

Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Ермакова Анастасия Алексеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация переходов в NASM	8
4.2	Изучение структуры файла листинга	13
4.3	Выполнение заданий для самостоятельной работы	16
5	Выводы	23
6	Список литературы	24

Список иллюстраций

4.1	Создание директории и файла	8
4.2	Текст программы файла lab7-1.asm	9
4.3	Результат работы программы	9
4.4	Измененный текст файла	10
4.5	Результат работы программы	10
4.6	Измененный текст программы	11
4.7	Результат работы программы	11
4.8	Создание файла lab7-2.asm	11
4.9	Текст программы	12
4.10	Результат работы программы	13
4.11	Создание файла листинга	13
4.12	Текст файла листинга	14
4.13	Удаление одного операнда	15
4.14	Выполнение трансляции	15
4.15	Ошибка в файле листинга	16
4.16	Текст программы файла lab7-3.asm	17
4.17	Результат работы программы	18
4.18	Текст программы файла lab7-4.asm	20
4.19	Результат работы программы	21

Список таблиц

1 Цель работы

Цель данной лабораторной работы - изучение команды 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 файла листинга
3. 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

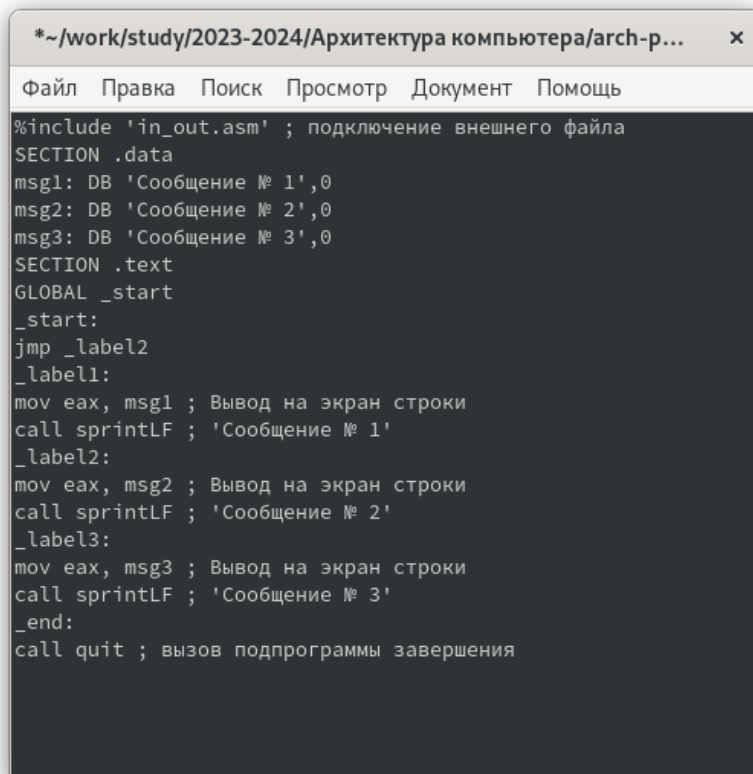
4.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы №7, перехожу в него, создаю там файл lab7-1.asm и открываю его для редактирования с помощью mousepad. (рис. 4.1).

```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc$ mkdir lab07
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc$ cd lab07
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ touch lab7-1.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ mousepad lab7-1.asm
```

Рис. 4.1: Создание директории и файла

Ввожу в файл текст листинга 7.1. (рис. 4.2).

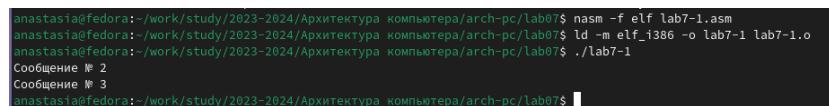


```
*~/work/study/2023-2024/Архитектура компьютера/arch-p... x
Файл  Правка  Поиск  Просмотр  Документ  Помощь

%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 sprintfLF ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.2: Текст программы файла lab7-1.asm

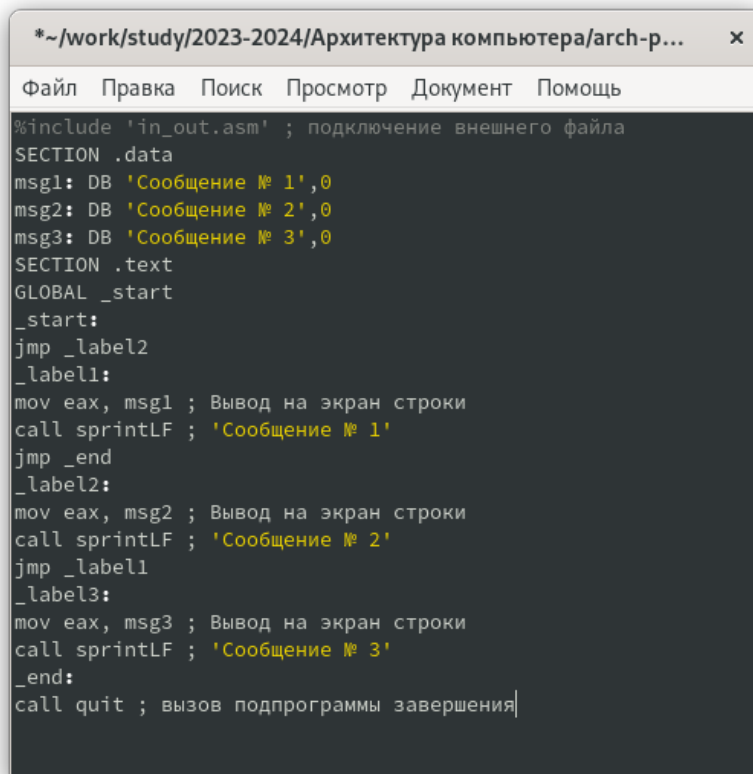
Создаю исполняемый файл и запускаю его. Результат работы данной программы совпадает с результатом в файле на ТУИС. (рис. 4.3).



```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.3: Результат работы программы

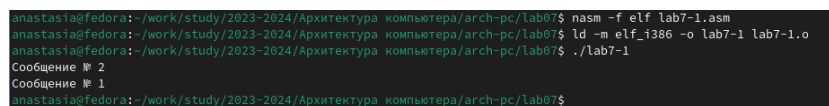
Изменяю текст программы в соответствии с листингом 7.2. (рис. 4.4).



```
*~/work/study/2023-2024/Архитектура компьютера/arch-p... x
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. 4.4: Измененный текст файла

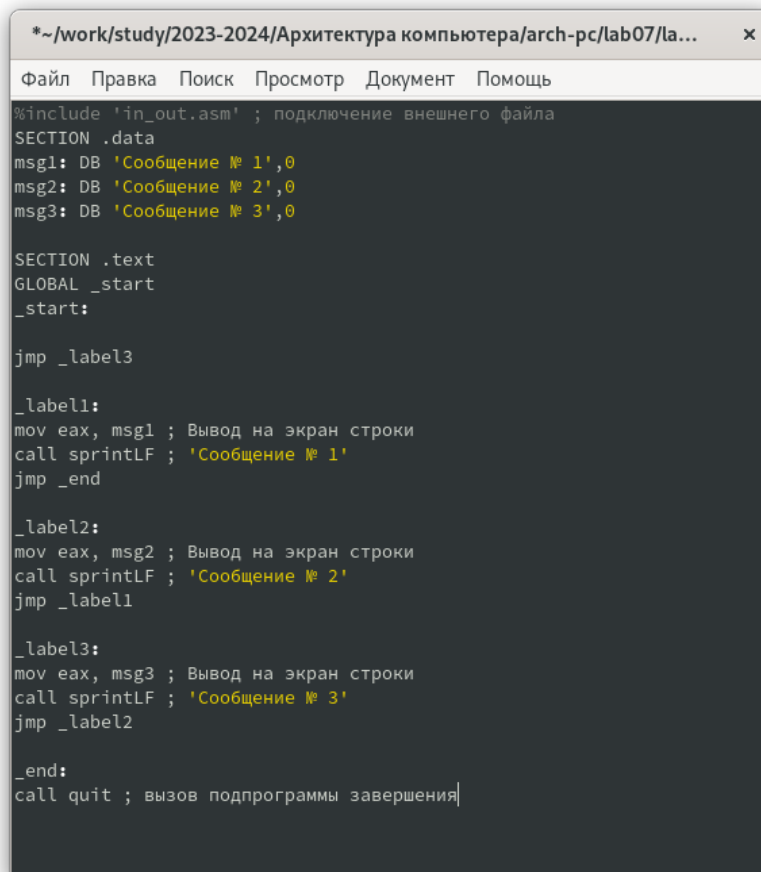
Создаю исполняемый файл и запускаю его. Программа выводит сначала Сообщение № 2, а затем Сообщение № 1, что соответствует заданию. (рис. 4.5).



```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.5: Результат работы программы

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



```
*~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07/la... x
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 sprintf ; 'Сообщение № 1'
jmp _end

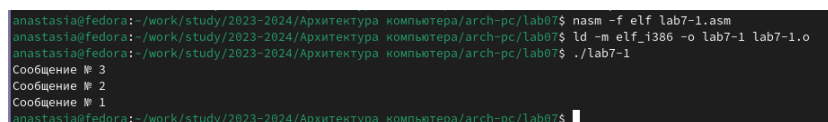
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1

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

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.6: Измененный текст программы

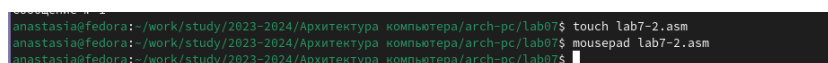
Создаю исполняемый файл и запускаю его. Программа работает верно. (рис. 4.7).



```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.7: Результат работы программы

Создаю файл lab7-2.asm в текущем каталоге и открываю его для редактирования с помощью mousepad. (рис. 4.8).



```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ touch lab7-2.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ mousepad lab7-2.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.8: Создание файла lab7-2.asm

Ввожу в него текст программы из листинга 7.3. (рис. 4.9).



```
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07/lab7-2.asm - Mousepad x
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'
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. 4.9: Текст программы

Создаю исполняемый файл и запускаю его. Проверяю его работу для разных значений B (15, 35, 65). Программа работает исправно. (рис. 4.10).

```

anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-2.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите B: 15
Наибольшее число: 50
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите B: 35
Наибольшее число: 50
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите B: 65
Наибольшее число: 65
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$

```

Рис. 4.10: Результат работы программы

4.2 Изучение структуры файла листинга

Создаю файл листинга для программы из файла lab7-2.asm и открываю его с помощью mcedit. (рис. 4.11).

```

anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ mcedit lab7-2.lst

```

Рис. 4.11: Создание файла листинга

Внимательно ознакамливаюсь с его форматом и содержимым. (рис. 4.12). Объясню следующие три строки листинга (строки 8, 9 и 10):

- `cmp byte [eax], 0` (строка 8): Эта команда сравнивает байт, на который указывает регистр `eax`, с нулем. Если байт равен 0, флаг нуля (ZF) устанавливается в 1.
- `jz finished` (строка 9): Эта команда выполняет переход к метке `finished`, если флаг нуля установлен (то есть если байт, на который указывает `eax`, равен 0). Это условный переход, и если `cmp` не установит флаг, выполнение продолжится.
- `inc eax` (строка 10): Если предыдущая команда не привела к переходу, эта команда увеличивает значение в регистре `eax` на 1.

```
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07
lab7-2.lst  [----]  0 L: 1+ 0 1/225] +(0 /14458b) 0032 0x020 [x] [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> 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> ----- sprint -----
19     <1> ; Функция печати сообщения
20     <1> ; входные данные: mov eax, <message>
21     <1> sprint:
22     0000000F 52      <1> push edx
23     00000010 51      <1> push ecx
24     00000011 53      <1> push ebx
25     00000012 50      <1> push eax
26     00000013 E8E8FFFF <1> call slen
27     <1> -----
28     <1> -----
29     00000018 89C2    <1> mov edx, eax
30     0000001A 5B      <1> pop eax
31     <1> -----
32     0000001B 89C1    <1> mov ecx, eax
33     0000001D B801000000 <1> mov ebx, 1
34     00000022 B804000000 <1> mov eax, 4
35     00000027 CD00    <1> int 80h
```

Рис. 4.12: Текст файла листинга

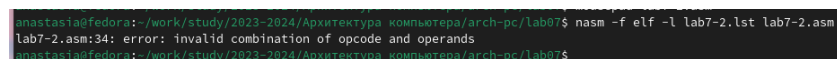
Открываю файл с программой lab7-2.asm и в инструкции с двумя операндами удаляю один из них. (рис. 4.13).



```
*~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07/lab7-2.asm - Mousepad x
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,[
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. 4.13: Удаление одного операнда

Выполняю трансляцию с получением файла листинга. (рис. 4.14). Команда выводит ошибку.



```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:34: error: invalid combination of opcode and operands
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.14: Выполнение трансляции

В новом файле листинга показывает ошибку, возникшую в результате трансляции файла. Никакие выходные данные при этом не создаются. (рис. 4.15).

```

29 00000122 7F0C          jg check_B ; если 'A>C', то переход на метку 'check_B',
30 00000124 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
31 0000012A 890D[00000000] mov [max],ecx ; 'max = C'
32          ; ----- Преобразование 'max(A,C)' из символа в число
33          check_B:
34          mov eax,
34          ***** error: invalid combination of opcode and operands
35 00000130 E867FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
36 00000135 A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
37          ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013A 8B0D[00000000] mov ecx,[max]
39 00000140 3B0D[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'

```

Рис. 4.15: Ошибка в файле листинга

4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab7-3.asm для выполнения первого задания самостоятельной работы и пишу программу нахождения наименьшей из трех целочисленных переменных a, b и c. (рис. 4.16).


```

%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наименьшее число: ",0h
A dd '46'
C dd '74'
section .bss
min 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 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
str 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 [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]
str ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
jbl 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 sprint ; Вывод сообщения 'Наименьшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход

```

Рис. 4.16: Текст программы файла lab7-3.asm

Создаю исполняемый файл и запускаю его. Проверяю его работу, введя переменные из таблицы в соответствии со своим вариантом, полученным в результате выполнения предыдущей лабораторной работы (вариант 19). Ввожу значение 32, программа выводит результат, наименьшую переменную. Программа работает

исправно, 32 и правда наименьшая из трех переменных. (рис. 4.17).

```
anastasiag@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-3.asm
anastasiag@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
anastasiag@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-3
Введите B: 32
Наименьшее число: 32
anastasiag@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.17: Результат работы программы

Код 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
a: RESB 80

SECTION .text
GLOBAL _start
_start:
mov eax, msg_x
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax ; edi = x

mov eax, msg_a
call sprint
mov ecx, a
```

```

mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax ; esi = a

cmp edi, esi ; сравниваю x и a
jg add_values ; если x > a, перехожу в add_values
mov eax, edi ; если x <= a, вывожу x
jmp print_result

add_values: ; x > a, вывожу a + x
mov eax, edi
add eax, esi ; eax = a + x

print_result:
mov edi, eax
mov eax, res
call sprint

mov eax, edi
call iprintLF
call quit

```

Для второго задания самостоятельной работы создаю файл lab7-4.asm и пишу программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Выбираю вид функции $f(x)$ в соответствии с моим вариантом 19. (рис. 4.18).

```

#include 'in_out.asm'

SECTION .data
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprint

mov ecx, x
mov edx, 80
call sread

mov eax, x
call atoi

xor edx, edx
mov ebx, 3
div ebx
add eax, 5
mov ebx, 7
mul ebx

mov edi, eax

mov eax, rem
call sprint
mov eax, edi
call iprintLF

call quit

```

Рис. 4.18: Текст программы файла lab7-4.asm

Создаю исполняемый файл и запускаю его. Проверяю его работу, введя значения переменных из таблицы 7.6. Сначала ввожу первую пару переменных (4 и 5), затем вторую (3 и 2). Программа работает исправно. (рис. 4.19).

```

anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-4.asm
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 4
Введите значение переменной a: 5
Результат: 4
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 3
Введите значение переменной a: 2
Результат: 5
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$

```

Рис. 4.19: Результат работы программы

Код программы:

```

#include 'in_out.asm'

section .data
msg1 db 'Введите B: ',0h
msg2 db "Наименьшее число: ",0h
A dd '46'
C dd '74'

section .bss
min 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 '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 ; 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:
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'
jb 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 sprint ; Вывод сообщения 'Наименьшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход

```

5 Выводы

6 Список литературы