

# **Отчет по лабораторной работе №7**

**Дисциплина: Архитектура компьютера**

Гашимова Эсма Эльшан кызы

# Содержание

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Цель работы</b>                           | <b>5</b>  |
| <b>2</b> | <b>Задание</b>                               | <b>6</b>  |
| <b>3</b> | <b>Теоретическое введение</b>                | <b>7</b>  |
| <b>4</b> | <b>Выполнение лабораторной работы</b>        | <b>8</b>  |
| 4.1      | Реализация переходов в NASM . . . . .        | 8         |
| 4.2      | Изучение структуры файла листинга . . . . .  | 12        |
| 4.3      | Задания для самостоятельной работы . . . . . | 14        |
| <b>5</b> | <b>Выводы</b>                                | <b>20</b> |
|          | <b>Список литературы</b>                     | <b>21</b> |

# Список иллюстраций

|      |   |    |
|------|---|----|
| 4.1  | Создание каталога и файла для программы . . . . . | 8  |
| 4.2  | Сохранение программы . . . . .                    | 8  |
| 4.3  | Запуск программы . . . . .                        | 9  |
| 4.4  | Изменение программы . . . . .                     | 9  |
| 4.5  | Запуск измененной программы . . . . .             | 9  |
| 4.6  | Изменение программы . . . . .                     | 10 |
| 4.7  | Проверка изменений . . . . .                      | 10 |
| 4.8  | Сохранение новой программы . . . . .              | 11 |
| 4.9  | Проверка программы из листинга . . . . .          | 11 |
| 4.10 | Проверка файла листинга . . . . .                 | 12 |
| 4.11 | Удаление операнда из программы . . . . .          | 13 |
| 4.12 | Просмотр ошибки в файле листинга . . . . .        | 13 |
| 4.13 | Первая программа самостоятельной работы . . . . . | 14 |
| 4.14 | Проверка работы первой программы . . . . .        | 16 |
| 4.15 | Вторая программа самостоятельной работы . . . . . | 16 |
| 4.16 | 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. Самостоятельное написание программ по материалам лабораторной работы

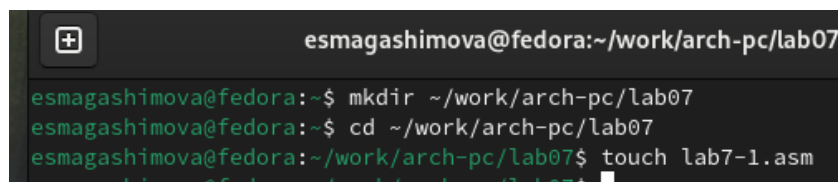
### **3 Теоретическое введение**

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

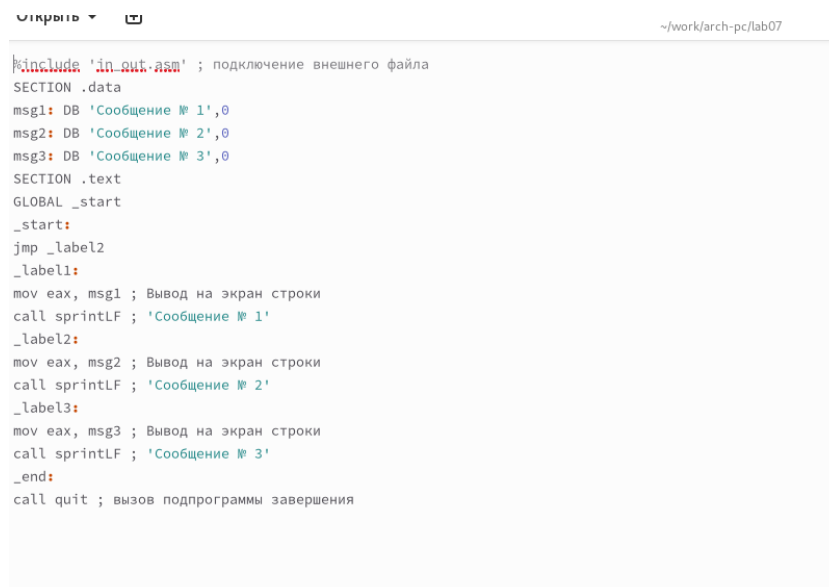
Создаю каталог для программ лабораторной работы №7 (рис. 4.1).



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

Рис. 4.1: Создание каталога и файла для программы

Копирую код из листинга в файл будущей программы. (рис. 4.2).



```
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. 4.2: Сохранение программы

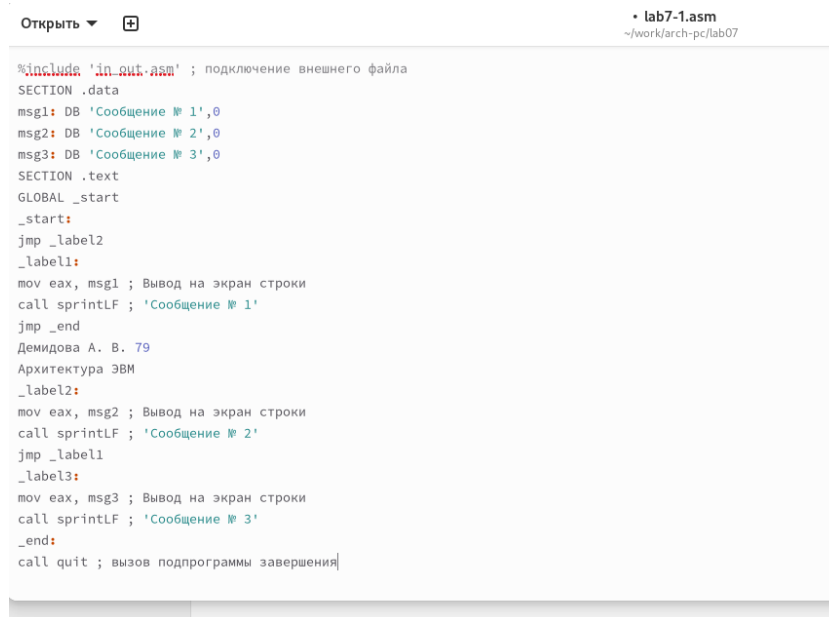


При запуске программы я убедилась в том, что безусловный переход действительно изменяет порядок выполнения инструкций (рис. 4.3).

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

Рис. 4.3: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения функций (рис. 4.4).



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

%include 'in_quit.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
Демидова А. В. 79
Архитектура ЭВМ
_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
```

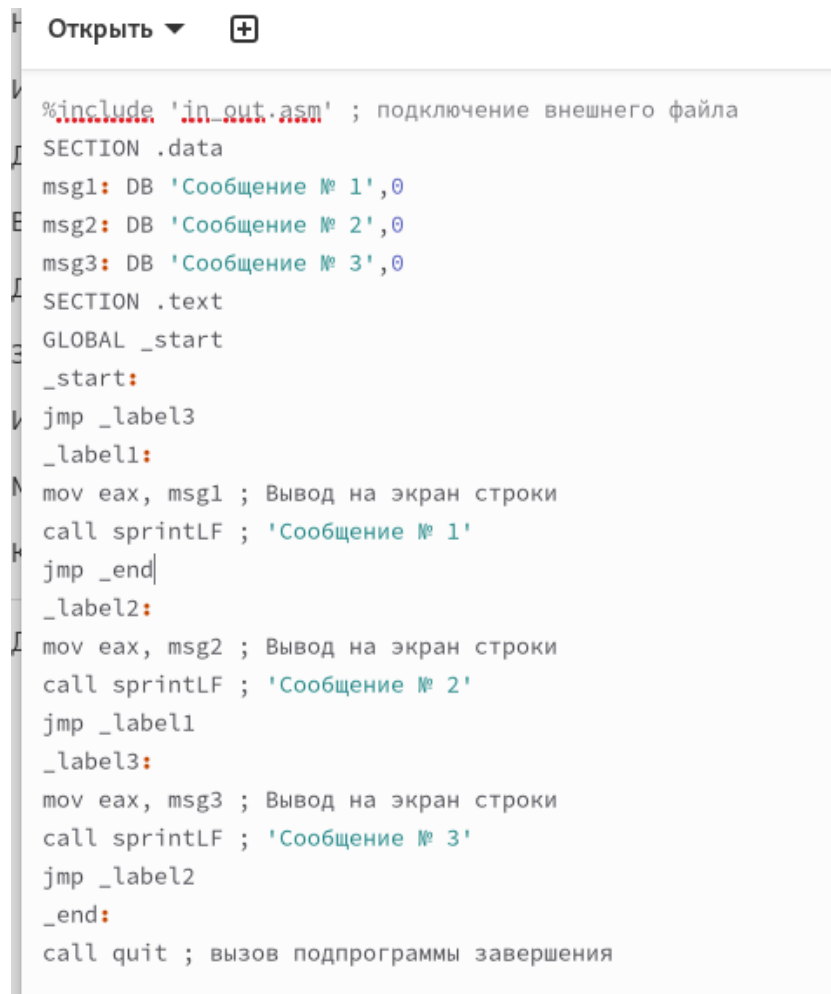
Рис. 4.4: Изменение программы

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

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

Рис. 4.5: Запуск измененной программы

Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. 4.6).




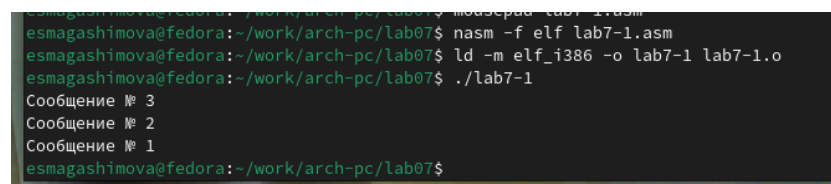
```
Открыть ▼   
%include 'in_out.asm' ; подключение внешнего файла  
SECTION .data  
msg1: DB 'Сообщение № 1',0  
msg2: DB 'Сообщение № 2',0  
msg3: DB 'Сообщение № 3',0  
SECTION .text  
GLOBAL _start  
_start:  
jmp _label3  
_label1:  
mov eax, msg1 ; Вывод на экран строки  
call sprintf ; 'Сообщение № 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. 4.6: Изменение программы

Работа выполнена корректно, программа в нужном порядке выводит сообщения (рис. 4.7).



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

Рис. 4.7: Проверка изменений

Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. 4.8).

```

*~/work/arch-pc/lab07/lab7-2.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
%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 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'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'

```

Рис. 4.8: Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяя работу программы с разными входными данными (рис. 4.9).

```

esmagashimova@fedora:~/work/arch-pc/lab07$ cat in_out.asm
esmagashimova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 25
Наибольшее число: 50
esmagashimova@fedora:~/work/arch-pc/lab07$ 60
bash: 60: команда не найдена...
esmagashimova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: ^[[C^[[2~
Наибольшее число: 50
esmagashimova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
esmagashimova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
esmagashimova@fedora:~/work/arch-pc/lab07$

```

Рис. 4.9: 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 mousepad (рис. 4.10).

```
esmagashimova@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
esmagashimova@fedora:~/work/arch-pc/lab07$ mousepad lab7-2.lst
```

| Файл                   | Правка | Поиск | Просмотр | Документ                                 | Помощь |
|------------------------|--------|-------|----------|--|--------|
| 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> |        |
| 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                              |        |
| 31                     |        |       |          | <1>                                      |        |
| 32 0000001B 89C1       |        |       |          | <1> mov ecx, eax                         |        |
| 33 0000001D BB01000000 |        |       |          | <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                              |        |
| -- -----               |        |       |          | -  |        |

Рис. 4.10: Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями.

Удаляю один операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. 4.11).

```

_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'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,|
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход

```

Рис. 4.11: Удаление операнда из программы

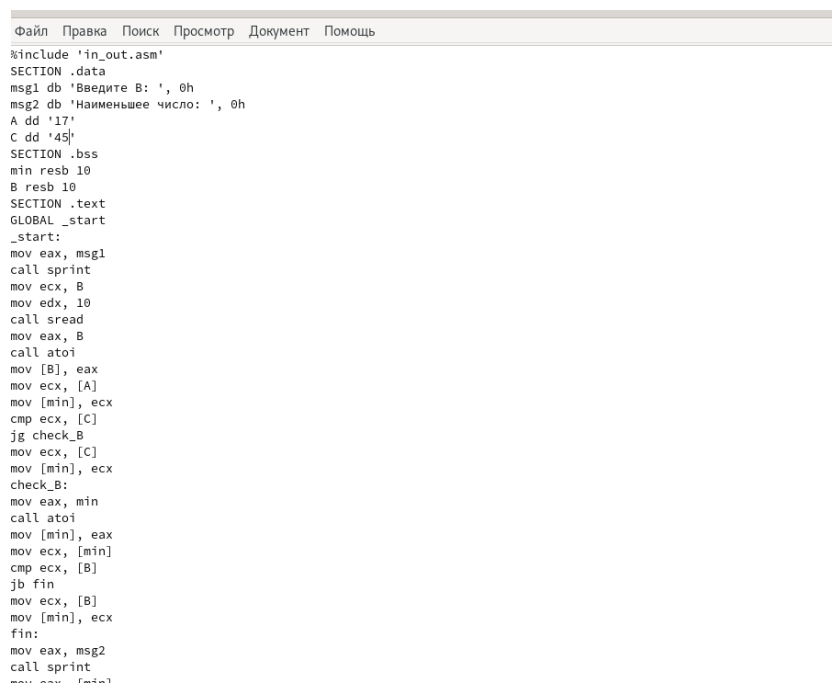
В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. 4.12).

|  |  |
|--|--|
| <pre> 32 33 34 35 00000130 E867FFFFFF 36 00000135 A3[00000000] 37 38 0000013A 8B0D[00000000] 39 00000140 3B0D[0A000000] 40 00000146 7F0C 41 00000148 8B0D[0A000000] 42 0000014E 890D[00000000] 43 44 45 00000154 B8[13000000] 46 00000159 E8B1FEFFFF 47 0000015E A1[00000000] 48 00000163 E81EFFFFFF 49 00000168 E86EFFFFFF </pre> | <pre> ; ----- Преобразование 'max(A,C)' из символа в число check_B: mov eax error: invalid combination of opcode and operands call atoi ; Вызов подпрограммы перевода символа в число mov [max],eax ; запись преобразованного числа в 'max' ; ----- Сравниваем 'max(A,C)' и 'B' (как числа) mov ecx,[max] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B' jg fin ; если 'max(A,C)&gt;B', то переход на 'fin', mov ecx,[B] ; иначе 'ecx = B' mov [max],ecx ; ----- Вывод результата fin: mov eax, msg2 call sprint ; Вывод сообщения 'Наибольшее число: ' mov eax,[max] call iprintLF ; Вывод 'max(A,B,C)' call quit ; Выход </pre> |
|--|--|

Рис. 4.12: Просмотр ошибки в файле листинга

## 4.3 Задания для самостоятельной работы

Использую свой первый вариант из предыдущей лабораторной работы. Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 4.13).



```
Файл  Правка  Поиск  Просмотр  Документ  Помощь

%include 'in_out.asm'
SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '17'
C dd '45'
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]
jg check_B
mov ecx, [C]
mov [min], ecx
check_B:
mov eax, min
call atoi
mov [min], eax
mov ecx, [min]
cmp ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx
fin:
mov eax, msg2
call sprint
mov ecx, [min]
```

Рис. 4.13: Первая программа самостоятельной работы

Код первой программы:

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '17'
C dd '45'

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]
jg check_B
mov ecx, [C]
mov [min], ecx
check_B:
mov eax, min
call atoi
mov [min], eax
mov ecx, [min]
cmp ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx
fin:
mov eax, msg2
call sprint

```

```

mov eax, [min]
call iprintLF
call quit

```

Проверяю корректность написания первой программы (рис. 4.14).

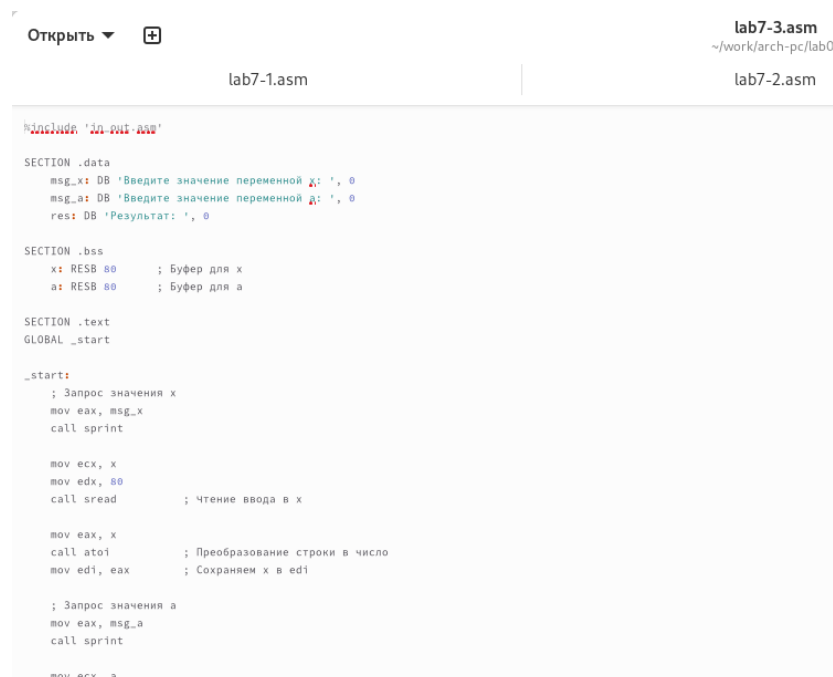
```

esmagashimova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
esmagashimova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2.o
ld: отсутствуют входные файлы
esmagashimova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
esmagashimova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 23
Наименьшее число: 17
esmagashimova@fedora:~/work/arch-pc/lab07$

```

Рис. 4.14: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатуры переменных а и х (рис. 4.15).



```

%include 'in_out.asm'

SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 0

SECTION .bss
x: RESB 80 ; Буфер для x
a: RESB 80 ; Буфер для a

SECTION .text
GLOBAL _start

_start:
; Запрос значения x
mov eax, msg_x
call sprint

mov ecx, x
mov edx, 80
call sread ; Чтение ввода в x

mov eax, x
call atoi ; Преобразование строки в число
mov edi, eax ; Сохраняем x в edi

; Запрос значения a
mov eax, msg_a
call sprint

mov ecx, a

```

Рис. 4.15: Вторая программа самостоятельной работы

Код второй программы:

```

%include 'in_out.asm'

```



## SECTION .data

```
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 0
```

## SECTION .bss

```
x: RESB 80      ; Буфер для x
a: RESB 80      ; Буфер для a
```

## SECTION .text

### GLOBAL \_start

\_start:

*; Запрос значения x*

```
mov eax, msg_x
```

```
call sprint
```

```
mov ecx, x
```

```
mov edx, 80
```

```
call sread      ; Чтение ввода в x
```

```
mov eax, x
```

```
call atoi      ; Преобразование строки в число
```

```
mov edi, eax    ; Сохраняем x в edi
```

*; Запрос значения a*

```
mov eax, msg_a
```

```
call sprint
```

```

mov ecx, a
mov edx, 80
call sread          ; Чтение ввода в a

mov eax, a
call atoi           ; Преобразование строки в число
mov esi, eax        ; Сохраняем a в esi

; Сравнение x и a
cmp edi, esi        ; Сравниваем x и a
jl calculate_1      ; Если x < a, то переходим к вычислению 2a - x

; Если x >= a, результат = 8
mov eax, 8
jmp print_result

calculate_1:
; Если x < a, результат = 2a - x
mov eax, esi        ; eax = a
shl eax, 1          ; eax = 2a
sub eax, edi        ; eax = 2a - x

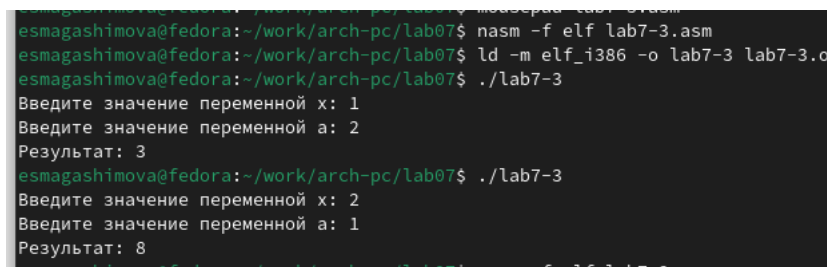
print_result:
; Выводим результат
mov edi, eax
mov eax, res
call sprint
mov eax, edi
call iprintLF

```

*; Завершаем программу*

**call** quit

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений а и х (рис. 4.16).



```
esmagashimova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
esmagashimova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
esmagashimova@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 1
Введите значение переменной a: 2
Результат: 3
esmagashimova@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 2
Введите значение переменной a: 1
Результат: 8
```

Рис. 4.16: Проверка работы второй программы

## **5 Выводы**

При выполнении лабораторной работы я изучила команды условных и безусловных переходов, а также приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файлов листинга.

# Список литературы

1. Курс на ТУИС
2. Лабораторная работа №7
3. Программирование на языке ассемблера NASM Столяров А. В.