

Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Луцкая Алиса Витальевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация переходов в NASM	8
4.2	Изучение структуры файла листинга	12
4.3	Задания для самостоятельной работы	15
5	Выводы	21
6	Список литературы	22

Список иллюстраций

4.1	Создание каталога и файла для программы	8
4.2	Ввод программы	9
4.3	Запуск программы	9
4.4	Изменение программы	10
4.5	Запуск измененной программы	10
4.6	Изменение программы	11
4.7	Проверка изменений	11
4.8	Создание файла	11
4.9	Ввод программы	12
4.10	Проверка программы из листинга	12
4.11	Проверка файла листинга	13
4.12	Удаление операнда из программы	14
4.13	Просмотр ошибки в файле листинга	14
4.14	Первая программа самостоятельной работы	15
4.15	Проверка работы первой программы	16
4.16	Вторая программа самостоятельной работы	18
4.17	Проверка работы второй программы	19

Список таблиц

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. Задания для самостоятельной работы

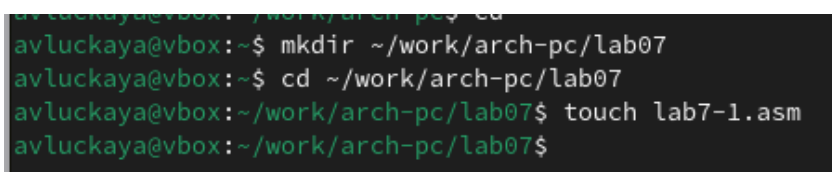
3 Теоретическое введение

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

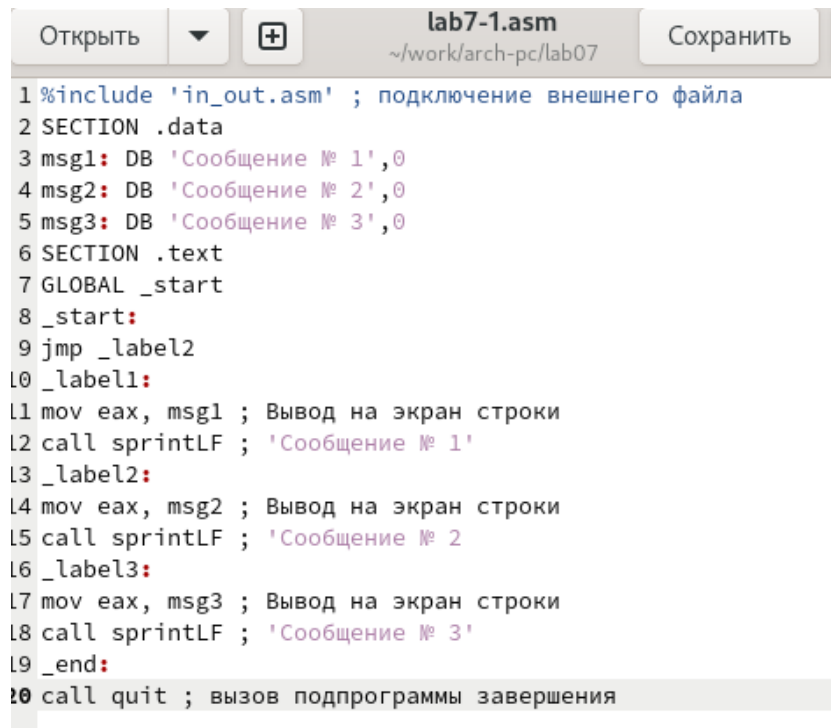
Создаю каталог для программ лабораторной работы №7, а также перехожу в него и создаю файл lab7-1.asm: (рис. -fig. 4.1).

A screenshot of a terminal window showing a series of commands and their outputs. The user is 'avluckaya' on a machine named 'vbox'. The commands are: 'mkdir ~/work/arch-pc/lab07', 'cd ~/work/arch-pc/lab07', and 'touch lab7-1.asm'. The prompt changes to reflect the current directory after the 'cd' command.

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

Рис. 4.1: Создание каталога и файла для программы

Ввожу в файл lab7-1.asm текст программы из листинга 7.1. (рис. -fig. 4.2).

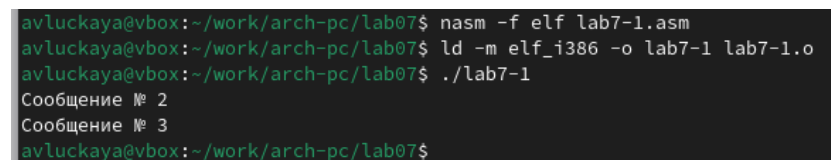


```
Открыть  lab7-1.asm  Сохранить
~/work/arch-pc/lab07

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 sprintfLF ; 'Сообщение № 1'
13 _label2:
14 mov eax, msg2 ; Вывод на экран строки
15 call sprintfLF ; 'Сообщение № 2'
16 _label3:
17 mov eax, msg3 ; Вывод на экран строки
18 call sprintfLF ; 'Сообщение № 3'
19 _end:
20 call quit ; вызов подпрограммы завершения
```

Рис. 4.2: Ввод программы

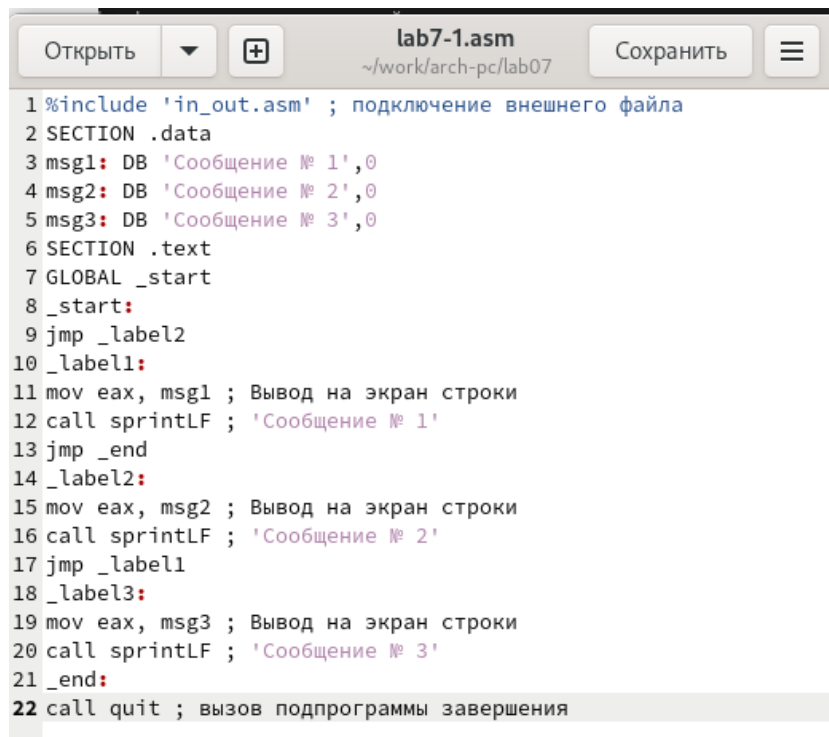
Создаю исполняемый файл и запускаю его. Убеждаюсь, что инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения. (рис. -fig. 4.3).



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

Рис. 4.3: Запуск программы

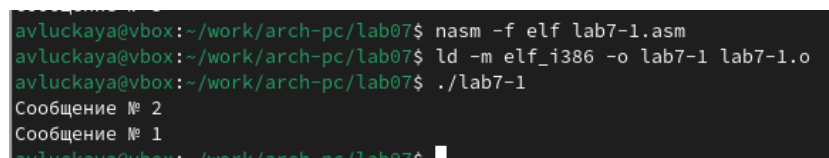
Изменяю текст программы в соответствии с листингом 7.2., чтобы поменялся порядок выполнения функций (рис. -fig. 4.4).



```
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 sprintf ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintf ; 'Сообщение № 3'
21 _end:
22 call quit ; вызов подпрограммы завершения
```

Рис. 4.4: Изменение программы

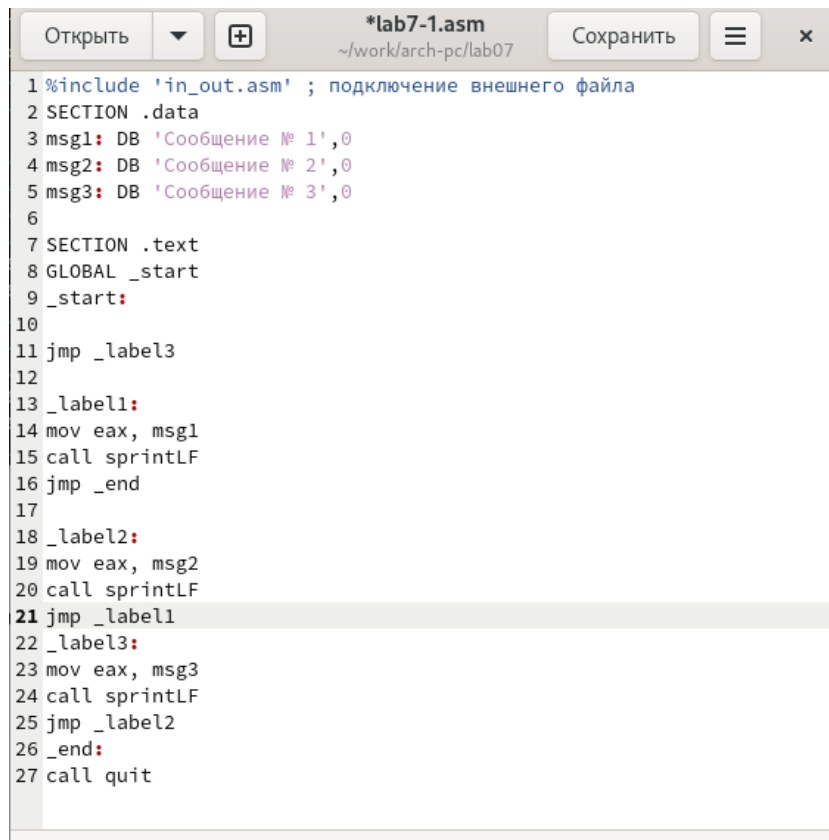
Запускаю программу и проверяю, что примененные изменения верны (рис. -fig. 4.5).



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

Рис. 4.5: Запуск измененной программы

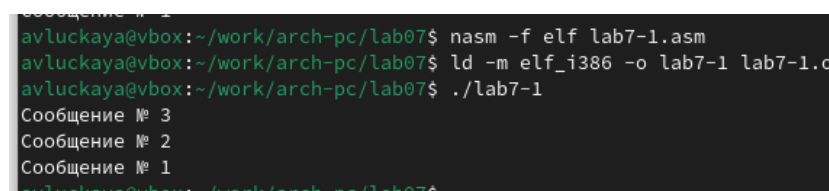
Изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. -fig. 4.6).



```
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
7 SECTION .text
8 GLOBAL _start
9 _start:
10
11 jmp _label3
12
13 _label1:
14 mov eax, msg1
15 call sprintfLF
16 jmp _end
17
18 _label2:
19 mov eax, msg2
20 call sprintfLF
21 jmp _label1
22 _label3:
23 mov eax, msg3
24 call sprintfLF
25 jmp _label2
26 _end:
27 call quit
```

Рис. 4.6: Изменение программы

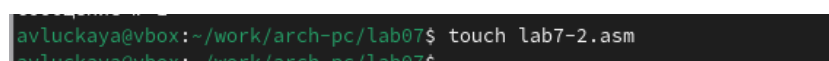
Проверяю корректность выполнения программы (рис. -fig. 4.7). Все работает верно.



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

Рис. 4.7: Проверка изменений

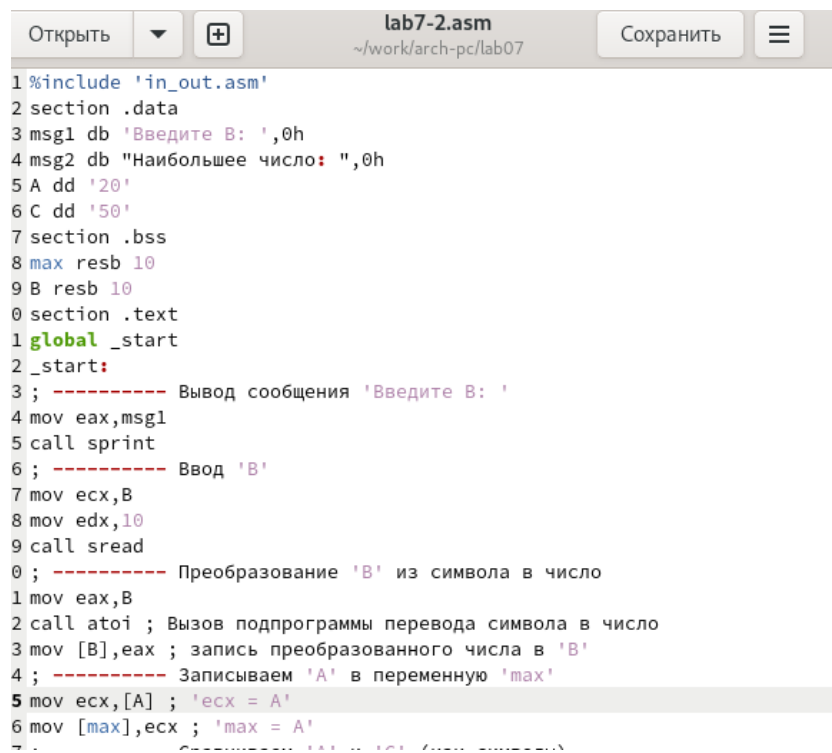
Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучаю текст программы из листинга 7.3(рис. -fig. 4.8).



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

Рис. 4.8: Создание файла

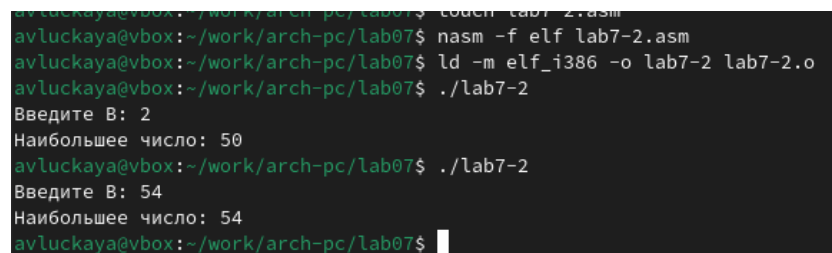
Ввожу текст программы из листинга 7.3 в lab7-2.asm.(рис. -fig. 4.9).



```
1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '20'
6 C dd '50'
7 section .bss
8 max resb 10
9 B resb 10
0 section .text
1 global _start
2 _start:
3 ; ----- Вывод сообщения 'Введите B: '
4 mov eax,msg1
5 call sprint
6 ; ----- Ввод 'B'
7 mov ecx,B
8 mov edx,10
9 call sread
0 ; ----- Преобразование 'B' из символа в число
1 mov eax,B
2 call atoi ; Вызов подпрограммы перевода символа в число
3 mov [B],eax ; запись преобразованного числа в 'B'
4 ; ----- Записываем 'A' в переменную 'max'
5 mov ecx,[A] ; 'ecx = A'
6 mov [max],ecx ; 'max = A'
```

Рис. 4.9: Ввод программы

Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными (рис. -fig. 4.10).



```
avluckaya@vbox: ~/work/arch-pc/lab07$ cd ~/lab7-2
avluckaya@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
avluckaya@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
avluckaya@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 2
Наибольшее число: 50
avluckaya@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 54
Наибольшее число: 54
avluckaya@vbox:~/work/arch-pc/lab07$
```

Рис. 4.10: 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 gedit (рис. -fig. 4.11).

```

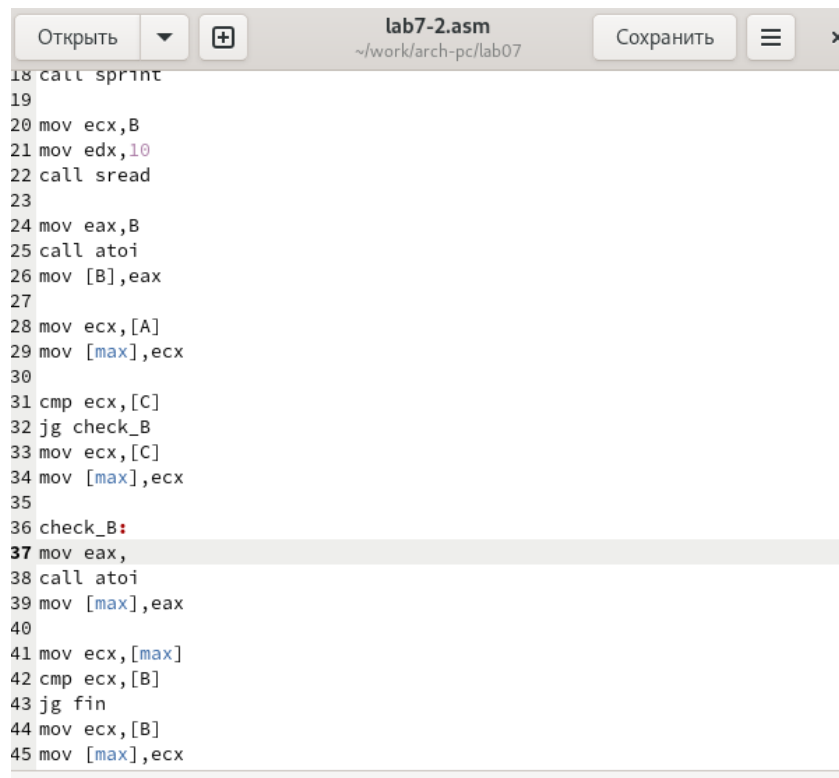
avluckaya@vbox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
avluckaya@vbox:~/work/arch-pc/lab07$ gedit lab7-2.lst

```

Line	Label	Hex	Comment	Instruction
1	1			%include 'in_out.asm'
2	1			<1> ;----- slen
3	2			<1> ; Функция вычисления длины
4	3		сообщения	<1> slen:
5	4	00000000 53		<1> push ebx
6	5	00000001 89C3		<1> mov ebx, eax
7	6			<1>
8	7			<1> nextchar:
9	8	00000003 803800		<1> cmp byte [eax], 0
10	9	00000006 7403		<1> jz finished
11	10	00000008 40		<1> inc eax
12	11	00000009 EBF8		<1> jmp nextchar
13	12			<1>
14	13			<1> finished:
15	14	0000000B 29D8		<1> sub eax, ebx
16	15	0000000D 5B		<1> pop ebx
17	16	0000000E C3		<1> ret
18	17			<1>

Рис. 4.11: Проверка файла листинга

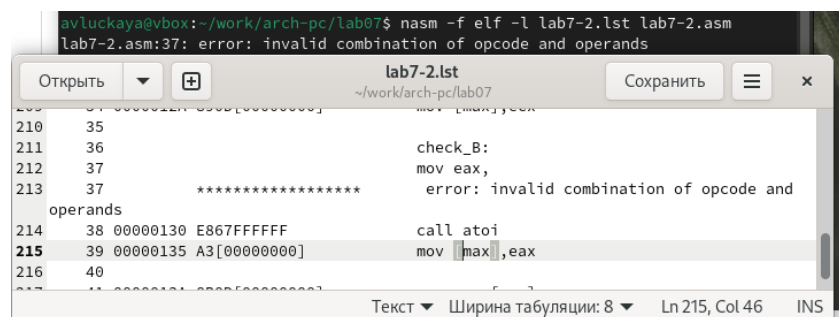
127 00000009C 53 <1> push ebx: Эта строка сохраняет содержимое регистра ebx на вершину стека. push — это инструкция, которая уменьшает указатель стека на 4 байта (размер регистра) и записывает значение регистра ebx в освободившееся место в памяти 128 00000009D 51 <1> push esx: Аналогично, эта строка сохраняет содержимое регистра esx на вершину стека. Значение esx записывается на вершину стека поверх значения ebx. 136 0000000AC 31DB <1> xor ebx, ebx: Эта команда устанавливает регистр ebx в ноль. xor — это побитовая операция XOR (исключающее ИЛИ). При применении xor к регистру самому себе он обнуляется. Удаляю один операнд из случайной инструкции(рис. -fig. 4.12).



```
18 call sprint
19
20 mov ecx, B
21 mov edx, 10
22 call sread
23
24 mov eax, B
25 call atoi
26 mov [B], eax
27
28 mov ecx, [A]
29 mov [max], ecx
30
31 cmp ecx, [C]
32 jg check_B
33 mov ecx, [C]
34 mov [max], ecx
35
36 check_B:
37 mov eax,
38 call atoi
39 mov [max], eax
40
41 mov ecx, [max]
42 cmp ecx, [B]
43 jg fin
44 mov ecx, [B]
45 mov [max], ecx
```

Рис. 4.12: 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. -fig. 4.13).



```
avluckaya@vbox: ~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:37: error: invalid combination of opcode and operands

lab7-2.lst
~/work/arch-pc/lab07
Сохранить
x

210 35
211 36 check_B:
212 37 mov eax,
213 37 ***** error: invalid combination of opcode and
operands
214 38 00000130 E867FFFFFF call atoi
215 39 00000135 A3[00000000] mov [max],eax
216 40
217 41 *****

Текст Ширина табуляции: 8 Ln 215, Col 46 INS
```

Рис. 4.13: Просмотр ошибки в файле листинга

4.3 Задания для самостоятельной работы

Выбираю вариант полученный в ходе выполнения лабораторной работы №6, мой вариант- 11.

С помощью команды touch создаю новый файл lab7-3.asm. Пишу программу для нахождения наименьшего числа из трех (рис. -fig. 4.14).



```
1 %include 'in_out.asm'
2
3 section .data
4 msg1 db 'Введите B: ',0h
5 msg2 db "Наименьшее число: ",0h
6 A dd '21'
7 C dd '34'
8
9 section .bss
10 min resb 10
11 B resb 10
12
13 section .text
14 global _start
15 _start:
16
17 mov eax,msg1
18 call sprint
19
20 mov ecx,B
21 mov edx,10
22 call sread
23
24 mov eax,B
25 call atoi
26 mov [B],eax
27
28 mov ecx,[A]
29 mov [min],ecx
30
31 cmp ecx,[C]
32 jl check_B
33 mov ecx,[C]
34 mov [min],ecx
35
36 check_B:
37 mov eax, min
38 call atoi
39 mov [min],eax
40
41 mov ecx,[min]
42 cmp ecx,[B]
43 jl fin
44 mov ecx,[B]
45 mov [min],ecx
46
```

Рис. 4.14: Первая программа самостоятельной работы

Проверяю корректность написания первой программы. Программа работает верно (рис. -fig. 4.15).

```

avluckaya@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
avluckaya@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
avluckaya@vbox:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 28
Наименьшее число: 21
avluckaya@vbox:~/work/arch-pc/lab07$ █

```

Рис. 4.15: Проверка работы первой программы

Код первой программы:

```

#include 'in_out.asm'

section .data
msg1 db 'Введите B: ',0h
msg2 db "Наименьшее число: ",0h
A dd '21'
C dd '34'

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]
jl check_B
mov ecx,[C]
mov [min],ecx

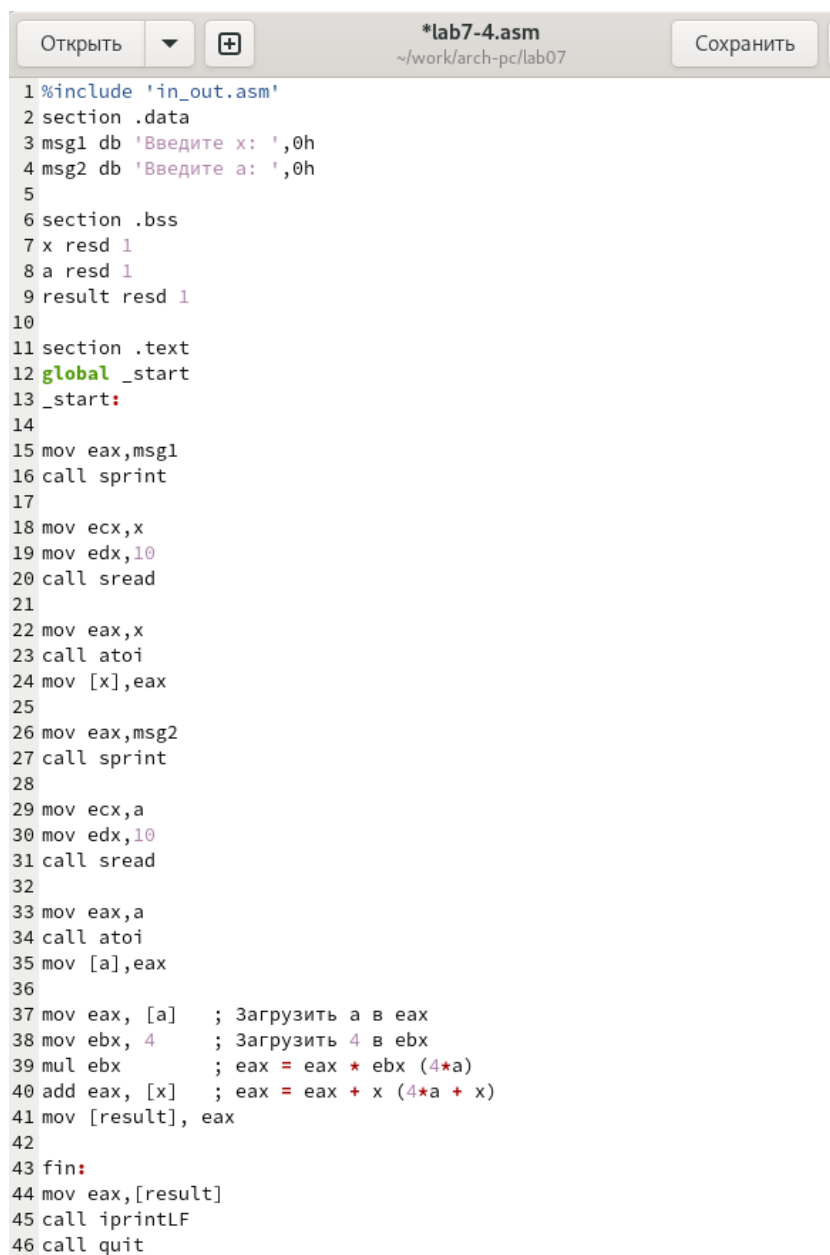
check_B:
mov eax,min
call atoi
mov [min],eax

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

```

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатуры переменных а и х (рис. -fig. 4.16).



```
1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите x: ',0h
4 msg2 db 'Введите a: ',0h
5
6 section .bss
7 x resd 1
8 a resd 1
9 result resd 1
10
11 section .text
12 global _start
13 _start:
14
15 mov eax,msg1
16 call sprint
17
18 mov ecx,x
19 mov edx,10
20 call sread
21
22 mov eax,x
23 call atoi
24 mov [x],eax
25
26 mov eax,msg2
27 call sprint
28
29 mov ecx,a
30 mov edx,10
31 call sread
32
33 mov eax,a
34 call atoi
35 mov [a],eax
36
37 mov eax, [a] ; Загрузить а в eax
38 mov ebx, 4 ; Загрузить 4 в ebx
39 mul ebx ; eax = eax * ebx (4*a)
40 add eax, [x] ; eax = eax + x (4*a + x)
41 mov [result], eax
42
43 fin:
44 mov eax,[result]
45 call iprintLF
46 call quit
```

Рис. 4.16: Вторая программа самостоятельной работы

Проверяю корректность написания первой программы. Программа работает верно (рис. -fig. 4.17).

```

avluckaya@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
avluckaya@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
avluckaya@vbox:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 0
Введите a: 3
12
avluckaya@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
avluckaya@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
avluckaya@vbox:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 1
Введите a: 2
9
avluckaya@vbox:~/work/arch-pc/lab07$

```

Рис. 4.17: Проверка работы второй программы

Код второй программы:

```

#include 'in_out.asm'

section .data
msg1 db 'Введите x: ',0h
msg2 db 'Введите a: ',0h

section .bss
x resd 1
a resd 1
result resd 1

section .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx,x
mov edx,10
call sread

```

```
mov eax,x
call atoi
mov [x],eax
```

```
mov eax,msg2
call sprint
```

```
mov ecx,a
mov edx,10
call sread
```

```
mov eax,a
call atoi
mov [a],eax
```

```
mov eax, [a]    ; Загрузить a в eax
mov ebx, 4      ; Загрузить 4 в ebx
mul ebx         ; eax = eax * ebx (4*a)
add eax, [x]    ; eax = eax + x (4*a + x)
mov [result], eax
```

```
fin:
mov eax,[result]
call iprintLF
call quit
```

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.

6 Список литературы

1. <https://esystem.rudn.ru/course/view.php?id=112>