

Лабораторная работа №7

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

Богату Ирина Владимировна

Содержание

Цель работы	5
Выполнение лабораторной работы	6
Выполнение задания для самостоятельной работы	17

Список иллюстраций

1	Создание рабочей директории и файла lab7-1.asm	6
2	Запуск Midnight commander	6
3	Вставка кода из файла листинга 7.1	7
4	Копирование файла in_out.asm в рабочую директорию	7
5	Сборка программы из файла lab7-1.asm и её запуск	8
6	Изменение файла lab7-1.asm согласно листингу 7.2	8
7	Повторная сборка программы из файла lab7-1.asm и её запуск . .	9
8	Редактирование файла lab7-1.asm	9
9	Повторная сборка программы из файла lab7-1.asm и её запуск . .	10
10	Создание второго файла: lab7-2.asm	10
11	Запись кода из листинга 7.3 в файл lab7-2.asm	11
12	сборка программы из файла lab7-2.asm и её запуск	11
13	Создание файла листинга из файла lab6-2.asm	12
14	Открытие файла листинга в текстовом редакторе	13
15	Вид файла листинга	14
16	Нахождение нашей программы в файле листинга	14
17	Изменение исходного файла	15
18	Вывод ошибки при сборке объектного файла	15
19	Отображение ошибки в листинге	16
1	Создание первого файла самостоятельной работы	17
2	Код первого файла самостоятельной работы	18
3	Код первого файла самостоятельной работы (продолжение) . . .	18
4	Сборка и запуск программы первого задания самостоятельной ра- боты, а также результат выполнения	18
5	Создание второго файла самостоятельной работы	19
6	Код второго файла самостоятельной работы	20
7	Код второго файла самостоятельной работы (продолжение)	20
8	Сборка и тестирование второго файла самостоятельной работы .	20

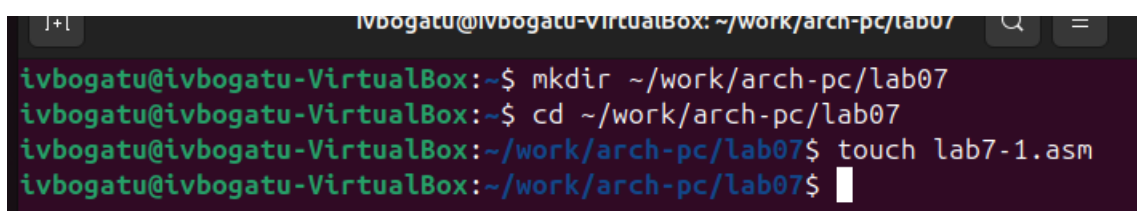
Список таблиц

Цель работы

Понять принцип работы условных и безусловных переходов в Ассемблере и научиться писать программы с командами, отвечающими за переходы. Научиться работать с файлами листинга и уметь их читать.

Выполнение лабораторной работы

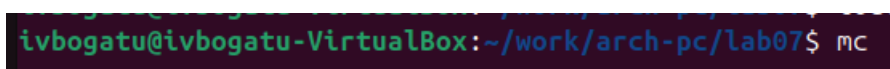
Для начала выполнения лабораторной работы необходимо создать рабочую папку lab07 и файл lab7-1.asm (рис. 2.1):



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

Рис. 1: Создание рабочей директории и файла lab7-1.asm

После чего, для удобства, запустить Midnight commander (рис. 2.2):



```
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ mc
```

Рис. 2: Запуск Midnight commander

Вставим код в файл lab7-1.asm из файла листинга (рис. 2.3):

```
GNU nano 7.2 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 _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Рис. 3: Вставка кода из файла листинга 7.1

Теперь скопируем файл `in_out.asm` из рабочей директории прошлой лабораторной работы (рис. 2.4):

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

Рис. 4: Копирование файла `in_out.asm` в рабочую директорию

Теперь соберём программу из файла `lab7-1.asm` и запустим её (рис. 2.5):

```
GNU nano 7.2                                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 _label2
_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'
[ Read 22 lines ]
```

Рис. 5: Сборка программы из файла lab7-1.asm и её запуск

Изменим файл lab7-1.asm согласно листингу 7.2 (рис. 2.6):

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

Рис. 6: Изменение файла lab7-1.asm согласно листингу 7.2

Снова соберём программу и запустим её (рис. 2.7):


```
GNU nano 7.2                                lab7-1.asm
#include 'in_out.asm'
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
call sprintLF
jmp _end
_label2:
mov eax, msg2
call sprintLF
jmp _label1
_label3:
mov eax, msg3
call sprintLF
```

[Read 23 lines]

Рис. 7: Повторная сборка программы из файла lab7-1.asm и её запуск

Теперь сделаем так, чтобы код выводил сообщения в обратном порядке (от 3 сообщения к первому). Для этого внесём в код следующие изменения (рис. 2.8):

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

Рис. 8: Редактирование файла lab7-1.asm

И запустим её, предварительно собрав (рис. 2.9):

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

Рис. 9: Повторная сборка программы из файла lab7-1.asm и её запуск

Теперь создадим файл lab7-2.asm (рис. 2.10):

```
GNU nano 7.2 lab7-2.asm
#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
[ Read 49 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

Рис. 10: Создание второго файла: lab7-2.asm

Запишем код из листинга 7.3 в файл lab7-2.asm (рис. 2.11):

```

ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 15
Наибольшее число: 50
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 35
Наибольшее число: 50
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 55
Наибольшее число: 55
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$

```

Рис. 11: Запись кода из листинга 7.3 в файл lab7-2.asm

И запустим его, предварительно собрав (рис. 2.12):

```

ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ mcedit lab7-2.lst

```

Рис. 12: сборка программы из файла lab7-2.asm и её запуск

Теперь попробуем создать файл листинга при сборке файла lab7-2.asm (рис. 2.13):

```
/home/iv~7-2.lst  [----]  0 L:[ 1+ 0  1/225] *(0  /14458b) 0032 0x020 [*][X]
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>
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit
```

Рис. 13: Создание файла листинга из файла lab6-2.asm

Теперь посмотрим, как выглядит файл листинга изнутри. Для этого откроем его в mcedit (рис. 2.14):

```
/home/iv~7-2.lst  [----]  0 L:[171+ 0 171/225] *(10588/14458b) 0032 0x020[*][X]
170 000000E7 C3 <1> ret
2 section .data
3 00000000 D092D0B2D0B5D0B4D0- msg1 db 'Введите B: ',0h
3 00000009 B8D182D0B520423A20-
3 00000012 00.....
4 00000013 D09DD0B0D0B8D0B1D0- msg2 db "Наибольшее число: ",0h
4 0000001C BED0BBD18CD188D0B5-
4 00000025 D0B520D187D0B8D181-
4 0000002E D0BBD0BE3A2000.....
5 00000035 32300000 A dd '20'
6 00000039 35300000 C dd '50'
7 section .bss
8 00000000 <res Ah> max resb 10
9 0000000A <res Ah> B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B:
14 000000E8 B8[00000000] mov eax,msg1
15 000000ED E81DFFFFFF call sprint
16 ; ----- Ввод 'B'
17 000000F2 B9[0A000000] mov ecx,B
1 Help 2 Save 3 Mark 4 Replac 5 Copy 6 Move 7 Search 8 Delete 9 PullDn 10 Quit
```

Рис. 14: Открытие файла листинга в текстовом редакторе

Открыв его, мы видим следующую картину (рис. 2.15):

```
GNU nano 7.2 lab7-2.asm
#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
call sread
; ----- Преобразование 'B' из символа в число

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

Рис. 15: Вид файла листинга

Наша программа находится чуть ниже (рис. 2.16):

```
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab
7-2.asm
lab7-2.asm:18: error: invalid combination of opcode and operands
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 16: Нахождение нашей программы в файле листинга

Разберём несколько строк файла листинга:

1. Строка под номером 14 перемещает содержимое msg1 в регистр eax. Адрес указывается сразу после номера. Следом идёт машинный код, который представляет собой исходную ассемблированную строку в виде шестнадцатиричной системы. Далее идёт исходный код
2. 15-ая строка отвечает за вызов функции sprint. Она также имеет адрес и машинный код

3. Строка 17 отвечает за запись переменной B в регистр ecx. Как видно, все строки имеют номер, адрес, машинный код и исходный код.

Теперь попробуем намеренно допустить ошибку в нашем коде, убрав у команды move 1 операнд (рис. 2.17):

```
/home/iv~7-2.lst  [----] 38 L:[179+ 2 181/226] *(11117/14546b) 0032 0x020[*][X]
 4 0000002E D0BBD0BE3A2000.....
 5 00000035 32300000                A dd '20'
 6 00000039 35300000                C dd '50'
 7                                     section .bss
 8 00000000 <res Ah>                max resb 10
 9 0000000A <res Ah>                B resb 10
10                                     section .text
11                                     global _start
12                                     _start:
13                                     ; ----- Вывод сообщения 'Введите B:
14 000000E8 B8[00000000]            mov eax,msg1
15 000000ED E81DFFFFFF            call sprint
16                                     ; ----- Ввод 'B'
17 000000F2 B9[0A000000]            mov ecx,B
18                                     mov edx
18                                     ***** error: invalid combination of opcode an
19 000000F7 E847FFFFFF            call sread
20                                     ; ----- Преобразование 'B' из симво
21 000000FC B8[0A000000]            mov eax,B
22 00000101 E896FFFFFF            call atoi ; Вызов подпрограммы перевода
23 00000106 A3[0A000000]            mov [B],eax ; запись преобразованного чи
24                                     ; ----- Записываем 'A' в переменную
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit
```

Рис. 17: Изменение исходного файла

И попробуем собрать файл с ошибкой, генерируя файл листинга (рис. 2.18):

```
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ touch tasl1v9.asm
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ S
```

Рис. 18: Вывод ошибки при сборке объектного файла

Мы видим, что объектный файл не создался, однако появился файл листинга. Теперь зайдём в файл листинга, и посмотрим, отображается ли в нём ошибка (рис. 2.19):

```
GNU nano 7.2          taslv9.asm *
#include 'in_out.asm'
section .data
msg2 db "Наименьшее число: ",0h
A dd '24'
B dd '98'
C dd '15'
section .bss
min resb 10
section .text
global _start
_start:
; ----- 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'

mov eax,A
call atoi ; Вызов подпрограммы перевода символа в число
mov [A],eax ; запись преобразованного числа в A

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Рис. 19: Отображение ошибки в листинге

Как видим, в листинге прописана ошибка

Выполнение задания для самостоятельной работы

Создадим файл для выполнения самостоятельной работы. Мой вариант - 9 (рис. 3.1):

```
GNU nano 7.2          tasl1v9.asm *
jl check_B ; если 'A<C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число

check_B:
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
jl fin ; если 'min(A,C)<B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprintf ; Вывод сообщения 'Наименьшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход
```

Рис. 1: Создание первого файла самостоятельной работы

Напишем код для выполнения задания. Код выглядит так (рис. 3.2 и рис. 3.3):

```

ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf task1v9.asm
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o task1v9 task1v9.o
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ ./task 1v9
bash: ./task: No such file or directory
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ ./task1v9
Наименьшее число: 15
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$

```

Рис. 2: Код первого файла самостоятельной работы

```

ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ touch task2v9.asm
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$

```

Рис. 3: Код первого файла самостоятельной работы (продолжение)

Соберём, запустим его и посмотрим на результат (рис. 3.4):

```

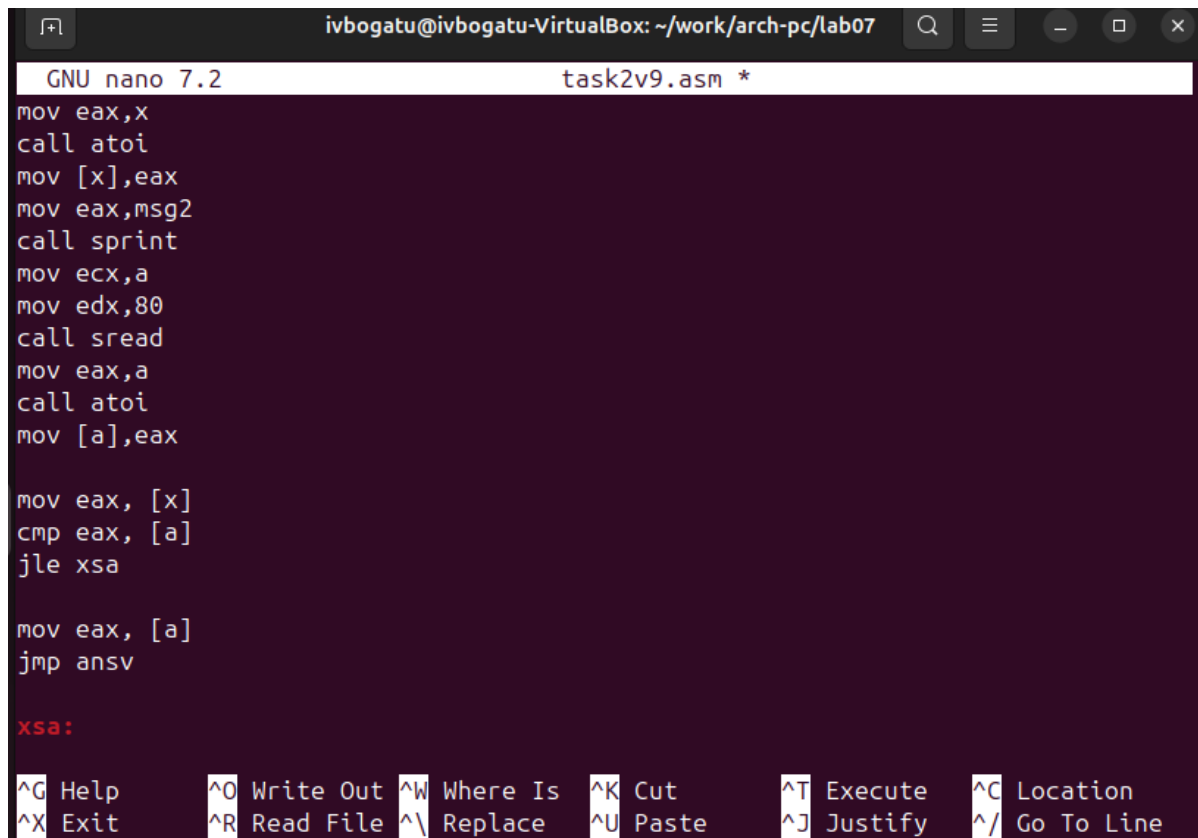
GNU nano 7.2 task2v9.asm *
%include 'in_out.asm'
section .data
msg1 DB "Введите X: ",0h
msg2 DB "Введите A: ",0h
msg3 DB "Ответ=",0h
section .bss
x: RESB 80
a: RESB 80
ans: RESB 80
section .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,x
mov edx,80
call sread
mov eax,x
call atoi
mov [x],eax

```

[^]G Help [^]O Write Out [^]W Where Is [^]K Cut [^]T Execute [^]C Location
[^]X Exit [^]R Read File [^]\ Replace [^]U Paste [^]J Justify [^]/ Go To Line

Рис. 4: Сборка и запуск программы первого задания самостоятельной работы, а также результат выполнения

Теперь создадим второй файл самостоятельной работы для второго задания (рис. 3.5):



```
GNU nano 7.2 task2v9.asm *
mov eax,x
call atoi
mov [x],eax
mov eax,msg2
call sprint
mov ecx,a
mov edx,80
call sread
mov eax,a
call atoi
mov [a],eax

mov eax, [x]
cmp eax, [a]
jle xsa

mov eax, [a]
jmp ansv

xsa:
```

Terminal window showing GNU nano 7.2 editor editing task2v9.asm. The code contains assembly instructions for memory operations and control flow. The terminal window title is ivbogatu@ivbogatu-VirtualBox: ~/work/arch-pc/lab07. The bottom status bar shows keyboard shortcuts: ^G Help, ^O Write Out, ^W Where Is, ^K Cut, ^T Execute, ^C Location, ^X Exit, ^R Read File, ^\ Replace, ^U Paste, ^J Justify, ^_ Go To Line.

Рис. 5: Создание второго файла самостоятельной работы

Код будет выглядеть так (рис. 3.6 и рис. 3.7):

```

    jmp ansv

xsa:
mov eax, [a]
add eax, [x]

ansv:
mov [ans],eax
mov eax,msg3
call sprint
mov eax,[ans]
call iprintLF
call quit

```

Рис. 6: Код второго файла самостоятельной работы

```

ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf task2v9.asm
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o task2v9 task2v9.o
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ ./task2v9
Введите X: 5
Введите A: 7
Ответ=12
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$

```

Рис. 7: Код второго файла самостоятельной работы (продолжение)

Соберём исполняемый файл и запустим его (рис. 3.8):

```

ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf task2v9.asm
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o task2v9 task2v9.o
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$ ./task2v9
Введите X: 5
Введите A: 7
Ответ=12
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab07$

```

Рис. 8: Сборка и тестирование второго файла самостоятельной работы

```
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc
k2v9.o
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc
Введите X: 5
Введите A: 7
Ответ=12
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc
```

Как видим, программа всё посчитала правильно

Выводы

В результате работы над лабораторной работой были написаны программы, которые используют команды условных и безусловных переходов, были получены навыки работы с этими командами, а также были созданы и успешно прочитаны листинги для некоторых из программ.