Шаблон отчёта по лабораторной работе

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

Акрур Имад НКАбд-06-24

Содержание

# 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:

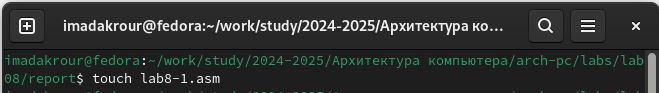
Ознакомиться с использованием циклов в языке ассемблера NASM, применяя инструкцию loop для реализации повторяющихся операций.

## 2.2 описание выполняемого задания :

### 2.2.1 Реализация циклов в NASM :

1. Создать файл для программы:

* touch lab8-1.asm

* 
* Рис. 1: Создание файла программы

1. 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: 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.

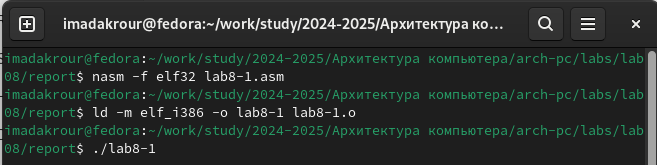


Рис. 3: Скриншот выполнения команд компиляции и запуска программы

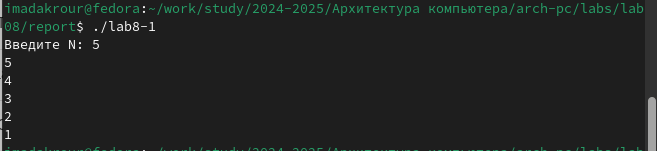


Рис. 4: Скриншот вывода программы после ввода значения N

**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:**

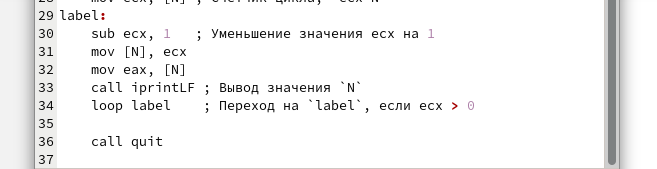


Рис. 5: Скриншот редактора с введённым текстом программы

;----------------------------------------------------------------  
; 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:**

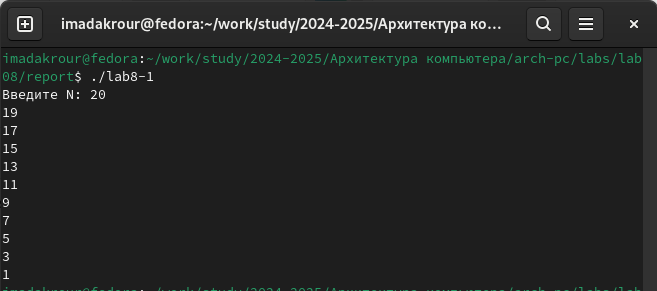


Рис. 6: Результат работы программы

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:**

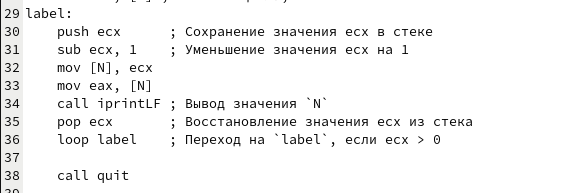


Рис. 7: Скриншот редактора с введённым текстом программы

;----------------------------------------------------------------  
; 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:**

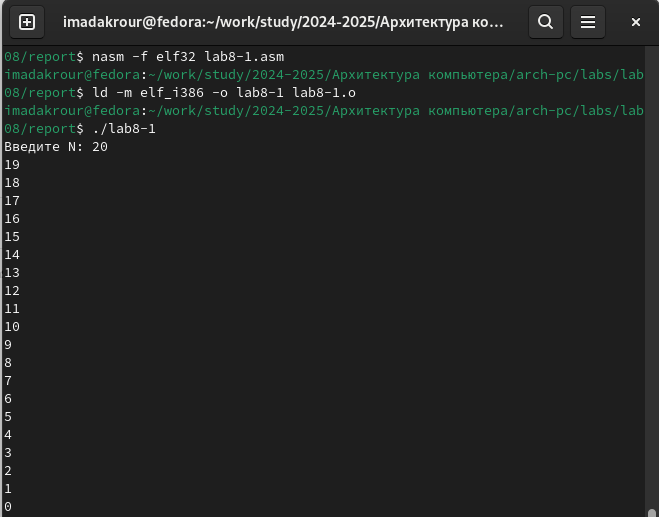


Рис. 8: Результат работы программы

При использовании стека (push и pop) для сохранения и восстановления значения регистра ecx, 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 ecx.
* Выявлена проблема при изменении значения ecx 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

Рис. 9: Создание файла программы

Рис. 9: Создание файла программы

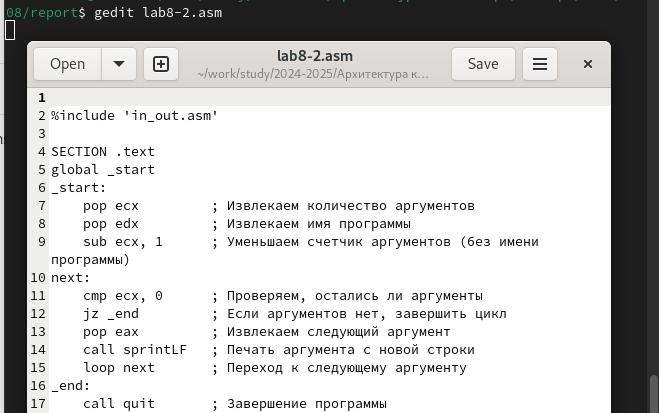


Рис. 10: Скриншот редактора с введённым текстом программы

**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 sprintLF ; Печать аргумента с новой строки  
 loop next ; Переход к следующему аргументу  
\_end:  
 call quit ; Завершение программы

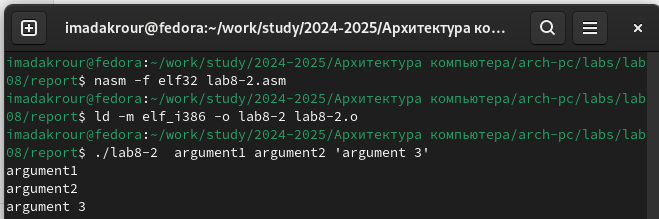


Рис. 11: Вывод программы с аргументами:

**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

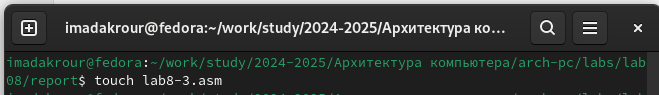


Рис. 12: Создание файла программы

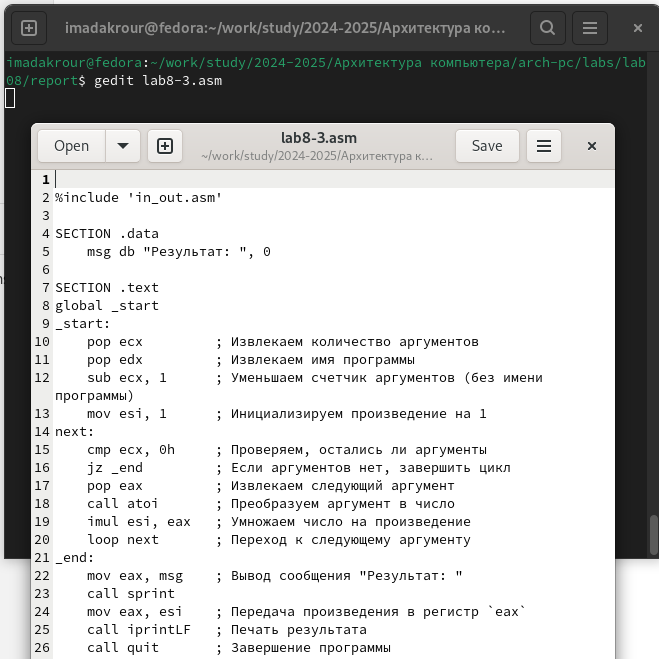


Рис. 13: Скриншот редактора с введённым текстом программы

**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

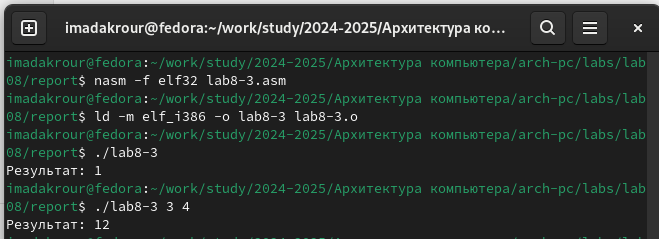


Рис. 14: Результат вывода суммы аргументов

### 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 заданийдля 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, , 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 ).

## 3.2 описание выполняемого задания

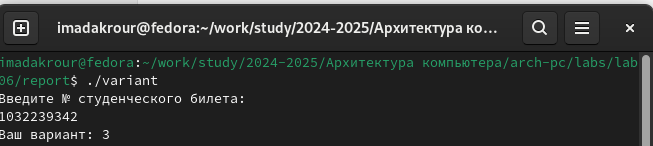


Рис. 15: Выбор функции

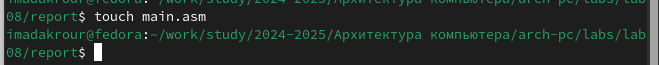


Рис. 16: Создание файла программы

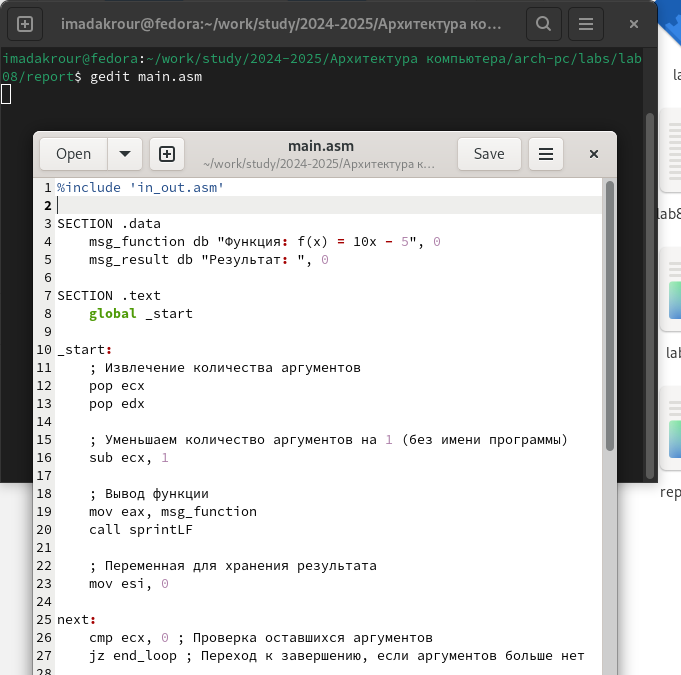


Рис. 17: Скриншот редактора с введённым текстом программы

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 sprintLF  
  
 ; 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

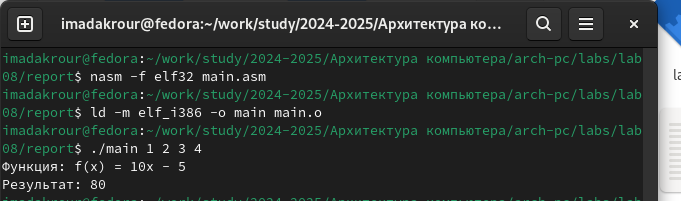


Рис. 18: Результат выполнения программы

**Пример 2:**  
**Команда:**  
bash ./main 5 10 15  
**Вывод:**  
Функция: f(x) = 10x - 5 Результат: 370

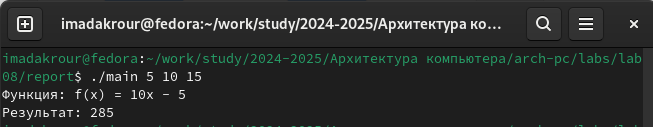


Рис. 19: Результат выполнения программы

## 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.