

# **Отчёт по лабораторной работе №8**

**Дисциплина: архитектура компьютера**

Аветисян Алина Эдуардовна

# Содержание

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Цель работы</b>                        | <b>5</b>  |
| <b>2</b> | <b>Выполнение лабораторной работы</b>     | <b>6</b>  |
| <b>3</b> | <b>Задание для самостоятельной работы</b> | <b>13</b> |
| <b>4</b> | <b>Выводы</b>                             | <b>15</b> |

## Список иллюстраций

|      |   |    |
|------|---|----|
| 2.1  | Создание каталога и файла. . . . .                                    | 6  |
| 2.2  | Открываю файл lab8-1.asm с помощью текстового редактора nano. . . . . | 6  |
| 2.3  | Исполнение файла. . . . .   | 7  |
| 2.4  | Изменение программы. . . . .  | 7  |
| 2.5  | Исполнение файла. . . . .   | 8  |
| 2.6  | Изменение программы. . . . .  | 8  |
| 2.7  | Исполнение файла. . . . .   | 9  |
| 2.8  | Создание файла. . . . .   | 9  |
| 2.9  | Ввод программы. . . . .   | 10 |
| 2.10 | Исполнение файла. . . . .   | 10 |
| 2.11 | Создание файла. . . . .   | 10 |
| 2.12 | Ввод программы. . . . .   | 11 |
| 2.13 | Исполнение файла. . . . .   | 11 |
| 2.14 | Изменение программы. . . . .  | 12 |
| 2.15 | Исполнение файла. . . . .   | 12 |
| 3.1  | Создание файла. . . . .   | 13 |
| 3.2  | Написание программы. . . . .  | 14 |
| 3.3  | Исполнение файла. . . . .   | 14 |

## Список таблиц

# 1 Цель работы

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

## 2 Выполнение лабораторной работы

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

```
aeavetisyan@dk8n64 ~ $ mkdir ~/work/arch-pc/lab08
aeavetisyan@dk8n64 ~ $ cd ~/work/arch-pc/lab08
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ touch lab8-1.asm
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $
```

Рис. 2.1: Создание каталога и файла.

Открываю созданный файл lab8-1.asm, вставляю в него программу вывода значений регистра ecx.

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/ae/aeavetisyan/work/arch-pc/lab08/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:
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
```

Рис. 2.2: Открываю файл lab8-1.asm с помощью текстового редактора nano.

Создаю исполняемый файл и проверяю его работу. Беру значение N=12.

```
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 12
12
11
10
9
8
7
6
5
4
3
2
1
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $
```

Рис. 2.3: Исполнение файла.

Изменяю текст программы в файле lab8-1.asm.

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/a/e/aeavetisyan/work/arch-pc/lab08/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 sprintf

; ----- 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 sprintf
loop label

call quit
```

Рис. 2.4: Изменение программы.

Создаю исполняемый файл и проверяю его работу.

```
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 12
11
9
7
5
3
1
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $
```

Рис. 2.5: Исполнение файла.

Регистр `ecx` уменьшается на 2 в цикле. Число проходов не соответствует значению `N` введенному с клавиатуры.

Вношу изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop`.

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/a/e/aeavetisyan/work/arch-pc/lab08/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:
push ecx ; добавление значения ecx в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call sprintf
pop ecx ; извлечение значения ecx из стека

loop label

call quit
```

Рис. 2.6: Изменение программы.

Создаю исполняемый файл и проверяю его работу.



```
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 12
11
10
9
8
7
6
5
4
3
2
1
0
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $
```

Рис. 2.7: Исполнение файла.

В данном случае число проходов цикла соответствует значению ☒ введенному с клавиатуры(12=12).

Создаю файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввожу в него текст программы из листинга 8.2(программу выводящую на экран аргументы командной строки).

```
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ touch lab8-2.asm
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $
```

Рис. 2.8: Создание файла.

```

/afs/.dk.sci.pfu.edu.ru/home/a/e/aeavetisyan/work/arch-pc/lab08/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

```

Рис. 2.9: Ввод программы.

Создаю исполняемый файл и запускаю его, указав аргументы.

```

aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $

```

Рис. 2.10: Исполнение файла.

Аргументов было обработано программой : 4.

Создаю файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 и ввожу в него текст программы из листинга 8.3(программу вычисления суммы аргументов командной строки).

```

aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ touch lab8-3.asm
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $

```

Рис. 2.11: Создание файла.

```

/afs/.dk.sci.pfu.edu.ru/home/a/e/aeavetisyan/work/arch-pc/lab08/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 iprintLF    ; печать результата
call quit        ; завершение программы

```

Рис. 2.12: Ввод программы.

Создаю исполняемый файл и запускаю его, указав аргументы.

```

aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-3 12 15 8 11 6
Результат: 52
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $

```

Рис. 2.13: Исполнение файла.

Изменяю текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.

```

/afs/.dk.sci.pfu.edu.ru/home/a/e/aeavetisyan/work/arch-pc/lab08/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, 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')
pop eax          ; иначе извлекаем следующий аргумент из стека
call atoi        ; преобразуем символ в число
mul esi          ; добавляем к промежуточному произведению
                 ; след. аргумент 'esi=esi*eax'
mov 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. 2.14: Изменение программы.

Создаю исполняемый файл и проверяю его работу.

```

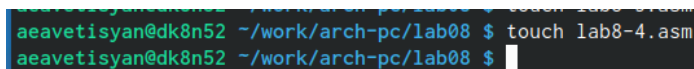
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 54600
aeavetisyan@dk8n64 ~/work/arch-pc/lab08 $ █

```

Рис. 2.15: Исполнение файла.

### 3 Задание для самостоятельной работы

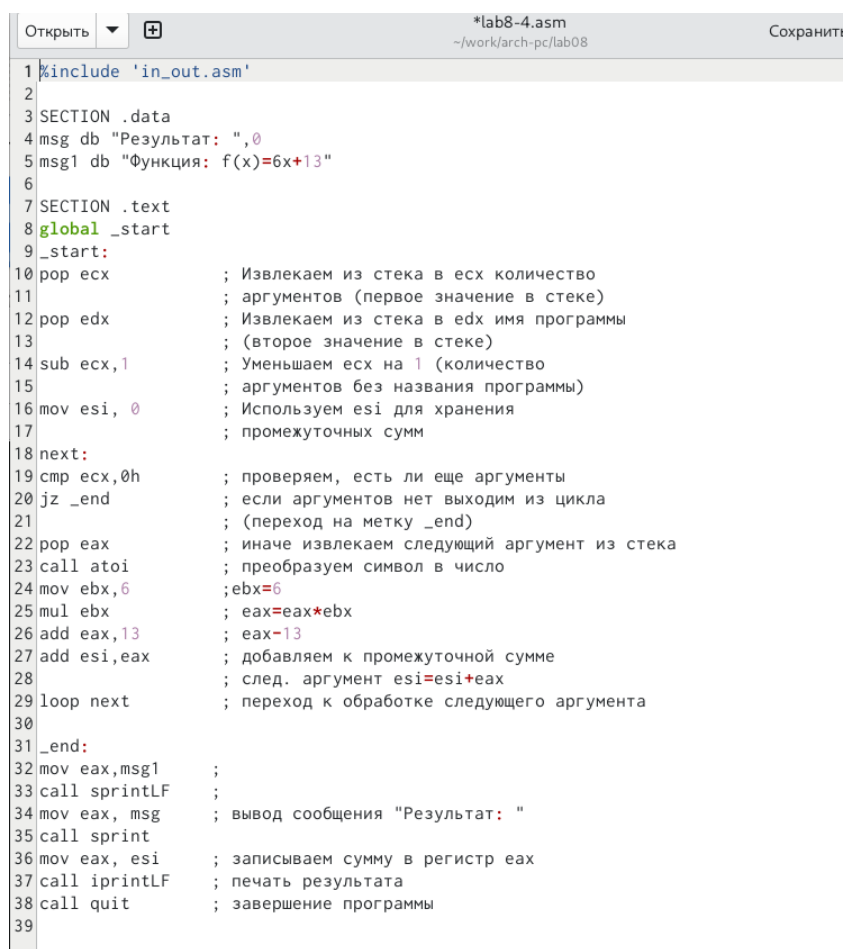
Создаю файл lab8-4.asm с помощью утилиты touch.

A terminal window with a dark background. The prompt is 'aeavetisyan@dk8n52 ~/work/arch-pc/lab08 \$'. The user enters 'touch lab8-4.asm'. The prompt changes to 'aeavetisyan@dk8n52 ~/work/arch-pc/lab08 \$' followed by a cursor.

```
aeavetisyan@dk8n52 ~/work/arch-pc/lab08 $ touch lab8-4.asm
aeavetisyan@dk8n52 ~/work/arch-pc/lab08 $
```

Рис. 3.1: Создание файла.

Пишу программу, которая находит сумму значений функции  $f(x)=6x + 13$  (Вариант 15) для  $x = x_1, x_2, \dots, x_n$ , т.е. программа должна выводить значение  $x(x_1) + x(x_2) + \dots + x(x_n)$ . Значения  $x_n$  передаются как аргументы.



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg db "Результат: ",0
5 msg1 db "Функция: f(x)=6x+13"
6
7 SECTION .text
8 global _start
9 _start:
10 pop ecx          ; Извлекаем из стека в ecx количество
11                 ; аргументов (первое значение в стеке)
12 pop edx          ; Извлекаем из стека в edx имя программы
13                 ; (второе значение в стеке)
14 sub ecx,1        ; Уменьшаем ecx на 1 (количество
15                 ; аргументов без названия программы)
16 mov esi, 0       ; Используем esi для хранения
17                 ; промежуточных сумм
18 next:
19 cmp ecx,0h       ; проверяем, есть ли еще аргументы
20 jz _end          ; если аргументов нет выходим из цикла
21                 ; (переход на метку _end)
22 pop eax          ; иначе извлекаем следующий аргумент из стека
23 call atoi        ; преобразуем символ в число
24 mov ebx,6        ; ebx=6
25 mul ebx          ; eax=eax*ebx
26 add eax,13       ; eax=13
27 add esi,eax      ; добавляем к промежуточной сумме
28                 ; след. аргумент esi=esi+eax
29 loop next        ; переход к обработке следующего аргумента
30
31 _end:
32 mov eax,msg1     ;
33 call sprintf     ;
34 mov eax, msg     ; вывод сообщения "Результат: "
35 call sprintf     ;
36 mov eax, esi     ; записываем сумму в регистр eax
37 call iprintLF    ; печать результата
38 call quit        ; завершение программы
39
```

Рис. 3.2: Написание программы.

Создаю исполняемый файл и проверяю его работу.



```
aeavetisyan@dk2n24 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
aeavetisyan@dk2n24 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
aeavetisyan@dk2n24 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3 4
Функция: f(x)=6x+13
Результат: 112
aeavetisyan@dk2n24 ~/work/arch-pc/lab08 $
```

Рис. 3.3: Исполнение файла.

## 4 Выводы

При выполнении данной лабораторной работы я приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки.