

# **Отчет по лабораторной работе №9**

*дисциплина: Архитектура компьютера*

Галацан Николай, НПИбд-01-22

# Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Выполнение заданий для самостоятельной работы	11
4	Выводы	14

## Список иллюстраций

2.1	Редактирование файла lab9-1.asm . . . . .	5
2.2	Запуск исполняемого файла lab 9-1 . . . . .	6
2.3	Запуск измененного исполняемого файла lab 9-1 . . . . .	6
2.4	Запуск исполняемого файла lab 9-1 после 3-го изменения . . . . .	7
2.5	Редактирование файла lab 9-2.asm . . . . .	8
2.6	Запуск исполняемого файла lab 9-2 . . . . .	8
2.7	Запуск исполняемого файла lab9-3 . . . . .	9
2.8	Изменение текста программы lab9-3.asm для вычисления произведе- дения . . . . .	9
2.9	Запуск измененного исполняемого файла lab9-3.asm . . . . .	10
3.1	Запуск исполняемого файла lab9-sam . . . . .	13

# 1 Цель работы

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

## 2 Выполнение лабораторной работы

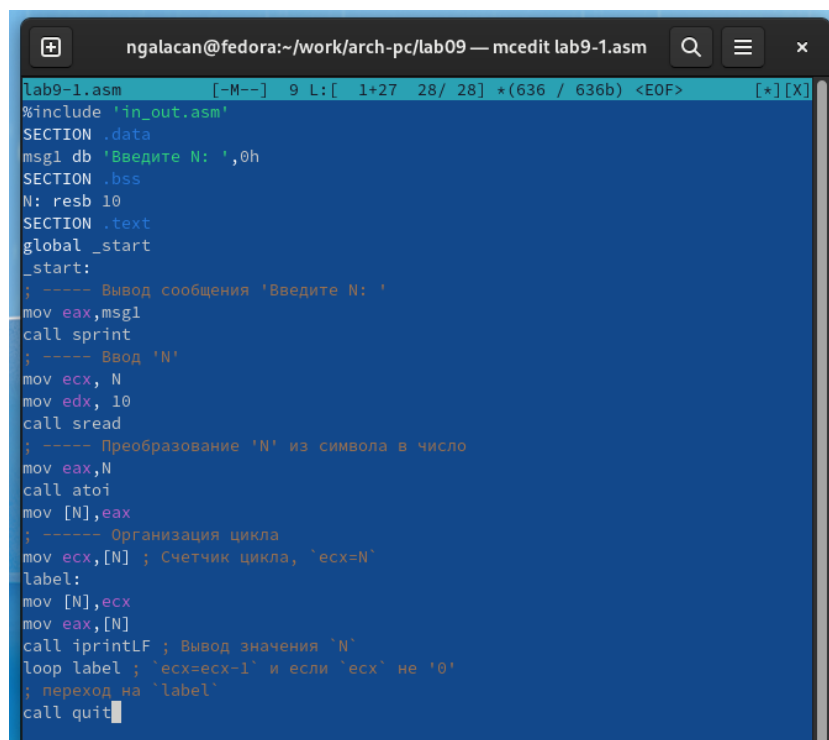
Ввожу команды для создания каталога лабораторной работы, перехожу в него, создаю файл lab9-1.asm

```
mkdir ~/work/arch-pc/lab09
```

```
cd ~/work/arch-pc/lab09
```

```
touch lab9-1.asm
```

Ввожу в файл lab9-1.asm текст программы из листинга 9.1, сохраняю файл. (рис. 2.1).



```
lab9-1.asm  [-M--]  9 L: [ 1+27  28/ 28] *(636 / 636b) <EOF>  [*] [X]
%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. 2.1: Редактирование файла lab9-1.asm

Создаю исполняемый файл и запускаю его (рис. 2.2).

```
[ngalacan@fedora lab09]$ nasm -f elf lab9-1.asm
[ngalacan@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[ngalacan@fedora lab09]$ ./lab9-1
Введите N: 5
5
4
3
2
1
[ngalacan@fedora lab09]$
```

Рис. 2.2: Запуск исполняемого файла lab 9-1

В результате выводится верное количество проходов по циклу.

Редактирую текст программы, добавив изменение значения регистра `eax` в цикле:

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

Создаю исполняемый файл и запускаю его (рис. 2.3).



```
ngalacan@fedora: ~/work/arch-pc/lab09 — ./lab9-1
4293854326
4293854324
4293854322
4293854320
4293854318
4293854316
4293854314
4293854312
4293854310
4293854308
4293854306
4293854304
4293854302
4293854300
4293854298
4293854296
4293854294
```

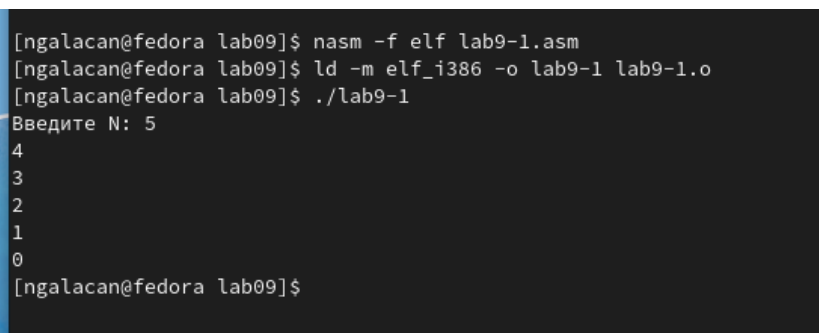
Рис. 2.3: Запуск измененного исполняемого файла lab 9-1

В результате исполнения программа заикливается. Число проходов цикла не соответствует значению, введенному с клавиатуры. Программа работает некорректно.

Редактирую текст программы, чтобы можно было использовать регистр `ecx` в цикле с помощью стека:

```
label:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx
loop label
```

Создаю исполняемый файл и запускаю его (рис. 2.4).

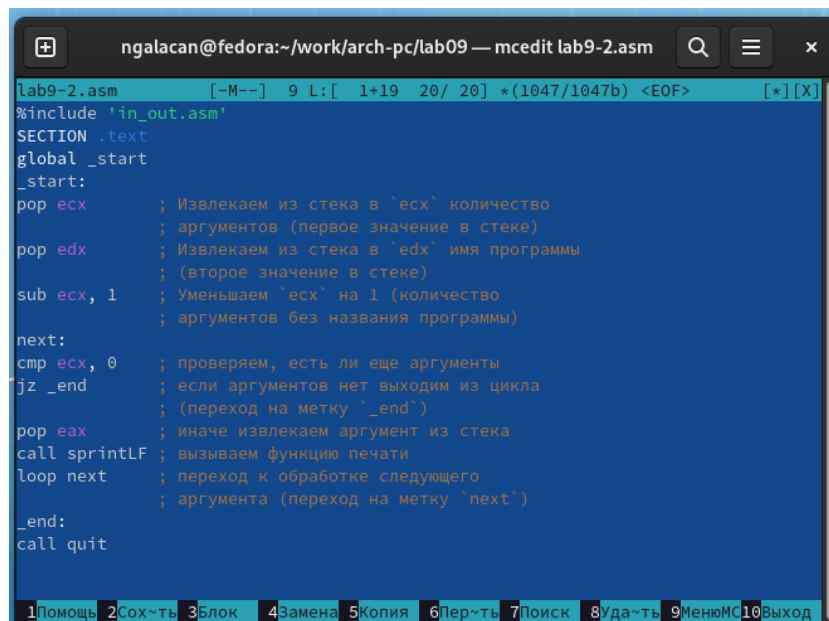


```
[ngalacan@fedora lab09]$ nasm -f elf lab9-1.asm
[ngalacan@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[ngalacan@fedora lab09]$ ./lab9-1
Введите N: 5
4
3
2
1
0
[ngalacan@fedora lab09]$
```

Рис. 2.4: Запуск исполняемого файла lab 9-1 после 3-го изменения

Число проходов цикла соответствует значению, введенному с клавиатуры.

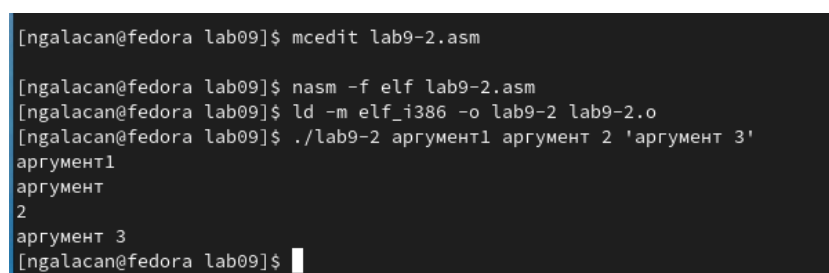
Создаю новый файл: `touch lab9-2.asm`. Ввожу в него текст программы из листинга 9.2, внимательно изучив (рис. 2.5).



```
lab9-2.asm      [-M--]  9 L: [ 1+19  20/ 20] *(1047/1047b) <EOF>      [*][X]
%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 sprintf     ; вызываем функцию печати
loop next        ; переход к обработке следующего
                 ; аргумента (переход на метку 'next')
_end:
call quit
```

Рис. 2.5: Редактирование файла lab 9-2.asm

Создаю исполняемый файл и запускаю его (рис. 2.6).



```
[ngalacan@fedora lab09]$ mcedit lab9-2.asm

[ngalacan@fedora lab09]$ nasm -f elf lab9-2.asm
[ngalacan@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[ngalacan@fedora lab09]$ ./lab9-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
2
аргумент 3
[ngalacan@fedora lab09]$
```

Рис. 2.6: Запуск исполняемого файла lab 9-2

Программой обработано 4 аргумента (аргументы отделяются друг от друга пробелом, из-за чего аргумент и 2 программа восприняла отдельно).

Создаю новый файл: `touch lab9-3.asm`. Ввожу в него текст программы из листинга 9.3. Создаю исполняемый файл и запускаю его, указав разные аргументы (рис. 2.7).



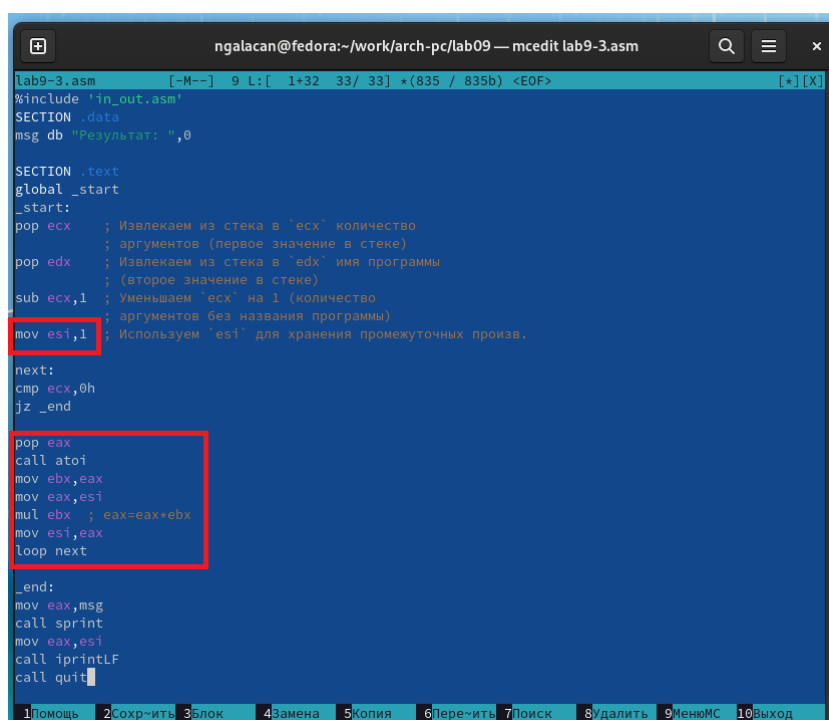
```

[ngalacan@fedora lab09]$ nasm -f elf lab9-3.asm
[ngalacan@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[ngalacan@fedora lab09]$ ./lab9-3 12 13 7 10 5
Результат: 47
[ngalacan@fedora lab09]$ ./lab9-3 15 6 21
Результат: 42
[ngalacan@fedora lab09]$

```

Рис. 2.7: Запуск исполняемого файла lab9-3

Редактирую текст программы lab9-3.asm для вычисления произведения аргументов командной строки (рис. 2.8).



```

lab9-3.asm  [-M--]  9 L:[ 1+32 33/ 33] *(835 / 835b) <EOF>  [*][X]
#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 произв.

next:
cmp ecx,0h
jz _end

pop eax
call atoi
mov ebx,eax
mov eax,esi
mul ebx      ; eax=eax*ebx
mov esi,eax
loop next

_end:
mov eax,msg
call sprint
mov eax,esi
call iprintLF
call quit

```

Рис. 2.8: Изменение текста программы lab9-3.asm для вычисления произведения

Значение esi меняю на 1 для корректного умножения. Аргумент из командной строки переносится в ebx, в eax помещается значение регистра esi. Выполняется умножение, и результат умножения переносится из eax в esi.

Создаю исполняемый файл и запускаю его, вводя разные аргументы (рис. 2.9).

```
[ngalacan@fedora lab09]$ nasm -f elf lab9-3.asm
[ngalacan@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[ngalacan@fedora lab09]$ ./lab9-3 2 4
Результат: 8
[ngalacan@fedora lab09]$ ./lab9-3 8 4 10 2
Результат: 640
[ngalacan@fedora lab09]$
```

Рис. 2.9: Запуск измененного исполняемого файла lab9-3.asm

Программа вычисляет верное произведение аргументов командной строки.

### 3 Выполнение заданий для самостоятельной работы

В лабораторной работе №7 был получен 4-ый вариант заданий.

1. Необходимо написать программу для нахождения суммы значений функции, где значения  $x$  передаются как аргументы. В соответствии вариантом выбрана функция  $f(x)=2(x-1)$ .

Создаю файл `lab9-sam.asm` и набираю текст программы. Написанная программа имеет следующий вид:

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fn db "Вариант 4. Функция: f(x)=2(x-1)."
```

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 результатов
```

```
next:
```

```
cmp ecx,0h
```

```
jz _end
```

```
pop eax ; eax=x
```

```
call atoi
```

```
sub eax,1 ; eax=x-1
```

```
mov ebx,2
```

```
mul ebx ; eax=(x-1)*2
```

```
add esi,eax ; esi=esi+eax
```

```
loop next
```

```
_end:
```

```
mov eax,fn
```

```
call sprintf
```

```
mov eax,msg
```

```
call sprintf
```

```
mov eax,esi
```

```
call sprintf
```

```
call quit
```

Создаю исполняемый файл и запускаю, проверяю работу на нескольких наборах  $x$  (рис. 3.1):

- 1 2 3
- 3 7 10 4 2
- 8 12
- 14 18 23 1 3 6

```
[ngalacan@fedora lab09]$ nasm -f elf lab9-sam.asm
[ngalacan@fedora lab09]$ ld -m elf_i386 -o lab9-sam lab9-sam.o
[ngalacan@fedora lab09]$ ./lab9-sam 1 2 3
Вариант 4. Функция:  $f(x)=2(x-1)$ .
Результат: 6
[ngalacan@fedora lab09]$ ./lab9-sam 3 7 10 4 2
Вариант 4. Функция:  $f(x)=2(x-1)$ .
Результат: 42
[ngalacan@fedora lab09]$ ./lab9-sam 8 12
Вариант 4. Функция:  $f(x)=2(x-1)$ .
Результат: 36
[ngalacan@fedora lab09]$ ./lab9-sam 14 18 23 1 3 6
Вариант 4. Функция:  $f(x)=2(x-1)$ .
Результат: 118
[ngalacan@fedora lab09]$
```

Рис. 3.1: Запуск исполняемого файла lab9-sam

Вычисляю аналитически сумму значений функции при каждом наборе. Программа выводит верный результат.

## 4 Выводы

Были приобретены навыки программирования циклов. Изучена и применена обработка аргументов командной строки.