

Отчёт по лабораторной работе №8

НКАбд-03-25

Кулаженкова Яна Сергеевна

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	11
5	Задание для самостоятельной работы	14
6	Выводы	15

Список иллюстраций

4.1	Переход в каталог лабораторной работы №8	8
4.2	Содержимое файла lab8-1	9
4.3	Продемонстрируем работу файла lab8-1	9
4.4	Изменим код	10
4.5	Работа измененного кода	10
4.6	Добавление push и pop	11
4.7	Выполнение нового кода	11
4.8	Содержимое файла lab8-2	12
4.9	Работа кода файла lab8-2	12
4.10	Программа в файле lab8-3	13
4.11	Результат работы программы lab8-3	13
5.1	Самостоятельная работа	14
5.2	Выполнение программы для поиска суммы	14

Список таблиц

1 Цель работы

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

2 Задание

Освоить принципы организации циклов и обработки аргументов командной строки в языке ассемблера NASM.

3 Теоретическое введение

Работа с циклами и обработка аргументов командной строки являются важными аспектами программирования на низком уровне, особенно в языке ассемблера. В данной лабораторной работе основное внимание уделено двум ключевым вопросам:

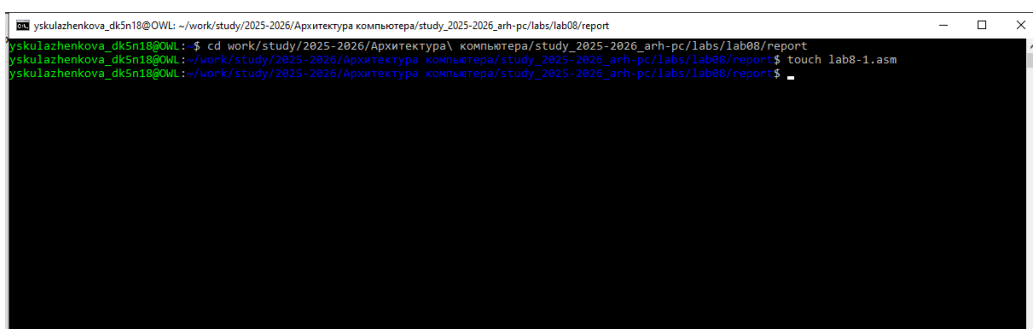
1. Организация циклов в ассемблере NASM.
2. Обработка аргументов командной строки.

Часто возникает потребность передавать приложениям некоторые данные прямо при запуске программы. Такие данные называются аргументами командной строки. Процессор сохраняет их в специальной области памяти — стеке. Чтобы обработать эти аргументы, необходимо уметь обращаться к данным в стеке и выделять необходимые элементы. Кроме того, важным элементом работы с циклами и аргументами командной строки является правильное использование стека, так как большинство команд, связанных с передачей аргументов и организацией циклов, активно используют этот механизм.

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

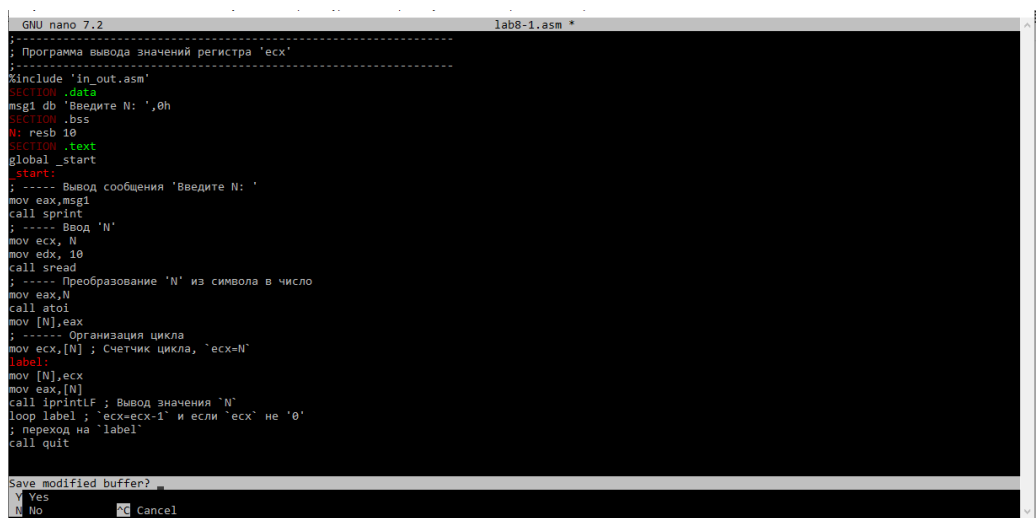
Перейдём в каталог для программ лабораторной работы (рис. 4.1).



```
yskulazhenkova_dk5n18@OWL: ~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report
yskulazhenkova_dk5n18@OWL:~$ cd work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ touch lab8-1.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$
```

Рисунок 4.1: Переход в каталог лабораторной работы №8

Изучим реализацию циклов в NASM с использованием инструкции `loop`. В качестве примера рассмотрим программу, которая выводит значение регистра `ecx` (рис. 4.2).

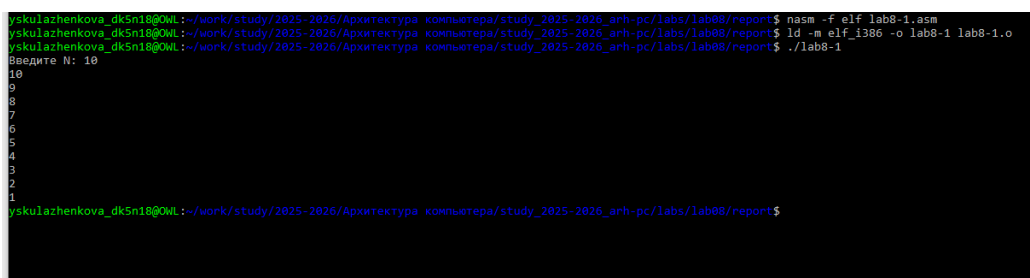


```
GNU nano 7.2 lab8-1.asm *
; Программа вывода значений регистра 'ecx'
;-----
include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov esi, 10
call read
; ----- 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 iprintf ; Вывод значения 'N'
loop label ; 'ecx-ecx-1' и если 'ecx' не '0'
; переход на 'label'
call quit

Save modified buffer?
Y Yes
N No Cancel
```

Рисунок 4.2: Содержимое файла lab8-1

Создадим исполняемый файл и проверим его работу (рис. 4.3).



```
yskulazhenkova_dk5n10@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab8/report$ nasm -f elf lab8-1.asm
yskulazhenkova_dk5n10@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab8/report$ ld -m elf_i386 -o lab8-1 lab8-1.o
yskulazhenkova_dk5n10@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab8/report$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
yskulazhenkova_dk5n10@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab8/report$
```

Рисунок 4.3: Продемонстрируем работу файла lab8-1

Заметим, что использование регистра ecx в теле цикла loop приводит к некорректной работе программы. Изменим текст программы (рис. 4.4).

```

GNU nano 7.2                                lab8-1.asm *
;-----
; 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:
sub ecx,1 ; 'ecx=ecx-1'
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
call quit

```

Рисунок 4.4: Изменим код

Продемонстрируем работу файла (рис. 4.5).

```

yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ nano lab8-1.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ nasm -f elf lab8-1.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ ld -m elf_i386 -o lab8-1 lab8-1.o
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ ./lab8-1
Введите N: 10
9
7
5
3
1
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$

```

Рисунок 4.5: Работа измененного кода

Теперь напишем программу для использования стека. Для этого добавим в текст программы команды push и pop (рис. 4.6).

```

GNU nano 7.2                                lab8-1.asm *
;-----
; 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:
push ecx ; добавление значения ecx в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintf
pop ecx ; извлечение значения ecx из стека
loop label
call quit

```

Рисунок 4.6: Добавление push и pop

Проверим работу кода (рис. 4.7).

```

yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ nano lab8-1.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ nasm -f elf lab8-1.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ ld -m elf_i386 -o lab8-1 lab8-1.o
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ _

```

Рисунок 4.7: Выполнение нового кода

4.2 Обработка аргументов командной строки

Рассмотрим общие принципы взаимодействия со стеком. Для этого реализуем программу, которая выводит на экран аргументы командной строки. Создадим файл lab8-2.asm и введем в него текст программы из листинга 8.2 (рис. 4.8).

```

GNU nano 7.2                                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)
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 printf ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit_

```

Рисунок 4.8: Содержимое файла lab8-2

Запустим программу (рис. 4.9).

```

yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ _

```

Рисунок 4.9: Работа кода файла lab8-2

В качестве ещё одного примера напишем код программы, которая выводит сумму аргументов. Для этого в файле lab8-3.asm запишем следующее (рис. 4.10).

```

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

```

Рисунок 4.10: Программа в файле lab8-3

Создадим исполняемый файл и запустим его, указав аргументы (рис. 4.11).

```

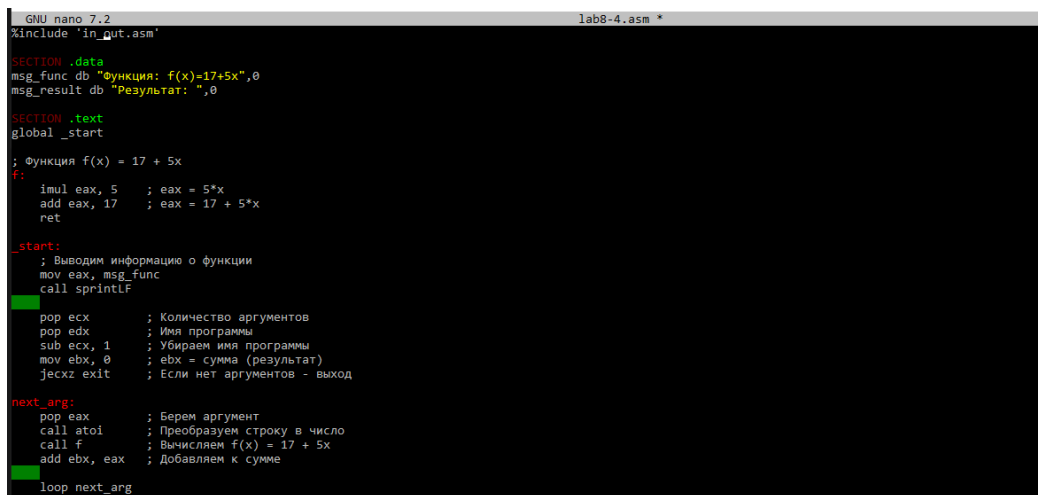
yskulazhenkova_dk5n18@OWL: ~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ ./lab8-2 аргумент1 аргумент 2 'аргуме
аргумент1
аргумент
2
аргумент 3
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ touch lab8-3.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ nano lab8-3.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ nasm -f elf lab8-3.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ ld -m elf_i386 -o lab8-3 lab8-3.o
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$ ./lab8-3 12 13 7 10 5
Результат: 47
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab08/report$

```

Рисунок 4.11: Результат работы программы lab8-3

5 Задание для самостоятельной работы

По условию задания нам необходимо написать программу, которая находит сумму значений функции (рис. 5.1).



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

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

SECTION .text
global _start

; Функция f(x) = 17 + 5x
f:
    imul eax, 5    ; eax = 5*x
    add eax, 17    ; eax = 17 + 5*x
    ret

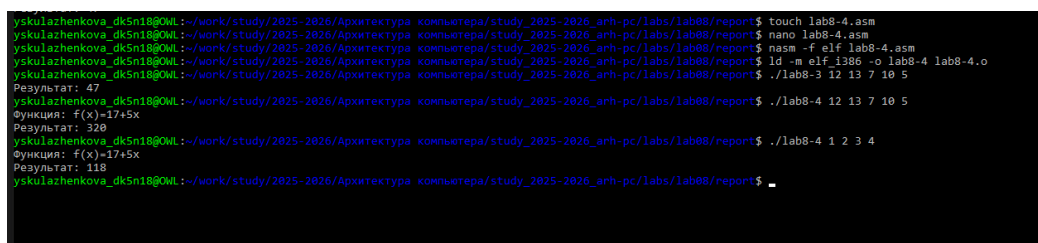
_start:
; Выводим информацию о функции
mov eax, msg_func
call sprintf

    pop ecx        ; Количество аргументов
    pop edx        ; Имя программы
    sub ecx, 1     ; Убираем имя программы
    mov ebx, 0     ; ebx = сумма (результат)
    jecxz exit     ; Если нет аргументов - выход

next_arg:
    pop eax        ; Берем аргумент
    call atoi      ; Преобразуем строку в число
    call f         ; Вычисляем f(x) = 17 + 5x
    add ebx, eax   ; Добавляем к сумме
    loop next_arg
```

Рисунок 5.1: Самостоятельная работа

Проверим правильность работы программы (рис. 5.2).



```
yksulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab8/report$ touch lab8-4.asm
yksulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab8/report$ nano lab8-4.asm
yksulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab8/report$ nasm -f elf lab8-4.asm
yksulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab8/report$ ld -m elf_i386 -o lab8-4 lab8-4.o
yksulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab8/report$ ./lab8-3 12 13 7 10 5
Результат: 47
yksulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab8/report$ ./lab8-4 12 13 7 10 5
Функция: f(x)=17+5x
Результат: 320
yksulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab8/report$ ./lab8-4 1 2 3 4
Функция: f(x)=17+5x
Результат: 118
yksulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab8/report$
```

Рисунок 5.2: Выполнение программы для поиска суммы

6 Выводы

В ходе выполнения лабораторной работы были получены и закреплены важные навыки программирования на языке ассемблера NASM. Был освоен механизм организации циклов, также были изучены ключевые инструкции, такие как `loop`, `jmp`, `push` и `pop`.

Полученные знания позволят проектировать и разрабатывать эффективные и надежные программы на ассемблере, обеспечивающие быструю и гибкую обработку больших объемов данных.