

# Лаборатору Отчет No7

ДЭВИД МАЙКЛ ФРАНСИС

## Цель работы

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

## Выполнение лабораторной работы

### Реализация циклов в NASM

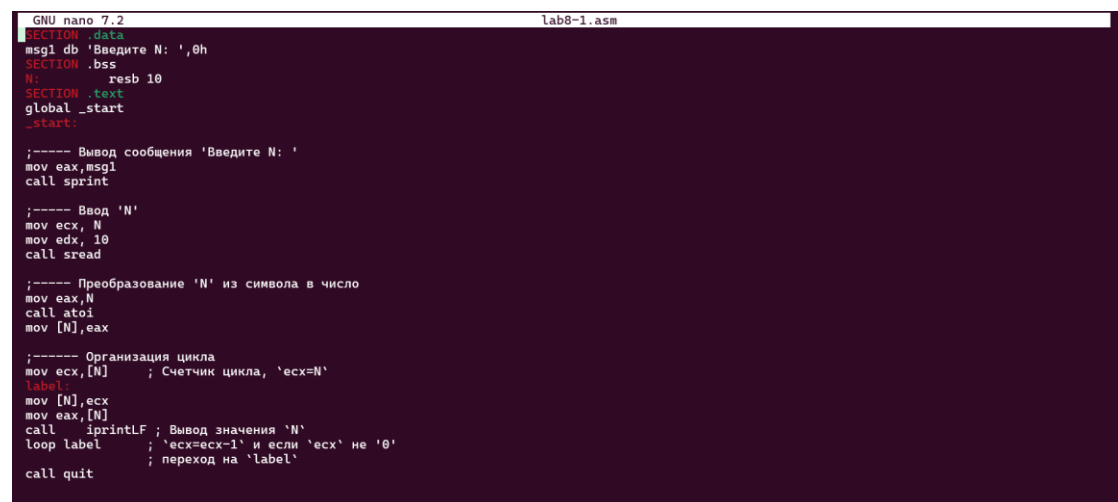
С помощью утилиты mkdir создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №8. Перехожу в созданный каталог с помощью утилиты cd.

С помощью утилиты touch создаю файл lab8-1.asm

```
(base) mike@DESKTOP-I4EKTQ:~$ cd laboratory-work
(base) mike@DESKTOP-I4EKTQ:~/laboratory-work$ cd "Архитектура компьютера/arch-pc/labs"
(base) mike@DESKTOP-I4EKTQ:~/laboratory-work/Архитектура компьютера/arch-pc/labs$ mkdir lab08
(base) mike@DESKTOP-I4EKTQ:~/laboratory-work/Архитектура компьютера/arch-pc/labs$ cd lab08
(base) mike@DESKTOP-I4EKTQ:~/laboratory-work/Архитектура компьютера/arch-pc/labs/lab08$ touch lab8-1.asm
(base) mike@DESKTOP-I4EKTQ:~/laboratory-work/Архитектура компьютера/arch-pc/labs/lab08$ nano lab8-1.asm
```

### Screenshot1

Открываю созданный файл lab8-1.asm, вставляю в него программу вывода значения регистра eax



```
GNU nano 7.2 lab8-1.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
```

### Screenshot2

Создаю исполняемый файл программы и запускаю его. Данный пример показывает, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы.

```
(base) mike@DESKTOP-I4EKT0:~/laboratory-work/Архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-1.asm
m
(base) mike@DESKTOP-I4EKT0:~/laboratory-work/Архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
(base) mike@DESKTOP-I4EKT0:~/laboratory-work/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8-1
Введите N: 2
2
1
```

### Screenshot3

Измените текст программы и внесите изменения в значение регистра `ecx` в цикле

```
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:

;---- 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 ; 'ecx=ecx-1'
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
call quit
```

### Screenshot4

Создаю исполняемый файл программы и запускаю его. Количество проходов цикла не соответствует значению `N`, введенному с клавиатуры

```
(base) mike@DESKTOP-I4EKT0:~/laboratory-work/Архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-1.asm
(base) mike@DESKTOP-I4EKT0:~/laboratory-work/Архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
(base) mike@DESKTOP-I4EKT0:~/laboratory-work/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8-1
Введите N: 2
1
(base) mike@DESKTOP-I4EKT0:~/laboratory-work/Архитектура компьютера/arch-pc/labs/lab08$ 10
10: command not found
(base) mike@DESKTOP-I4EKT0:~/laboratory-work/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
```

### Screenshot5

внес изменения в текст программы, добавив команды `push` и `pop` для сохранения значения счетчика циклов

```

#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        ; 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
call quit

```

## Screenshot6

Создаю исполняемый файл программы и запускаю его.Количество проходов цикла соответствует значению N, введенному с клавиатуры

```

(base) mike@DESKTOP-I4EKT0: ~/Laboratory-work/Архитектура компьютера/arch-pc/labs/lab8$ nasm -f elf lab8-1.asm
(base) mike@DESKTOP-I4EKT0: ~/Laboratory-work/Архитектура компьютера/arch-pc/labs/lab8$ ld -m elf_i386 -o lab8-1 lab8-1.o
(base) mike@DESKTOP-I4EKT0: ~/Laboratory-work/Архитектура компьютера/arch-pc/labs/lab8$ ./lab8-1
Введите N: 2
1
0
(base) mike@DESKTOP-I4EKT0: ~/Laboratory-work/Архитектура компьютера/arch-pc/labs/lab8$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0

```

## Screenshot7

С помощью утилиты touch создаю файл lab8-2.asm. Открываю созданный файл lab8-2.asm, вставляю в него программу вывода значения регистра eax

```

GNU nano 7.2                                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 sprintLF ; вызываем функцию печати
loop next     ; переход к обработке следующего
              ; аргумента (переход на метку 'next')

_end:
call quit

```

## Screenshot8

Создаю исполняемый файл программы и запускаю его. Программа обработала три аргумента

```
(base) mike@DESKTOP-I4EKT0: ~/Laboratory-work/Архитектура компьютера/arch-pc/labs/lab8$ nano lab8-2.asm
(base) mike@DESKTOP-I4EKT0: ~/Laboratory-work/Архитектура компьютера/arch-pc/labs/lab8$ nasm -f elf lab8-2.asm
(base) mike@DESKTOP-I4EKT0: ~/Laboratory-work/Архитектура компьютера/arch-pc/labs/lab8$ ld -m elf_i386 -o lab8-2 lab8-2.o
(base) mike@DESKTOP-I4EKT0: ~/Laboratory-work/Архитектура компьютера/arch-pc/labs/lab8$ ./lab8-2
./lab8-2 аргумент1 аргумент2 'аргумент3'
аргумент1
аргумент2
аргумент3
```

### Screenshot9

С помощью утилиты touch создаю файл lab8-3.asm. Открываю созданный файл lab8-3.asm, вставляю в него программу вывода значения регистра eax

```
GNU nano 7.2 lab8-3.asm
#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 iprintf ; печать результата
call quit    ; завершение программы
```

### Screenshot10

Создаю исполняемый файл программы и запускаю его. я указал аргументы, а программа добавила сумму аргументов, и результат получился равным 47

```
(base) mike@DESKTOP-I4EKT0: ~/Laboratory-work/Архитектура компьютера/arch-pc/labs/lab8$ nasm -f elf lab8-3.asm
(base) mike@DESKTOP-I4EKT0: ~/Laboratory-work/Архитектура компьютера/arch-pc/labs/lab8$ ld -m elf_i386 -o lab8-3 lab8-3.o
(base) mike@DESKTOP-I4EKT0: ~/Laboratory-work/Архитектура компьютера/arch-pc/labs/lab8$ ./lab8-3
Результат: 0
(base) mike@DESKTOP-I4EKT0: ~/Laboratory-work/Архитектура компьютера/arch-pc/labs/lab8$ ./lab8-3 12 13 7 10 5
Результат: 47
```

### Screenshot11

## Выполнение заданий для самостоятельной работы

1. Создаю файл lab8-4.asm с помощью утилиты touch. Открываю созданный файл для редактирования, и написал Написал программу, которая находит сумму значений функции  $f(x)$  для  $2x + 15$  (вариант 1)

```

#include 'in_out.asm'

SECTION .data
msg_function db "Функция: f(x) = 2x + 15", 0
msg db "Результат: ", 0

SECTION .text
global _start

_start:
; Извлекаем из стека количество аргументов
pop ecx          ; Количество аргументов в ecx
pop edx          ; Имя программы (пропускаем)
sub ecx, 1       ; Уменьшаем ecx на 1 (исключаем имя программы)

mov esi, 0       ; Инициализация суммы (0)

next:
cmp ecx, 0       ; Проверяем, есть ли еще аргументы
jz _end          ; Если аргументов нет, выходим из цикла

pop eax          ; Извлекаем следующий аргумент из стека
call atoi        ; Преобразуем символы в число (результат в eax)

; f(x) = 2x + 15
shl eax, 1       ; eax = 2x (умножаем x на 2)
add eax, 15      ; eax = 2x + 15

; Добавляем результат f(x) к промежуточной сумме
add esi, eax     ; esi = esi + f(x)

; ...

_end:
; Выводим результат
mov eax, msg     ; Адрес сообщения "Результат: "
call sprint      ; Выводим сообщение

mov eax, esi     ; Загружаем итоговую сумму в eax
call iprintfLF   ; Печатаем результат с переводом строки

call quit       ; Завершаем программу

```

Создаю и запускаю исполняемый файл. Я использовал 1 2 3 4 в качестве x и получил 80

```

(base) mike@DESKTOP-I4EKT0:~/Laboratory-work/Архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-4.asm
(base) mike@DESKTOP-I4EKT0:~/Laboratory-work/Архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
(base) mike@DESKTOP-I4EKT0:~/Laboratory-work/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8-4 1 2 3 4
Результат: 80

```

*Screenshot14*

## Выводы

Выполняя эту работу, я научился писать программы, используя циклы и обрабатывая аргументы командной строки

Ссылка на официальный сайт [Github](#)