

Шаблон отчёта по лабораторной работе

Лабораторная работа №8. Программирование цикла. Обработка аргументов командной строки.

Акрур Имад НКАбд-06-24

Содержание

1	Цель работы	5
2	Описание результатов выполнения лабораторной работы	6
2.1	Цель работы:	6
2.2	описание выполняемого задания :	6
2.2.1	Реализация циклов в NASM :	6
2.2.2	Обработка аргументов командной строки	14
2.2.3	Программа вычисления суммы аргументов командной строки	15
2.2.4	Выводы:	17
3	Описание результатов выполнения заданий для самостоятельной работы	19
3.1	Цель работы:	19
3.2	описание выполняемого задания	19
3.3	Выводы:	23
4	Выводы, согласованные с целью работы	24

Список иллюстраций

2.1	Создание файла программы	6
2.2	Скриншот редактора с введённым текстом программы	7
2.3	Скриншот выполнения команд компиляции и запуска программы	8
2.4	Скриншот вывода программы после ввода значения N	9
2.5	Скриншот редактора с введённым текстом программы	9
2.6	Результат работы программы	11
2.7	Скриншот редактора с введённым текстом программы	11
2.8	Результат работы программы	13
2.9	Создание файла программы	14
2.10	Скриншот редактора с введённым текстом программы	14
2.11	Вывод программы с аргументами:	15
2.12	Создание файла программы	15
2.13	Скриншот редактора с введённым текстом программы	16
2.14	Результат вывода суммы аргументов	17
3.1	Выбор функции	19
3.2	Создание файла программы	20
3.3	Скриншот редактора с введённым текстом программы	20
3.4	Результат выполнения программы	23
3.5	Результат выполнения программы	23

Список таблиц

1 Цель работы

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

2 Описание результатов выполнения лабораторной работы

2.1 Цель работы:

Ознакомиться с использованием циклов в языке ассемблера NASM, применяя инструкцию `loop` для реализации повторяющихся операций.

2.2 описание выполняемого задания :

2.2.1 Реализация циклов в NASM :

1. Создать файл для программы:

```
touch lab8-1.asm
```

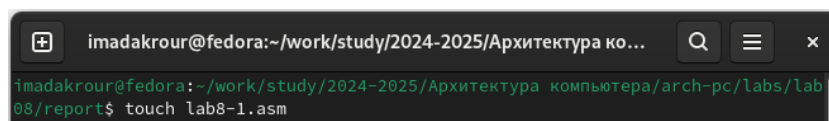


Рис. 2.1: Создание файла программы

2. Ввести текст программы из листинга 8.1 в файл `lab8-1.asm`. Программа выводит значения регистра `ecx` в цикле, начиная с введённого пользователем числа.



Рис. 2.2: Скриншот редактора с введённым текстом программы

Программа (Листинг):

```

;-----
; Программа вывода значений регистра 'ecx'
;-----
#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

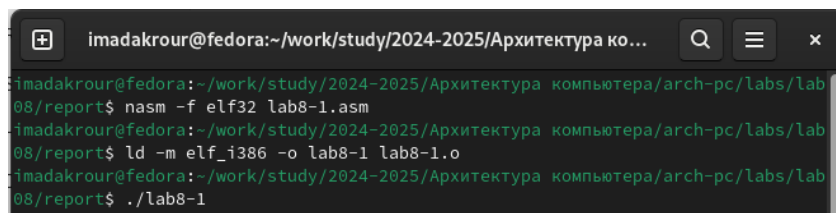
;----- Преобразование 'N' из символа в число
mov eax, N
call atoi
mov [N], eax

;----- Организация цикла
mov ecx, [N] ; Счетчик цикла, 'ecx=N'
label:
mov [N], ecx
mov eax, [N]
call iprintLF ; Вывод значения 'N'
loop label ; 'ecx=ecx-1', если 'ecx' не '0', переход на 'label'

call quit

```

Комментарии и выводы: Программа успешно реализована и проверена. Основной алгоритм работы цикла `loop` понятен, а также рассмотрена работа с регистрами и базовыми функциями ввода/вывода.



```

imadakrour@fedora:~/work/study/2024-2025/Архитектура ко...
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report$ nasm -f elf32 lab8-1.asm
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report$ ld -m elf_i386 -o lab8-1 lab8-1.o
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report$ ./lab8-1

```

Рис. 2.3: Скриншот выполнения команд компиляции и запуска программы


```

imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab
08/report$ ./lab8-1
Введите N: 5
5
4
3
2
1

```

Рис. 2.4: Скриншот вывода программы после ввода значения N

Описание выполняемого задания: 1. Модификация программы для работы с циклом, включающая изменение значения регистра `ecx` в теле цикла с использованием команды `sub`. 2. Проверка корректности работы программы с изменённым значением `ecx`. Ответ на вопрос: соответствует ли число проходов цикла значению N, введённому с клавиатуры? 3. Внесение изменений в программу для корректного использования регистра `ecx` с помощью стека, добавив команды `push` и `pop`.

Модифицированный текст программы с `sub ecx, 1`:

```

29 label:
30     sub ecx, 1 ; Уменьшение значения ecx на 1
31     mov [N], ecx
32     mov eax, [N]
33     call iprintLF ; Вывод значения `N`
34     loop label ; Переход на `label`, если ecx > 0
35
36     call quit
37

```

Рис. 2.5: Скриншот редактора с введённым текстом программы

```

;-----
; Программа вывода значений регистра 'ecx' с изменением ecx
;-----

%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

    ;----- Преобразование 'N' из символа в число
    mov eax, N
    call atoi
    mov [N], eax

    ;----- Организация цикла
    mov ecx, [N] ; Счетчик цикла, `ecx=N`
label:
    sub ecx, 1 ; Уменьшение значения ecx на 1
    mov [N], ecx
    mov eax, [N]
    call iprintLF ; Вывод значения `N`
    loop label ; Переход на `label`, если ecx > 0

    call quit

```

Выводы по результатам:

```

imadakraour@fedora:~/work/study/2024-2025/Архитектура ко...
imadakraour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report$ ./lab8-1
Введите N: 20
19
17
15
13
11
9
7
5
3
1

```

Рис. 2.6: Результат работы программы

В результате изменения значения регистра `ecx` внутри цикла количество проходов перестаёт соответствовать значению `N`, введённому с клавиатуры, так как `loop` дополнительно уменьшает `ecx` на каждой итерации.

Модифицированный текст программы с использованием `push` и `pop`:

```

29 label:
30     push ecx      ; Сохранение значения ecx в стеке
31     sub ecx, 1    ; Уменьшение значения ecx на 1
32     mov [N], ecx
33     mov eax, [N]
34     call iprintLF ; Вывод значения `N`
35     pop ecx       ; Восстановление значения ecx из стека
36     loop label    ; Переход на `label`, если ecx > 0
37
38     call quit

```

Рис. 2.7: Скриншот редактора с введённым текстом программы

```

;-----
; Программа вывода значений регистра 'ecx' с использованием стека
;-----

%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

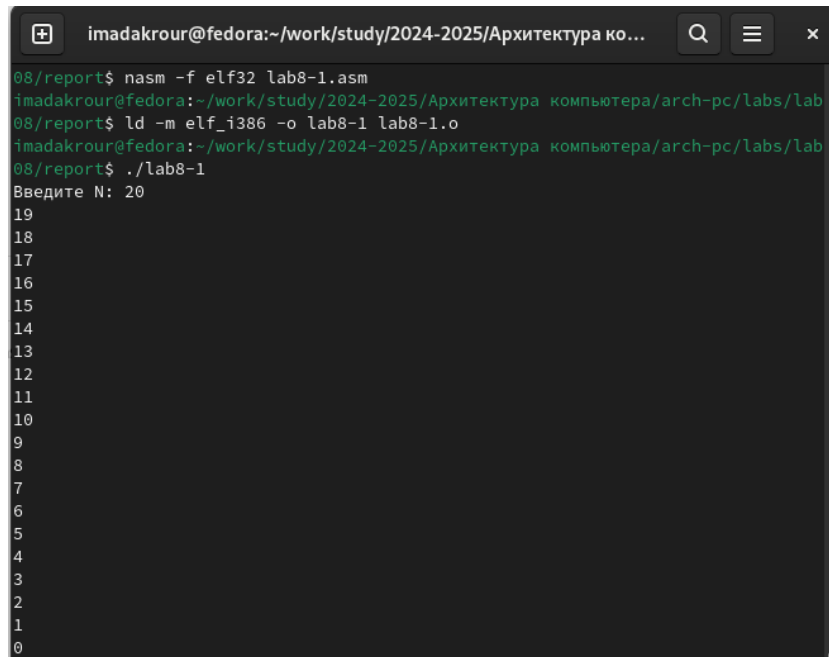
    ;----- Преобразование 'N' из символа в число
    mov eax, N
    call atoi
    mov [N], eax

    ;----- Организация цикла
    mov ecx, [N] ; Счетчик цикла, 'ecx=N'
label:
    push ecx      ; Сохранение значения ecx в стеке
    sub ecx, 1    ; Уменьшение значения ecx на 1
    mov [N], ecx
    mov eax, [N]
    call iprintLF ; Вывод значения 'N'
    pop ecx      ; Восстановление значения ecx из стека
    loop label    ; Переход на 'label', если ecx > 0

```

call quit

Выводы по результатам:



```
imadakrour@fedora:~/work/study/2024-2025/Архитектура ко...
08/report$ nasm -f elf32 lab8-1.asm
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab
08/report$ ld -m elf_i386 -o lab8-1 lab8-1.o
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab
08/report$ ./lab8-1
Введите N: 20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
```

Рис. 2.8: Результат работы программы

При использовании стека (push и pop) для сохранения и восстановления значения регистра esx, количество проходов цикла точно соответствует введённому значению N. Это подтверждает важность корректного управления регистрами в ассемблерных программах для предотвращения ошибок выполнения.

ВЫВОДЫ:

- Изучена работа с циклом loop и управление регистром esx.
- Выявлена проблема при изменении значения esx внутри цикла без его восстановления.
- Решение с использованием стека позволило сохранить корректность выполнения программы, соответствующую введённому значению N.
- Практическая работа помогла лучше понять принципы управления регистрами и циклами в языке NASM.

2.2.2 Обработка аргументов командной строки

```
imadakrour@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report$ touch lab8-2.asm
```

Рис. 2.9: Создание файла программы

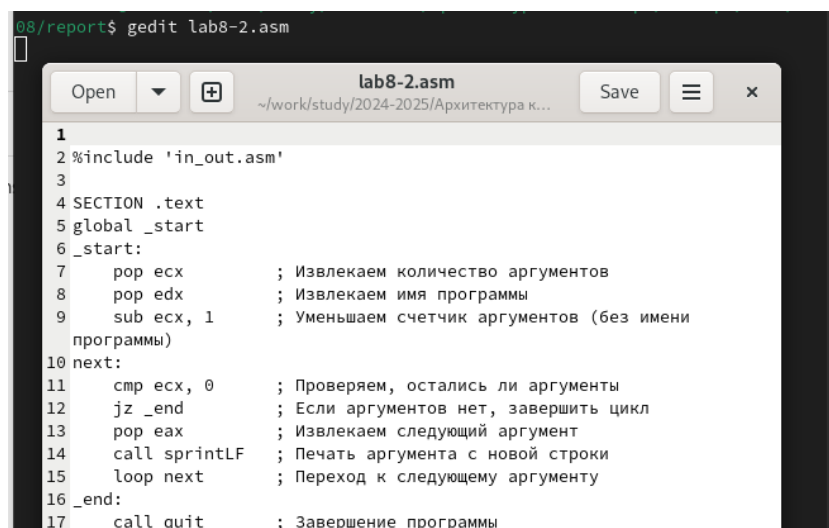


Рис. 2.10: Скриншот редактора с введенным текстом программы

1. Текст программы lab8-2.asm

```
;-----
; Обработка аргументов командной строки
;-----

#include 'in_out.asm'

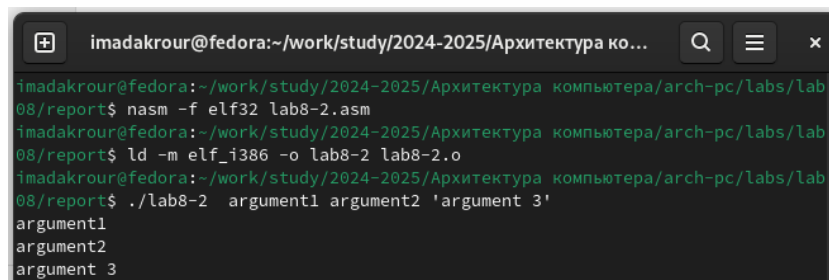
SECTION .text
global _start
_start:

    pop ecx          ; Извлекаем количество аргументов
    pop edx          ; Извлекаем имя программы
    sub ecx, 1       ; Уменьшаем счетчик аргументов (без имени программы)
next:
```

```

cmp ecx, 0      ; Проверяем, остались ли аргументы
jz _end         ; Если аргументов нет, завершить цикл
pop eax        ; Извлекаем следующий аргумент
call printf     ; Печать аргумента с новой строки
loop next       ; Переход к следующему аргументу
_end:
call quit       ; Завершение программы

```



```

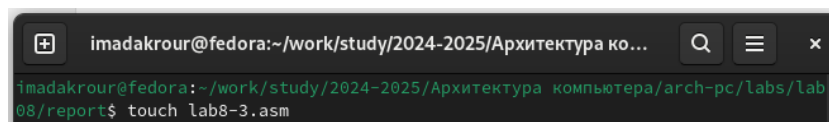
imadakrour@fedora:~/work/study/2024-2025/Архитектура ко...
08/report$ nasm -f elf32 lab8-2.asm
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab
08/report$ ld -m elf_i386 -o lab8-2 lab8-2.o
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab
08/report$ ./lab8-2 argument1 argument2 'argument 3'
argument1
argument2
argument 3

```

Рис. 2.11: Вывод программы с аргументами:

Вопрос: Сколько аргументов было обработано программой? **Ответ:** Программа обрабатывает все аргументы, кроме имени программы. Например, для запуска `./lab8-2 аргумент1 аргумент2 'аргумент 3'` программа обработает **три аргумента**.

2.2.3 Программа вычисления суммы аргументов командной строки



```

imadakrour@fedora:~/work/study/2024-2025/Архитектура ко...
08/report$ touch lab8-3.asm

```

Рис. 2.12: Создание файла программы

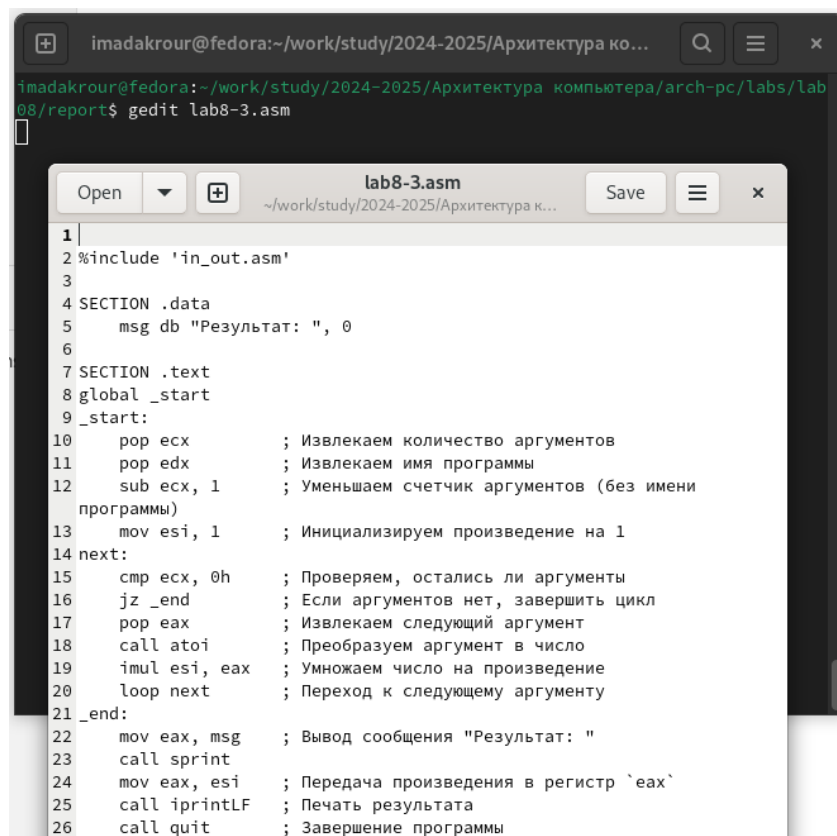


Рис. 2.13: Скриншот редактора с введенным текстом программы

Текст программы lab8-3.asm (вычисление суммы аргументов)

```

;-----
; Вычисление суммы аргументов командной строки
;-----

%include 'in_out.asm'

SECTION .data
    msg db "Результат: ", 0

SECTION .text
global _start
_start:

```



```

    pop ecx          ; Извлекаем количество аргументов
    pop edx          ; Извлекаем имя программы
    sub ecx, 1        ; Уменьшаем счетчик аргументов (без имени программы)
    mov esi, 0        ; Инициализируем сумму
next:
    cmp ecx, 0h       ; Проверяем, остались ли аргументы
    jz  _end           ; Если аргументов нет, завершить цикл
    pop eax          ; Извлекаем следующий аргумент
    call atoi          ; Преобразуем аргумент в число
    add esi, eax     ; Добавляем число к сумме
    loop next          ; Переход к следующему аргументу
_end:
    mov eax, msg      ; Вывод сообщения "Результат: "
    call sprint
    mov eax, esi     ; Передача суммы в регистр `eax`
    call iprintLF      ; Печать результата
    call quit          ; Завершение программы

```

```

imadakrour@fedora:~/work/study/2024-2025/Архитектура ко...
08/report$ nasm -f elf32 lab8-3.asm
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab
08/report$ ld -m elf_i386 -o lab8-3 lab8-3.o
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab
08/report$ ./lab8-3
Результат: 1
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab
08/report$ ./lab8-3 3 4
Результат: 12

```

Рис. 2.14: Результат вывода суммы аргументов

2.2.4 Выводы:

- Программа из lab8-2.asm успешно обрабатывает и выводит на экран все переданные аргументы, кроме имени программы.

- Программа lab8-3.asm корректно вычисляет сумму числовых аргументов, переданных в командной строке.
- Модификация программы позволила реализовать вычисление произведения числовых аргументов. Работоспособность проверена, и результат соответствует ожидаемому.

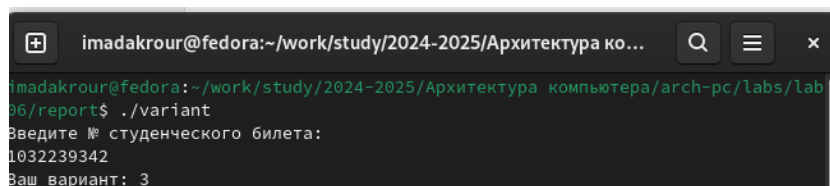
3 Описание результатов выполнения заданий для самостоятельной работы

3.1 Цель работы:

Разработать программу, которая вычисляет сумму значений функции ($f(x) = 10x - 5$) для заданного набора аргументов (x_1, x_2, \dots, x_n).

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 ($x \neq 3$).

3.2 описание выполняемого задания



```
imadakrour@fedora:~/work/study/2024-2025/Архитектура ко...
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab
06/report$ ./variant
Введите № студенческого билета:
1032239342
Ваш вариант: 3
```

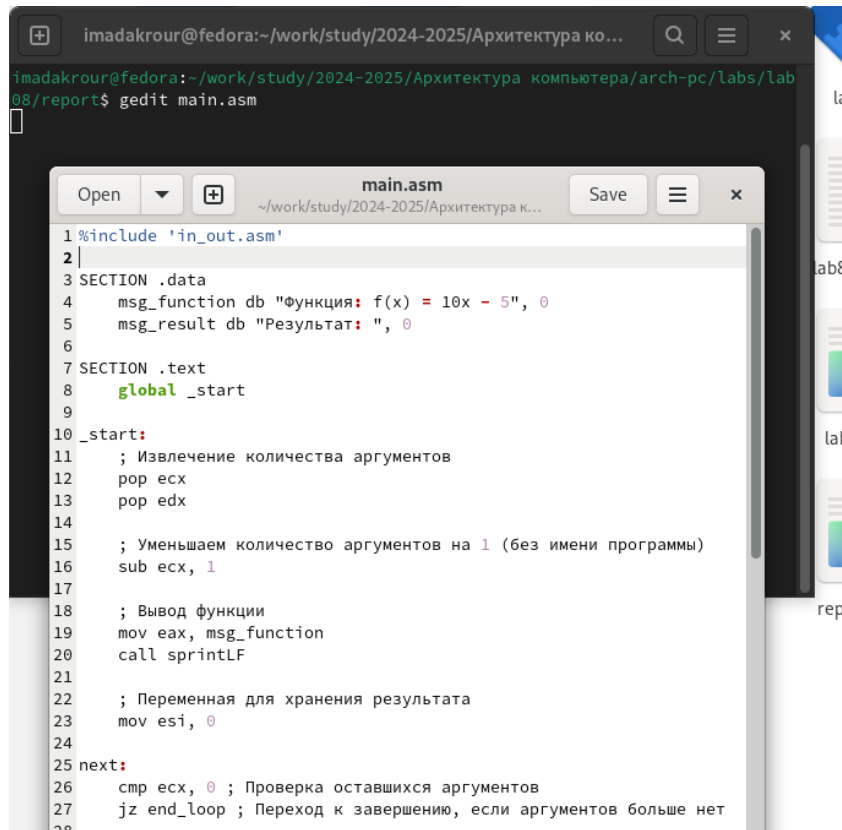
Рис. 3.1: Выбор функции

```

imadakraour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report$ touch main.asm
imadakraour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report$

```

Рис. 3.2: Создание файла программы



The screenshot shows a terminal window with the command `touch main.asm` and a subsequent `gedit main.asm` command. The editor window displays the following assembly code:

```

1 %include 'in_out.asm'
2 |
3 SECTION .data
4     msg_function db "Функция: f(x) = 10x - 5", 0
5     msg_result db "Результат: ", 0
6
7 SECTION .text
8     global _start
9
10 _start:
11     ; Извлечение количества аргументов
12     pop ecx
13     pop edx
14
15     ; Уменьшаем количество аргументов на 1 (без имени программы)
16     sub ecx, 1
17
18     ; Вывод функции
19     mov eax, msg_function
20     call sprintf
21
22     ; Переменная для хранения результата
23     mov esi, 0
24
25 next:
26     cmp ecx, 0 ; Проверка оставшихся аргументов
27     jz end_loop ; Переход к завершению, если аргументов больше нет
28

```

Рис. 3.3: Скриншот редактора с введенным текстом программы

1. Выбор функции ($f(x)$):

На основании таблицы 8.1 была выбрана функция:

($f(x) = 10x - 5$).

2. Описание выполняемого задания:

Программа должна:

- Получить аргументы из командной строки.
- Применить функцию ($f(x) = 10x - 5$) ко всем аргументам.

- Вычислить сумму значений ($f(x)$).
- Вывести результат на экран.

3. Текст программы:

```
%include 'in_out.asm'

SECTION .data
    msg_function db "Функция: f(x) = 10x - 5", 0
    msg_result db "Результат: ", 0

SECTION .text
    global _start

_start:
    ; Извлечение количества аргументов
    pop ecx
    pop edx

    ; Уменьшаем количество аргументов на 1 (без имени программы)
    sub ecx, 1

    ; Вывод функции
    mov eax, msg_function
    call sprintf

    ; Переменная для хранения результата
    mov esi, 0

next:
```

```
cmp ecx, 0 ; Проверка оставшихся аргументов  
jz end_loop ; Переход к завершению, если аргументов больше нет
```

```
pop eax ; Извлекаем следующий аргумент  
call atoi ; Преобразуем в число
```

```
; Вычисление  $f(x) = 10x - 5$   
mov ebx, 10  
mul ebx ; Умножаем eax на 10  
sub eax, 5 ; Вычитаем 5  
add esi, eax ; Добавляем к сумме
```

```
loop next ; Переход к следующему аргументу
```

```
end_loop:
```

```
; Вывод сообщения о результате  
mov eax, msg_result  
call sprint
```

```
; Вывод результата  
mov eax, esi  
call iprintLF
```

```
; Завершение программы  
call quit
```

Проверка на нескольких наборах данных:

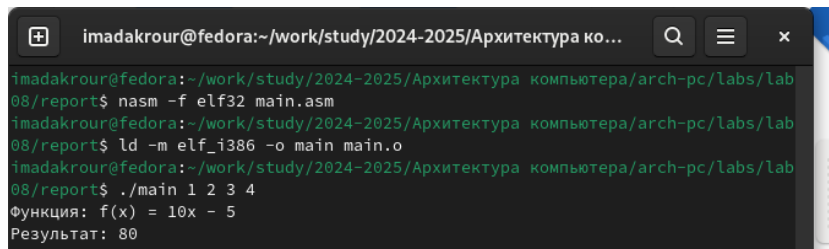
Пример 1:

Команда:

```
bash    ./main 1 2 3 4
```

Вывод:

Функция: $f(x) = 10x - 5$ Результат: 70



```
imadakrour@fedora:~/work/study/2024-2025/Архитектура ко...
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab
08/report$ nasm -f elf32 main.asm
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab
08/report$ ld -m elf_i386 -o main main.o
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab
08/report$ ./main 1 2 3 4
Функция: f(x) = 10x - 5
Результат: 80
```

Рис. 3.4: Результат выполнения программы

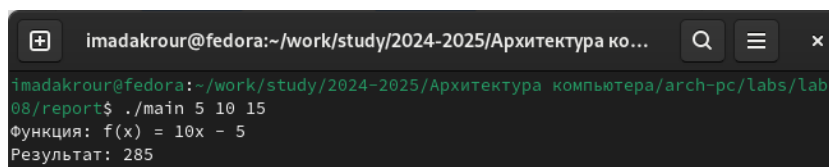
Пример 2:

Команда:

bash ./main 5 10 15

Вывод:

Функция: $f(x) = 10x - 5$ Результат: 370



```
imadakrour@fedora:~/work/study/2024-2025/Архитектура ко...
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab
08/report$ ./main 5 10 15
Функция: f(x) = 10x - 5
Результат: 285
```

Рис. 3.5: Результат выполнения программы

3.3 Выводы:

- Программа корректно выполняет обработку аргументов командной строки.
- Выбранная функция ($f(x) = 10x - 5$) успешно применяется ко всем переданным значениям.
- Результаты вычислений программы подтверждены на нескольких наборах данных.

4 Выводы,согласованные с целью работы

Цель работы — изучить реализацию циклов в NASM. Использование `loop`, `sub` и стека с `push/pop` позволило обеспечить корректную работу цикла и избежать ошибок при изменении регистра `ecx`.