

Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Гафоров Нурмухамад Вомикович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Вывод	17
	Список литературы	18

Список иллюстраций

4.1	Создание директории	8
4.2	Переходили в каталог	8
4.3	Создание файла	8
4.4	Создание копии файла	9
4.5	Открыли каталог	9
4.6	входили в каталог	9
4.7	Редактирование файла	10
4.8	Открытие файла для просмотра	10
4.9	Запуск исполняемого файла	11
4.10	Редактирование файла	11
4.11	Создание файла	11
4.12	Редактирование файла	12
4.13	Запуск исполняемого файла	12
4.14	Написали код	12
4.15	Открыл файл листинга	13
4.16	Написали код	13
4.17	Удалил один операнд	14
4.18	Удалил один операнд и программа показывает нам ошибку	14
4.19	Написание програми	15
4.20	Запуск файла и проверка его работы	15
4.21	Написание программы	16
4.22	Запускаем файла и проверяем	16

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 файлы 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

##Реализация переходов в NASM

С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №7 (рис. [4.1])

```
nvgaforov@dk8n51 ~/work/arch-pc $ mkdir ~/work/arch-pc/lab07
```

Рис. 4.1: Создание директории

Перехожу в созданный каталог с помощью утилиты `cd`. (рис. [4.2])

```
nvgaforov@dk8n51 ~/work/arch-pc $ cd ~/work/arch-pc/lab07
nvgaforov@dk8n51 ~/work/arch-pc/lab07 $
```

Рис. 4.2: Переходили в каталог

С помощью утилиты `touch` создаю файл `lab7-1.asm` (рис. [4.3])

```
nvgaforov@dk8n51 ~/work/arch-pc/lab07 $ touch lab7-1.asm
nvgaforov@dk8n51 ~/work/arch-pc/lab07 $
```

Рис. 4.3: Создание файла

Копирую в текущий каталог файл `in_out.asm`, т.к. он будет использоваться в других программах (рис. [4.4]).

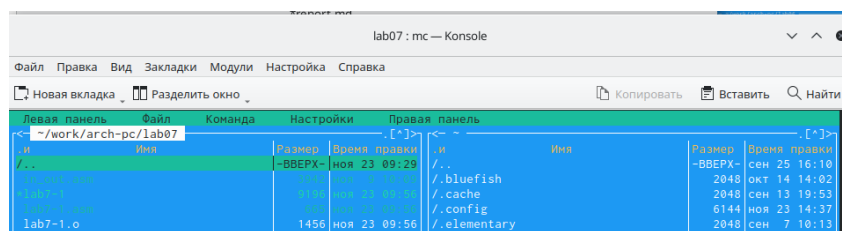


Рис. 4.4: Создание копии файла

с помощью mc откроем созданный файл lab7-1.asm (рис. [4.5]).

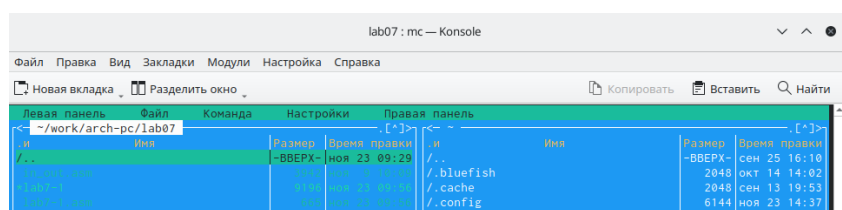


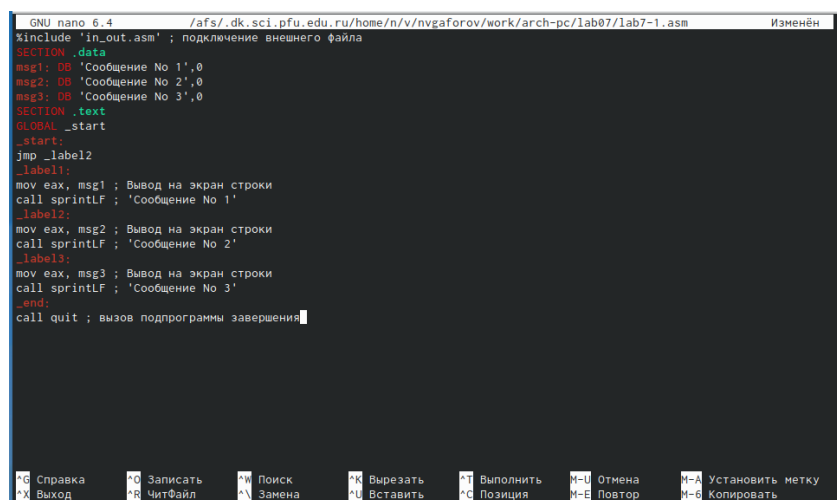
Рис. 4.5: Открыли каталог

Входим в созданный каталог lab7-1.asm (рис. [4.6]).



Рис. 4.6: входили в каталог

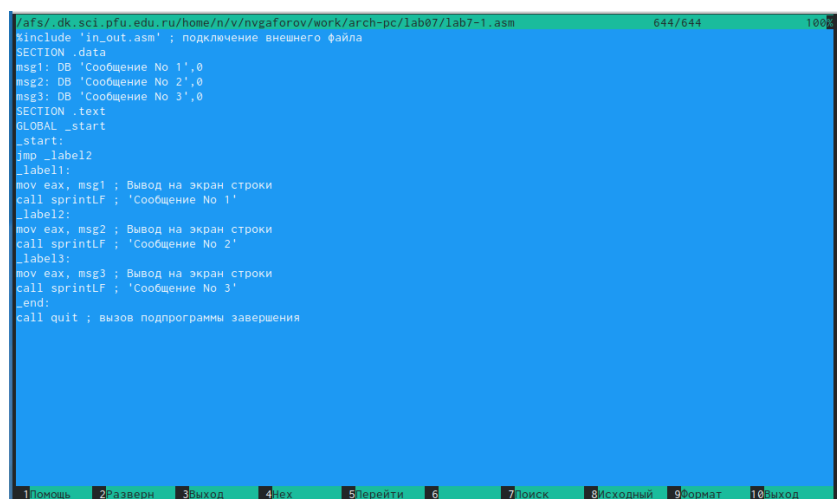
Открываю созданный файл lab7-1.asm, вставляю в него программу вывода значения регистра eax (рис. [4.7]).



```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/work/arch-pc/lab07/lab7-1.asm
#include "in_out.asm" ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.7: Редактирование файла

С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы. Открытие файла для просмотра (рис. [4.8]).



```
/afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/work/arch-pc/lab07/lab7-1.asm 644/644 100%
#include "in_out.asm" ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.8: Открытие файла для просмотра

Создаю исполняемый файл программы и запускаю его (рис. [4.9]). Вывод программы: Сообщение № 2 Сообщение № 3.

```
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 3
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $
```

Рис. 4.9: Запуск исполняемого файла

Изменяем текст программы и проверяем его работу как он работает (рис. [4.10]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/work/arch-pc/lab07/lab7-1.asm Изменён
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:
call quit ; вызов подпрограммы завершения

Сохранить изменённый буфер?
Y Да
N Нет CQ Отмена
```

Рис. 4.10: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. [??]). Теперь

вывелся на экран.Сообщение № 2 Сообщение № 1

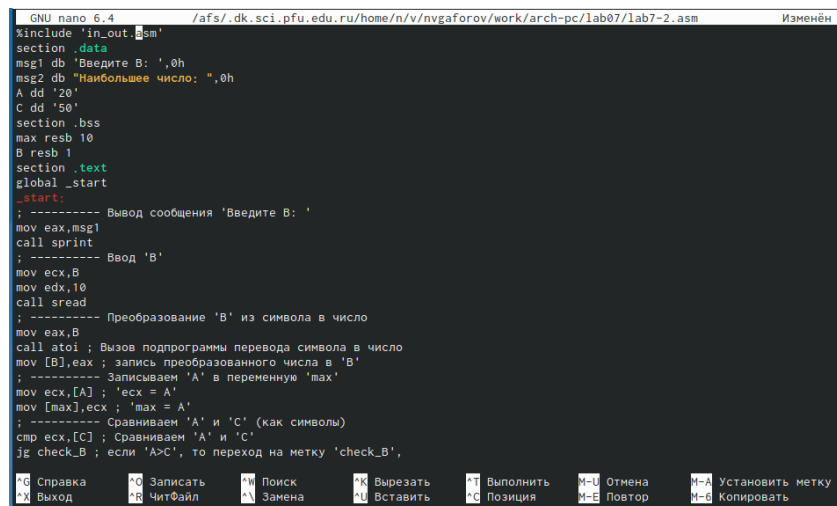
```
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 1
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $
```

Создаю новый файл lab7-2.asm с помощью утилиты touch (рис. [4.11]).

```
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $ touch lab7-2.asm
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $
```

Рис. 4.11: Создание файла

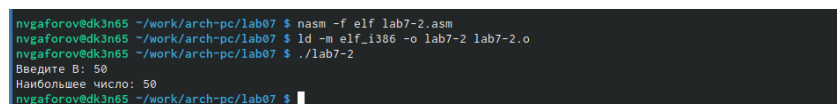
Ввожу в файл текст другой программы для вывода значения регистра (рис. [4.12]).



```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/work/arch-pc/lab07/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 1
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
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
je check_B ; если 'A>C', то переход на метку 'check_B',
;C Справка ;C Записать ;N Поиск ;N Вырезать ;N Выполнить ;N-U Отмена ;N-A Установить метку
;X Выход ;B ЧитФайл ;A Замена ;C Вставить ;C Позиция ;N-E Повтор ;N-B Копировать
```

Рис. 4.12: Редактирование файла

Создаю и запускаю исполняемый файл lab7-2 (рис. [4.13]). Теперь выводим 50 , на экран показывает Неиболе число 50

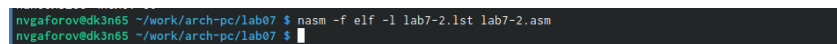


```
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 50
Наибольшее число: 50
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $
```

Рис. 4.13: Запуск исполняемого файла

##Изучение структуры файлы листинга

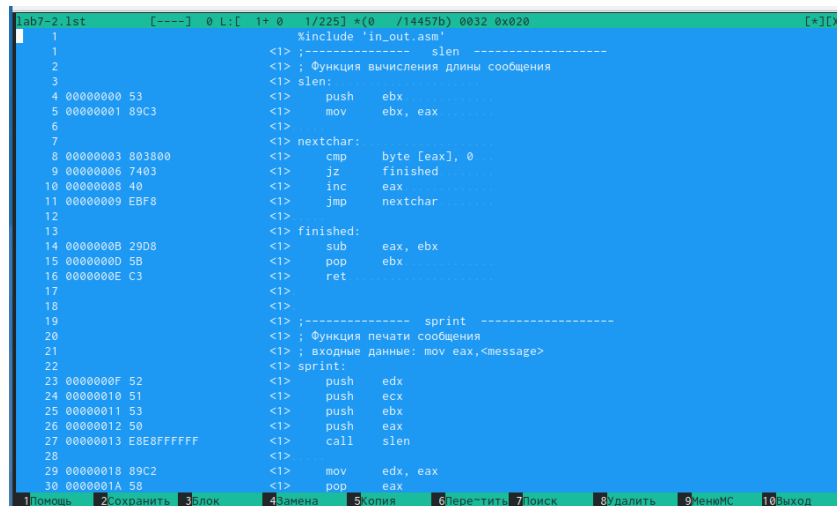
Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в командной строке. (рис. [4.14]).



```
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $
```

Рис. 4.14: Написали код

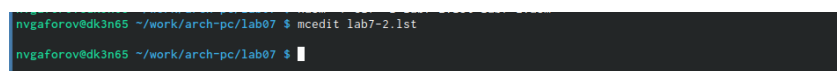
Создайте файл листинга для программы из файла lab7-2.asm (рис. [4.15]).



```
lab7-2.lst  [----]  0 L:[ 1+ 0 1/225] *(0 /14457b) 0032 0x020  [*][X]
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: 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 2908    <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
```

Рис. 4.15: Открыл файл листинга

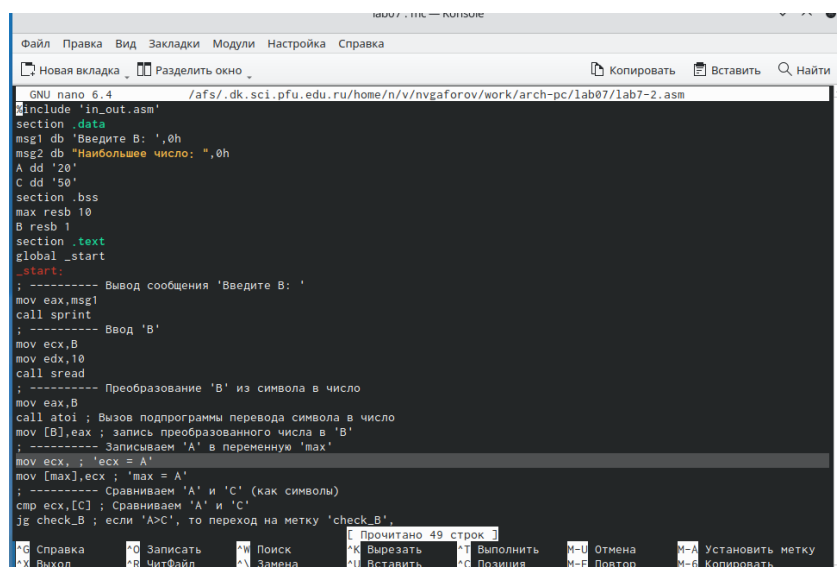
Откройте файл листинга lab7-2.lst с помощью любого текстового редактора, например mcedit:(рис. [4.16]).



```
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $ mcedit lab7-2.lst
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $
```

Рис. 4.16: Написали код

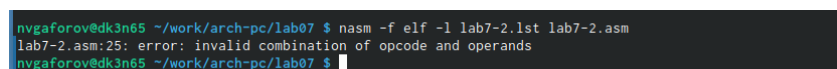
Откройте файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалить один операнд.(рис. [4.17]).



```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/work/arch-pc/lab07/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 1
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
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx, ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
je check_B ; если 'A>C', то переход на метку 'check_B',
```

Рис. 4.17: Удалил один операнд

Выполните трансляцию с получением файла листинга (рис. [4.18]).



```
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:25: error: invalid combination of opcode and operands
nvgaforov@dk3n65 ~/work/arch-pc/lab07 $
```

Рис. 4.18: Удалил один операнд и программа показывает нам ошибку

##Задания для самостоятельная работа

1. Пишу программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбираю из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Мой вариант под номером 5, поэтому мои значения - 54, 62 и 87. (рис. [4.19]).

```

GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/n/v/vnagaforov/work/arch-pc/lab07/lab7-3.asm
%include 'in_out.asm'
section .data
msg db "Наименьшее число: ",0h
A dd '54'
B dd '62'
C dd '87'
section .bss
min resb 10
section .text
global _start
_start:
; ----- Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в min
; ----- 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'

```

Рис. 4.19: Написание программы

Создаю исполняемый файл и проверяю его работу, подставляя необходимые значение.(рис. [4.20]).

```

nvgaforov@dk5n52 ~/work/arch-pc/lab07 $ touch lab7-3.asm
nvgaforov@dk5n52 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
nvgaforov@dk5n52 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
nvgaforov@dk5n52 ~/work/arch-pc/lab07 $ ./lab7-3
Наименьшее число: 54
nvgaforov@dk5n52 ~/work/arch-pc/lab07 $

```

Рис. 4.20: Запуск файла и проверка его работы

Программа работает

Код Программа:

%include 'in_out.asm' section .data msg db "Наименьшее число:",0h A dd '54' B dd '62' C dd '87' section .bss min resb 10 section .text global _start _start: ; ----- Записываем 'A' в переменную 'min' mov ecx,[A] ; 'ecx = A' mov [min],ecx ; 'min = A' ; ----- Сравниваем 'A' и 'C' (как символы) cmp ecx,[C] ; Сравниваем 'A' и 'C' jg check_B mov ecx,[C] ; иначе 'ecx = C' mov [min],ecx ; 'min = C' ; ----- Преобразование 'min(A,C)' из символа в число check_B: mov eax,min call atoi ; Вызов подпрограммы перевода символа в число mov [min],eax ; запись преобразованного числа в min ; ----- 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, msg call sprint ; Вывод сообщения ‘Наименьшее число:’ mov eax,[min] call iprintLF ; Вывод ‘min(A,B,C)’ call quit ; Выход

2. Пишу программу, которая для введенных с клавиатуры значений x и a вычисляет значение и выводит результат вычислений заданной для моего варианта функции f(x): (рис. [4.21]).

```

lab07: mc — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
[+] Новая вкладка [x] Разделить окно [K] Копировать [V] Вставить [F] Найти
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/work/arch-pc/lab07/lab7-4.asm Изменён
#include "in_out.asm"
section .data
vvdax: db "Введите x: ",0
vvoda: db "Введите a: ",0
vivod: db "Результат: ",0
section .bss
x: resb 80
a: resb 80
section .text
global _start
_start:
mov eax,vvdax
call sprint
mov ecx,x
mov edx,80
call sread
mov eax,x
call atoi
cmp eax,2
je _functionx
mov eax,vvoda
call sprint
mov ecx,a
mov edx,80
call sread
mov eax,a
call atoi
jmp _functiona
_functiona:

```

Рис. 4.21: Написание программы

Создаю исполняемый файл и проверяю его работу для значений x и a соответственно: (3;0), (1;2) (рис. [4.22]).

```

nvgaforov@dk5n52 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
nvgaforov@dk5n52 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
nvgaforov@dk5n52 ~/work/arch-pc/lab07 $ ./lab7-4
Введите x: 3
Результат: 1
nvgaforov@dk5n52 ~/work/arch-pc/lab07 $ ./lab7-4
Введите x: 1
Введите a: 2
Результат: 6
nvgaforov@dk5n52 ~/work/arch-pc/lab07 $

```

Рис. 4.22: Запускаем файла и проверяем

5 Вывод

По итогам данной лабораторной работы я изучила команды условного и безусловного переходов, приобрела навыки написания программ с использованием переходов и ознакомилась с назначением и структурой файла листинга, что поможет мне при выполнении последующих лабораторных работ.

Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс,
- 11.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
13. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
14. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ- Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-

- е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.
16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,
18. — 1120 с. — (Классика Computer Science).