

Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Толстых Александра Андреевна

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Реализация переходов в NASM	6
3.2	Изучение структуры файлы листинга	10
3.3	Задание для самостоятельной работы	12
4	Выводы	14

Список иллюстраций

3.1	Создание каталога и файла	6
3.2	Написание программы	6
3.3	Запуск программы	7
3.4	Изменение программы	7
3.5	Запуск программы	8
3.6	Изменение программы	8
3.7	Запуск программы	8
3.8	Написание программы	9
3.9	Запуск программы	9
3.10	Создания файла листинга	10
3.11	Изучение строки	10
3.12	Изучение строки	10
3.13	Изучение строки	10
3.14	Удаление операнда	11
3.15	Трансляция с получением файла листинга	11
3.16	Вывод содержимого каталога	11
3.17	Изменения в файле листинга	11
3.18	Написание программы	12
3.19	Создание исполняемого файла и его запуск	12
3.20	Написание программы	13
3.21	Создание исполняемого файла и его запуск	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 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлы листинга
3. Задание для самостоятельной работы

3 Выполнение лабораторной работы

3.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы №7, перехожу в него и создаю файл lab7-1.asm (рис. 3.1).

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

Рис. 3.1: Создание каталога и файла

Ввожу в созданный файл текст программы из листинга (рис. 3.2).



```
Open  ▾  lab7-1.asm
~/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 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
```

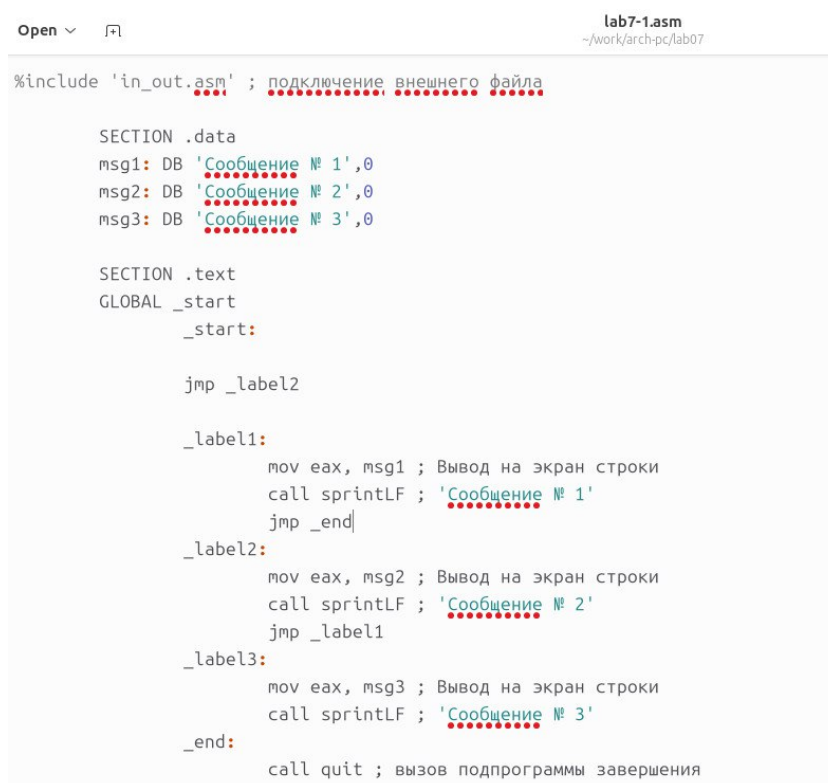
Рис. 3.2: Написание программы

Создаю исполняемый файл и запускаю его (рис. 3.3). Вывод программы корректный.

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

Рис. 3.3: Запуск программы

Изменяю программу так, чтобы она сначала выводила второе сообщение, затем первое (рис. 3.4).



```
lab7-1.asm
~/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 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'

_end:
    call quit ; вызов подпрограммы завершения
```

Рис. 3.4: Изменение программы

Создаю исполняемый файл и запускаю его (рис. 3.5). Вывод программы корректный.

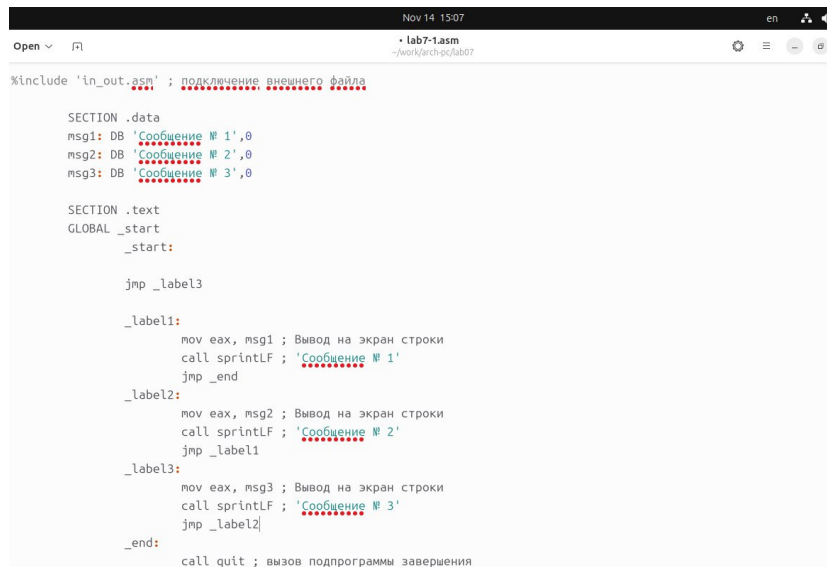
```

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

```

Рис. 3.5: Запуск программы

Изменяю программу так, чтобы она выводила все три сообщения, но в обратном порядке (рис. 3.6).



```

Nov 14 15:07
en
lab7-1.asm
~/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 _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. 3.6: Изменение программы

Создаю исполняемый файл и запускаю его (рис. 3.7). Вывод программы корректный.

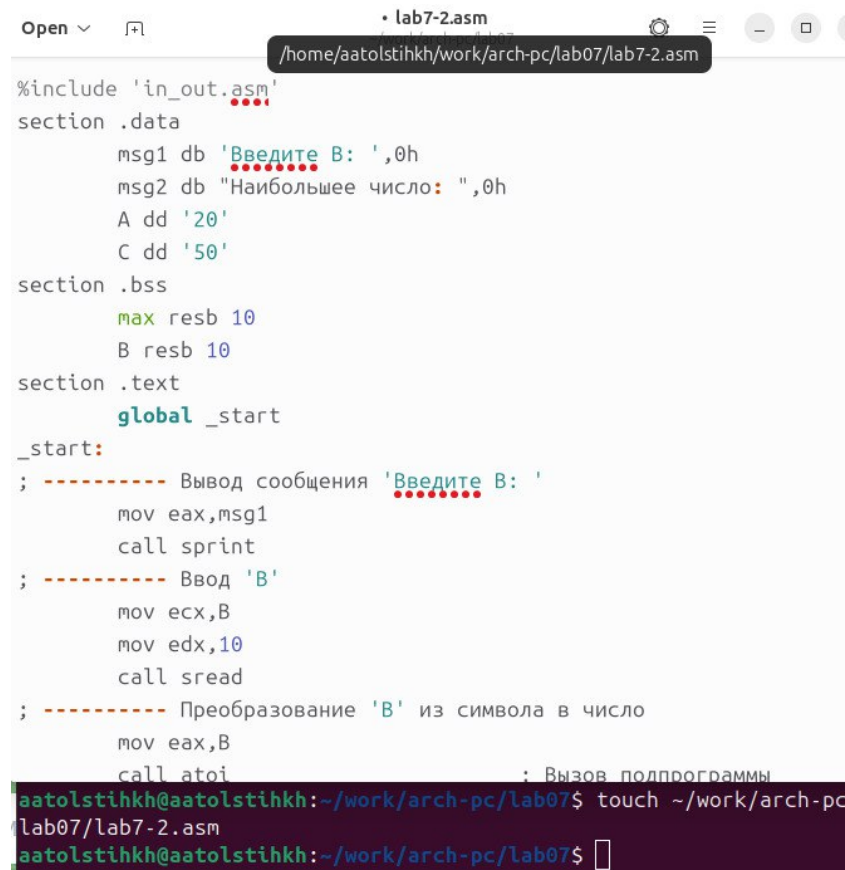
```

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

```

Рис. 3.7: Запуск программы

Создаю файл lab7-2 и ввожу в него указанный текст программы (рис. 3.8).

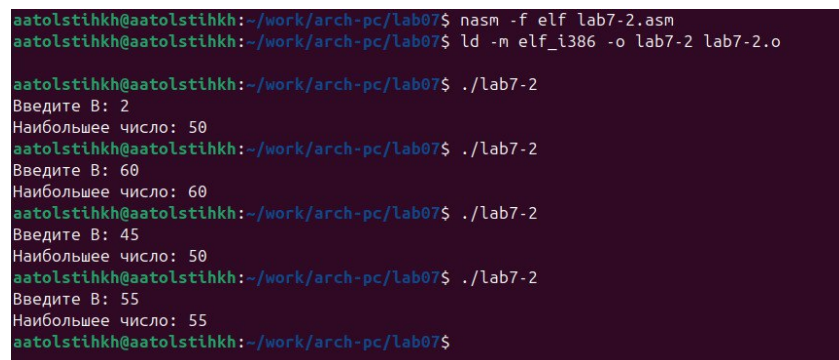


```
Open ▾  +  • lab7-2.asm  /home/aatolstikh/work/arch-pc/lab07/lab7-2.asm

%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 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
    mov eax,B
    call atoi
; ----- Вызов подпрограммы
aatolstikhk@aatolstikhk:~/work/arch-pc/lab07$ touch ~/work/arch-pc/
lab07/lab7-2.asm
aatolstikhk@aatolstikhk:~/work/arch-pc/lab07$
```

Рис. 3.8: Написание программы

Создаю исполняемый файл и запускаю его (рис. 3.9). Проверяю работу программы на нескольких значениях.



```
aatolstikhk@aatolstikhk:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aatolstikhk@aatolstikhk:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o

aatolstikhk@aatolstikhk:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 2
Наибольшее число: 50
aatolstikhk@aatolstikhk:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
aatolstikhk@aatolstikhk:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 45
Наибольшее число: 50
aatolstikhk@aatolstikhk:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 55
Наибольшее число: 55
aatolstikhk@aatolstikhk:~/work/arch-pc/lab07$
```

Рис. 3.9: Запуск программы

3.2 Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.asm (рис. 3.10).

```
Наибольшее число: 55
aato1stihkh@aato1stihkh:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
aato1stihkh@aato1stihkh:~/work/arch-pc/lab07$ mcodeit lab7-2.lst
```

Рис. 3.10: Создания файла листинга

После ознакомления с файлом и его содержимым, а затем выбираю три строки и изучаю их детально.

Первой рассматриваю строку номер 25 (рис. 3.11). Ее адрес “00000110”, Машинный код - 8B0D[35000000], а `mov ecx,[A]` - исходный текст программы, означающий что в регистр `ecx` мы вносим значения переменной `A`.

```
25 00000110 8B0D[35000000] mov ecx,[A]
```

Рис. 3.11: Изучение строки

Далее рассматриваю строку номер 35 (рис. 3.12). Ее адрес “00000135”, Машинный код - E862FFFFFF, а `call atoi` - исходный текст программы, означающий что символ, лежащий в строке выше, переводится в число.

```
35 00000135 E862FFFFFF call atoi
```

Рис. 3.12: Изучение строки

Последней строкой я изучаю строку номер 38 (рис. 3.13). Ее адрес “0000013F”, Машинный код - 8B0D[00000000], а `mov ecx,[max]` - исходный текст программы, означающий что в регистр `ecx` мы вносим значения переменной `max`.

```
38 0000013F 8B0D[00000000] mov ecx,[max]
```

Рис. 3.13: Изучение строки

Открываю файл программы и убираю второй операнд у одной из инструкций (рис. 3.14).

```

mov eax, msg2
call sprint                ; Вывод сообщения 'Наибольшее число: '
mov eax                   ; Убран второй аргумент [max]
call iprintLF             ; Вывод 'max(A,B,C)'
call quit                 ; Выход

```

Рис. 3.14: Удаление операнда

Выполняю трансляцию файла с получением файла листинга (рис. 3.15).

```

aamolstihkh@aamolstihkh:~/work/arch-pc/lab07$
aamolstihkh@aamolstihkh:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:47: error: invalid combination of opcode and operands
aamolstihkh@aamolstihkh:~/work/arch-pc/lab07$

```

Рис. 3.15: Трансляция с получением файла листинга

Подробно вывожу содержимое каталога, чтобы понять какие файлы создались (рис. 3.16). После возникновения ошибки создан только файл листинга.

```

aamolstihkh@aamolstihkh:~/work/arch-pc/lab07$ ls -la
total 64
drwxrwxr-x 2 aamolstihkh aamolstihkh 4096 Nov 14 16:10 .
drwxrwxr-x 4 aamolstihkh aamolstihkh 4096 Nov 14 15:10 ..
-rw-rw-r-- 1 aamolstihkh aamolstihkh 3942 Nov 10 01:04 in_out.asm
-rwxrwxr-x 1 aamolstihkh aamolstihkh 9200 Nov 14 15:20 lab7-1
-rw-rw-r-- 1 aamolstihkh aamolstihkh 737 Nov 14 15:07 lab7-1.asm
-rw-rw-r-- 1 aamolstihkh aamolstihkh 1456 Nov 14 15:20 lab7-1.o
-rwxrwxr-x 1 aamolstihkh aamolstihkh 9240 Nov 14 15:21 lab7-2
-rw-rw-r-- 1 aamolstihkh aamolstihkh 1873 Nov 14 16:09 lab7-2.asm
-rw-rw-r-- 1 aamolstihkh aamolstihkh 14678 Nov 14 16:10 lab7-2.lst
aamolstihkh@aamolstihkh:~/work/arch-pc/lab07$

```

Рис. 3.16: Вывод содержимого каталога

Внимательно изучаю файл листинга и нахожу отличие: после строки, на которой возникла ошибка - находится сообщение об этом, с указанием конкретной ошибки (рис. 3.17).

```

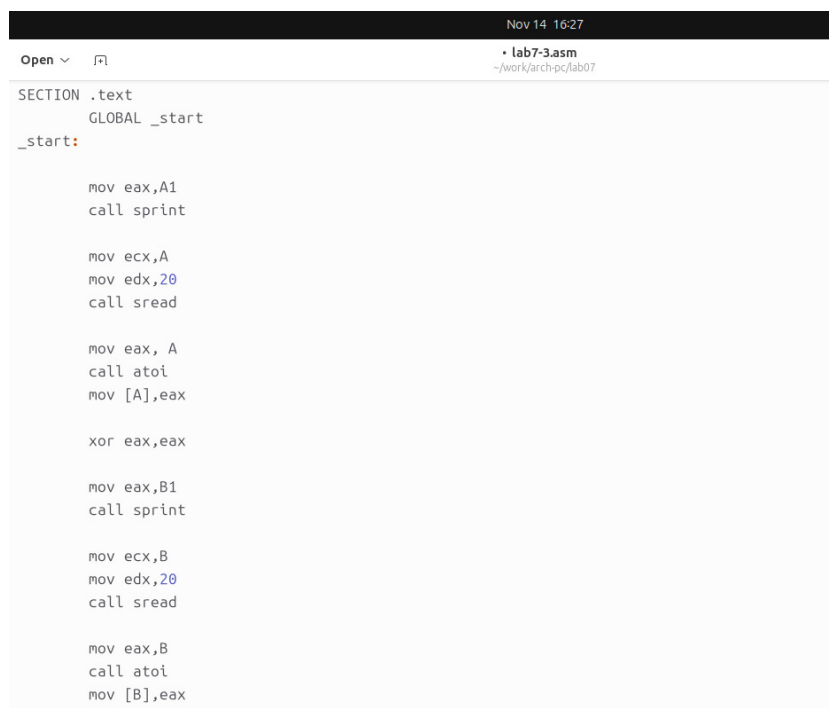
44                                     fin:
45 00000159 b8[13000000]      mov eax, msg2
46 0000015E E8ACFEFFFF      call sprint                ; Вывод сообщения 'Наибольшее число: '
47                                     mov eax                   ; Убран второй аргумент [max]
47 ***** error: invalid combination of opcode and operands
48 00000163 E81EFFFFF8      call iprintLF             ; Вывод 'max(A,B,C)'
49 00000168 E86EFFFFF8      call quit                 ; Выход

```

Рис. 3.17: Изменения в файле листинга

3.3 Задание для самостоятельной работы

Перехожу к выполнению заданий для самостоятельной работы. Создаю программу, которая определяет какое из трех введенных чисел - наименьшее (рис. 3.18).



```
Nov 14 16:27
• lab7-3.asm
~/work/arch-pc/lab07

Open ▾
SECTION .text
GLOBAL _start
_start:

    mov eax,A1
    call sprint

    mov ecx,A
    mov edx,20
    call sread

    mov eax, A
    call atoi
    mov [A],eax

    xor eax,eax

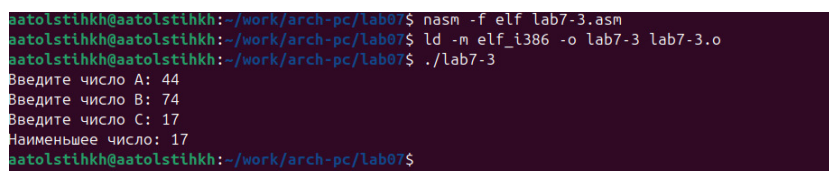
    mov eax,B1
    call sprint

    mov ecx,B
    mov edx,20
    call sread

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

Рис. 3.18: Написание программы

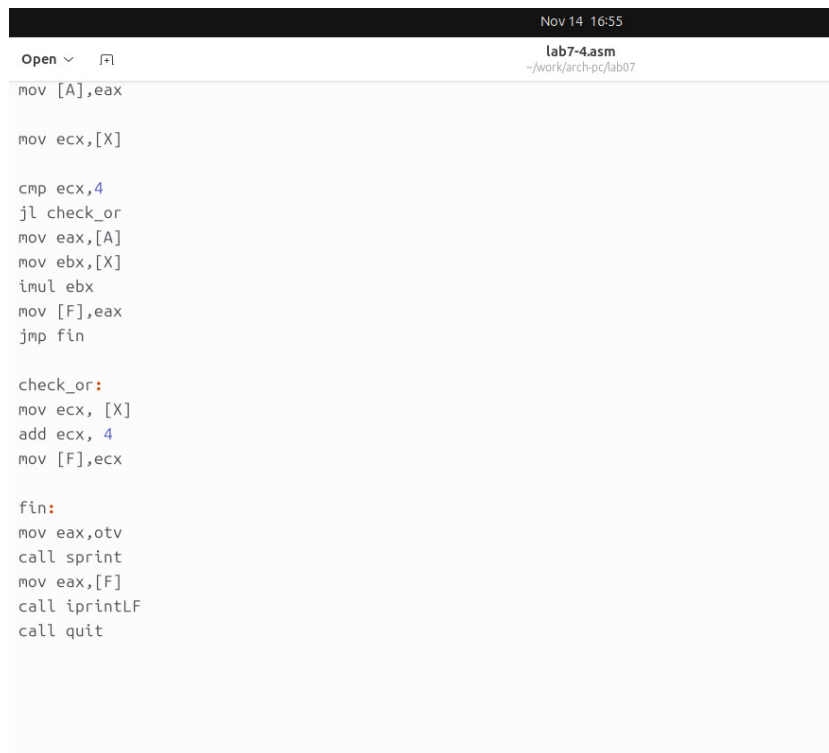
Создаю исполняемый файл и проверяю его работу на трех значениях, соответствующих моему варианту (16) (рис. 3.19).



```
aatolstihkh@aatolstihkh:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
aatolstihkh@aatolstihkh:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
aatolstihkh@aatolstihkh:~/work/arch-pc/lab07$ ./lab7-3
Введите число A: 44
Введите число B: 74
Введите число C: 17
Наименьшее число: 17
aatolstihkh@aatolstihkh:~/work/arch-pc/lab07$
```

Рис. 3.19: Создание исполняемого файла и его запуск

Убеждаюсь, что все работает корректно, а затем создаю новую программу. Она вычисляет значение заданной функции (16 вариант) для введенных значений x , a (рис. 3.20).



The screenshot shows a text editor window titled "lab7-4.asm" with the path "~/work/arch-pc/lab07". The code is written in assembly language and includes comments in Russian. It defines variables A, X, and F, and implements a loop that checks if X is less than 4. If so, it jumps to a 'check_or' label. Otherwise, it moves the value of A to EAX, multiplies it by X, and stores the result in F. The 'check_or' label increments X by 4 and loops back. The 'fin' label prints the value of F and exits the program.

```
Nov 14 16:55
lab7-4.asm
~/work/arch-pc/lab07

Open ▾ [F]

mov [A],eax

mov ecx,[X]

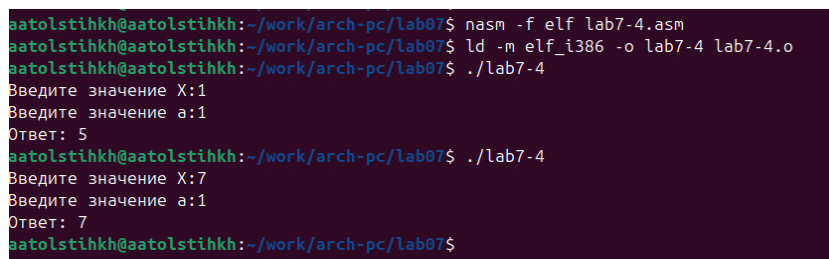
cmp ecx,4
jl check_or
mov eax,[A]
mov ebx,[X]
imul ebx
mov [F],eax
jmp fin

check_or:
mov ecx, [X]
add ecx, 4
mov [F],ecx

fin:
mov eax,otv
call sprint
mov eax,[F]
call iprintLF
call quit
```

Рис. 3.20: Написание программы

Создаю исполняемый файл и проверяю его работу на значениях, соответствующих моему варианту (16) (рис. 3.21).



The screenshot shows a terminal window with the following commands and output:

```
aatolstihkh@aatolstihkh:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
aatolstihkh@aatolstihkh:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
aatolstihkh@aatolstihkh:~/work/arch-pc/lab07$ ./lab7-4
Введите значение X:1
Введите значение a:1
Ответ: 5
aatolstihkh@aatolstihkh:~/work/arch-pc/lab07$ ./lab7-4
Введите значение X:7
Введите значение a:1
Ответ: 7
aatolstihkh@aatolstihkh:~/work/arch-pc/lab07$
```

Рис. 3.21: Создание исполняемого файла и его запуск

4 Выводы

В ходе выполнения лабораторной работы я изучила команды условного и безусловного переходов, а также приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файла листинга.