

Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Машков Илья Евгеньевич

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Реализация циклов в NASM	6
3.2	Обработка аргументов командной строки	11
3.3	Выполнение задания для самостоятельной работы	15
4	Выводы	19
5	Список литературы	20

Список иллюстраций

3.1	Директория lab08.	6
3.2	Код программы lab8-1.asm.	7
3.3	Результат выполнения программы.	7
3.4	Изменённый код программы lab8-1.asm.	8
3.5	Изменённый код программы lab8-1.asm.	10
3.6	Результат выполнения программы.	11
3.7	Создание файла lab8-2.asm.	11
3.8	Код программы lab8-2.asm.	12
3.9	Результат выполнения программы.	12
3.10	Результат второго выполнения программы.	12
3.11	Создание файла lab8-3.asm.	13
3.12	Код программы lab8-3.asm.	13
3.13	Результат выполнения программы.	14
3.14	Изменённый код программы lab8-3.asm.	14
3.15	Результат второго выполнения программы.	15
3.16	Создание файла lab8-4.asm.	15
3.17	Код программы lab8-4.asm.	16
3.18	Результат выполнения программы.	18

1 Цель работы

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

2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки
3. Выполнение задания для самостоятельной работы

3 Выполнение лабораторной работы

3.1 Реализация циклов в NASM

Для начала в папке локального репозитория я создаю директорию **lab08** для дальнейшей работы в ней, а также перехожу в созданный мной каталог и создаю файл **lab8-1.asm** с помощью команды **'touch'**, а также копирую файл **in_out.asm** (рис. [3.1]).

```
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ mkdir lab08
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ ls
CHANGELOG.md  config  COURSE  lab04  lab05  lab06  lab07  lab08  labs  lankdir  LICENSE  Makefile  prepare  presentation
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ cd lab08
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ touch lab8-1.asm
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ls
lab8-1.asm
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$
```

Рис. 3.1: Директория lab08.

Затем я ввожу код в .asm файл (Рис. [3.2]).

```

#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. 3.2: Код программы lab8-1.asm.

Создаю исполняемый файл и запускаю программу. В выводе программы я получаю 5 переходов от 5-ки и до единицы.(Рис. [3.3]).

```

ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ nasm -f elf -l lab8-1.lst lab8-1.asm
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -n elf_i386 -o lab8-1 lab8-1.o
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ 

```

Рис. 3.3: Результат выполнения программы.

Теперь я меняю кусок программы под меткой **label** (Рис. [3.4]):

```

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

```

```

#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
call quit

```

Рис. 3.4: Изменённый код программы lab8-1.asm.

После создания исполняемого файла и его запуска, при введении значения 'N

= 5', программа начала выполнять цикл от **4 230 000 000** до единицы. Понятное дело, что я не смог дождаться конца работы программы.

После этого я снова меняю код программы (Рис. [3.5]):

```
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
```

```

#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

```

Рис. 3.5: Изменённый код программы lab8-1.asm.

Создаю исполняемый файл и запускаю его (Рис. [3.6]). В выводе получаю пять переходов от 4-ёх до нуля.

```

iemashkov@iemashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ nasm -f elf -l lab8-1.lst lab8-1.asm
iemashkov@iemashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
iemashkov@iemashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-1
Введите N: 5
4
3
2
1
0
iemashkov@iemashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$

```

Рис. 3.6: Результат выполнения программы.

3.2 Обработка аргументов командной строки

Создаю файл **lab8-2.asm** (Рис. [3.7]).

```

iemashkov@iemashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ touch lab8-2.asm
iemashkov@iemashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ls
in_out.asm lab8-1 lab8-1.asm lab8-1.lst lab8-1.o lab8-2.asm
iemashkov@iemashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$

```

Рис. 3.7: Создание файла lab8-2.asm.

Ввожу код программы (Рис. [3.8]).

```

#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

```

Рис. 3.8: Код программы lab8-2.asm.

Создаю исполняемый файл и запускаю программу (Рис. [3.9]). В выводе получаю два обработанных аргумента.

```

tenashkov@tenashkov: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ nasm -f elf -l lab8-2.lst lab8-2.asm
tenashkov@tenashkov: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
tenashkov@tenashkov: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-2 аргумент 1 аргумент 2 "аргумент 3"
аргумент
1
аргумент
2
аргумент 3

```

Рис. 3.9: Результат выполнения программы.

Теперь я убрал кавычки с **аргумент 3** и, в итоге, получил три обработанных аргумента (Рис. [3.10]).

```

tenashkov@tenashkov: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-2 аргумент 1 аргумент 2 аргумент 3
аргумент
1
аргумент
2
аргумент
3

```

Рис. 3.10: Результат второго выполнения программы.

Создаю файл **lab8-3.asm** (Рис. [3.11]).

```
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab8$ touch lab8-3.asm
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab8$ ls
in_out.asm lab8-1 lab8-1.asm lab8-1.lst lab8-1.o lab8-2 lab8-2.asm lab8-2.lst lab8-2.o lab8-3.asm
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab8$
```

Рис. 3.11: Создание файла lab8-3.asm.

Ввожу в него код программы (Рис. [3.12]).

```
%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. 3.12: Код программы lab8-3.asm.

Создаю исполняемый файл и запускаю программу. При введении значений '12 13 7 10 5' я получаю их сумму, т.е. '47' (Рис. [3.13]).

```

lenashkov@lenashkov:~/work/study/2023-2024/Архитектура_компьютера/arch-pc/lab08$ nasm -f elf -l lab8-3.lst lab8-3.asm
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура_компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура_компьютера/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура_компьютера/arch-pc/lab08$ 

```

Рис. 3.13: Результат выполнения программы.

Теперь я меняю код программы так, чтобы она выводила произведение значений, введённых с клавиатуры. Для этого необходимо поменять '0' на '1' в строчке `'mov esi, 0'`, чтобы в промежуточных произведениях не происходило умножение на 0 (Рис. [3.14]).

```

#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
por ecx ; Извлекаем из стека в `ecx` количество
        ; аргументов (первое значение в стеке)
por 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, 1 ; Используем `esi` для хранения
        ; промежуточных произведений
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
        ; (переход на метку `_end`)
por eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
imul esi, eax ; добавляем к промежуточному произведению
        ; 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 произведение в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы

```

Рис. 3.14: Изменённый код программы lab8-3.asm.

Создаю исполняемый файл и запускаю программу. При введении значений 5

6 я получаю их произведение, т.е. **30** (Рис. [3.15]).

```
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ nasm -f elf -l lab8-3.lst lab8-3.asm
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-3 5 6
Результат: 30
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$
```

Рис. 3.15: Результат второго выполнения программы.

3.3 Выполнение задания для самостоятельной работы

1. Создаю файл **lab8-4.asm** (Рис. [3.16]). Так как код схож с тем, который был в **lab8-3.asm**, я использовал команду **'cp'**, а не **'touch'**.

```
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ cp lab8-3.asm lab8-4.asm
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ls
in_out.asm lab8-1.asm lab8-1.o lab8-2.asm lab8-2.o lab8-3.asm lab8-3.o
lab8-1 lab8-1.lst lab8-2 lab8-2.lst lab8-3 lab8-3.lst lab8-4.asm
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$
```

Рис. 3.16: Создание файла lab8-4.asm.

Теперь ввожу код программы, которая будет находить сумму значений функции (Рис. [3.17]). У меня пятый вариант, а это значит, что моя функция равна **'4x + 3'**.

```

#include 'in_out.asm'
SECTION .data
msg db "Функция: f(x) = 4*x + 3",0
msg1 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 произведений

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
mov ebx, 4
mul ebx ; "eax = eax*ebx = x*4"
add eax, 3 ; "eax = eax + 3 = 4*x + 3"
add esi, eax ; добавляем к промежуточному произведению
        ; след. аргумент `esi=esi*eax`
loop next ; переход к обработке следующего аргумента

_end:
mov eax, msg ; вывод сообщения "Функция: f(x) = 4*x + 3 "
call printf
mov eax, msg1 ; вывод сообщения "Результат: "
call printf
mov eax, esi ; записываем произведение в регистр `eax`
call printf ; печать результата
call quit ; завершение программы

```

Рис. 3.17: Код программы lab8-4.asm.

Сам код:

```

#include 'in_out.asm'

SECTION .data
msg db "Функция: f(x) = 4*x + 3",0

```



```

msg1 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 произведений

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
mov ebx, 4
mul ebx ; "eax = eax*ebx = x*4"
add eax, 3 ; "eax = eax + 3 = 4*x + 3"
add esi, eax ; добавляем к промежуточному произведению
        ; след. аргумент `esi=esi*eax`
loop next ; переход к обработке следующего аргумента

_end:
mov eax, msg ; вывод сообщения "Функция: f(x) = 4*x + 3 "

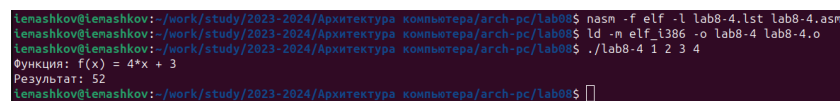
```

```

call sprintLF
mov eax, msg1 ; вывод сообщения "Результат: "
call sprint
mov eax, esi ; записываем произведение в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы

```

Теперь я создаю исполняемый файл и запускаю программу (Рис. [3.18]). При введении значений **‘1 2 3 4’** я получаю в выводе **‘52’**, что и является правильным ответом.



```

lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ nasm -f elf -l lab8-4.lst lab8-4.asm
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-4 1 2 3 4
Функция: F(x) = 4*x + 3
Результат: 52
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ 

```

Рис. 3.18: Результат выполнения программы.

4 Выводы

При выполнении данной лабораторной работы я освоил работу с циклами в NASM, а также научился писать программы, которые обрабатывают аргументы командной строки.

5 Список литературы

Архитектура ЭВМ