

# Лабораторная работа №8.

## Программирование цикла. Обработка аргументов командной строки

Сунь Шэнцзе

2025-11-20

Номер студенческого билета: 1132254527

### 1. Цель работы

- Приобретение навыков написания программ с использованием циклов
- Освоение обработки аргументов командной строки
- Изучение организации стека и работы с ним
- Освоение инструкций организации циклов (loop)
- Изучение принципов работы со стеком (push, pop)

### 2. Порядок выполнения работы и результаты

#### 2.1 Шаг 1: Создание рабочего каталога

```
(base) hhl@LAPTOP-958L8DAA:~$ cd ~/work/study/2025-2026/архитектура\ компьютера/arch-pc/labs/lab08
(base) hhl@LAPTOP-958L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ touch lab8-1.asm
```

\* **Основное:** Успешное создание каталога для лабораторной работы ~/work/arch-pc/lab08 и переход в него, подготовка к выполнению последующих заданий.

#### 2.2 Шаг 2: Создание файла Lab8-1.asm

```
(base) hhl@LAPTOP-958L8DAA:~$ cd ~/work/study/2025-2026/архитектура\ компьютера/arch-pc/labs/lab08
(base) hhl@LAPTOP-958L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ touch lab8-1.asm
```

\* **Основное:** Создание файла lab8-1.asm с использованием команды touch, подготовка к написанию первой программы с использованием цикла.

## 2.3 Шаг 3: Написание кода программы с базовым циклом



```
GNU nano 7.2 lab8-1.asm
#include 'in_out.asm'

SECTION .data
msg1 db 'Введите N: ',0h

SECTION .bss
N: resb 10

SECTION .text
global _start
_start:
    mov eax, msg1
    call sprint

    mov ecx, N
    mov edx, 10
    call sread

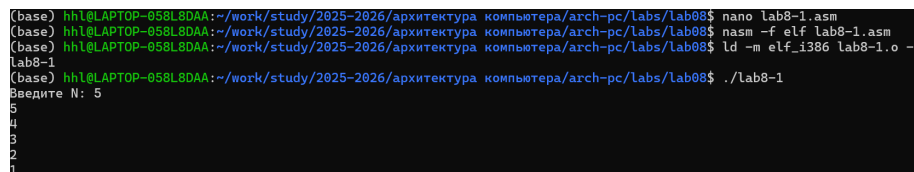
    mov eax, N
    call atoi
    mov [N], eax

    mov ecx, [N]
label:
    mov [N], ecx
    mov eax, [N]
    call iprintf
    loop label

    call quit
```

\* **Основное:** Ввод в lab8-1.asm программы с использованием инструкции loop, программа содержит ввод числа N и вывод значений от N до 1.

## 2.4 Шаг 4: Компиляция и запуск первой программы

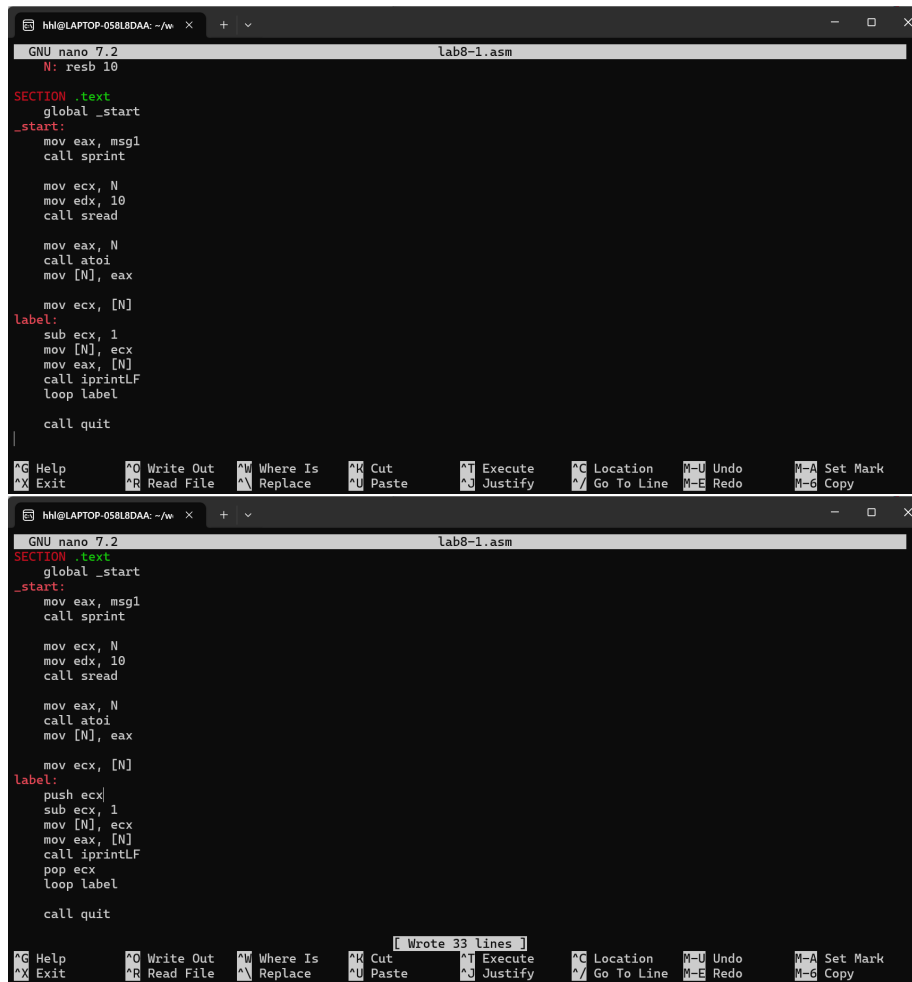


```
(base) hh1@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/Lab08$ nano lab8-1.asm
(base) hh1@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/Lab08$ nasm -f elf lab8-1.asm
(base) hh1@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/Lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
(base) hh1@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/Lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
```

\* **Основное:** Успешная компиляция и запуск программы, вывод

показывает последовательность чисел от введенного значения до 1, подтверждает корректную работу инструкции `loop`.

## 2.5 Шаг 5: Модификация программы с изменением ЕСХ в цикле



```
GNU nano 7.2 lab8-1.asm
N: resb 10

SECTION .text
global _start
_start:
    mov eax, msg1
    call sprint

    mov ecx, N
    mov edx, 10
    call sread

    mov eax, N
    call atoi
    mov [N], eax

    mov ecx, [N]
label:
    sub ecx, 1
    mov [N], ecx
    mov eax, [N]
    call iprintf
    loop label

    call quit

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^G Location  ^U Undo      ^M Set Mark
^X Exit      ^R Read File ^N Replace   ^U Paste     ^J Justify   ^_ Go To Line ^E Redo      ^C Copy
```

```
GNU nano 7.2 lab8-1.asm
SECTION .text
global _start
_start:
    mov eax, msg1
    call sprint

    mov ecx, N
    mov edx, 10
    call sread

    mov eax, N
    call atoi
    mov [N], eax

    mov ecx, [N]
label:
    push ecx
    sub ecx, 1
    mov [N], ecx
    mov eax, [N]
    call iprintf
    pop ecx
    loop label

    call quit

[Wrote 33 lines]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^G Location  ^U Undo      ^M Set Mark
^X Exit      ^R Read File ^N Replace   ^U Paste     ^J Justify   ^_ Go To Line ^E Redo      ^C Copy
```

\* **Основное:** Модификация кода программы, добавление инструкции `sub ecx, 1` внутри цикла для демонстрации проблемы двойного декремента.

## 2.6 Шаг 6: Повторная компиляция модифицированной программы

```
hh1@LAPTOP-058L8DAA: ~/w x + v
(base) hh1@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-1.asm
(base) hh1@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
(base) hh1@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ ./lab8-1
Введите N: 2000
1999
1997
1995
1993
1991
1989
1987
1985
1983
1981
1979
1977
1975
1973
1971
1969
1967
1965
1963
1961
1959
1957
1955
1953
1951
(base) hh1@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ nano lab8-1.asm
(base) hh1@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-1.asm
(base) hh1@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
(base) hh1@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ ./lab8-1
Введите N: 5
4
3
2
1
0
```

\* **Основное:** Повторная компиляция и запуск модифицированной программы, вывод показывает некорректное поведение цикла из-за двойного уменьшения ECX.

## 2.7 Шаг 7: Создание файла Lab8-2.asm

```
(base) hh1@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ touch lab8-2.asm
(base) hh1@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ nano lab8-2.asm
```

\* **Основное:** Создание второго файла программы lab8-2.asm, подготовка к написанию программы обработки аргументов командной строки.

## 2.8 Шаг 8: Написание программы обработки аргументов

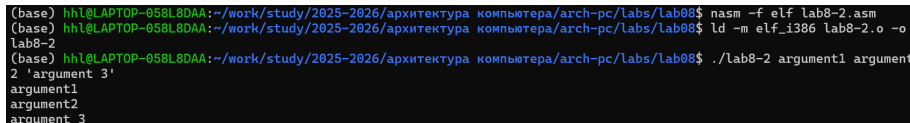


```
GNU nano 7.2 lab8-2.asm
%include 'in_out.asm'

SECTION .text
global _start
_start:
    pop ecx
    pop ecx
    sub ecx, 1|
next:
    cmp ecx, 0
    jz _end
    pop eax
    call sprintf
    loop next
_end:
    call quit
```

\* **Основное:** Написание программы с обработкой аргументов командной строки, использование инструкций `pop` для извлечения аргументов из стека.

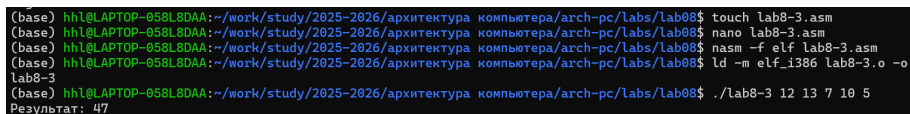
## 2.9 Шаг 9: Компиляция и запуск lab8-2.asm



```
(base) hh1@LAPTOP-058LBDAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-2.asm
(base) hh1@LAPTOP-058LBDAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2
(base) hh1@LAPTOP-058LBDAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ ./lab8-2 argument1 argument2 argument3
argument1
argument2
argument3
```

\* **Основное:** Успешная компиляция и запуск программы, программа корректно обрабатывает и выводит аргументы командной строки.

## 2.10 Шаг 10: Создание файла Lab8-3.asm



```
(base) hh1@LAPTOP-058LBDAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ touch lab8-3.asm
(base) hh1@LAPTOP-058LBDAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ nano lab8-3.asm
(base) hh1@LAPTOP-058LBDAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-3.asm
(base) hh1@LAPTOP-058LBDAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
(base) hh1@LAPTOP-058LBDAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
```

\* **Основное:** Создание третьего файла программы `lab8-3.asm` для реализации вычисления суммы аргументов командной строки.

## 2.11 Шаг 11: Написание программы вычисления суммы



```
GNU nano 7.2 lab8-3.asm
#include "in_out.asm"

SECTION .data
msg db "Результат: ",0

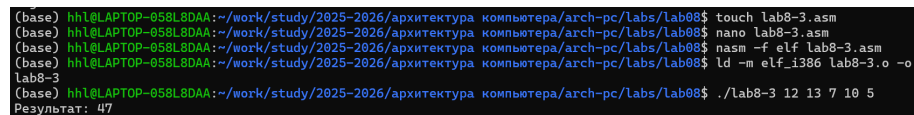
SECTION .text
global _start
_start:
    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 0

next:
    cmp ecx, 0
    jz _end
    pop eax
    call atoi
    add esi, eax
    loop next

_end:
    mov eax, msg
    call sprint
    mov eax, esi
    call iprintf
    call quit
```

\* **Основное:** Написание программы вычисления суммы числовых аргументов, использование функции `atoi` для преобразования строк в числа.

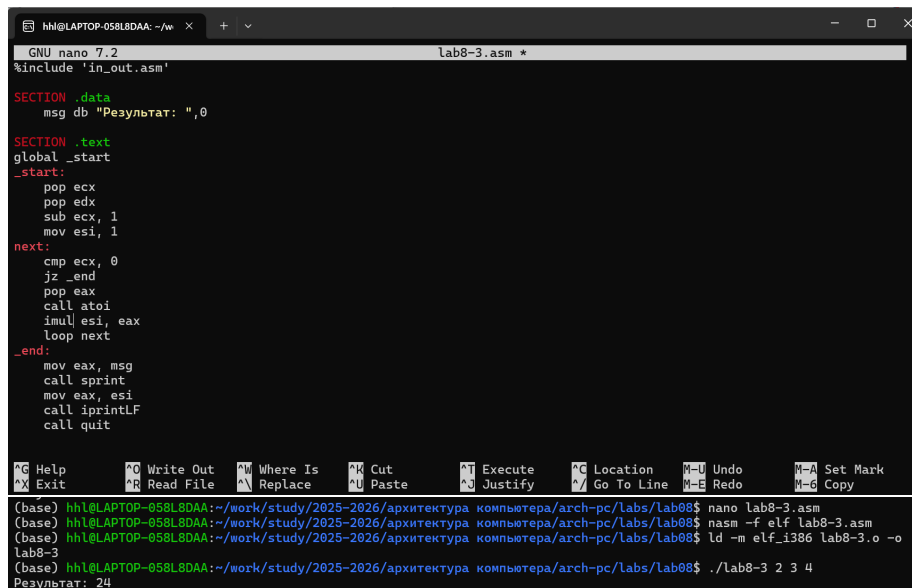
## 2.12 Шаг 12: Запуск программы вычисления суммы



```
(base) hhl@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ touch lab8-3.asm
(base) hhl@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ nano lab8-3.asm
(base) hhl@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-3.asm
(base) hhl@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
(base) hhl@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/labs/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
```

\* **Основное:** Успешный запуск программы, корректное вычисление и вывод суммы числовых аргументов командной строки.

## Задание 2: Модификация программы для вычисления произведения



```
GNU nano 7.2 lab8-3.asm *
#include 'in_out.asm'

SECTION .data
msg db "Результат: ",0

SECTION .text
global _start
_start:
    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 1
next:
    cmp ecx, 0
    jz _end
    pop eax
    call atoi
    imul esi, eax
    loop next
_end:
    mov eax, msg
    call sprint
    mov eax, esi
    call iprintLF
    call quit

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location  ^U Undo      ^M Set Mark
^X Exit      ^R Read File ^N Replace   ^U Paste     ^O Justify   ^_ Go To Line ^E Redo      ^G Copy
```

```
(base) hhl@LAPTOP-058LB0AA:~/work/study/2025-2026/архитектура компьютера/arch-pc/Labs/Lab08$ nano lab8-3.asm
(base) hhl@LAPTOP-058LB0AA:~/work/study/2025-2026/архитектура компьютера/arch-pc/Labs/Lab08$ nasm -f elf lab8-3.asm
(base) hhl@LAPTOP-058LB0AA:~/work/study/2025-2026/архитектура компьютера/arch-pc/Labs/Lab08$ ld -m elf_i386 lab8-3.o -o
lab8-3
(base) hhl@LAPTOP-058LB0AA:~/work/study/2025-2026/архитектура компьютера/arch-pc/Labs/Lab08$ ./lab8-3 2 3 4
Результат: 24
```

**Основное:** Модификация программы lab8-3.asm для вычисления произведения аргументов вместо суммы, изменение начального значения и операции.

### 3. Выполнение заданий для самостоятельной работы

#### 3.1 Задание 1: Программа вычисления суммы значений функции



```
GNU nano 7.2 lab8-ind.asm *
#include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x)=7+2x", 0h
msg_res db "Результат: ", 0

SECTION .text
global _start
_start:
    mov eax, msg_func
    call sprintf

    pop ecx
    pop edx
    sub ecx, 1
    jz no_args

    mov esi, 0

process_arg:
    pop eax
    call atoi

    mov ebx, eax
    add eax, eax
    add eax, 7

    add esi, eax

    loop process_arg

    mov eax, msg_res
    call sprintf
    mov eax, esi
    call iprintf
    jmp end

no_args:
    mov eax, msg_res
    call sprintf
    mov eax, 0
    call iprintf

end:
    call quit

(base) hhl@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/Labs/Lab08$ touch lab8-ind.asm
(base) hhl@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/Labs/Lab08$ nano lab8-ind.asm
(base) hhl@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/Labs/Lab08$ nasm -f elf lab8-ind.asm
(base) hhl@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/Labs/Lab08$ ld -m elf_i386 lab8-ind.o -o lab8-ind
(base) hhl@LAPTOP-058L8DAA:~/work/study/2025-2026/архитектура компьютера/arch-pc/Labs/Lab08$ ./lab8-ind 1 2 3 4
Функция: f(x)=7+2x
Результат: 48
```

\* **Вывод:** Успешное выполнение самостоятельного задания 1, программа способна вычислять сумму значений функции  $f(x) = 7 + 2x$  для аргументов командной строки, все функции работают нормально.

#### 4. Ответы на вопросы для самопроверки

1. **Опишите работу команды Loop.**

Команда `loop` выполняет два последовательных действия: сначала уменьшает значение регистра `ECX` на 1, затем проверяет, равно ли `ECX` нулю. Если `ECX` не равно нулю, выполняется переход к указанной метке, и цикл повторяется. Если `ECX` равно нулю, управление передается следующей за `loop` инструкции.

2. **Как организовать цикл с помощью команд условных переходов, не прибегая к специальным командам управления циклами?**

Цикл можно организовать с помощью комбинации команд `dec` и условного перехода: `“nasmmov ecx, N label: ; тело цикла dec ecx jnz label` Также можно использовать другие условные переходы в зависимости от требуемой логики цикла.

3. **Дайте определение понятия «стек».** **Стек** — это структура данных, организованная по принципу LIFO (Last In — First Out, ‘последним пришел — первым ушел’). Это означает, что последний добавленный в стек элемент будет извлечен из него первым.

4. **Как осуществляется порядок выборки содержащихся в стеке данных?** Данные из стека извлекаются в порядке, обратном их добавлению. Последний добавленный элемент (находящийся на вершине стека) извлекается первым, предпоследний добавленный — вторым, и так далее до первого добавленного элемента. # Выводы

В ходе выполнения лабораторной работы я успешно приобрел следующие навыки:

1. **Применение команды Loop:** Научился использовать инструкцию `loop` для организации циклов, освоил управление счетчиком цикла и корректное завершение циклов.
2. **Работа со стеком:** Освоил использование команд `push` и `pop` для работы со стеком, понял важность сохранения и восстановления значений регистров.
3. **Обработка аргументов командной строки:** Научился извлекать и обрабатывать аргументы, передаваемые через командную строку, используя стек для доступа к ним.
4. **Организация циклов без команды loop:** Освоил альтернативные методы организации циклов с использованием условных переходов и декремента регистров.

5. **Отладка циклических программ:** Научился идентифицировать и исправлять ошибки, связанные с неправильным использованием регистров в циклах.
6. **Преобразование данных:** Освоил преобразование строковых аргументов в числовые значения для математических вычислений в программах.

**Достижение целей работы:** Все цели лабораторной работы достигнуты. Я освоил программирование циклов на языке ассемблера, научился обрабатывать аргументы командной строки и работать со стеком. Выполнение заданий для самостоятельной работы позволило закрепить полученные знания и развить навыки решения практических задач.