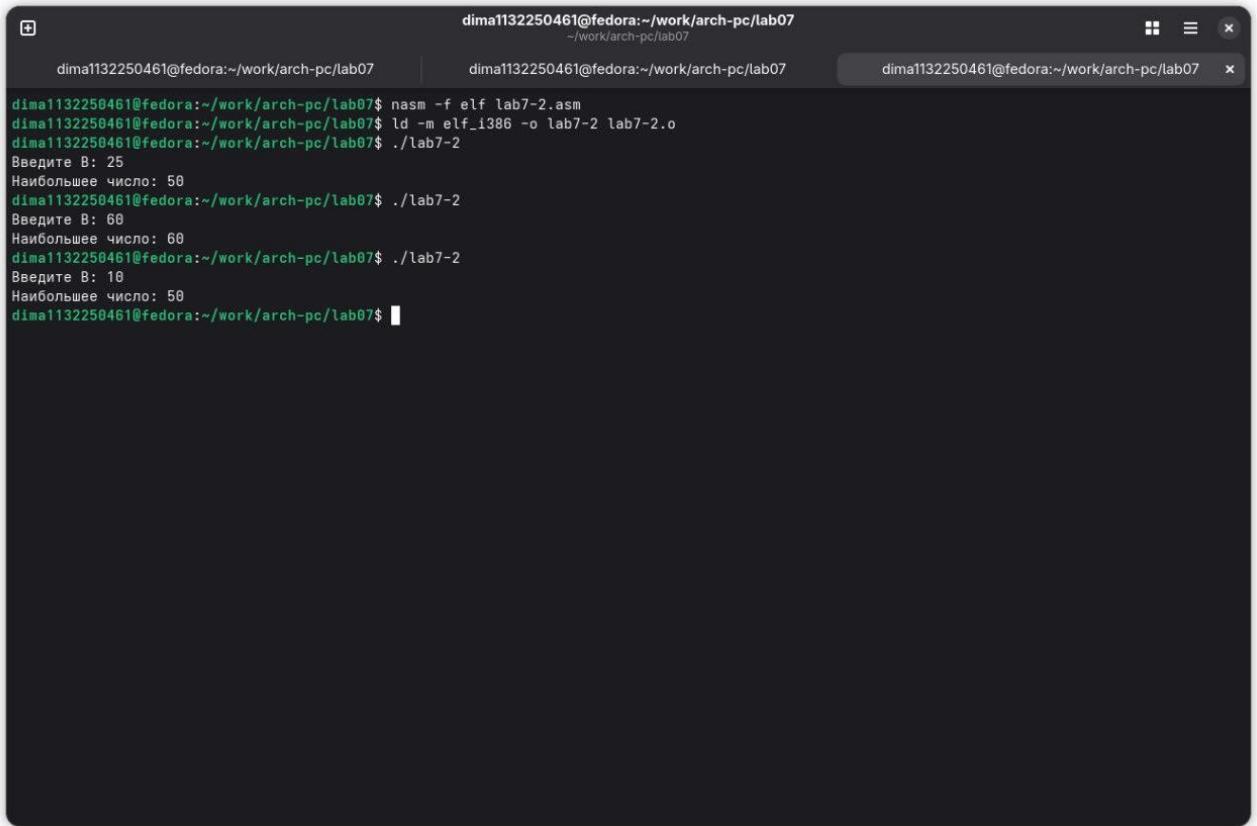
```
    cmp ecx, [B]
    jg fin
    mov ecx, [B]
    mov [max], ecx
```

fin:

```
    mov eax, msg2
    call sprint
    mov eax, [max]
    call iprintlnF
    call quit
```

*Рис. 8: Сохранение новой программы*

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



```
dima1132250461@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
dima1132250461@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
dima1132250461@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 25
Наибольшее число: 50
dima1132250461@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
dima1132250461@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
dima1132250461@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).

~/work/arch-pc/lab07/lab7-2.lst - Mousepad

x

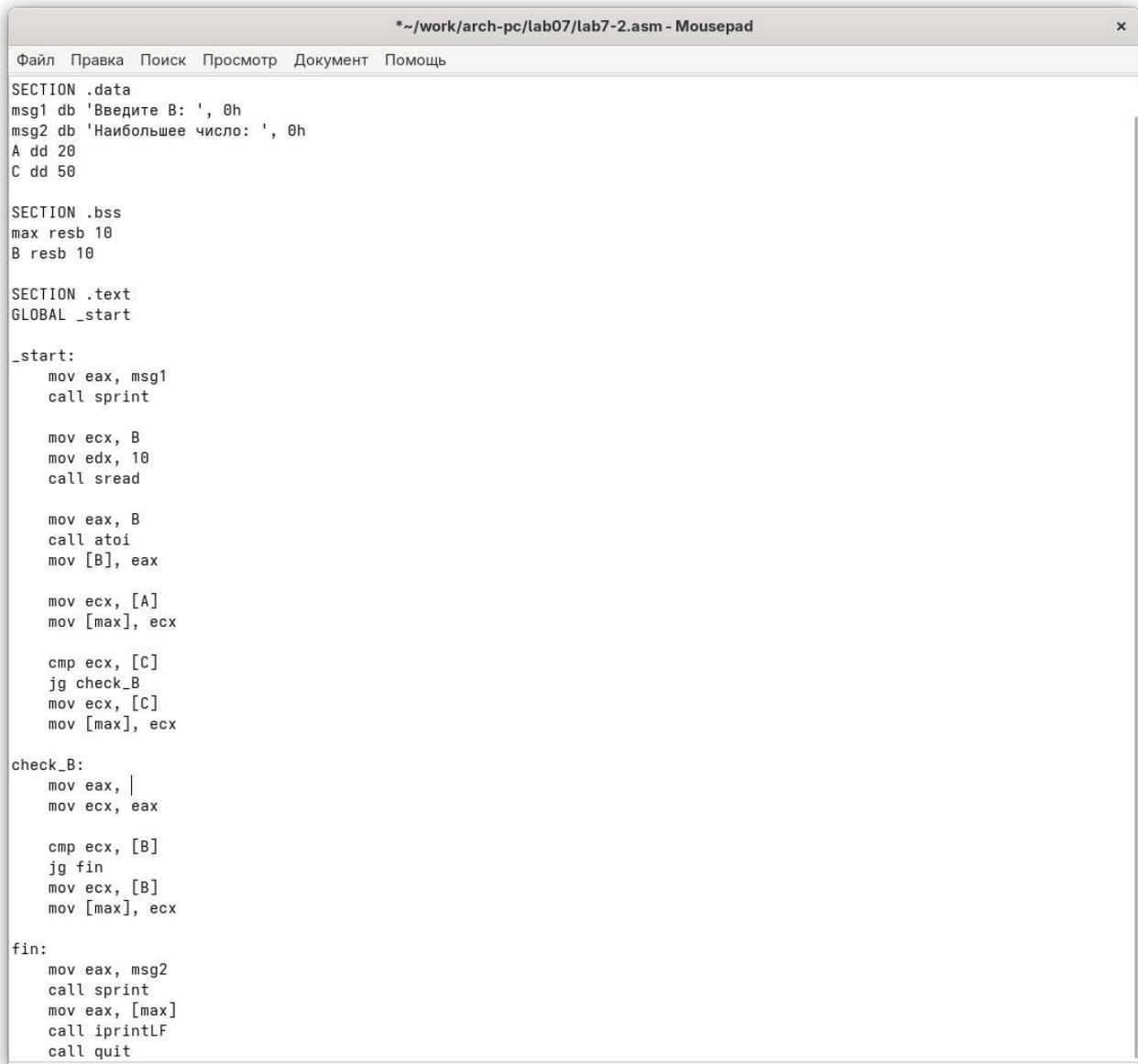
Файл Правка Поиск Просмотр Документ Помощь

```
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 E8E8FFFFFF             <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
45                                     <1> ; входные данные: mov eax,<message>
46                                     <1> sprintLF:
47 AAAAAAA2D F800FFFFFFFFFF         <1>     call    sprint
```

*Рис. 10: Проверка файла листинга*

Первое значение в файле листинга - номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями.

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



The screenshot shows a window titled "\*~/work/arch-pc/lab07/lab7-2.asm - Mousepad". The menu bar includes Файл, Правка, Поиск, Просмотр, Документ, Помощь. The assembly code is as follows:

```
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:
    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, |
    mov ecx, eax

    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. 11: Удаление операнда из программы*

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. 12).

Файл Правка Поиск Просмотр Документ Помощь

```
5 0000002E D0BB00BE3A2000
6 00000035 14000000          A dd 20
7 00000039 32000000          C dd 50
8
9                      SECTION .bss
10 00000000 <res Ah>        max resb 10
11 0000000A <res Ah>        B resb 10
12
13                      SECTION .text
14                      GLOBAL _start
15
16                      _start:
17 000000E8 B8[00000000]      mov eax, msg1
18 000000ED E81DFFFF         call sprint
19
20 000000F2 B9[0A000000]      mov ecx, B
21 000000F7 BA0A000000        mov edx, 10
22 000000FC E842FFFFFF        call sread
23
24 00000101 B8[0A000000]      mov eax, B
25 00000106 E891FFFF         call atoi
26 0000010B A3[0A000000]      mov [B], eax
27
28 00000110 8B0D[35000000]    mov ecx, [A]
29 00000116 890D[00000000]    mov [max], ecx
30
31 0000011C 3B0D[39000000]    cmp ecx, [C]
32 00000122 7F0C              jg check_B
33 00000124 8B0D[39000000]    mov ecx, [C]
34 0000012A 890D[00000000]    mov [max], ecx
35
36                      check_B:
37                      *****
38 00000130 89C1              mov eax,
39
40 00000132 3B0D[0A000000]    mov ecx, eax
41 00000138 7F0C              cmp ecx, [B]
42 0000013A 8B0D[0A000000]    jg fin
43 00000140 890D[00000000]    mov ecx, [B]
44
45                      fin:
46 00000146 B8[13000000]      mov eax, msg2
47 0000014B E8BFEEFFFF        call sprint
48 00000150 A1[00000000]      mov eax, [max]
49 00000155 E82CFFFFFF        call iprintLF
50 0000015A E87CFFFFFF        call quit
```

Рис. 12: Просмотр ошибки в файле листинга

#### 4.3 Задания для самостоятельной работы

Искренне не понимаю, какой вариант я должен был получить во время 7 лабораторной работы, поэтому буду использовать свой вариант - девятый - из предыдущей лабораторной работы. Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 13).

The screenshot shows a terminal window with the title bar reading "\*~/work/arch-pc/lab07/lab7-2.asm - Mousepad". The window contains the following assembly code:

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите В: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '24'
C dd '15'

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
```

Рис. 13: Первая программа самостоятельной работы

Код первой программы:

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите В: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '24'
C dd '15'

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
    call quit
```

Проверяю корректность написания первой программы (рис. 14).

The screenshot shows a terminal window with a dark background and light-colored text. The terminal title is "dima1132250461@fedora:~/work/arch-pc/lab07". The command history at the top shows several commands related to the assembly file "lab7-2.asm": "mousepad lab7-2.asm", "nasm -f elf lab7-2.asm", "ld -m elf\_i386 -o lab7-2 lab7-2.o", and "./lab7-2". Below the command history, the terminal displays user input and output:

```
dima1132250461@fedora:~/work/arch-pc/lab07$ mousepad lab7-2.asm
dima1132250461@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
dima1132250461@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
dima1132250461@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 98
Наименьшее число: 15
dima1132250461@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).

The screenshot shows a terminal window titled "Mousepad" with the path "/work/arch-pc/lab07/lab7-3.asm". The window contains assembly code for a C program. The code includes sections for data and text, with various memory operations like mov, add, and print.

```
%include 'in_out.asm'
SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 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
    jle add_values
    mov eax, esi
    jmp print_result

add_values:
    mov eax, edi
    add eax, esi

print_result:
    mov edi, eax
    mov eax, res
    call sprint
    mov eax, edi
    call iprintf
    call quit
```

Рис. 15: Вторая программа самостоятельной работы

Код второй программы:

```
%include 'in_out.asm'
SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 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
jle add_values
mov eax, esi
jmp print_result

add_values:
mov eax, edi
add eax, esi

print_result:
mov edi, eax
mov eax, res
call sprint
mov eax, edi
call iprintLF
call quit
```

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений a и x (рис. 16).

```
dima1132250461@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
dima1132250461@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
dima1132250461@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 12
dima1132250461@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 4
dima1132250461@fedora:~/work/arch-pc/lab07$
```

Рис. 16: Проверка работы второй программы

## 5 Выводы

При выполнении лабораторной работы я изучил команды условных и безусловных переходов, а также приобрел навыки написания программ с использованием переходов, познакомился с назначением и структурой файлов листинга.

## Список литературы

1. Курс на ТУИС
2. Лабораторная работа №7
3. Программирование на языке ассемблера NASM Столяров А. В.