

Отчет по лабораторной работе №7

Команды безусловного и условного переходов в Nasm. Программирование ветвлений

ФИО: Маркеш Виейра Нанке Грасимилде

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. Задание 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. -fig.4.1).

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

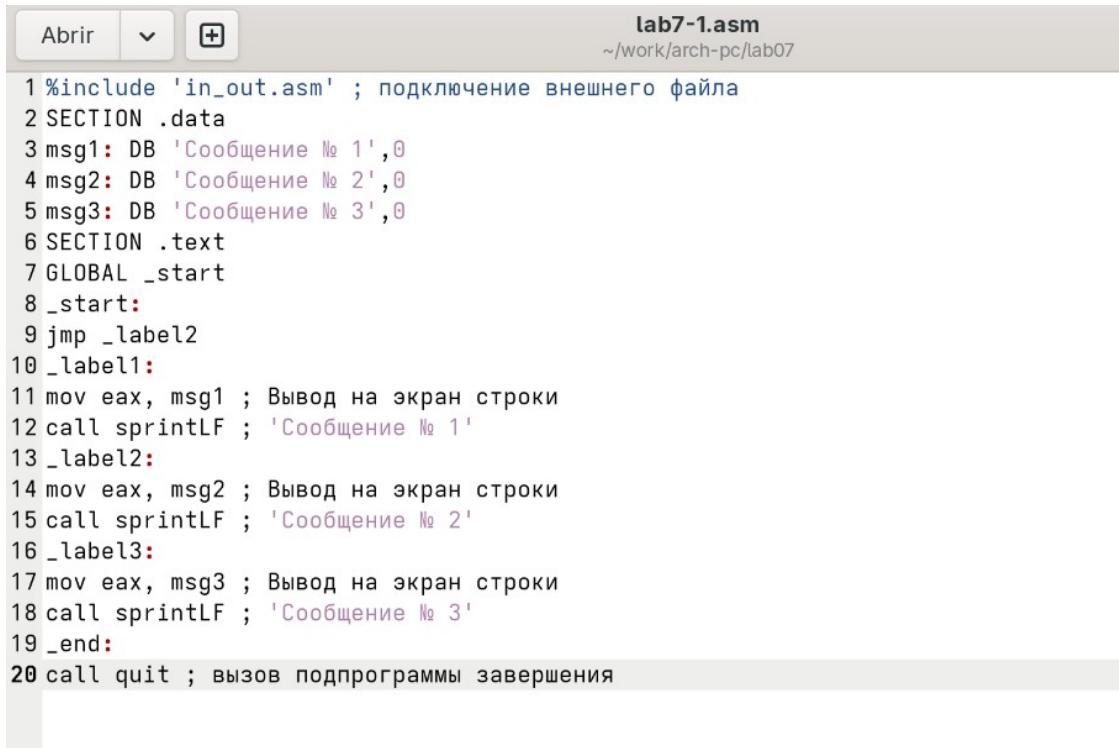
Рис 4.1: Создание каталога и файла для программы

Копирую файл in_out.asm в каталог для выполнения лабораторной работы (рис. -fig.4.2).

```
gracimildevieira@fedora:~/work/arch-pc/lab07$ cp ~/Downloads/in_out.asm in_out.asm
```

Рис 4.2: Сохранение программы

Копирую код из листинга в файл будущей программы (рис. -fig.4.3).



The screenshot shows a text editor window titled "lab7-1.asm" located at "~/work/arch-pc/lab07". The code in the editor is as follows:

```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintLF ; 'Сообщение № 1'
13 _label2:
14 mov eax, msg2 ; Вывод на экран строки
15 call sprintLF ; 'Сообщение № 2'
16 _label3:
17 mov eax, msg3 ; Вывод на экран строки
18 call sprintLF ; 'Сообщение № 3'
19 _end:
20 call quit ; вызов подпрограммы завершения
```

Рис 4.3: Запуск программы

Создаю исполняемый файл и запускаю его. (рис. -fig.4.4)

```
gracimildevieira@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
gracimildevieira@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
gracimildevieira@fedora:~/work/arch-pc/lab07$ ./lab7-1
```

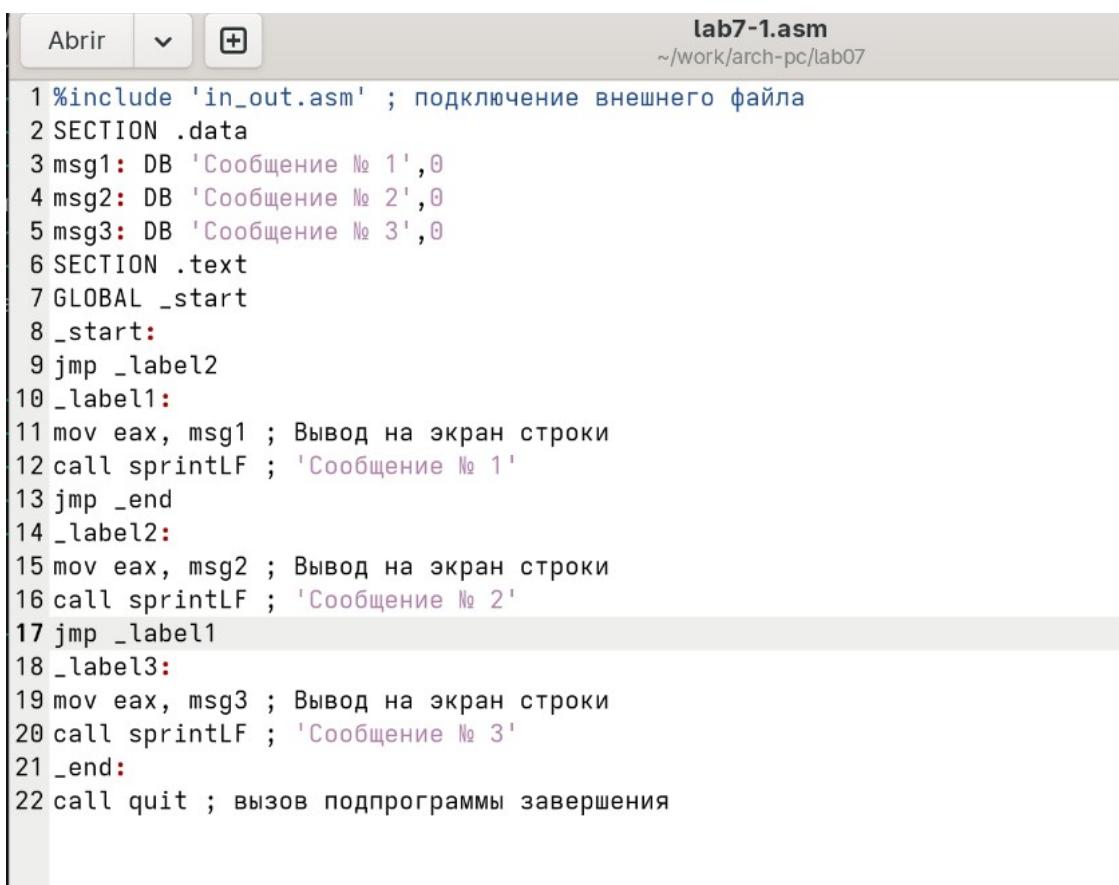
Рис 4.4: Изменение программы

Вывод программы - вывелось все, кроме Сообщения №1 (рис. -fig.4.5).

```
gracimildevieira@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
gracimildevieira@fedora:~/work/arch-pc/lab07$
```

Рис 4.5: Запуск измененной программы

Теперь изменяю текст программы в соответствии с листингом 7.2 (рис. -fig.4.6).



The screenshot shows a text editor window titled "lab7-1.asm" with the file path "~/work/arch-pc/lab07". The code in the editor is as follows:

```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintLF ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintLF ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintLF ; 'Сообщение № 3'
21 _end:
22 call quit ; вызов подпрограммы завершения
```

Рис 4.6: Изменение программы

Создаю исполняемый файл. (рис. -fig.4.7).

```
gracimildevieira@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
gracimildevieira@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
gracimildevieira@fedora:~/work/arch-pc/lab07$ ./lab7-1
```

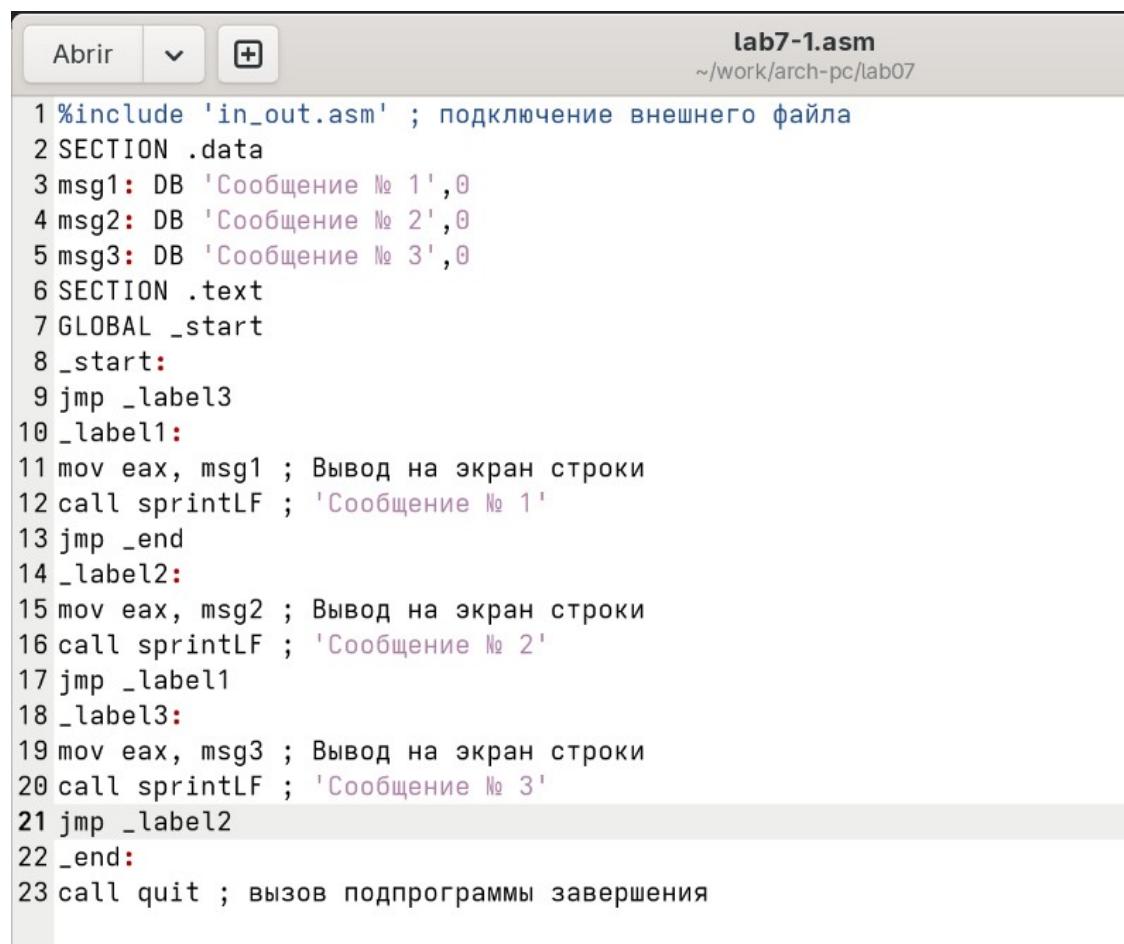
Рис 4.7: Проверка изменений

В этот раз вывод начался с Сообщения №2, затем вывелось Сообщение №1. (рис. -fig.4.8).

```
gracimildevieira@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
gracimildevieira@fedora:~/work/arch-pc/lab07$
```

Рис 4.8: Сохранение новой программы

Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. -fig.4.9).



The screenshot shows a text editor window titled "lab7-1.asm" located at "/work/arch-pc/lab07". The assembly code is as follows:

```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label3
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintLF ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintLF ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintLF ; 'Сообщение № 3'
21 jmp _label2
22 _end:
23 call quit ; вызов подпрограммы завершения
```

Рис 4.9: Проверка программы из листинга

Работа выполнена корректно, программа в нужном мне порядке выводит сообщения(рис. -fig.4.10).

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

Рис 4.10: Проверка файла листинга

Создаю новый рабочий файл, проверяю с помощью ls что файл был создан. (рис. -fig.4.11).

```
gracimildevieira@fedora:~/work/arch-pc/lab07$ touch lab7-2.asm
gracimildevieira@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2.asm
gracimildevieira@fedora:~/work/arch-pc/lab07$ █
```

Рис 4.11: Удаление операнда из программы

Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными (рис. -fig.4.12).

```
gracimildevieira@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
gracimildevieira@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
gracimildevieira@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 8
Наибольшее число: 50
gracimildevieira@fedora:~/work/arch-pc/lab07$ █
```

Рис 4.12: Просмотр ошибки в файле листинга

```
gracimildevieira@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 100
Наибольшее число: 100
gracimildevieira@fedora:~/work/arch-pc/lab07$ █
```

Рис 4.13: Первая программа самостоятельной работы

4.2 Изучение структуры файла листинга

Создаю файл листинга с помощью флага -l команды nasm и открываю его с помощью текстового редактора gedit (рис. -fig.4.10).

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

Рис 4.14: Проверка файла листинга

```
gracimildevieira@fedora:~/work/arch-pc/lab07$ gedit lab7-2.lst
```

Рис 4.15: Удаление операнда из программы

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.

The screenshot shows a terminal window with the following content:

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

1 1
2 1
3 2
4 3
5 4 00000000 53
6 5 00000001 89C3
7 6
8 7
9 8 00000003 803800
10 9 00000006 7403
11 10 00000008 40
12 11 00000009 EBF8
13 12
14 13
15 14 0000000B 29D8
16 15 0000000D 5B
17 16 0000000E C3
18 17
19 18 |
20 19
21 20
22 21
23 22
24 23 0000000F 52
25 24 00000010 51
26 25 00000011 53
27 26 00000012 50
28 27 00000013 E8E8FFFF
29 28
30 29 00000018 89C2
31 30 0000001A 58
32 31
33 32 0000001B 89C1
34 33 0000001D B8A1000000
```

The code is an assembly listing for a program named 'in_out.asm'. It includes comments in Russian explaining the purpose of certain sections like 'slen' and 'sprint'. Line 19 contains a cursor at the start of a blank line.

Рис 4.16: Просмотр ошибки в файле листинга

Удаляю один operand из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. -fig.4.11).

```

188      14 000000E8 B8[00000000]      , Вывод сообщения 'Вводите С'
189      15 000000ED E81DFFFFFF
190      16
191      17 000000F2 B9[0A000000]      ; ----- Ввод 'B'
192      18 000000F7 BA6A000000
193      19 000000FC E842FFFFFF
194      20
195      21 00000101 B8[0A000000]
196      22 00000106 E891FFFFFF
197      23 0000010B A3[0A000000]
198      24
199      25 00000110 880D[35000000]
200      26 00000116 890D[00000000]
201      27
202      28 0000011C 3B0D[39000000]
203      29 00000122 7F0C
204      30 00000124 880D[39000000]
205      31 0000012A 890D[00000000]
206      32
207      33
208      34
209      34 *****
210      35 00000130 E867FFFFFF
211      36 00000135 A3[00000000]
212      37
213      38 0000013A 880D[00000000]
214      39 00000140 3B0D[0A000000]
215      40 00000146 7F0C
216      41 00000148 880D[0A000000]
217      42 0000014E 890D[00000000]
218      43
219      44
220      45 00000154 B8[13000000]
221      46 00000159 E8B1FFFFFF
222      47 0000015E A1[00000000]
223      48 00000163 E81EFFFFFF
224      49 00000168 E86EFFFFFF

```

Текст ▾ Ширина табуляции: 8 ▾ Ln 17, Col 69 INS

Рис 4.17: Проверка работы второй программы

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. -fig.4.12).

```

189 14 000000E8 B8[00000000]      ,         . . . . .
190 15 000000ED E81DFFFFFF
191 16
192 17 000000F2 B9[0A000000]      mov eax,msg1
193 18 000000F7 BA0A000000       call sprint
194 19 000000FC E842FFFFFF
195 20
196 21 00000101 B8[0A000000]      mov ecx,B
197 22 00000106 E891FFFFFF       ; ----- Ввод 'B'
198 23 0000010B A3[0A000000]      mov edx,10
199 24
200 25 00000110 880D[35000000]    call sread
201 26 00000116 890D[00000000]
202 27
203 28 0000011C 3B0D[39000000]    ; ----- Преобразование 'B' из символа в число
204 29 00000122 7F0C              call atoi ; Вызов подпрограммы перевода символа в число
205 30 00000124 880D[39000000]    mov [B],eax ; запись преобразованного числа в 'B'
206 31 0000012A 890D[00000000]
207 32
208 33
209 34
210 34  *****
211 35 00000130 E867FFFFFF       mov [max],eax ; 'max = A'
212 36 00000135 A3[00000000]
213 37
214 38 0000013A 880D[00000000]   ; ----- Сравниваем 'A' и 'C' (как символы)
215 39 00000140 3B0D[0A000000]   cmp ecx,[C] ; Сравниваем 'A' и 'C'
216 40 00000146 7F0C              jg check_B ; если 'A>C', то переход на метку 'check_B',
217 41 00000148 880D[0A000000]   mov ecx,[C] ; иначе 'ecx = C'
218 42 0000014E 890D[00000000]
219 43
220 44
221 45 00000154 B8[13000000]    ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
222 46 00000159 E8B1FFFFFF       call atoi ; Вызов подпрограммы перевода символа в число
223 47 0000015E A1[00000000]     mov eax,[max]
224 48 00000163 E81EFFFFFF       mov ecx,[B] ; Сравниваем 'max(A,C)>B', то переход на 'fin',
225 49 00000168 E86EFFFFFF       mov [max],ecx ; 'max = C'
                                         ; ----- Преобразование 'max(A,C)' из символа в число
                                         check_B:
                                         mov eax,
                                         error: invalid combination of opcode and operands
                                         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 iprintfLF ; Вывод 'max(A,B,C)'
                                         call quit ; Выход

```

Текст ▾ Ширина табуляции: 8 ▾ Ln 210, Col 16 INS

Рис 4.18: Проверка работы второй программы

4.3 Задания для самостоятельной работы

Буду использовать свой вариант - седьмой - из предыдущей лабораторной работы. Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением.

Код первой программы:

```
%include 'in_out.asm'
```

SECTION .data

```
msg1 db 'Введите B: ', 0h
```

```
msg2 db 'Наименьшее число: ', 0h
```

```
A dd '45'
```

```
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. -fig.4.14).

```

gracimildevieira@fedora:~/work/arch-pc/lab07$ gedit lab7-3.asm
gracimildevieira@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-3.lst lab7-3.asm
gracimildevieira@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
gracimildevieira@fedora:~/work/arch-pc/lab07$ ./lab7-3
Ведите В: 67
Наименьшее число: 45
gracimildevieira@fedora:~/work/arch-pc/lab07$ █

```

Рис 4.19: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатурых переменных а и х (рис. -fig.4.15).

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Ведите В: ', 0h
4 msg2 db 'Наименьшее число: ', 0h
5 A dd '45'
6 C dd '15'
7 SECTION .bss
8 min resb 10
9 B resb 10
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg1
14 call sprint
15 mov ecx, B
16 mov edx, 10
17 call sread
18 mov eax, B
19 call atoi
20 mov [B], eax
21 mov ecx, [A]
22 mov [min], ecx
23 cmp ecx, [C]
24 jg check_B
25 mov ecx, [C]
26 mov [B], ecx|  

27 check_B:  

28 mov eax, min
29 call atoi
30 mov [min], eax
31 mov ecx, [min]
32 cmp ecx, [B]
33 jb fin
34

```

Рис 4.20: 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 iprintf
    call quit
```

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений а и х (рис. -fig.4.16).

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите В: ', 0h
4 msg2 db 'Наименьшее число: ', 0h
5 A dd '45'
6 C dd '15'
7 SECTION .bss
8 min resb 10
9 B resb 10
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg1
14 call sprint
15 mov ecx, B
16 mov edx, 10
17 call sread
18 mov eax, B
19 call atoi
20 mov [B], eax
21 mov ecx, [A]
22 mov [min], ecx
23 cmp ecx, [C]
24 jg check_B
25 mov ecx, [C]
26 mov [B], ecx
27 check_B:
28 mov eax, min
29 call atoi
30 mov [min], eax
```

Рис 4.21: Проверка работы второй программы

5 Выводы

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

6 Список литературы

1. Лабораторная работа №7