

Отчёт по лабораторной работе №8

дисциплина: Архитектура компьютера

Комаров Владимир Артемович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация циклов в NASM	8
4.2	Обработка аргументов командной строки	10
5	Задания для самостоятельной работы	13
6	Выводы	16

Список иллюстраций

4.1	Переход в каталог и создание файла	8
4.2	Программа вывода значений регистра есх	8
4.3	Исполнение программы из листинга 8.1	9
4.4	Исправленный текст программы lab8-1.asm	9
4.5	Исполнение программы lab8-1	9
4.6	Исправленный текст программы lab8-1.asm	10
4.7	Исполнение программы lab8-1.asm	10
4.8	Текст программы из листинга 8.2	10
4.9	Исполнение программы	10
4.10	Текст программы из листинга 8.3	11
4.11	Исполнение программы	11
4.12	Измененный текст программы из листинга 8.3	12
4.13	Исполнение программы	12
5.1	Текст программы lab8-4.asm	13
5.2	Запуск программы	13

Список таблиц

1 *Цель работы*

Целью работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки в NASM.

2 Задание

1. . 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 $x = x_1, x_2, \dots, x_n$.

3 Теоретическое введение

4 Выполнение лабораторной работы

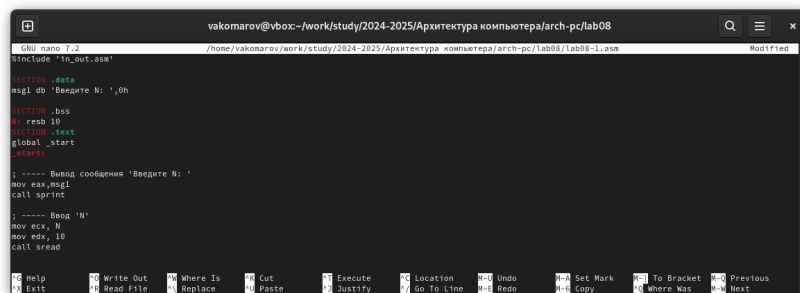
4.1 Реализация циклов в NASM

1. Создаю каталог для программ лабораторной работы №8, перехожу в него и создаю файл lab8-1.asm.

```
vakomarov@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc$ mkdir lab08
vakomarov@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc$ cd lab08
vakomarov@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ touch lab08-1.asm
```

Рис. 4.1: Переход в каталог и создание файла

2. Ввожу в файл lab8-1.asm текст программы из листинга 8.1. Запускаю исполняемый файл.



```
GNU nano 2.2 /home/vakomarov/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08/lab08-1.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 printf

; ----- Ввод "N"
mov ecx, N
mov edx, 10
call $read
```

Рис. 4.2: Программа вывода значений регистра ecx


```
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab08-1.asm
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 lab08-1.o -o lab08-1
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab08-1
Введите N: 45
45
44
43
42
41
40
39
38
37
36
35
34
```

Рис. 4.3: Исполнение программы из листинга 8.1

3. Изменим текст программы, добавив изменение значение регистра `ecx` в цикле. Запустим исправленную программу. Число проходов цикла не соответствует значению, введенному с клавиатуры.

```
label:
sub ecx, 1
mov [N], ecx
mov eax, [N]
call printfLF ; Вывод значения 'N'
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'

; переход на 'label'
call quit
```

Рис. 4.4: Исправленный текст программы `lab8-1.asm`

```
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab08-1.asm
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 lab08-1.o -o lab08-1
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab08-1
Введите N: 4
3
1
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$
```

Рис. 4.5: Исполнение программы `lab8-1`

4. Внесем изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop`. Запустим программу и проверим ее работу. Теперь число проходов цикла соответствует числу, введенному с клавиатуры.

```

label:
push ecx
sub ecx, 1
mov [N], ecx
mov eax, [N]
call iprintfLF ; Вывод значения 'N'
pop ecx

loop label ; 'ecx=ecx-1' и если 'ecx' не '0'

; переход на 'label'
call quit

```

Рис. 4.6: Исправленный текст программы lab8-1.asm

```

vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab08-1.asm
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 lab08-1.o -o lab08-1
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab08-1
Введите N: 5
4
3
2
1
0

```

Рис. 4.7: Исполнение программы lab8-1.asm

4.2 Обработка аргументов командной строки

5. Создаем файл lab8-2.asm. Вводим в него программу из листинга 8.2. Программа обработала 4 аргумента.

```

GNU nano 7.2 /home/vakomarov/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08
#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 sprintfLF ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку 'next')
_end:
call quit

```

Рис. 4.8: Текст программы из листинга 8.2

```

vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab08-2.asm
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 lab08-2.o -o lab08-2
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab08-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
2
аргумент 3

```

Рис. 4.9: Исполнение программы

6. Создадим файл lab8-3.asm и введем в него текст программы из листинга 8.3.

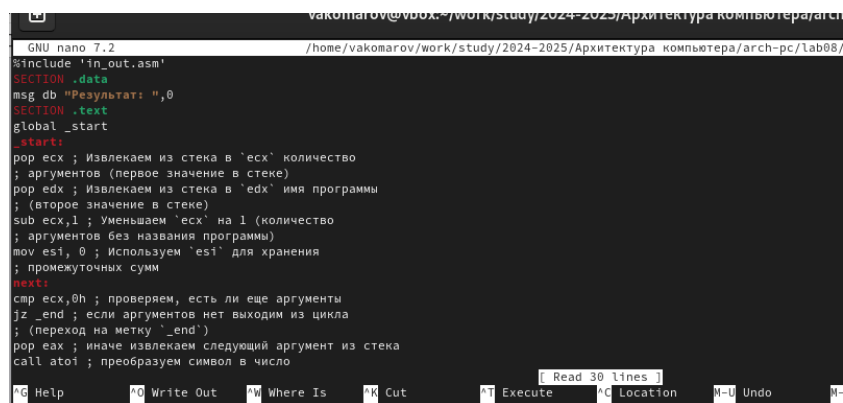
```
GNU nano 7.2 /home/vakomarov/work/st
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
Демидова А. В. 91
Архитектура ЭВМ
    pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
    pop edx ; Извлекаем из стека в `edx` 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`)
```

Рис. 4.10: Текст программы из листинга 8.3

```
vakomarov@box:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab08-3.asm
vakomarov@box:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 lab08-3.o -o lab08-3
vakomarov@box:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab08-3 12 13 7 10 5
Результат: 47
```

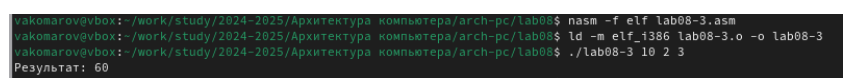
Рис. 4.11: Исполнение программы

7. Изменяю текст программы для вычисления произведения аргументов командной строки.



```
GNU nano 7.2 /home/vakomarov/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08/
#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
; Read 30 lines
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location ^M-U Undo ^M-
```

Рис. 4.12: Измененный текст программы из листинга 8.3



```
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab08-3.asm
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 lab08-3.o -o lab08-3
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab08-3 10 2 3
Результат: 60
```

Рис. 4.13: Исполнение программы

5 Задания для самостоятельной работы

1. Напишем программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$.
Мой вариант - 18. Создадим исполняемый файл и проверим его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$. Программа работает корректно.



```
GNU nano 7.2 /home/vakomarov/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08/
#include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 17 + 5 * x", 0
msg_result db "Результат: ", 0

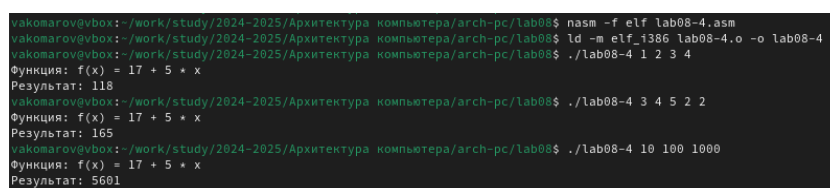
SECTION .text
GLOBAL _start

_start:
    mov eax, msg_func
    call sprintf

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

next:
    cmp ecx, 0h
```

Рис. 5.1: Текст программы lab8-4.asm



```
vakomarov@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab08-4.asm
vakomarov@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 lab08-4.o -o lab08-4
vakomarov@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab08-4 1 2 3 4
Функция: f(x) = 17 + 5 * x
Результат: 118
vakomarov@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab08-4 3 4 5 2 2
Функция: f(x) = 17 + 5 * x
Результат: 165
vakomarov@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab08-4 10 100 1000
Функция: f(x) = 17 + 5 * x
Результат: 5601
```

Рис. 5.2: Запуск программы

Код из файла lab08-4.asm

```
%include 'in_out.asm'
```

```
SECTION .data
msg_func db "Функция:  $f(x) = 17 + 5 * x$ ", 0
msg_result db "Результат: ", 0
```

```
SECTION .text
GLOBAL _start
```

```
_start:
    mov eax, msg_func
    call sprintf

    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 0
```

```
next:
    cmp ecx, 0h
    jz _end
    pop eax
    call atoi

    mov ebx, 5
    mul ebx
    add eax, 17
    add esi, eax
    loop next
```

```
_end:
```

```
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit
```

6 Выводы

В результате выполнения лабораторной работы я приобрёл навыки написания программ с использованием циклов и обработкой аргументов командной строки в NASM.