

Лабораторная работа №7

**Команды безусловного и условного перехода в Nasm.
Программирование ветвлений.**

Полина Алексеевна Ларионова

Содержание

1	Цель работы	3
2	Выполнение лабораторной работы	4
3	Задания для самостоятельной работы	10
4	Выводы	13

1 Цель работы

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

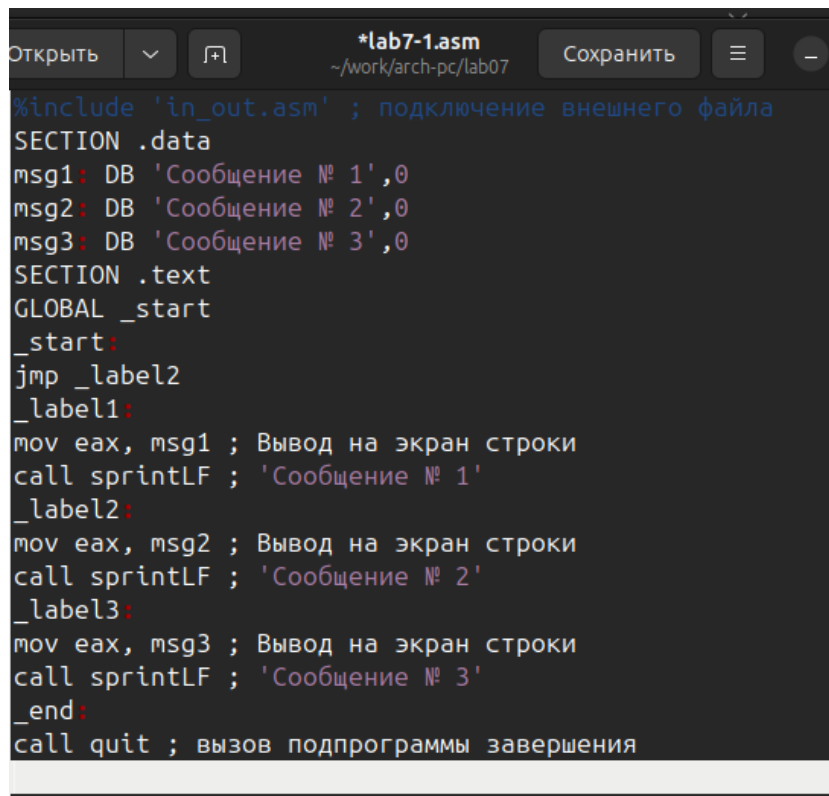
2 Выполнение лабораторной работы

Я создала каталог для программ лабораторной работы №7 и файл lab7-1.asm.

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

Рис. 2.1: каталог

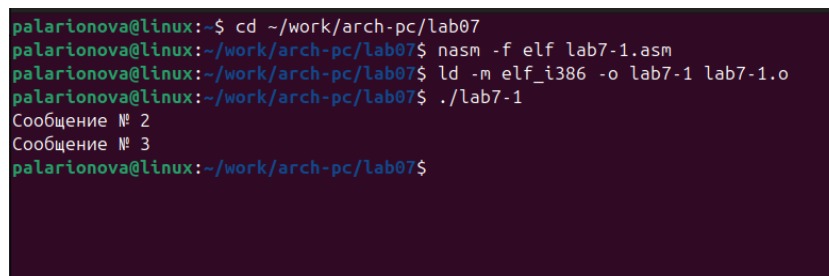
Далее я ввела в файл текст программы из листинга 7.1



```
Открыть  ▾  [F1]  *lab7-1.asm  Сохранить  ≡  -  
~/work/arch-pc/lab07  
%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
```

Рис. 2.2: lab7-1.asm

и проверила его работу, создав исполняемый файл.



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

Рис. 2.3: lab7-1

Дальше я изменила текст программы в соответствии с листингом 7.2

```

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
23

```

Рис. 2.4: Изменение текста

и создала исполняемый файл.

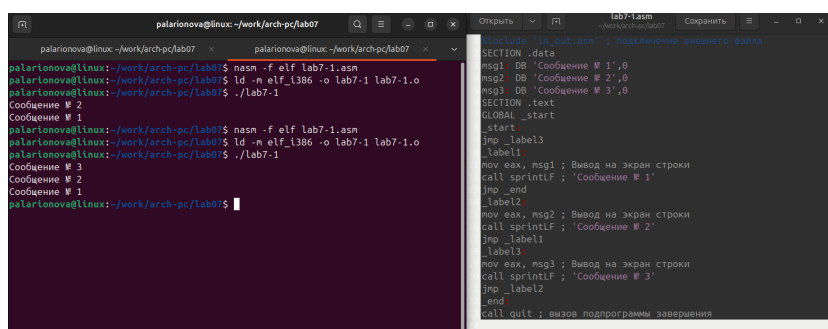
```

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

```

Рис. 2.5: Исполняемый файл

Затем я изменила текст программы еще раз, чтобы получить результат, указанный в лабораторной работе и снова создала исполняемый файл.



```

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

```

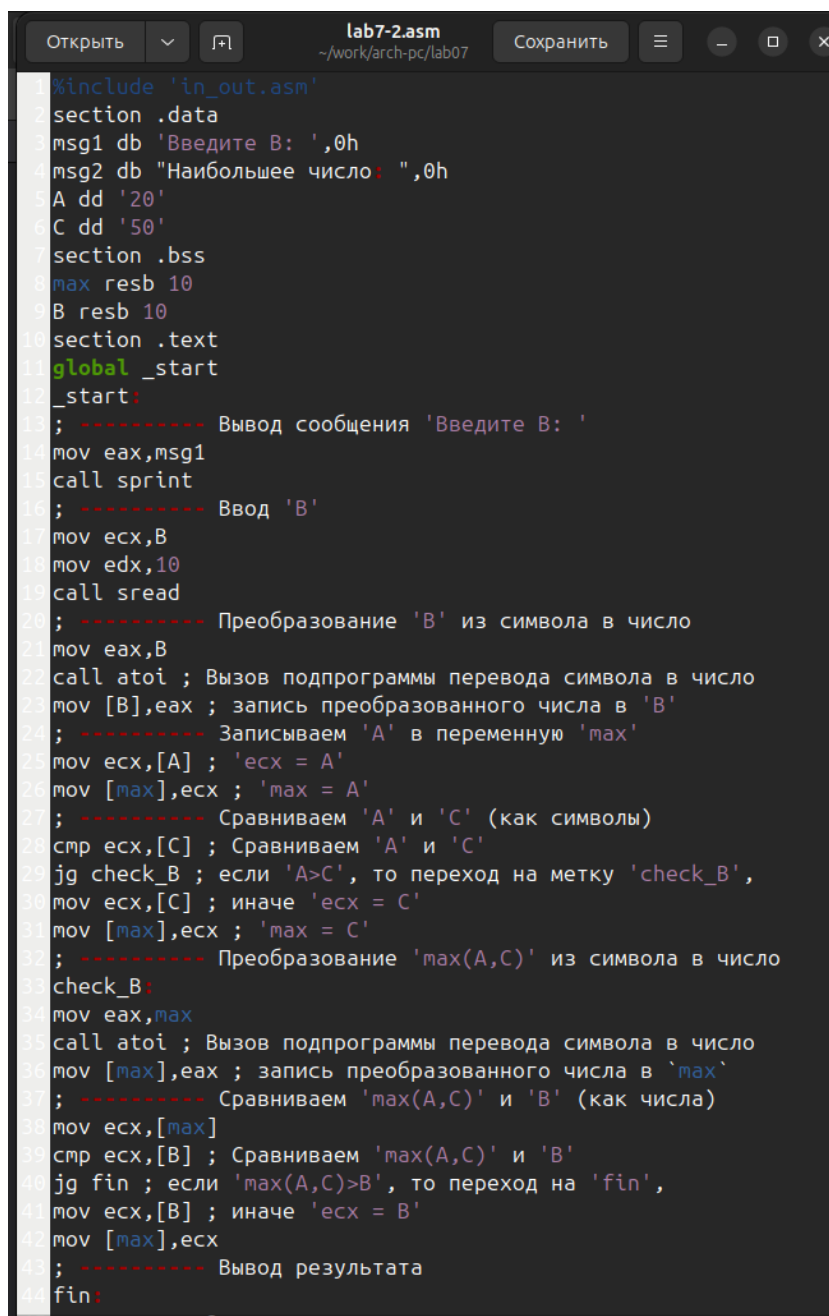
```

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 sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 2.6: Изменение в тексте

После этого я создала файл lab7-2.asm и ввела в него текст программы из листинга 7.3



```
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
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод 'B'
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,B
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A] ; 'ecx = A'
26 mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 jg check_B ; если 'A>C', то переход на метку 'check_B',
30 mov ecx,[C] ; иначе 'ecx = C'
31 mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,max
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [max],eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 mov ecx,[max]
39 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 mov ecx,[B] ; иначе 'ecx = B'
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
```

Рис. 2.7: lab7-2.asm

и создала исполняемый файл, проверив его работу, введя различные значения B.

```

palarionova@linux:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
palarionova@linux:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
palarionova@linux:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 3
Наибольшее число: 50
palarionova@linux:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
palarionova@linux:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 1
Наибольшее число: 50
palarionova@linux:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 80
Наибольшее число: 80
palarionova@linux:~/work/arch-pc/lab07$

```

Рис. 2.8: Работа программы для разных B

Я создала файл листинга для программы из файла lab7-2.asm и изучила его структуру, открыв в редакторе mcedit.

```

/home/palarionova/work/arch-pc/lab07/lab7-2.lst  [----] 0 L: [ 1+ 0 1/225] *(0 /14458b) 0032 0x020
1      %include "in_out.asm"
2      <1> ;----- slen -----
3      <1> ; Функция вычисления длины сообщения
4      <1> slen:-----
5      00000000 53      <1> push    ebx
6      00000001 89C3    <1> mov     ebx, eax
7      <1>-----
8      00000003 803800   <1> nextchar:-----
9      00000006 7403     <1> cmp     byte [eax], 0
10     00000008 40      <1> jz      finished
11     00000009 EBF8     <1> inc     eax
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> ;----- sprintf -----
20     <1> ; Функция печати сообщения
21     <1> ; входные данные: mov eax,<message>
22     <1> sprintf:-----
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 E8E8FFFF <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 B801000000 <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>-----

```

Рис. 2.9: lab7-2.lst

Проанализировав файл, я поняла, какие значения он выводит и как работает. Первая строка находится на 49 месте. Адрес этой строки "00000168", а машинный код - E86EFFFFFF. Call quit оканчивает работу программы и выходит из нее.


```
49 00000168 E86EFFFFFF      call quit ; Выход
```

Рис. 2.10: Первая строка

Вторая строка находится на 17 месте, ее адрес “000000F2”, машинный код - B9 [0A000000], а mov ecx, B - это исходный текст программы, который означает введение переменной B.

```
17 000000F2 B9[0A000000]      mov ecx, B
```

Рис. 2.11: Вторая строка

Третья строка находится на 35 месте. Её адрес “00000130”, машинный код - E867FFFFFF, call atoi - исходный текст программы, который означает перевод символа в лежащей выше строке в число.

```
35 00000130 E867FFFFFF      call atoi ; Вызов подпрограммы перевода
```

Рис. 2.12: Третья строка

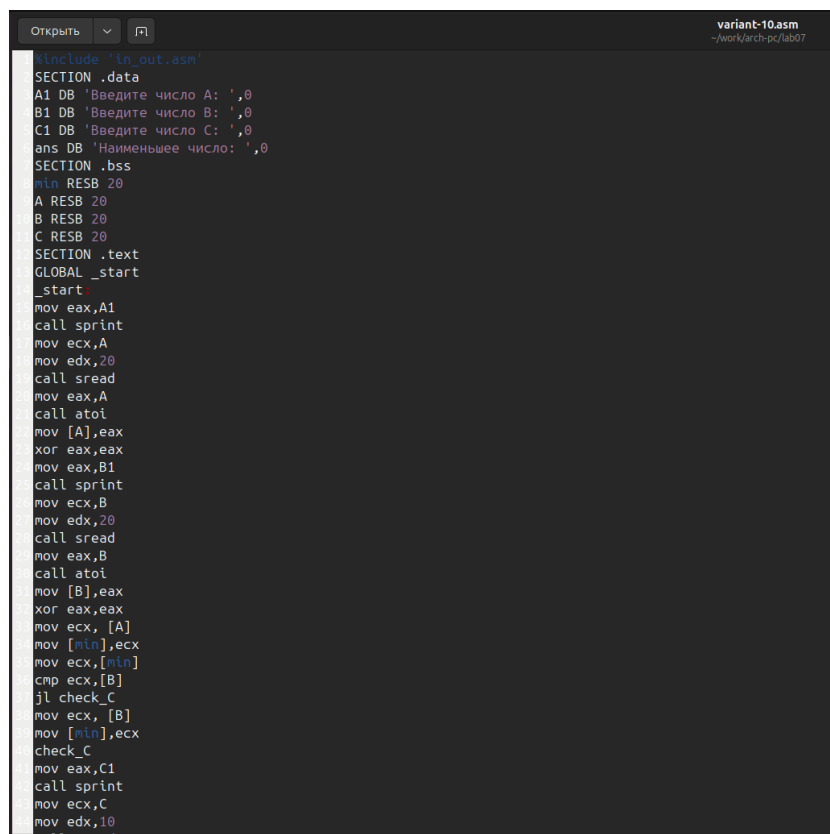
Открыв файл с программой lab7-2.asm, я удалила один из операндов в инструкции, но выходные файлы не создались, так как для работы программы требуется два операнда.

```
palarionova@linux:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:21: error: invalid combination of opcode and operands
palarionova@linux:~/work/arch-pc/lab07$
```

Рис. 2.13: Ошибка трансляции

3 Задания для самостоятельной работы

- 1) Я написала программу нахождения наименьшей из 3 целочисленных переменных в соответствии с полученным 10 вариантом.



```
1 include 'in_out.asm'
2 SECTION .data
3 A1 DB 'Введите число A: ',0
4 B1 DB 'Введите число B: ',0
5 C1 DB 'Введите число C: ',0
6 ans DB 'Наименьшее число: ',0
7 SECTION .bss
8 min RESB 20
9 A RESB 20
10 B RESB 20
11 C RESB 20
12 SECTION .text
13 GLOBAL _start
14 _start:
15 mov eax,A1
16 call sprintf
17 mov ecx,A
18 mov edx,20
19 call sread
20 mov eax,A
21 call atoi
22 mov [A],eax
23 xor eax,eax
24 mov eax,B1
25 call sprintf
26 mov ecx,B
27 mov edx,20
28 call sread
29 mov eax,B
30 call atoi
31 mov [B],eax
32 xor eax,eax
33 mov ecx,[A]
34 mov [min],ecx
35 mov ecx,[min]
36 cmp ecx,[B]
37 jl check_C
38 mov ecx,[B]
39 mov [min],ecx
40 check_C:
41 mov eax,C1
42 call sprintf
43 mov ecx,C
44 mov edx,10
```

Рис. 3.1: variant-10.asm

Затем я создала исполняемый файл и ввела числа, указанные в варианте 10.

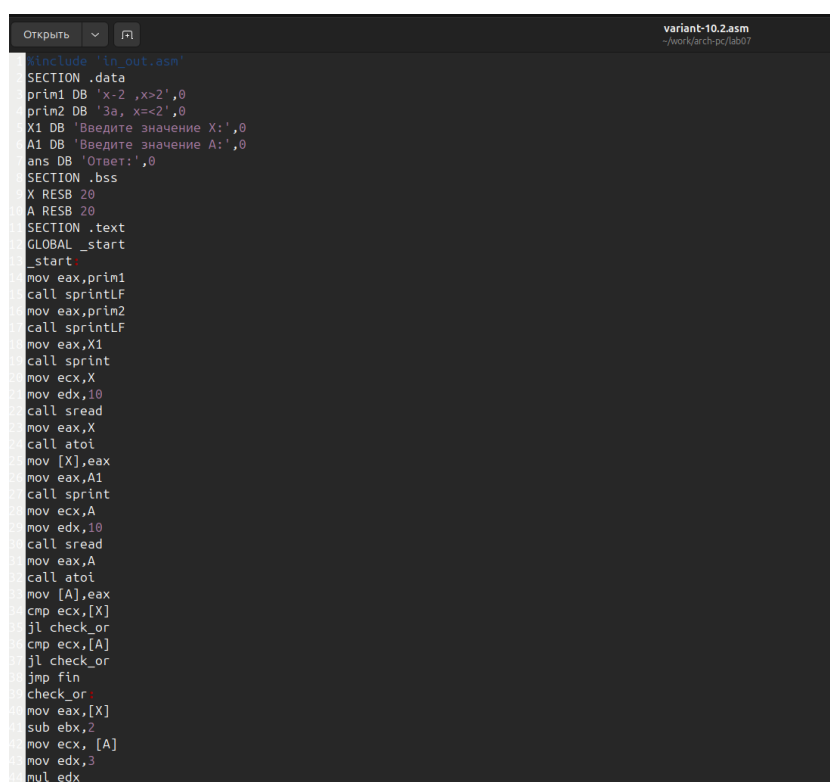
```

palarionova@linux:~/work/arch-pc/lab07$ nasm -f elf variant-10.asm
variant-10.asm:40: warning: label alone on a line without a colon might be in error [-w+label-orphan]
palarionova@linux:~/work/arch-pc/lab07$ ld -m elf_i386 -o variant-10 variant-10.o
palarionova@linux:~/work/arch-pc/lab07$ ./variant-10
Введите число A: 41
Введите число B: 62
Введите число C: 35
Наименьшее число: 35
palarionova@linux:~/work/arch-pc/lab07$

```

Рис. 3.2: Исполняемый файл

- 2) Я написала программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ в соответствии с заданием варианта 10.



```

variant-10.2.asm
~/work/arch-pc/lab07

#include 'in_out.asm'
SECTION .data
prim1 DB 'x<2',0
prim2 DB '3a, x<2',0
X1 DB 'Введите значение X:',0
A1 DB 'Введите значение A:',0
ans DB 'Ответ:',0
SECTION .bss
X RESB 20
A RESB 20
SECTION .text
GLOBAL _start
_start
mov eax,prim1
call sprintf
mov eax,prim2
call sprintf
mov eax,X1
call sprintf
mov ecx,X
mov edx,10
call sread
mov eax,X
call atoi
mov [X],eax
mov eax,A1
call sprintf
mov ecx,A
mov edx,10
call sread
mov eax,A
call atoi
mov [A],eax
cmp ecx,[X]
jl check_or
jl check_or
jmp fin
check_or
mov eax,[X]
sub ebx,2
mov ecx,[A]
mov edx,3
mul edx

```

Рис. 3.3: variant-10.2.asm

Далее я создала исполняемый файл и ввела значения для x и a , указанные в варианте 10.

```
palarionova@linux:~/work/arch-pc/lab07$ nasm -f elf variant-10.2.asm
palarionova@linux:~/work/arch-pc/lab07$ ld -m elf_i386 -o variant-10.2 variant-1
0.2.o
palarionova@linux:~/work/arch-pc/lab07$
palarionova@linux:~/work/arch-pc/lab07$
palarionova@linux:~/work/arch-pc/lab07$ ./variant-10.2
x-2 ,x>2
За, x=<2
Введите значение X:3
Введите значение A:0
Ответ:3
palarionova@linux:~/work/arch-pc/lab07$ ./variant-10.2
x-2 ,x>2
За, x=<2
Введите значение X:1
Введите значение A:2
Ответ:1
```

Рис. 3.4: Исполняемый файл

4 Выводы

При выполнении лабораторной работы №7 я изучила команды условного и безусловного перехода, а также научилась писать программы с переходами.