

Отчет по лабораторной работе №8

Дисциплина: Архитектура компьютеров

Воронов Александр Валерьевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация циклов в NASM	8
4.2	Обработка аргументов командной строки	12
4.3	Задание для самостоятельной работы	15
5	Выводы	17
6	Список литературы	18

Список иллюстраций

4.1	Создание каталога	8
4.2	Копирование программы из листинга	9
4.3	Запуск программы	9
4.4	Изменение программы	10
4.5	Запуск измененной программы	10
4.6	Добавление push и pop в цикл программы	11
4.7	Запуск измененной программы	11
4.8	Копирование программы из листинга	12
4.9	Запуск второй программы	13
4.10	Копирование программы из третьего листинга	13
4.11	Запуск третьей программы	14
4.12	Изменение третьей программы	14
4.13	Запуск измененной третьей программы	15
4.14	Написание программы для самостоятельной работы	16
4.15	Запуск программы для самостоятельной работы	16

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

1. Реализация циклом в NASM
2. Обработка аргументов командной строки
3. Самостоятельное написание программы по материалам лабораторной работы

3 Теоретическое введение

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды. Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров.

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы №8 (рис. -fig. 4.1).

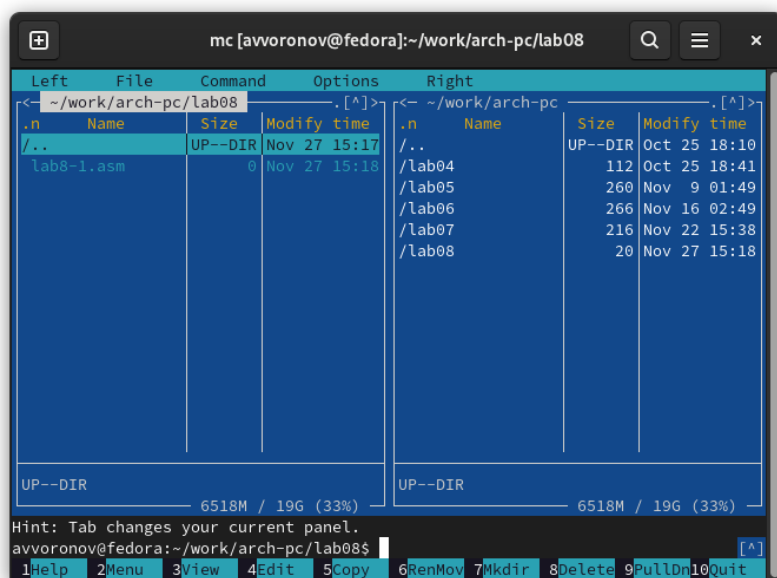
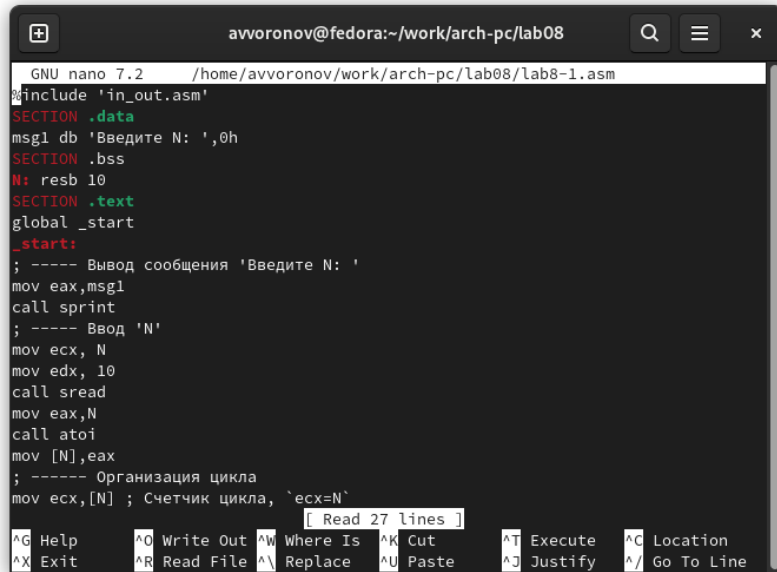


Рис. 4.1: Создание каталога

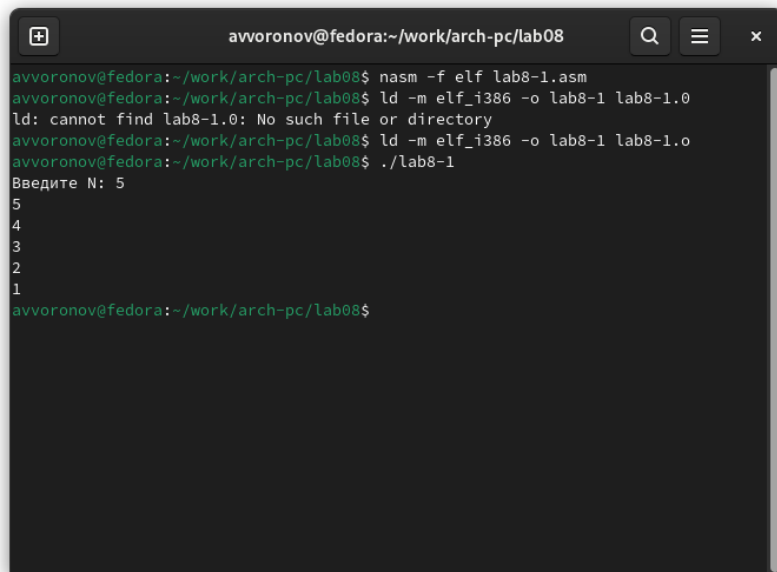
Копирую в созданный файл программу из листинга. (рис. -fig. 4.2).



```
GNU nano 7.2 /home/avvoronov/work/arch-pc/lab08/lab8-1.asm
#include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ---- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ---- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
[ Read 27 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Рис. 4.2: Копирование программы из листинга

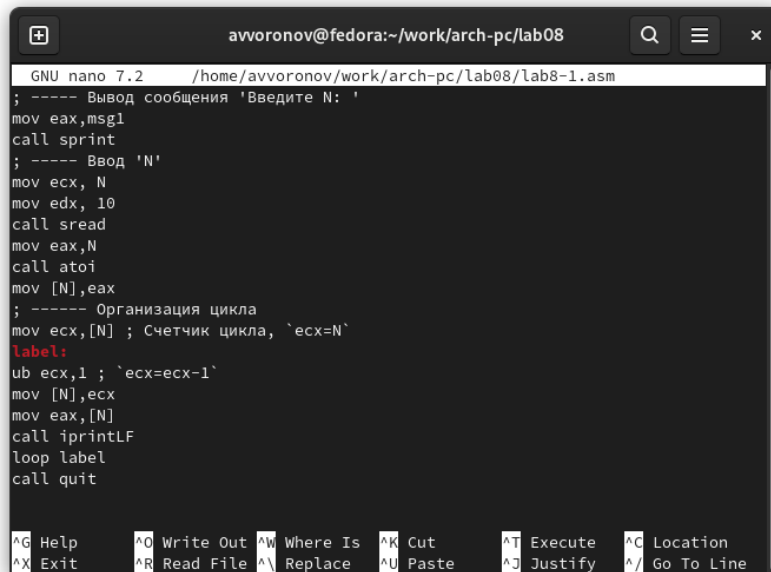
Запускаю программу, она показывает работу циклов в NASM (рис. -fig. 4.3).



```
avvoronov@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
avvoronov@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
ld: cannot find lab8-1.o: No such file or directory
avvoronov@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
avvoronov@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
avvoronov@fedora:~/work/arch-pc/lab08$
```

Рис. 4.3: Запуск программы

Заменяю программу изначальную так, что в теле цикла я изменяю значение регистра ecx (рис. -fig. 4.4).

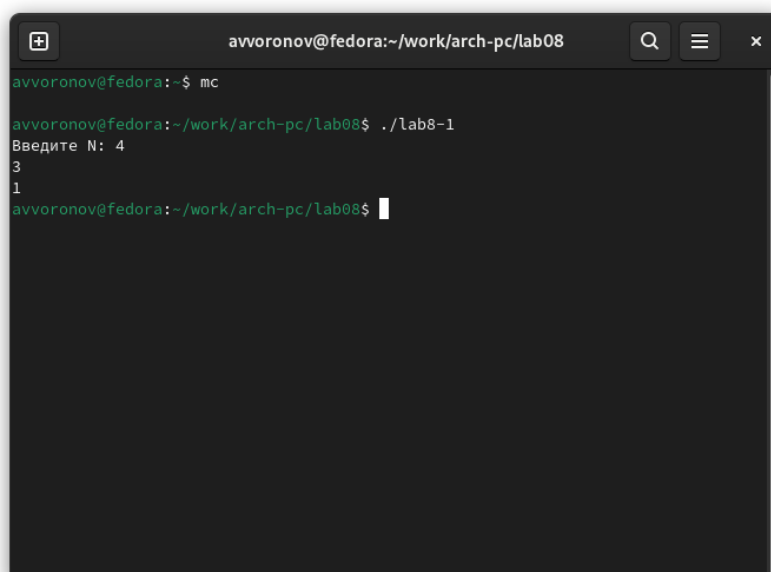


```
GNU nano 7.2 /home/avvoronov/work/arch-pc/lab08/lab8-1.asm
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
ub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintfLF
loop label
call quit

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Рис. 4.4: Изменение программы

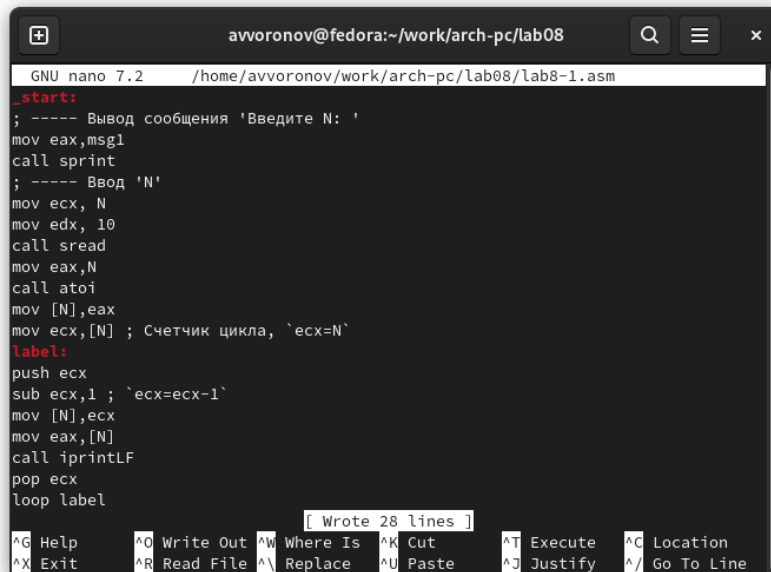
Из-за того, что теперь регистр ecx на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое (рис. -fig. 4.5).



```
avvoronov@fedora:~$ mc
avvoronov@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
avvoronov@fedora:~/work/arch-pc/lab08$
```

Рис. 4.5: Запуск измененной программы

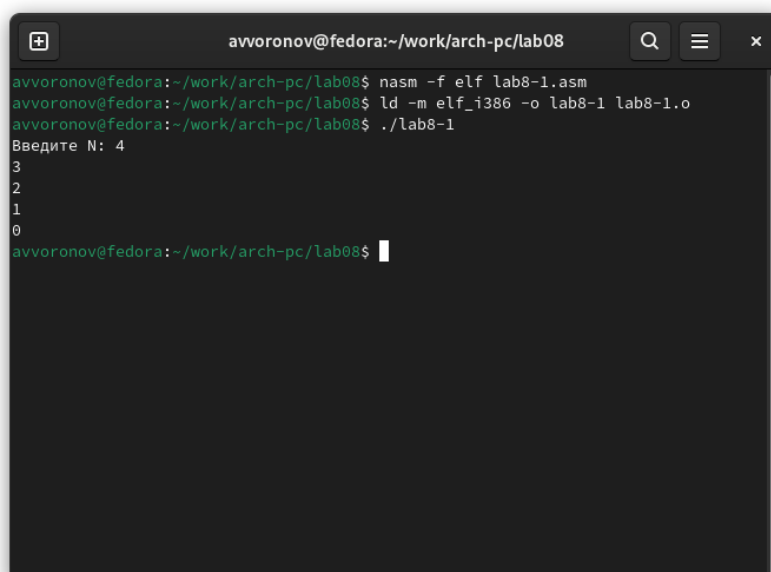
Добавляю команды push и pop в программу (рис. -fig. 4.6).



```
GNU nano 7.2 /home/avvoronov/work/arch-pc/lab08/lab8-1.asm
_start:
; ---- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ---- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
mov eax,N
call atoi
mov [N],eax
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
push ecx
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx
loop label
[Wrote 28 lines]
```

Рис. 4.6: Добавление push и pop в цикл программы

Теперь количество итераций совпадает введенному N, но произошло смещение выводимых чисел на -1 (рис. -fig. 4.7).

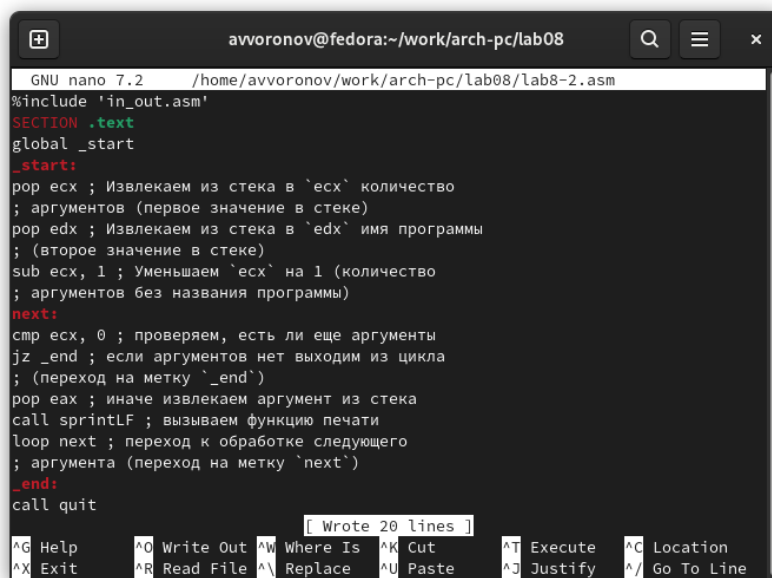


```
avvoronov@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
avvoronov@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
avvoronov@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
2
1
0
avvoronov@fedora:~/work/arch-pc/lab08$
```

Рис. 4.7: Запуск измененной программы

4.2 Обработка аргументов командной строки

Создаю новый файл для программы и копирую в него код из следующего листинга (рис. -fig. 4.8).



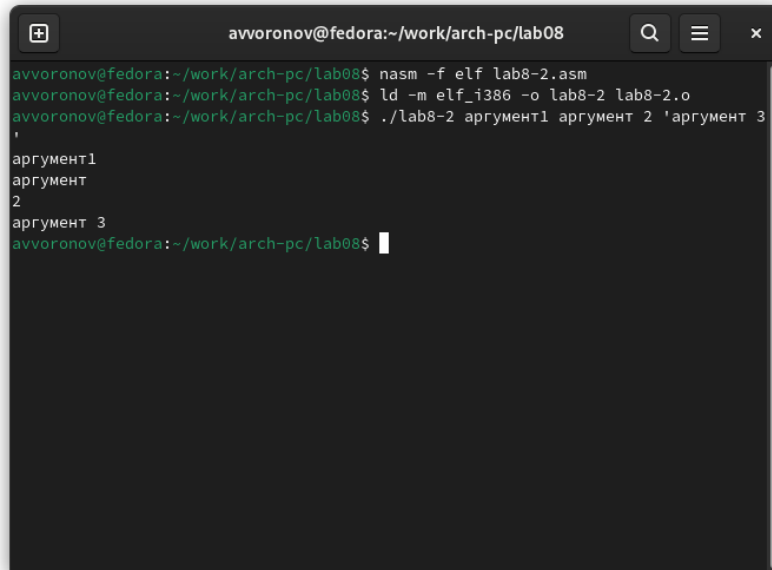
```
GNU nano 7.2 /home/avvoronov/work/arch-pc/lab08/lab8-2.asm
#include 'in_out.asm'
SECTION .text
global _start
_start:
    pop ecx ; Извлекаем из стека в `ecx` количество
             ; аргументов (первое значение в стеке)
    pop edx ; Извлекаем из стека в `edx` имя программы
             ; (второе значение в стеке)
    sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
             ; аргументов без названия программы)
next:
    cmp ecx, 0 ; проверяем, есть ли еще аргументы
    jz _end ; если аргументов нет выходим из цикла
             ; (переход на метку `_end`)
    pop eax ; иначе извлекаем аргумент из стека
    call printf ; вызываем функцию печати
    loop next ; переход к обработке следующего
             ; аргумента (переход на метку `next`)
_end:
    call quit
```

[Wrote 20 lines]

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Location
^X Exit	^R Read File	^_ Replace	^U Paste	^J Justify	^_ Go To Line

Рис. 4.8: Копирование программы из листинга

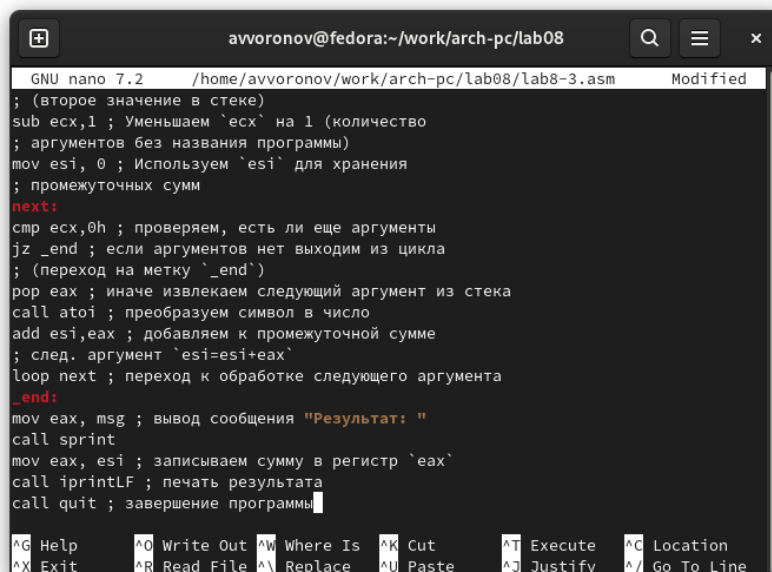
Компилирую программу и запускаю, указав аргументы. Программой было обработано то же количество аргументов, что и было введено (рис. -fig. 4.9).



```
avvoronov@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
avvoronov@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
avvoronov@fedora:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
'
аргумент1
аргумент
2
аргумент 3
avvoronov@fedora:~/work/arch-pc/lab08$
```

Рис. 4.9: Запуск второй программы

Создаю новый файл для программы и копирую в него код из третьего листинга (рис. -fig. 4.10).

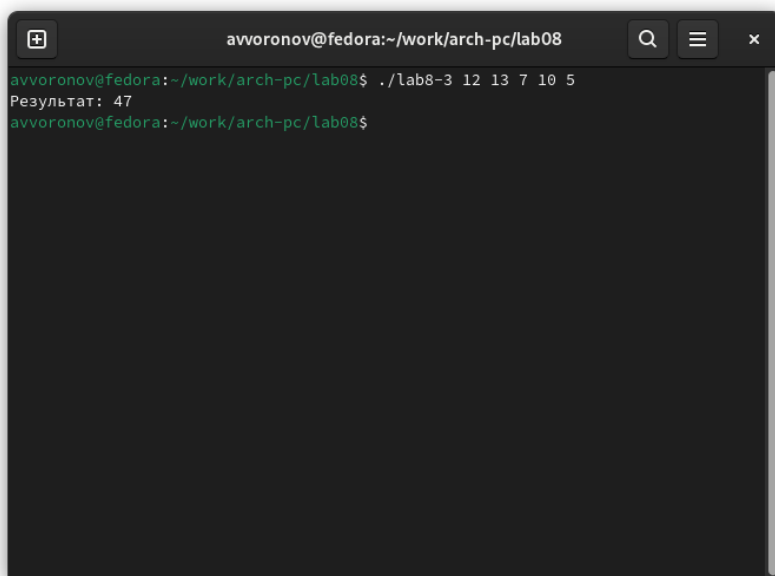


```
GNU nano 7.2 /home/avvoronov/work/arch-pc/lab08/lab8-3.asm Modified
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
mov esi, 0 ; Используем `esi` для хранения
; промежуточных сумм
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
add esi,eax ; добавляем к промежуточной сумме
; след. аргумент `esi=esi+eax`
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, esi ; записываем сумму в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы
```

Рис. 4.10: Копирование программы из третьего листинга

Компилирую программу и запускаю, указав в качестве аргументов некоторые

числа, программа их складывает (рис. -fig. 4.11).

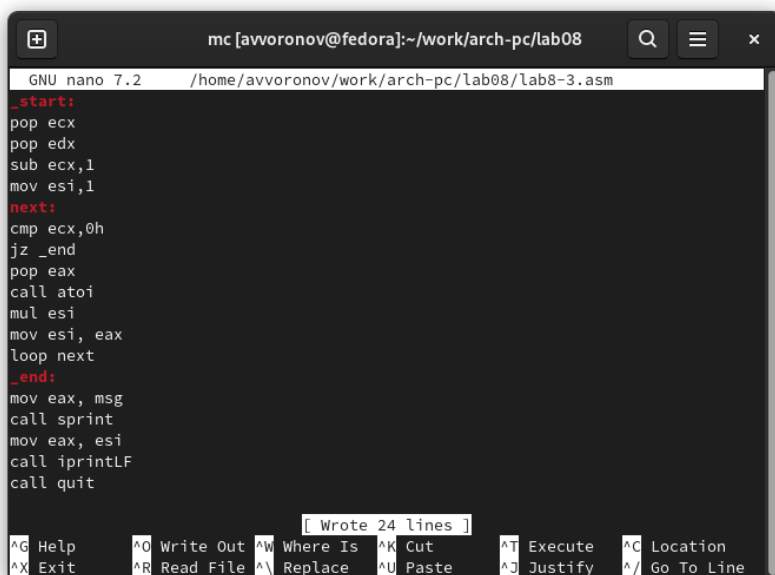


A terminal window titled 'avvoronov@fedora:~/work/arch-pc/lab08'. The prompt is 'avvoronov@fedora:~/work/arch-pc/lab08\$'. The user has entered './lab8-3 12 13 7 10 5'. The output is 'Результат: 47'. The prompt is now 'avvoronov@fedora:~/work/arch-pc/lab08\$'.

```
avvoronov@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
avvoronov@fedora:~/work/arch-pc/lab08$
```

Рис. 4.11: Запуск третьей программы

Изменяю поведение программы так, чтобы указанные аргументы она умножала, а не складывала (рис. -fig. 4.12).



A terminal window titled 'mc [avvoronov@fedora]:~/work/arch-pc/lab08'. The prompt is 'mc [avvoronov@fedora]:~/work/arch-pc/lab08\$'. The user has entered 'nano /home/avvoronov/work/arch-pc/lab08/lab8-3.asm'. The editor shows the following assembly code:

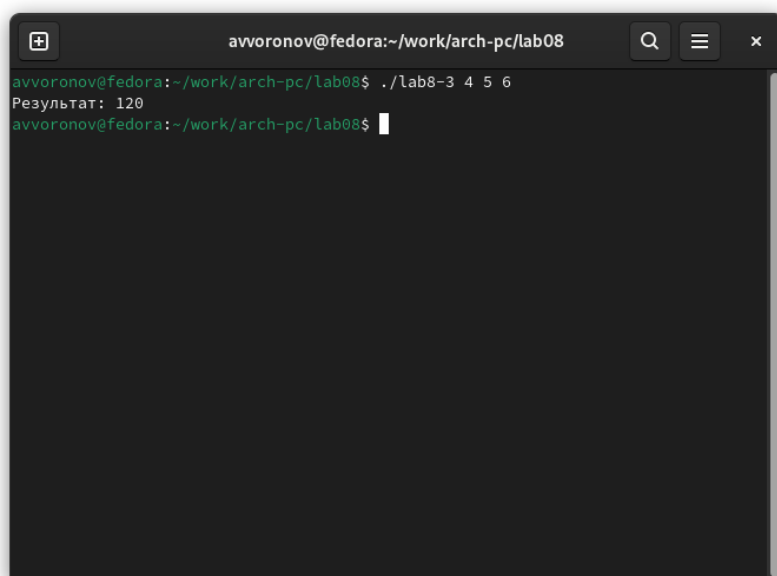
```
_start:
pop ecx
pop edx
sub ecx,1
mov esi,1
next:
cmp ecx,0h
jz _end
pop eax
call atoi
mul esi
mov esi, eax
loop next
_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

The status bar at the bottom indicates '[Wrote 24 lines]'. The bottom of the window shows a menu with the following options:

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Location
^X Exit	^R Read File	^_ Replace	^U Paste	^J Justify	^_ Go To Line

Рис. 4.12: Изменение третьей программы

Программа действительно теперь умножает данные на вход числа (рис. -fig. 4.13).

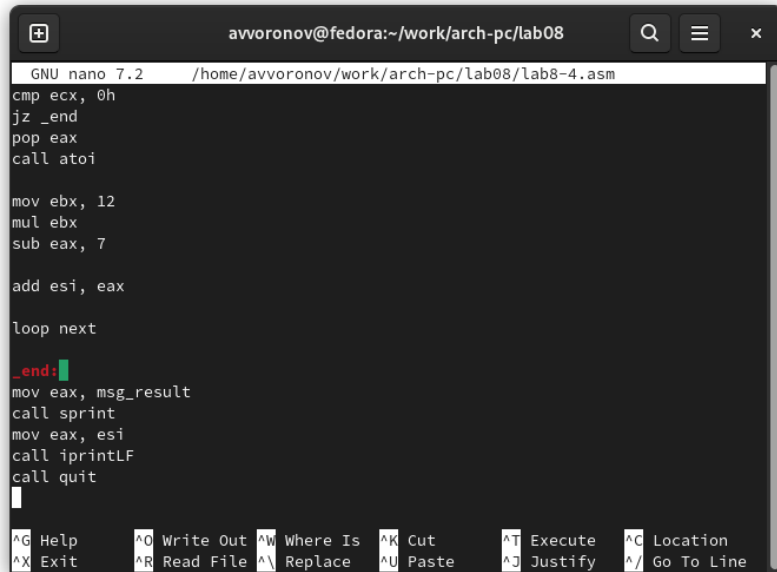
A terminal window with a dark background. The title bar shows 'avvoronov@fedora:~/work/arch-pc/lab08'. The prompt is 'avvoronov@fedora:~/work/arch-pc/lab08\$'. The user has entered './lab8-3 4 5 6'. The output is 'Результат: 120'. The prompt is now 'avvoronov@fedora:~/work/arch-pc/lab08\$' with a cursor.

```
avvoronov@fedora:~/work/arch-pc/lab08$ ./lab8-3 4 5 6
Результат: 120
avvoronov@fedora:~/work/arch-pc/lab08$
```

Рис. 4.13: Запуск измененной третьей программы

4.3 Задание для самостоятельной работы

Пишу программму, которая будет находить сумма значений для функции $f(x) = 12x-7$, которая совпадает с моим девытым вариантом (рис. -fig. 4.14).



```
GNU nano 7.2 /home/avvoronov/work/arch-pc/lab08/lab8-4.asm
cmp ecx, 0h
jz _end
pop eax
call atoi

mov ebx, 12
mul ebx
sub eax, 7

add esi, eax

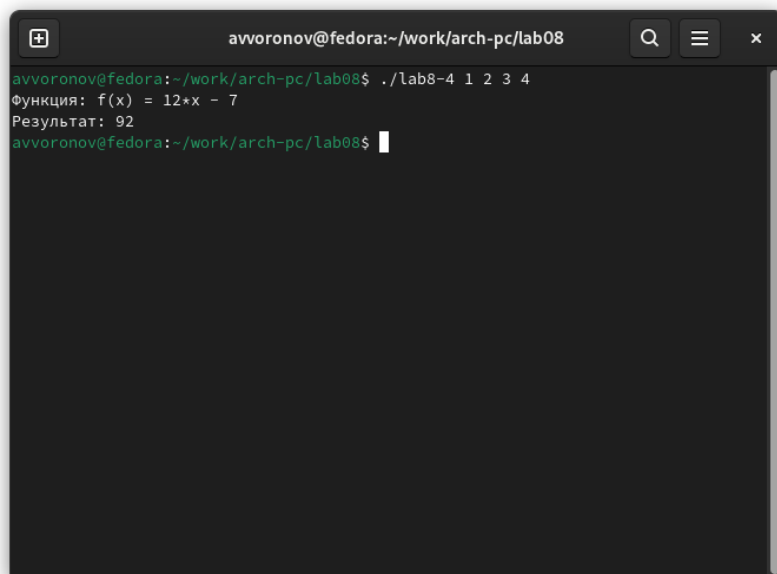
loop next

_end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^_ Go To Line

Рис. 4.14: Написание программы для самостоятельной работы

Проверяю работу программы, указав в качестве аргумента несколько чисел (рис. -fig. 4.15).



```
avvoronov@fedora:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
Функция: f(x) = 12*x - 7
Результат: 92
avvoronov@fedora:~/work/arch-pc/lab08$
```

Рис. 4.15: Запуск программы для самостоятельной работы

5 Выводы

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием циклов а также научился обрабатывать аргументы командной строки.

6 Список литературы

1. Курс на ТУИС
2. Лабораторная работа №8
3. Программирование на языке ассемблера NASM Столяров А. В.