

Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютера

Терещенкова Маргарита Владимировна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
4.1	Реализация переходов в NASM	7
4.2	Изучение структуры файлы листинга	12
5	Задания для самостоятельной работы	16
6	Выводы	20
	Список литературы	21

Список иллюстраций

4.1	Создание каталога	7
4.2	Создание файла	7
4.3	Копирование файла	7
4.4	Редактирование файла	8
4.5	Запуск исполняемого файла	8
4.6	Редактирование файла	9
4.7	Запуск исполняемого файла	9
4.8	Редактирование файла	10
4.9	Запуск исполняемого файла	10
4.10	Создание файла	11
4.11	Редактирование файла	11
4.12	Запуск исполняемого файла	11
4.13	Создание файла	12
4.14	Открытие файла в mouserad	13
4.15	Удаление операнда из программы и запуск программы	15
4.16	Просмотр ошибки в файле листинга	15
5.1	Создание файла	16
5.2	Редактирование файла	17
5.3	Запуск исполняемого файла	17
5.4	Создание файла	18
5.5	Написание программы	18
5.6	Запуск исполняемого файла	19

1 Цель работы

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

2 Задание

1. Программа с использованием инструкции `jmp` (листинг 1)
2. Программа с использованием инструкции `jmp` (листинг 2)
3. Изучение структуры файла листинга
4. Выполнение заданий 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

4.1 Реализация переходов в NASM

1. Создаю каталог для программ лабораторной работы № 7 с помощью команды **mkdir**.

```
mvtereshenkova@margo-pc:~$ mkdir ~/work/study/2024-2025/Архитектура\ компьютера/arch-pc/lab07
```

Рис. 4.1: Создание каталога

Перехожу в него и создаю файл lab7-1.asm с помощью утилиты **touch**.

```
mvtereshenkova@margo-pc:~$ cd ~/work/study/2024-2025/Архитектура\ компьютера/arch-pc/lab07
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ touch lab7-1.asm
```

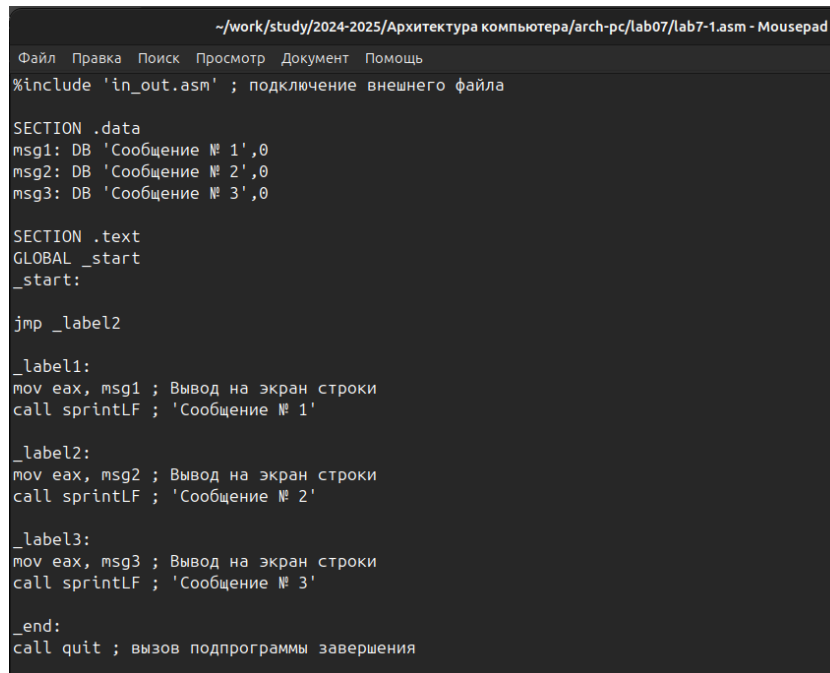
Рис. 4.2: Создание файла

Копирую в текущий каталог файл in_out.asm с помощью утилиты **cp**, так как он будет использоваться в других программах. И проверяю наличие файла в данной директории с помощью команды **ls**.

```
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ cp ~/Загрузки/in_out.asm in_out.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ls
in_out.asm  lab7-1.asm
```

Рис. 4.3: Копирование файла

2. Открываю созданный файл lab7-1.asm, вставляю в него текст программы из листинга 7.1 (**Программа с использованием инструкции jmp**).



```
~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07/lab7-1.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
%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'

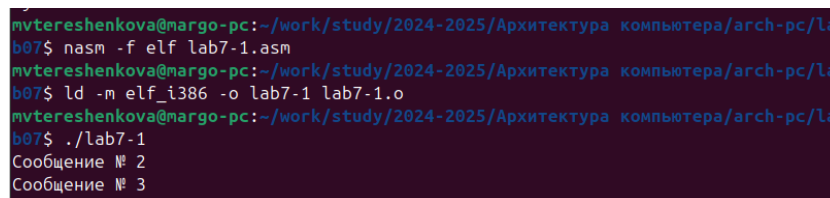
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.4: Редактирование файла

Создаю исполняемый файл программы и запускаю его.



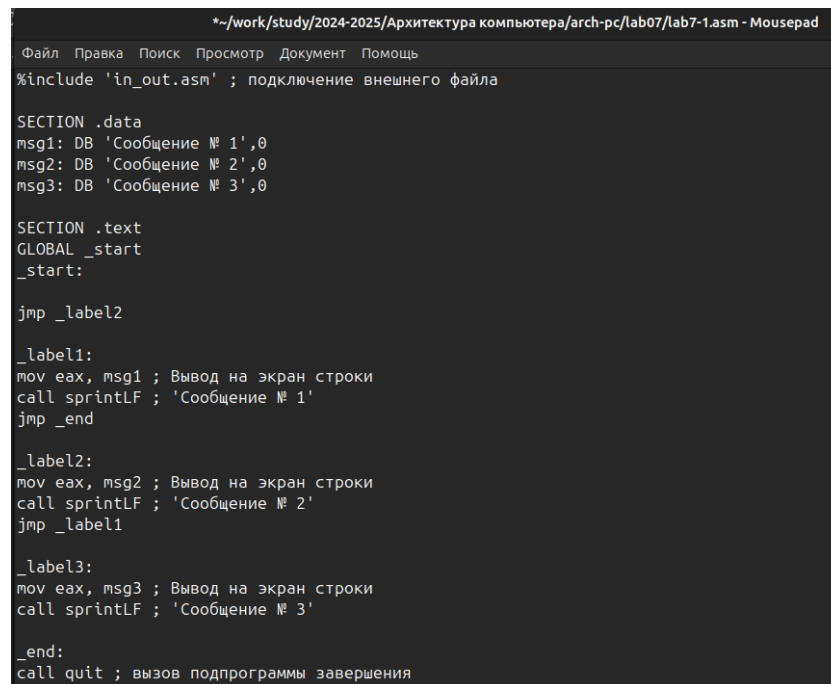
```
mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 4.5: Запуск исполняемого файла

Таким образом, использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения.

Изменила программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`). Изменяю текст программы в соответствии с листингом 7.2(Программа с

использованием инструкции `jmp`).



```
*~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07/lab7-1.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
#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

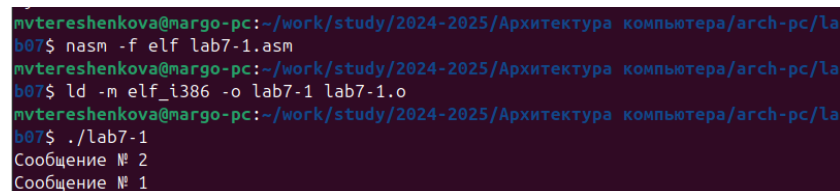
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.6: Редактирование файла

Создаю исполняемый файл программы и запускаю его.



```
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис. 4.7: Запуск исполняемого файла

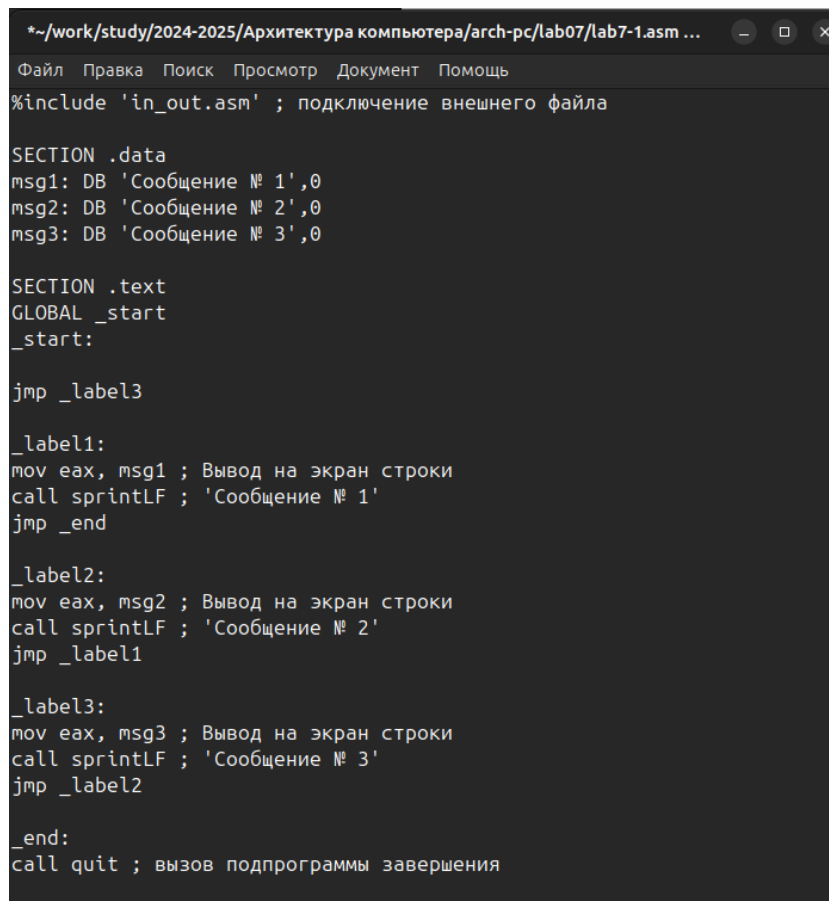
Изменила текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим:

```
mvtereshenkova@margo-pc:~$ ./lab7-1
```

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
*~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07/lab7-1.asm ...
Файл  Правка  Поиск  Просмотр  Документ  Помощь
%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 sprintfLF ; 'Сообщение № 1'
jmp _end

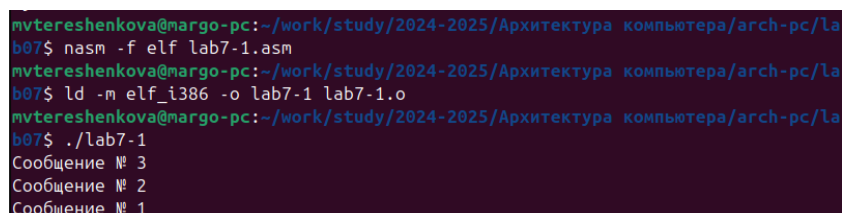
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 3'
jmp _label2

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.8: Редактирование файла

Создаю исполняемый файл программы и запускаю его.



```
mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 4.9: Запуск исполняемого файла

3. Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучаю текст программы из листинга 7.3 (**Программа, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А,В и С**) и ввожу в lab7-2.asm.

```

mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07$ touch lab7-2.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2.asm

```

Рис. 4.10: Создание файла

```

*~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07/lab7-2.asm ...
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'

```

Рис. 4.11: Редактирование файла

```

mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07$ nasm -f elf lab7-2.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07$ ld -m elf_i386 -o lab7-2 lab7-2.o
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07$ ./lab7-2
Введите B: 20
Наибольшее число: 50
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07$ ./lab7-2
Введите B: 51
Наибольшее число: 51

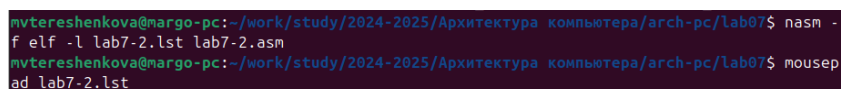
```

Рис. 4.12: Запуск исполняемого файла

При введении числа до 50, программа выводит наибольшее число 50, при введении числа больше 50, программа выводит введенное нами число. Программа сравнивает число А (значение 20) и С (значение 50) и инициализирует переменную max значением большего из них. Сравнивает текущее значение max с введенным числом В и обновляет max, если В больше. Выводит сообщение “Наибольшее число:” и затем значение переменной max, которая содержит наибольшее из трёх чисел: А, В и С.

4.2 Изучение структуры файлы листинга

4. Создаю файл листинга для программы из файла lab7-2.asm. Открываю его через mousepad.



```
mvtreshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
mvtreshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ mousepad lab7-2.lst
```

Рис. 4.13: Создание файла

```
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      <1> nextchar:
9      00000003 803800    <1>      cmp     byte [eax], 0
10     00000006 7403     <1>      jz      finished
11     00000008 40        <1>      inc     eax
12     00000009 EBF8     <1>      jmp     nextchar
13     <1>
14     <1> finished:
15     0000000B 29D8     <1>      sub     eax, ebx
16     0000000D 5B        <1>      pop     ebx
17     0000000E C3        <1>      ret
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 50        <1>      pop     ebx
```

Рис. 4.14: Открытие файла в mousepad

При компиляции и сборке программы на ассемблере создаются следующие файлы:

- Объектный файл (.o): Это промежуточный файл, содержащий машинный код, но ещё не готовый для выполнения.
- Исполняемый файл: После связывания объектных файлов с библиотеками (например, с помощью ld), создается исполняемый файл, который можно запустить.
- Файл листинга (.lst): Это текстовый файл, который включает исходный код программы вместе с адресами и скомпилированным машинным кодом. В этом файле обычно содержатся комментарии и информация о процессе компиляции.

В файл листинга могут быть добавлены следующие элементы:

- Исходный код: Полный исходный код программы, как он написан в ассемблере.
- Адреса: Для каждой инструкции будут указаны адреса в памяти, по которым эти инструкции будут располагаться после компиляции.
- Машинный код: Бинарный код, соответствующий каждой инструкции, представленный в шестнадцатеричном формате.
- Комментарии: Комментарии из исходного кода, которые могут помочь понять логику программы.
- Информация о секциях: Данные о том, как разделены секции кода (.text, .data, .bss и т.д.) и их размеры.
- Ошибки и предупреждения: Если при компиляции были обнаружены ошибки или предупреждения, они также могут быть записаны в файл 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. Открыла файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалить один операнд. Далее выполнила трансляцию с получением файла листинга: **nasm -f elf -l lab7-2.lst lab7-2.asm**

```
mousepad lab7-2.asm

(mousepad:22484): GLib-CRITICAL **: 19:21:46.941: g_strjoinv: assertion 'str_array !=
NULL' failed

(mousepad:22484): GLib-CRITICAL **: 19:21:46.941: g_strjoinv: assertion 'str_array !=
NULL' failed
nvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:27: error: invalid combination of opcode and operands
nvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
mousepad lab7-2.lst
```

Рис. 4.15: Удаление операнда из программы и запуск программы

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

```
~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07/lab7-2.lst - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
17      ; ----- Вывод сообщения 'Введите B: '
18 000000E8 B8[00000000]    mov eax,msg1
19 000000ED E81DFFFFFF    call sprint
20
21      ; ----- Ввод 'B'
22 000000F2 B9[0A000000]    mov ecx,B
23 000000F7 BA0A000000    mov edx,10
24 000000FC E842FFFFFF    call sread
25
26      ; ----- Преобразование 'B' из символа в число
27      mov eax,
27      ***** error: invalid combination of opcode and operands
28 00000101 E896FFFFFF    call atoi ; Вызов подпрограммы перевода символа в число
29 00000106 A3[0A000000]    mov [B],eax ; запись преобразованного числа в 'B'
30
31      ; ----- Записываем 'A' в переменную 'max'
32 00000108 8B0D[35000000]    mov ecx,[A] ; 'ecx = A'
33 00000111 890D[00000000]    mov [max],ecx ; 'max = A'
34
35      ; ----- Сравниваем 'A' и 'C' (как символы)
36 00000117 3B0D[39000000]    cmp ecx,[C] ; Сравниваем 'A' и 'C'
37 0000011D 7F0C    jg check_B ; если 'A>C', то переход на метку 'check_B',
38 0000011F 8B0D[39000000]    mov ecx,[C] ; иначе 'ecx = C'
39 00000125 890D[00000000]    mov [max],ecx ; 'max = C'
40
41      ; ----- Преобразование 'max(A,C)' из символа в число
42      check_B:
43 0000012B B8[00000000]    mov eax,max
44 00000130 E867FFFFFF    call atoi ; Вызов подпрограммы перевода символа в число
45 00000135 A3[00000000]    mov [max],eax ; запись преобразованного числа в 'max'
46
```

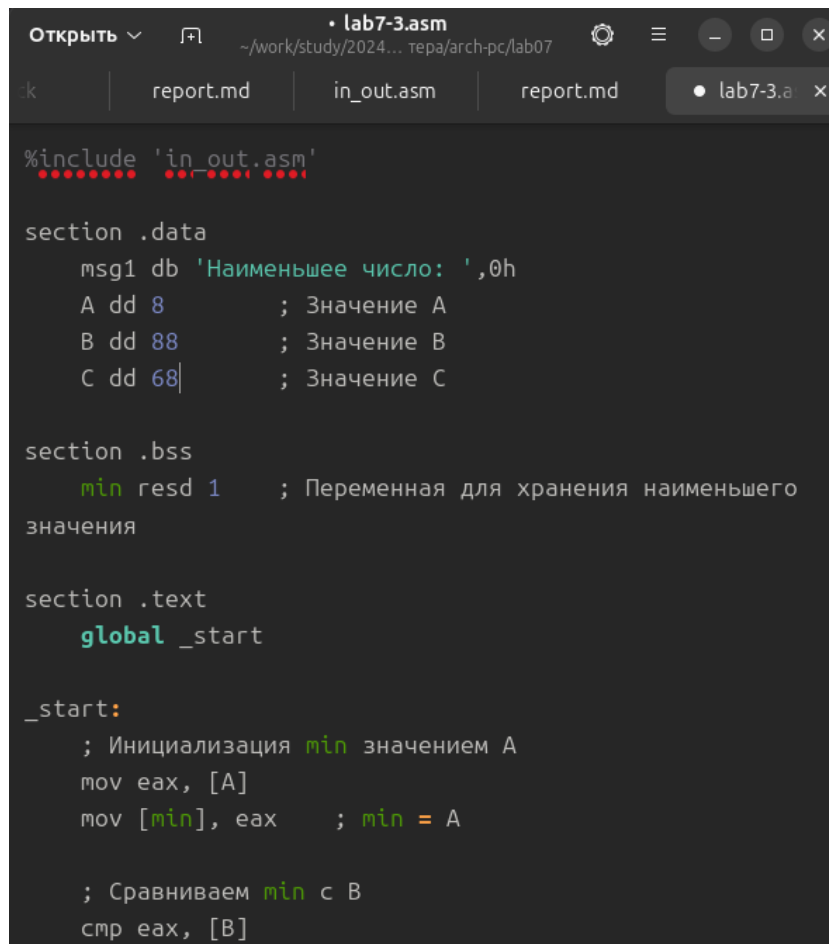
Рис. 4.16: Просмотр ошибки в файле листинга

5 Задания для самостоятельной работы

1. Мой вариант номер 4. Создаю файл с названием lab7-3.asm, написала программу для нахождения наименьшего из 3 переменных, значения переменных беру, исходя из своего варианта, полученного в ходе лабораторной работы номер 6. Сама программа прикреплена в ТУИС и в Git Hub.

```
nvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07$ touch lab7-3.asm
nvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07$ ls
in_out.asm  lab7-1.asm  lab7-2      lab7-2.lst  lab7-3.asm
lab7-1      lab7-1.o    lab7-2.asm  lab7-2.o
```

Рис. 5.1: Создание файла



```
Открыть ▾  • lab7-3.asm  ~/work/study/2024... тепа/arch-pc/lab07  report.md  in_out.asm  report.md  • lab7-3.a x

%include 'in_out.asm'

section .data
    msg1 db 'Наименьшее число: ',0h
    A dd 8      ; Значение A
    B dd 88     ; Значение B
    C dd 68     ; Значение C

section .bss
    min resd 1  ; Переменная для хранения наименьшего значения

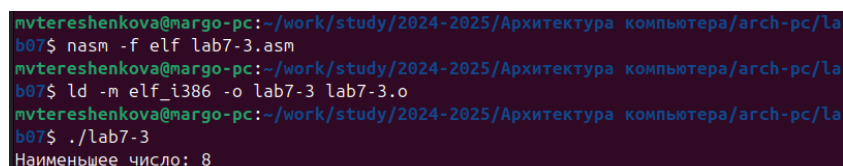
section .text
    global _start

_start:
    ; Инициализация min значением A
    mov eax, [A]
    mov [min], eax    ; min = A

    ; Сравниваем min с B
    cmp eax, [B]
```

Рис. 5.2: Редактирование файла

Проверяю работу программы, программа работает корректно.



```
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07$ nasm -f elf lab7-3.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07$ ld -m elf_i386 -o lab7-3 lab7-3.o
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07$ ./lab7-3
Наименьшее число: 8
```

Рис. 5.3: Запуск исполняемого файла

2. Создаю файл с названием lab7-4.asm.

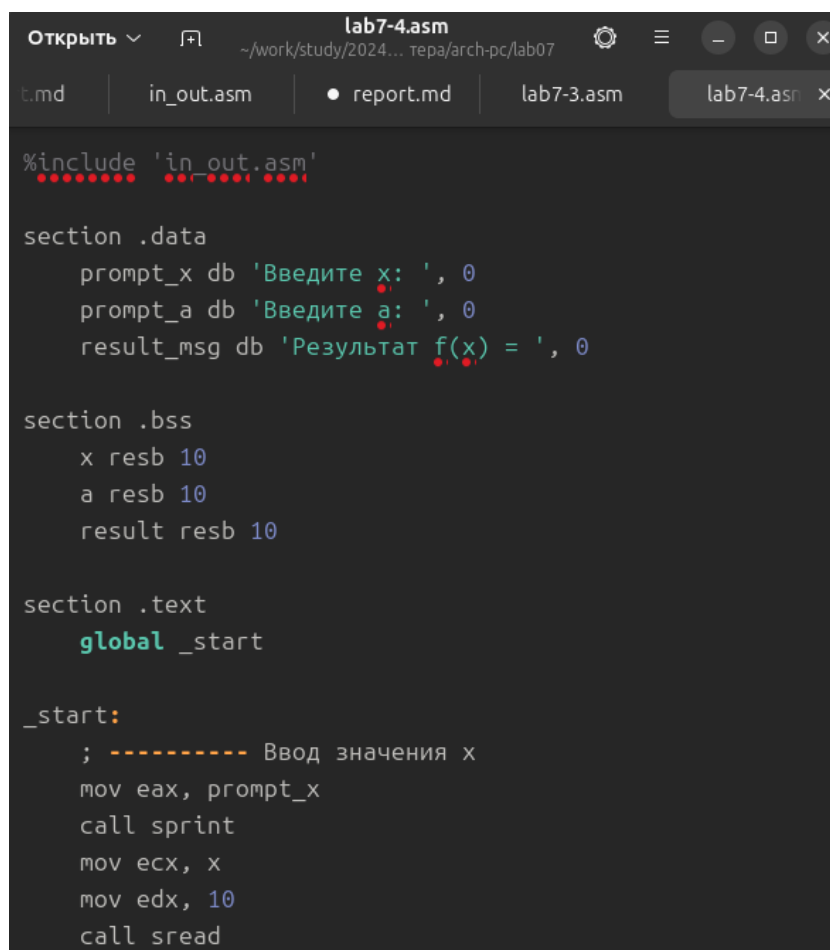
```

mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ touch lab7-4.asm
mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ls
in_out.asm  lab7-1.asm  lab7-2      lab7-2.lst  lab7-3      lab7-3.o
lab7-1      lab7-1.o    lab7-2.asm  lab7-2.o    lab7-3.asm  lab7-4.asm

```

Рис. 5.4: Создание файла

Написала программу для вычисления $f(x)$. Сама программа прикреплена в ТУИС и в Git Hub.



```

lab7-4.asm
~/work/study/2024... тепа/arch-pc/lab07
t.md | in_out.asm | ● report.md | lab7-3.asm | lab7-4.asm x

%include 'in_out.asm'

section .data
    prompt_x db 'Введите x: ', 0
    prompt_a db 'Введите a: ', 0
    result_msg db 'Результат f(x) = ', 0

section .bss
    x resb 10
    a resb 10
    result resb 10

section .text
    global _start

_start:
    ; ----- Ввод значения x
    mov eax, prompt_x
    call sprint
    mov ecx, x
    mov edx, 10
    call sread

```

Рис. 5.5: Написание программы

```
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07$ nasm -f elf lab7-4.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07$ ld -m elf_i386 -o lab7-4 lab7-4.o
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07$ ./lab7-4
Введите x: 3
Введите a: 0
Результат f(x) = 7
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07$ ./lab7-4
Введите x: 3
Введите a: 2
Результат f(x) = 8
```

Рис. 5.6: Запуск исполняемого файла

Программа работает корректно; проверила, подставив соответствующие x и a , указанные для моего варианта.

6 Выводы

Благодаря данной лабораторной работе изучила команды условного и безусловного переходов; приобрела навыки написания программ с использованием переходов; познакомилась с назначением и структурой файла листинга.

Список литературы

1. Архитектура компьютера