

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ЛАБОРАТОРНАЯ РАБОТА № 8

дисциплина: *Архитектура компьютера*

Студент: Мизинов М.Г.

Группа: НКАбд-04-25

№ ст. билета: 1032253540

МОСКВА

2025 г.

СОДЕРЖАНИЕ

Список иллюстраций.....	3
Список таблиц.....	4
Основная часть.....	5
1. Цель работы	5
2. Теоретическое введение	5
3. Задание	5
4. Выполнение лабораторной работы.....	6
4.1 Реализация циклов в NASM	6
4.2 Обработка аргументов командной строки	8
5. Задание для самостоятельной работы	11
Выводы	13
Список литературы.....	14

Список иллюстраций

Рисунок 1 – Создаю начальные условия	6
Рисунок 2 – Файл lab8-1.asm	6
Рисунок 3 – Выполнение lab8-1.asm	6
Рисунок 4 – Изменённый lab8-1.asm	7
Рисунок 5 – Запуск изменённого lab8-1.asm	7
Рисунок 6 – Дважды изменённый lab8-1.asm	8
Рисунок 7 – Запуск дважды изменённого lab8-1.asm	8
Рисунок 8 – Создание lab8-2.asm	9
Рисунок 9 – Запуск lab8-2.asm	9
Рисунок 10 – Создание lab8-3.asm	10
Рисунок 11 – Запуск lab8-3.asm	10
Рисунок 12 – Изменённый lab8-3.asm.....	11
Рисунок 13 – Запуск изменённого lab8-3.asm	11
Рисунок 14 – Создание uniq.asm	12
Рисунок 15 – Выполнение uniq.asm.....	12

Список таблиц

Основная часть

1. Цель работы

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

2. Теоретическое введение

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды.

Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров.

3. Задание

На основе методических указаний провести ознакомительную работу с программированием цикла и обработкой аргументов командной строки.

4. Выполнение лабораторной работы

4.1 Реализация циклов в NASM

Создал папку lab8 и файл asm (рис. 1).

```
mgmizinov@mint:~$ cd ~/Mizinov-study_2025-2026_arh-pc/labs/  
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs$ mkdir lab8  
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs$ cd lab8  
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ touch lab8-1.asm  
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$
```

Рис. 1: Создаю начальные условия

Код файла lab8-1.asm (рис. 2).

```
%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|
```

Рис. 2: Файл lab8-1.asm

Запуск программы lab8-1.asm (рис. 3).

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs$ cd lab8  
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ touch lab8-1.asm  
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ nasm -f elf lab8-1.asm  
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ ld -m elf_i386 -o lab8-1 lab8-1.o  
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ ./lab8-1  
Введите N: 6  
6  
5  
4  
3  
2  
1  
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$
```

Рис. 3: Выполнение lab8-1.asm

Изменим код программы (рис. 4).

```
%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
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
```

Рис. 4: Изменённый lab8-1.asm

А затем запустим изменённый файл (рис. 5).

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ nasm -f elf lab8-1.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ ld -m elf_i386 -o lab8-1 lab8-1.o
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ ./lab8-1
Введите N: 6
5
3
1
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ █
```

Рис. 5: Запуск изменённого lab8-1.asm

Видно, что теперь программа пропускает каждое второе число.

Теперь снова изменим файл добавив строки с push и pop(рис. 6).

```
%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 iprintfLF
pop ecx ; извлечение значения ecx из стека
loop label

; переход на `label`
call quit
```

Рис. 6: Дважды изменённый lab8-1.asm

Запуск изменённой дважды программы lab8-1.asm (рис. 7).

```
mgininov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ nasm -f elf lab8-1.asm
mgininov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ ld -m elf_i386 -o lab8-1 lab8-1.o
mgininov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ ./lab8-1
Введите N: 6
5
4
3
2
1
0
```

Рис. 7: Запуск дважды изменённого lab8-1.asm

Теперь выводятся цифры начиная с N-1.

4.2 Обработка аргументов командной строки

Создадим программу lab8-2.asm с кодом из листинга (рис. 8).

```
%include 'in_out.asm'
SECTION .text
global _start
_start:
    pop ecx ; Извлекаем из стека в `есх` количество
    ; аргументов (первое значение в стеке)
    pop edx ; Извлекаем из стека в `edx` имя программы
    ; (второе значение в стеке)
    sub ecx, 1 ; Уменьшаем `есх` 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|
```

Рис. 8: Создание lab8-2.asm

А затем запустим её (рис. 9).

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ nasm -f elf lab8-2.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ ld -m elf_i386 -o lab8-2 lab8-2.o
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ ./lab8-2
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ ./lab8-2 1 2 3
1
2
3
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ ./lab8-2 a1 a 2 'a3'
a1
a
2
a3
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ █
```

Рис. 9: Запуск lab8-2.asm

Создадим программу lab8-3.asm с кодом из листинга (рис. 10).

```
%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: Создание lab8-3.asm

А затем запустим её (рис. 11).

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ nasm -f elf lab8-3.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ ld -m elf_i386 -o lab8-3 lab8-3.o
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ ./lab8-3 12 13 7 10 5
Результат: 47
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$
```

Рис. 11: Запуск lab8-3.asm

Изменим код программы заменив сложение на умножение (рис. 12).

```
%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,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
mul esi
mov esi,eax
; 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. 12: Изменённый lab8-3.asm

А затем запустим изменённый файл (рис. 13).

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ nasm -f elf lab8-3.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ ld -m elf_i386 -o lab8-3 lab8-3.o
Результат: 60
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ ./lab8-3 12 5
```

Рис. 13: Запуск изменённого lab8-3.asm

5. Задание для самостоятельной работы

У меня вариант номер 1, как я выяснил во время бй лабораторной работы.

А значит использую следующие данные:

Выражение для $f(x) = 2x + 15$

Реализуем файл uniq.asm для индивидуального задания (рис. 14).

```
%include 'in_out.asm'
SECTION .data
msg_func db "Функция: f(x) = 2x + 15", 0
msg_result db "Результат: ", 0
SECTION .text
GLOBAL _start
_start:
    mov eax, msg_func
    call sprintLF
    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 0
next:
    cmp ecx, 0h
    jz _end
    pop eax
    call atoi
    mov ebx, 2
    mul ebx
    add eax, 15
    add esi, eax
    loop next
_end:
    mov eax, msg_result
    call sprint
    mov eax, esi
    call iprintLF
    call quit
```

Рис. 14: Создание uniq.asm

Запуск программы code1.asm и проверка табличных значений (рис. 14).

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ nasm -f elf uniq.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ ld -m elf_i386 -o uniq uniq.o
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ ./uniq 1 2
Функция: f(x) = 2x + 15
Результат: 36
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab8$ █
```

Рис. 15: Выполнение uniq.asm

Проверим вручную

$$(2*(1)+15) + (2*(2)+15) = 17 + 19 = 36$$

Программа работает исправно.

Ссылка на github: https://github.com/MihailMizinov/Mizinov-study_2025-2026_arh-pc

Выводы

При выполнении данной лабораторной работы я приобрёл практические навыки использования циклов и обработки аргументов командной строки.

Список литературы

1) Лабораторная работа №8.

[https://esystem.rudn.ru/pluginfile.php/2089095/mod_resource/content/0/%D0%9B%D0%
%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%
%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E
2%84%968.%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%
D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%
B5%20%D1%86%D0%B8%D0%BA%D0%BB%D0%B0.%20%D0%9E%D0%B1%D1%
80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%20%D0%B0%D1%80
%D0%B3%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%
B2%20%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%
B4%D0%BD%D0%BE%D0%
B9%20%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%
B8..pdf](https://esystem.rudn.ru/pluginfile.php/2089095/mod_resource/content/0/%D0%9B%D0%
%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%
%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E
2%84%968.%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%
D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%
B5%20%D1%86%D0%B8%D0%BA%D0%BB%D0%B0.%20%D0%9E%D0%B1%D1%
80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%20%D0%B0%D1%80
%D0%B3%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%
B2%20%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%
B4%D0%BD%D0%BE%D0%
B9%20%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%
B8..pdf)

2) Википедия. <https://en.wikipedia.org/wiki/GitHub>