Отчёт по лабораторной работе №7

дисциплина: Архитектура компьютера

Маслова Анна Павловна

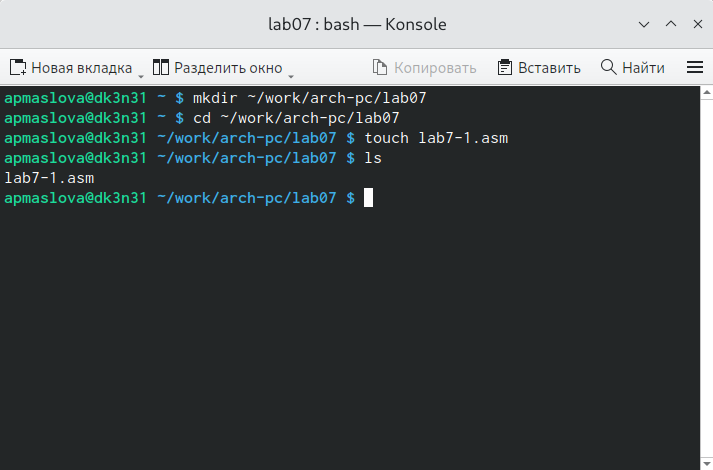
Содержание

# 1 Цель работы

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

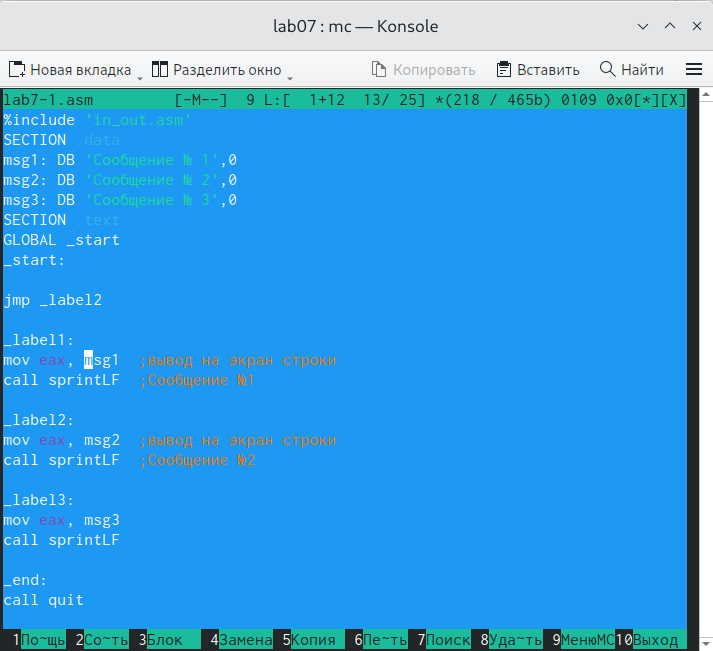
# 2 Выполнение лабораторной работы

Создаём каталог для программ лабораторной работы №7, перейдём в него и создадими файл *lab7-1.asm* (рис. ??).



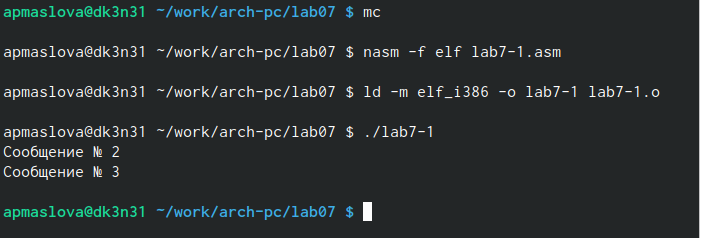
Создание каталога lab07 и файла lab7-1.asm

Введём в файл *lab7-1.asm* текст программы с использованием инструкции *jmp* (рис. ??).



Текст программы файла lab7-1.asm

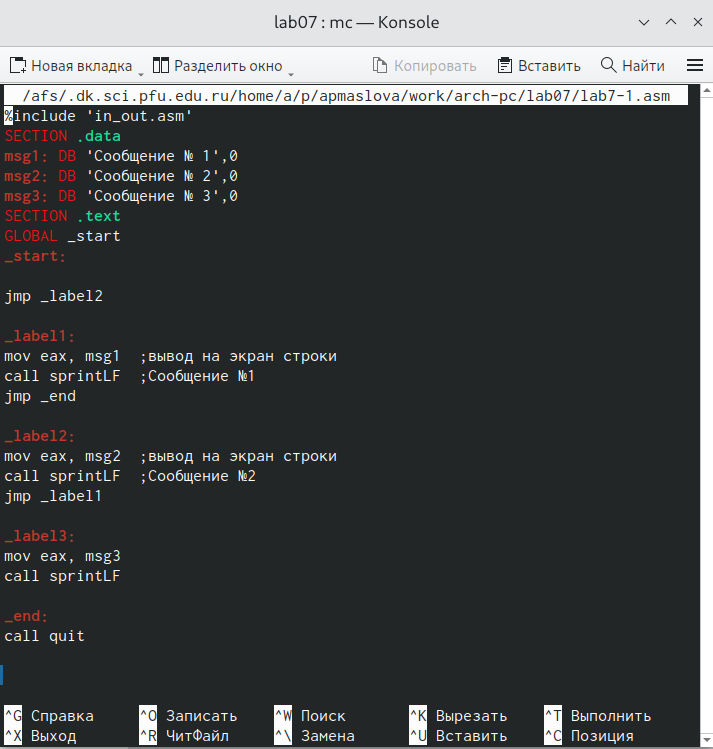
Создадим исполняемый файл и запустим его (рис. ??).



Запуск файла lab7-1

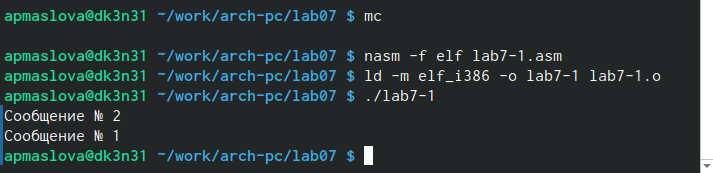
Мы видим, что “Сообщение №1” на экран не вывелось, но осуществился вывод “Сообщение №2” и “Сообщение №3”. Таким образом, использование инструкции jmp \_label2 меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки \_label2, пропустив вывод первого сообщения.

Изменим программу так, 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) (рис. ??).



Изменённый текст программы файла lab7-1.asm

Создадим исполняемый файл и проверим его работу (рис. ??).

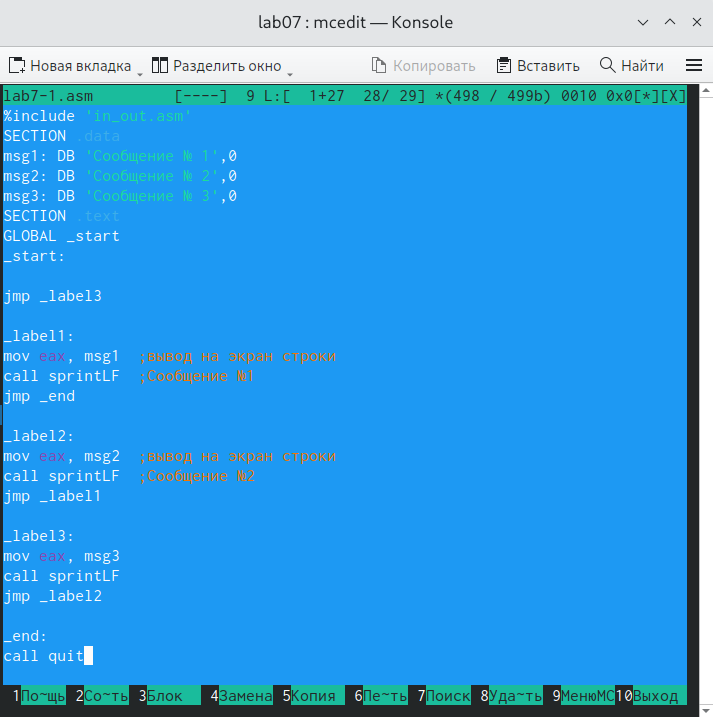


Запуск изменённого файла lab7-1

Как мы видим, программа работает корректно: выводится сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’, и работа завершилась.

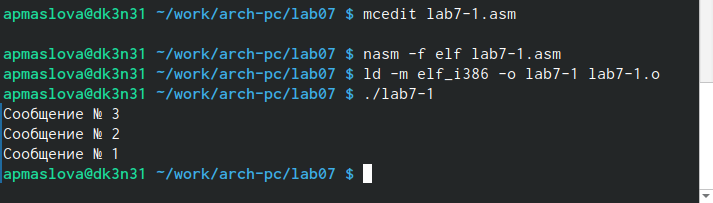
Теперь изменим текст программы (рис. ??)так, чтобы вывод программы был следующим:

Сообщение № 3  
Сообщение № 2  
Сообщение № 1



Повторно изменённый текст файла lab7-1.asm

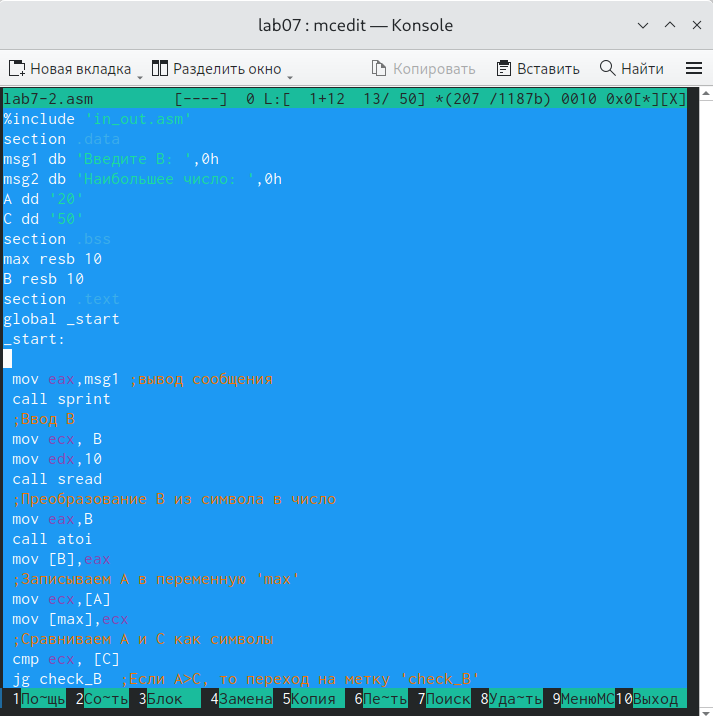
Создадим исполняемый файл и проверим его работу (рис. ??).



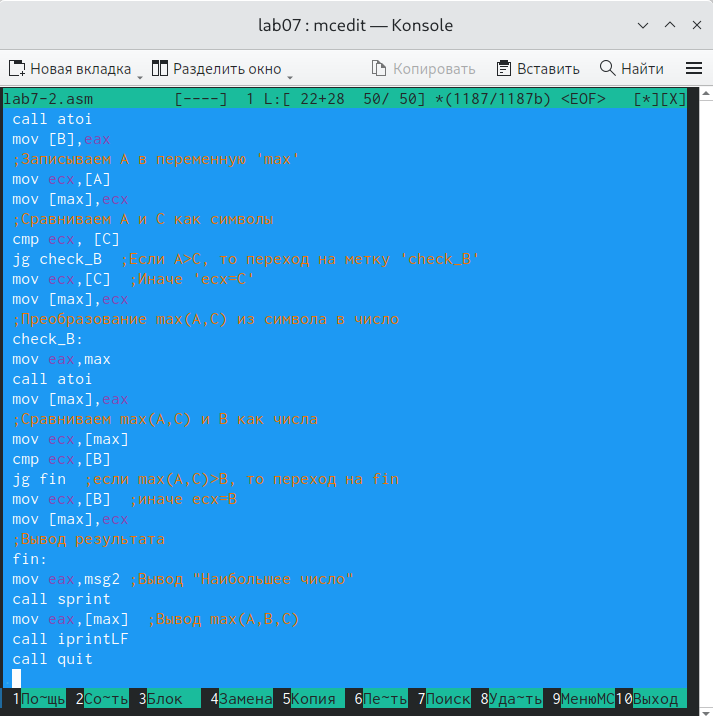
Запуск повторно изменённого файла lab7-1

Как мы видим, программа выводит нужные сообщения в верной последовательности.

Далее создадим файл *lab7-2.asm* в каталоге *~/work/arch-pc/lab07*. В этот файл введём текст программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C (рис. ??, рис. ??).

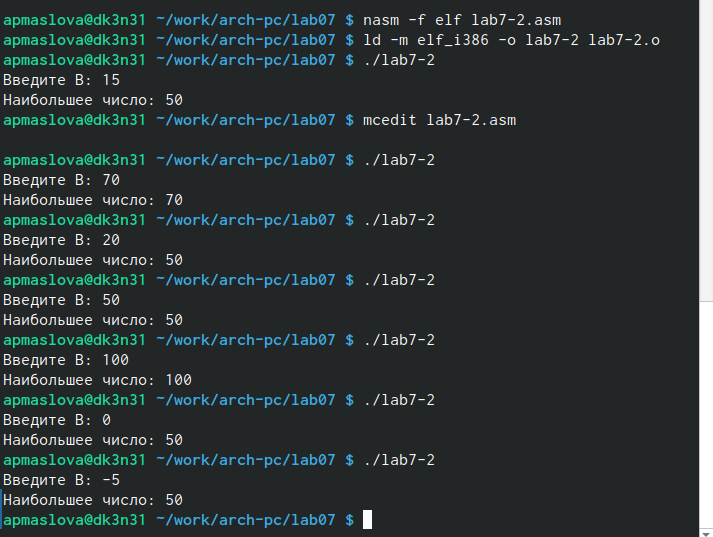


Текст файла lab7-2.asm



Текст файла lab7-2.asm

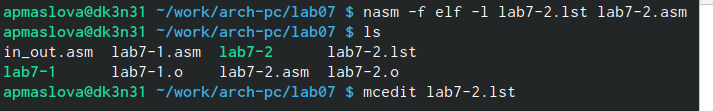
Теперь создадим исполняемый файл и проверим его работу (рис. ??).



Запуск файла lab7-2

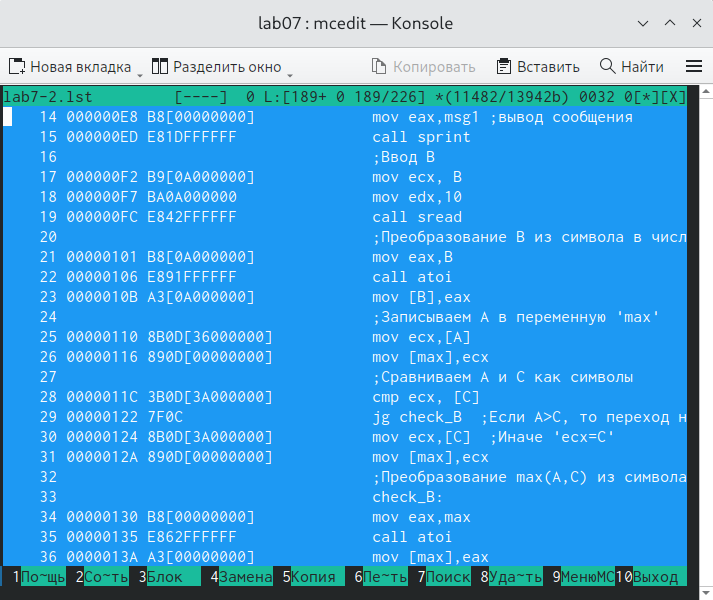
Как мы видим, с при разных введённых В программа выдаёт корректные результаты.

Теперь изучим структуру файла листинга. Создадим файл листинга для программы из файла *lab7-2.asm* и откроем этот файл с помощью редактора *mcedit* (рис. ??).



Создание файла листинга для программы из файла lab7-2.asm

Текст файла листинга представлен следующим образом: (рис. ??).

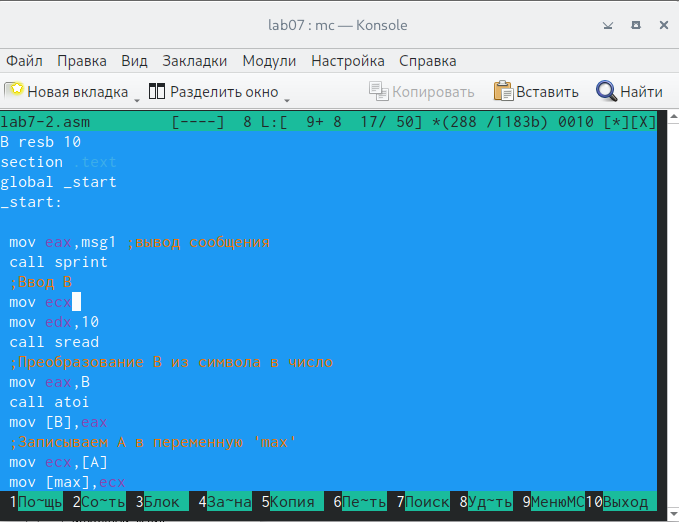


Файл листинга lab7-2.lst

Рассмотрим подробно строки 20, 21 и 22.

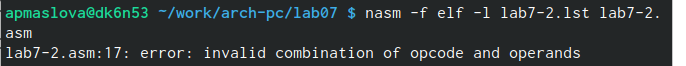
* В строке 20 содержится только комментарий ;Преобразование В из символа в число. Этой строке присвоен определённый номер, однако в ней не генерируется никакой машинный код. Отсутствуют также поля с адресом и исходным текстом программы.
* В строке 21 содержится следующий текст программы: mov eax,B. Адрес 00000101 соответствует смещению машинного кода B8[0a000000] от начала текущего сегмента
* В строке 22 содержится следующий текст программы: call atoi. Адрес 00000106 соответствует смещению машинного кода E891FFFFFF от начала текущего сегмента.

Откроем файл с программой *lab7-2.asm* и в инструкции *mov* удалим один из двух операндов (рис. ??).



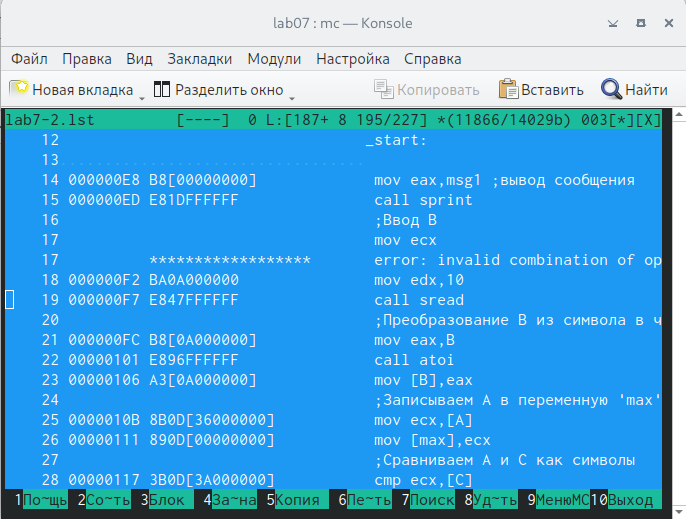
Удаление одного из друх операндов инструкции mov в файле lab7-2.asm

Выполним трансляцию с получением файла листинга (рис. ??).



Трансляция с получением файла листинга

Как мы видим, транслятор обнаружил ошибку при ассемблировании и вывел её на экран. Посмотрим, что произошло с созданным файлом листинга: (рис. ??)



Файл листинга после изменений

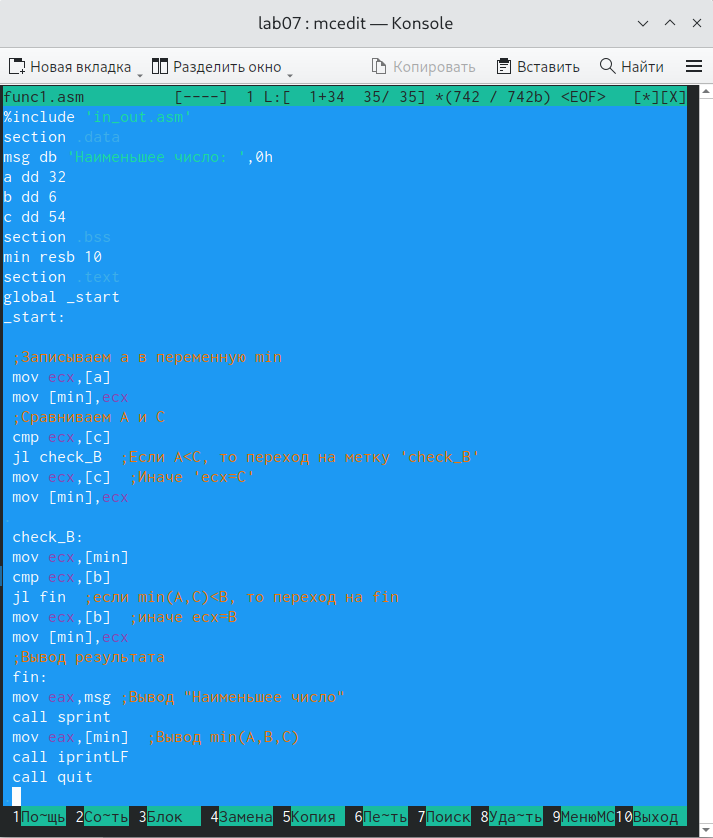
Как мы видим, в файле листинга также сказано, что в файле ошибка.

# 3 Выполнение заданий для самостоятельной работы

Теперь напишем программу нахождения наименьшей из 3 целочисленных переменных *a,b8* и *c*. Значения переменных возьмём в сооответствии с вариантом №15: . Создадим файл *func1.asm* и введём в него текст программы из листинга 7.1 (рис. ??).

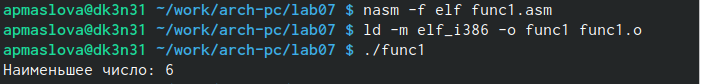
**Листинг 7.1. Программа нахождения наименьшей из 3 целочисленных переменных a,b и c**

%include 'in\_out.asm'  
section .data  
msg db 'Наименьшее число: ',0h  
a dd 32  
b dd 6  
c dd 54  
section .bss  
min resb 10  
section .text  
global \_start  
\_start:  
  
 ;Записываем a в переменную min  
 mov ecx,[a]  
 mov [min],ecx  
 ;Сравниваем А и С  
 cmp ecx,[c]  
 jl check\_B ;Если А<C, то переход на метку 'check\_B'  
 mov ecx,[c] ;Иначе 'ecx=C'  
 mov [min],ecx  
   
 check\_B:  
 mov ecx,[min]  
 cmp ecx,[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 ;Вывод "Наименьшее число"  
 call sprint  
 mov eax,[min] ;Вывод min(A,B,C)  
 call iprintLF  
 call quit



Текст файла func1.asm

Создадим исполняемый файл и запустим программу (рис. ??).



Запуск файла func1

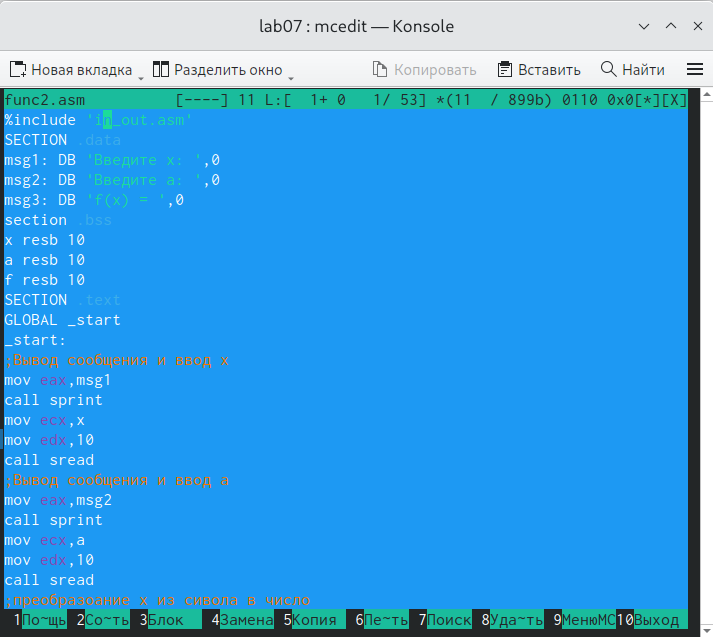
На экран программа вывела наименьшую из трёх переменных, равную 6. Программа работает корректно.

Теперь напишем программу, которая для введенных с клавиатуры значений и вычисляет значение заданной функции и выводит результат вычислений. Вид функции , соответствующий варианту №15, следующий:

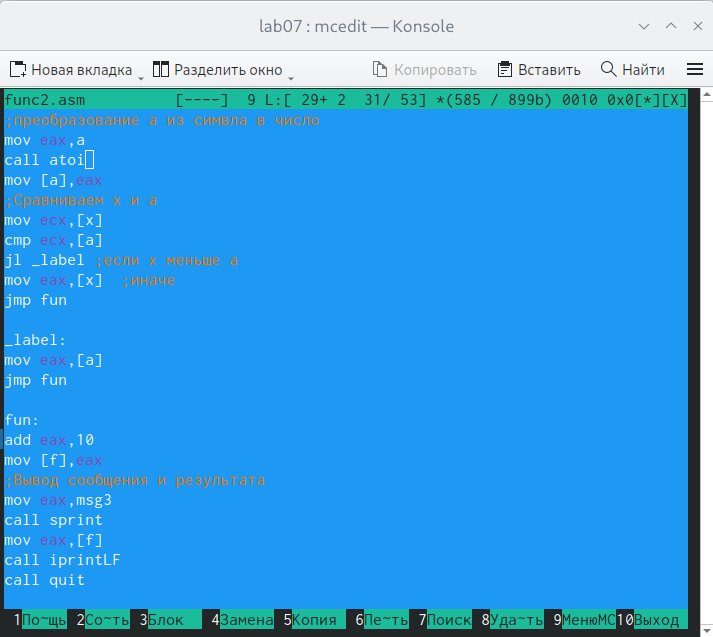
Создадим файл *func2.asm* и введём в него текст программы из листинга 7.2 (рис. ??, рис. ??).

**Листинг 7.2. Программа вычисления значения функции**

%include 'in\_out.asm'  
SECTION .data  
msg1: DB 'Введите x: ',0  
msg2: DB 'Введите a: ',0  
msg3: DB 'f(x) = ',0  
section .bss  
x resb 10  
a resb 10  
f resb 10  
SECTION .text  
GLOBAL \_start  
\_start:  
;Вывод сообщения и ввод х  
mov eax,msg1  
call sprint  
mov ecx,x  
mov edx,10  
call sread  
;Вывод сообщения и ввод а  
mov eax,msg2  
call sprint  
mov ecx,a  
mov edx,10  
call sread  
;преобразоание х из сивола в число  
mov eax,x  
call atoi  
mov [x],eax  
;преобразование а из симвла в число  
mov eax,a  
call atoi  
mov [a],eax  
;Сравниваем х и а  
mov ecx,[x]  
cmp ecx,[a]  
jl \_label ;если х меньше а  
mov eax,[x] ;иначе  
jmp fun  
  
\_label:  
mov eax,[a]  
jmp fun  
  
fun:  
add eax,10  
mov [f],eax  
;Вывод сообщения и результата  
mov eax,msg3  
call sprint  
mov eax,[f]  
call iprintLF  
call quit

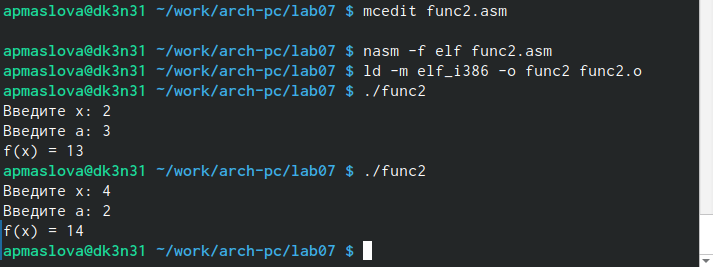


Текст файла func2.asm



Текст файла func2.asm

Создадим исполняемый файл и проверим его работу для заданных значений: $ x\_1 , a\_1 = (2;3), x\_2 , a\_2 = (4;2) $ (рис. ??)



Запуск файла func2

После проверки убедились, что программа работает верно.

# 4 Выводы

Мы познакомились с командами условного и безусловного переходов языка ассемблера NASM и научились писать программы с их использованием. Также изучили назначение и структуру файла листинга.

# Список литературы

1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.
2. GNU Bash Manual. — 2016. — URL: https://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.
6. Robbins A. Bash Pocket Reference. — O’Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: https://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,