Шаблон отчёта по лабораторной работе

Лабораторная работа №7.Команды безусловного и условного переходов в Nasm.Программирование ветвлений.

Акрур Имад НКАбд-06-24

Содержание

# 1 Цель работы

Изучение команд безусловного и условного переходов. Приобретение навыков написания программ с использованием переходов. Ознакомление с назначением и структурой файла листинга.

# 2 Описание результатов выполнения лабораторной работы

##описание выполняемого задания :

### 2.0.1 Реализация переходов в NASM :

1. Создать файл lab7-1.asm :

touch lab7-1.asm

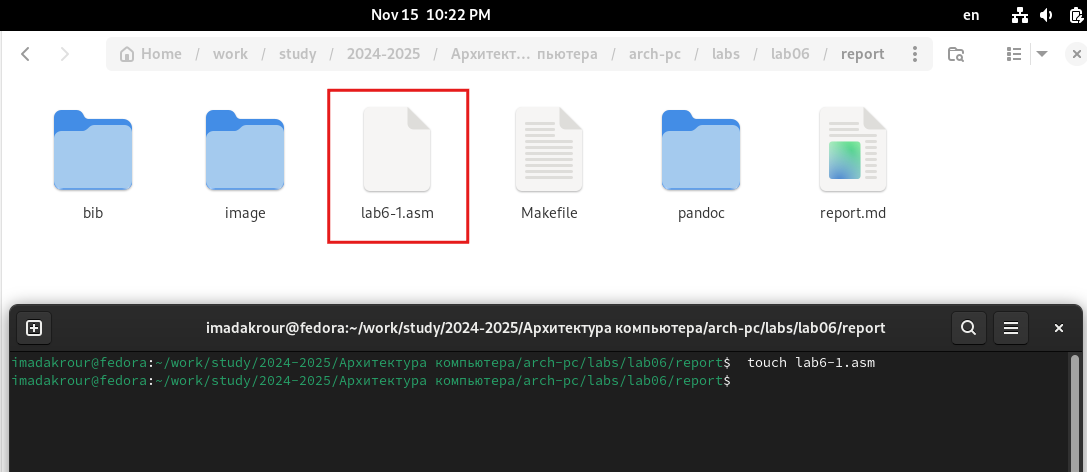


Рис. 1: Создание файла программы lab6-1.asm

Каталог создан и подготовлен файл lab7-1.asm для записи кода программы.

1. В файл lab7-1.asm введён код программы из листинга 7.1

**Код программы (Листинг 7.1):**

%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 \_label2Лабораторная работа №7.Команды безусловного и  
 условного переходов в Nasm.Программирование  
 ветвлений.  
  
 \_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

1. Создан исполняемый файл и запущена программа.

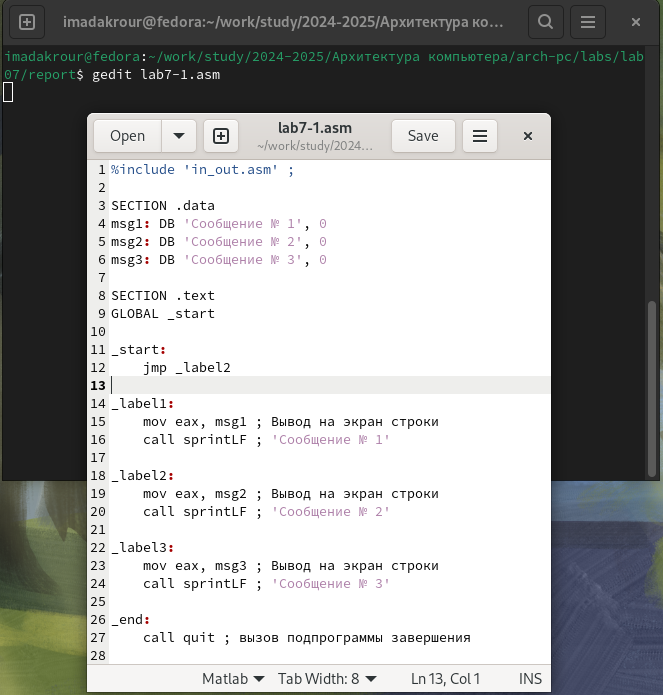


Рис. 2: Сборка и запуск программы lab7-1

**Команды для сборки и выполнения:**

nasm -f elf31 lab7-1.asm  
 ld -m elf\_i386 -o lab7-1   
 ./lab7-1

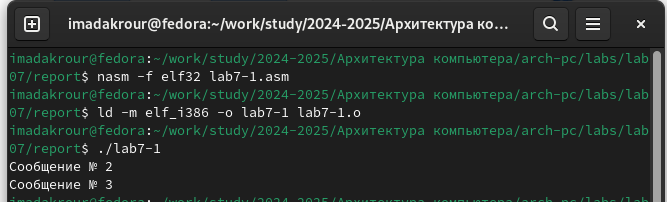


Рис. 3: Компиляция lab7-1.asm с использованием NASM

**Комментарии:**

Использование инструкции jmp \_label2 изменяет порядок исполнения инструкций, пропуская вывод первого сообщения и начиная выполнение программы с метки \_label2.

**Результат работы программы:**

/lab7-1  
Сообщение № 2  
Сообщение № 3

### 2.0.2 Изменение программы для вывода других сообщений

**Описание выполняемого задания:**

Программа изменена так, чтобы она выводила сообщения в следующем порядке: 1. “Сообщение № 2” 2. “Сообщение № 1” 3. Завершение работы.

**Изменённый код программы :**

%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'  
 jmp \_end  
  
\_label2:  
 mov eax, msg2 ; Вывод на экран строки  
 call sprintLF ; 'Сообщение № 2'  
 jmp \_label1  
  
\_label3:  
 mov eax, msg3 ; Вывод на экран строки  
 call sprintLF ; 'Сообщение № 3'  
  
\_end:  
 call quit ; вызов подпрограммы завершения

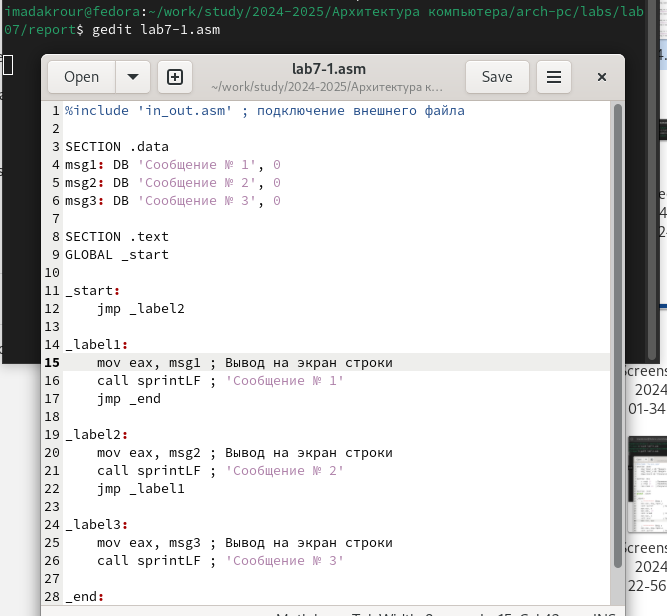


Рис. 4: Изменение программы lab7-1 для изменения порядка вывода

**Результат работы программы:**

/lab7-1  
Сообщение № 2  
Сообщение № 1

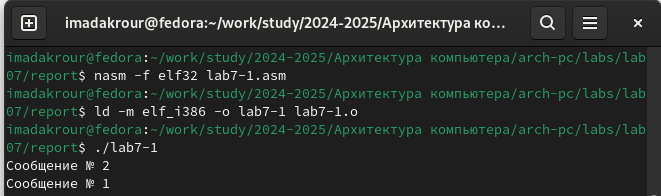


Рис. 5: Результат выполнения изменённой программы lab7-1

**Комментарии:**

Добавление инструкций jmp \_label1 и jmp \_end позволяет задать порядок выполнения программы, который отличается от последовательного.

### 2.0.3 использование инструкции jmp \_label2 :

так, чтобы она выводила сообщения в следующем порядке: 1. “Сообщение № 3” 2. “Сообщение № 2” 3. “Сообщение № 1”

**Изменённый код программы:**

%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 ; Переход сразу к выводу "Сообщение № 3"  
  
\_label1:  
 mov eax, msg1 ; Вывод на экран строки  
 call sprintLF ; 'Сообщение № 1'  
 jmp \_end ; Завершение программы  
  
\_label2:  
 mov eax, msg2 ; Вывод на экран строки  
 call sprintLF ; 'Сообщение № 2'  
 jmp \_label1 ; Переход к выводу "Сообщение № 1"  
  
\_label3:  
 mov eax, msg3 ; Вывод на экран строки  
 call sprintLF ; 'Сообщение № 3'  
 jmp \_label2 ; Переход к выводу "Сообщение № 2"  
  
\_end:  
 call quit ; Завершение программы

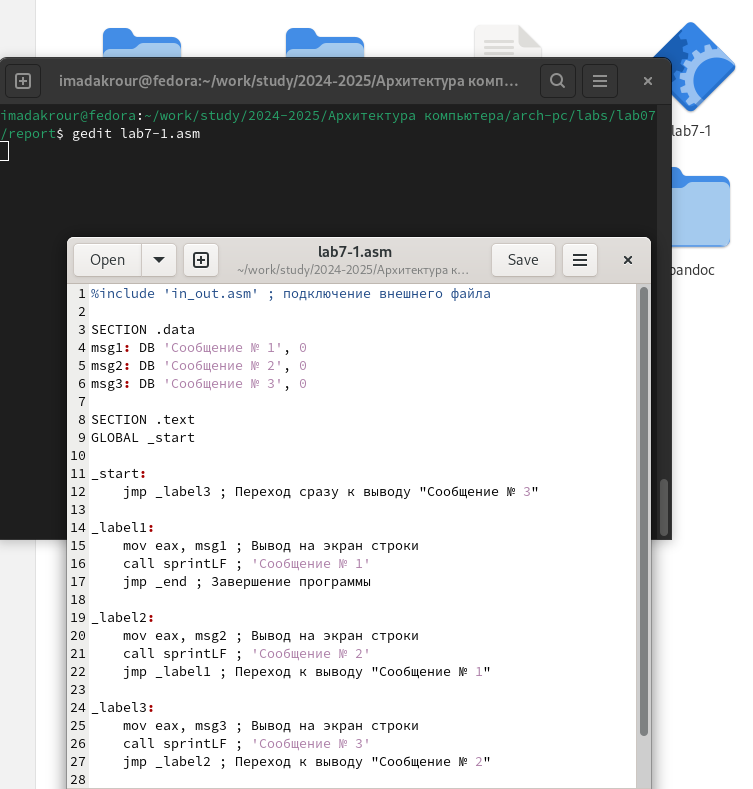


Рис. 6: Выполнение lab7-1 с сообщениями в обратном порядке

**Команды для сборки и выполнения:**

nasm -f elf32 lab7-1.asm  
ld -m lf\_i386 -o lab7-1 lab7-1.o  
./lab7-1

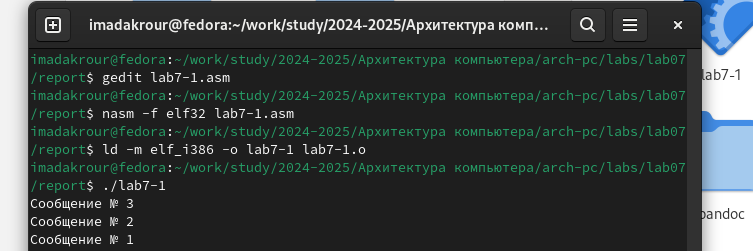


Рис. 7: Компиляция и выполнение обновлённой программы lab7-1

**Результат работы программы:**

./lab7-1  
Сообщение № 3  
Сообщение № 2  
Сообщение № 1

**Комментарии:**

* Использование инструкций jmp \_label2 и jmp \_label1 позволяет управлять порядком выполнения программы и выводить сообщения в обратном порядке.

### 2.0.4 Программа, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C.

Создать программу, которая определяет наибольшую из трёх целочисленных переменных: A, B и C. - Значения A и C задаются в коде программы. - Значение B вводится пользователем с клавиатуры.

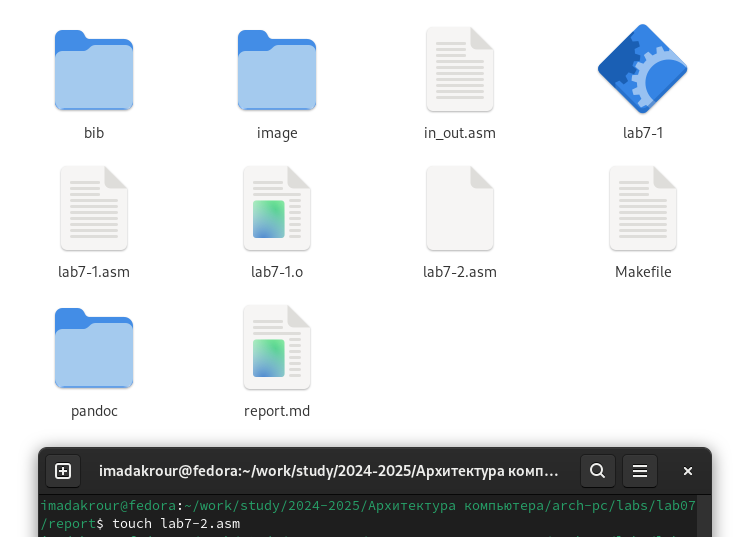


Рис. 8: Определение максимума трёх целых чисел в lab7-2

**Код программы (Листинг 7.3):**

%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  
 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 'С'  
 cmp ecx, [C]  
 jg check\_B ; Если 'A > C', перейти на 'check\_B'  
 mov ecx, [C] ; Иначе 'ecx = C'  
 mov [max], ecx ; 'max = C'  
  
check\_B:  
 ; ---------- Преобразование 'max' из символа в число  
 mov eax, max  
 call atoi  
 mov [max], eax  
  
 ; ---------- Сравниваем 'max' и 'B'  
 mov ecx, [max]  
 cmp ecx, [B]  
 jg fin ; Если 'max > B', перейти на 'fin'  
 mov ecx, [B] ; Иначе 'ecx = B'  
 mov [max], ecx  
  
fin:  
 ; ---------- Вывод результата  
 mov eax, msg2  
 call sprint ; Вывод сообщения 'Наибольшее число: '  
 mov eax, [max]  
 call iprintLF ; Вывод 'max(A, B, C)'  
 call quit ; Завершение программы

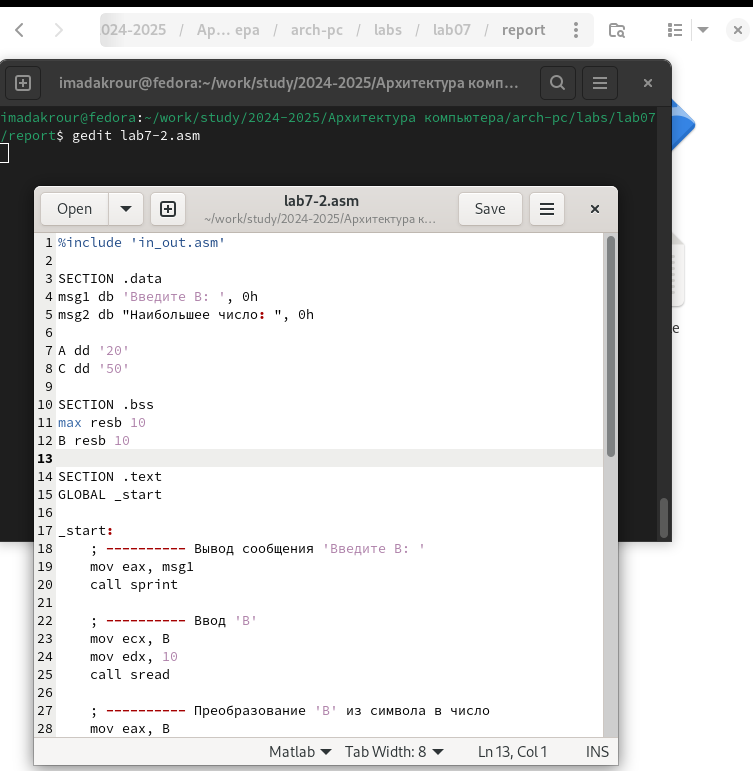


Рис. 9: Компиляция lab7-2.asm с использованием NASM

**Команды для сборки и выполнения:**

nasm -f elf -o lab7-2.o lab7-2.asm  
ld -o lab7-2 lab7-2.o  
./lab7-2

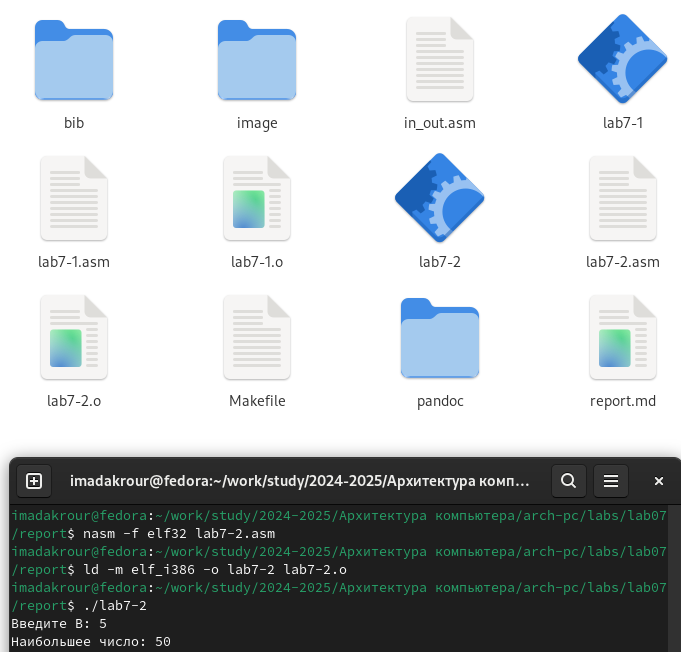


Рис. 10: Результат выполнения программы lab7-2, показывающий максимум

**Комментарии:**

* Переменные A и C задаются в программе.
* Значение B вводится пользователем.
* Наибольшее значение из трёх переменных определяется с использованием условных переходов (cmp, jg) и функции atoi для преобразования символов в числа.

**Результаты работы:** - При вводе разных значений B, программа корректно определяет максимальное из трёх чисел.

### 2.0.5 Создание файла листинга\*

1. Для программы lab7-2.asm создан файл листинга с помощью следующей команды:

nasm -f elf -l lab7-2.lst lab7-2.asm

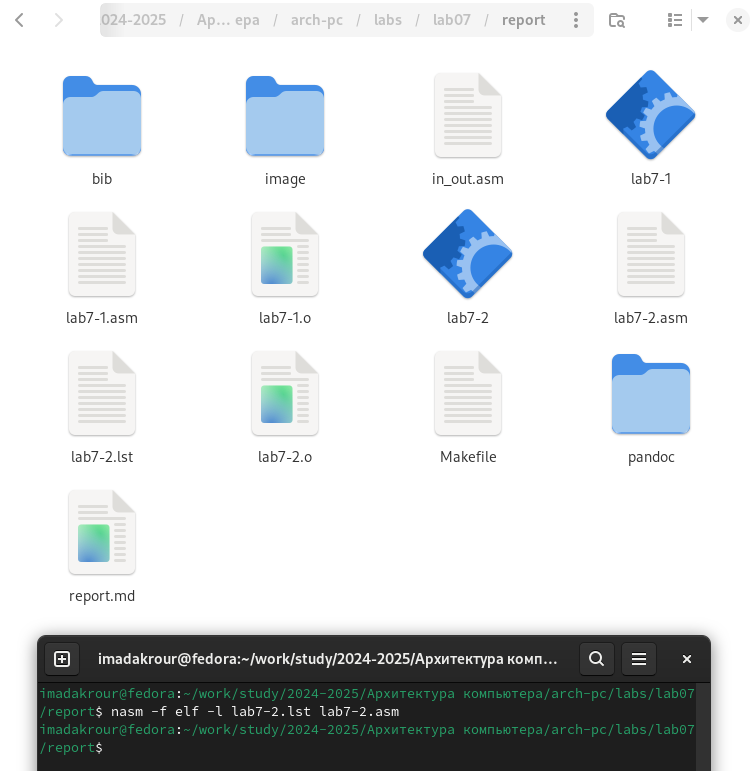


Рис. 11: Создание файла листинга для lab7-2.asm

1. Анализ содержимого файла листинга\*\*

Открыт файл lab7-2.lst в текстовом редакторе:

gedit lab7-2.lst

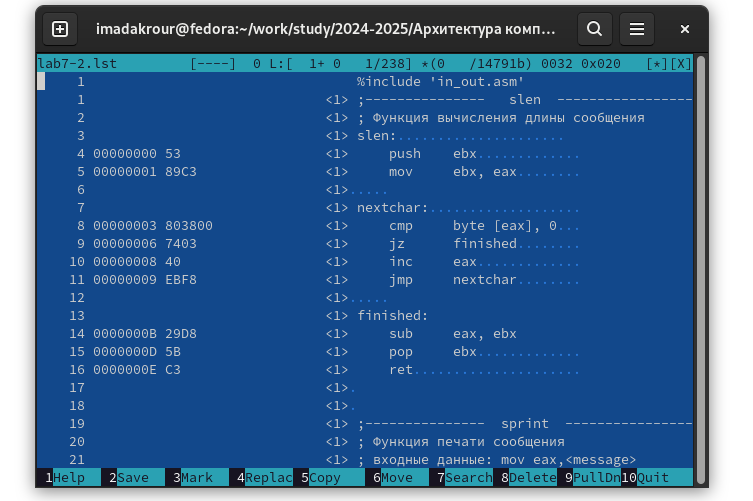


Рис. 12: Просмотр сгенерированного файла листинга в текстовом редакторе

1. Пример содержимого файла листинга (три строки):

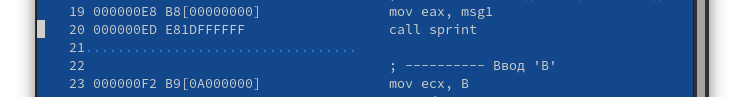


Рис. 13: Анализ строк файла листинга lab7-2.lst

1 000000E8 B8[00000000] mov eax, msg1  
2 000000ED E81DFFFFFF call sprint  
3 000000F2 B9[0A000000] mov ecx, B

**Подробный разбор строк:** **Строка 1:**

1 000000E8 B8[00000000] mov eax, msg1

* **1:** Номер строки из исходного файла программы.
* **000000E8:** Смещение (адрес) команды в сегменте кода.
* **B8[00000000]** Машинный код команды mov eax, msg1. Эта команда загружает адрес сообщения msg1 в регистр eax.

**Строка 2:**

2 000000ED E81DFFFFFF call sprint

* **2:** Номер строки исходного файла.
* **000000ED:** Смещение команды в памяти.
* **E81DFFFFFF:** Машинный код команды call sprint. E8 — машинный код вызова подпрограммы, а E81DFFFFFF — адрес подпрограммы, который будет заполнен на этапе связывания.

**Строка 3:**

3 000000F2 B9[0A000000] mov ecx, B

* **3:** Номер строки из исходного файла программы.
* **000000F2:** Смещение команды в сегменте.
* **B9[0A000000]**: Машинный код команды mov ecx, [B], которая загружает значение переменной B в регистр ecx.

1. Удаление операнда и повторное создание листинга:

В исходном файле lab7-2.asm из команды mov ecx, [B] удалён один операнд. Команда изменилась на:

mov ecx

Рис. 14: Удаление операнд

Рис. 14: Удаление операнд

**Трансляция:**

nasm -f elf -l lab7-2.lst lab7-2.asm

**Результат:**

1. Файл **lab7-2.o** не создан, так как транслатор обнаружил ошибку.
2. В файл **lab7-2.lst** добавлено сообщение об ошибке:

error: instruction expects 2 operands

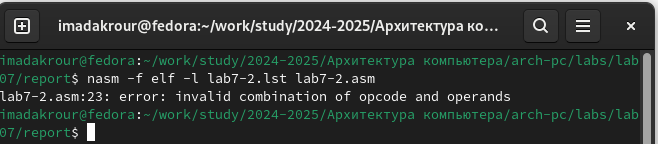


Рис. 15: Ошибка в листинге из-за отсутствия операнда

* Листинг дополнен диагностикой, описывающей проблему в строке с неполной инструкцией.

**Комментарии и выводы:**

* Файл листинга полезен для отладки, так как он отображает соответствие между исходным кодом, машинным кодом и адресами.
* При наличии ошибок в исходном коде, NASM добавляет сообщения об ошибках в файл листинга, но не создаёт объектный файл.

## 2.1 **Выводы по результатам выполнения заданий**

1. Программа с использованием безусловных переходов успешно реализована, демонстрируя изменение порядка выполнения инструкций через команды jmp.
2. Изменённая программа корректно выводит сообщения в заданной последовательности благодаря управлению переходами.
3. Программа для определения максимального из трёх чисел эффективно использует условные переходы (cmp, jg), обеспечивая корректный результат.
4. Генерация и анализ файла листинга показали его полезность для отладки и выявления ошибок в коде.

# 3 Описание результатов выполнения заданийдля самостоятельной работы

## 3.1 **описание выполняемого задания**

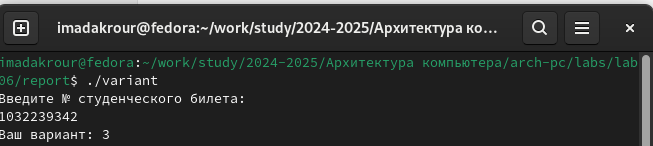


Рис. 16: вариант номер

### 3.1.1 Программа нахождения наименьшей из трёх целочисленных переменных\*\*

Написать программу для определения наименьшей из трёх целочисленных переменных ( a, b, c ).

Значения переменных выбираются из **таблицы 7.5**, для **варианта 3**:

* ( a = 94 )
* ( b = 5 )
* ( c = 58 )

Код программы:

section .data  
 msg\_min db "Наименьшее число: ", 0h ; Сообщение для вывода результата  
 a dd 94 ; Значение переменной a  
 b dd 5 ; Значение переменной b  
 c dd 58 ; Значение переменной c  
section .bss  
 min resb 10 ; Память для результата  
  
section .text  
global \_start  
  
\_start:  
 ; ---------- Инициализируем min значением 'a'  
 mov eax, [a]  
 mov [min], eax  
  
 ; ---------- Сравниваем 'min' с 'b'  
 mov eax, [min]  
 cmp eax, [b]  
 jl check\_c ; Если min < b, перейти на проверку c  
 mov eax, [b]  
 mov [min], eax  
  
check\_c:  
 ; ---------- Сравниваем 'min' с 'c'  
 mov eax, [min]  
 cmp eax, [c]  
 jl fin ; Если min < c, перейти к выводу результата  
 mov eax, [c]  
 mov [min], eax  
  
fin:  
 ; ---------- Выводим результат  
 mov eax, msg\_min  
 call sprint ; Вывод сообщения  
 mov eax, [min]  
 call iprintLF ; Вывод минимального значения  
  
 ; ---------- Завершение программы  
 call quit

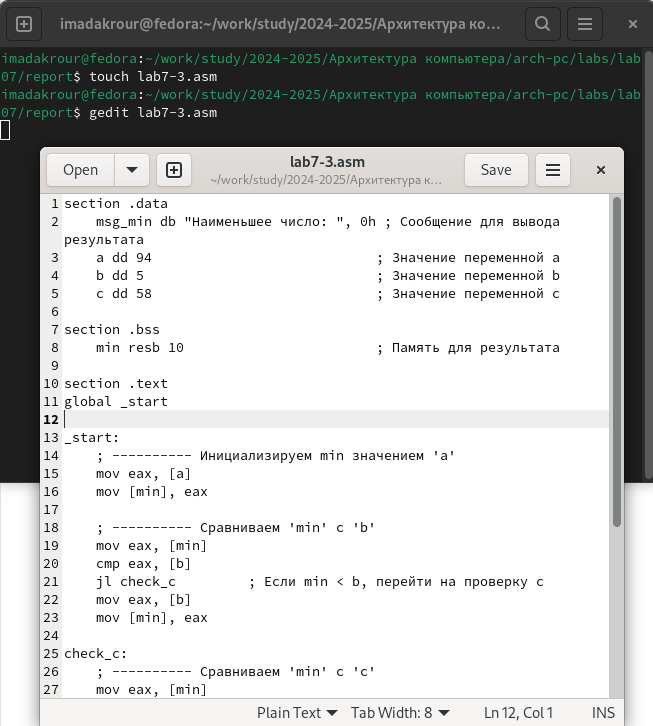


Рис. 17: Создание файла программы lab6-3.asm

**Команды для сборки и запуска программы:**

nasm -f elf32 lab7-3.asm  
ld -m elf\_i386 -o lab7-3 lab7-3.o  
./lab7-3

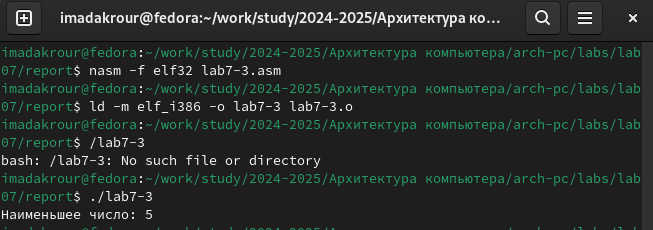


Рис. 18: Результат программы, показывающий минимальное значение

**Результаты работы программы:**

При значениях ( a = 94 ), ( b = 5 ), ( c = 58 ),

программа выдаёт:

Наименьшее число: 5

### 3.1.2 Программа вычисления функции ( f(x) )

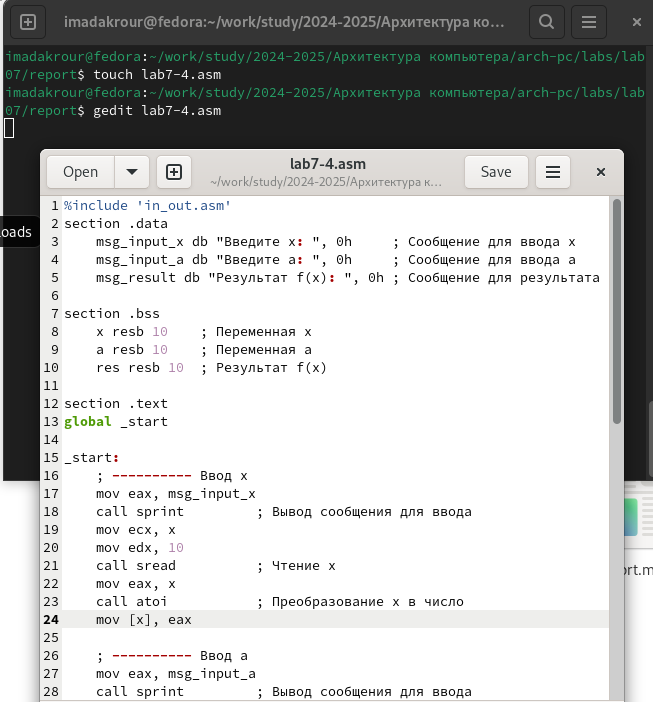


Рис. 19: Создание файла программы lab6-4.asm

Для введённых с клавиатуры значений ( x ) и ( a ), вычислить значение функции ( f(x) ). Функция для **варианта 3** из **таблицы 7.6**:

Начальные значения: 1. ( x\_1 = 3, a\_1 = 4 ) 2. ( x\_2 = 1, a\_2 = 4 )

**Код программы:**

section .data  
 msg\_input\_x db "Введите x: ", 0h ; Сообщение для ввода x  
 msg\_input\_a db "Введите a: ", 0h ; Сообщение для ввода a  
 msg\_result db "Результат f(x): ", 0h ; Сообщение для результата  
  
section .bss  
 x resb 10 ; Переменная x  
 a resb 10 ; Переменная a  
 res resb 10 ; Результат f(x)  
  
section .text  
global \_start  
  
\_start:  
 ; ---------- Ввод x  
 mov eax, msg\_input\_x  
 call sprint ; Вывод сообщения для вводаЛабораторная работа №7.Команды безусловного и  
 условного переходов в Nasm.Программирование  
 ветвлений.  
 mov ecx, x  
 mov edx, 10  
 call sread ; Чтение x  
 mov eax, x  
 call atoi ; Преобразование x в число  
 mov [x], eax  
  
 ; ---------- Ввод a  
 mov eax, msg\_input\_a  
 call sprint ; Вывод сообщения для ввода  
 mov ecx, a  
 mov edx, 10  
 call sread ; Чтение a  
 mov eax, a  
 call atoi ; Преобразование a в число  
 mov [a], eax  
  
 ; ---------- Вычисление f(x)  
 mov eax, [x]  
 cmp eax, 3 ; Сравнение x с 3  
 je equal\_x ; Если x == 3, перейти на equal\_x  
  
not\_equal\_x:  
 mov eax, [a]  
 add eax, 1 ; Вычисление a + 1  
 jmp output\_result  
  
equal\_x:  
 mov eax, [x]  
 imul eax, 3 ; Вычисление 3x  
  
output\_result:  
 mov [res], eax ; Сохранение результата в res  
  
 ; ---------- Вывод результата  
 mov eax, msg\_result  
 call sprint ; Вывод сообщения результата  
 mov eax, [res]  
 call iprintLF ; Вывод значения f(x)  
  
 ; ---------- Завершение программы  
 call quit

**Команды для сборки и запуска программы:**

nasm -f elf -o lab7-4.asm  
ld -o lab7-4 lab7-4.o  
./lab7-4

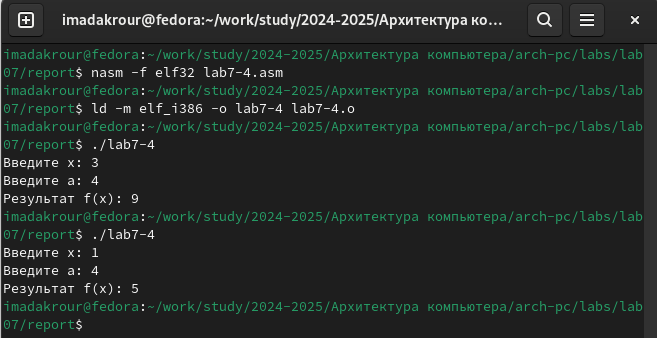


Рис. 20: Вывод результата программы для вычисления f(x)

**Пример ввода и вывода программы:**

1. При вводе ( x = 3, a = 4 ):

Введите x: 3  
 Введите a: 4  
 Результат f(x): 9

1. При вводе ( x = 1, a = 4 ):

Введите x: 1  
 Введите a: 4  
 Результат f(x): 5

## 3.2 Выводы по результатам выполнения самостоятельной работы

1. Программа для нахождения наименьшего числа успешно реализована и корректно определяет минимальное значение из трёх целых чисел.
2. Программа для вычисления функции ( f(x) ) корректно обрабатывает входные данные и вычисляет результат в зависимости от условия ( x = 3 ) или ( x ).

# 4 **Выводы**

1. Работа успешно выполнена, продемонстрированы навыки использования безусловных и условных переходов в NASM.
2. Реализованы программы для управления порядком выполнения инструкций, нахождения максимума и минимума, а также вычисления функции ( f(x) ).
3. Генерация и анализ файла листинга укрепили понимание структуры программ на ассемблере и инструментов отладки.