# Отчёт по лабораторной работе №7

**Дисциплина: архитектура компьютеров и операционные системы**

Хань Цзянтао

# Содержание

1. [Цель работы](#_bookmark0) 5
2. [Задание](#_bookmark1) 6
3. [Теоретическое введение](#_bookmark2) 7
4. [Выполнение лабораторной работы](#_bookmark3) 8
   1. [Реализация переходов в NASM](#_bookmark4) . . . . . . . . . . . . . . . . . . 8
   2. [Изучение структуры файлы листинга](#_bookmark15) 13
   3. [Задания для самостоятельной работы](#_bookmark21) 14
5. [Выводы](#_bookmark26) 20
6. [Список литературы](#_bookmark27) 21

**Список иллюстраций**

* 1. [Создание файлов для лабораторной работы](#_bookmark5) . . . . . . . . . . . . 8
  2. [Ввод текста программы из листинга 7.1](#_bookmark6) . . . . . . . . . . . . . . . 9
  3. [Запуск программного кода](#_bookmark7) . . . . . . . . . . . . . . . . . . . . . 9
  4. [Изменение текста программы](#_bookmark8) 10
  5. [Создание исполняемого файла](#_bookmark9) 10
  6. [Изменение текста программы](#_bookmark10) 11
  7. [Вывод программы](#_bookmark11) 11
  8. [Создание файла](#_bookmark12) 12
  9. [Ввод текста программы из листинга 7.3](#_bookmark13) 12
  10. [Проверка работы файла](#_bookmark14) 12
  11. [Создание файла листинга](#_bookmark16) 13
  12. [Изучение файла листинга](#_bookmark17) 13
  13. [Выбранные строки файла](#_bookmark18) 13
  14. [Удаление выделенного операнда из кода](#_bookmark19) 14
  15. [Получение файла листинга](#_bookmark20) 14
  16. [Написание программы](#_bookmark22) 15
  17. [Запуск файла и проверка его работы](#_bookmark23) 15
  18. [Написание программы](#_bookmark24) 17
  19. [Запуск файла и проверка его работы](#_bookmark25) 18

# Список таблиц

# Цель работы

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

# Задание

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

# Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые ко- манды 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.

Безусловный переход выполняется инструкцией jmp. Инструкция cmp является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция cmp является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания. Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имееттекстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную

информацию.

# Выполнение лабораторной работы

## Реализация переходов в NASM

Создаю каталог для программ лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm. (рис. [4.19).](#_bookmark25)

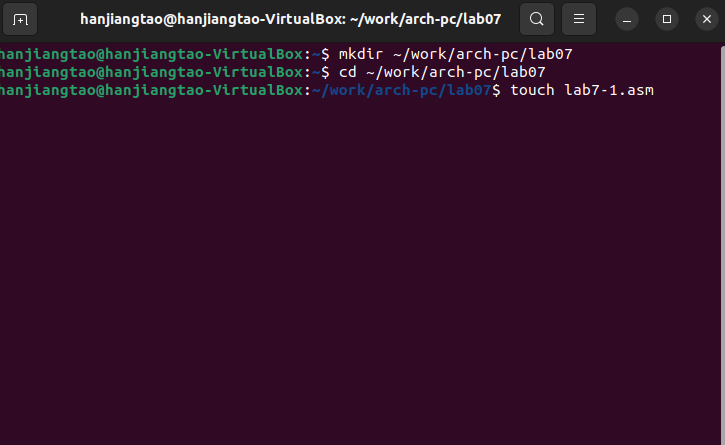


Рис. 4.1: Создание файлов для лабораторной работы

Ввожу в файл lab7-1.asm текст программы из листинга 7.1. (рис. [4.19).](#_bookmark25)

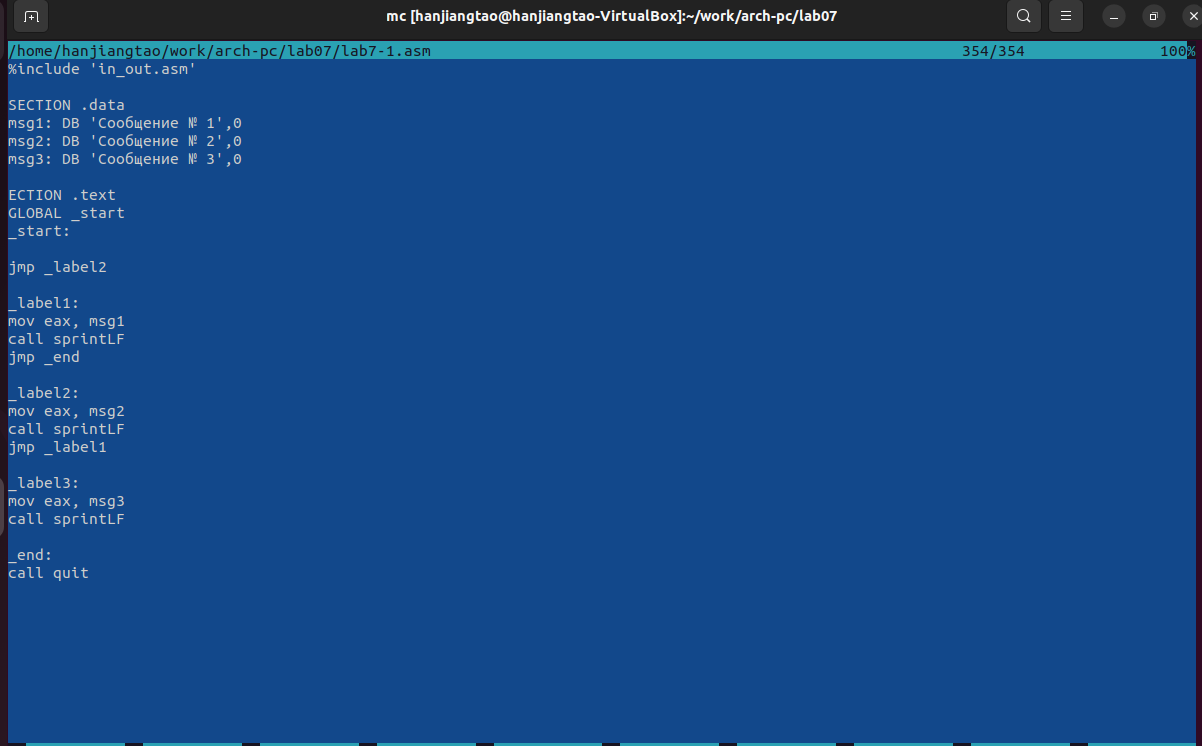


Рис. 4.2: Ввод текста программы из листинга 7.1

Создаю исполняемый файл и запускаю его. (рис. [4.19).](#_bookmark25)

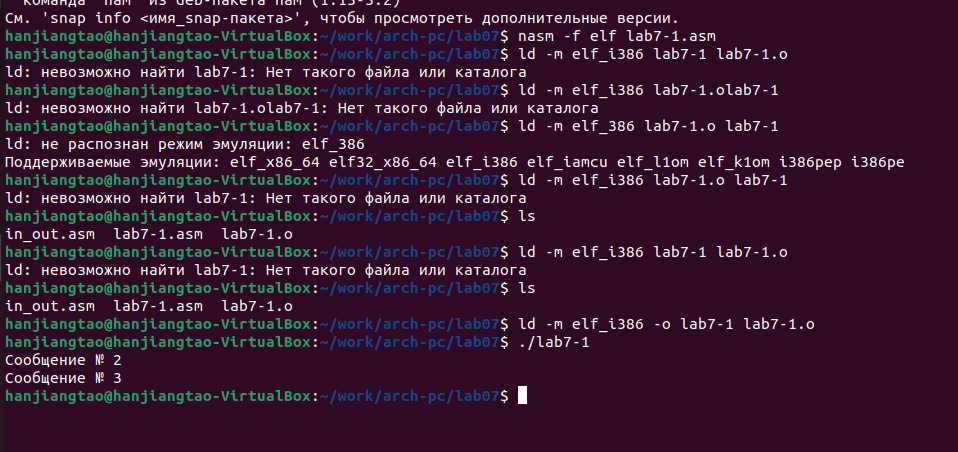


Рис. 4.3: Запуск программного кода

Таким образом, использование инструкции jmp \_label2 меняет порядок испол- нения инструкций и позволяет выполнить инструкции начиная с метки \_label2, пропустив вывод первого сообщения.

Изменю программу таким образом, чтобы она выводила сначала ‘Сообщение

№ 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого изменяю текст программы в соответствии с листингом 7.2. (рис. [4.19).](#_bookmark25)

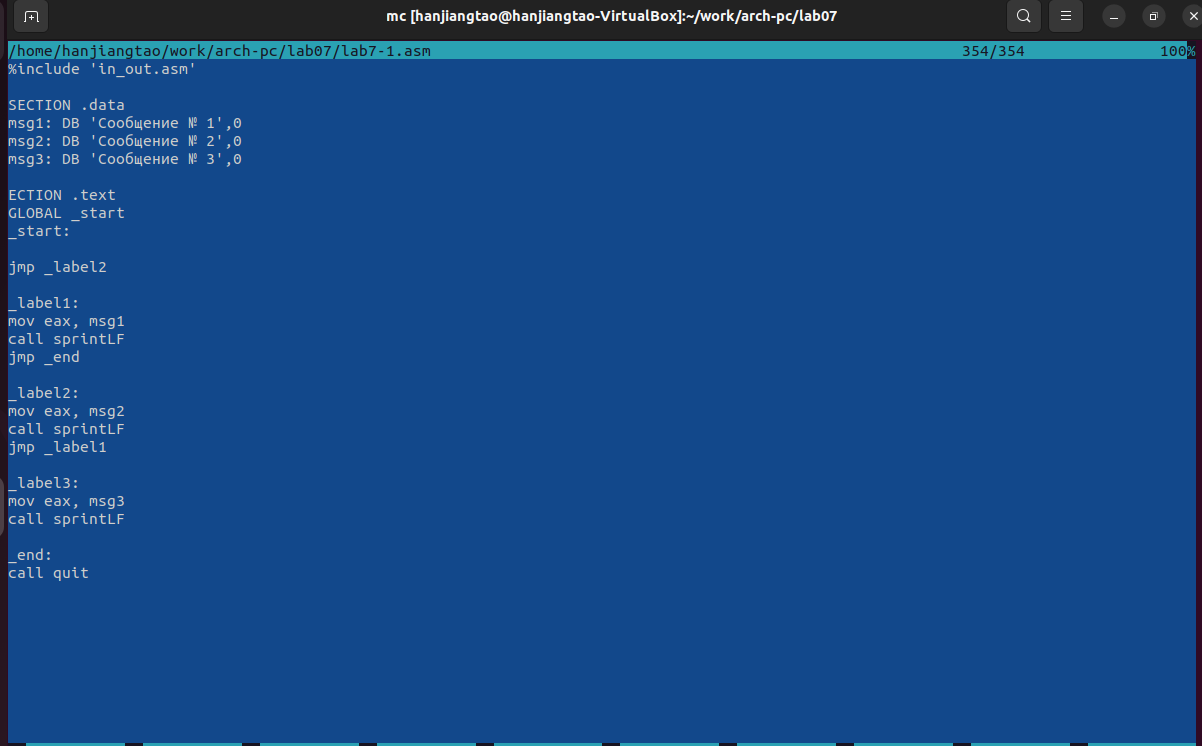


Рис. 4.4: Изменение текста программы

Создаю исполняемый файл и проверяю его работу. (рис. [4.19).](#_bookmark25)

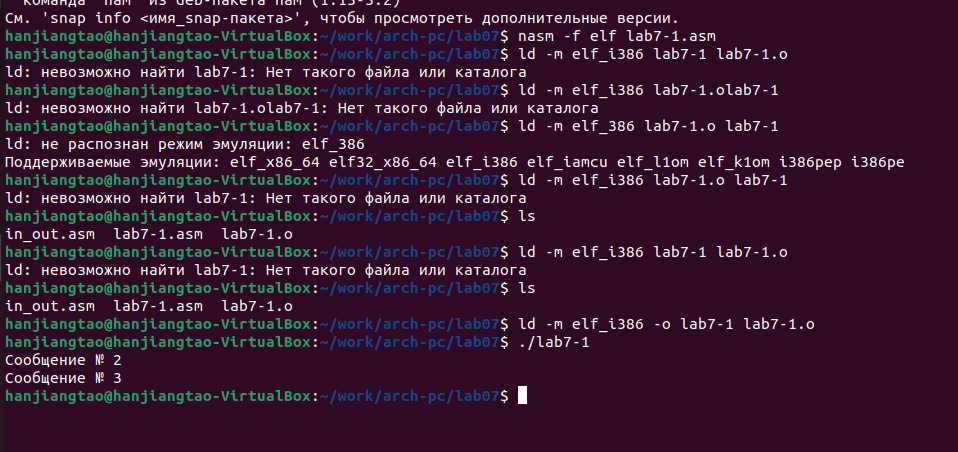


Рис. 4.5: Создание исполняемого файла

Затем изменяю текст программы, добавив в начале программы jmp \_label3, jmp \_label2 в конце метки jmp \_label3, jmp \_label1 добавляю в конце метки jmp

\_label2, и добавляю jmp \_end в конце метки jmp \_label1, (рис. [4.19).](#_bookmark25)

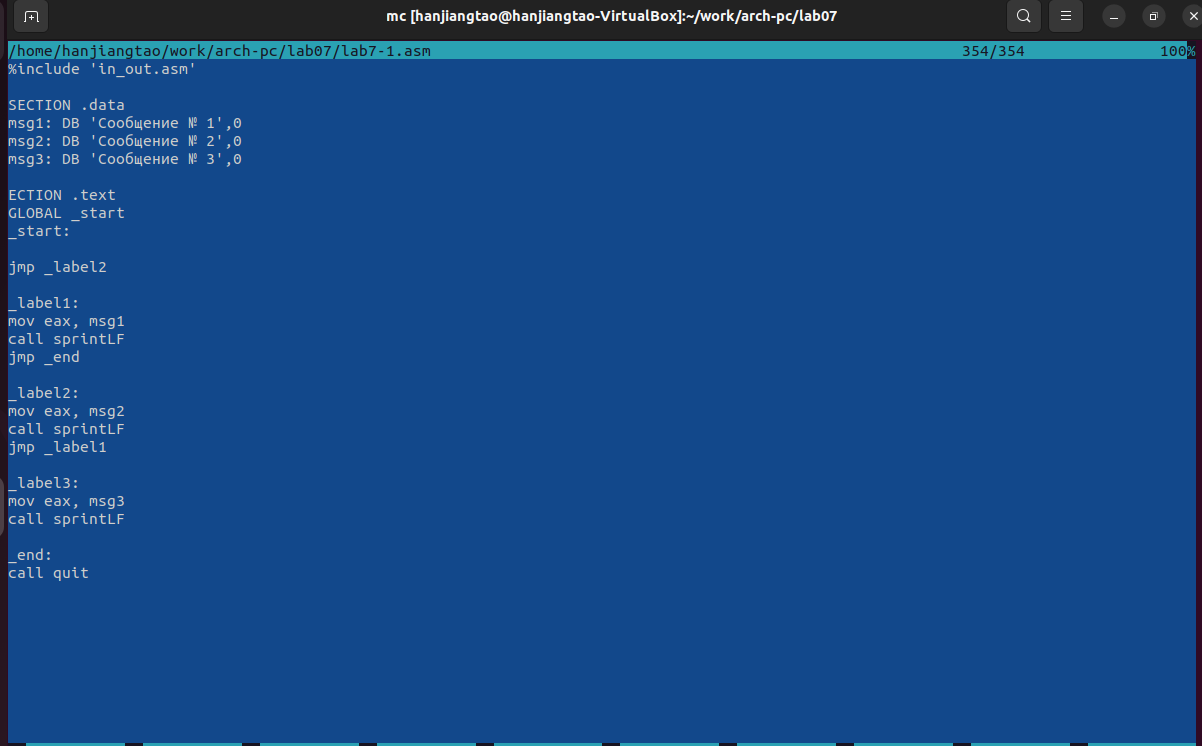


Рис. 4.6: Изменение текста программы

чтобы вывод программы был следующим: (рис. [4.19).](#_bookmark25)

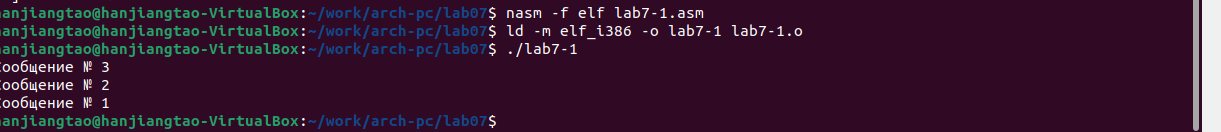


Рис. 4.7: Вывод программы

Рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C. Значения для A и C задаются в про- грамме, значение B вводиться с клавиатуры.

Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. (рис. [4.19).](#_bookmark25)

08

Рис. 4.8: Создание файла

Текст программы из листинга 7.3 ввожу в lab7-2.asm. (рис. [4.19).](#_bookmark25)

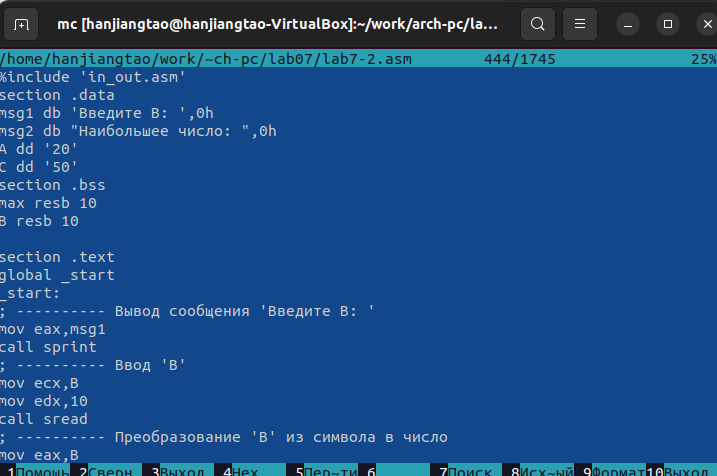


Рис. 4.9: Ввод текста программы из листинга 7.3

Создаю исполняемый файл и проверьте его работу. (рис. [4.19).](#_bookmark25)

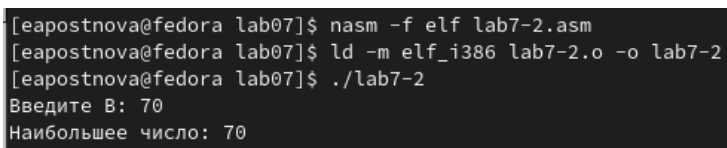


Рис. 4.10: Проверка работы файла

Файл работает корректно.

## Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.asm. (рис. [4.19).](#_bookmark25)

11

Рис. 4.11: Создание файла листинга

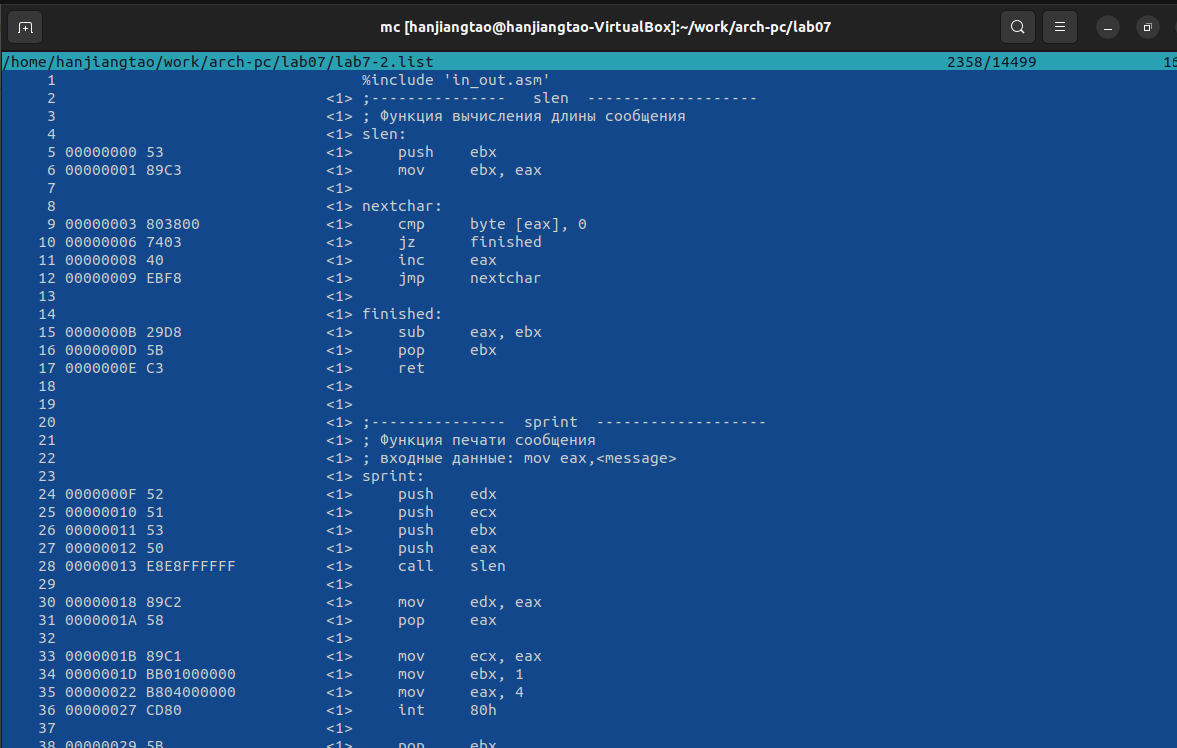
Открываю файл листинга lab7-2.lst с помощью текстового редактора и внима- тельно изучаю его формат и содержимое. (рис. [4.19).](#_bookmark25)

Рис. 4.12: Изучение файла листинга

В представленных трех строчках содержаться следующие данные: (рис. [4.19).](#_bookmark25)

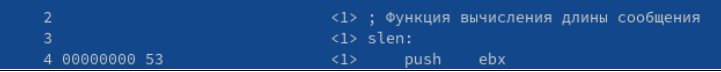


Рис. 4.13: Выбранные строки файла

“2” - номер строки кода, “; Функция вычисления длинны сообщения” - коммен- тарий к коду, не имеет адреса и машинного кода.

“3” - номер строки кода, “slen” - название функции, не имеет адреса и машин- ного кода.

“4” - номер строки кода, “00000000” - адрес строки, “53” - машинный код, “push ebx” - исходный текст программы, инструкция “push” помещает операнд “ebx” в стек.

Открываю файл с программой lab7-2.asm и в выбранной мной инструкции с двумя операндами удаляю выделенный операнд. (рис. [4.19).](#_bookmark25)

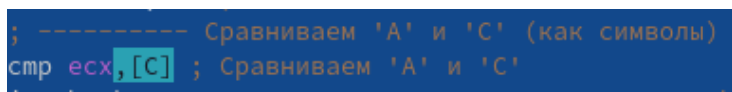


Рис. 4.14: Удаление выделенного операнда из кода

Выполняю трансляцию с получением файла листинга. (рис. [4.19).](#_bookmark25)

15

Рис. 4.15: Получение файла листинга

На выходе я не получаю ни одного файла из-за ошибки:инструкция mov (един- ственная в коде содержит два операнда) не может работать, имея только один операнд, из-за чего нарушается работа кода.

## Задания для самостоятельной работы

* + 1. Пишу программу нахождения наименьшей из 3 целочисленных перемен- ных a, b и c. Значения переменных выбираю из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Мой вариант под номером 10, поэтому мои значения - 41, 62 и 35. (рис. [4.19).](#_bookmark25)

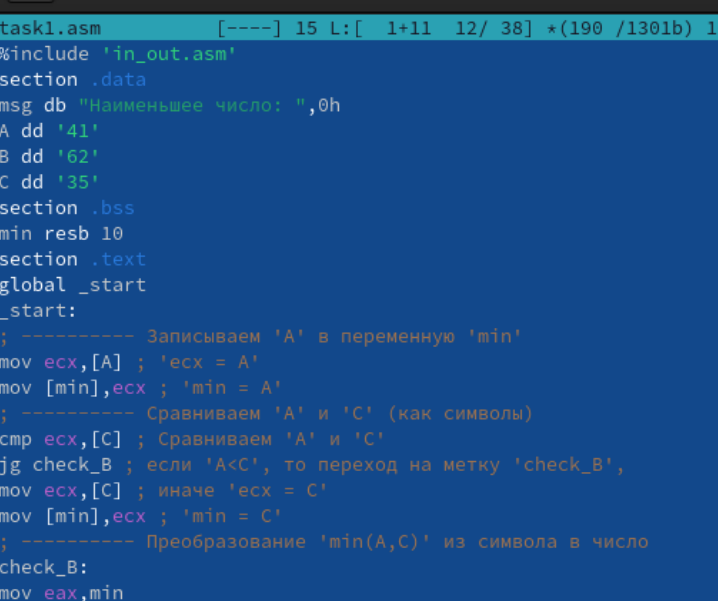


Рис. 4.16: Написание программы

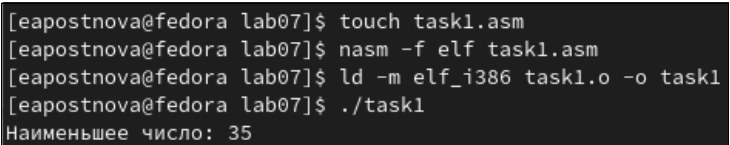
Создаю исполняемый файл и проверяю его работу, подставляя необходимые значение. (рис. [4.19).](#_bookmark25)

Рис. 4.17: Запуск файла и проверка его работы Программа работает корректно.

Код программы:

%include ‘in\_out.asm’ section .data

msg db “Наименьшее число:”,0h A dd ‘41’

B dd ‘62’

C dd ‘35’

section .bss min resb 10 section .text global \_start

\_start:

; ———- Записываем ‘A’ в переменную ‘min’ mov ecx,[A] ; ‘ecx = A’

mov [min],ecx ; ‘min = A’

; ———- Сравниваем ‘A’ и ‘С’ (как символы) cmp ecx,[C] ; Сравниваем ‘A’ и ‘С’

jg 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’

jl fin ; если ‘min(A,C)<B’, то переход на ‘fin’, mov ecx,[B] ; иначе ‘ecx = B’

mov [min],ecx

; ———- Вывод результата fin:

mov eax, msg

call sprint ; Вывод сообщения ‘Наименьшее число:’ mov eax,[min]

call iprintLF ; Вывод ‘min(A,B,C)’ call quit ; Выход

* + 1. Пишу программу, которая для введенных с клавиатуры значений х и а вычисляет значение и выводит результат вычислений заданной для моего варианта функции f(x):

х - 2, если х > 2 3 a, если х <= 2

(рис. [4.19).](#_bookmark25)

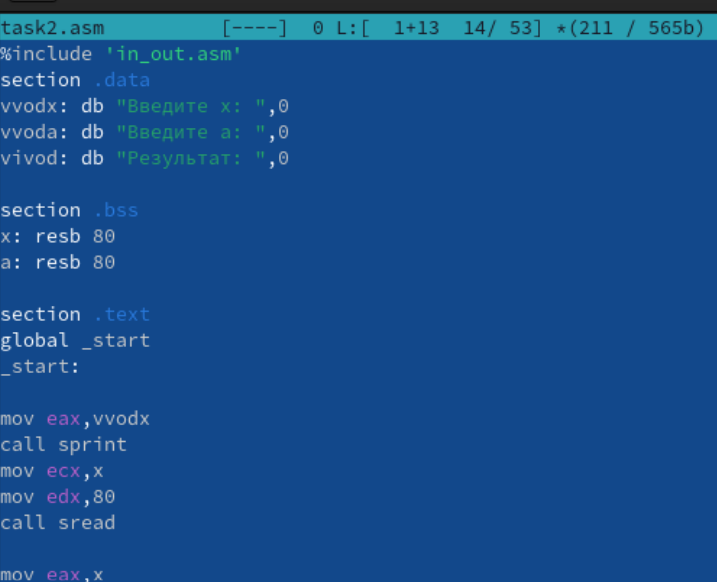


Рис. 4.18: Написание программы

Создаю исполняемый файл и проверяю его работу для значений х и а соответ- ственно: (3;0), (1;2). (рис. [4.19).](#_bookmark25)

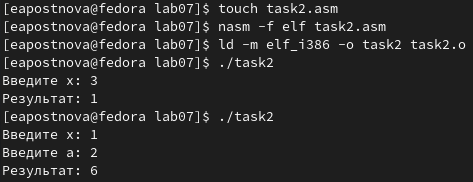


Рис. 4.19: Запуск файла и проверка его работы

Программа работает корректно. Код программы:

%include ‘in\_out.asm’ section .data

vvodx: db “Введите x:”,0 vvoda: db “Введите а:”,0 vivod: db “Результат:”,0 section .bss

x: resb 80

a: resb 80 section .text global \_start

\_start:

mov eax,vvodx call sprint mov ecx,x mov edx,80 call sread

mov eax,x call atoi cmp eax,2

jg \_functionx mov eax,vvoda call sprint mov ecx,a mov edx,80 call sread

mov eax,a call atoi

jmp \_functiona

\_functiona: mov edx,3 mul edx jmp \_end

\_functionx: add eax,-2 jmp \_end

\_end:

mov ecx,eax mov eax,vivod call sprint mov eax,ecx call iprintLF call quit

# Выводы

По итогам данной лабораторной работы я изучила команды условного и без- условного переходов, приобрела навыки написания программ с использованием переходов и ознакомилась с назначением и структурой файла листинга, что поможет мне при выполнении последующих лабораторных работ.

# Список литературы

* 1. GDB: The GNU Project Debugger. — URL: ht[tps://w](http://www.gnu.org/software/gdb/)ww[.gnu.org/software/gdb/.](http://www.gnu.org/software/gdb/)
  2. GNU Bash Manual. — 2016. — URL: ht[tps://w](http://www.gnu.org/software/bash/manual/)ww[.gnu.org/software/bash/manual/.](http://www.gnu.org/software/bash/manual/)
  3. Midnight Commander Development Center. — 2021. — URL: https://midnight- commander. org/.
  4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.
  5. Newham C. Learning the bash Shell: Unix Shell Programming. — O’Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: [http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658.](http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658)
  6. Robbins A. Bash Pocket Reference. — O’Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
  7. The NASM documentation. — 2021. — URL: ht[tps://w](http://www.nasm.us/docs.php)ww[.nasm.us/docs.php.](http://www.nasm.us/docs.php)
  8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
  9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
  10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
  11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
  12. Расширенный ассемблер: NASM.— 2021.— URL: [https://w](http://www.opennet.ru/docs/RUS/nasm/)ww[.opennet.ru/docs/RUS/nasm/.](http://www.opennet.ru/docs/RUS/nasm/)
  13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
  14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix.— 2-

е изд.— М.: МАКС Пресс, 2011.— URL: [http://www.stolyarov.info/books/asm\_unix.](http://www.stolyarov.info/books/asm_unix)

* 1. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
  2. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,2015. — 1120 с. — (Классика Computer Science).