

Лабораторная работа № 8

ЧжуЖуйи

30 ноября 2025 г.

Цель работы

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

Порядок выполнения лабораторной работы

1. Реализация циклов в NASM

Создайте каталог для программам лабораторной работы № 8, перейдите в него и создайте файл lab8-1.asm:

```
mkdir ~/work/arch-pc/lab08
cd ~/work/arch-pc/lab08
touch lab8-1.asm
```

```
zhuruiyi@ubuntu:~$ mkdir ~/work/arch-pc/lab08
zhuruiyi@ubuntu:~$ cd ~/work/arch-pc/lab08
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ touch lab8-1.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$
```

Рис. 1: создадим файл

Листинг 8.1. Программа вывода значений регистра ecx

```
;-----
; 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:

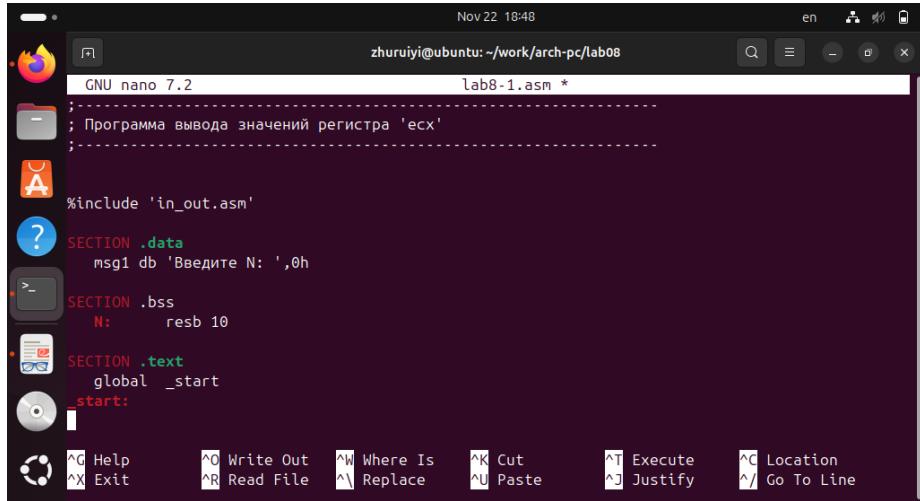
; ----- Вывод сообщения 'Введите N: '
    mov eax,msg1
    call sprint

; ----- 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` и если `ecx` не '0'
                      ; переход на `label`
    call quit

```



The screenshot shows a terminal window titled "GNU nano 7.2" with the command "zhuruiyi@ubuntu: ~/work/arch-pc/lab08". The file "lab8-1.asm" is open. The code is as follows:

```
GNU nano 7.2
; 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:

```

Рис. 2: Введём в файл lab8-1.asm текст программы из листинга 8.1. Создадим исполняемый файл.



```
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 1
1
zhuruiyi@ubuntu:~/work/arch-pc/lab08$
```

Рис. 3: проверим его работу

Данный пример показывает, что использование регистра ecx в теле цикла loop может привести к некорректной работе программы. Измените текст программы добавив изменение значение регистра ecx в цикле:

```
label:
    sub ecx,1      ; `ecx=ecx-1`
    mov [N],ecx
    mov eax,[N]
    call iprintLF

loop label
```

The screenshot shows a terminal window titled "GNU nano 7.2" with the file name "lab8-1.asm". The code is as follows:

```
GNU nano 7.2          zhuruiyi@ubuntu: ~/work/arch-pc/lab08
lab8-1.asm *
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      ; `ecx=ecx-1'
mov [N],ecx
mov eax,[N]
call iprintLF
loop label

^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: Создадим исполняемый файл

The screenshot shows a terminal window displaying the output of the assembly program. The output consists of a series of numbers, each followed by a carriage return and line feed (CRLF). The numbers are: 4294694964, 4294694962, 4294694960, 4294694958, 4294694956, 4294694954, 4294694952, 4294694950, 4294694948, 4294694946, 4294694944, 4294694942, 4294694940, and 4294694938.

Рис. 5: проверим его работу

Ответ на вопрос:

- (1) В данном цикле регистр ecx принимает значения, которые уменьшаются на 2 при каждой итерации. Это происходит потому, что в каждой итерации цикла: · Инструкция sub ecx,1 явно уменьшает ecx на 1 · Инструкция loop неявно уменьшает ecx на 1 · В результате ecx уменьшается на 2 за каждую итерацию
- (2) Нет, число проходов цикла не соответствует значению N, введенному с клавиатуры. Фактическое количество итераций цикла составляет примерно N/2.

Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Внесите изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop:

```
label:  
    push ecx          ; добавление значения ecx в стек  
    sub ecx,1  
    mov [N],ecx  
    mov eax,[N]  
    call iprintLF  
    pop ecx          ; извлечение значения ecx из стека  
  
loop label
```

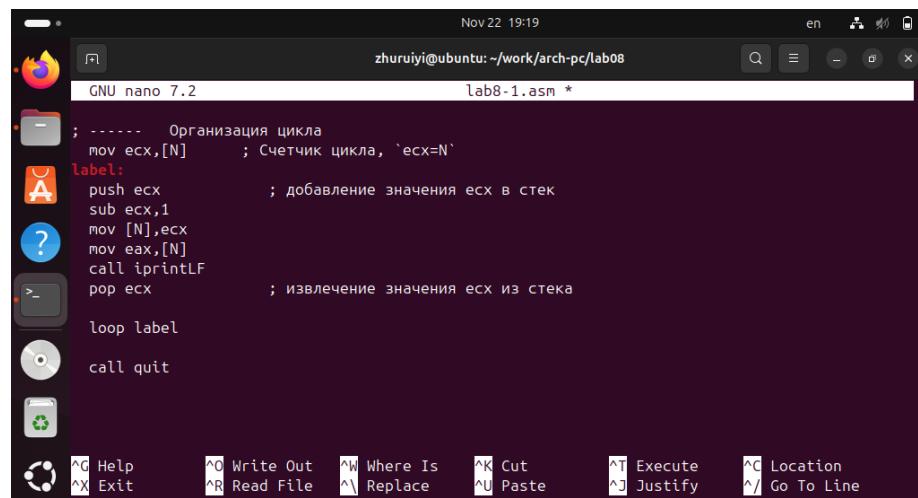


Рис. 6: Создадим исполняемый файл

```
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nano lab8-1.asm  
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm  
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1  
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ./lab8-1  
Введите N: 5  
4  
3  
2  
1  
0  
zhuruiyi@ubuntu:~/work/arch-pc/lab08$
```

Рис. 7: проверим его работу

Ответ на вопрос:

- (1) Да, после внесения изменений с использованием стека число проходов цикла соответствует значению N, введенному с клавиатуры.

2.Обработка аргументов командной строки

Листинг 8.2. Программа выводящая на экран аргументы командной строки

```
;-----  
; Обработка аргументов командной строки  
;-----  
%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 sprintLF    ; вызываем функцию печати  
    loop next         ; переход к обработке следующего  
                      ; аргумента (переход на метку `next`)  
_end:  
    call quit
```

Создайте файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введите в него текст программы из листинга 8.2.

```

GNU nano 7.2
zhuruiyi@ubuntu:~/work/arch-pc/lab08
lab8-2.asm *
; -----
; Обработка аргументов командной строки
;
%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)

```

Рис. 8: Создадим файл

Создайте исполняемый файл и запустите его, указав аргументы:

`./lab8-2 аргумент1 аргумент2 'аргумент3'`

```

zhuruiyi@ubuntu:~/work/arch-pc/lab08$ touch lab8-2.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nano lab8-2.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент2 'аргумент3'
аргумент1
аргумент2
аргумент3
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ 
```

Рис. 9: Результат

Сколько аргументов было обработано программой? Ответ: 3 аргумента.

Листинг 8.3. Программа вычисления суммы аргументов командной строки

```

%include    'in_out.asm'

SECTION .data
msg db "Результат: ",0

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)
    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

```

The screenshot shows a terminal window titled 'GNU nano 7.2' with the file 'lab8-3.asm *'. The code in the editor matches the one provided in the previous block. The terminal window also displays the command '%include "in_out.asm"' at the top.

```

%include "in_out.asm"

SECTION .data
msg db "Результат: ",0

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)
    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. 10: Создадим файл lab8-3.asm в каталоге `~/work/arch-pc/lab08` и введём в него текст программы из листинга 8.3.

Создайте исполняемый файл и запустите его, указав аргументы.
Пример результата работы программы:

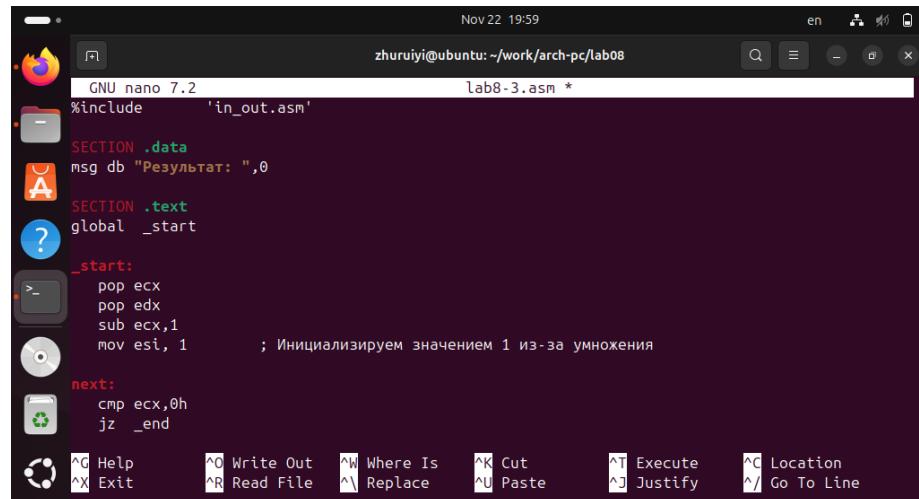
```
./lab8-3 12 13 7 10 5
```

Результат: 47

```
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ touch lab8-3.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nano lab8-3.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
zhuruiyi@ubuntu:~/work/arch-pc/lab08$
```

Рис. 11: Результат

Измените текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.



The screenshot shows a terminal window titled "GNU nano 7.2" with the file "lab8-3.asm" open. The assembly code is as follows:

```
%include 'in_out.asm'

SECTION .data
msg db "Результат: ",0

SECTION .text
global _start

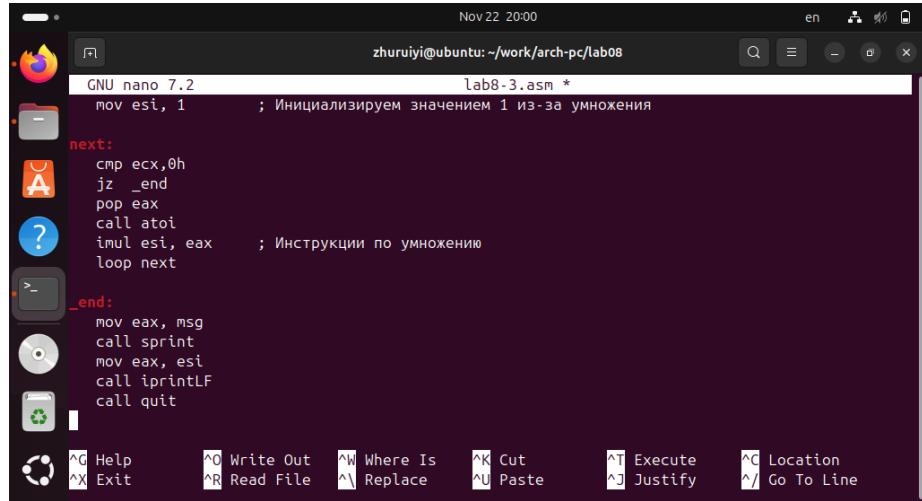
_start:
    pop ecx
    pop edx
    sub ecx,1
    mov esi, 1        ; Инициализируем значением 1 из-за умножения

next:
    cmp ecx,0h
    jz _end

    mov edx,esi
    imul edx,ecx
    mov esi,edx
    add esi,1
    jmp next

_end:
    mov eax,esi
    mov [msg],eax
    mov eax,4
    mov ebx,1
    int 21h
```

Рис. 12: Изменим текст программы



```
GNU nano 7.2          zhuruiyi@ubuntu: ~/work/arch-pc/lab08
mov esi, 1           ; Инициализируем значением 1 из-за умножения
next:
    cmp ecx,0h
    jz _end
    pop eax
    call atoi
    imul esi, eax      ; Инструкции по умножению
    loop next
_end:
    mov eax, msg
    call sprint
    mov eax, esi
    call iprintfLF
    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. 13: Изменим текст программы

```
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nano lab8-3.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 54600
zhuruiyi@ubuntu:~/work/arch-pc/lab08$
```

Рис. 14: Результат

Задание для самостоятельной работы

- Напишите программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения x_i передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$.

The screenshot shows a terminal window titled "GNU nano 7.2" with the file name "lab8-4.asm". The code is as follows:

```
GNU nano 7.2                               lab8-4.asm *
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x)=30x+11",0      ; Номер варианта f(x) 16
msg_result db "Результат: ",0

SECTION .text
global _start

_start:
    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 0

    mov eax, msg_func
    call sprintLF

    mov eax, msg_result
    call sprintLF

    mov eax, 30
    imul eax, ebx
    sub eax, 11

    add esi, eax
    loop next

_end:
    mov eax, msg_result
    call sprintLF
```

Рис. 15: Создадим исполняемый файл

The screenshot shows a terminal window titled "GNU nano 7.2" with the file name "lab8-4.asm". The code is as follows, with comments added:

```
GNU nano 7.2                               lab8-4.asm *
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x)=30x+11",0      ; Номер варианта f(x) 16
msg_result db "Результат: ",0

SECTION .text
global _start

_start:
    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 0

    mov eax, msg_func
    call atoi

    mov ebx, 30      ; ebx = 30
    imul eax, ebx   ; eax = 30 * x
    sub eax, 11      ; eax = 30x - 11

    add esi, eax
    loop next

_end:
    mov eax, msg_result
    call sprintLF
```

Рис. 16: Создадим исполняемый файл

```

Nov 30 10:40
zhuruiyi@ubuntu: ~/work/arch-pc/lab08
GNU nano 7.2          lab8-4.asm *

mov ebx, 30    ; ebx = 30
imul eax, ebx ; eax = 30 * x
sub eax, 11   ; eax = 30x - 11

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. 17: Создадим исполняемый файл

Работы программы для функции $f(x) = 30x - 11$ и набора $x_1 = 1$, $x_2 = 7$, $x_3 = 5$, $x_4 = 7$:

```

./lab8-4 1 7 5 7
Функция: f(x)=30x-11
Результат: 556

```

```

zhuruiyi@ubuntu:~/work/arch-pc/lab08$ touch lab8-4.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nano lab8-4.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-4.o -o lab8-4
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ./lab8-4 1 7 5 7
Функция: f(x)=30x-11
Результат: 556
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ 

```

Рис. 18: Результат

Вывод:

В ходе выполнения лабораторной работы №8 мы глубоко изучили правильные методы управления циклами, работы со стеком и обработки аргументов в ассемблере, в частности, научились избегать распространенных проблем конфликтов регистров.